Manual för PostGIS 3.7.0beta2

Innehåll

1 Introduktion **1**
1.1 What Is PostGIS? . . . 1
1.2 What Is in the PostGIS Distribution? . . . 1
1.3 The PostGIS Project Family . . . 2
1.4 How to Use This Manual . . . 2
1.5 Standards Support . . . 3
1.6 Project and Community . . . 3

2 Installation av PostGIS **4**
2.1 Kort version . . . 4
2.2 Kompilera och installera från källkod . . . 4
2.2.1 Hämta källan . . . 5
2.2.2 Krav för installation . . . 5
2.2.3 Bygg konfiguration . . . 7
2.2.4 Byggnad . . . 8
2.2.5 Bygga PostGIS-tillägg och distribuera dem . . . 9
2.2.6 Testar . . . 11
2.2.7 Installation . . . 14
2.3 Vanliga problem under installationen . . . 14

3 PostGIS-administration **16**
3.1 Prestandajustering . . . 16
3.1.1 Uppstart . . . 16
3.1.2 Körtid . . . 17
3.2 Konfigurera stöd för raster . . . 18
3.3 Skapa spatiala databaser . . . 18
3.3.1 Spatial aktivering av databas med hjälp av EXTENSION . . . 18
3.3.2 Spatial aktivering av databas utan att använda EXTENSION (avrådes) . . . 19
3.4 Uppgradering av spatiala databaser . . . 19
3.4.1 Mjuk uppgradering . . . 20
3.4.1.1 Soft upgrade using extensions . . . 20
3.4.1.2 Soft upgrade without extensions . . . 21
3.4.1.3 Hård uppgradering . . . 21

4 Datahantering **24**
4.1 Modell för spatial data . . . 24
4.1.1 OGC Geometri . . . 24
4.1.1.1 Point . . . 25
4.1.1.2 LineString . . . 25
4.1.1.3 LinearRing . . . 25
4.1.1.4 Polygon . . . 25
4.1.1.5 MultiPoint . . . 26
4.1.1.6 MultiLineString . . . 26
4.1.1.7 MultiPolygon . . . 26
4.1.1.8 GeometriSamling . . . 26
4.1.1.9 PolyhedralSurface . . . 26
4.1.1.10Triangel . . . 27
4.1.1.11TIN . . . 27
4.1.2 SQL/MM Del 3 - Kurvor . . . 27
4.1.2.1 CircularString . . . 27
4.1.2.2 CompoundCurve . . . 27
4.1.2.3 CurvePolygon . . . 28
4.1.2.4 MultiCurve . . . 28
4.1.2.5 MultiSurface . . . 28
4.1.3 WKT och WKB . . . 28
4.2 Datatypen Geometry . . . 30
4.2.1 PostGIS EWKB och EWKT . . . 30
4.3 Datatypen Geography . . . 32
4.3.1 Skapa geografiska tabeller . . . 32
4.3.2 Använda geografiska tabeller . . . 33
4.3.3 När ska datatypen Geography användas . . . 35
4.3.4 Avancerade frågor och svar om Geography . . . 36
4.4 Validering av geometri . . . 36
4.4.1 Enkel geometri . . . 37
4.4.2 Giltig geometri . . . 39
4.4.3 Hantering av validitet . . . 43
4.5 Spatiala referenssystem . . . 45
4.5.1 Tabellen SPATIAL_REF_SYS . . . 46
4.5.2 Användardefinierade spatiala referenssystem . . . 47
4.6 Spatiala tabeller . . . 48
4.6.1 Skapa en spatial tabell . . . 48
4.6.2 Visa GEOMETRY_COLUMNS . . . 49
4.6.3 Manuell registrering av geometrikolumner . . . 50

4.7 Läs in spatial data . . . 52
4.7.1 Använda SQL för att läsa in data . . . 52
4.7.2 Använda Shapefile-inläsaren . . . 53
4.8 Extrahera spatiala data . . . 55
4.8.1 Använda SQL för att extrahera data . . . 55
4.8.2 Använda Shapefile Dumper . . . 56
4.9 Spatiala index . . . 57
4.9.1 GiST-index . . . 58
4.9.2 BRIN-index . . . 58
4.9.3 SP-GiST-index . . . 60
4.9.4 Anpassning av indexanvändning . . . 61

5 Spatiala frågor **63**
5.1 Fastställande av spatiala relationer . . . 63
5.1.1 Dimensionsutvidgad 9-intersektionell modell . . . 63
5.1.2 Namngivna spatiala relationer . . . 67
5.1.3 Allmänna spatiala relationer . . . 67
5.2 Använda spatiala index . . . 69
5.3 Exempel på Spatial SQL . . . 70

6 Tips om prestanda **73**
6.1 Små tabeller med stora geometrier . . . 73
6.1.1 Beskrivning av problemet . . . 73
6.1.2 Lösningar . . . 73
6.2 CLUSTERING på geometriindex . . . 74
6.3 Undvik dimensionskonvertering . . . 74

7 PostGIS-referens **75**
7.1 PostGIS-datatyper Geometry/Geography/Box . . . 75
7.1.1 box2d . . . 75
7.1.2 box3d . . . 76
7.1.3 geometry . . . 76
7.1.4 geometry_dump . . . 77
7.1.5 geography . . . 77
7.2 Funktioner för tabellhantering . . . 78
7.2.1 AddGeometryColumn . . . 78
7.2.2 DropGeometryColumn . . . 80
7.2.3 DropGeometryTable . . . 81
7.2.4 Find_SRID . . . 82
7.2.5 Populate_Geometry_Columns . . . 83

7.2.6 UpdateGeometrySRID . . . 85
7.3 Geometry Constructors . . . 86
7.3.1 ST_Collect . . . 86
7.3.2 ST_LineFromMultiPoint . . . 90
7.3.3 ST_MakeEnvelope . . . 91
7.3.4 ST_MakeLine . . . 92
7.3.5 ST_MakePoint . . . 96
7.3.6 ST_MakePointM . . . 97
7.3.7 ST_MakePolygon . . . 99
7.3.8 ST_Point . . . 103
7.3.9 ST_PointZ . . . 105
7.3.10 ST_PointM . . . 105
7.3.11 ST_PointZM . . . 106
7.3.12 ST_Polygon . . . 107
7.3.13 ST_TileEnvelope . . . 109
7.3.14 ST_HexagonGrid . . . 111
7.3.15 ST_Hexagon . . . 115
7.3.16 ST_SquareGrid . . . 117
7.3.17 ST_Square . . . 118
7.3.18 ST_Letters . . . 119
7.3.19 ST_MakeNurbsCurve . . . 121
7.4 Geometriåtkomst . . . 124
7.4.1 GeometryType . . . 124
7.4.2 ST_Boundary . . . 126
7.4.3 ST_BoundingDiagonal . . . 132
7.4.4 ST_CoordDim . . . 133
7.4.5 ST_Dimension . . . 134
7.4.6 ST_Dump . . . 137
7.4.7 ST_DumpPoints . . . 140
7.4.8 ST_DumpSegments . . . 145
7.4.9 ST_DumpRings . . . 148
7.4.10 ST_EndPoint . . . 150
7.4.11 ST_Envelope . . . 154
7.4.12 ST_ExteriorRing . . . 158
7.4.13 ST_GeometryN . . . 161
7.4.14 ST_GeometryType . . . 165
7.4.15 ST_HasArc . . . 166
7.4.16 ST_InteriorRingN . . . 167
7.4.17 ST_NumCurves . . . 169

7.4.18ST_CurveN . . . 170
7.4.19ST_IsClosed . . . 172
7.4.20ST_IsCollection . . . 176
7.4.21ST_IsEmpty . . . 178
7.4.22ST_IsPolygonCCW . . . 180
7.4.23ST_IsPolygonCW . . . 181
7.4.24ST_IsRing . . . 182
7.4.25ST_IsSimple . . . 184
7.4.26ST_M . . . 186
7.4.27ST_MemSize . . . 187
7.4.28ST_NDims . . . 188
7.4.29ST_NPoints . . . 189
7.4.30ST_NRings . . . 190
7.4.31ST_NumGeometries . . . 191
7.4.32ST_NumInteriorRings . . . 192
7.4.33ST_NumInteriorRing . . . 193
7.4.34ST_NumPatches . . . 193
7.4.35ST_NumPoints . . . 194
7.4.36ST_PatchN . . . 195
7.4.37ST_PointN . . . 196
7.4.38ST_Points . . . 199
7.4.39ST_StartPoint . . . 200
7.4.40ST_StartM . . . 205
7.4.41ST_EndM . . . 208
7.4.42ST_SetStartM . . . 211
7.4.43ST_SetEndM . . . 214
7.4.44ST_Summary . . . 217
7.4.45ST_X . . . 219
7.4.46ST_Y . . . 219
7.4.47ST_Z . . . 220
7.4.48ST_ZMFlag . . . 221
7.4.49ST_HasZ . . . 222
7.4.50ST_HasM . . . 223
7.4.51ST_ControlPoints . . . 225
7.4.52ST_Degree . . . 226
7.4.53ST_Weights . . . 228
7.4.54ST_Knots . . . 230
7.4.55ST_NumControlPoints . . . 232
7.4.56ST_NurbsCurveIsRational . . . 233

7.4.57ST_IsNurbsCurve . . . 235
7.4.58ST_Evaluate . . . 237
7.4.59ST_NurbsToLineString . . . 238
7.5 Geometriredigerare . . . 240
7.5.1 ST_AddPoint . . . 240
7.5.2 ST_CollectionExtract . . . 242
7.5.3 ST_CollectionHomogenize . . . 245
7.5.4 ST_CurveToLine . . . 249
7.5.5 ST_Scroll . . . 253
7.5.6 ST_FlipCoordinates . . . 254
7.5.7 ST_Force2D . . . 255
7.5.8 ST_Force3D . . . 257
7.5.9 ST_Force3DZ . . . 259
7.5.10ST_Force3DM . . . 261
7.5.11ST_Force4D . . . 263
7.5.12ST_ForceCollection . . . 265
7.5.13ST_ForceCurve . . . 268
7.5.14ST_ForcePolygonCCW . . . 269
7.5.15ST_ForcePolygonCW . . . 270
7.5.16ST_ForceSFS . . . 272
7.5.17ST_ForceRHR . . . 273
7.5.18ST_LineExtend . . . 274
7.5.19ST_LineToCurve . . . 275
7.5.20ST_Multi . . . 277
7.5.21ST_Normalize . . . 278
7.5.22ST_Project . . . 279
7.5.23ST_QuantizeCoordinates . . . 281
7.5.24ST_RemovePoint . . . 283
7.5.25ST_RemoveRepeatedPoints . . . 284
7.5.26ST_RemoveIrrelevantPointsForView . . . 287
7.5.27ST_RemoveSmallParts . . . 292
7.5.28ST_Reverse . . . 294
7.5.29ST_Segmentize . . . 295
7.5.30ST_SetPoint . . . 298
7.5.31ST_ShiftLongitude . . . 301
7.5.32ST_WrapX . . . 304
7.5.33ST_SnapToGrid . . . 304
7.5.34ST_Snap . . . 308
7.5.35ST_SwapOrdinates . . . 312

7.6 Validering av geometri . . . 313
7.6.1 ST_IsValid . . . 313
7.6.2 ST_IsValidDetail . . . 316
7.6.3 ST_IsValidReason . . . 318
7.6.4 ST_MakeValid . . . 320
7.7 Funktioner för spatialt referenssystem . . . 324
7.7.1 ST_InverseTransformPipeline . . . 324
7.7.2 ST_SetSRID . . . 326
7.7.3 ST_SRID . . . 328
7.7.4 ST_Transform . . . 329
7.7.5 ST_TransformPipeline . . . 333
7.7.6 postgis_srs_codes . . . 336
7.7.7 postgis_srs . . . 337
7.7.8 postgis_srs_all . . . 338
7.7.9 postgis_srs_search . . . 338
7.8 Inmatning av geometri . . . 340
7.8.1 Well-Known Text (WKT) . . . 340
7.8.1.1 ST_BdPolyFromText . . . 340
7.8.1.2 ST_BdMPolyFromText . . . 341
7.8.1.3 ST_GeogFromText . . . 341
7.8.1.4 ST_GeographyFromText . . . 342
7.8.1.5 ST_GeomCollFromText . . . 343
7.8.1.6 ST_GeomFromEWKT . . . 344
7.8.1.7 ST_GeomFromMARC21 . . . 346
7.8.1.8 ST_GeometryFromText . . . 350
7.8.1.9 ST_GeomFromText . . . 351
7.8.1.10 ST_LineFromText . . . 356
7.8.1.11 ST_MLineFromText . . . 357
7.8.1.12 ST_MPointFromText . . . 358
7.8.1.13 ST_MPolyFromText . . . 359
7.8.1.14 ST_PointFromText . . . 361
7.8.1.15 ST_PolygonFromText . . . 363
7.8.1.16 ST_WKTToSQL . . . 365
7.8.2 Välkänd binär (WKB) . . . 365
7.8.2.1 ST_GeogFromWKB . . . 365
7.8.2.2 ST_GeomFromEWKB . . . 366
7.8.2.3 ST_GeomFromWKB . . . 368
7.8.2.4 ST_LineFromWKB . . . 370
7.8.2.5 ST_LinestringFromWKB . . . 372

7.8.2.6 ST_PointFromWKB . . . 374
7.8.2.7 ST_WKBToSQL . . . 375
7.8.3 Andra format . . . 376
7.8.3.1 ST_Box2dFromGeoHash . . . 376
7.8.3.2 ST_GeomFromGeoHash . . . 376
7.8.3.3 ST_GeomFromGML . . . 378
7.8.3.4 ST_GeomFromGeoJSON . . . 382
7.8.3.5 ST_GeomFromKML . . . 384
7.8.3.6 ST_GeomFromTWKB . . . 385
7.8.3.7 ST_GMLToSQL . . . 387
7.8.3.8 ST_LineFromEncodedPolyline . . . 388
7.8.3.9 ST_PointFromGeoHash . . . 389
7.8.3.10 ST_FromFlatGeobufToTable . . . 392
7.8.3.11 ST_FromFlatGeobuf . . . 392
7.9 Geometriutdata . . . 392
7.9.1 Well-Known Text (WKT) . . . 392
7.9.1.1 ST_AsEWKT . . . 392
7.9.1.2 ST_AsText . . . 395
7.9.2 Välkänd binär (WKB) . . . 397
7.9.2.1 ST_AsBinary . . . 397
7.9.2.2 ST_AsEWKB . . . 399
7.9.2.3 ST_AsHEXEWKB . . . 400
7.9.3 Andra format . . . 401
7.9.3.1 ST_AsEncodedPolyline . . . 401
7.9.3.2 ST_AsFlatGeobuf . . . 403
7.9.3.3 ST_AsGeobuf . . . 403
7.9.3.4 ST_AsGeoJSON . . . 404
7.9.3.5 ST_AsGML . . . 407
7.9.3.6 ST_AsKML . . . 410
7.9.3.7 ST_AsLatLonText . . . 411
7.9.3.8 ST_AsMARC21 . . . 413
7.9.3.9 ST_AsMVTGeom . . . 416
7.9.3.10 ST_AsMVT . . . 417
7.9.3.11 ST_AsSVG . . . 419
7.9.3.12 ST_AsTWKB . . . 422
7.9.3.13 ST_AsX3D . . . 423
7.9.3.14 ST_GeoHash . . . 428
7.10 Operatorer . . . 429
7.10.1 Operatorer för avgränsande box . . . 429

7.10.1.1 && . . . 429
7.10.1.2 &&(geometry,box2df) . . . 431
7.10.1.3 &&(box2df,geometry) . . . 431
7.10.1.4 &&(box2df,box2df) . . . 432
7.10.1.5 &&& . . . 433
7.10.1.6 &&&(geometry,gidx) . . . 435
7.10.1.7 &&&(gidx,geometry) . . . 436
7.10.1.8 &&&(gidx,gidx) . . . 437
7.10.1.9 &< . . . 438
7.10.1.10 &<| . . . 440
7.10.1.11 &> . . . 441
7.10.1.12 <& . . . 442
7.10.1.13 <&| . . . 444
7.10.1.14 << . . . 445
7.10.1.15 <<| . . . 448
7.10.1.16 @ . . . 449
7.10.1.17 @(geometry,box2df) . . . 450
7.10.1.18 @(box2df,geometry) . . . 451
7.10.1.19 @(box2df,box2df) . . . 452
7.10.1.20 |&> . . . 453
7.10.1.21 |>> . . . 454
7.10.1.22 ~ . . . 455
7.10.1.23 ~(geometry,box2df) . . . 457
7.10.1.24 ~(box2df,geometry) . . . 458
7.10.1.25 ~(box2df,box2df) . . . 459
7.10.1.26 ~= . . . 460
7.10.2 Avståndsoperatorer . . . 461
7.10.2.1 <-> . . . 461
7.10.2.2 |=| . . . 463
7.10.2.3 <#> . . . 464
7.10.2.4 <<->> . . . 466
7.11 Spatiala relationer . . . 466
7.11.1 Topologiska relationer . . . 466
7.11.1.1 ST_3DIntersects . . . 466
7.11.1.2 ST_Contains . . . 469
7.11.1.3 ST_ContainsProperly . . . 480
7.11.1.4 ST_CoveredBy . . . 483
7.11.1.5 ST_Covers . . . 485
7.11.1.6 ST_Crosses . . . 487

7.11.1.7 ST_Disjoint . . . 491
7.11.1.8 ST_Equals . . . 493
7.11.1.9 ST_Intersects . . . 495
7.11.1.10 ST_LineCrossingDirection . . . 498
7.11.1.11 ST_OrderingEquals . . . 501
7.11.1.12 ST_Overlaps . . . 504
7.11.1.13 ST_Relate . . . 510
7.11.1.14 ST_RelateMatch . . . 514
7.11.1.15 ST_Touches . . . 515
7.11.1.16 ST_Within . . . 523
7.11.2 Relationer på distans . . . 526
7.11.2.1 ST_3DDWithin . . . 526
7.11.2.2 ST_3DDFullyWithin . . . 527
7.11.2.3 ST_DFullyWithin . . . 529
7.11.2.4 ST_DWithin . . . 530
7.11.2.5 ST_PointInsideCircle . . . 533
7.12 Mätningsfunktioner . . . 534
7.12.1 ST_Area . . . 534
7.12.2 ST_Azimuth . . . 538
7.12.3 ST_Angle . . . 540
7.12.4 ST_ClosestPoint . . . 543
7.12.5 ST_3DClosestPoint . . . 545
7.12.6 ST_Distance . . . 548
7.12.7 ST_3DDistance . . . 550
7.12.8 ST_DistanceSphere . . . 552
7.12.9 ST_DistanceSpheroid . . . 553
7.12.10 ST_FrechetDistance . . . 555
7.12.11 ST_HausdorffDistance . . . 556
7.12.12 ST_Length . . . 559
7.12.13 ST_Length2D . . . 562
7.12.14 ST_3DLength . . . 562
7.12.15 ST_LengthSpheroid . . . 564
7.12.16 ST_LongestLine . . . 566
7.12.17 ST_3DLongestLine . . . 569
7.12.18 ST_MaxDistance . . . 572
7.12.19 ST_3DMaxDistance . . . 575
7.12.20 ST_MinimumClearance . . . 576
7.12.21 ST_MinimumClearanceLine . . . 577
7.12.22 ST_Perimeter . . . 578

7.12.2 ST_Perimeter2D . . . 582
7.12.3 ST_3DPerimeter . . . 583
7.12.4 ST_ShortestLine . . . 584
7.12.5 ST_3DShortestLine . . . 586
7.13 Funktioner för överlagring . . . 589
7.13.1 ST_ClipByBox2D . . . 589
7.13.2 ST_Difference . . . 590
7.13.3 ST_Intersection . . . 592
7.13.4 ST_MemUnion . . . 596
7.13.5 ST_Node . . . 596
7.13.6 ST_Split . . . 598
7.13.7 ST_Subdivide . . . 602
7.13.8 ST_SymDifference . . . 606
7.13.9 ST_UnaryUnion . . . 608
7.13.10 ST_Union . . . 608
7.14 Geometribearbetning . . . 614
7.14.1 ST_Buffer . . . 614
7.14.2 ST_BuildArea . . . 623
7.14.3 ST_Centroid . . . 626
7.14.4 ST_ChaikinSmoothing . . . 632
7.14.5 ST_CatmullRomSmoothing . . . 635
7.14.6 ST_ConcaveHull . . . 638
7.14.7 ST_ConvexHull . . . 642
7.14.8 ST_DelaunayTriangles . . . 644
7.14.9 ST_FilterByM . . . 647
7.14.10 ST_GeneratePoints . . . 649
7.14.11 ST_GeometricMedian . . . 651
7.14.12 ST_LineMerge . . . 652
7.14.13 ST_MaximumInscribedCircle . . . 657
7.14.14 ST_LargestEmptyCircle . . . 659
7.14.15 ST_MinimumBoundingCircle . . . 661
7.14.16 ST_MinimumBoundingRadius . . . 663
7.14.17 ST_OrientedEnvelope . . . 664
7.14.18 ST_OffsetCurve . . . 666
7.14.19 ST_PointOnSurface . . . 673
7.14.20 ST_Polygonize . . . 678
7.14.21 ST_ReducePrecision . . . 681
7.14.22 ST_SharedPaths . . . 685
7.14.23 ST_Simplify . . . 687

7.14.24 ST_SimplifyPreserveTopology . . . 690
7.14.25 ST_SimplifyPolygonHull . . . 693
7.14.26 ST_SimplifyVW . . . 696
7.14.27 ST_SetEffectiveArea . . . 700
7.14.28 ST_TriangulatePolygon . . . 701
7.14.29 ST_VoronoiLines . . . 704
7.14.30 ST_VoronoiPolygons . . . 705
7.15 Täckning . . . 707
7.15.1 ST_CoverageInvalidEdges . . . 707
7.15.2 ST_CoverageClean . . . 709
7.15.3 ST_CoverageSimplify . . . 711
7.15.4 ST_CoverageEdges . . . 712
7.15.5 ST_CoverageUnion . . . 713
7.16 Affina transformationer . . . 715
7.16.1 ST_Affine . . . 715
7.16.2 ST_Rotate . . . 718
7.16.3 ST_RotateX . . . 720
7.16.4 ST_RotateY . . . 721
7.16.5 ST_RotateZ . . . 723
7.16.6 ST_Scale . . . 725
7.16.7 ST_Translate . . . 728
7.16.8 ST_TransScale . . . 731
7.17 Klustringsfunktioner . . . 733
7.17.1 ST_ClusterDBSCAN . . . 733
7.17.2 ST_ClusterIntersecting . . . 735
7.17.3 ST_ClusterIntersectingWin . . . 736
7.17.4 ST_ClusterRelateWin . . . 738
7.17.5 ST_MinimumSpanningTree . . . 739
7.17.6 ST_ClusterKMeans . . . 741
7.17.7 ST_ClusterWithin . . . 745
7.17.8 ST_ClusterWithinWin . . . 746
7.18 Funktioner för avgränsande box . . . 748
7.18.1 Box2D . . . 748
7.18.2 Box3D . . . 750
7.18.3 ST_EstimatedExtent . . . 752
7.18.4 ST_Expand . . . 753
7.18.5 ST_Extent . . . 756
7.18.6 ST_3DExtent . . . 758
7.18.7 ST_MakeBox2D . . . 759

7.18.8 ST_3DMakeBox . . . 759
7.18.9 ST_XMax . . . 760
7.18.10 ST_XMin . . . 762
7.18.11 ST_YMax . . . 764
7.18.12 ST_YMin . . . 767
7.18.13 ST_ZMax . . . 769
7.18.14 ST_ZMin . . . 771
7.18.15 ST_XSize . . . 774
7.18.16 ST_YSize . . . 775
7.18.17 ST_ZSize . . . 777
7.18.18 ST_MMin . . . 778
7.18.19 ST_MMax . . . 779
7.18.20 ST_MSize . . . 779
7.19 Linjär referenstagning . . . 780
7.19.1 ST_LineInterpolatePoint . . . 780
7.19.2 ST_3DLineInterpolatePoint . . . 784
7.19.3 ST_LineInterpolatePoints . . . 785
7.19.4 ST_LineLocatePoint . . . 787
7.19.5 ST_LineSubstring . . . 790
7.19.6 ST_LocateAlong . . . 796
7.19.7 ST_LocateBetween . . . 797
7.19.8 ST_LocateBetweenElevations . . . 799
7.19.9 ST_InterpolatePoint . . . 801
7.19.10 ST_3DInterpolatePoint . . . 802
7.19.11 ST_AddMeasure . . . 804
7.20 Trajektoriefunktioner . . . 807
7.20.1 ST_IsValidTrajectory . . . 807
7.20.2 ST_ClosestPointOfApproach . . . 808
7.20.3 ST_DistanceCPA . . . 810
7.20.4 ST_CPAWithin . . . 811
7.21 Versionsfunktioner . . . 812
7.21.1 PostGIS_Extensions_Upgrade . . . 812
7.21.2 PostGIS_Full_Version . . . 813
7.21.3 PostGIS_GEOS_Version . . . 814
7.21.4 PostGIS_GEOS_Compiled_Version . . . 814
7.21.5 PostGIS_Liblwgeom_Version . . . 815
7.21.6 PostGIS_LibXML_Version . . . 816
7.21.7 PostGIS_LibJSON_Version . . . 816
7.21.8 PostGIS_Lib_Build_Date . . . 817

7.21.9 PostGIS_Lib_Version . . . 817
7.21.10 PostGIS_PROJ_Version . . . 818
7.21.11 PostGIS_PROJ_Compiled_Version . . . 818
7.21.12 PostGIS_Wagyu_Version . . . 819
7.21.13 PostGIS_Scripts_Build_Date . . . 820
7.21.14 PostGIS_Scripts_Installed . . . 820
7.21.15 PostGIS_Scripts_Released . . . 821
7.21.16 PostGIS_Version . . . 822
7.22 Grand Unified Custom Variables (GUC) . . . 822
7.22.1 postgis.gdal_datapath . . . 822
7.22.2 postgis.gdal_enabled_drivers . . . 823
7.22.3 postgis.enable_outdb_rasters . . . 825
7.22.4 postgis.gdal_vsi_options . . . 826
7.22.5 postgis.gdal_cpl_debug . . . 827
7.23 Felsökningsfunktioner . . . 827
7.23.1 PostGIS_AddBBox . . . 827
7.23.2 PostGIS_DropBBox . . . 828
7.23.3 PostGIS_HasBBox . . . 829

8 SFCGAL Funktioner Referens 830
8.1 SFCGAL-hanteringsfunktioner . . . 830
8.1.1 postgis_sfcgal_version . . . 830
8.1.2 postgis_sfcgal_full_version . . . 830
8.2 SFCGAL-accessorer och Setters . . . 831
8.2.1 CG_ForceLHR . . . 831
8.2.2 CG_IsPlanar . . . 832
8.2.3 CG_IsSolid . . . 833
8.2.4 CG_MakeSolid . . . 834
8.2.5 CG_Orientation . . . 835
8.2.6 CG_Area . . . 835
8.2.7 CG_3DArea . . . 836
8.2.8 CG_Volume . . . 837
8.3 SFCGAL bearbetnings- och relationsfunktioner . . . 838
8.3.1 CG_Intersection . . . 838
8.3.2 CG_Intersects . . . 840
8.3.3 CG_3DIntersects . . . 842
8.3.4 CG_Difference . . . 843
8.3.5 CG_3DDifference . . . 844
8.3.6 CG_Distance . . . 846

8.3.7 CG_3DDistance . . . 847
8.3.8 CG_3DConvexHull . . . 848
8.3.9 CG_3DIntersection . . . 851
8.3.10 CG_Union . . . 854
8.3.11 CG_3DUnion . . . 855
8.3.12 CG_AlphaShape . . . 857
8.3.13 CG_ApproxConvexPartition . . . 860
8.3.14 CG_ApproximateMedialAxis . . . 862
8.3.15 CG_GenerateFlatRoof . . . 864
8.3.16 CG_GenerateHippedRoof . . . 865
8.3.17 CG_GenerateGableRoof . . . 866
8.3.18 CG_GenerateSkillionRoof . . . 867
8.3.19 CG_GenerateRoof . . . 869
8.3.20 CG_PolygonRepair . . . 870
8.3.21 CG_ConstrainedDelaunayTriangles . . . 873
8.3.22 CG_Extrude . . . 875
8.3.23 CG_ExtrudeStraightSkeleton . . . 877
8.3.24 CG_GreeneApproxConvexPartition . . . 879
8.3.25 CG_MinkowskiSum . . . 880
8.3.26 CG_OptimalAlphaShape . . . 882
8.3.27 CG_OptimalConvexPartition . . . 884
8.3.28 CG_StraightSkeleton . . . 886
8.3.29 CG_Tesselate . . . 888
8.3.30 CG_Triangulate . . . 890
8.3.31 CG_Visibility . . . 892
8.3.32 CG_YMonotonePartition . . . 894
8.3.33 CG_StraightSkeletonPartition . . . 895
8.3.34 CG_3DBuffer . . . 897
8.3.35 CG_Rotate . . . 900
8.3.36 CG_2DRotate . . . 901
8.3.37 CG_3DRotate . . . 902
8.3.38 CG_RotateX . . . 903
8.3.39 CG_RotateY . . . 904
8.3.40 CG_RotateZ . . . 905
8.3.41 CG_Scale . . . 906
8.3.42 CG_3DScale . . . 907
8.3.43 CG_3DScaleAroundCenter . . . 908
8.3.44 CG_Translate . . . 909
8.3.45 CG_3DTranslate . . . 910

8.3.46CG_Simplify . . . 911
8.3.47CG_3DAlphaWrapping . . . 916
8.3.48CG_NurbsCurveInterpolate . . . 917
8.3.49CG_NurbsCurveApproximate . . . 919
8.3.50CG_NurbsCurveDerivative . . . 922
8.4 Deprecated SFCGAL Function Aliases . . . 925
8.4.1 ST_3DIntersection . . . 925
8.4.2 ST_3DDifference . . . 926
8.4.3 ST_3DUnion . . . 926
8.4.4 ST_Tesselate . . . 926
8.4.5 ST_3DArea . . . 927
8.4.6 ST_Extrude . . . 927
8.4.7 ST_ForceLHR . . . 928
8.4.8 ST_Orientation . . . 928
8.4.9 ST_MinkowskiSum . . . 929
8.4.10ST_StraightSkeleton . . . 929
8.4.11ST_ApproximateMedialAxis . . . 929
8.4.12ST_IsPlanar . . . 930
8.4.13ST_Volume . . . 930
8.4.14ST_MakeSolid . . . 931
8.4.15ST_IsSolid . . . 931
8.4.16ST_ConstrainedDelaunayTriangles . . . 932
8.4.17ST_3DConvexHull . . . 932
8.4.18ST_AlphaShape . . . 932
8.4.19ST_OptimalAlphaShape . . . 933

9 Topologi 934
9.1 Building Faces from Linework . . . 934
9.2 Topology Primitive Tables . . . 936
9.3 Topologityper . . . 937
9.3.1 getfaceedges_returntype . . . 937
9.3.2 TopoGeometry . . . 937
9.3.3 validatetopology_returntype . . . 938
9.4 Topologidomäner . . . 938
9.4.1 TopoElement . . . 938
9.4.2 TopoElementArray . . . 939
9.5 Hantering av topologi och topogeometri . . . 940
9.5.1 AddTopoGeometryColumn . . . 940
9.5.2 RenameTopoGeometryColumn . . . 941

9.5.3 DropTopology . . . 942
9.5.4 RenameTopology . . . 942
9.5.5 DropTopoGeometryColumn . . . 943
9.5.6 FixCorruptTopoGeometryColumn . . . 943
9.5.7 Populate_Topology_Layer . . . 944
9.5.8 TopologySummary . . . 946
9.5.9 ValidateTopology . . . 946
9.5.10 ValidateTopologyRelation . . . 949
9.5.11 ValidateTopologyPrecision . . . 950
9.5.12 MakeTopologyPrecise . . . 951
9.5.13 FindTopology . . . 951
9.5.14 FindLayer . . . 952
9.5.15 TotalTopologySize . . . 953
9.5.16 UpgradeTopology . . . 954
9.6 Hantering av topologistatistik . . . 954
9.7 Topology Constructors . . . 954
9.7.1 CreateTopology . . . 954
9.7.2 CopyTopology . . . 955
9.7.3 ST_InitTopoGeo . . . 956
9.7.4 ST_CreateTopoGeo . . . 957
9.7.5 TopoGeo_AddPoint . . . 958
9.7.6 TopoGeo_AddLineString . . . 958
9.7.7 TopoGeo_AddPolygon . . . 959
9.7.8 TopoGeo_LoadGeometry . . . 960
9.8 Topologiredigerare . . . 960
9.8.1 ST_AddIsoNode . . . 960
9.8.2 ST_AddIsoEdge . . . 961
9.8.3 ST_AddEdgeNewFaces . . . 962
9.8.4 ST_AddEdgeModFace . . . 962
9.8.5 ST_RemEdgeNewFace . . . 963
9.8.6 ST_RemEdgeModFace . . . 964
9.8.7 ST_ChangeEdgeGeom . . . 964
9.8.8 ST_ModEdgeSplit . . . 965
9.8.9 ST_ModEdgeHeal . . . 966
9.8.10 ST_NewEdgeHeal . . . 967
9.8.11 ST_MoveIsoNode . . . 967
9.8.12 ST_NewEdgesSplit . . . 968
9.8.13 ST_RemoveIsoNode . . . 969
9.8.14 ST_RemoveIsoEdge . . . 970

9.9 Topologi-accessorer . . . 971
9.9.1 GetEdgeByPoint . . . 971
9.9.2 GetFaceByPoint . . . 972
9.9.3 GetFaceContainingPoint . . . 973
9.9.4 GetNodeByPoint . . . 973
9.9.5 GetTopologyID . . . 974
9.9.6 GetTopologySRID . . . 975
9.9.7 GetTopologyName . . . 976
9.9.8 ST_GetFaceEdges . . . 976
9.9.9 ST_GetFaceGeometry . . . 977
9.9.10 GetRingEdges . . . 978
9.9.11 GetNodeEdges . . . 979
9.9.12 FindVertexSegmentPairsBelowDistance . . . 980
9.10 Bearbetning av topologi . . . 980
9.10.1 Polygonize . . . 980
9.10.2 AddNode . . . 981
9.10.3 AddEdge . . . 981
9.10.4 AddFace . . . 983
9.10.5 ST_Simplify . . . 984
9.10.6 RemoveUnusedPrimitives . . . 985
9.11 TopoGeometry Constructors . . . 985
9.11.1 CreateTopoGeom . . . 985
9.11.2 toTopoGeom . . . 988
9.11.3 TopoElementArray_Agg . . . 990
9.11.4 TopoElement . . . 990
9.12 TopoGeometry-redigerare . . . 991
9.12.1 clearTopoGeom . . . 991
9.12.2 TopoGeom_addElement . . . 992
9.12.3 TopoGeom_remElement . . . 992
9.12.4 TopoGeom_addTopoGeom . . . 993
9.12.5 toTopoGeom . . . 993
9.13 TopoGeometry-accessorer . . . 994
9.13.1 GetTopoGeomElementArray . . . 994
9.13.2 GetTopoGeomElements . . . 994
9.13.3 ST_SRID . . . 995
9.14 TopoGeometry-utdata . . . 996
9.14.1 AsGML . . . 996
9.14.2 AsTopoJSON . . . 998
9.15 Spatiala relationer mellan topologier . . . 999

9.15.1 Equals . . . 999
9.15.2 Intersects . . . 1000
9.16 Importera och exportera topologier . . . 1001
9.16.1 Använda topologiexportören . . . 1002
9.16.2 Använda topologiimportören . . . 1002

10 Rasterdatahantering, frågor och applikationer 1003
10.1 Läs in och skapa raster . . . 1003
10.1.1 Använda raster2pgsql för att läsa in raster . . . 1003
10.1.1.1 Exempel på användning . . . 1003
10.1.1.2 raster2pgsql-alternativ . . . 1004
10.1.2 Skapa raster med hjälp av PostGIS rasterfunktioner . . . 1006
10.1.3 Använda öut db-molnraster . . . 1007
10.2 Raster Coverages and Raster/Vector Analysis . . . 1008
10.3 Rasterkataloger . . . 1009
10.3.1 Rasterkolumner Katalog . . . 1009
10.3.2 Raster-översikter . . . 1010
10.4 Bygga anpassade applikationer med PostGIS Raster . . . 1011
10.4.1 PHP-exempel Utdata med ST_AsPNG i kombination med andra rasterfunktioner . . . 1012
10.4.2 ASP.NET C# Exempel Utdata med hjälp av ST_AsPNG tillsammans med andra rasterfunktioner . . . 1012
10.4.3 Java-konsolapp som matar ut rasterfråga som bildfil . . . 1014
10.4.4 Använd PLPython för att dumpa bilder via SQL . . . 1016
10.4.5 Utdata av raster med PSQL . . . 1016

11 Referens för raster 1018
11.1 Rasterstöd Datatyper . . . 1019
11.1.1 geomval . . . 1019
11.1.2 addbandarg . . . 1019
11.1.3 rastbandarg . . . 1020
11.1.4 raster . . . 1020
11.1.5 reclassarg . . . 1021
11.1.6 summarystats . . . 1022
11.1.7 unionarg . . . 1022
11.2 Rasterhantering . . . 1023
11.2.1 AddRasterConstraints . . . 1023
11.2.2 DropRasterConstraints . . . 1025
11.2.3 AddOverviewConstraints . . . 1026
11.2.4 DropOverviewConstraints . . . 1027
11.2.5 PostGIS_GDAL_Version . . . 1027

11.2.6 PostGIS_Raster_Lib_Build_Date . . . 1028
11.2.7 PostGIS_Raster_Lib_Version . . . 1028
11.2.8 ST_GDALDrivers . . . 1029
11.2.9 UpdateRasterSRID . . . 1032
11.2.10 ST_CreateOverview . . . 1032
11.3 Raster Constructors . . . 1033
11.3.1 ST_AddBand . . . 1033
11.3.2 ST_AsRaster . . . 1036
11.3.3 ST_AsRasterAgg . . . 1039
11.3.4 ST_Band . . . 1040
11.3.5 ST_MakeEmptyCoverage . . . 1044
11.3.6 ST_MakeEmptyRaster . . . 1046
11.3.7 ST_Tile . . . 1047
11.3.8 ST_Retile . . . 1049
11.3.9 ST_FromGDALRaster . . . 1050
11.4 Raster-accessorer . . . 1051
11.4.1 ST_GeoReference . . . 1051
11.4.2 ST_Height . . . 1052
11.4.3 ST_IsEmpty . . . 1053
11.4.4 ST_MemSize . . . 1053
11.4.5 ST_MetaData . . . 1054
11.4.6 ST_NumBands . . . 1055
11.4.7 ST_PixelHeight . . . 1056
11.4.8 ST_PixelWidth . . . 1057
11.4.9 ST_ScaleX . . . 1059
11.4.10 ST_ScaleY . . . 1059
11.4.11 ST_RasterToWorldCoord . . . 1060
11.4.12 ST_RasterToWorldCoordX . . . 1061
11.4.13 ST_RasterToWorldCoordY . . . 1062
11.4.14 ST_Rotation . . . 1063
11.4.15 ST_SkewX . . . 1064
11.4.16 ST_SkewY . . . 1065
11.4.17 ST_SRID . . . 1066
11.4.18 ST_Summary . . . 1067
11.4.19 ST_UpperLeftX . . . 1067
11.4.20 ST_UpperLeftY . . . 1068
11.4.21 ST_Width . . . 1068
11.4.22 ST_WorldToRasterCoord . . . 1069
11.4.23 ST_WorldToRasterCoordX . . . 1070

11.4.28 ST_WorldToRasterCoordY . . . 1071
11.5 Rasterband-accessorer . . . 1072
11.5.1 ST_BandMetaData . . . 1072
11.5.2 ST_BandNoDataValue . . . 1073
11.5.3 ST_BandIsNoData . . . 1074
11.5.4 ST_BandPath . . . 1076
11.5.5 ST_BandFileSize . . . 1076
11.5.6 ST_BandFileTimestamp . . . 1077
11.5.7 ST_BandPixelType . . . 1078
11.5.8 ST_MinPossibleValue . . . 1079
11.5.9 ST_HasNoBand . . . 1079
11.6 Raster Pixel-accessorer och Setters . . . 1080
11.6.1 ST_PixelAsPolygon . . . 1080
11.6.2 ST_PixelAsPolygons . . . 1081
11.6.3 ST_PixelAsPoint . . . 1082
11.6.4 ST_PixelAsPoints . . . 1083
11.6.5 ST_PixelAsCentroid . . . 1085
11.6.6 ST_PixelAsCentroids . . . 1086
11.6.7 ST_Value . . . 1088
11.6.8 ST_NearestValue . . . 1091
11.6.9 ST_SetZ . . . 1093
11.6.10 ST_SetM . . . 1095
11.6.11 ST_Neighborhood . . . 1097
11.6.12 ST_SetValue . . . 1099
11.6.13 ST_SetValues . . . 1100
11.6.14 ST_DumpValues . . . 1108
11.6.15 ST_PixelOfValue . . . 1109
11.7 Raster-redigerare . . . 1110
11.7.1 ST_SetGeoReference . . . 1110
11.7.2 ST_SetRotation . . . 1112
11.7.3 ST_SetScale . . . 1113
11.7.4 ST_SetSkew . . . 1114
11.7.5 ST_SetSRID . . . 1115
11.7.6 ST_SetUpperLeft . . . 1116
11.7.7 ST_Resample . . . 1116
11.7.8 ST_Rescale . . . 1118
11.7.9 ST_Reskew . . . 1120
11.7.10 ST_SnapToGrid . . . 1121
11.7.11 ST_Resize . . . 1122

11.7.12 ST_Transform . . . 1124
11.8 Rasterbandredigerare . . . 1128
11.8.1 ST_SetBandNoDataValue . . . 1128
11.8.2 ST_SetBandIsNoData . . . 1129
11.8.3 ST_SetBandPath . . . 1131
11.8.4 ST_SetBandIndex . . . 1132
11.9 Statistik och analys av rasterband . . . 1133
11.9.1 ST_Count . . . 1133
11.9.2 ST_CountAgg . . . 1134
11.9.3 ST_Histogram . . . 1135
11.9.4 ST_Quantile . . . 1137
11.9.5 ST_SummaryStats . . . 1139
11.9.6 ST_SummaryStatsAgg . . . 1142
11.9.7 ST_ValueCount . . . 1143
11.10 Raster-indata . . . 1146
11.10.1 ST_RastFromWKB . . . 1146
11.10.2 ST_RastFromHexWKB . . . 1147
11.11 Rasterutdata . . . 1147
11.11.1 ST_AsBinary/ST_AsWKB . . . 1147
11.11.2 ST_AsHexWKB . . . 1148
11.11.3 ST_AsGDALRaster . . . 1149
11.11.4 ST_AsJPEG . . . 1150
11.11.5 ST_AsPNG . . . 1151
11.11.6 ST_AsTIFF . . . 1152
11.12 Rasterbearbetning: Kartalgebra . . . 1153
11.12.1 ST_Clip . . . 1153
11.12.2 ST_ColorMap . . . 1159
11.12.3 ST_Grayscale . . . 1162
11.12.4 ST_Intersection . . . 1166
11.12.5 ST_MapAlgebra (callback function version) . . . 1169
11.12.6 ST_MapAlgebra (expression version) . . . 1177
11.12.7 ST_MapAlgebraExpr . . . 1179
11.12.8 ST_MapAlgebraExpr . . . 1183
11.12.9 ST_MapAlgebraFct . . . 1191
11.12.10 ST_MapAlgebraFct . . . 1195
11.12.11 ST_MapAlgebraFctNgb . . . 1201
11.12.12 ST_Reclass . . . 1205
11.12.13 ST_ReclassExact . . . 1207
11.12.14 ST_Union . . . 1208

11.13 Inbyggda Map Algebra återuppringningsfunktioner . . . 1210
11.13.1 ST_Distinct4ma . . . 1210
11.13.2 ST_InvDistWeight4ma . . . 1211
11.13.3 ST_Max4ma . . . 1212
11.13.4 ST_Mean4ma . . . 1213
11.13.5 ST_Min4ma . . . 1215
11.13.6 ST_MinDist4ma . . . 1216
11.13.7 ST_Range4ma . . . 1217
11.13.8 ST_StdDev4ma . . . 1218
11.13.9 ST_Sum4ma . . . 1219
11.14 Rasterbearbetning: DEM (höjdsystem) . . . 1221
11.14.1 ST_Aspect . . . 1221
11.14.2 ST_HillShade . . . 1223
11.14.3 ST_Roughness . . . 1225
11.14.4 ST_Slope . . . 1226
11.14.5 ST_TPI . . . 1228
11.14.6 ST_TRI . . . 1229
11.14.7 ST_InterpolateRaster . . . 1230
11.14.8 ST_Contour . . . 1231
11.15 Rasterbearbetning: Raster till geometri . . . 1232
11.15.1 Box3D . . . 1232
11.15.2 ST_ConvexHull . . . 1233
11.15.3 ST_DumpAsPolygons . . . 1234
11.15.4 ST_Envelope . . . 1235
11.15.5 ST_MinConvexHull . . . 1235
11.15.6 ST_Polygon . . . 1237
11.15.7 ST_IntersectionFractions . . . 1239
11.16 Rasteroperatorer . . . 1240
11.16.1 && . . . 1240
11.16.2 &< . . . 1241
11.16.3 &> . . . 1242
11.16.4 = . . . 1242
11.16.5 @ . . . 1243
11.16.6 ~= . . . 1243
11.16.7 ~ . . . 1244
11.17 Raster och Rasterband - spatiala relationer . . . 1245
11.17.1 ST_Contains . . . 1245
11.17.2 ST_ContainsProperly . . . 1246
11.17.3 ST_Covers . . . 1247

11.17.4 ST_CoveredBy . . . 1248
11.17.5 ST_Disjoint . . . 1249
11.17.6 ST_Intersects . . . 1250
11.17.7 ST_Overlaps . . . 1251
11.17.8 ST_Touches . . . 1252
11.17.9 ST_SameAlignment . . . 1253
11.17.10 ST_NotSameAlignmentReason . . . 1254
11.17.11 ST_Within . . . 1255
11.17.12 ST_DWithin . . . 1256
11.17.13 ST_DFullyWithin . . . 1257
11.18 Tips om raster . . . 1258
11.18.1 Ut-DB Rasters . . . 1258
11.18.1.1 Katalog som innehåller många filer . . . 1258
11.18.1.2 Maximalt antal öppna filer . . . 1258
11.18.1.2.1 Maximalt antal öppna filer för hela systemet . . . 1259
11.18.1.2.2 Maximalt antal öppna filer per process . . . 1259

12 PostGIS Extrafunktioner 1262

13 Index över specialfunktioner i PostGIS 1263
13.1 PostGIS aggregerade funktioner . . . 1263
13.2 Funktioner i PostGIS-fönstret . . . 1264
13.3 PostGIS SQL-MM-kompatibla funktioner . . . 1265
13.4 Stödfunktioner för geografi i PostGIS . . . 1268
13.5 Stödfunktioner för PostGIS Raster . . . 1270
13.6 PostGIS Geometri / Geografi / Raster Dumpfunktioner . . . 1276
13.7 PostGIS Box-funktioner . . . 1277
13.8 PostGIS-funktioner med stöd för 3D . . . 1278
13.9 Stödfunktioner för krökt geometri i PostGIS . . . 1284
13.10 PostGIS stödfunktioner för polyedriska ytor . . . 1288
13.11 PostGIS matris för funktionsstöd . . . 1291
13.12 Nya, förbättrade eller ändrade PostGIS-funktioner . . . 1310
13.12.1 PostGIS Funktioner nya eller förbättrade i 3.7 . . . 1310
13.12.2 PostGIS Funktioner nya eller förbättrade i 3.6 . . . 1312
13.12.3 PostGIS Funktioner nya eller förbättrade i 3.5 . . . 1314
13.12.4 PostGIS Funktioner nya eller förbättrade i 3.4 . . . 1315
13.12.5 PostGIS Funktioner nya eller förbättrade i 3.3 . . . 1317
13.12.6 PostGIS Funktioner nya eller förbättrade i 3.2 . . . 1318
13.12.7 PostGIS Funktioner nya eller förbättrade i 3.1 . . . 1319

13.12.8 PostGIS Funktioner nya eller förbättrade i 3.0 . . . 1321
13.12.9 PostGIS Funktioner nya eller förbättrade i 2.5 . . . 1323
13.12.10 PostGIS Funktioner nya eller förbättrade i 2.4 . . . 1324
13.12.11 PostGIS Funktioner nya eller förbättrade i 2.3 . . . 1325
13.12.12 PostGIS Funktioner nya eller förbättrade i 2.2 . . . 1327
13.12.13 PostGIS Funktioner nya eller förbättrade i 2.1 . . . 1330
13.12.14 PostGIS Funktioner nya eller förbättrade i 2.0 . . . 1336
13.12.15 PostGIS Funktioner nya eller förbättrade i 1.5 . . . 1347
13.12.16 PostGIS Funktioner nya eller förbättrade i 1.4 . . . 1348
13.12.17 PostGIS Funktioner nya eller förbättrade i 1.3 . . . 1349

14 Rapportering av problem 1350
14.1 Rapportering av programvarubuggar . . . 1350
14.2 Rapportering av dokumentationsproblem . . . 1350

15 Credits and Acknowledgments 1352
15.1 Project Steering Committee . . . 1352
15.2 Core Contributors Present . . . 1352
15.3 Core Contributors Past . . . 1353
15.4 PostGIS Raster Origins . . . 1353
15.5 Other Contributors . . . 1354

A Anteckningar för release 1358
A.1 PostGIS 3.7.0 . . . 1358
A.1.1 Förändringar . . . 1358
A.1.2 Nya funktioner . . . 1358
A.1.3 Enhancements . . . 1359

Sammanfattning

PostGIS är en utökning till PostgreSQL-objektrelationsdatabassystemet som gör att GIS-objekt (geografiska informationssystem) kan lagras i databasen. PostGIS inkluderar stöd för GiST-baserade R-Tree spatiala index och funktioner för analys och bearbetning av GIS-objekt.

Detta är manualen för version 3.7.0beta2

Detta arbete är licensierat under en Creative Commons Attribution-Share Alike 3.0-licens. Du är välkommen att använda detta material på vilket sätt du vill, men vi ber dig att ange PostGIS Project som källa och om möjligt en länk tillbaka till https://postgis.net..

Kapitel 1

Introduktion

1.1 What Is PostGIS?

PostGIS extends PostgreSQL so that geospatial data can be stored, indexed, queried, and processed in the database. It keeps location data beside ordinary relational data, so a spatial query can use the same SQL, transactions, constraints, access controls, backup tools, and application connections as the rest of a PostgreSQL database.

The main `postgis` extension adds:

- `geometry` for planar coordinate systems and `geography` for geodetic coordinates on the earth.
- Spatial indexes that let the PostgreSQL planner avoid scanning every row when objects can be filtered by location.
- Functions for measurement, relationships, overlays, geometry construction and editing, coordinate transformation, clustering, and spatial data exchange.
- Support for common geospatial encodings and formats, including WKT, WKB, GeoJSON, GML, KML, FlatGeobuf, Geobuf, and Mapbox Vector Tiles.

This makes PostgreSQL a spatial database: applications can ask both ordinary questions and location-aware questions without moving data to a separate GIS processing system. Desktop GIS, map servers, ETL tools, and application frameworks can connect through normal PostgreSQL interfaces.

1.2 What Is in the PostGIS Distribution?

This manual documents the main PostGIS source distribution. Depending on how PostGIS was built and packaged, it can provide several PostgreSQL extensions and command-line tools:

`postgis` The core geometry and geography types, spatial indexes, and most spatial functions.

`postgis_raster` Raster storage, analysis, loading, and export. Raster support uses GDAL and can be omitted from a build.

`postgis_topology` SQL/MM topology types and functions for models based on shared nodes, edges, and faces.

`postgis_sfcgal` Additional 2D and 3D processing functions backed by the optional SFCGAL library.

Loaders and utilities **shp2pgsql**, **pgsql2shp**, and **raster2pgsql** move data between common file formats and PostGIS. Other utilities support upgrades, profiling, and development.

The source tree also contains internal libraries such as `liblwgeom` and `libpgcommon`. They are implementation layers shared by PostGIS components, not separately released PostgreSQL extensions. Most users start by enabling the core extension with `CREATE EXTENSION postgis`; see Kapitel 2 for available components and build options.

1.3 The PostGIS Project Family

This manual covers the main `postgis/postgis` distribution. The official PostGIS source organization also hosts or mirrors complementary projects. These projects have their own release cycles, installation instructions, and documentation; installing the core extension does not install them.

- Address Standardizer parses and normalizes postal addresses. It moved from the main PostGIS source tree to a separate repository for the PostGIS 3.7 development cycle.
- PostGIS TIGER Geocoder provides United States address geocoding and reverse geocoding using Census TIGER data. It is also maintained separately from the main distribution.
- H3-pg provides PostgreSQL bindings for the H3 hierarchical hexagonal indexing system and interoperates with PostGIS.
- PostGIS Java provides Java bindings for PostGIS types.
- Introduction to PostGIS is the hands-on workshop maintained by the PostGIS project.

Other PostgreSQL spatial extensions can complement PostGIS while remaining independent projects. For example, pgRouting adds routing and graph analysis. Use each project's own manual for its functions and support policy.

1.4 How to Use This Manual

The manual is both a guide to the PostGIS data model and the complete reference for the SQL interface. Choose a starting point based on what you are trying to do:

If you are new to spatial SQL Start with the Introduction to PostGIS workshop for a tutorial, then use Kapitel 4 to understand spatial types, coordinate systems, validity, loading, and indexes.

If you are installing or upgrading PostGIS Read Kapitel 2 and Kapitel 3. Package-specific instructions and supported release information are maintained on the PostGIS website.

If you are writing spatial queries Read Kapitel 5 for spatial relationships and index-aware query patterns, and Kapitel 6 for common performance problems.

If you are connecting a desktop GIS Use the client's current database documentation for its user interface and supported geometry types. For example, the QGIS DB Manager lesson covers PostgreSQL connections, table layers, SQL query layers, import, and export. A query layer should return a stable unique identifier and a geometry column with a known SRID and geometry type. Expensive read-only queries can be exposed through a view or materialized view and indexed as appropriate.

If you need a particular function Use Kapitel 7 or Kapitel 13. Function pages state accepted types, dimensional support, standards conformance, dependency requirements, and version availability.

If you use an optional component See Kapitel 9, Kapitel 10, Kapitel 11, or Kapitel 8.

If something goes wrong Collect the versions reported by PostGIS_Full_Version and follow Kapitel 14. Check Appendix A when behavior differs between release lines.

1.5 Standards Support

This manual marks individual functions as compliant with spatial standards so users can tell which behavior is portable and which behavior is a PostGIS extension. PostGIS implements the OGC Simple Features for SQL model, including the geometry type, Well-Known Text (WKT), Well-Known Binary (WKB), and the common spatial accessors, constructors, predicates, and processing functions. PostGIS also implements many SQL/MM Spatial types and functions.

PostGIS supports geometry families defined by later standards, including SQL/MM curved geometry and OGC Simple Features Access 1.2 PolyhedralSurface, Triangle, and TIN. Support for these geometry families is function-specific, and each function page documents the supported input types and conformance level.

PostGIS also includes widely used extensions and integrations. Examples of PostGIS extensions include Extended WKT (EWKT), Extended WKB (EWKB), SRID metadata in those extended formats, geography, and raster. PostGIS also supports input and output exchange formats defined by other standards, such as GeoJSON, GML, and KML. These features are documented individually and should not be read as OGC Simple Features or SQL/MM requirements unless the specific function or format says so.

When the implementation intentionally differs from a standard, or when a standard leaves behavior ambiguous, the relevant function page describes the PostGIS behavior. Kapitel 13 summarizes SQL/MM compliance and geometry-type support across functions.

1.6 Project and Community

PostGIS began in 2001 as a spatial database research project at Refractions Research. It is now an OSGeo project developed by contributors and organizations around the world. The Project Steering Committee coordinates releases, project policy, and the general direction of development. The complete contributor and sponsor record is in Kapitel 15 at the end of this manual.

The manual is the authoritative reference for the released SQL interface. The project website and source tree provide the surrounding material:

- Documentation routes readers to getting-started instructions, versioned manuals, tips, training material, and the FAQ.
- Community lists mailing lists, chat, events, videos, and other support channels.
- Developer documentation covers development environments, testing, coding style, internals, maintenance, governance, releases, and website work.

Kapitel 2

Installation av PostGIS

I detta kapitel beskrivs de steg som krävs för att installera PostGIS.

2.1 Kort version

För att kompilera förutsatt att du har alla beroenden i din sökväg:

Code

```
tar -xvzf postgis-3.7.0beta2.tar.gz
cd postgis-3.7.0beta2
./configure
make
make install
```

När PostGIS har installerats måste det aktiveras (Avsnitt 3.3) eller uppgraderas (Avsnitt 3.4) i varje enskild databas som du vill använda det i.

2.2 Kompilera och installera från källkod

Notera
Många OS-system innehåller nu förbyggda paket för PostgreSQL / PostGIS. I många fall är kompilering endast nödvändig om du vill ha de mest blödande kantversionerna eller om du är en paketunderhållare.
This section includes general compilation instructions. If you are compiling for Windows or another operating system, see the PostGIS getting started guides and the development environment guides.
Pre-Built Packages for various OS are listed in the PostGIS getting started guides
Om du är en Windows-användare kan du få stabila builds via Stackbuilder eller PostGIS Windows nedladdningssida Vi har också mycket blödande avancerade Windows experimentella builds som byggs vanligtvis en eller två gånger i veckan eller när något spännande händer. Du kan använda dessa för att experimentera med de pågående utgåvorna av PostGIS

PostGIS-modulen är en utökning av PostgreSQL-backendservern. Som sådan *kräver* PostGIS & last_release_ full tillgång till PostgreSQL-serverhuvud för att kunna kompilera. Det kan byggas mot PostgreSQL-versioner &min_postgres_version; - &max_postgres_version;. Tidigare versioner av PostgreSQL stöds *inte*..

Se installationsguiderna för PostgreSQL om du inte redan har installerat PostgreSQL. https://www.postgresql.org .

Notera
För GEOS-funktionalitet, när du installerar PostgreSQL kan du behöva länka PostgreSQL uttryckligen mot standard C ++ -biblioteket:

Code

```
LDFLAGS=-lstdc++ ./configure [YOUR OPTIONS HERE]
```

Detta är en lösning för felaktiga C++-undantag i äldre utvecklingsverktyg. Om du upplever konstiga problem (backend oväntat stängd eller liknande saker) kan du prova det här tricket. Detta kommer naturligtvis att kräva omkompilering av din PostgreSQL från grunden.

Följande steg beskriver konfiguration och kompilering av PostGIS-källan. De är skrivna för Linux-användare och kommer inte att fungera på Windows eller Mac.

2.2.1 Hämta källan

Hämta källarkivet för PostGIS från nedladdningswebbplatsen https://download.osgeo.org/postgis/source/postgis-3.7.0beta2.tar.gz

Code

```
wget https://download.osgeo.org/postgis/source/postgis-3.7.0beta2.tar.gz
tar -xvzf postgis-3.7.0beta2.tar.gz
cd postgis-3.7.0beta2
```

Detta kommer att skapa en katalog som heter `postgis-&last_release_version;` i den aktuella arbetskatalogen.

Alternativt kan du hämta källkoden från git-förrådet https://git.osgeo.org/gitea/postgis/postgis/ .

Code

```
git clone https://git.osgeo.org/gitea/postgis/postgis.git postgis
cd postgis
sh autogen.sh
```

Byt till den nyskapade `postgis-katalogen` för att fortsätta installationen.

Code

```
./configure
```

2.2.2 Krav för installation

PostGIS har följande krav för uppbyggnad och användning:

Obligatorisk

- PostgreSQL 14 - 18. En fullständig installation av PostgreSQL (inklusive serverhuvuden) krävs. PostgreSQL finns tillgängligt på https://www.postgresql.org 18 .
  For current release-line support and end-of-life policy, see https://postgis.net/development/versions_eol/-

- GNU C-kompilator (`gcc`). Vissa andra ANSI C-kompilatorer kan användas för att kompilera PostGIS, men vi hittar mycket färre problem när vi kompilerar med `gcc`.
- GNU Make (`gmake` eller `make`). För många system är GNU `make` standardversionen av make. Kontrollera versionen genom att anropa `make -v`. Andra versioner av `make` kanske inte bearbetar PostGIS `Makefile` på rätt sätt.
- Proj reprojektionsbibliotek. Proj 6.1 eller senare krävs. Proj-biblioteket används för att tillhandahålla stöd för koordinatreprojektion inom PostGIS. Proj finns tillgängligt för nedladdning från https://proj.org/ . .
- GEOS geometribibliotek, version 3.8.0 eller högre, men GEOS 3.14+ krävs för att kunna utnyttja alla nya funktioner och egenskaper fullt ut. GEOS finns tillgängligt för hämtning från https://libgeos.org .
- LibXML2, version 2.5.x eller högre. LibXML2 används för närvarande i vissa importfunktioner (ST_GeomF och ST_GeomFromKML). LibXML2 finns att ladda ner från https://gitlab.gnome.org/GNOME/libxml2/--/releases..
- JSON-C, version 0.9 eller högre. JSON-C används för närvarande för att importera GeoJSON via funktionen ST_GeomFromGeoJson. JSON-C finns tillgängligt för nedladdning från https://github.com/-json-c/json-c/releases/.
- The GNU Multiple Precision Arithmetic Library (libgmp). Used to control precision of math operations in PostGIS Topology. GMP is available for download from https://gmplib.org/.
- GDAL, version 3+ är att föredra. Detta krävs för rasterstöd. https://gdal.org/download.html.
- Om du kompilerar med PostgreSQL + JIT krävs LLVM-version >=6 https://trac.osgeo.org/postgis/-ticket/4125.

Valfritt

- GDAL (pseudo-valfritt) endast om du inte vill ha raster kan du utelämna det. Se också till att aktivera de drivrutiner som du vill använda enligt beskrivningen i Avsnitt 3.2.
- GTK (kräver GTK+2.0, 2.8+) för att kompilera shp2pgsql-gui shape-filinläsaren. http://www. gtk.org/ .
- SFCGAL, 1.4.1 or higher is required, 2.3+ is needed to be able to use all functionality. SFCGAL can be used to provide additional 2D and 3D advanced analysis functions to PostGIS cf Kapitel 8. https://sfcgal.gitlab.io/sfcgal/.
- För att aktivera ST_AsMVT krävs protobuf-c-biblioteket 1.1.0 eller högre (för användning) och protoc-c-kompilatorn (för byggande). Dessutom krävs pkg-config för att verifiera den korrekta minimiversionen av protobuf-c. Se protobuf-c. Som standard kommer Postgis att använda Wagyu för att validera MVT-polygoner snabbare, vilket kräver en c++11-kompilator. Den kommer att använda CXXFLAGS och samma kompilator som PostgreSQL-installationen. För att inaktivera detta och använda GEOS istället använder du `--without-wagyu` under konfigurationssteget.
- CUnit (`CUnit`). Detta behövs för regressionstestning. http://cunit.sourceforge.net/
- DocBook (`xsltproc`) krävs för att bygga upp dokumentationen. Docbook finns tillgängligt från http://www. .
- DBLatex (`dblatex`) krävs för att skapa dokumentationen i PDF-format. DBLatex är tillgängligt från http://dblatex.sourceforge.net/ .
- PDF documentation needs an SVG rasterizer for its high-resolution figure fallbacks. The build prefers `rsvg-convert`, and can also use GraphicsMagick (`gm convert`) or ImageMagick (`magick` or `convert`).

2.2.3 Bygg konfiguration

Som med de flesta Linux-installationer är det första steget att generera den Makefile som ska användas för att bygga källkoden. Detta görs genom att köra skalskriptet

Code

```
./configure
```

Utan ytterligare parametrar försöker detta kommando att automatiskt hitta de komponenter och bibliotek som behövs för att bygga PostGIS-källkoden på ditt system. Även om detta är den vanligaste användningen av **./configure**, accepterar skriptet flera parametrar för dem som har de nödvändiga biblioteken och programmen på icke-standardiserade platser.

I följande lista visas endast de vanligaste parametrarna. För en fullständig lista, använd parametrarna **--help** eller **--help=short**.

--with-library-minor-version Från och med PostGIS 3.0 kommer de biblioteksfiler som genereras som standard inte längre att ha den mindre versionen som en del av filnamnet. Detta innebär att alla PostGIS 3-bibliotek kommer att sluta på `postgis-3`. Detta gjordes för att göra pg_upgrade enklare, med nackdelen att du bara kan installera en version av PostGIS 3-serien på din server. För att få det gamla beteendet för filen inklusive den mindre versionen: t.ex. `postgis-3.0` lägg till denna switch i din configure-sats.

--prefix=PREFIX This is the installation prefix for PostGIS command-line utilities. By default, utilities are installed in the executable directory reported by the detected PostgreSQL installation.

Observera
PostgreSQL extension files are installed in the directories reported by **pg_config**, so the server can load them. This includes the PostGIS shared library and SQL extension files. Use `--with-pgconfig=FILE` to choose the PostgreSQL installation that PostGIS builds and installs against.

--with-pgconfig=FILE PostgreSQL tillhandahåller ett verktyg som heter **pg_config** för att göra det möjligt för tillägg som PostGIS att hitta PostgreSQL-installationskatalogen. Använd den här parametern**(--with-pgconfig=/path/to/pg_config**) för att manuellt ange en viss PostgreSQL-installation som PostGIS kommer att bygga mot.

--with-gdalconfig=FILE GDAL, ett obligatoriskt bibliotek, tillhandahåller funktionalitet som behövs för rasterstöd **gdal-config** för att göra det möjligt för programvaruinstallationer att hitta GDAL-installationskatalogen. Använd denna parameter**(--with-gdalconfig=/path/to/gdal-config**) för att manuellt ange en viss GDAL-installation som PostGIS ska bygga mot.

--with-geosconfig=FILE GEOS, ett nödvändigt geometribibliotek, tillhandahåller ett verktyg som heter **geos-config** för att göra det möjligt för programvaruinstallationer att hitta GEOS installationskatalog. Använd denna parameter**(--with-geosconfig=/path/to/geos-config**) för att manuellt ange en viss GEOS-installation som PostGIS ska bygga mot.

--with-xml2config=FILE LibXML är det bibliotek som krävs för att göra GeomFromKML/GML-processer. Det hittas normalt om du har libxml installerat, men om inte eller om du vill att en specifik version ska användas, måste du peka PostGIS mot en specifik `xml2-config` confi-fil för att göra det möjligt för programvaruinstallationer att hitta LibXML-installationskatalogen. Använd denna parameter**(>--with-xml2config=/path/to/xml2-config**) för att manuellt ange en viss LibXML-installation som PostGIS ska bygga mot.

--with-projdir=DIR Proj är ett reprojektionsbibliotek som krävs av PostGIS. Använd denna parameter**(--with-projdir=/path/to/projdir**) för att manuellt ange en viss Proj-installationskatalog som PostGIS ska bygga mot.

--with-libiconv=DIR Katalog där iconv är installerat.

--with-jsondir=DIR JSON-C är ett MIT-licensierat JSON-bibliotek som krävs för PostGIS ST_GeomFromJSO stöd. Använd denna parameter**(--with-jsondir=/path/to/jsondir**) för att manuellt ange en viss JSON-C-installationskatalog som PostGIS kommer att bygga mot.

--with-gui Kompilera GUI för dataimport (kräver GTK+2.0). Detta kommer att skapa shp2pgsql-gui grafiskt gränssnitt till shp2pgsql.

--without-raster Kompilera utan stöd för raster.

--without-topology Kompilera utan topologistöd.

--with-gettext=no Som standard kommer PostGIS att försöka upptäcka stöd för gettext och kompilera med det, men om du stöter på inkompatibilitetsproblem som orsakar att laddaren bryts kan du inaktivera det helt med det här kommandot. Se biljett http://trac.osgeo.org/postgis/ticket/748 för ett exempel på ett problem som löstes genom att konfigurera med detta. OBS: att du inte missar mycket genom att stänga av detta. Detta används för internationell hjälp/etikettstöd för GUI-laddaren som ännu inte är dokumenterad och fortfarande experimentell.

--with-sfcgal=PATH Som standard kommer PostGIS inte att installeras med sfcgal-stöd utan denna switch. PATH är ett valfritt argument som gör det möjligt att ange en alternativ PATH till sfcgal-config.

--without-phony-revision Inaktivera uppdatering av postgis_revision.h för att matcha aktuell HEAD i git-arkivet.

Notera
If you obtained PostGIS from the code repository , the first step is really to run the script

---

Code

```
./autogen.sh
```

Detta skript kommer att generera **configure-skriptet** som i sin tur används för att anpassa installationen av PostGIS.

Om du istället fick PostGIS som en tarball är det inte nödvändigt att köra **./autogen.sh** eftersom **configure** redan har genererats.

2.2.4 Byggnad

När Makefile har genererats är det enkelt att bygga PostGIS genom att köra

Code

```
make
```

Output

```
[last line of output]
PostGIS was built successfully. Ready to install.
```

All the functions have comments generated from the documentation. If you wish to install these comments into your spatial databases later, run the command which requires docbook. The postgis_comments.sql and other package comment files are also packaged in the tar.gz distribution in the doc folder so no need to make comments if installing from the tar ball. Comments are also included as part of the CREATE EXTENSION install.

Code

```
make comments
```

The **make cheatsheets** target generates html cheat sheets suitable for quick reference or for student handouts. This requires xsltproc to build and will generate 4 files in doc folder `topology_cheatsheet.html`, `raster_cheatsheet.html`, `postgis_cheatsheet.html`

Du kan ladda ner några förbyggda tillgängliga i html och pdf från PostGIS / PostgreSQL Study Guides

Code

```
make cheatsheets
```

2.2.5 Bygga PostGIS-tillägg och distribuera dem

The PostGIS extensions are built and installed automatically when PostgreSQL extension support is available.

Om du bygger från källkatalogen måste du först bygga funktionsbeskrivningarna. Dessa byggs om du har docbook installerat. Du kan också bygga manuellt med uttalandet:

Code

```
make comments
```

Det är inte nödvändigt att bygga kommentarerna om du bygger från en release-tarball eftersom dessa redan är färdigbyggda med tarballen.

Tilläggen bör automatiskt byggas som en del av make-installationsprocessen. Du kan vid behov bygga från tilläggsmapparna eller kopiera filer om du behöver dem på en annan server.

Set `PGUSER` if you need to override the psql connection user, then test before installing. After installation, run the extension regression tests.

Code

```
cd extensions
cd postgis
make clean
make
export PGUSER=postgres
make check
make install
make check RUNTESTFLAGS=--extension
```

Notera
`make check` använder psql för att köra tester och kan därför använda psql-miljövariabler. Vanliga variabler som är användbara att åsidosätta är `PGUSER`,`PGPORT` och `PGHOST`. Hänvisa till psql-miljövariabler

Tilläggsfilerna kommer alltid att vara desamma för samma version av PostGIS och PostgreSQL oavsett operativsystem, så det går bra att kopiera över tilläggsfilerna från ett operativsystem till ett annat så länge du har PostGIS-binärerna redan installerade på dina servrar.

Om du vill installera tillägg manuellt på en separat server som skiljer sig från din utveckling måste du kopiera följande filer från tilläggsmappen till mappen PostgreSQL `/ share / extension` i din PostgreSQL-installation samt de nödvändiga binärerna för vanlig PostGIS om du inte redan har dem på servern.

- Det här är kontrollfilerna som anger information som t.ex. vilken version av tillägget som ska installeras om det inte anges. `postgis.control, postgis_topology`.control.
- Alla filer i mappen /sql för varje tillägg. Observera att dessa måste kopieras till roten till PostgreSQL share/extension-mappen `extensions/postgis/sql/*.sql`, `extensions/postgis_topology/sql/*.sql`

När du har gjort det bör du se `postgis`, `postgis_topology` som tillgängliga tillägg i PgAdmin -> tillägg.

Om du använder psql kan du kontrollera att tilläggen är installerade genom att köra den här frågan:

Code

```
SELECT name, default_version, installed_version
FROM pg_available_extensions
WHERE name LIKE 'postgis%'
ORDER BY name;
```

Output

```
+------------------+-----------------+-------------------+
| name             | default_version | installed_version |
+------------------+-----------------+-------------------+
| postgis          | 3.7.0beta2      | 3.7.0beta2        |
| postgis_raster   | 3.7.0beta2      | 3.7.0beta2        |
| postgis_sfcgal   | 3.7.0beta2      |                   |
| postgis_topology | 3.7.0beta2      |                   |
+------------------+-----------------+-------------------+
```

Om du har tillägget installerat i den databas du frågar kommer du att se ett omnämnande i kolumnen `installed_version`. Om du inte får några poster tillbaka betyder det att du inte har några postgis-tillägg installerade på servern alls. PgAdmin III 1.14+ kommer också att tillhandahålla denna information i `tilläggsavsnittet` i databasbläddrarträdet och kommer till och med att tillåta uppgradering eller avinstallation genom att högerklicka.

Om du har tilläggen tillgängliga kan du installera postgis-tillägget i din valfria databas genom att antingen använda pgAdmin-tilläggsgränssnittet eller köra dessa sql-kommandon:

Code

```
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION postgis_sfcgal;
CREATE EXTENSION postgis_topology;
```

I psql kan du använda för att se vilka versioner du har installerat och även vilket schema de är installerade.

Code

```
\connect mygisdb
\x
\dx postgis*
```

Output

```
List of installed extensions
+-[ RECORD 1 ]-------------------------------------------+
| Name        | postgis                                  |
| Version     | 3.7.0beta2                                   |
| Schema      | public                                   |
| Description | PostGIS geometry, geography, and raster...     |
+-[ RECORD 2 ]-------------------------------------------+
| Name        | postgis_raster                           |
| Version     | 3.7.0beta2                                   |
| Schema      | public                                   |
```

```
| Description | PostGIS raster types and functions            |
+-[ RECORD 3 ]-----------------------------------------------+
| Name        | postgis_topology                              |
| Version     | 3.7.0beta2                                        |
| Schema      | topology                                      |
| Description | PostGIS topology spatial types and functions  |
+-----------------------------------------------------------+
```

Varning
Extension tables `spatial_ref_sys`, `layer`, `topology` can not be explicitly backed up. They can only be backed up when the respective `postgis` or `postgis_topology` extension is backed up, which only happens when you back up the whole database. Only srid records not packaged with PostGIS are captured in backups, so don't change the entries we ship and expect the modifications to persist. Put in a ticket if you find an issue. The structures of extension tables are never backed up since they are created with `CREATE EXTENSION` and assumed to be the same for a given version of an extension. These behaviors are built into the current PostgreSQL extension model.

Om du installerade 3.7.0beta2, utan att använda vårt underbara utökningssystem, kan du ändra det till att vara utökningsbaserat genom att köra kommandona nedan för att paketera funktionerna i deras respektive utökning. Installation med `unpackaged` togs bort i PostgreSQL 13, så du rekommenderas att byta till en utökningsbyggnad innan du uppgraderar till PostgreSQL 13.

Code

```
CREATE EXTENSION postgis FROM unpackaged;
CREATE EXTENSION postgis_raster FROM unpackaged;
CREATE EXTENSION postgis_topology FROM unpackaged;
```

2.2.6 Testar

Om du vill testa PostGIS-bygget kör du

Code

```
make check
```

Ovanstående kommando kommer att köra igenom olika kontroller och regressionstester med hjälp av det genererade biblioteket mot en faktisk PostgreSQL-databas.

Notera
Om du konfigurerade PostGIS med icke-standardiserade PostgreSQL-, GEOS- eller Proj-platser kan du behöva lägga till deras bibliotekplatser i miljövariabeln `LD_LIBRARY_PATH`.

Observera
För närvarande förlitar sig **make-kontrollen** på `PATH`- och `PGPORT`-`miljövariablerna` när de utför kontrollerna - den använder *inte* PostgreSQL-versionen som kan ha angetts med hjälp av konfigurationsparametern **--with-pgconfig**. Så se till att ändra din PATH så att den matchar den upptäckta PostgreSQL-installationen under konfigurationen eller var beredd att hantera de överhängande huvudvärkarna.

Notera
Sandboxed build accounts that are not PostgreSQL superusers can delegate database ownership during the regression cycle by exporting POSTGIS_REGRESS_DB_OWNER. The harness will create the temporary regression database owned by the nominated role while continuing to connect using the less privileged account. Combine this with POSTGIS_REGRESS_ROLE_EXT_CREATOR when the extension creation role must differ from the database owner.
These variables allow automated environments to exercise the full upgrade and extension install paths without promoting the calling account to superuser, provided the target PostgreSQL instance permits extension installation by those delegate roles.

Om make check lyckas kommer resultatet av nästan 500 tester att visas. Resultaten kommer att se ut ungefär som följande (många rader utelämnade nedan):

Code

```
CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/

  .
  .
  .

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites     44     44    n/a      0        0
               tests    300    300    300      0        0
             asserts   4215   4215   4215      0      n/a
Elapsed time =    0.229 seconds

  .
  .
  .

Running tests

  .
  .
  .

Run tests: 134
Failed: 0


-- if you build with SFCGAL

  .
  .
  .

Running tests

  .
  .
  .

Run tests: 13
Failed: 0

-- if you built with raster support
```

```
  .
  .
  .

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites     12     12    n/a      0        0
               tests     65     65     65      0        0
             asserts  45896  45896  45896      0      n/a


  .
  .
  .

Running tests

  .
  .
  .

Run tests: 101
Failed: 0

-- topology regress

.
.
.

Running tests

  .
  .
  .

Run tests: 51
Failed: 0

-- if you built --with-gui, you should see this too

     CUnit - A unit testing framework for C - Version 2.1-2
     http://cunit.sourceforge.net/

  .
  .
  .

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      2      2    n/a      0        0
               tests      4      4      4      0        0
             asserts      4      4      4      0      n/a
```

utdata bör se ut som:

Output

```
============== dropping database "contrib_regression" ==============
DROP DATABASE
============== creating database "contrib_regression" ==============
CREATE DATABASE
ALTER DATABASE
============== installing fuzzystrmatch               ==============
CREATE EXTENSION
============== installing postgis                     ==============
```

```
CREATE EXTENSION
============== running regression test queries        ==============
test test-normalize_address   ... ok

====================
All 2 tests passed.
====================
```

2.2.7 Installation

För att installera PostGIS skriver du

Code

```
make install
```

Detta kommer att kopiera PostGIS-installationsfilerna till lämplig underkatalog som anges av konfigurationsparametern **--prefix**. I synnerhet:

- Binärfilerna för laddaren och dumpern installeras i `[prefix]/bin`.
- SQL-filerna, till exempel `postgis.sql`, installeras i `[prefix]/share/contrib`.
- PostGIS-biblioteken är installerade i `[prefix]/lib`.

Om du tidigare körde kommandot **make comments** för att generera filen `postgis_comments.sql`, `raster_comments.sq` l, installerar du sql-filen genom att köra

Code

```
make comments-install
```

Notera
`postgis_comments.sql`, `raster_comments.sql`, `topology_comments.sql` separerades från de typiska bygg- och installationsmålen eftersom det medför det extra beroendet av **xsltproc**.

2.3 Vanliga problem under installationen

Det finns flera saker att kontrollera när installationen eller uppgraderingen inte går som du förväntat dig.

PostgreSQL version mismatch.

Kontrollera att du har installerat PostgreSQL 14 eller nyare, och att du kompilerar mot samma version av PostgreSQL-källan som den version av PostgreSQL som körs. Mix-ups kan uppstå när din (Linux) distribution redan har installerat PostgreSQL, eller om du annars har installerat PostgreSQL tidigare och glömt bort det. PostGIS fungerar bara med PostgreSQL 14 eller nyare, och konstiga, oväntade felmeddelanden kommer att resultera om du använder en äldre version. För att kontrollera vilken version av PostgreSQL som körs, anslut till databasen med psql och kör den här frågan:

Code

```
SELECT version();
```

If you are running an RPM based distribution, you can check for the existence of pre-installed packages using the **rpm** command as follows:

Code

```
rpm -qa | grep postgresql
```

Restoring an upgrade into a database without PostGIS.

Om uppgraderingen misslyckas, se till att du återställer till en databas som redan har PostGIS installerat.

Code

```
SELECT PostGIS_Full_Version();
```

Kontrollera också att configure korrekt har upptäckt platsen och versionen av PostgreSQL, Proj-biblioteket och GEOS-biblioteket.

Utdata från configure används för att generera filen `postgis_config.h`. Kontrollera att variablerna `POSTGIS_PGSQL_VERSION`, `POSTGIS_PROJ_VERSION` och `POSTGIS_GEOS_VERSION` har ställts in korrekt.

Kapitel 3

PostGIS-administration

3.1 Prestandajustering

Inställning för PostGIS-prestanda är ungefär som att ställa in för någon PostgreSQL-arbetsbelastning. Det enda ytterligare övervägandet är att geometrier och raster vanligtvis är stora, så minnesrelaterade optimeringar har i allmänhet större inverkan på PostGIS än andra typer av PostgreSQL-frågor.

För allmänna detaljer om optimering av PostgreSQL, se Tuning your PostgreSQL Server.

PostgreSQL configuration can be set at the server level without touching `postgresql.conf` or `postgresql.a` by using the `ALTER SYSTEM` command.

Set `work_mem`, reload the configuration so new connections use the change, and display the current value. Use **SHOW ALL** to list all settings.

Code

```
ALTER SYSTEM SET work_mem = '256MB';
SELECT pg_reload_conf();
SHOW work_mem;
```

Förutom Postgres-inställningarna har PostGIS några anpassade inställningar som listas i Avsnitt 7.22.

3.1.1 Uppstart

Dessa inställningar konfigureras i `postgresql.conf`:

constraint_exclusion

- Standard: partition
- This is generally used for table partitioning. The default of partitionforces the planner to only analyze tables for constraint consideration if they are in an inherited hierarchy and avoids paying the planner penalty otherwise.

shared_buffers

- Default: ~128MB
- Ställ in på ca 25% to 40% of tillgängligt RAM-minne. På Windows kanske du inte kan ställa in så högt.

max_worker_processes controls the maximum number of background processes available to the server and therefore also bounds how many workers can participate in parallel queries.

- Standard: 8
- Ställer in det maximala antalet bakgrundsprocesser som systemet kan stödja. Denna parameter kan endast ställas in vid serverstart.

3.1.2 Körtid

work_mem - anger storleken på det minne som används för sorteringsoperationer och komplexa frågor

- Standard: 1-4MB
- Justera upp för stora databaser, komplexa frågor, mycket RAM
- Justera ner för många samtidiga användare eller lågt RAM-minne.
- Om du har mycket RAM-minne och få utvecklare:

Code

```
SET work_mem TO '256MB';
```

maintenance_work_mem - den minnesstorlek som används för VACUUM, CREATE INDEX etc.

- Standard: 16-64 MB
- Generellt för låg - binder upp I/O, låser objekt medan minne byts ut
- Rekommenderar 32 MB till 1 GB på produktionsservrar med mycket RAM, men det beror på antalet samtidiga användare. Om du har mycket RAM och få utvecklare:

Code

```
SET maintenance_work_mem TO '1GB';
```

max_parallel_workers_per_gather

Parallel query support in modern PostgreSQL and PostGIS uses this setting to decide how many workers a plan may request. If set to higher than 0, then some queries such as those involving relation functions like `ST_Intersects` can use multiple processes and can run more than twice as fast when doing so. If you have a lot of processors to spare, you should change the value of this to as many processors as you have. Also make sure to bump up `max_worker_processes` to at least as high as this number.

- Standard: 0
- Ställer in det maximala antalet arbetare som kan startas av en enda `Gather-nod`. Parallella arbetare hämtas från den pool av processer som fastställs av `max_worker_processes`. Observera att det begärda antalet arbetare kanske inte är tillgängligt vid körningstillfället. Om detta inträffar kommer planen att köras med färre arbetare än förväntat, vilket kan vara ineffektivt. Om detta värde sätts till 0, vilket är standard, inaktiveras parallell frågeexekvering.

3.2 Konfigurera stöd för raster

Om du har aktiverat rasterstöd kan du läsa nedan om hur du konfigurerar det på rätt sätt.

Out-of-db rasters and all raster drivers are disabled by default. In order to re-enable these, you need to set the following environment variables `POSTGIS_GDAL_ENABLED_DRIVERS` and `POSTGIS_ENABLE_OUTDB_RAST` in the server environment. You can also use the more cross-platform approach of setting the corresponding Avsnitt 7.22.

Om du vill aktivera offline-raster:

Code

```
POSTGIS_ENABLE_OUTDB_RASTERS=1
```

Alla andra inställningar eller ingen inställning alls inaktiverar raster utanför databasen.

För att aktivera alla GDAL-drivrutiner som finns tillgängliga i din GDAL-installation, ställ in denna miljövariabel enligt följande

Code

```
POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL
```

Om du endast vill aktivera specifika drivrutiner ställer du in din miljövariabel enligt följande:

Code

```
POSTGIS_GDAL_ENABLED_DRIVERS="GTiff PNG JPEG GIF XYZ"
```

Notera
Om du använder Windows, använd inte citattecken för drivrutinslistan

Setting environment variables varies depending on OS. For PostgreSQL installed on Ubuntu or Debian via apt-postgresql, the preferred way is to edit `/etc/postgresql/MAJOR/CLUSTER/environment` where the placeholders refer to the PostgreSQL major version and cluster name.

I Windows, om du kör som en tjänst, kan du ställa in via systemvariabler som du hittar i Windows 7 genom att högerklicka på Dator->Egenskaper Avancerade systeminställningar eller i utforskaren genom att navigera till `Kontrollpanelen\Alla kontrollpanelsalternativ\System`. Klicka sedan på *Avancerade systeminställningar ->Avancerat->Miljövariabler* och lägg till nya systemvariabler.

När du har ställt in miljövariablerna måste du starta om din PostgreSQL-tjänst för att ändringarna ska träda i kraft.

3.3 Skapa spatiala databaser

3.3.1 Spatial aktivering av databas med hjälp av EXTENSION

If you have compiled and installed the extensions/postgis modules, you can turn a database into a spatial one using the EXTENSION mechanism.

Core postgis-tillägget inkluderar geometri, geografi, spatial_ref_sys och alla funktioner och kommentarer. Raster och topologi är paketerade som ett separat tillägg.

Kör följande SQL-snutt i den databas som du vill aktivera spatialt:

Code

```
CREATE EXTENSION IF NOT EXISTS plpgsql;
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_raster; -- OPTIONAL
CREATE EXTENSION postgis_topology; -- OPTIONAL
```

3.3.2 Spatial aktivering av databas utan att använda EXTENSION (avrådes)

Notera
Detta behövs i allmänhet bara om du inte kan eller inte vill få PostGIS installerat i PostgreSQL-tilläggskatalogen (till exempel under testning, utveckling eller i en begränsad miljö).

Att lägga till PostGIS-objekt och funktionsdefinitioner i din databas görs genom att läsa in de olika sql-filerna som finns i `[prefix]/share/contrib` enligt specifikationerna under byggfasen.

De centrala PostGIS-objekten (geometri- och geograftyper och deras stödfunktioner) finns i skriptet postgis `.sql`. Rasterobjekt finns i skriptet `rtpostgis.sql`. Topologiobjekt finns i skriptet topology `.sql`.

För en fullständig uppsättning EPSG-koordinatsystemdefinitionsidentifierare kan du också läsa in definitionsfilen `spatial_ref_sys.sql` och fylla i tabellen `spatial_ref_sys`. Detta gör att du kan utföra ST_Transform()-operationer på geometrier.

Om du vill lägga till kommentarer till PostGIS-funktionerna kan du hitta dem i skriptet `postgis_comments.sql`. Kommentarer kan visas genom att helt enkelt skriva **\dd [function_name]** från ett **psql-terminalfönster**.

Kör följande Shell-kommandon i din terminal:

Code

```
DB=[yourdatabase]
SCRIPTSDIR=`pg_config --sharedir`/contrib/postgis-3.6/

# Core objects
psql -d ${DB} -f ${SCRIPTSDIR}/postgis.sql
psql -d ${DB} -f ${SCRIPTSDIR}/spatial_ref_sys.sql
psql -d ${DB} -f ${SCRIPTSDIR}/postgis_comments.sql # OPTIONAL

# Raster support (OPTIONAL)
psql -d ${DB} -f ${SCRIPTSDIR}/rtpostgis.sql
psql -d ${DB} -f ${SCRIPTSDIR}/raster_comments.sql # OPTIONAL

# Topology support (OPTIONAL)
psql -d ${DB} -f ${SCRIPTSDIR}/topology.sql
psql -d ${DB} -f ${SCRIPTSDIR}/topology_comments.sql # OPTIONAL
```

3.4 Uppgradering av spatiala databaser

Att uppgradera befintliga spatiala databaser kan vara svårt eftersom det kräver ersättning eller införande av nya PostGIS-objektdefinitioner.

Tyvärr är det inte alla definitioner som enkelt kan ersättas i en levande databas, så ibland är det bästa alternativet en dump/reload-process.

PostGIS tillhandahåller en SOFT UPGRADE-procedur för mindre eller bugfix-utgåvor och en HARD UPGRADE-procedur för större utgåvor.

Innan du försöker uppgradera PostGIS är det alltid värt att säkerhetskopiera dina data. Om du använder flaggan -Fc till pg_dump kommer du alltid att kunna återställa dumpningen vid en hård uppgradering.

3.4.1 Mjuk uppgradering

Om du installerade din databas med hjälp av tillägg måste du också uppgradera med hjälp av tilläggsmodellen. Om du installerade med den gamla sql-skriptmetoden rekommenderas du att byta din installation till tillägg eftersom skriptmetoden inte längre stöds.

3.4.1.1 Soft upgrade using extensions

If you originally installed PostGIS with extensions, then you need to upgrade using extensions as well. Doing a minor upgrade with extensions is fairly painless.

Use the PostGIS_Extensions_Upgrade function to upgrade to the latest version you have installed.

Code

```
SELECT postgis_extensions_upgrade();
```

If you are upgrading a database that still runs PostGIS 2.5, issue an **ALTER EXTENSION** followed by `postgis_extensions_upgrade()` to make sure the legacy `postgis_raster` objects are folded back into the main extension.

Code

```
ALTER EXTENSION postgis UPDATE;
SELECT postgis_extensions_upgrade();
```

This second call is needed to rebundle the `postgis_raster` extension.

Code

```
SELECT postgis_extensions_upgrade();
```

Om du har flera versioner av PostGIS installerade och inte vill uppgradera till den senaste, kan du uttryckligen ange versionen på följande sätt:

Code

```
ALTER EXTENSION postgis UPDATE TO "3.7.0beta2";
ALTER EXTENSION postgis_topology UPDATE TO "3.7.0beta2";
```

Om du får ett felmeddelande något i stil med:

Output

```
No migration path defined for b''…b'' to 3.7.0beta2
```

Sedan måste du säkerhetskopiera din databas, skapa en ny enligt beskrivningen i Avsnitt 3.3.1 och sedan återställa din säkerhetskopia ovanpå den nya databasen.

Om du får ett meddelande som:

Output

```
Version "3.7.0beta2" of extension "postgis" is already installed
```

Då är allt redan uppdaterat och du kan tryggt ignorera det. OM du **inte** försöker uppgradera från en utvecklingsversion till nästa (som inte får ett nytt versionsnummer); i så fall kan du lägga till nexttill versionssträngen, och nästa gång måste du släppa next-suffixet igen:

Code

```
ALTER EXTENSION postgis UPDATE TO "3.7.0beta2next";
        ALTER EXTENSION postgis_topology UPDATE TO "3.7.0beta2next";
```

Notera
Om du installerade PostGIS ursprungligen utan en angiven version kan du ofta hoppa över ominstallationen av postgis-tillägget innan du återställer eftersom säkerhetskopian bara har `CREATE EXTENSION postgis` och därmed plockar upp den senaste versionen under återställningen.

Notera
If you are upgrading PostGIS extension from a version prior to 3.0.0, you will have a new extension *postgis_raster* which you can safely drop, if you don't need raster support.

Code

```
DROP EXTENSION postgis_raster;
```

3.4.1.2 Soft upgrade without extensions

Extensions have been the supported installation method for many releases. If you still rely on an unpackaged installation, perform a HARD UPGRADE into a database created with extensions and migrate your data there.

3.4.1.3 Hård uppgradering

Med HÅRD UPPGRADERING menar vi fullständig dumpning/omladdning av PostGIS-aktiverade databaser. Du behöver en HÅRD UPPGRADERING när PostGIS-objektens interna lagring ändras eller när MJUK UPPGRADERING inte är möjlig. Bilagan Release Notes rapporterar för varje version om du behöver en dumpning/omladdning (HARD UPGRADE) för att uppgradera.

The dump/reload process is assisted by the **postgis_restore** script which takes care of skipping from the dump all definitions which belong to PostGIS (including old ones), allowing you to restore your schemas and data into a database with PostGIS installed without getting duplicate symbol errors or bringing forward deprecated objects.

Supplementary instructions for windows users are available in the PostGIS Windows installation guide.

Förfarandet är som följer:

1. Skapa en dump i änpassat formatäv den databas du vill uppgradera (låt oss kalla den `olddb`) inklusive binära blobs (-b) och verbose (-v) utdata. Användaren kan vara ägaren av db, behöver inte vara postgres superkonto.

Code

```
pg_dump -h localhost -p 5432 -U postgres -Fc -b -v -f "/somepath/olddb.backup" olddb
```

2. Gör en nyinstallation av PostGIS i en ny databas - vi kallar denna databas för `newdb`. Se Avsnitt 3.3.2 och Avsnitt 3.3.1 för instruktioner om hur du gör detta.

   De spatial_ref_sys-poster som finns i din dump kommer att återställas, men de kommer inte att åsidosätta befintliga poster i spatial_ref_sys. Detta är för att säkerställa att korrigeringar i den officiella uppsättningen kommer att spridas korrekt till återställda databaser. Om du av någon anledning verkligen vill ha dina egna åsidosättningar av standardposter, ladda bara inte spatial_ref_sys.sql-filen när du skapar den nya db.

   Om din databas är riktigt gammal eller om du vet att du har använt funktioner som inte längre används i dina vyer och funktioner, kan du behöva läsa in `legacy.sql` för att alla dina funktioner och vyer etc. ska komma tillbaka på rätt sätt. Gör detta endast om det _verkligen_ behövs. Överväg att uppgradera dina vyer och funktioner innan du dumpar dem, om möjligt. De föråldrade funktionerna kan senare tas bort genom att läsa in `uninstall_legacy.sql`.

3. Återställ din säkerhetskopia till din `nya newdb-databas` med postgis_restore. Oväntade fel, om några, kommer att skrivas ut till standardfelströmmen av psql. Förvara en logg över dessa.

   Code

   ```
   postgis_restore "/somepath/olddb.backup" | psql -h localhost -p 5432 -U postgres newdb ↩
       2> errors.txt
   ```

Fel kan uppstå i följande fall:

1. Några av dina vyer eller funktioner använder sig av föråldrade PostGIS-objekt. För att åtgärda detta kan du försöka läsa in `legacy.sql-skriptet` före återställningen eller så måste du återställa till en version av PostGIS som fortfarande innehåller dessa objekt och försöka migrera igen efter att ha portat din kod. Om `legacy.sql` fungerar för dig, glöm inte att fixa din kod så att den slutar använda föråldrade funktioner och ta bort dem genom att läsa in `uninstall_legacy.sql`.

2. Vissa anpassade poster i spatial_ref_sys i dumpfilen har ett ogiltigt SRID-värde. Giltiga SRID-värden är större än 0 och mindre än 999000. Värden i intervallet 999000.999999 är reserverade för internt bruk medan värden över 999999 inte kan användas alls. Alla dina anpassade poster med ogiltiga SRID kommer att behållas, med de > 999999 flyttade till det reserverade intervallet, men tabellen spatial_ref_sys skulle förlora en kontrollbegränsning som skyddar för att invariant ska hålla och eventuellt också dess primära nyckel (när flera ogiltiga SRIDS konverteras till samma reserverade SRID-värde).

   För att åtgärda detta bör du kopiera din anpassade SRS till en SRID med ett giltigt värde (kanske i intervallet 910000..910999), konvertera alla dina tabeller till den nya srid (se UpdateGeometrySRID), ta bort den ogiltiga posten från spatial_ref_sys och konstruera om kontrollen (erna) med:

   Code

   ```
   ALTER TABLE spatial_ref_sys ADD CONSTRAINT spatial_ref_sys_srid_check check (srid
   > 0 AND srid < 999000 );
   ```

   Code

   ```
   ALTER TABLE spatial_ref_sys ADD PRIMARY KEY(srid));
   ```

   Om du uppgraderar en gammal databas som innehåller fransk IGN-kartografi kommer du förmodligen att ha SRID: er utanför intervallet och du kommer att se, när du importerar din databas, problem som detta:

Code

```
WARNING: SRID 310642222 converted to 999175 (in reserved zone)
```

I det här fallet kan du prova följande steg: först kasta ut IGN helt från sql som är resultatet av postgis_restore. Så, efter att ha kört :

Code

```
postgis_restore "/somepath/olddb.backup" > olddb.sql
```

kör detta kommando :

Code

```
grep -v IGNF olddb.sql > olddb-without-IGN.sql
```

Create your newdb, activate the required PostGIS extensions, and insert the French IGN spatial reference definitions using this script. After these operations, import your data:

Code

```
psql -h localhost -p 5432 -U postgres -d newdb -f olddb-without-IGN.sql  2> errors.txt
```

Kapitel 4

Datahantering

4.1 Modell för spatial data

4.1.1 OGC Geometri

Open Geospatial Consortium (OGC) har utvecklat *Enkel åtkomst till funktioner* (SFA) för att tillhandahålla en modell för geospatiala data. Den definierar den grundläggande spatiala typen av **geometri**, tillsammans med operationer som manipulerar och transformerar geometrivärden för att utföra spatiala analysuppgifter. PostGIS implementerar OGC Geometry-modellen som PostgreSQL-datatyperna geometri och geografi.

Geometry är en *abstrakt* typ. Geometry-värden tillhör en av dess *konkreta* subtyper som representerar olika typer och dimensioner av geometriska former. Dessa inkluderar **atomtyperna** Point, LineString, LinearRing och Polygon, samt **samlingstyperna** MultiPoint, MultiLineString, MultiPolygon och GeometryCollection. De *Access till enkla funktioner - Del 1: Gemensam arkitektur v1.2.1* lägger till subtyper för strukturerna PolyhedralSurface, Triangle och TIN.

Geometry modellerar former i det 2-dimensionella kartesiska planet. Typerna PolyhedralSurface, Triangle och TIN kan också representera former i ett 3-dimensionellt rum. Formernas storlek och placering anges av deras **koordinater**. Varje koordinat har ett X- och **Y-ordinatvärde** som bestämmer dess läge i planet. Former konstrueras av punkter eller linjesegment, där punkter specificeras av en enda koordinat och linjesegment av två koordinater.

Coordinates may contain optional Z and M ordinate values. The Z ordinate is often used to represent elevation. The M ordinate contains a measure value, which may represent time or distance. If Z or M values are present in a geometry value, they must be defined for each point in the geometry. If a geometry has Z or M ordinates the **coordinate dimension** is 3D; if it has both Z and M the coordinate dimension is 4D. The coordinate dimension is at least 2, because every geometry has at least X and Y coordinates.

Geometrivärden associeras med ett **spatialt referenssystem** som anger det koordinatsystem i vilket det är inbäddat. Det spatiala referenssystemet identifieras av geometrins SRID-nummer. Enheterna för X- och Y-axlarna bestäms av det spatiala referenssystemet. I **plana** referenssystem representerar X- och Y-koordinaterna vanligtvis ost- och nordlig riktning, medan de i **geodetiska** system representerar longitud och latitud. SRID 0 representerar ett oändligt kartesiskt plan utan några enheter tilldelade dess axlar. Se Avsnitt 4.5.

The geometry **dimension** is a property of geometry types. Point types have dimension 0, linear types have dimension 1, polygonal types have dimension 2, and solid types have dimension 3. Collections have the dimension of the maximum element dimension.

Ett geometrivärde kan vara **tomt**. Tomma värden innehåller inga toppar (för atomära geometrityper) eller inga element (för samlingar).

En viktig egenskap hos geometrivärden är deras spatiala **extent** eller **bounding box**, som i OGC-modellen kallas **envelope**. Detta är den 2- eller 3-dimensionella box som omsluter koordinaterna för en geometri. Det är ett effektivt sätt att representera en geometris utbredning i koordinatrymden och att kontrollera om två geometrier interagerar.

The geometry model allows evaluating topological spatial relationships as described in Avsnitt 5.1.1. To support this the concepts of **interior**, **boundary** and **exterior** are defined for each geometry type. Geometries are topologically closed, so they always contain their boundary. The boundary is a geometry of dimension one less than that of the geometry itself. For example, the boundary of a Polygon is its rings (LineStrings), the boundary of a LineString is its endpoints (Points), and the boundary of a Point is empty.

The OGC geometry model defines validity rules for each geometry type. These rules ensure that geometry values represents realistic situations (e.g. it is possible to specify a polygon with a hole lying outside its shell, but this makes no sense geometrically and is thus invalid). PostGIS also allows storing and manipulating invalid geometry values. This allows detecting and fixing them if needed. See Avsnitt 4.4

4.1.1.1 Point

En punkt är en 0-dimensionell geometri som representerar en enda plats i koordinatrymden.

Code

```
POINT (1 2)
POINT Z (1 2 3)
POINT ZM (1 2 3 4)
```

4.1.1.2 LineString

En LineString är en 1-dimensionell linje som består av en sammanhängande sekvens av linjesegment. Varje linjesegment definieras av två punkter, där slutpunkten för ett segment utgör startpunkten för nästa segment. En OGC-valid LineString har antingen noll eller två eller flera punkter, men PostGIS tillåter även LineStrings med en enda punkt. LineStrings kan korsa sig själva (self-intersect). En LineString är **sluten** om start- och slutpunkterna är desamma. En LineString är **enkel** om den inte korsar sig själv.

Code

```
LINESTRING (1 2, 3 4, 5 6)
```

4.1.1.3 LinearRing

En LinearRing är en LineString som är både sluten och enkel. Den första och den sista punkten måste vara lika och linjen får inte skära sig själv.

Code

```
LINEARRING (0 0 0, 4 0 0, 4 4 0, 0 4 0, 0 0 0)
```

4.1.1.4 Polygon

En polygon är en 2-dimensionell plan region som avgränsas av en yttre gräns (skalet) och noll eller flera inre gränser (hål). Varje gräns är en LinearRing.

Code

```
POLYGON ((0 0 0, 4 0 0, 4 4 0, 0 4 0, 0 0 0), (1 1 0, 2 1 0, 2 2 0, 1 2 0, 1 1 0))
```

4.1.1.5 MultiPoint

En MultiPoint är en samling punkter.

Code

```
MULTIPOINT ( (0 0), (1 2) )
```

4.1.1.6 MultiLineString

En MultiLineString är en samling LineStrings. En MultiLineString är sluten om vart och ett av dess element är slutet.

Code

```
MULTILINESTRING ( (0 0, 1 1, 1 2), (2 3, 3 2, 5 4) )
```

4.1.1.7 MultiPolygon

A MultiPolygon is a collection of non-overlapping, non-adjacent Polygons. Polygons in the collection may touch only at a finite number of points. (Two polygons are adjacent if they share an edge. They touch if they share only points or edges on their boundaries). For more details on MultiPolygon validity, see Avsnitt 4.4.

Code

```
MULTIPOLYGON (((1 5, 5 5, 5 1, 1 1, 1 5)), ((6 5, 9 1, 6 1, 6 5)))
```

4.1.1.8 GeometriSamling

En GeometryCollection är en heterogen (blandad) samling av geometrier.

Code

```
GEOMETRYCOLLECTION ( POINT(2 3), LINESTRING(2 3, 3 4))
```

4.1.1.9 PolyhedralSurface

En PolyhedralSurface är en sammanhängande samling av patcher eller facetter som delar vissa kanter. Varje patch är en plan polygon. Om polygonens koordinater har Z-ordinater är ytan 3-dimensionell.

Code

```
POLYHEDRALSURFACE Z (
((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )
```

4.1.1.10 Triangel

En triangel är en polygon som definieras av tre distinkta, icke-kollinjära hörn. Eftersom en triangel är en polygon anges den med fyra koordinater, där den första och den fjärde är lika.

Code

```
TRIANGLE ((0 0, 0 9, 9 0, 0 0))
```

4.1.1.11 TIN

En TIN är en samling icke-överlappande trianglarsom representerar ett triangulerat oregelbundet nätverk.

Code

```
TIN Z ( ((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0)) )
```

4.1.2 SQL/MM Del 3 - Kurvor

ISO/IEC 13249-3 *ISO/IEC 13249-3 SQL Multimedia - Spatial* (SQL/MM) utökar OGC SFA för att definiera Geometry-subtyper som innehåller kurvor med cirkelbågar. SQL/MM-typerna stöder 3DM-, 3DZ- och 4D-koordinater.

Notera
Alla jämförelser med flyttal i SQL-MM-implementationen utförs med en angiven tolerans, för närvarande 1E-8.

4.1.2.1 CircularString

CircularString är den grundläggande kurvtypen, ungefär som LineString i den linjära världen. Ett enkelt bågsegment specificeras av tre punkter: start- och slutpunkterna (första och tredje) samt någon annan punkt på bågen. För att specificera en sluten cirkel är start- och slutpunkterna desamma och mittpunkten är den motsatta punkten på cirkeldiametern (som är bågens mittpunkt). I en sekvens av bågar är slutpunkten för den föregående bågen startpunkten för nästa båge, precis som segmenten i en LineString. Detta innebär att en CircularString måste ha ett udda antal punkter som är större än 1.

Code

```
CIRCULARSTRING(0 0, 1 1, 1 0)

CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0)
```

4.1.2.2 CompoundCurve

En CompoundCurve är en enda kontinuerlig kurva som kan innehålla både cirkelbågssegment och linjära segment. Det innebär att förutom att ha välformade komponenter måste slutpunkten för varje komponent (utom den sista) sammanfalla med startpunkten för följande komponent.

Code

```
COMPOUNDCURVE( CIRCULARSTRING(0 0, 1 1, 1 0), (1 0, 0 1))
```

4.1.2.3 CurvePolygon

En CurvePolygon är som en polygon, med en yttre ring och noll eller flera inre ringar. Skillnaden är att en ring kan vara en CircularString eller CompoundCurve såväl som en LineString.

Från och med PostGIS 1.4 har PostGIS stöd för sammansatta kurvor i en kurvpolygon.

Code

```
CURVEPOLYGON(
CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),
(1 1, 3 3, 3 1, 1 1) )
```

Exempel: En CurvePolygon med skalet definierat av en CompoundCurve som innehåller en CircularString och en LineString, och ett hål definierat av en CircularString

Code

```
CURVEPOLYGON(
COMPOUNDCURVE( CIRCULARSTRING(0 0, 2 0, 2 1, 2 3, 4 3),
               (4 3, 4 5, 1 4, 0 0)),
CIRCULARSTRING(1.7 1, 1.4 0.4, 1.6 0.4, 1.6 0.5, 1.7 1) )
```

4.1.2.4 MultiCurve

En MultiCurve är en samling kurvor som kan innehålla LineStrings, CircularStrings eller CompoundCurves.

Code

```
MULTICURVE( (0 0, 5 5), CIRCULARSTRING(4 0, 4 4, 8 4))
```

4.1.2.5 MultiSurface

En MultiSurface är en samling ytor, som kan vara (linjära) polygoner eller CurvePolygoner.

Code

```
MULTISURFACE(
CURVEPOLYGON(
  CIRCULARSTRING( 0 0, 4 0, 4 4, 0 4, 0 0),
  (1 1, 3 3, 3 1, 1 1)),
((10 10, 14 12, 11 10, 10 10), (11 11, 11.5 11, 11 11.5, 11 11)))
```

4.1.3 WKT och WKB

OGC SFA-specifikationen definierar två format för att representera geometriska värden för extern användning: Well-Known Text (WKT) och Well-Known Binary (WKB). Både WKT och WKB innehåller information om typen av objekt och de koordinater som definierar det.

Well-Known Text (WKT) tillhandahåller en standardiserad textuell representation av spatiala data. Exempel på WKT-representationer av spatiala objekt är:

- PUNKT(0 0)
- PUNKT Z (0 0 0)
- PUNKT ZM (0 0 0 0)
- PUNKT EMPTY
- LINESTRING(0 0,1 1,1 2)
- LINESTRING TOM
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT((0 0),(1 2))
- MULTIPOINT Z ((0 0 0),(1 2 3))
- MULTIPOINT TOM
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1 -1)))
- GEOMETRYCOLLECTION(POINT(2 3), LINESTRING(2 3,3 4))
- GEOMETRYUPPCOLLECTION TOM

In- och utdata för WKT tillhandahålls av funktionerna ST_AsText och ST_GeomFromText:

Code

```
text WKT = geometry;
geometry = ST_GeomFromText(text WKT, SRID);
```

For example, a statement to create and insert a spatial object from WKT with SRID 312 is:

Code

```
INSERT INTO geotable ( geom, name )
VALUES ( ST_GeomFromText('POINT(-126.4 45.32)', 312), 'A Place');
```

Well-Known Binary (WKB) tillhandahåller en portabel representation med full precision av spatiala data som binära data (matriser av bytes). Exempel på WKB-representationer av spatiala objekt är:

- WKT: PUNKT(1 1)
  WKB: 01010000000000000000F03F000000000000F03
- WKT: LINESTRING (2 2, 9 9)
  WKB: 0102000000020000000000000000000040000000000000004000000000000022400000000000002

In- och utdata för WKB tillhandahålls av funktionerna ST_AsBinary och ST_GeomFromWKB:

Code

```
bytea WKB = ST_AsBinary(geometry);
geometry = ST_GeomFromWKB(bytea WKB, SRID);
```

Ett villkor för att skapa och infoga ett spatialt objekt från WKB är t.ex:

Code

```
INSERT INTO geotable ( geom, name )
VALUES ( ST_GeomFromWKB('\x01010000000000000000f03f000000000000f03f', 312), 'A Place');
```

4.2 Datatypen Geometry

PostGIS implementerar OGC Simple Features-modellen genom att definiera en PostgreSQL-datatyp som heter `geometry`. Den representerar alla undertyper av geometri med hjälp av en intern typkod (se GeometryType och ST_GeometryType). Detta gör det möjligt att modellera spatiala funktioner som rader i tabeller som definieras med en kolumn av typen `geometri`.

The `geometry` data type is *opaque*, which means that all access is done via invoking functions on geometry values. Functions allow creating geometry objects, accessing or updating all internal fields, and computing new geometry values. PostGIS supports all the functions specified in the OGC *Simple feature access - Part 2: SQL option* (SFS) specification, as well many others. See Kapitel 7 for the full list of functions.

Notera
PostGIS följer SFA-standarden genom att spatiala funktioner prefixeras med ST_". Detta var tänkt att stå för Spatial and Temporal", men den temporala delen av standarden utvecklades aldrig. Istället kan det tolkas som Spatial Type".

SFA-standarden anger att spatiala objekt ska innehålla en identifierare för spatialt referenssystem (Spatial Reference System identifier, SRID). SRID krävs när spatiala objekt skapas för införande i databasen (den kan vara 0 som standard). Se ST_SRID och Avsnitt 4.5

För att göra geometriska frågor effektiva definierar PostGIS olika typer av spatiala index och spatiala operatorer för att använda dem. Se Avsnitt 4.9 och Avsnitt 5.2 för mer information.

4.2.1 PostGIS EWKB och EWKT

OGC SFA-specifikationerna stödde ursprungligen endast 2D-geometrier, och geometrins SRID ingår inte i inmatnings-/utdatarepresentationerna. OGC SFA-specifikationen 1.2.1 (som överensstämmer med ISO 19125-standarden) lägger till stöd för 3D (ZYZ) och uppmätta (XYM och XYZM) koordinater, men inkluderar fortfarande inte SRID-värdet.

På grund av dessa begränsningar har PostGIS definierat utökade EWKB- och EWKT-format. De ger stöd för koordinater i 3D (XYZ och XYM) och 4D (XYZM) och inkluderar SRID-information. Genom att inkludera all geometriinformation kan PostGIS använda EWKB som format för registrering (t.ex. i DUMP-filer).

EWKB och EWKT används för de "kanoniska formernaför PostGIS dataobjekt. För indata är den kanoniska formen för binära data EWKB, och för textdata accepteras antingen EWKB eller EWKT. Detta gör det möjligt att skapa geometrivärden genom att kasta ett textvärde i antingen HEXEWKB eller EWKT till ett geometrivärde med hjälp av `::geometry`. För utdata är den kanoniska formen för binära data EWKB och för text är den HEXEWKB (hex-kodad EWKB).

Till exempel skapar denna sats en geometri genom casting från ett EWKT-textvärde och matar ut den med den kanoniska formen av HEXEWKB:

Code

```
SELECT 'SRID=4;POINT(0 0)'::geometry;
```

Output

```
geometry
----------------------------------------------------
0101000020040000000000000000000000000000000000000000
```

PostGIS EWKT-utdata har några skillnader jämfört med OGC WKT:

- För 3DZ-geometrier utelämnas Z-kvalificeraren:
  OGC: PUNKT Z (1 2 3)
  EWKT: PUNKT (1 2 3)
- För 3DM-geometrier inkluderas M-kvalificeraren:
  OGC: PUNKT M (1 2 3)
  EWKT: POINTM (1 2 3)
- För 4D-geometrier utelämnas ZM-kvalificeraren:
  OGC: PUNKT ZM (1 2 3 4)
  EWKT: PUNKT (1 2 3 4)

EWKT undviker överspecificering av dimensionalitet och de inkonsekvenser som kan uppstå med OGC/ISO-formatet, t.ex:

- POINT ZM (1 1)
- PUNKT ZM (1 1 1)
- PUNKT (1 1 1 1 1)

Observera
PostGIS utökade format är för närvarande en superset av OGC:s format, så att varje giltig OGC WKB/WKT också är giltig EWKB/EWKT. Detta kan dock variera i framtiden, om OGC utökar ett format på ett sätt som strider mot PosGIS-definitionen. Därför SKA du INTE förlita dig på denna kompatibilitet!

Exempel på EWKT-textrepresentation av spatiala objekt är:

- PUNKT(0 0 0) -- XYZ
- SRID=32632;POINT(0 0) -- XY med SRID
- POINTM(0 0 0) -- XYM
- PUNKT(0 0 0 0) -- XYZM
- SRID=4326;MULTIPOINTM(0 0 0,1 2 1) -- XYM med SRID
- MULTILINESTRING((0 0 0,1 1 0,1 2 1),(2 3 1,3 2 1,5 4 1))
- POLYGON((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0))
- MULTIPOLYGON(((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0)),((-1 -1 0,-1 -2 0,-2 -2 0,-2 -1 0,-1 -1 0)))
- GEOMETRYCOLLECTIONM(PUNKTM(2 3 9), LINESTRINGM(2 3 4, 3 4 5))
- MULTICURVE((0 0, 5 5), CIRCULARSTRING(4 0, 4 4, 8 4))
- POLYHEDRALSURFACE(((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)), ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)), ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))
- TRIANGEL ((0 0, 0 10, 10 0, 0 0))
- TIN(((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0))))

Inmatning och utdata med dessa format kan göras med hjälp av följande funktioner:

Code

```
bytea EWKB = ST_AsEWKB(geometry);
text EWKT = geometry;
geometry = ST_GeomFromEWKB(bytea EWKB);
geometry = ST_GeomFromEWKT(text EWKT);
```

Ett villkor för att skapa och infoga ett spatialt PostGIS-objekt med EWKT är t.ex:

Code

```
INSERT INTO geotable ( geom, name )
VALUES ( ST_GeomFromEWKT('SRID=312;POINTM(-126.4 45.32 15)'), 'A Place' )
```

4.3 Datatypen Geography

Datatypen PostGIS `geography` ger inbyggt stöd för spatiala egenskaper som representeras av geografiska”koordinater (ibland kallade geodetiska”koordinater, eller lat/lon”, eller lon/lat”). Geografiska koordinater är sfäriska koordinater uttryckta i vinkelenheter (grader).

Basen för PostGIS datatyp geometri är ett plan. Den kortaste vägen mellan två punkter på planet är en rät linje. Det innebär att funktioner på geometrier (ytor, avstånd, längder, skärningspunkter etc.) beräknas med hjälp av rätlinjiga vektorer och kartesisk matematik. Detta gör dem enklare att implementera och snabbare att utföra, men gör dem också felaktiga för data på jordens sfäroida yta.

PostGIS datatyp för geografi är baserad på en sfärisk modell. Den kortaste vägen mellan två punkter på sfären är en storcirkelbåge. Funktioner på geografier (områden, avstånd, längder, skärningspunkter etc.) beräknas med hjälp av bågar på sfären. Genom att ta hänsyn till världens sfäroida form ger funktionerna mer exakta resultat.

Eftersom den underliggande matematiken är mer komplicerad finns det färre funktioner definierade för geograftypen än för geometritypen. Med tiden, när nya algoritmer läggs till, kommer funktionerna för geograftypen att utökas. Som en nödlösning kan man konvertera fram och tillbaka mellan geometri- och geograftyperna.

Like the geometry data type, geography data is associated with a spatial reference system via a spatial reference system identifier (SRID). Any geodetic (long/lat based) spatial reference system defined in the `spatial_ref_sys` table can be used. You can add your own custom geodetic spatial reference system as described in Avsnitt 4.5.2.

För alla spatiala referenssystem är de enheter som returneras av mätfunktioner (t.ex. ST_Distance, ST_Length, ST_Perimeter, ST_Area) och för distansargumentet i ST_DWithin i meter.

4.3.1 Skapa geografiska tabeller

Du kan skapa en tabell för att lagra geografiska data genom att använda SQL-satsen CREATE TABLE med en kolumn av typen `geography`. I följande exempel skapas en tabell med en geografikolumn som lagrar 2D LineStrings i det geodetiska koordinatsystemet WGS84 (SRID 4326):

Code

```
CREATE TABLE global_points (
  id SERIAL PRIMARY KEY,
  name VARCHAR(64),
  location geography(POINT, 4326)
);
```

Geografitypen har stöd för två valfria typmodifierare:

- modifieraren spatial type begränsar vilka typer av former och dimensioner som tillåts i kolumnen. Tillåtna värden för den spatiala typen är: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION. Geografitypen stöder inte kurvor, TINS eller POLYHEDRALSURFACEs. Modifieraren stöder begränsningar av koordinatdimensionalitet genom att lägga till suffix: Z, M och ZM. En modifierare av typen LINESTRINGMtillåter t.ex. endast linestrings med tre dimensioner och behandlar den tredje dimensionen som ett mått. På samma sätt kräver POINTZMfyrdimensionella (XYZM) data.
- sRID-modifieraren begränsar det spatiala referenssystemets SRID till ett visst nummer. Om SRID utelämnas är standardvärdet 4326 (WGS84 geodetic) och alla beräkningar utförs med WGS84.

Exempel på hur man skapar tabeller med geografikolumner:

Skapa en tabell med 2D POINT-geografi med standard SRID 4326 (WGS84 long/lat):

Code

```
CREATE TABLE ptgeogwgs(gid serial PRIMARY KEY, geog geography(POINT));
```

Skapa en tabell med 2D POINT-geografi i NAD83 longlat:

Code

```
CREATE TABLE ptgeognad83(gid serial PRIMARY KEY, geog geography(POINT, 4269));
```

Skapa en tabell med 3D (XYZ) POINTs och en explicit SRID på 4326:

Code

```
CREATE TABLE ptzgeogwgs84(gid serial PRIMARY KEY, geog geography(POINTZ, 4326));
```

Skapa en tabell med 2D LINESTRING-geografi med standard SRID 4326:

Code

```
CREATE TABLE lgeog(gid serial PRIMARY KEY, geog geography(LINESTRING));
```

Skapa en tabell med 2D POLYGON-geografi med SRID 4267 (NAD 1927 long lat):

Code

```
CREATE TABLE lgeognad27(gid serial PRIMARY KEY, geog geography(POLYGON, 4267));
```

Geografiska fält registreras i systemvyn `geography_columns`. Du kan ställa en fråga till vyn `geography_colu` och se att tabellen finns med i listan:

Code

```
SELECT * FROM geography_columns;
```

Att skapa ett spatialt index fungerar på samma sätt som för geometrikolumner. PostGIS kommer att notera att kolumntypen är GEOGRAPHY och skapa ett lämpligt sfärbaserat index istället för det vanliga plana indexet som används för GEOMETRY.

Code

```
CREATE INDEX global_points_gix ON global_points USING GIST ( location );
```

4.3.2 Använda geografiska tabeller

Du kan infoga data i geograftabeller på samma sätt som i geometritabeller. Geometridata kommer att autokastas till geograftypen om den har SRID 4326. EWKT- och EWKB-formaten kan också användas för att ange geografiska värden.

Code

```
INSERT INTO global_points (name, location) VALUES ('Town', 'SRID=4326;POINT(-110 30)');
INSERT INTO global_points (name, location) VALUES ('Forest', 'SRID=4326;POINT(-109 29)');
INSERT INTO global_points (name, location) VALUES ('London', 'SRID=4326;POINT(0 49)');
```

Alla geodetiska (lång/lat) spatiala referenssystem som förtecknas i tabellen `spatial_ref_sys` kan anges som ett geografiskt SRID. Icke-geodetiska koordinatsystem ger upphov till ett fel om de används.

A NAD83 longitude/latitude geography value:

Code

```
SELECT 'SRID=4269;POINT(-123 34)'::geography;
```

Output

```
geography
----------------------------------------------------
 0101000020AD10000000000000000C05EC00000000000004140
```

A NAD27 longitude/latitude geography value:

Code

```
SELECT 'SRID=4267;POINT(-123 34)'::geography;
```

Output

```
geography
----------------------------------------------------
 0101000020AB10000000000000000C05EC00000000000004140
```

NAD83 UTM zone meters fails because it is a meter-based planar projection.

Code

```
SELECT 'SRID=26910;POINT(-123 34)'::geography;
```

Output

```
ERROR:  Only lon/lat coordinate systems are supported in geography.
```

Fråge- och mätfunktioner använder enheter i meter. Avståndsparametrar bör därför uttryckas i meter och returvärden bör förväntas i meter (eller kvadratmeter för områden).

This query finds rows within a 1000 km tolerance:

Code

```
SELECT name FROM global_points WHERE ST_DWithin(location, 'SRID=4326;POINT(-110 29)':: ↩
    geography, 1000000);
```

Du kan se geografins kraft i aktion genom att beräkna hur nära ett plan som flyger en cirkelrutt från Seattle till London (LINESTRING(-122,33 47,606, 0,0 51,5)) kommer Reykjavik (POINT(-21,96 64,15))(kartlägg rutten).

Geografitypen beräknar det kortaste verkliga avståndet på 122,235 km över sfären mellan Reykjavik och storcirkelflygvägen mellan Seattle och London.

Code

```
SELECT ST_Distance('LINESTRING(-122.33 47.606,0.0 51.5)'::geography, 'POINT(-21.96 64.15)' ↩
    ::geography);
```

Output

```
st_distance
-----------------
 122235.23815667
```

Geometritypen beräknar ett meningslöst kartesiskt avstånd mellan Reykjavik och den raka vägen från Seattle till London utritad på en platt världskarta. Den nominella enheten för resultatet är grader”, men resultatet motsvarar inte någon verklig vinkelskillnad mellan punkterna, så att kalla dem gg-rader”är felaktigt.

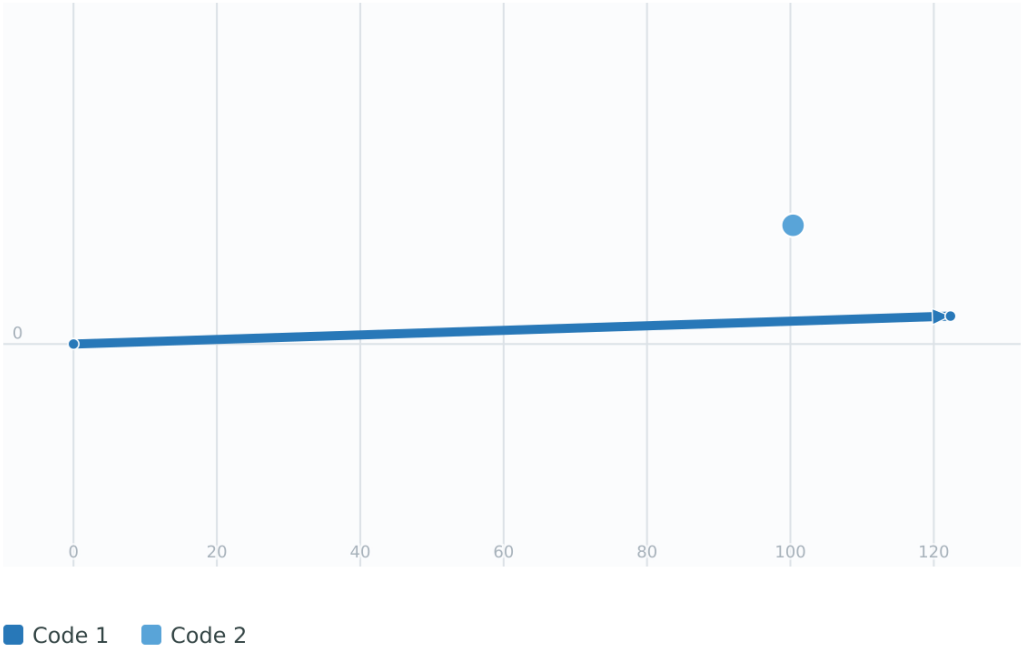
Code

```
SELECT ST_Distance('LINESTRING(-122.33 47.606,0.0 51.5)'::geometry, 'POINT(-21.96 64.15)':: ↩
    geometry);
```

Output

```
st_distance
-------------------
 13.342271221453624
```

Figure



4.3.3 När ska datatypen Geography användas

Med datatypen Geography kan du lagra data i koordinater för longitud/latitud, men till en kostnad: det finns färre funktioner definierade för GEOGRAPHY än för GEOMETRY; de funktioner som är definierade tar mer CPU-tid att utföra.

Den datatyp du väljer bör bestämmas av det förväntade arbetsområdet för den applikation du bygger. Kommer dina data att omfatta hela jordklotet eller ett stort kontinentalt område, eller är de lokala för en stat, ett län eller en kommun?

- Om dina data finns i ett litet område kan det hända att du tycker att det är bäst att välja en lämplig projektion och använda GEOMETRY, med tanke på prestanda och tillgängliga funktioner.
- Om dina data är globala eller täcker en kontinental region kanske du tycker att GEOGRAPHY gör att du kan bygga ett system utan att behöva oroa dig för projektionsdetaljer. Du lagrar dina data i longitud/latitud och använder de funktioner som har definierats i GEOGRAPHY.
- Om du inte förstår projektioner, inte vill lära dig mer om dem och är beredd att acceptera de begränsningar i funktionalitet som finns i GEOGRAPHY, kan det vara lättare för dig att använda GEOGRAPHY än GEOMETRY. Läs in helt enkelt upp dina data som longitud/latitud och fortsätt därifrån.

Se Avsnitt 13.11 för en jämförelse mellan vad som stöds för Geography respektive Geometry. För en kort lista och beskrivning av Geography-funktioner, se Avsnitt 13.4

4.3.4 Avancerade frågor och svar om Geography

1. *Räknar du på sfären eller sfäroiden?*

   By default, all distance and area calculations are done on the spheroid. You should find that the results of calculations in local areas match up well with local planar results in good local projections. Over larger areas, the spheroidal calculations will be more accurate than any calculation done on a projected plane. The difference is that the spheroid takes into account the curvature and non-spherical shape of the Earth, whereas planar calculations assume a flat surface. Alla geografifunktioner har möjlighet att använda en sfärberäkning genom att ställa in en sista boolesk parameter till FALSE". Detta snabbar upp beräkningarna något, särskilt i fall där geometrierna är mycket enkla.

2. *Hur är det med datumlinjen och polerna?*

   Alla beräkningar har ingen uppfattning om datumlinje eller poler, koordinaterna är sfäriska (longitud/latitud) så en form som korsar datumlinjen är, ur en beräkningssynpunkt, inte annorlunda än någon annan form.

3. *Vilken är den längsta båge du kan bearbeta?*

   Vi använder storcirkelbågar som "interpoleringslinjemellan två punkter. Det innebär att två punkter faktiskt sammanfogas på två olika sätt, beroende på vilken riktning du färdas längs storcirkeln. All vår kod förutsätter att punkterna sammanfogas av den *kortaste* av de två vägarna längs storcirkeln. Som en följd av detta kommer former som har bågar på mer än 180 grader inte att modelleras korrekt.

4. *Varför är det så långsamt att beräkna området Europa / Ryssland / infoga stor geografisk region här ?*

   Eftersom polygonen är så jäkla stor! Stora områden är dåliga av två skäl: deras gränser är enorma, så indexet tenderar att dra funktionen oavsett vilken fråga du kör; antalet toppar är enormt, och tester (avstånd, inneslutning) måste korsa topplistan minst en gång och ibland N gånger (med N som antalet toppar i den andra kandidatfunktionen). Precis som med GEOMETRY rekommenderar vi att du, när du har mycket stora polygoner men gör sökningar i små områden, denormaliserardina geometriska data i mindre bitar så att indexet effektivt kan göra underfrågor på delar av objektet och så att sökningarna inte behöver ta fram hela objektet varje gång. Vänligen se ST_Subdivide funktionsdokumentation. Bara för att du *kan* lagra hela Europa i en polygon betyder det inte att du *bör* göra det.

4.4 Validering av geometri

PostGIS är förenligt med Open Geospatial Consortiums (OGC) specifikation för enkla funktioner. Denna standard definierar begreppen *enkel* och *giltig* geometri. Dessa definitioner gör det möjligt för

geometrimodellen Simple Features att representera spatiala objekt på ett konsekvent och otvetydigt sätt som stöder effektiva beräkningar. (Obs: OGC SF och SQL/MM har samma definitioner för enkel och giltig)

4.4.1 Enkel geometri

En *enkel* geometri är en geometri som inte har några avvikande geometriska punkter, t.ex. självskärning eller självtangentiering.

En `POINT` är i sig själv *enkel* som ett 0-dimensionellt geometriobjekt.

`MULTIPOINT`sär *enkla* om inte två koordinater`(POINT`s`)`är lika (har identiska koordinatvärden).

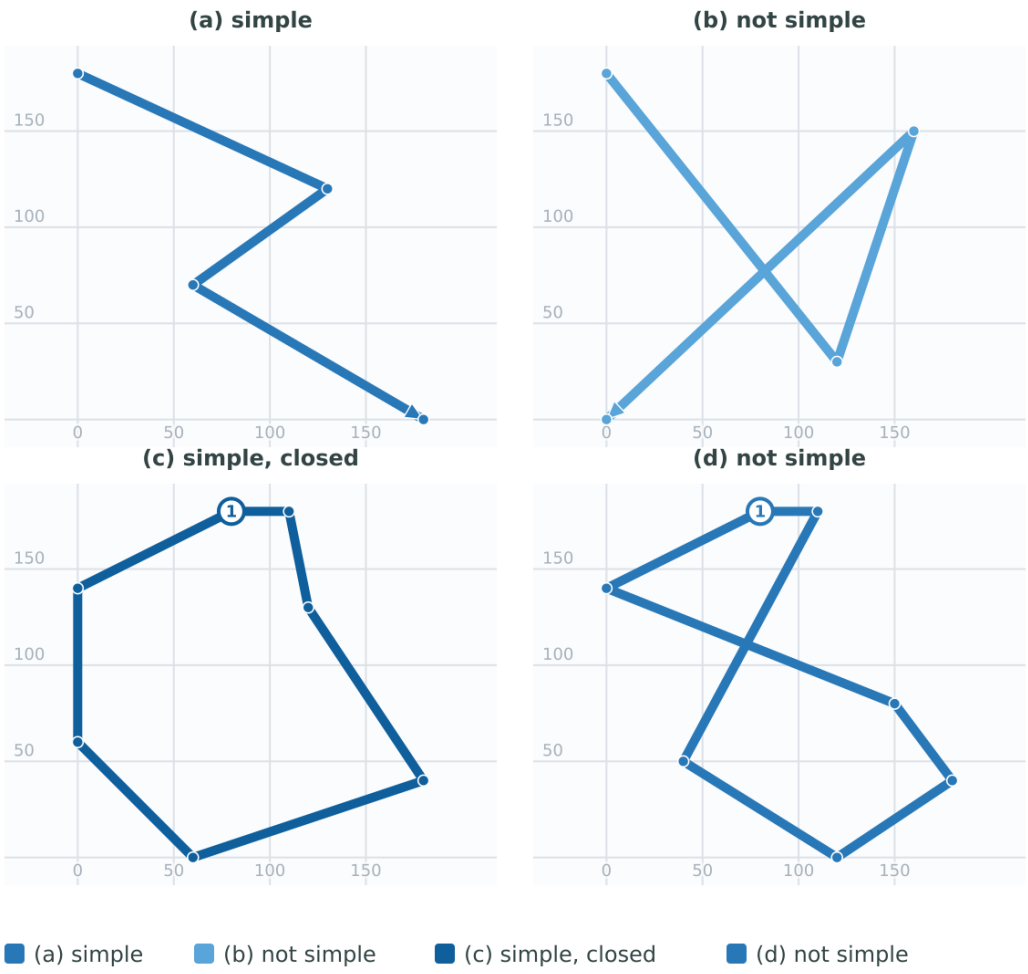
En `LINESTRING` är *enkel* om den inte passerar genom samma punkt två gånger, med undantag för ändpunkterna. Om ändpunkterna på en enkel LineString är identiska kallas den *sluten* och kallas för en Linear Ring.

Use ST_IsSimple to classify linear geometries. Each labeled result below is returned only when the function agrees with the stated classification; example **(c)** also verifies that the simple LineString is closed.

Code

```
WITH cases(label, geom) AS (VALUES
  ('a', 'LINESTRING (10 190, 140 130, 70 80, 190 10)'::geometry),
  ('b', 'LINESTRING (10 190, 130 40, 170 160, 10 10)'::geometry),
  ('c', 'LINESTRING (90 190, 120 190, 130 140, 190 50, 70 10, 10 70, 10 150, 90 190)':: ↩
    geometry),
  ('d', 'LINESTRING (90 190, 120 190, 50 60, 130 10, 190 50, 160 90, 10 150, 90 190)':: ↩
    geometry)
)
SELECT
  (SELECT geom FROM cases WHERE label = 'a' AND ST_IsSimple(geom))
    AS "input_(a) simple",
  (SELECT geom FROM cases WHERE label = 'b' AND NOT ST_IsSimple(geom))
    AS "input_(b) not simple",
  (SELECT geom FROM cases
             WHERE label = 'c' AND ST_IsSimple(geom) AND ST_IsClosed(geom))
    AS "input_(c) simple, closed",
  (SELECT geom FROM cases WHERE label = 'd' AND NOT ST_IsSimple(geom))
    AS "input_(d) not simple";
```

Figure



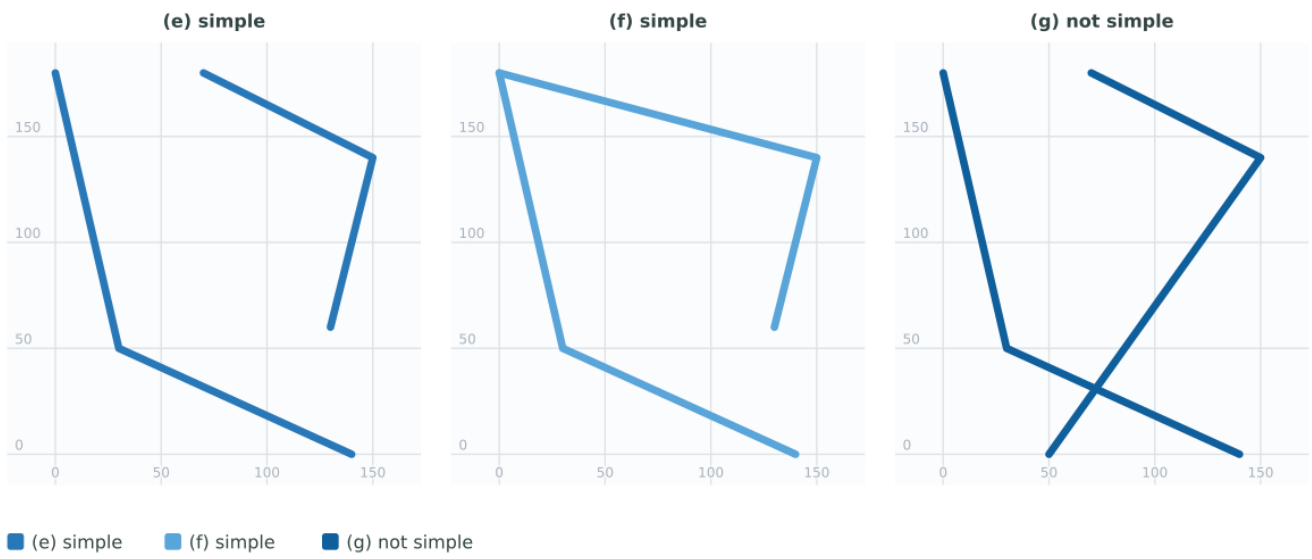
En `MULTILINESTRING` är *enkel* endast om alla dess element är enkla och den enda skärningen mellan två element sker i punkter som ligger på båda elementens gränser.

The next example applies the same classification to the three MultiLineStrings. In **(f)** the component lines meet only at a shared endpoint, while in **(g)** they cross in their interiors.

Code

```
WITH cases(label, geom) AS (VALUES
  ('e', 'MULTILINESTRING ((30 190, 60 60, 170 10), (100 190, 180 150, 160 70))'::geometry),
  ('f', 'MULTILINESTRING ((30 190, 60 60, 170 10), (30 190, 180 150, 160 70))'::geometry),
  ('g', 'MULTILINESTRING ((30 190, 60 60, 170 10), (100 190, 180 150, 80 10))'::geometry)
)
SELECT
  (SELECT geom FROM cases WHERE label = 'e' AND ST_IsSimple(geom))
    AS "input_(e) simple",
  (SELECT geom FROM cases WHERE label = 'f' AND ST_IsSimple(geom))
    AS "input_(f) simple",
  (SELECT geom FROM cases WHERE label = 'g' AND NOT ST_IsSimple(geom))
    AS "input_(g) not simple";
```

Figure



`POLYGONER`bildas av linjära ringar, så giltig polygonal geometri är alltid *enkel*.

Generellt kräver inte PostGIS-funktioner att geometriska argument ska vara enkla. Enkelhet används främst som en grund för att definiera geometrisk validitet. Det är också ett krav för vissa typer av spatiala datamodeller (t.ex. tillåter linjära nätverk ofta inte linjer som korsar varandra). Multipunkts- och linjärgeometri kan göras enkel med hjälp av ST_UnaryUnion.

4.4.2 Giltig geometri

Geometrisk validitet gäller främst för 2-dimensionella geometrier(`POLYGONER`och `MULTIPOLYGONER`). Validiteten definieras av regler som gör att polygonal geometri kan modellera plana områden på ett otvetydigt sätt.

En `POLYGON` är *giltig* om:

1. polygonens begränsningsringar (den yttre skalringen och de inre hålringarna) är *enkla* (korsar inte varandra och berör inte varandra). På grund av detta kan en polygon inte ha snittlinjer, spikar eller öglor. Detta innebär att polygonhål måste representeras som inre ringar, snarare än genom att den yttre ringen självberör (ett s.k. ”inverterat hål”).
2. gränsringarna korsar inte varandra
3. begränsningsringar kan beröra varandra i punkter men endast som en tangent (dvs. inte i en linje)
4. inre ringar är inneslutna i den yttre ringen
5. polygonens inre är enkelt sammanhängande (dvs. ringarna får inte beröra varandra på ett sätt som delar upp polygonen i mer än en del)

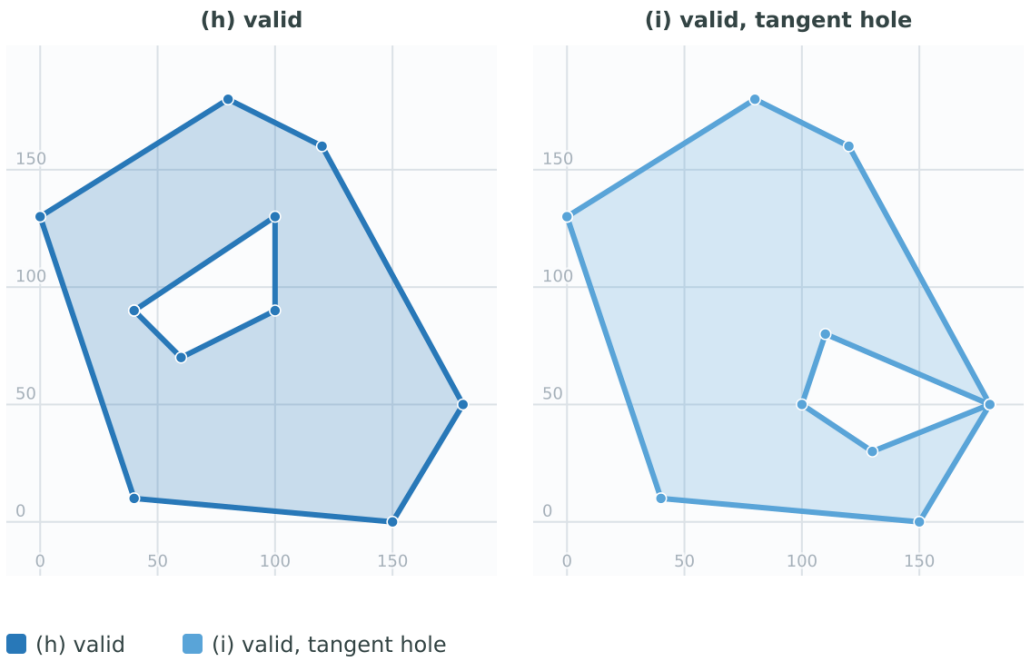
Consecutive repeated points in a polygon boundary ring do not make the ring non-simple. Under the SFS curve model, a curve is interpreted by a continuous parameterization; consecutive equal coordinates are topologically indistinguishable from a single vertex. PostGIS therefore treats them as redundant vertices, so they do not make an otherwise valid polygon invalid.

The following examples use ST_IsValid to check the validity rules. Each labeled geometry is returned only if its computed validity agrees with the label.

Code

```
WITH cases(label, geom) AS (VALUES
  ('h', 'POLYGON ((10 140, 90 190, 130 170, 190 60, 160 10, 50 20, 10 140),
                  (50 100, 70 80, 110 100, 110 140, 50 100))'::geometry),
  ('i', 'POLYGON ((10 140, 90 190, 130 170, 190 60, 160 10, 50 20, 10 140),
                  (190 60, 140 40, 110 60, 120 90, 190 60))'::geometry)
)
SELECT
  (SELECT geom FROM cases WHERE label = 'h' AND ST_IsValid(geom))
    AS "input_(h) valid",
  (SELECT geom FROM cases WHERE label = 'i' AND ST_IsValid(geom))
    AS "input_(i) valid, tangent hole";
```

Figure



Examples **(j-m)** are invalid. They show, respectively, a disconnected interior, crossing rings, a cut line, and a hole outside the shell. The invalidity reason is checked with ST_IsValidReason. The disconnected interior in **(j)** can instead be represented as a valid `MULTIPOLYGON`.

Code

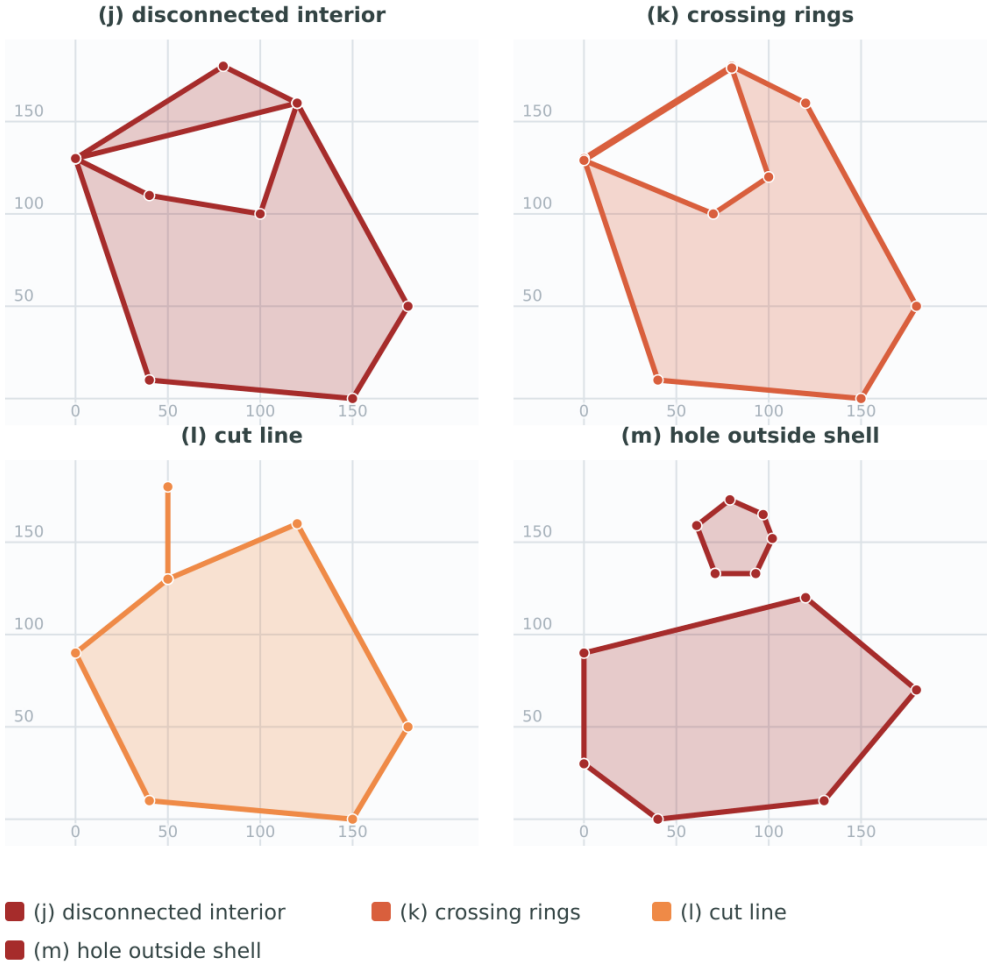
```
WITH cases(label, geom, reason_pattern) AS (VALUES
  ('(j) disconnected interior',
   'POLYGON ((10 140, 90 190, 130 170, 190 60, 160 10, 50 20, 10 140),
             (130 170, 10 140, 50 120, 110 110, 130 170))'::geometry,
   'Interior is disconnected%'),
  ('(k) crossing rings',
   'POLYGON ((10 140, 90 190, 130 170, 190 60, 160 10, 50 20, 10 140),
             (90 189, 10 139, 80 110, 110 130, 90 189))'::geometry,
   'Self-intersection%'),
  ('(l) cut line',
   'POLYGON ((10 100, 60 140, 60 190, 60 140, 130 170,
              190 60, 160 10, 50 20, 10 100))'::geometry,
   'Ring Self-intersection%'),
  ('(m) hole outside shell',
   'POLYGON ((10 40, 10 100, 130 130, 190 80, 140 20, 50 10, 10 40),
             (81 143, 103 143, 112 162, 107 175, 89 183, 71 169, 81 143))'::geometry,
```

```
   'Hole lies outside shell%')
)
SELECT label, geom
FROM cases
WHERE NOT ST_IsValid(geom)
  AND ST_IsValidReason(geom) LIKE reason_pattern
ORDER BY label;
```

Output

```
label          | geom
-----------------------+-----------------------------------------------------------------------------------------------------------------------------------
 (j) disconnected interior | POLYGON((10 140,90 190,130 170,190 60,160 10,50 20,10 140),(130 170,10 140,50 120,110 ↩
    110,130 170))
 (k) crossing rings       | POLYGON((10 140,90 190,130 170,190 60,160 10,50 20,10 140),(90 189,10 139,80 110,110 ↩
    130,90 189))
 (l) cut line             | POLYGON((10 100,60 140,60 190,60 140,130 170,190 60,160 10,50 20,10 100))
 (m) hole outside shell   | POLYGON((10 40,10 100,130 130,190 80,140 20,50 10,10 40),(81 143,103 143,112 162,107 ↩
    175,89 183,71 169,81 143))
```

Figure



En `MULTIPOLYGON` är *giltig* om:

1. dess element `POLYGONER`är giltiga
2. elementen inte överlappar varandra (dvs. deras insidor får inte skära varandra)
3. elementen berör varandra endast i punkter (dvs. inte längs en linje)

The next example checks the MultiPolygon rules. In **(n)** the polygons touch at only one point. In **(o)** they share an edge, and in **(p)** their interiors overlap.

Code

```
WITH cases(label, geom, valid, reason_pattern) AS (VALUES
  ('(n) point touch',
   'MULTIPOLYGON (((10 40, 29 118, 100 75, 180 88, 140 30, 70 10, 10 40)),
                  ((50 170, 100 190, 160 180, 180 140, 180 90, 120 120,
                    30 120, 50 170)))'::geometry,
   true, NULL),
  ('(o) shared edge',
   'MULTIPOLYGON (((10 40, 30 120, 120 120, 180 88, 140 30, 70 10, 10 40)),
                  ((50 170, 100 190, 160 180, 180 140, 180 90, 120 120,
                    30 120, 50 170)))'::geometry,
   false, 'Self-intersection%'),
  ('(p) overlap',
   'MULTIPOLYGON (((10 40, 42 93, 130 150, 180 86, 140 30, 70 10, 10 40)),
                  ((50 170, 100 190, 138 182, 149 156, 146 137, 100 100,
                    58 140, 50 170)))'::geometry,
   false, 'Self-intersection%')
)
SELECT label, geom
FROM cases
WHERE ST_IsValid(geom) = valid
  AND (valid OR ST_IsValidReason(geom) LIKE reason_pattern)
ORDER BY label;
```

Output

```
label       | geom
------------------+-----------------------
```

För linjär geometri är den enda giltighetsregeln att `LINESTRING`småste ha minst två punkter och ha en längd som inte är noll (eller motsvarande, ha minst två distinkta punkter.) Observera att icke-enkla (självskärande) linjer är giltiga.

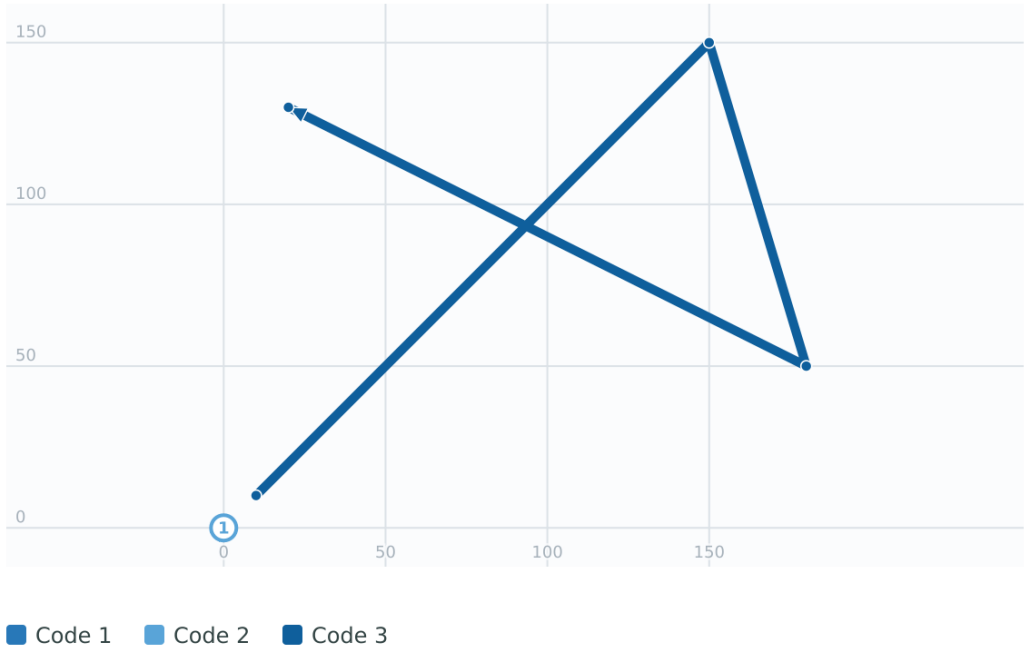
Code

```
SELECT
ST_IsValid('LINESTRING(0 0,1 1)') AS len_nonzero,
ST_IsValid('LINESTRING(0 0,0 0,0 0)') AS len_zero,
ST_IsValid('LINESTRING(10 10,150 150,180 50,20 130)') AS self_int;
```

Output

```
len_nonzero | len_zero | self_int
-------------+----------+----------
 t           | f        | t
```

Figure



`POINT-` och `MULTIPOINT-geometrierna` har inga giltighetsregler.

4.4.3 Hantering av validitet

PostGIS tillåter skapande och lagring av både giltig och ogiltig geometri. Detta gör att ogiltig geometri kan upptäckas och flaggas eller fixas. Det finns också situationer där OGC:s giltighetsregler är strängare än önskat (exempel på detta är linestrings med noll längd och polygoner med inverterade hål)

Många av de funktioner som PostGIS tillhandahåller förutsätter att geometriargumenten är giltiga. Det är till exempel inte meningsfullt att beräkna arean av en polygon som har ett hål definierat utanför polygonen, eller att konstruera en polygon från en icke-enkel begränsningslinje. Genom att anta giltiga geometriska indata kan funktioner arbeta mer effektivt, eftersom de inte behöver kontrollera att de är topologiskt korrekta. (Noterbara undantag är att nollängdslinjer och polygoner med inversioner i allmänhet hanteras korrekt) Dessutom producerar de flesta PostGIS-funktioner giltiga geometriska utdata om indata är giltiga. Detta gör att PostGIS-funktioner kan kedjas ihop på ett säkert sätt.

Om du stöter på oväntade felmeddelanden när du anropar PostGIS-funktioner (t.ex. GEOS Intersection() kastade ett fel!") bör du först bekräfta att funktionsargumenten är giltiga. Om de inte är det bör du överväga att använda någon av nedanstående tekniker för att säkerställa att de data du bearbetar är giltiga.

Notera
Om en funktion rapporterar ett fel med giltiga indata kan du ha hittat ett fel i antingen PostGIS eller ett av de bibliotek som används, och du bör rapportera detta till PostGIS-projektet. Detsamma gäller om en PostGIS-funktion returnerar en ogiltig geometri för giltig indata.

För att testa om en geometri är giltig kan du använda funktionen ST_IsValid:

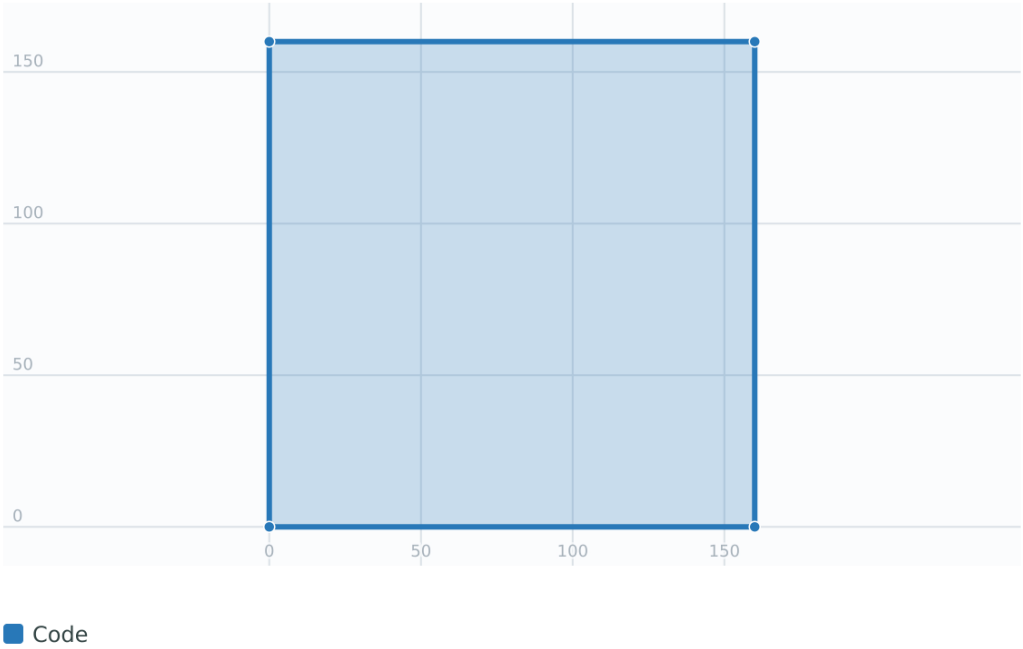
Code

```
SELECT ST_IsValid('POLYGON ((20 180,180 180,180 20,20 20,20 180))');
```

Output

```
t
```

Figure



Information om arten av och platsen för en geometrisk invaliditet tillhandahålls av funktionen ST_IsValidDeta

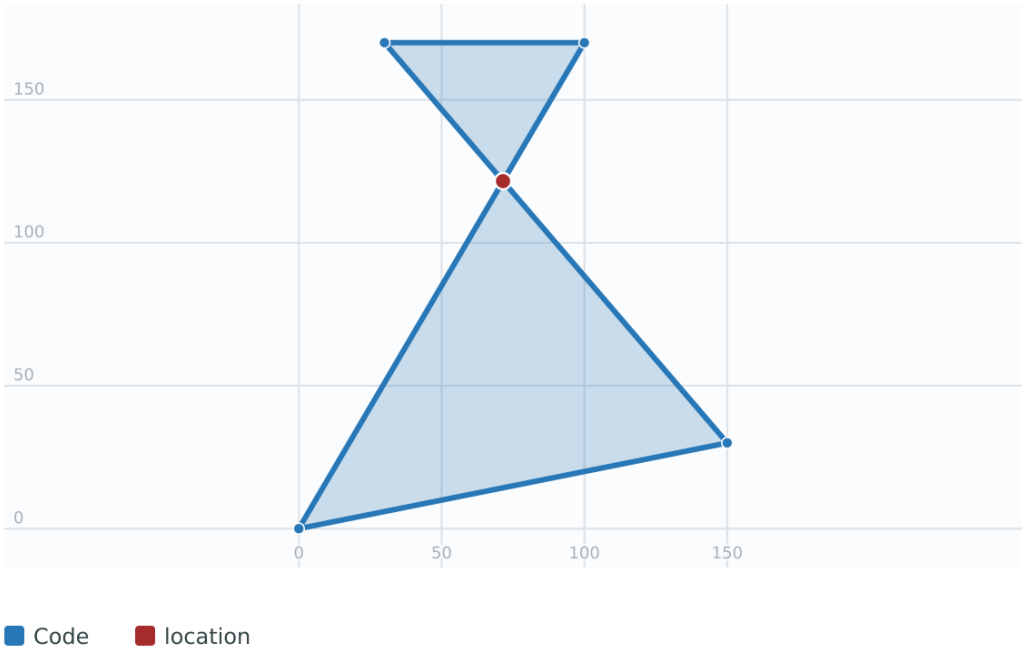
Code

```
SELECT valid, reason,location AS location
        FROM ST_IsValidDetail('POLYGON ((20 20,120 190,50 190,170 50,20 20))') AS t;
```

Output

```
valid |      reason       |                  location
------+-------------------+--------------------------------------------
 f    | Self-intersection | POINT(91.51162790697674 141.56976744186048)
```

Figure



I vissa situationer är det önskvärt att korrigera ogiltig geometri automatiskt. Använd funktionen ST_MakeValid för att göra detta.(`ST_MakeValid` är ett exempel på en spatial funktion som *tillåter* ogiltiga indata!)

Som standard kontrollerar PostGIS inte giltigheten vid laddning av geometri, eftersom giltighetstestning kan ta mycket CPU-tid för komplexa geometrier. Om du inte litar på dina datakällor kan du tvinga fram en giltighetskontroll på dina tabeller genom att lägga till en check constraint:

Code

```
ALTER TABLE mytable
ADD CONSTRAINT geometry_valid_check
  CHECK (ST_IsValid(geom));
```

4.5 Spatiala referenssystem

Ett spatialt referenssystem (SRS) (även kallat koordinatreferenssystem (CRS)) definierar hur geometri refereras till platser på jordens yta. Det finns tre typer av SRS:

- En **geodetisk** SRS använder vinkelkoordinater (longitud och latitud) som kartläggs direkt på jordytan.
- En **projicerad** SRS använder en matematisk projektionstransformation för att platta till”ytan på den sfäroida jorden till ett plan. Den tilldelar platskoordinater på ett sätt som gör det möjligt att direkt mäta storheter som avstånd, yta och vinkel. Koordinatsystemet är kartesiskt, vilket innebär att det har en definierad ursprungspunkt och två vinkelräta axlar (vanligtvis orienterade mot norr och öster). Varje projicerad SRS använder en angiven längdenhet (vanligtvis meter eller fot). En projicerad SRS kan vara begränsad i sitt tillämpningsområde för att undvika distorsion och passa inom de definierade koordinatgränserna.
- Ett **lokalt** SRS är ett kartesiskt koordinatsystem som inte är refererat till jordytan. I PostGIS specificeras detta med ett SRID-värde på 0.

Det finns många olika spatiala referenssystem i bruk. Vanliga SRS är standardiserade i European Petroleum Survey Group EPSG-databasen. För enkelhetens skull hänvisar PostGIS (och många andra spatiala system) till SRS-definitioner med hjälp av en heltalsidentifierare som kallas SRID.

En geometri är associerad med ett spatialt referenssystem genom sitt SRID-värde, som nås via ST_SRID. SRID för en geometri kan tilldelas med hjälp av ST_SetSRID. Vissa geometry constructor-funktioner gör det möjligt att ange en SRID (t.ex. ST_Point och ST_MakeEnvelope). EWKT-formatet stöder SRID med prefixet `SRID=n;`.

Spatiala funktioner som bearbetar par av geometrier (t.ex. överlagrings- och relationsfunktioner) kräver att de ingående geometrierna finns i samma spatiala referenssystem (har samma SRID). Geometridata kan omvandlas till ett annat spatialt referenssystem med hjälp av ST_Transform och ST_TransformPipel
Geometri som returneras från funktioner har samma SRS som indatageometrierna.

4.5.1 Tabellen SPATIAL_REF_SYS

Tabellen `SPATIAL_REF_SYS` som används av PostGIS är en OGC-kompatibel databastabell som definierar de tillgängliga spatiala referenssystemen. Den innehåller de numeriska SRID:erna och textbeskrivningarna av koordinatsystemen.

Definitionen av tabellen `spatial_ref_sys` är:

Code

```
CREATE TABLE spatial_ref_sys (
  srid       INTEGER NOT NULL PRIMARY KEY,
  auth_name  VARCHAR(256),
  auth_srid  INTEGER,
  srtext     VARCHAR(2048),
  proj4text  VARCHAR(2048)
)
```

Kolumnerna är..:

`srid` En heltalskod som unikt identifierar det spatiala referenssystemet (SRS) i databasen.

`auth_name` Namnet på den standard eller det standardiseringsorgan som åberopas för detta referenssystem. Exempelvis är ”EPSG”ett giltigt `auth_name`.

`auth_srid` ID för det spatiala referenssystemet enligt definitionen av den myndighet som anges i `auth_name`. När det gäller EPSG är detta EPSG-koden.

`srtext` Representationen av det spatiala referenssystemet i Well-Known Text. Ett exempel på en WKT SRS-representation är:

Code

```
PROJCS["NAD83 / UTM Zone 10N",
  GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
      SPHEROID["GRS 1980", 6378137, 298.257222101]
    ],
    PRIMEM["Greenwich", 0],
    UNIT["degree", 0.0174532925199433]
  ],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin", 0],
  PARAMETER["central_meridian", -123],
  PARAMETER["scale_factor", 0.9996],
  PARAMETER["false_easting", 500000],
  PARAMETER["false_northing", 0],
  UNIT["metre", 1]
]
```

För en diskussion om SRS WKT, se OGC-standarden Well-known text representation of coordinate reference systems.

proj4text PostGIS använder PROJ-biblioteket för att tillhandahålla funktioner för koordinattransformation. Kolumnen `proj4text` innehåller PROJ-koordinatdefinitionssträngen för en viss SRID. Till exempel:

Code

```
+proj=utm +zone=10 +ellps=clrk66 +datum=NAD27 +units=m
```

Mer information finns på PROJ-webbplatsen. Filen `spatial_ref_sys.sql` innehåller både `srtext`- och `proj4text-definitioner` för alla EPSG-projektioner.

Vid hämtning av definitioner av spatiala referenssystem för användning i transformationer använder PostGIS följande strategi:

- Om `auth_name` och `auth_srid` finns (icke-NULL) används PROJ SRS baserat på dessa poster (om en sådan finns).
- Om `srtext` finns kan du om möjligt skapa en SRS med hjälp av den.
- Om `proj4text` finns kan du om möjligt skapa en SRS med hjälp av den.

4.5.2 Användardefinierade spatiala referenssystem

Tabellen PostGIS `spatial_ref_sys` innehåller över 3000 av de vanligaste definitionerna av spatiala referenssystem som hanteras av PROJ-projektionsbiblioteket. Men det finns många koordinatsystem som den inte innehåller. Du kan lägga till SRS-definitioner i tabellen om du har den information som krävs om det spatiala referenssystemet. Eller så kan du definiera ditt eget spatiala referenssystem om du är bekant med PROJ constructs. Tänk på att de flesta spatiala referenssystem är regionala och inte har någon betydelse när de används utanför de gränser som de är avsedda för.

En resurs för att hitta spatiala referenssystem som inte definieras i kärnuppsättningen är http://spatialrefere

Some commonly used spatial reference systems are: 4326 - WGS 84 Long Lat, 4269 - NAD 83 Long Lat, 3395 - WGS 84 World Mercator, 2163 - US National Atlas Equal Area, and the 60 WGS84 UTM zones. UTM zones are one of the most ideal for measurement, but only cover 6-degree regions. For WGS84 longitude/latitude data, the UTM zone is `least(floor((longitude + 180) / 6) + 1, 60)`; the EPSG code is `32600 + zone` in the northern hemisphere or `32700 + zone` in the southern hemisphere.

Amerikanska delstater använder spatiala referenssystem i State Plane (meter- eller fotbaserade) - vanligtvis finns ett eller två per delstat. De flesta av de meterbaserade finns i kärnuppsättningen, men många av de fotbaserade eller ESRI-skapade måste kopieras från spatialreference.org.

Du kan även definiera koordinatsystem som inte är jordbaserade, t.ex. Mars 2000 Mars koordinatsystem är inte planar (det är i grader sfäroidalt), men du kan använda det med `geograftypen` för att få längd- och avståndsmätningar i meter i stället för grader.

Här är ett exempel på laddning av ett anpassat koordinatsystem med hjälp av en icke tilldelad SRID och PROJ-definitionen för en US-centrisk Lambert Conformal-projektion:

Code

```
INSERT INTO spatial_ref_sys (srid, proj4text)
VALUES ( 990000,
  '+proj=lcc  +lon_0=-95 +lat_0=25 +lat_1=25 +lat_2=25 +x_0=0 +y_0=0 +datum=WGS84 +units=m  ↩
    +no_defs'
);
```

4.6 Spatiala tabeller

4.6.1 Skapa en spatial tabell

Du kan skapa en tabell för att lagra geometridata genom att använda SQL-satsen CREATE TABLE med en kolumn av typen `geometry`. I följande exempel skapas en tabell med en geometrikolumn som lagrar 2D (XY) LineStrings i BC-Albers koordinatsystem (SRID 3005):

Code

```
CREATE TABLE roads (
  id SERIAL PRIMARY KEY,
  name VARCHAR(64),
  geom geometry(LINESTRING, 3005)
);
```

`Geometritypen` har stöd för två valfria **typmodifierare**:

- **Modifieraren för spatial typ** begränsar vilken typ av former och dimensioner som tillåts i kolumnen. Värdet kan vara vilken som helst av de undertyper av geometri som stöds (t.ex. POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION etc.). Modifieraren stöder begränsningar av koordinatdimensionalitet genom att lägga till suffix: Z, M och ZM. En modifierare av typen LINESTRINGMtillåter t.ex. endast linestrings med tre dimensioner och behandlar den tredje dimensionen som ett mått. På samma sätt kräver POINTZMfyrdimensionella (XYZM) data.
- **sRID-modifieraren** begränsar det spatiala referenssystemets SRID till ett visst nummer. Om den utelämnas är SRID standardvärdet 0.

Exempel på hur man skapar tabeller med geometrikolumner:

Skapa en tabell som innehåller alla typer av geometrier med standard-SRID:

Code

```
CREATE TABLE geoms(gid serial PRIMARY KEY, geom geometry);
```

Skapa en tabell med 2D POINT-geometri med standard-SRID:

Code

```
CREATE TABLE pts(gid serial PRIMARY KEY, geom geometry(POINT));
```

Skapa en tabell med 3D (XYZ) POINTs och en explicit SRID på 3005:

Code

```
CREATE TABLE pts(gid serial PRIMARY KEY, geom geometry(POINTZ, 3005));
```

Skapa en tabell med 4D (XYZM) LINESTRING-geometri med standard SRID:

Code

```
CREATE TABLE lines(gid serial PRIMARY KEY, geom geometry(LINESTRINGZM));
```

Skapa en tabell med 2D POLYGON-geometri med SRID 4267 (NAD 1927 long lat):

Code

```
CREATE TABLE polys(gid serial PRIMARY KEY, geom geometry(POLYGON, 4267));
```

Det är möjligt att ha mer än en geometrikolumn i en tabell. Detta kan anges när tabellen skapas, eller så kan en kolumn läggas till med SQL-satsen ALTER TABLE. I detta exempel läggs en kolumn till som kan innehålla 3D LineStrings:

Code

```
ALTER TABLE roads ADD COLUMN geom2 geometry(LINESTRINGZ, 4326);
```

4.6.2 Visa GEOMETRY_COLUMNS

OGC:s *Simple Features Specification for SQL* definierar metadatatabellen `GEOMETRY_COLUMNS` för att beskriva geometritabellens struktur. I PostGIS är `geometry_columns` en vy som läser från databassystemets katalogtabeller. Detta säkerställer att den spatiala metadatainformationen alltid överensstämmer med de tabeller och vyer som för närvarande definieras. Vyn har följande struktur:

Code

```
\d geometry_columns
```

Output

```
View "public.geometry_columns"
      Column       |          Type          | Modifiers
-------------------+------------------------+-----------
 f_table_catalog   | character varying(256) |
 f_table_schema    | character varying(256) |
 f_table_name      | character varying(256) |
 f_geometry_column | character varying(256) |
 coord_dimension   | integer                |
 srid              | integer                |
 type              | character varying(30)  |
```

Kolumnerna är..:

`f_table_catalog, f_table_schema, f_table_name` Det fullständigt kvalificerade namnet på funktionstabellen som innehåller geometrikolumnen. Det finns ingen PostgreSQL-analog av ”katalogså den kolumnen lämnas tom. För schemaänvänds PostgreSQL-schemanamnet(`offentligt` är standard).

`f_geometry_column` Namnet på geometrikolumnen i feature-tabellen.

`coord_dimension` The coordinate dimension (2, 3 or 4) of the column. The coordinate dimension is at least 2, representing X and Y coordinates.

`srid` ID för det spatiala referenssystem som används för koordinatgeometrin i denna tabell. Det är en främmande nyckelreferens till tabellen `spatial_ref_sys` (se Avsnitt 4.5.1).

`type` Typen av det spatiala objektet. För att begränsa den spatiala kolumnen till en enda typ, använd en av följande: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION eller motsvarande XYM-versioner POINTM, LINESTRINGM, POLYGONM, MULTIPOINTM, MULTILINESTRINGM, MULTIPOLYGONM, GEOMETRYCOLLECTIONM. För heterogena samlingar (av blandad typ) kan du använda GEOMETRYsom typ.

4.6.3 Manuell registrering av geometrikolumner

Två av de fall där du kan behöva detta är vid SQL Views och bulkinsättningar. För bulkinsättningar kan du korrigera registreringen i geometry_columns-tabellen genom att begränsa kolumnen eller göra en alter table. För vyer kan du exponera med hjälp av en CAST-operation. Observera att om din kolumn är typmodbaserad kommer skapandeprocessen att registrera den korrekt, så du behöver inte göra någonting. Även vyer som inte har någon spatial funktion applicerad på geometrin kommer att registreras på samma sätt som den underliggande tabellgeometrikolumnen.

For example, suppose you have a view created like this.

Code

```
CREATE VIEW public.vwmytablemercator AS
SELECT gid, ST_Transform(geom, 3395) As geom, f_name
FROM public.mytable;
```

To register the geometry column correctly, cast the transformed geometry.

Code

```
DROP VIEW public.vwmytablemercator;
CREATE VIEW  public.vwmytablemercator AS
    SELECT gid, ST_Transform(geom, 3395)::geometry(Geometry, 3395) As geom, f_name
    FROM public.mytable;
```

If the geometry type is known to be a 2D Polygon, specify it in the cast.

Code

```
DROP VIEW public.vwmytablemercator;
CREATE VIEW  public.vwmytablemercator AS
    SELECT gid, ST_Transform(geom, 3395)::geometry(Polygon, 3395) As geom, f_name
    FROM public.mytable;
```

For a derivative table created by a bulk insert, create the table first.

Code

```
SELECT poi.gid, poi.geom, citybounds.city_name
INTO myschema.my_special_pois
FROM poi INNER JOIN citybounds ON ST_Intersects(citybounds.geom, poi.geom);
```

Create a 2D spatial index on the new table.

Code

```
CREATE INDEX idx_myschema_myspecialpois_geom_gist
ON myschema.my_special_pois USING gist(geom);
```

If the points are 3D or measured, an n-D index may be more appropriate.

Code

```
CREATE INDEX my_special_pois_geom_gist_nd
ON my_special_pois USING gist(geom gist_geometry_ops_nd);
```

Manually register the new table's geometry column in `geometry_columns`. This also changes the underlying table structure to make the column typmod-based.

Code

```
SELECT Populate_Geometry_Columns('myschema.my_special_pois'::regclass);
```

To retain constraint-based definition behavior, such as for inherited tables where all children do not have the same type and SRID, set the optional `use_typmod` argument to false.

Code

```
SELECT Populate_Geometry_Columns('myschema.my_special_pois'::regclass, false);
```

Although the old-constraint based method is still supported, a constraint-based geometry column used directly in a view, will not register correctly in geometry_columns, as will a typmod one. In this example we define a column using typmod and another using constraints.

Code

```
CREATE TABLE pois_ny(gid SERIAL PRIMARY KEY, poi_name text, cat text, geom geometry(POINT, ↩
    4326));
SELECT AddGeometryColumn('pois_ny', 'geom_2160', 2160, 'POINT', 2, false);
```

If we run **\d pois_ny** in psql, we observe they are defined differently -- one is typmod, one is constraint.

Code

```
\d pois_ny
```

Output

```
Table "public.pois_ny"
  Column   |         Type          |                      Modifiers

-----------+-----------------------+-----------------------------------------------------
 gid       | integer               | not null default nextval('pois_ny_gid_seq'::regclass)
 poi_name  | text                  |
 cat       | character varying(20) |
 geom      | geometry(Point,4326)  |
 geom_2160 | geometry              |
Indexes:
    "pois_ny_pkey" PRIMARY KEY, btree (gid)
Check constraints:
    "enforce_dims_geom_2160" CHECK (st_ndims(geom_2160) = 2)
    "enforce_geotype_geom_2160" CHECK (geometrytype(geom_2160) = 'POINT'::text
        OR geom_2160 IS NULL)
    "enforce_srid_geom_2160" CHECK (st_srid(geom_2160) = 2160)
```

I geometry_columns registreras de båda korrekt

Code

```
SELECT f_table_name, f_geometry_column, srid, type
FROM geometry_columns
WHERE f_table_name = 'pois_ny';
```

Output

```
f_table_name | f_geometry_column | srid | type
-------------+-------------------+------+-------
pois_ny      | geom              | 4326 | POINT
pois_ny      | geom_2160         | 2160 | POINT
```

Men - om vi skulle skapa en vy så här

Code

```
CREATE VIEW vw_pois_ny_parks AS
SELECT *
  FROM pois_ny
  WHERE cat='park';

SELECT f_table_name, f_geometry_column, srid, type
    FROM geometry_columns
    WHERE f_table_name = 'vw_pois_ny_parks';
```

The typmod based geom view column registers correctly, but the constraint based one does not.

Output

```
f_table_name   | f_geometry_column | srid |   type
-----------------+-------------------+------+----------
 vw_pois_ny_parks | geom              | 4326 | POINT
 vw_pois_ny_parks | geom_2160         |    0 | GEOMETRY
```

This may change in future versions of PostGIS, but for now to force the constraint-based view column to register correctly, you need to do this:

Code

```
DROP VIEW vw_pois_ny_parks;
CREATE VIEW vw_pois_ny_parks AS
SELECT gid, poi_name, cat,
  geom,
  geom_2160::geometry(POINT,2160) As geom_2160
  FROM pois_ny
  WHERE cat = 'park';
SELECT f_table_name, f_geometry_column, srid, type
    FROM geometry_columns
    WHERE f_table_name = 'vw_pois_ny_parks';
```

Output

```
f_table_name   | f_geometry_column | srid | type
-----------------+-------------------+------+-------
 vw_pois_ny_parks | geom              | 4326 | POINT
 vw_pois_ny_parks | geom_2160         | 2160 | POINT
```

4.7 Läs in spatial data

När du har skapat en spatial tabell är du redo att ladda upp spatiala data till databasen. Det finns två inbyggda sätt att få spatiala data till en PostGIS/PostgreSQL-databas: med hjälp av formaterade SQL-satser eller med hjälp av Shapefile-laddaren.

4.7.1 Använda SQL för att läsa in data

Om spatiala data kan konverteras till en textrepresentation (som antingen WKT eller WKB) kan SQL vara det enklaste sättet att få in data i PostGIS. Data kan bulk-laddas in i PostGIS/PostgreSQL genom att ladda en textfil med SQL `INSERT-satser` med hjälp av SQL-verktyget `psql`.

En SQL-laddningsfil (till exempel`roads.sql`) kan se ut så här:

Code

```
BEGIN;
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (1, 'LINESTRING(191232 243118,191108 243242)', 'Jeff Rd');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (2, 'LINESTRING(189141 244158,189265 244817)', 'Geordie Rd');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (3, 'LINESTRING(192783 228138,192612 229814)', 'Paul St');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (4, 'LINESTRING(189412 252431,189631 259122)', 'Graeme Ave');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (5, 'LINESTRING(190131 224148,190871 228134)', 'Phil Tce');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (6, 'LINESTRING(198231 263418,198213 268322)', 'Dave Cres');
COMMIT;
```

SQL-filen kan laddas in i PostgreSQL med `psql`:

Code

```
psql -d [database] -f roads.sql
```

4.7.2 Använda Shapefile-inläsaren

`Shp2pgsql-dataladdaren` konverterar Shapefiles till SQL som är lämplig för införande i en PostGIS / PostgreSQL-databas antingen i geometri- eller geografiformat. Laddaren har flera driftlägen som väljs med kommandoradsflaggor.

The input must be an unpacked shapefile path. The loader does not read `.zip` archives directly; unpack the `.shp`, `.shx`, and `.dbf` files before loading.

Det finns också ett grafiskt gränssnitt `shp2pgsql-gui` med de flesta av alternativen som kommandoradsladdaren. Detta kan vara lättare att använda för engångsladdning utan skript eller om du är nybörjare på PostGIS. Det kan också konfigureras som ett plugin till PgAdminIII.

(c|a|d|p) Dessa är ömsesidigt uteslutande alternativ:

-c Skapar en ny tabell och fyller i den från Shapefilen. *Detta är standardläget.*

-a Lägger till data från Shapefilen i databastabellen. Observera att filerna måste ha samma attribut och samma datatyper om du vill använda det här alternativet för att läsa in flera filer.

-d Slänger databastabellen innan en ny tabell skapas med data från Shapefilen.

-p Producerar endast SQL-koden för tabellskapande, utan att lägga till några faktiska data. Detta kan användas om du helt vill separera stegen för att skapa tabeller och läsa in data.

-? Visa hjälpskärm.

-D Använd PostgreSQL dump-formatet för utdata. Detta kan kombineras med -a, -c och -d. Det är mycket snabbare att läsa in än standard SQL-formatet ”insert”. Använd detta för mycket stora datamängder.

-f <fid_column>, --feature-id-column <fid_column> Specify the name of the feature id column. The default is `gid`. The value must be a simple PostgreSQL identifier and cannot use a PostgreSQL system column name.

-s [<FROM_SRID>:]<SRID> Skapar och fyller på geometritabellerna med den angivna SRID:en. Anger eventuellt att den ingående shapefilen använder den angivna FROM_SRID, i vilket fall geometrierna kommer att omprojiceras till mål-SRID.

-k Behåll identifierarnas versaler (kolumn, schema och attribut). Observera att alla attribut i Shapefile är UPPPPERCASE.

-i Tvinga alla heltal till standard 32-bitars heltal, skapa inte 64-bitars bigints, även om DBF-headerns signatur verkar motivera det.

-I Skapa ett GiST-index på geometrikolumnen.

-u Create the target table as `UNLOGGED`. This can speed up loading transient staging data, but PostgreSQL does not preserve unlogged table contents after a crash or unclean shutdown.

-m Specify a file containing a set of mappings of (long) column names to 10 character DBF column names. The content of the file is one or more lines of two names separated by white space and no trailing or leading space. For example:

```
COLUMNNAME DBFFIELD1
AVERYLONGCOLUMNNAME DBFFIELD2
```

-S Generera enkla geometrier i stället för MULTI-geometrier. Kommer endast att lyckas om alla geometrier faktiskt är enkla (t.ex. en MULTIPOLYGON med ett enda skal eller en MULTIPOINT med ett enda vertex).

-t <dimensionality> Tvingar utdatageometrin att ha den angivna dimensionaliteten. Använd följande strängar för att ange dimensionaliteten: 2D, 3DZ, 3DM, 4D.

Om indata har färre dimensioner än vad som anges, kommer dessa dimensioner att fyllas i med nollor i utdata. Om indata har fler dimensioner än vad som anges, kommer de oönskade dimensionerna att tas bort.

-w Utdata i WKT-format istället för WKB. Observera att detta kan medföra koordinatavvikelser på grund av förlorad precision.

-e Exekvera varje sats för sig, utan att använda en transaktion. Detta gör det möjligt att läsa in majoriteten av bra data när det finns några dåliga geometrier som genererar fel. Observera att detta inte kan användas med flaggan -D eftersom dump-formatet alltid använder en transaktion.

-W <encoding> Ange kodning av indata (dbf-fil). När detta används konverteras alla attribut i dbf-filen från den angivna kodningen till UTF8. Den resulterande SQL-utdata kommer att innehålla ett `SET CLIENT_ENCODING till UTF8-kommando`, så att backend kan konvertera om från UTF8 till den kodning som databasen är konfigurerad att använda internt.

-N <policy> Policy för hantering av NULL-geometrier (insert*,skip,abort)

-n -n Importera endast DBF-fil. Om dina data inte har någon motsvarande shapefil kommer den automatiskt att växla till detta läge och bara läsa in dbf-filen. Så att ställa in denna flagga behövs bara om du har en fullständig shapefiluppsättning och du bara vill ha attributdata och ingen geometri.

-G Använd geografin i stället för geometrin (kräver lon/lat-data) i WGS84 long lat (SRID=4326)

-T <tablespace> Ange tablespace för den nya tabellen. Index kommer fortfarande att använda standardbordsytan om inte -X-parametern också används. PostgreSQL-dokumentationen har en bra beskrivning av när man ska använda anpassade tablespaces.

-X <tablespace> Ange tablespace för den nya tabellens index. Detta gäller för primärnyckelindexet och det spatiala GIST-indexet om -I också används.

-Z När denna flagga används förhindrar den att `ANALYZE-satser` genereras. Utan flaggan -Z (standardbeteende) kommer `ANALYZE-satserna` att genereras.

Ett exempel på en session där laddaren används för att skapa en inmatningsfil och ladda den kan se ut så här:

Code

```
shp2pgsql -c -D -s 4269 -i -I shaperoads.shp myschema.roadstable > roads.sql
psql -d roadsdb -f roads.sql
```

En konvertering och laddning kan göras i ett steg med hjälp av UNIX pipes:

Code

```
shp2pgsql shaperoads.shp myschema.roadstable | psql -d roadsdb
```

To load a directory of Shapefiles, run the loader once per file and derive the table name from the file name. On Windows `cmd.exe`, use a single percent sign interactively and double percent signs in a batch file:

Code

```
REM In a .bat file, use %%F. At an interactive cmd.exe prompt, use %F.
for %%F in ("C:\data\shapefiles\*.shp") do (
  shp2pgsql -d -D -s 4269 -I -W UTF-8 "%%~fF" "myschema.%%~nF" | psql -d roadsdb
)
```

If every Shapefile has the same attributes and data types and should be appended into one table, create the table from the first file and use `-a` for the remaining files.

4.8 Extrahera spatiala data

Spatiala data kan extraheras från databasen med hjälp av antingen SQL eller Shapefile-dumpern. I avsnittet om SQL presenteras några av de funktioner som finns tillgängliga för att göra jämförelser och frågor om spatiala tabeller.

4.8.1 Använda SQL för att extrahera data

Det enklaste sättet att extrahera spatiala data ur databasen är att använda en SQL `SELECT-fråga` för att definiera den datauppsättning som ska extraheras och dumpa de resulterande kolumnerna i en textfil som kan analyseras:

Code

```
SELECT road_id,road_geom AS geom, road_name FROM roads;
```

Output

```
road_id | geom                                      | road_name
--------+-------------------------------------------+-----------
      1 | LINESTRING(191232 243118,191108 243242) | Jeff Rd
      2 | LINESTRING(189141 244158,189265 244817) | Geordie Rd
      3 | LINESTRING(192783 228138,192612 229814) | Paul St
      4 | LINESTRING(189412 252431,189631 259122) | Graeme Ave
      5 | LINESTRING(190131 224148,190871 228134) | Phil Tce
      6 | LINESTRING(198231 263418,198213 268322) | Dave Cres
      7 | LINESTRING(218421 284121,224123 241231) | Chris Way
(6 rows)
```

Det kommer att finnas tillfällen då någon form av begränsning är nödvändig för att minska antalet poster som returneras. När det gäller attributbaserade begränsningar använder du samma SQL-syntax som för en icke spatial tabell. När det gäller spatiala begränsningar är följande funktioner användbara:

`ST_Intersects` Denna funktion visar om två geometrier delar på något utrymme.

= Detta testar om två geometrier är geometriskt identiska. Till exempel om POLYGON((0 0,1 1,1 0,0 0))”är samma sak som POLYGON((0 0,1 1,1 0,0 0))”(det är det).

Därefter kan du använda dessa operatorer i frågor. Observera att när du anger geometrier och boxar på SQL-kommandoraden måste du uttryckligen omvandla strängrepresentationerna till geometrier-funktionen. 312 är ett fiktivt spatialt referenssystem som matchar våra data. Så till exempel:

Code

```
SELECT road_id, road_name
FROM roads
WHERE roads_geom='SRID=312;LINESTRING(191232 243118,191108 243242)'::geometry;
```

Ovanstående fråga skulle returnera den enda post från tabellen ROADS_GEOMdär geometrin var lika med det värdet.

För att kontrollera om någon av vägarna passerar inom det område som definieras av en polygon:

Code

```
SELECT road_id, road_name
FROM roads
WHERE ST_Intersects(roads_geom, 'SRID=312;POLYGON((...))');
```

Den vanligaste spatiala frågan kommer förmodligen att vara en rambaseradfråga som används av klientprogramvara, t.ex. webbläsare och webbmappare, för att hämta en ”kartrammed data för visning.

När du använder operatorn ”&&”kan du ange antingen en BOX3D som jämförelsefunktion eller en GEOMETRY. När du anger en GEOMETRY kommer dock dess bounding box att användas för jämförelsen.

Om du använder ett BOX3D-objekt som ram ser en sådan fråga ut så här:

Code

```
SELECT roads_geom AS geom
FROM roads
WHERE
  roads_geom && ST_MakeEnvelope(191232, 243117,191232, 243119,312);
```

Notera användningen av SRID 312 för att specificera kuvertets projektion.

4.8.2 Använda Shapefile Dumper

Tabelldumpern `pgsql2shp` ansluter till databasen och konverterar en tabell (eventuellt definierad av en fråga) till en shape-fil. Den grundläggande syntaxen är:

Code

```
pgsql2shp [<options
>] <database
> [<schema
>.]<table>
```

Code

```
pgsql2shp [<options
>] <database
> <query>
```

Alternativen för kommandoraden är:

-f <filename> Skriv utdata till ett visst filnamn.

-h <host> Databasens värd att ansluta till.

-p <port> Porten att ansluta till på databasens värd.

-P <password> Det lösenord som ska användas vid anslutning till databasen.

-u <user> Det användarnamn som ska användas vid anslutning till databasen.

-g <geometry column> I tabeller med flera geometrikolumner, den geometrikolumn som ska användas när shape-filen skrivs.

-b Använd en binär markör. Detta gör operationen snabbare, men fungerar inte om något attribut i tabellen som inte är geometriskt saknar en cast till text.

-r Rått läge. Släpp inte fältet `gid` eller escape kolumnnamn.

-m filename Omvandla identifierare till namn med tio tecken. Filens innehåll består av rader med två symboler åtskilda av ett enda vitt mellanslag utan vare sig efterföljande eller inledande mellanslag: VERYLONGSYMBOL SHORTONE ANOTHERVERYLONGSYMBOL SHORTER etc.

4.9 Spatiala index

Spatiala index gör det möjligt att använda en spatial databas för stora datamängder. Utan indexering kräver en sökning efter funktioner en sekventiell genomsökning av varje post i databasen. Indexering snabbar upp sökningen genom att organisera data i en struktur som snabbt kan genomgås för att hitta matchande poster.

Den indexmetod med B-träd som vanligen används för attributdata är inte särskilt användbar för spatiala data, eftersom den endast stöder lagring och sökning av data i en enda dimension. Data som geometri (som har två eller flera dimensioner) kräver en indexmetod som stöder intervallfråga över alla datadimensioner. En av de viktigaste fördelarna med PostgreSQL för spatial datahantering är att den erbjuder flera typer av indexmetoder som fungerar bra för flerdimensionella data: GiST-, BRIN- och SP-GiST-index.

- GiST-index **(Generalized Search Tree)** delar upp data i saker på ena sidan”, saker som överlappar”, saker som är inutiöch kan användas på ett brett spektrum av datatyper, inklusive GIS-data. PostGIS använder ett R-Tree-index som implementerats ovanpå GiST för att indexera spatiala data. GiST är den vanligaste och mest mångsidiga metoden för spatiala index och erbjuder mycket bra prestanda vid sökningar.
- **BRIN-index (Block Range Index)** fungerar genom att sammanfatta den spatiala omfattningen av intervall av tabellposter. Sökningen görs via en skanning av intervallen. BRIN är endast lämpligt att använda för vissa typer av data (spatialt sorterade, med sällsynta eller inga uppdateringar). Men det ger mycket snabbare tid för att skapa index och mycket mindre indexstorlek.
- **SP-GiST (Space-Partitioned Generalized Search Tree)** är en generisk indexmetod som stöder partitionerade sökträd som quad-trees, k-d-träd och radix-träd (tries).

Spatiala index lagrar endast geometriernas avgränsande box. Spatiala frågor använder indexet som ett **primärt filter** för att snabbt fastställa en uppsättning geometrier som potentiellt matchar frågevillkoret. De flesta spatiala frågor kräver ett **sekundärt filter** som använder en spatial predikatfunktion för att testa ett mer specifikt spatialt villkor. Mer information om hur du ställer frågor med spatiala predikat finns på Avsnitt 5.2.

Se även PostGIS Workshop-avsnittet om spatiala index och PostgreSQL-handboken.

4.9.1 GiST-index

GiST står för Generalized Search Treeöch är en generisk form av indexering för flerdimensionella data. PostGIS använder ett R-Tree-index som implementerats ovanpå GiST för att indexera spatiala data. GiST är den vanligaste och mest mångsidiga metoden för spatial indexering och erbjuder mycket bra sökprestanda. Andra implementeringar av GiST används för att påskynda sökningar på alla typer av oregelbundna datastrukturer (heltalsarrayer, spektraldata etc.) som inte är mottagliga för normal B-Tree-indexering. För mer information se PostgreSQL-manualen.

När en spatial datatabell överstiger några tusen rader vill du bygga ett index för att påskynda spatiala sökningar av data (såvida inte alla dina sökningar baseras på attribut, i vilket fall du vill bygga ett normalt index på attributfälten).

Syntaxen för att bygga ett GiST-index på en geometri-kolumn är följande:

Code

```
CREATE INDEX [indexname] ON [tablename] USING GIST ( [geometryfield]);
```

Syntaxen ovan skapar alltid ett 2D-index. Om du vill få ett n-dimensionellt index för geometritypen kan du skapa ett sådant med denna syntax:

Code

```
CREATE INDEX [indexname] ON [tablename] USING GIST ([geometryfield] gist_geometry_ops_nd);
```

Att bygga ett spatialt index är en beräkningsintensiv övning. Det blockerar också skrivåtkomst till din tabell under den tid det skapas, så på ett produktionssystem kanske du vill göra det på ett långsammare CONCURRENTLY-medvetet sätt:

Code

```
CREATE INDEX CONCURRENTLY [indexname] ON [tablename] USING GIST ( [geometryfield]);
```

Efter att ha byggt ett index är det ibland bra att tvinga PostgreSQL att samla in tabellstatistik, som används för att optimera frågeplaner:

Code

```
VACUUM ANALYZE [table_name] [(column_name)];
```

4.9.2 BRIN-index

BRIN stands for Block Range Index”. It is a general-purpose index method provided by PostgreSQL. BRIN is a *lossy* index method, meaning that a secondary check is required to confirm that a record matches a given search condition (which is the case for all provided spatial indexes). It provides much faster index creation and much smaller index size, with reasonable read performance. Its primary purpose is to support indexing very large tables on columns which have a correlation with their physical location within the table. In addition to spatial indexing, BRIN can speed up searches on various kinds of attribute data structures (integer, arrays etc). For more information see the PostgreSQL manual.

När en spatial tabell överstiger några tusen rader vill du bygga ett index för att påskynda spatiala sökningar av data. GiST-index är mycket effektiva så länge deras storlek inte överstiger mängden RAM-minne som är tillgängligt för databasen och så länge du har råd med indexets lagringsstorlek och kostnaden för indexuppdatering vid skrivning. I annat fall kan BRIN-index betraktas som ett alternativ för mycket stora tabeller.

Ett BRIN-index lagrar den avgränsande box som omsluter alla geometrier som finns i raderna i en sammanhängande uppsättning tabellblock, ett så kallat *blockintervall*. När en fråga körs med hjälp

av indexet skannas blockområdena för att hitta dem som skär frågans utsträckning. Detta är effektivt endast om data är fysiskt ordnade så att blockintervallens avgränsande rutor har minimal överlappning (och helst är ömsesidigt uteslutande). Det resulterande indexet är mycket litet i storlek, men har vanligtvis sämre läsprestanda än ett GiST-index över samma data.

Att bygga ett BRIN-index är mycket mindre CPU-intensivt än att bygga ett GiST-index. Det är vanligt att ett BRIN-index är tio gånger snabbare att bygga än ett GiST-index över samma data. Och eftersom ett BRIN-index bara lagrar en bounding box för varje intervall av tabellblock, är det vanligt att det använder upp till tusen gånger mindre diskutrymme än ett GiST-index.

Du kan välja hur många block som ska sammanfattas i ett intervall. Om du minskar detta antal blir indexet större, men ger förmodligen bättre prestanda.

För att BRIN ska vara effektivt bör tabelldata lagras i en fysisk ordning som minimerar mängden överlappning av blockutbredningen. Det kan hända att data redan är sorterade på lämpligt sätt (t.ex. om de har laddats från en annan dataset som redan är sorterad i spatial ordning). Annars kan detta åstadkommas genom att sortera data efter en endimensionell spatial nyckel. Ett sätt att göra detta är att skapa en ny tabell sorterad efter geometrivärdena (som i de senaste PostGIS-versionerna använder en effektiv Hilbertkurvor-ordning):

Code

```
CREATE TABLE table_sorted AS
SELECT * FROM table  ORDER BY geom;
```

Alternativt kan data sorteras på plats genom att använda en GeoHash som ett (tillfälligt) index och klustra på det indexet:

Code

```
CREATE INDEX idx_temp_geohash ON table
    USING btree (ST_GeoHash(ST_Transform(geom, 4326 ), 20));
CLUSTER table USING idx_temp_geohash;
```

Syntaxen för att bygga ett BRIN-index på en `geometrikolumn` är:

Code

```
CREATE INDEX [indexname] ON [tablename] USING BRIN ( [geome_col]);
```

Syntaxen ovan skapar ett 2D-index. Om du vill bygga ett 3D-dimensionellt index använder du den här syntaxen:

Code

```
CREATE INDEX [indexname] ON [tablename]
USING BRIN ([geome_col] brin_geometry_inclusion_ops_3d);
```

Du kan också få ett 4D-dimensionellt index med hjälp av operatorsklassen 4D:

Code

```
CREATE INDEX [indexname] ON [tablename]
USING BRIN ([geome_col] brin_geometry_inclusion_ops_4d);
```

Ovanstående kommandon använder standardantalet block i ett intervall, vilket är 128. Om du vill ange antalet block som ska sammanfattas i ett intervall använder du följande syntax

Code

```
CREATE INDEX [indexname] ON [tablename]
USING BRIN ( [geome_col] ) WITH (pages_per_range = [number]);
```

Tänk på att ett BRIN-index bara lagrar en indexpost för ett stort antal rader. Om din tabell lagrar geometrier med ett blandat antal dimensioner är det troligt att det resulterande indexet kommer att ha dålig prestanda. Du kan undvika denna prestandaförlust genom att välja den operatorsklass som har minst antal dimensioner för de lagrade geometrierna

Datatypen `geography` stöds för BRIN-indexering. Syntaxen för att bygga ett BRIN-index på en geografikolumn är:

Code

```
CREATE INDEX [indexname] ON [tablename] USING BRIN ( [geog_col]);
```

Syntaxen ovan bygger upp ett 2D-index för geospatiala objekt på sfäroiden.

För närvarande finns endast ”inclusion support”, vilket innebär att endast operatorerna &&, ~ och @ kan användas för 2D-fall (för både `geometri` och `geografi`), och endast operatorn &&& för 3D-geometrier. Det finns för närvarande inget stöd för kNN-sökningar.

En viktig skillnad mellan BRIN och andra indextyper är att databasen inte underhåller indexet dynamiskt. Ändringar av spatiala data i tabellen läggs helt enkelt till i slutet av indexet. Detta leder till att indexets sökprestanda försämras med tiden. Indexet kan uppdateras genom att utföra en `VACUUM` eller genom att använda en specialfunktion `brin_summarize_new_values(regclass)`. Av denna anledning kan BRIN vara lämpligast att använda med data som är skrivskyddade eller endast sällan ändras. För mer information se manualen.

Sammanfattningsvis använder vi BRIN för spatiala data:

- Indexet byggs upp mycket snabbt och indexets storlek är mycket liten.
- Indexfrågetiden är långsammare än för GiST, men kan fortfarande vara mycket acceptabel.
- Kräver att tabelldata sorteras i en spatial ordning.
- Kräver manuellt indexunderhåll.
- Lämpar sig bäst för mycket stora tabeller, med låg eller ingen överlappning (t.ex. punkter), som är statiska eller ändras sällan.
- Mer effektiv för frågor som returnerar ett relativt stort antal dataposter.

4.9.3 SP-GiST-index

SP-GiST står för Space-Partitioned Generalized Search Treeöch är en generisk form av indexering för flerdimensionella datatyper som stöder partitionerade sökträd, t.ex. quad-trees, k-d trees och radix trees (tries). Gemensamt för dessa datastrukturer är att de upprepade gånger delar upp sökrymden i partitioner som inte behöver vara lika stora. Förutom spatial indexering används SP-GiST för att påskynda sökningar på många typer av data, till exempel telefonrouting, ip-routing, substringsökning etc. För mer information se PostgreSQL-manualen.

Liksom GiST-index är SP-GiST-index förlustbringande, i den meningen att de lagrar den avgränsande box som omger spatiala objekt. SP-GiST-index kan betraktas som ett alternativ till GiST-index.

När en GIS-datatabell överstiger några tusen rader kan ett SP-GiST-index användas för att påskynda spatiala sökningar av data. Syntaxen för att bygga ett SP-GiST-index på en geometri-kolumn är följande:

Code

```
CREATE INDEX [indexname] ON [tablename] USING SPGIST ( [geometryfield]);
```

Syntaxen ovan bygger upp ett 2-dimensionellt index. Ett 3-dimensionellt index för geometritypen kan skapas med hjälp av operatorsklassen 3D:

Code

```
CREATE INDEX [indexname] ON [tablename] USING SPGIST ([geometryfield] ↩
    spgist_geometry_ops_3d);
```

Att bygga ett spatialt index är en beräkningsintensiv operation. Det blockerar också skrivåtkomst till din tabell under den tid det skapas, så på ett produktionssystem kanske du vill göra det på ett långsammare CONCURRENTLY-medvetet sätt:

Code

```
CREATE INDEX CONCURRENTLY [indexname] ON [tablename] USING SPGIST ( [geometryfield]);
```

Efter att ha byggt ett index är det ibland bra att tvinga PostgreSQL att samla in tabellstatistik, som används för att optimera frågeplaner:

Code

```
VACUUM ANALYZE [table_name] [(column_name)];
```

Ett SP-GiST-index kan påskynda frågor som involverar följande operatorer:

- <<, &<, &>, >>, <<|, &<|, |&>, |>>, &&, @>, <@, och ~=, för 2-dimensionella index,
- &/&, ~==, @>>, och <<@, för 3-dimensionella index.

Det finns för närvarande inget stöd för kNN-sökningar.

4.9.4 Anpassning av indexanvändning

Vanligtvis påskyndar index osynligt dataåtkomst: när ett index har byggts bestämmer PostgreSQL-frågeplaneraren automatiskt när den ska användas för att förbättra frågeprestanda. Men det finns vissa situationer där planeraren inte väljer att använda befintliga index, så frågor slutar använda långsamma sekventiella skanningar istället för ett spatialt index.

Om du upptäcker att dina spatiala index inte används finns det några saker du kan göra:

- Granska frågeplanen och kontrollera att din fråga faktiskt beräknar det du behöver. En felaktig JOIN, antingen bortglömd eller till fel tabell, kan oväntat hämta tabellposter flera gånger. För att få fram frågeplanen kör du med `EXPLAIN` framför frågan.
- Se till att statistik samlas in om antalet och fördelningen av värden i en tabell, så att frågeplaneraren får bättre information för att fatta beslut om indexanvändning. **VACUUM ANALYZE** beräknar båda.

  Du bör regelbundet dammsuga dina databaser ändå. Många PostgreSQL DBA: er kör **VACUUM** som ett cron-jobb utanför topparna regelbundet.
- Om det inte hjälper att dammsuga kan du tillfälligt tvinga planeraren att använda indexinformationen genom att använda kommandot **SET ENABLE_SEQSCAN TO OFF;.** På så sätt kan du kontrollera om planeraren överhuvudtaget kan generera en indexaccelererad frågeplan för din fråga. Du bör bara använda det här kommandot för felsökning; i allmänhet vet planeraren bättre än du när index ska användas. När du har kört din fråga, glöm inte att köra **SET ENABLE_SEQSCAN TO ON;** så att planeraren fungerar normalt för andra frågor.

- Om **SET ENABLE_SEQSCAN TO OFF;** hjälper din fråga att köras snabbare är din Postgres sannolikt inte inställd för din maskinvara. Om du tycker att planeraren har fel om kostnaden för sekventiella skanningar jämfört med indexskanningar kan du försöka minska värdet på `RANDOM_PAGE_COST` i `postgresql.conf` eller använda **SET RANDOM_PAGE_COST TO 1.1;.** Standardvärdet för `RANDOM_PAGE_` är 4,0. Prova att ställa in det på 1,1 (för SSD) eller 2,0 (för snabba magnetiska diskar). Om du sänker värdet blir det mer sannolikt att planeraren använder indexskanningar.
- Om **SET ENABLE_SEQSCAN TO OFF;** inte hjälper din fråga kan det bero på att frågan använder en SQL construct som Postgres-planeraren ännu inte kan optimera. Det kan vara möjligt att skriva om frågan på ett sätt som planeraren kan hantera. Till exempel kanske en underfråga med en inline SELECT inte ger en effektiv plan, men kan eventuellt skrivas om med hjälp av en LATERAL JOIN.

Mer information finns i Postgres-manualens avsnitt om Query Planning.

Kapitel 5

Spatiala frågor

Syftet med spatiala databaser *är* att kunna utföra sökningar i databasen som normalt kräver GIS-funktioner på skrivbordet. För att PostGIS ska kunna användas effektivt måste man veta vilka spatiala funktioner som finns tillgängliga, hur de ska användas i frågor och se till att lämpliga index finns på plats för att ge bra prestanda.

5.1 Fastställande av spatiala relationer

Spatiala relationer anger hur två geometrier interagerar med varandra. De är en grundläggande egenskap för att ställa frågor om geometri.

5.1.1 Dimensionsutvidgad 9-intersektionell modell

Enligt OpenGIS Simple Features Implementation Specification for SQL är det grundläggande tillvägagångssättet för att jämföra två geometrier att göra parvisa tester av skärningarna mellan de två geometriernas inre, yttre och yttre delar och att klassificera förhållandet mellan de två geometrierna baserat på posterna i den resulterande 'skärningsmatrisen'"

I teorin om punktuppsättningstopologi kategoriseras punkterna i en geometri som är inbäddad i ett 2-dimensionellt rum i tre uppsättningar:

Begränsning

Gränsen för en geometri är uppsättningen av geometrier med nästa lägre dimension. För `POINTs`, som har dimensionen 0, är gränsen den tomma mängden. Gränsen för en `LINESTRING` är de två ändpunkterna. För `POLYGONER`är gränsen linjeverket för de yttre och inre ringarna.

Inredning

Det inre av en geometri är de punkter i en geometri som inte ligger i begränsningen. För `POINTs`är det inre själva punkten. Det inre av en `LINESTRING` är den uppsättning punkter som finns mellan ändpunkterna. För `POLYGONER`är det inre den areella ytan inuti polygonen.

Exteriör

En geometris utsida är resten av det rum i vilket geometrin är inbäddad; med andra ord alla punkter som inte ligger i geometrins inre eller på dess gräns. Det är en 2-dimensionell icke sluten yta.

DE-9IM (Dimensionally Extended 9-Intersection Model) beskriver det spatiala förhållandet mellan två geometrier genom att ange dimensionerna för de 9 skärningspunkterna mellan ovanstående uppsättningar för varje geometri. Dimensionerna på skärningspunkterna kan formellt representeras i en 3x3 **skärningsmatris**.

För en geometri *g* betecknas det *inre,* det *yttre och det yttre med* beteckningarna *I(g), B(g)* och *E(g)*. *Dim(s)* betecknar också dimensionen hos en mängd *s* med domänen `{0,1,2,F}`:

- `0` => punkt
- `1` => linje
- `2` => område
- `F` => tom uppsättning

Med hjälp av denna notation är intersektionsmatrisen för två geometrier *a* och *b:* är:

	Inredning	**Begränsning**	**Exteriör**
Inredning	*dim(I(a) ∩ I(b))*	*dim(I(a) ∩ B(b))*	*dim(I(a) ∩ E(b))*
Begränsning	*dim(B(a) ∩ I(b))*	*dim(B(a) ∩ B(b))*	*dim(B(a) ∩ E(b))*
Exteriör	*dim(E(a) ∩ I(b))*	*dim(E(a) ∩ B(b))*	*dim(E(a) ∩ E(b))*

The following overlapping polygons provide a concrete example. ST_Relate computes the matrix `212101212`; it matches the overlap pattern `T*T***T**` used by ST_Overlaps.

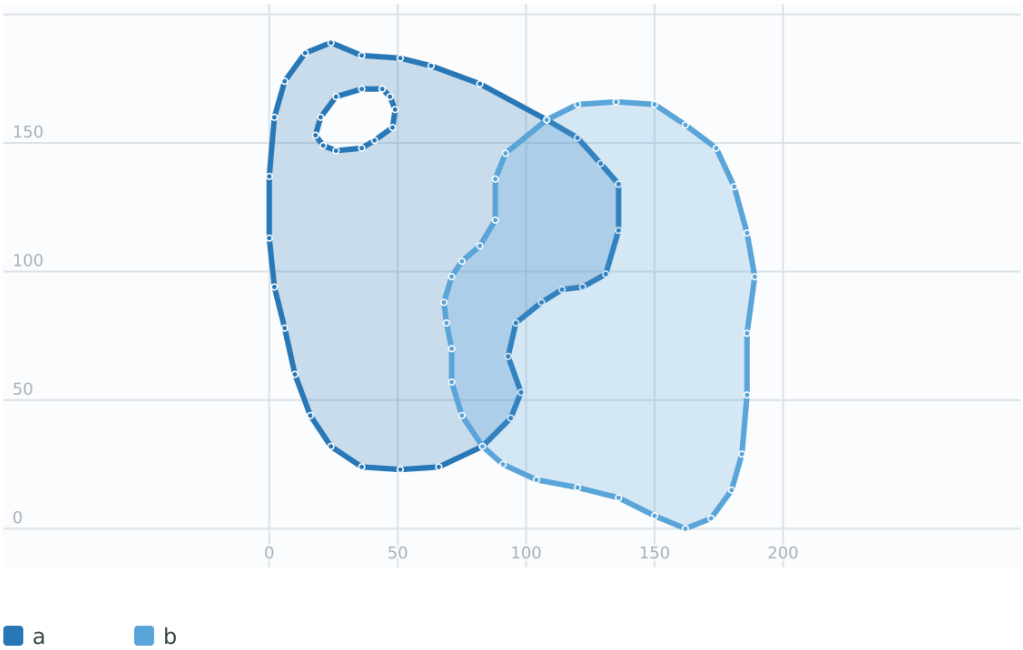
Code

```
WITH example AS (
  SELECT
    'POLYGON ((140 140,140 122,135 105,126 100,118 99,110 94,100 86,97 73,102 59,98 49,87 ↩
    38,70 30,55 29,40 30,28 38,20 50,14 66,10 84,6 100,4 119,4 143,6 166,10 180,18 191,28 ↩
    195,40 190,55 189,67 186,86 179,112 165,124 158,133 148,140 140),(48 177,40 177,30 ↩
    174,24 166,22 159,25 155,30 153,40 154,45 157,52 162,53 169,51 174,48 177))'::geometry ↩
    AS a,
    'POLYGON ((75 63,79 50,87 38,95 31,108 25,124 22,140 18,154 11,166 6,176 10,184 21,188 ↩
    35,190 58,190 82,193 104,190 121,185 139,178 154,166 163,154 171,139 172,124 171,112 ↩
    165,96 152,92 142,92 126,86 116,79 110,75 104,72 94,73 86,75 76,75 63))'::geometry AS b
), relation AS (
  SELECT a, b, ST_Relate(a, b) AS matrix
  FROM example
)
SELECT matrix,
       ST_RelateMatch(matrix, 'T*T***T**') AS matches_overlap_pattern,
       ST_Overlaps(a, b) AS overlaps
FROM relation;
```

Output

```
matrix   | matches_overlap_pattern | overlaps
-----------+-------------------------+----------
 212101212 | t                       | t
(1 row)
```

Figure



The executable query and generated overlay above are the visual source for the relation. The query below renders the nine cells of the same kind of matrix as a 3-by-3 set of examples. The final exterior/exterior cell is clipped to a finite viewport, since the true exterior/exterior intersection is unbounded.

Code

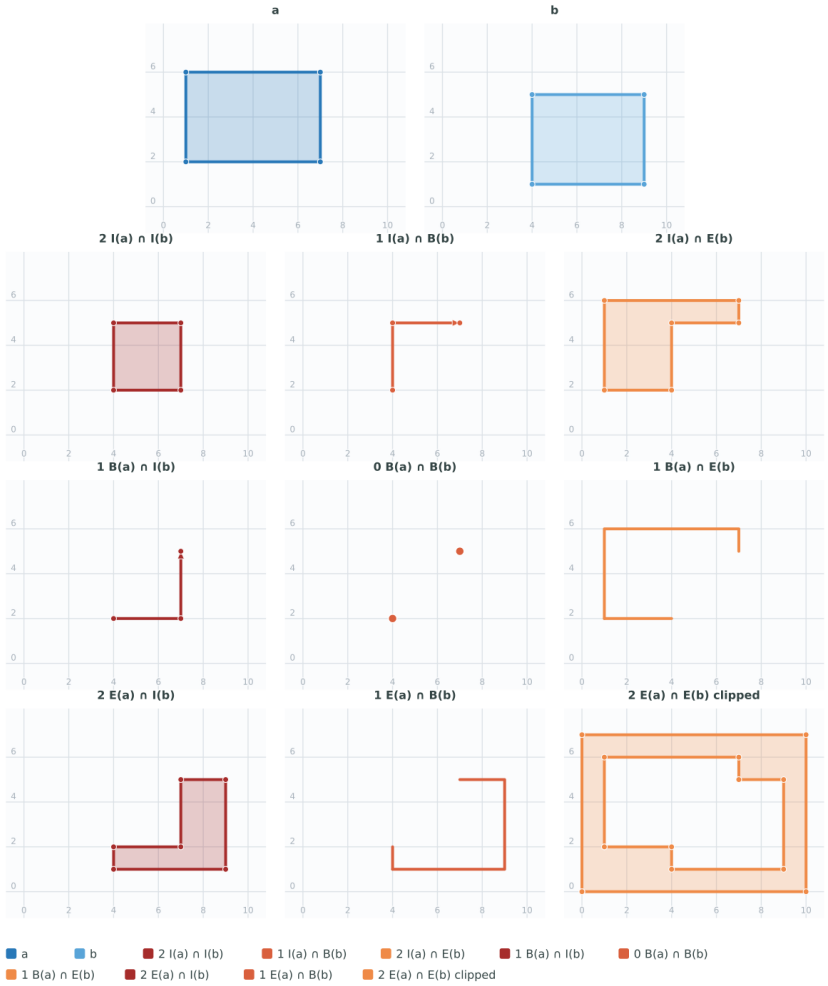
```
WITH example AS (
  SELECT
    ST_MakeEnvelope(0, 0, 6, 4) AS a,
    ST_MakeEnvelope(3, -1, 8, 3) AS b,
    ST_MakeEnvelope(-1, -2, 9, 5) AS viewport
), parts AS (
  SELECT
    a,
    b,
    ST_Boundary(a) AS ba,
    ST_Boundary(b) AS bb,
    viewport
  FROM example
)
SELECT
  ST_Relate(a, b) AS matrix,
  ST_AsText(ST_Normalize(a)) AS input_a,
  ST_AsText(ST_Normalize(b)) AS input_b,
  ST_AsText(ST_Normalize(ST_Intersection(a, b))) AS "2 I(a) b''ոb'' I(b)",
  ST_AsText(ST_Normalize(ST_Intersection(a, bb))) AS "1 I(a) b''ոb'' B(b)",
  ST_AsText(ST_Normalize(ST_Difference(a, b))) AS "2 I(a) b''ոb'' E(b)",
  ST_AsText(ST_Normalize(ST_Intersection(ba, b))) AS "1 B(a) b''ոb'' I(b)",
  ST_AsText(ST_Normalize(ST_Intersection(ba, bb))) AS "0 B(a) b''ոb'' B(b)",
  ST_AsText(ST_Normalize(ST_Collect(
      'LINESTRING(0 0,0 4,6 4,6 3)'::geometry,
      'LINESTRING(0 0,3 0)'::geometry
  ))) AS "1 B(a) b''ոb'' E(b)",
  ST_AsText(ST_Normalize(ST_Difference(b, a))) AS "2 E(a) b''ոb'' I(b)",
  ST_AsText(ST_Normalize(ST_Collect(
```

```
      'LINESTRING(3 -1,8 -1,8 3,6 3)'::geometry,
      'LINESTRING(3 -1,3 0)'::geometry
  ))) AS "1 E(a) b''nb'' B(b)",
  ST_AsText(ST_Normalize(ST_Difference(viewport, ST_Union(a, b))))
    AS "2 E(a) b''nb'' E(b) clipped"
FROM parts;
```

Output

```
-[ RECORD 1 ]----------+--------------------------------------------------------------------------------
matrix                 | 212101212
input_a                | POLYGON((0 0,0 4,6 4,6 0,0 0))
input_b                | POLYGON((3 -1,3 3,8 3,8 -1,3 -1))
2 I(a) b''nb'' I(b)          | POLYGON((3 0,3 3,6 3,6 0,3 0))
1 I(a) b''nb'' B(b)          | LINESTRING(3 0,3 3,6 3)
2 I(a) b''nb'' E(b)          | POLYGON((0 0,0 4,6 4,6 3,3 3,3 0,0 0))
1 B(a) b''nb'' I(b)          | LINESTRING(3 0,6 0,6 3)
0 B(a) b''nb'' B(b)          | MULTIPOINT((6 3),(3 0))
1 B(a) b''nb'' E(b)          | MULTILINESTRING((0 0,0 4,6 4,6 3),(0 0,3 0))
2 E(a) b''nb'' I(b)          | POLYGON((3 -1,3 0,6 0,6 3,8 3,8 -1,3 -1))
1 E(a) b''nb'' B(b)          | MULTILINESTRING((3 -1,8 -1,8 3,6 3),(3 -1,3 0))
2 E(a) b''nb'' E(b) clipped  | POLYGON((-1 -2,-1 5,9 5,9 -2,-1 -2),(0 0,3 0,3 -1,8 -1,8 3,6 3,6 4,0 4,0 0))
```

Figure



När man läser från vänster till höger och uppifrån och ner representeras skärningsmatrisen av text-strängen '**"212101212".**'.

För mer information, se:

- OpenGIS Simple Features Implementation Specification för SQL (version 1.1, avsnitt 2.1.13.2)

- Wikipedia: Dimensionellt utökad modell med nio intersektioner (DE-9IM)
- GeoTools: Punktuppsättningsteori och DE-9IM-matrisen

5.1.2 Namngivna spatiala relationer

För att göra det enkelt att fastställa vanliga spatiala relationer definierar OGC SFS en uppsättning *namngivna predikat för spatiala relationer*. PostGIS tillhandahåller dessa som funktionerna ST_Contains, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within. Det definierar också de icke-standardiserade relationspredikaten ST_Covers, ST_CoveredBy, och ST_ContainsProperly.

Spatiala predikat används vanligtvis som villkor i SQL `WHERE-` eller `JOIN-klausuler`. De namngivna spatiala predikaten använder automatiskt ett spatialt index om ett sådant finns tillgängligt, så det finns inget behov av att använda bounding box-operatorn && också. Till exempel:

Code

```
SELECT city.name, state.name, city.geom
FROM city JOIN state ON ST_Intersects(city.geom, state.geom);
```

För mer information och illustrationer, se PostGIS Workshop.

5.1.3 Allmänna spatiala relationer

In some cases the named spatial relationships are insufficient to provide a desired spatial filter condition. These requirements can be expressed by computing the full DE-9IM intersection matrix with ST_Relate.

För att testa ett visst spatialt förhållande används ett **intersektionsmatrismönster**. Detta är matrisrepresentationen utökad med tilläggssymbolerna `{T,*}`:

- `T` => intersection dimension is non-empty; i.e. is in `{0,1,2}`
- `*` => bryr sig inte

Using intersection matrix patterns, specific spatial relationships can be evaluated succinctly. The ST_Relate and ST_RelateMatch functions can both test these patterns.

These road segments share a line segment. ST_Crosses does not find this relationship for linear features, because it only returns `true` when their interiors intersect at a point. The pattern `1*1***1**` tests for a one-dimensional intersection instead.

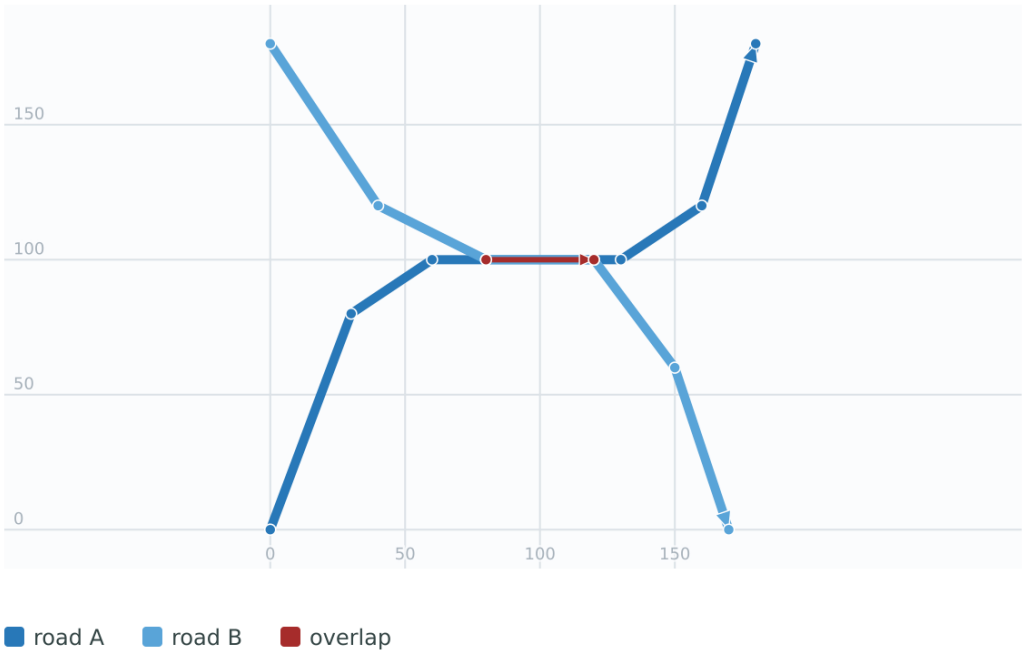
Code

```
WITH roads AS (
  SELECT
    'LINESTRING(10 10,40 90,70 110,140 110,170 130,190 190)'::geometry AS "road A",
    'LINESTRING(10 190,50 130,90 110,130 110,160 70,180 10)'::geometry AS "road B"
), relation AS (
  SELECT "road A", "road B", ST_Relate("road A", "road B") AS matrix
  FROM roads
)
SELECT
  ST_Intersection("road A", "road B") AS overlap,
  matrix,
  ST_Crosses("road A", "road B") AS "ST_Crosses",
  ST_RelateMatch(matrix, '1*1***1**') AS "matches 1*1***1**"
FROM relation;
```

Output

```
overlap           |  matrix   | ST_Crosses | matches 1*1***1**
--------------------------+-----------+------------+-------------------
 LINESTRING(90 110,130 110) | 1F1FF0102 | f          | t
(1 row)
```

Figure



The next example finds a wharf which runs from the lake interior onto its shoreline, with one endpoint on the shoreline. The other wharves provide nearby counterexamples in the figure. The pattern `102101FF2` expresses the complete required relationship in one test.

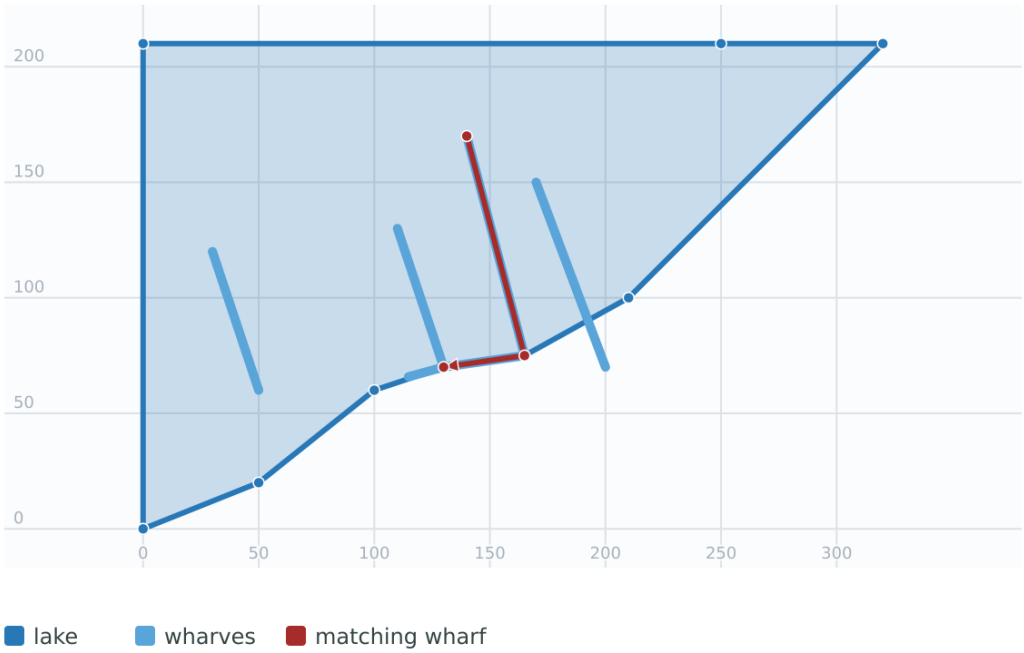
Code

```
WITH data AS (
  SELECT
    'POLYGON((-20 10,30 30,80 70,110 80,145 85,190 110,300 220,230 220,-20 220,-20 10))':: ↩
   geometry AS lake,
    'MULTILINESTRING((120 180,145 85,110 80),(10 130,30 70),(90 140,110  ↩
   80,94.81118881118876 75.83496503496498),(150 160,180 80))'::geometry AS wharves
), wharf AS (
  SELECT lake, (item).path[1] AS id, (item).geom
  FROM data CROSS JOIN LATERAL ST_Dump(wharves) AS item
), relation AS (
  SELECT id, geom, ST_Relate(lake, geom) AS matrix
  FROM wharf
)
SELECT
  id AS wharf,
  geom AS "matching wharf",
  matrix
FROM relation
WHERE ST_RelateMatch(matrix, '102101FF2')
ORDER BY id;
```

Output

```
wharf |           matching wharf           |  matrix
------+-------------------------------------+-----------
     1 | LINESTRING(120 180,145 85,110 80) | 102101FF2
(1 row)
```

Figure



5.2 Använda spatiala index

När du konstruerar frågor med spatiala villkor är det viktigt att se till att ett spatialt index används, om ett sådant finns (se Avsnitt 4.9), för bästa prestanda. För att göra detta måste en spatial operator eller en indexmedveten funktion användas i en `WHERE-` eller `ON-sats` i frågan.

Spatiala operatorer inkluderar bounding box-operatorer (av vilka den vanligaste är &&; se Avsnitt 7.10.1 för en fullständig lista) och avståndsoperatorer som används i frågor om närmaste granne (den vanligaste är <->; se Avsnitt 7.10.2 för en fullständig lista)

Indexmedvetna funktioner lägger automatiskt till en bounding box-operator till det spatiala villkoret. Indexmedvetna funktioner inkluderar de namngivna spatiala relationspredikaten ST_Contains, ST_ContainsProperly, ST_CoveredBy, ST_Covers, ST_Crosses, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within, ST_Within, och ST_3DIntersects, och avståndspredikaten ST_DWithin, ST_DFullyWithin, ST_3DDFullyWithin, och ST_3DDWithin.)

Funktioner som ST_Distance använder *inte* index för att optimera sin drift. Till exempel skulle följande fråga vara ganska långsam på en stor tabell:

Code

```
SELECT geom
FROM geom_table
WHERE ST_Distance(geom, 'SRID=312;POINT(100000 200000)') < 100
```

Denna fråga väljer ut alla geometrier i `geom_table` som ligger inom 100 enheter från punkten (100000, 200000). Den kommer att vara långsam eftersom den beräknar avståndet mellan varje punkt i tabellen och den angivna punkten, dvs. en `ST_Distance()` -beräkning görs för **varje** rad i tabellen.

Antalet rader som bearbetas kan minskas avsevärt genom att använda den indexmedvetna funktionen ST_DWithin:

Code

```
SELECT geom
FROM geom_table
WHERE ST_DWithin(geom, 'SRID=312;POINT(100000 200000)', 100)
```

Denna fråga väljer samma geometrier, men den gör det på ett effektivare sätt. Detta möjliggörs genom att `ST_DWithin()` använder `&&-operatorn` internt på en utökad begränsningsbox för frågegegeometrin. Om det finns ett spatialt index på `geom` kommer frågeplaneraren att inse att den kan använda indexet för att minska antalet rader som skannas innan avståndet beräknas. Det spatiala indexet gör det möjligt att endast hämta poster med geometrier vars avgränsande rutor överlappar den expanderade omfattningen och som därför *kan* ligga inom det avstånd som krävs. Det faktiska avståndet beräknas sedan för att bekräfta om posten ska inkluderas i resultatuppsättningen.

För mer information och exempel, se PostGIS Workshop.

5.3 Exempel på Spatial SQL

I exemplen i detta avsnitt används en tabell med linjära vägar och en tabell med polygonala kommungränser. Definitionen av tabellen `bc_roads` är:

Output

```
Column    | Type              | Description
----------+-------------------+-------------------
gid       | integer           | Unique ID
name      | character varying | Road Name
geom      | geometry          | Location Geometry (Linestring)
```

Definitionen av tabellen `bc_municipality` är:

Output

```
Column   | Type              | Description
---------+-------------------+-------------------
gid      | integer           | Unique ID
code     | integer           | Unique ID
name     | character varying | City / Town Name
geom     | geometry          | Location Geometry (Polygon)
```

1. *Vad är den totala längden på alla vägar, uttryckt i kilometer?*

   Du kan besvara den här frågan med ett mycket enkelt SQL-test:

   Code

   ```
   SELECT sum(ST_Length(geom))/1000 AS km_roads FROM bc_roads;
   ```

   Output

   ```
   km_roads
   ------------------
   70842.1243039643
   ```

2. *Hur stor är staden Prince George, räknat i hektar?*

   Denna fråga kombinerar ett attributvillkor (på kommunens namn) med en spatial beräkning (av polygonområdet):

Code

```
SELECT
  ST_Area(geom)/10000 AS hectares
FROM bc_municipality
WHERE name = 'PRINCE GEORGE';
```

Output

```
hectares
------------------
32657.9103824927
```

3. *Vilken är den största kommunen i provinsen, sett till ytan?*

   Denna fråga använder en spatial mätning som ett ordningsvärde. Det finns flera sätt att närma sig detta problem, men det mest effektiva är nedan:

   Code

   ```
   SELECT
     name,
     ST_Area(geom)/10000 AS hectares
   FROM bc_municipality
   ORDER BY hectares DESC
   LIMIT 1;
   ```

   Output

   ```
   name           | hectares
   ---------------+-----------------
   TUMBLER RIDGE  | 155020.02556131
   ```

   Observera att vi måste beräkna arean för varje polygon för att kunna besvara den här frågan. Om vi gjorde det här mycket skulle det vara vettigt att lägga till en områdeskolumn i tabellen som kan indexeras för prestanda. Genom att beställa resultaten i fallande riktning och använda PostgreSQL-kommandot LIMIT”kan vi enkelt välja bara det största värdet utan att använda en aggregerad funktion som MAX ().

4. *Hur långa är de vägar som helt och hållet ingår i varje kommun?*

   Detta är ett exempel på en spatial join”, som sammanför data från två tabeller (med en join) med hjälp av en spatial interaktion (”contained”) som join-villkor (i stället för den vanliga relationella metoden med join på en gemensam nyckel):

   Code

   ```
   SELECT
     m.name,
     sum(ST_Length(r.geom))/1000 as roads_km
   FROM bc_roads AS r
   JOIN bc_municipality AS m
     ON ST_Contains(m.geom, r.geom)
   GROUP BY m.name
   ORDER BY roads_km;
   ```

   Output

   ```
   name                         | roads_km
   -----------------------------+----------
   SURREY                       | 1539.476
   VANCOUVER                    | 1450.331
   LANGLEY DISTRICT             |  833.793
   ```

```
BURNABY                       |  773.769
PRINCE GEORGE                 |  694.376
...
```

Den här frågan tar ett tag, eftersom varje väg i tabellen sammanfattas i slutresultatet (ca 250 000 vägar för exempeltabellen). För mindre dataset (flera tusen poster på flera hundra) kan svaret vara mycket snabbt.

5. *Skapa en ny tabell med alla vägar inom staden Prince George.*

   Detta är ett exempel på en överlay", som tar in två tabeller och matar ut en ny tabell som består av spatialt klippta eller skurna resultat. Till skillnad från den spatial joinsom demonstreras ovan skapar den här frågan nya geometrier. En overlay är som en turboladdad spatial join och är användbar för mer exakt analysarbete:

   Code

```
CREATE TABLE pg_roads as
SELECT
  ST_Intersection(r.geom, m.geom) AS intersection_geom,
  ST_Length(r.geom) AS rd_orig_length,
  r.*
FROM bc_roads AS r
JOIN bc_municipality AS m
  ON ST_Intersects(r.geom, m.geom)
WHERE
  m.name = 'PRINCE GEORGE';
```

6. *Hur lång är Douglas St"i Victoria i kilometer?*

   Code

```
SELECT
  sum(ST_Length(r.geom))/1000 AS kilometers
FROM bc_roads r
JOIN bc_municipality m
  ON ST_Intersects(m.geom, r.geom
WHERE
  r.name = 'Douglas St'
  AND m.name = 'VICTORIA';
```

   Output

```
kilometers
------------------
4.89151904172838
```

7. *Vilken är den största kommunpolygonen som har ett hål?*

   Code

```
SELECT gid, name, ST_Area(geom) AS area
FROM bc_municipality
WHERE ST_NRings(geom)
> 1
ORDER BY area DESC LIMIT 1;
```

   Output

```
gid  | name         | area
-----+--------------+------------------
12   | SPALLUMCHEEN | 257374619.430216
```

Kapitel 6

Tips om prestanda

6.1 Små tabeller med stora geometrier

6.1.1 Beskrivning av problemet

Current PostgreSQL versions suffer from a query optimizer weakness regarding TOAST tables. TOAST tables are a kind of ”extension room”used to store large (in the sense of data size) values that do not fit into normal data pages (like long texts, images or complex geometries with lots of vertices), see the PostgreSQL Documentation for TOAST for more information).

Problemet uppstår om du råkar ha en tabell med ganska stora geometrier, men inte alltför många rader av dem (som en tabell som innehåller gränserna för alla europeiska länder i hög upplösning). Då är själva tabellen liten, men den använder massor av TOAST-utrymme. I vårt exempel hade tabellen i sig cirka 80 rader och använde bara 3 datasidor, men TOAST-tabellen använde 8225 sidor.

Ställ nu en fråga där du använder geometrioperatorn & & för att söka efter en avgränsningsbox som bara matchar mycket få av dessa rader. Nu ser frågeoptimeraren att tabellen bara har 3 sidor och 80 rader. Den uppskattar att en sekventiell skanning på en så liten tabell är mycket snabbare än att använda ett index. Och så beslutar den att ignorera GIST-indexet. Vanligtvis är denna uppskattning korrekt. Men i vårt fall måste operatorn hämta varje geometri från disken för att jämföra begränsningsrutorna, vilket innebär att alla TOAST-sidor också måste läsas.

För att se om du lider av den här frågan, använd kommandot ”EXPLAIN ANALYZEpostgresql. För mer information och de tekniska detaljerna kan du läsa tråden på PostgreSQL-prestandans e-postlista: http://archives.postgresql.org/pgsql-performance/2005-02/msg00030.php

och nyare tråd om PostGIS https://lists.osgeo.org/pipermail/postgis-devel/2017-June/026209.html

6.1.2 Lösningar

PostgreSQL-folket försöker lösa detta problem genom att göra frågeuppskattningen TOAST-medveten. För närvarande är här två lösningar:

Den första lösningen är att tvinga frågeplaneraren att använda indexet. Skicka SET enable_seqscan TO off;till servern innan du skickar frågan. Detta tvingar i princip frågeplaneraren att undvika sekventiella skanningar när det är möjligt. Så den använder GIST-indexet som vanligt. Men den här flaggan måste ställas in för varje anslutning, och det gör att frågeplaneraren gör felbedömningar i andra fall, så du bör SET enable_seqscan TO on;”efter frågan.

Den andra lösningen är att göra den sekventiella skanningen så snabb som frågeplaneraren tror. Detta kan uppnås genom att skapa en extra kolumn som ”cachelagrarbboxen och matchar mot denna. I vårt exempel ser kommandona ut så här:

Code

```
SELECT AddGeometryColumn('myschema', 'mytable', 'bbox', '4326', 'GEOMETRY', '2');
UPDATE mytable SET bbox = ST_Envelope(ST_Force2D(geom));
```

Ändra nu din fråga för att använda operatorn & & mot bbox istället för geom_column, som:

Code

```
SELECT geom_column
FROM mytable
WHERE bbox && ST_SetSRID('BOX3D(0 0,1 1)'::box3d,4326);
```

Om du ändrar eller lägger till rader i min tabell måste du naturligtvis hålla bbox synkroniserad”. Det mest transparenta sättet att göra detta skulle vara triggers, men du kan också ändra din applikation för att hålla bbox-kolumnen aktuell eller köra UPDATE-frågan ovan efter varje ändring.

6.2 CLUSTERING på geometriindex

The PostgreSQL CLUSTER command physically reorders table rows according to an index. For a mostly read-only table whose queries frequently use the same spatial index, this can reduce I/O and improve cache locality.

PostgreSQL supports clustering on GiST indices, including indices on nullable geometry columns. To cluster a table using a geometry index:

Code

```
CLUSTER my_table USING my_geom_index;
```

Output

```
CLUSTER
```

6.3 Undvik dimensionskonvertering

Ibland råkar du ha 3D- eller 4D-data i din tabell, men du kommer alltid åt dem med OpenGIS-kompatibla ST_AsText()- eller ST_AsBinary()-funktioner som bara matar ut 2D-geometrier. De gör detta genom att internt anropa funktionen ST_Force2D(), som medför en betydande overhead för stora geometrier. För att undvika denna overhead kan det vara möjligt att i förväg släppa dessa extra dimensioner en gång för alla:

Code

```
UPDATE mytable SET geom = ST_Force2D(geom);
VACUUM FULL ANALYZE mytable;
```

Observera att om du har lagt till din geometrikolumn med AddGeometryColumn() kommer det att finnas en begränsning för geometridimensionen. För att kringgå den måste du släppa begränsningen. Kom ihåg att uppdatera posten i geometry_columns-tabellen och återskapa begränsningen efteråt.

När det gäller stora tabeller kan det vara klokt att dela upp denna UPDATE i mindre delar genom att begränsa UPDATE till en del av tabellen via en WHERE-klausul och din primära nyckel eller andra genomförbara kriterier och köra en enkel ”VACUUM;mellan dina UPDATE. Detta minskar drastiskt behovet av tillfälligt diskutrymme. Om du har geometrier med blandade dimensioner kan du dessutom begränsa UPDATE med ÅHERE dimension(geom)>2öch slippa skriva om geometrier som redan finns i 2D.

Kapitel 7

PostGIS-referens

De funktioner som anges nedan är de som en användare av PostGIS sannolikt kommer att behöva. Det finns andra funktioner som är nödvändiga stödfunktioner för PostGIS-objekten och som inte är till nytta för en allmän användare.

Notera
PostGIS har påbörjat en övergång från den befintliga namngivningskonventionen till en SQL-MM-centrerad konvention. Som ett resultat av detta har de flesta av de funktioner som du känner till och älskar bytt namn med hjälp av standardprefixet för spatial typ (ST). Tidigare funktioner är fortfarande tillgängliga, men listas inte i detta dokument där uppdaterade funktioner är likvärdiga. De ST_-funktioner som inte listas i den här dokumentationen är föråldrade och kommer att tas bort i en framtida version, så SLUTA ANVÄNDA DEM.

7.1 PostGIS-datatyper Geometry/Geography/Box

7.1.1 box2d

box2d — Typ som representerar en 2-dimensionell avgränsande box.

Beskrivning

`box2d` är en spatial datatyp som används för att representera den tvådimensionella begränsningsbox som omsluter en geometri eller en samling geometrier. Aggregatfunktionen ST_Extent returnerar t.ex. ett `box2d-objekt`.

Representationen innehåller värdena `xmin, ymin, xmax, ymax`. Dessa är minimi- och maximivärdena för X- och Y-utsträckningarna.

`box2d`-objekt har en textrepresentation som ser ut som `BOX(1 2,5 6)`..

Casting-beteende

I denna tabell listas de automatiska och explicita casts som är tillåtna för denna datatyp:

Kasta till	Beteende
box3d	automatisk
geometri	automatisk

Se även

Avsnitt 13.7

7.1.2 box3d

box3d — Typ som representerar en 3-dimensionell avgränsande box.

Beskrivning

`box3d` är en spatial datatyp i PostGIS som används för att representera den tredimensionella begränsningsbox som omsluter en geometri eller en samling geometrier. Aggregatfunktionen ST_3DExtent returnerar till exempel ett `box3d-objekt`.

Representationen innehåller värdena `xmin, ymin, zmin, xmax, ymax, zmax`. Dessa är minimi- och maximivärdena för X-, Y- och Z-utsträckningarna.

`box3d`-objekt har en textrepresentation som ser ut som `BOX3D(1 2 3,5 6 5)`..

Casting-beteende

I denna tabell listas de automatiska och explicita casts som är tillåtna för denna datatyp:

Kasta till	Beteende
box	automatisk
box2d	automatisk
geometri	automatisk

Se även

Avsnitt 13.7

7.1.3 geometry

geometry — Den typ som representerar spatiala egenskaper med plana koordinatsystem.

Beskrivning

`geometry` är en grundläggande spatial datatyp i PostGIS som används för att representera en egenskap i plana (euklidiska) koordinatsystem.

Alla spatiala operationer på geometri använder enheterna i det spatiala referenssystem som geometrin befinner sig i.

Casting-beteende

I denna tabell listas de automatiska och explicita casts som är tillåtna för denna datatyp:

Kasta till	Beteende
box	automatisk
box2d	automatisk
box3d	automatisk
bytea	automatisk
geografi	automatisk
text	automatisk

Se även

Avsnitt 4.1, Avsnitt 13.3

7.1.4 geometry_dump

geometry_dump — En sammansatt typ som används för att beskriva delarna i en komplex geometri.

Beskrivning

`geometry_dump` är en sammansatt datatyp som innehåller fälten:

- `geom` - en geometri som representerar en komponent i den dumpade geometrin. Geometritypen beror på den ursprungliga funktionen.
- `path[]` - en heltalsarray som definierar navigeringsvägen inom den dumpade geometrin till `geom-komponen`
  Path-arrayen är 1-baserad (d.v.s. path `[1]` är det första elementet)

Den används av `ST_Dump*-familjen` av funktioner som en utdatatyp för att bryta ner en komplex geometri i sina beståndsdelar.

Se även

Avsnitt 13.6

7.1.5 geography

geography — Den typ som representerar spatiala egenskaper med geodetiska (ellipsoidiska) koordinatsystem.

Beskrivning

`geography` är en spatial datatyp som används för att representera en funktion i geodetiska koordinatsystem. Geodetiska koordinatsystem modellerar jorden med hjälp av en ellipsoid.

Spatiala operationer på den geografiska typen ger mer exakta resultat genom att ta hänsyn till den ellipsoidiska modellen.

Casting-beteende

I denna tabell listas de automatiska och explicita casts som är tillåtna för denna datatyp:

Kasta till	Beteende
geometri	explicit

Se även

Avsnitt 4.3, Avsnitt 13.4

7.2 Funktioner för tabellhantering

7.2.1 AddGeometryColumn

AddGeometryColumn — Lägger till en geometrikolumn i en befintlig tabell.

Synopsis

text **AddGeometryColumn**(varchar table_name, varchar column_name, integer srid, varchar type, integer dimension, boolean use_typmod=true);
text **AddGeometryColumn**(varchar schema_name, varchar table_name, varchar column_name, integer srid, varchar type, integer dimension, boolean use_typmod=true);
text **AddGeometryColumn**(varchar catalog_name, varchar schema_name, varchar table_name, varchar column_name, integer srid, varchar type, integer dimension, boolean use_typmod=true);

Beskrivning

Lägger till en geometrikolumn i en befintlig tabell med attribut. `Schema_name` är namnet på tabellschemat. `Srid` måste vara en heltalsreferens till en post i tabellen SPATIAL_REF_SYS. `Type` måste vara en sträng som motsvarar geometritypen, t.ex. POLYGON”eller MULTILINESTRING”. Ett fel uppstår om schemanamnet inte finns (eller inte är synligt i den aktuella sökvägen) eller om den angivna SRID:en, geometritypen eller dimensionen är ogiltig.

Ändrad: 2.0.0 Den här funktionen uppdaterar inte längre geometry_columns eftersom geometry_columns är en vy som läser från systemkataloger. Som standard skapar det inte heller begränsningar, utan använder istället det inbyggda typmodifieringsbeteendet för PostgreSQL. Så till exempel att bygga en wgs84 POINT-kolumn med den här funktionen motsvarar nu: `ALTER TABLE some_table ADD COLUMN geom geometry (Point,4326);`

Ändrad: 2.0.0 Om du vill ha det gamla beteendet för begränsningar, använd standardvärdet `use_typmod`, men sätt det till false.

Ändrad: 2.0.0 Vyer kan inte längre registreras manuellt i geometry_columns, men vyer som är byggda mot geometri typmod-tabellgeometrier och används utan omslagsfunktioner kommer att registrera sig korrekt eftersom de ärver typmod-beteendet för sin överordnade tabellkolumn. Vyer som använder geometrifunktioner som matar ut andra geometrier måste castas till typmod-geometrier för att dessa vygeometrikolumner ska registreras korrekt i geometry_columns. Se Avsnitt 4.6.3.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Förbättrad: 2.0.0 use_typmod-argumentet infördes. Standard är att skapa typmod-geometrikolumn istället för begränsningsbaserad.

Exempel

This example creates a schema to hold data.

Code

```
CREATE SCHEMA my_schema;
```

Create a simple PostgreSQL table in the schema.

Code

```
CREATE TABLE my_schema.my_spatial_table (id serial);
```

Describing the table shows a single `id` column.

Code

```
\d my_schema.my_spatial_table
```

Output

```
Table "my_schema.my_spatial_table"
 Column |  Type   |                                 Modifiers
--------+---------+-------------------------------------------------------------------------- ↩

 id     | integer | not null default nextval('my_schema.my_spatial_table_id_seq'::regclass)
```

Add a spatial column to the table.

Code

```
SELECT AddGeometryColumn('my_schema', 'my_spatial_table', 'geom', 4326, 'POINT', 2);
```

Add a point column using the old constraint-based behavior.

Code

```
SELECT AddGeometryColumn('my_schema', 'my_spatial_table', 'geom_c', 4326, 'POINT', 2, false ↩
    );
```

Add a CurvePolygon column using the old constraint-based behavior.

Code

```
SELECT AddGeometryColumn('my_schema', 'my_spatial_table', 'geomcp_c', 4326, 'CURVEPOLYGON', ↩
     2, false);
```

Output

```
my_schema.my_spatial_table.geomcp_c SRID:4326 TYPE:CURVEPOLYGON DIMS:2
```

Describe the table again to see the new geometry columns.

Code

```
\d my_schema.my_spatial_table
```

Output

```
Table "my_schema.my_spatial_table"

Columns:
  id       integer               not null default nextval('my_schema.my_spatial_table_id_seq'::regclass)
  geom     geometry(Point,4326)
  geom_c   geometry
  geomcp_c geometry

Check constraints:
    "enforce_dims_geom_c" CHECK (st_ndims(geom_c) = 2)
    "enforce_dims_geomcp_c" CHECK (st_ndims(geomcp_c) = 2)
    "enforce_geotype_geom_c" CHECK (geometrytype(geom_c) = 'POINT'::text OR geom_c IS NULL)
    "enforce_geotype_geomcp_c" CHECK (geometrytype(geomcp_c) = 'CURVEPOLYGON'::text OR geomcp_c IS NULL)
    "enforce_srid_geom_c" CHECK (st_srid(geom_c) = 4326)
    "enforce_srid_geomcp_c" CHECK (st_srid(geomcp_c) = 4326)
```

The `geometry_columns` view also registers the new columns.

Code

```
SELECT f_geometry_column As col_name, type, srid, coord_dimension As ndims
FROM geometry_columns
WHERE f_table_name = 'my_spatial_table' AND f_table_schema = 'my_schema';
```

Output

```
col_name |     type     | srid | ndims
----------+--------------+------+-------
 geom     | Point        | 4326 |     2
 geom_c   | Point        | 4326 |     2
 geomcp_c | CurvePolygon | 4326 |     2
```

Se även

DropGeometryColumn, DropGeometryTable, Avsnitt 4.6.2, Avsnitt 4.6.3

7.2.2 DropGeometryColumn

DropGeometryColumn — Tar bort en geometrikolumn från en spatial tabell.

Synopsis

text **DropGeometryColumn**(varchar table_name, varchar column_name);
text **DropGeometryColumn**(varchar schema_name, varchar table_name, varchar column_name);
text **DropGeometryColumn**(varchar catalog_name, varchar schema_name, varchar table_name, varchar column_name);

Beskrivning

Tar bort en geometrikolumn från en spatial tabell. Observera att schema_name måste matcha fältet f_table_schema i tabellens rad i tabellen geometry_columns.

Varning
`DropGeometryColumn` is retained for backward compatibility and is not recommended for new applications. Use the standard **ALTER TABLE ... DROP COLUMN** command instead. The `geometry_columns` view updates automatically from the system catalogs.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Changed: 2.0.0. Retained for backward compatibility after `geometry_columns` became a view of the system catalogs.

Exempel

Use standard SQL to drop the geometry column. The `geometry_columns` view updates automatically.

Code

```
ALTER TABLE my_schema.my_spatial_table DROP COLUMN geom;
```

The legacy compatibility function performs the equivalent operation.

Code

```
SELECT DropGeometryColumn ('my_schema', 'my_spatial_table', 'geom');
```

Output

```
my_schema.my_spatial_table.geom effectively removed.
```

Se även

AddGeometryColumn, DropGeometryTable, Avsnitt 4.6.2

7.2.3 DropGeometryTable

DropGeometryTable — Tar bort en tabell och alla dess referenser i geometry_columns.

Synopsis

boolean **DropGeometryTable**(varchar table_name);
boolean **DropGeometryTable**(varchar schema_name, varchar table_name);
boolean **DropGeometryTable**(varchar catalog_name, varchar schema_name, varchar table_name);

Beskrivning

Slopar en tabell och alla dess referenser i geometry_columns. Observera: använder current_schema() på schemamedvetna pgsql-installationer om schema inte har angetts.

Varning
`DropGeometryTable` is retained for backward compatibility and is not recommended for new applications. Use the standard **DROP TABLE** command instead. The `geometry_columns` view updates automatically from the system catalogs.

Changed: 2.0.0. Retained for backward compatibility after `geometry_columns` became a view of the system catalogs.

Exempel

Use standard SQL to drop a table containing geometry columns. The `geometry_columns` view updates automatically.

Code

```
DROP TABLE my_schema.my_spatial_table;
```

The legacy compatibility function performs the equivalent operation.

Code

```
SELECT DropGeometryTable ('my_schema', 'my_spatial_table');
```

Output

```
my_schema.my_spatial_table dropped.
```

Se även

AddGeometryColumn, DropGeometryColumn, Avsnitt 4.6.2

7.2.4 Find_SRID

Find_SRID — Returnerar den SRID som definierats för en geometrikolumn.

Synopsis

integer **Find_SRID**(varchar a_schema_name, varchar a_table_name, varchar a_geomfield_name);

Beskrivning

Returnerar heltals-SRID för den angivna geometrikolumnen genom sökning i tabellen GEOMETRY_COLUMN
Om geometrikolumnen inte har lagts till på rätt sätt (t.ex. med funktionen AddGeometryColumn) fungerar inte denna funktion.

Exempel

Code

```
SELECT Find_SRID('public', 'tiger_us_state_2007', 'geom_4269');
```

Output

```
4269
```

Se även

ST_SRID

7.2.5 Populate_Geometry_Columns

Populate_Geometry_Columns — Säkerställer att geometrikolumner definieras med typmodifierare eller har lämpliga spatiala begränsningar.

Synopsis

text **Populate_Geometry_Columns**(boolean use_typmod=true);
int **Populate_Geometry_Columns**(oid relation_oid, boolean use_typmod=true);

Beskrivning

Säkerställer att geometrikolumner har lämpliga typmodifierare eller spatiala begränsningar för att säkerställa att de registreras korrekt i vyn `geometry_columns`. Som standard konverteras alla geometrikolumner utan typmodifierare till sådana med typmodifierare.

För bakåtkompatibilitet och för spatiala behov, t.ex. tabellarv där varje underordnad tabell kan ha olika geometrityp, stöds fortfarande det gamla check constraint-beteendet. Om du behöver det gamla beteendet måste du skicka in det nya valfria argumentet som false `use_typmod=false`. När detta görs kommer geometrikolumner att skapas utan typmodifierare men med 3 definierade begränsningar. Detta innebär i synnerhet att varje geometrikolumn som tillhör en tabell har minst tre begränsningar:

- `enforce_dims_geom` - säkerställer att varje geometri har samma dimension (se ST_NDims)
- `enforce_geotype_geom` - säkerställer att alla geometrier är av samma typ (se GeometryType)
- `enforce_srid_geom` - säkerställer att varje geometri är i samma projektion (se ST_SRID)

Om en `oid` för tabellen anges försöker denna funktion fastställa srid, dimension och geometrityp för alla geometrikolumner i tabellen och lägger till begränsningar vid behov. Om det lyckas infogas en lämplig rad i tabellen geometry_columns, annars fångas undantaget upp och ett felmeddelande skickas ut som beskriver problemet.

Om `oid` för en vy anges, som med en tabell oid, försöker denna funktion bestämma srid, dimension och typ för alla geometrier i vyn och infogar lämpliga poster i tabellen `geometry_columns`, men ingenting görs för att upprätthålla begränsningar.

Den parameterlösa varianten är en enkel omslutning för den parameteriserade varianten som först trunkerar och fyller på geometry_columns-tabellen för varje spatial tabell och vy i databasen, och lägger till spatiala begränsningar i tabellerna där så är lämpligt. Den returnerar en sammanfattning av antalet geometrikolumner som upptäckts i databasen och det antal som infogats i tabellen `geometry_columns`. Den parametriserade versionen returnerar helt enkelt antalet rader som infogats i tabellen `geometry_columns`.

Tillgänglighet: 1.4.0

Ändrad: 2.0.0 Som standard används nu typmodifierare i stället för kontrollbegränsningar för att begränsa geometrityper. Du kan fortfarande använda check constraint-beteende istället genom att använda den nya `use_typmod` och ställa in den på false.

Förbättrad: 2.0.0 `use_typmod` valfritt argument introducerades som gör det möjligt att kontrollera om kolumner skapas med typmodifierare eller med kontrollbegränsningar.

Exempel

This example populates geometry column metadata using typmod behavior. For this to work, there must be existing data.

Code

```
CREATE TABLE public.myspatial_table(gid serial, geom geometry);
INSERT INTO myspatial_table(geom) VALUES(ST_GeomFromText('LINESTRING(1 2,3 4)', 4326) );
SELECT Populate_Geometry_Columns('public.myspatial_table'::regclass);
```

Output

```
1
```

Describe the table to see the typmod-based geometry column.

Code

```
\d myspatial_table
```

Output

```
Table "public.myspatial_table"

Columns:
  gid  integer                   not null default nextval('myspatial_table_gid_seq':: ↩
    regclass)
  geom geometry(LineString,4326)
```

This example changes the geometry columns to use constraints if they are not typmod or already constrained. For this to work, there must be existing data.

Code

```
CREATE TABLE public.myspatial_table_cs(gid serial, geom geometry);
INSERT INTO myspatial_table_cs(geom) VALUES(ST_GeomFromText('LINESTRING(1 2,3 4)', 4326) );
SELECT Populate_Geometry_Columns('public.myspatial_table_cs'::regclass, false);
```

Output

```
1
```

Describe the table to see the constraint-based geometry column.

Code

```
\d myspatial_table_cs
```

Output

```
Table "public.myspatial_table_cs"
 Column |   Type   |                           Modifiers
--------+----------+---------------------------------------------------------------
 gid    | integer  | not null default nextval('myspatial_table_cs_gid_seq'::regclass)
 geom   | geometry |
Check constraints:
    "enforce_dims_geom" CHECK (st_ndims(geom) = 2)
    "enforce_geotype_geom" CHECK (geometrytype(geom) = 'LINESTRING'::text OR geom IS NULL)
    "enforce_srid_geom" CHECK (st_srid(geom) = 4326)
```

7.2.6 UpdateGeometrySRID

UpdateGeometrySRID — Uppdaterar SRID för alla objekt i en geometrikolumn och metadata för tabellen.

Synopsis

text **UpdateGeometrySRID**(varchar table_name, varchar column_name, integer srid);
text **UpdateGeometrySRID**(varchar schema_name, varchar table_name, varchar column_name, integer srid);
text **UpdateGeometrySRID**(varchar catalog_name, varchar schema_name, varchar table_name, varchar column_name, integer srid);

Beskrivning

Uppdaterar SRID för alla objekt i en geometrikolumn och uppdaterar begränsningar och referens i geometry_columns. Om kolumnen styrdes av en typdefinition kommer typdefinitionen att ändras. Observera: använder current_schema() på schemamedvetna pgsql-installationer om schema inte har angetts.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Infoga geometrier i vägtabeller med en SRID-uppsättning som redan använder EWKT-format:

Code

```
COPY roads (geom) FROM STDIN;
SRID=4326;LINESTRING(0 0, 10 10)
SRID=4326;LINESTRING(10 10, 15 0)
\.
```

Detta kommer att ändra srid för vägtabellen till 4326 från vad det var tidigare:

Code

```
SELECT UpdateGeometrySRID('roads', 'geom', 4326);
```

Det tidigare exemplet motsvarar denna DDL-sats:

Code

```
ALTER TABLE roads
ALTER COLUMN geom TYPE geometry(MULTILINESTRING, 4326)
  USING ST_SetSRID(geom, 4326);
```

Om du fick fel projektion (eller tog in den som okänd) i lasten och du ville omvandla till webbmercator allt på en gång kan du göra det med DDL men det finns ingen motsvarande PostGIS-hanteringsfunktion för att göra det på en gång.

Code

```
ALTER TABLE roads
ALTER COLUMN geom TYPE geometry(MULTILINESTRING, 3857) USING ST_Transform(ST_SetSRID(geom, ↩
    4326), 3857) ;
```

Se även

UpdateRasterSRID, ST_SetSRID, ST_Transform, ST_GeomFromEWKT

7.3 Geometry Constructors

7.3.1 ST_Collect

ST_Collect — Skapar en GeometryCollection eller Multi* geometri från en uppsättning geometrier.

Synopsis

geometry **ST_Collect**(geometry g1, geometry g2);
geometry **ST_Collect**(geometry[] g1_array);
geometry **ST_Collect**(geometry set g1field);

Beskrivning

Samlar geometrier till en geometrisamling. Resultatet är antingen en Multi* eller en GeometryCollection, beroende på om indatageometrierna har samma eller olika typer (homogena eller heterogena). De ingående geometrierna lämnas oförändrade inom samlingen.

Variant 1: accepterar två inmatningsgeometrier

Variant 2: accepterar en array av geometrier

Variant 3: Aggregerad funktion som tar emot en rad med geometrier.

Varning
`ST_Collect` does not dissolve boundaries or resolve overlaps. Collecting overlapping polygons can produce an invalid MultiPolygon. If the result must be a valid polygonal geometry with overlaps merged, use ST_Union instead. `ST_Union` also splits linestrings at intersections and may return a single geometry after dissolving boundaries.

If any input geometry is a collection (Multi* or GeometryCollection), `ST_Collect` returns a GeometryCollection, since that is the only type which can contain nested collections. To prevent this, use ST_Dump in a subquery to expand input collections to their atomic elements (see example below).

Tillgänglighet: 1.4.0 - ST_Collect(geomarray) introducerades. ST_Collect förbättrades för att hantera fler geometrier snabbare.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Two-input variant.

Samla 2D-punkter.

Code

```
SELECT ST_Collect(
    'POINT(1 2)',
    'POINT(-2 3)' );
```

Output

```
MULTIPOINT((1 2),(-2 3))
```

Figure

Samla 3D-punkter.

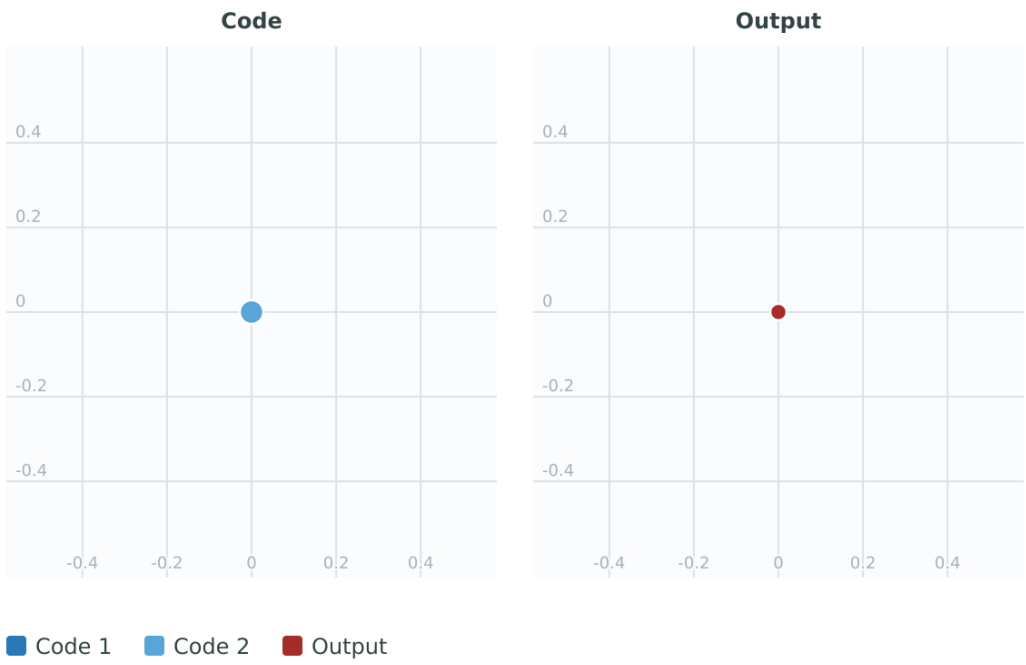
Code

```
SELECT ST_Collect(
        'POINT(1 2 3)',
        'POINT(1 2 4)' );
```

Output

```
MULTIPOINT(1 2 3,1 2 4)
```

Figure

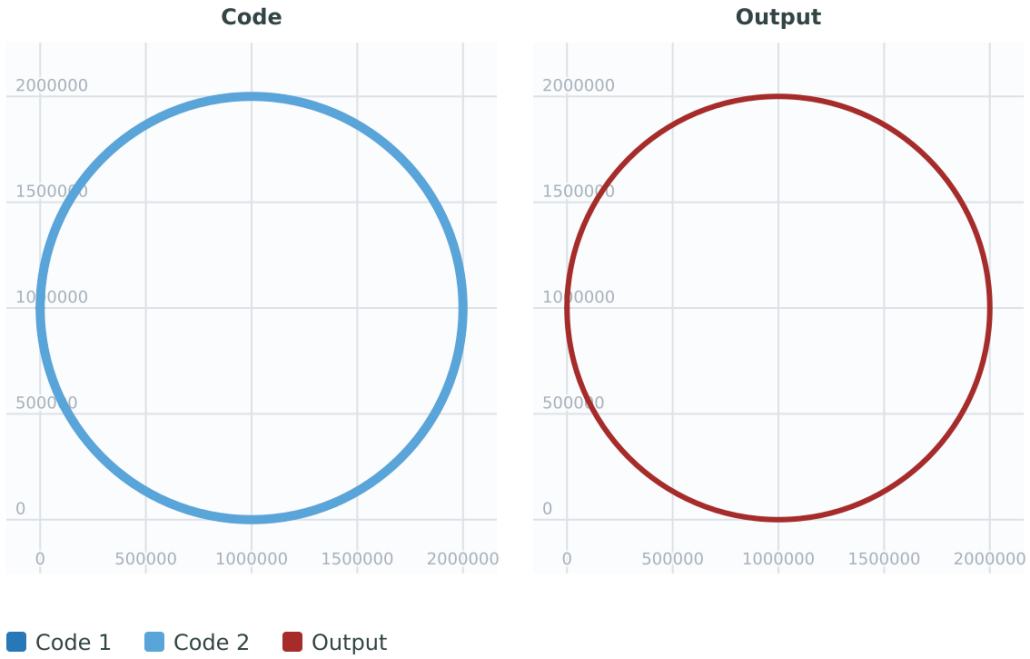


Samla kurvor.

Code

```
SELECT ST_Collect(
        'CIRCULARSTRING(220268 150415,220227 150505,220227 150406)',
        'CIRCULARSTRING(220227 150406,2220227 150407,220227 150406)');
```

Figure



Array variant.

Använda en array constructor för en subquery.

Code

```
SELECT ST_Collect(ARRAY(SELECT geom FROM sometable));
```

Använda en array constructor för värden.

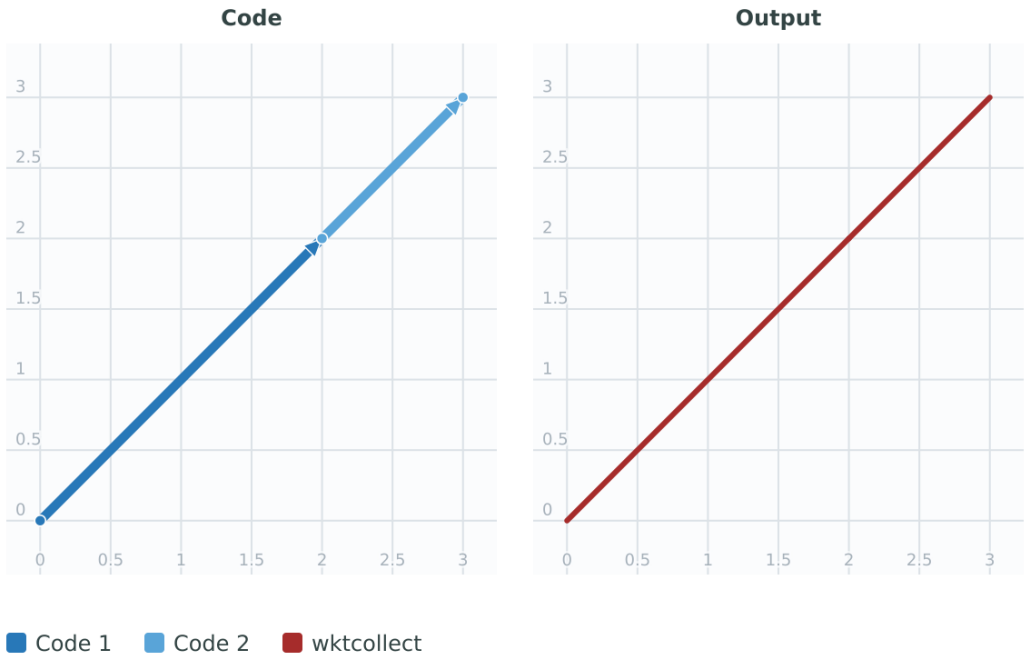
Code

```
SELECT ST_Collect(
        ARRAY[ 'LINESTRING(1 2,3 4)'::geometry,
            'LINESTRING(3 4,4 5)'::geometry ] ) As wktcollect;
```

Output

```
MULTILINESTRING((1 2,3 4),(3 4,4 5))
```

Figure



Aggregate variant.

Skapa flera samlingar genom att gruppera geometrier i en tabell.

Code

```
SELECT stusps, ST_Collect(geom) AS geom
FROM (
  SELECT stusps, (ST_Dump(geom)).geom AS geom
  FROM somestatetable
) AS f
GROUP BY stusps;
```

Se även

ST_Dump, ST_Union

7.3.2 ST_LineFromMultiPoint

ST_LineFromMultiPoint — Skapar en LineString från en MultiPoint-geometri.

Synopsis

geometry **ST_LineFromMultiPoint**(geometry aMultiPoint);

Beskrivning

Skapar en LineString från en MultiPoint-geometri.
Använd ST_MakeLine för att skapa linjer från Point- eller LineString-inmatningar.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Skapa en 3D-linestrings från en 3D MultiPoint

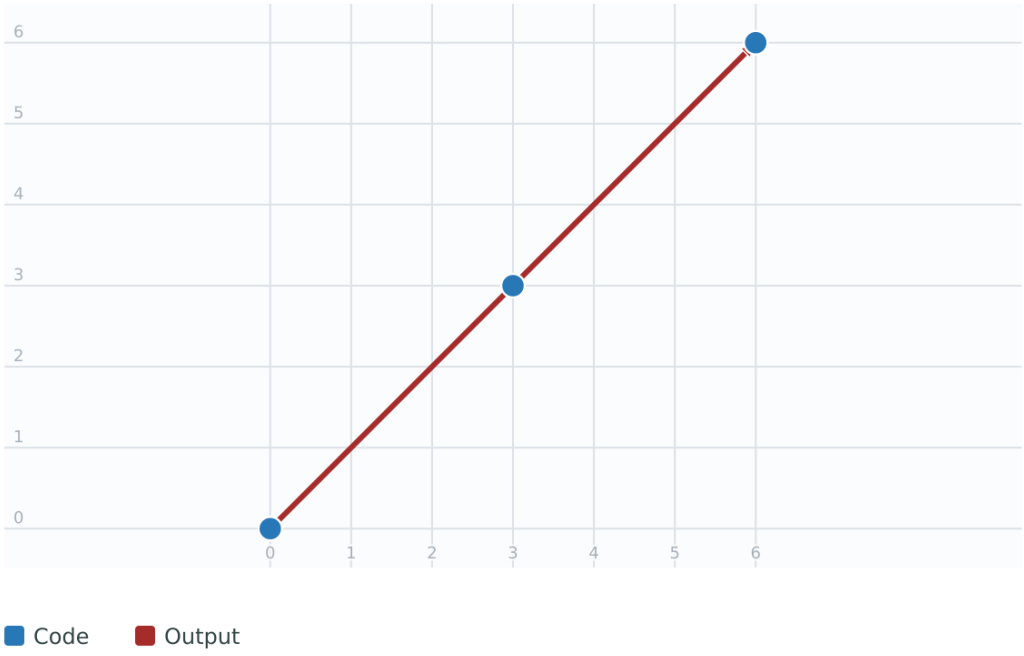
Code

```
SELECT ST_LineFromMultiPoint('MULTIPOINT(1 2 3,4 5 6,7 8 9)');
```

Output

```
LINESTRING(1 2 3,4 5 6,7 8 9)
```

Figure



Se även

ST_AsEWKT, ST_MakeLine

7.3.3 ST_MakeEnvelope

ST_MakeEnvelope — Skapar en rektangulär polygon från minimi- och maximikoordinater.

Synopsis

geometry **ST_MakeEnvelope**(float xmin, float ymin, float xmax, float ymax, integer srid=unknown);

Beskrivning

Skapar en rektangulär polygon från minimi- och maximivärdena för X och Y. Inmatade värden måste vara i det spatiala referenssystem som anges av SRID. Om inget SRID anges används det okända spatiala referenssystemet (SRID 0).

Tillgänglighet: 1,5

Förbättrad: 2.0: Möjlighet att ange ett kuvert utan att ange en SRID infördes.

Exempel

Building a bounding box polygon.

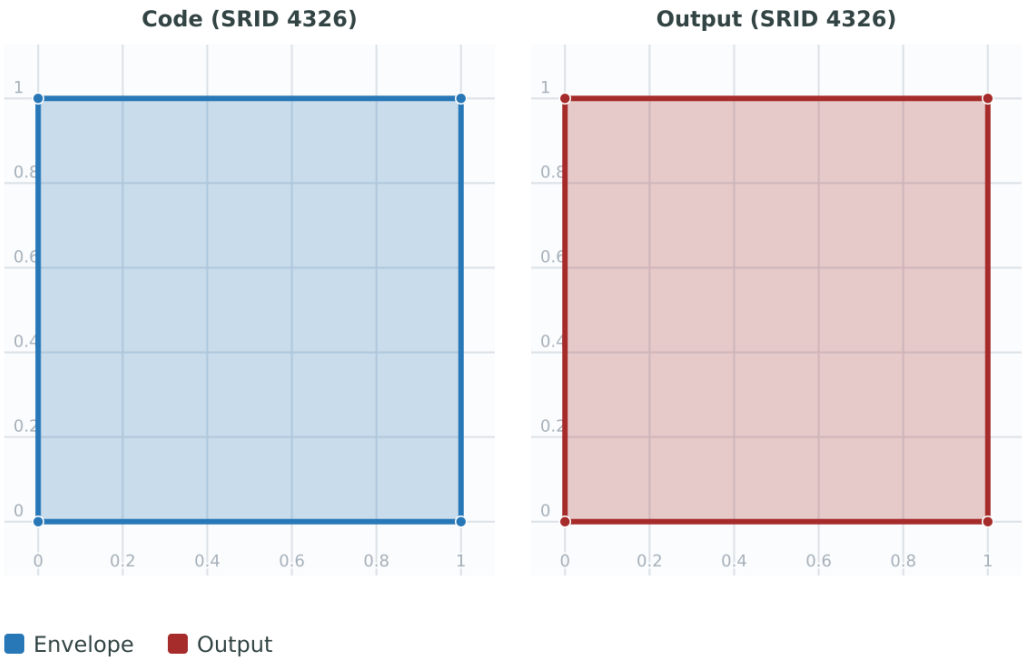
Code

```
SELECT ST_MakeEnvelope(10, 10, 11, 11, 4326);
```

Output

```
POLYGON((10 10, 10 11, 11 11, 11 10, 10 10))
```

Figure



Se även

ST_MakePoint, ST_MakeLine, ST_MakePolygon, ST_TileEnvelope

7.3.4 ST_MakeLine

ST_MakeLine — Skapar en LineString från Point-, MultiPoint- eller LineString-geometrier.

Synopsis

geometry **ST_MakeLine**(geometry geom1, geometry geom2);
geometry **ST_MakeLine**(geometry[] geoms_array);
geometry **ST_MakeLine**(geometry set geoms);

Beskrivning

Skapar en LineString som innehåller punkterna i geometrierna Point, MultiPoint eller LineString. Andra geometrityper orsakar ett fel.

Variant 1: accepterar två inmatningsgeometrier

Variant 2: accepterar en array av geometrier

Variant 3: Aggregerad funktion som accepterar en rad med geometrier. För att säkerställa ordningen på geometrierna i indata, använd `ORDER BY` i funktionsanropet eller en underfråga med en `ORDER BY-klausul`.

Repeated nodes at the beginning of input LineStrings are collapsed to a single point. Repeated points in Point and MultiPoint inputs are not collapsed. Components of MultiLineString are handled in the order they appear in the collection. ST_RemoveRepeatedPoints can be used to collapse repeated points from the output LineString.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Availability: 3.7.0 - Support for MultiLineString input elements was introduced

Tillgänglighet: 2.3.0 - Stöd för MultiPoint-indataelement infördes

Tillgänglighet: 2.0.0 - Stöd för LineString-indataelement infördes

Tillgänglighet: 1.4.0 - ST_MakeLine(geomarray) introducerades. ST_MakeLine aggregatfunktioner förbättrades för att hantera fler punkter snabbare.

Exempel

Two-input variant.

Skapa en linje som består av två punkter.

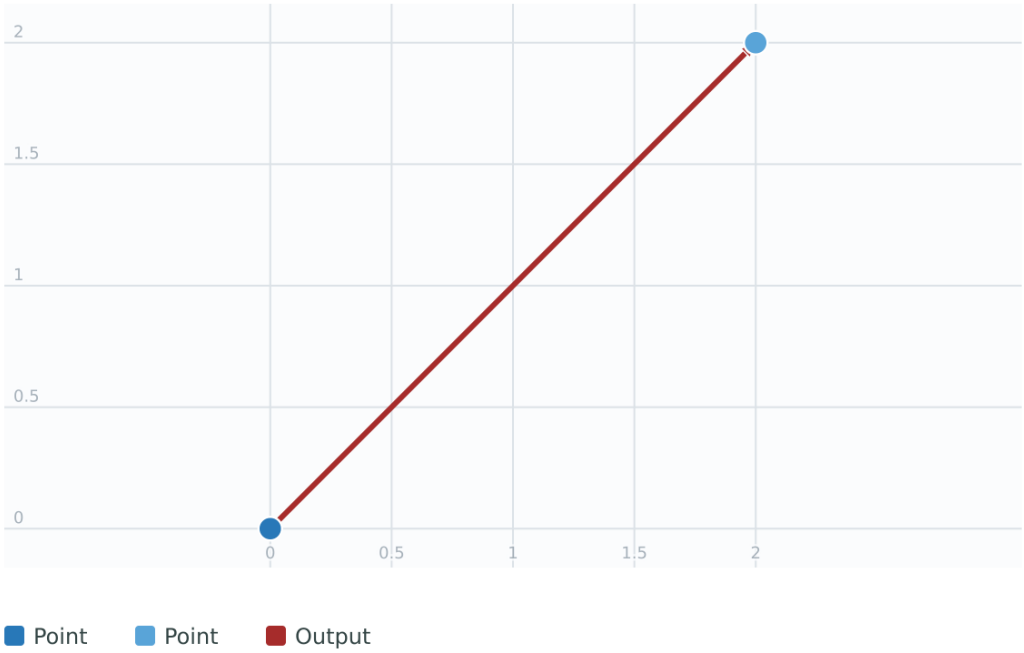
Code

```
SELECT ST_MakeLine(ST_Point(1, 2), ST_Point(3, 4));
```

Output

```
LINESTRING(1 2,3 4)
```

Figure



Skapa en 3D-linje från två 3D-punkter.

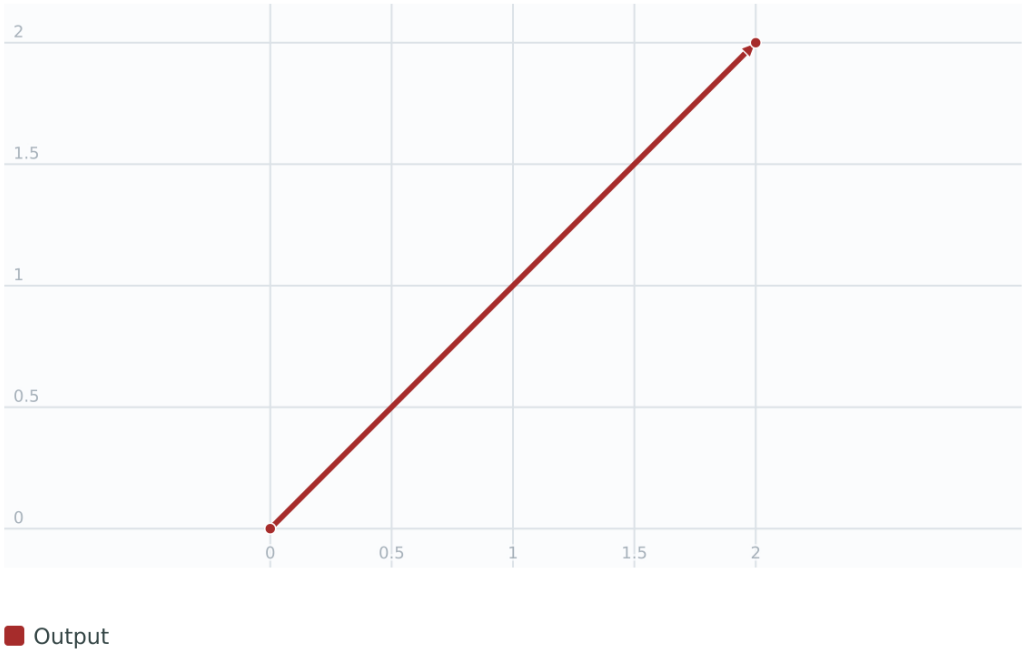
Code

```
SELECT ST_MakeLine(ST_MakePoint(1, 2, 3), ST_MakePoint(3, 4, 5));
```

Output

```
LINESTRING(1 2 3,3 4 5)
```

Figure



Skapa en linje från två åtskilda LineStrings.

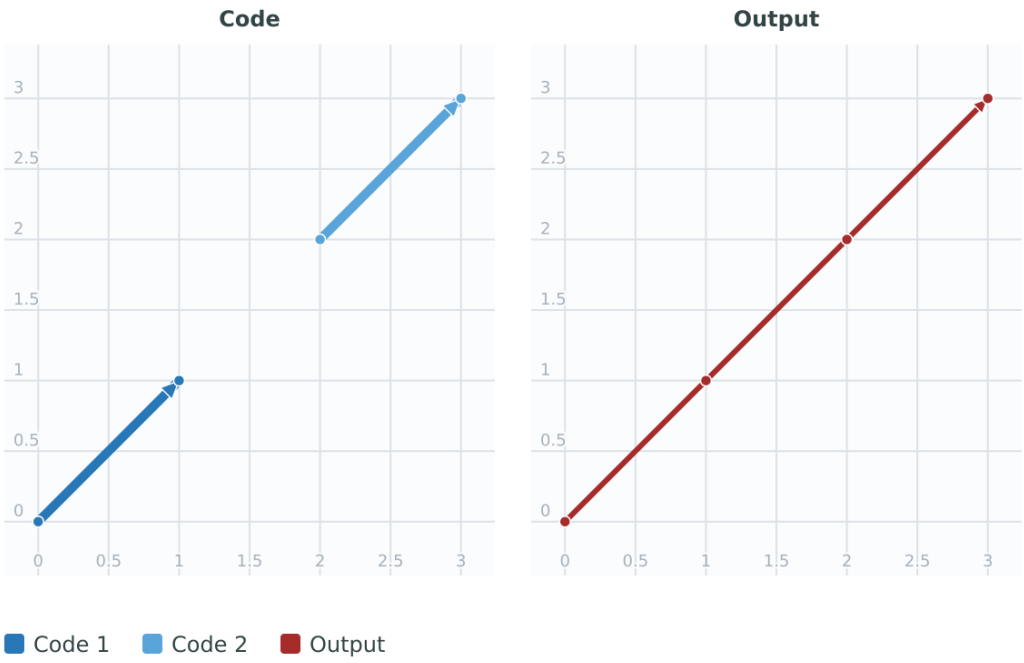
Code

```
SELECT ST_MakeLine('LINESTRING(0 0,1 1)', 'LINESTRING(2 2,3 3)');
```

Output

```
LINESTRING(0 0,1 1,2 2,3 3)
```

Figure



Array variant.

Skapa en rad från en matris som bildas av en underfråga med ordning.

Code

```
SELECT ST_MakeLine(ARRAY(SELECT ST_Centroid(geom) FROM visit_locations ORDER BY visit_time) ↩
    );
```

Skapa en 3D-linje från en array av 3D-punkter

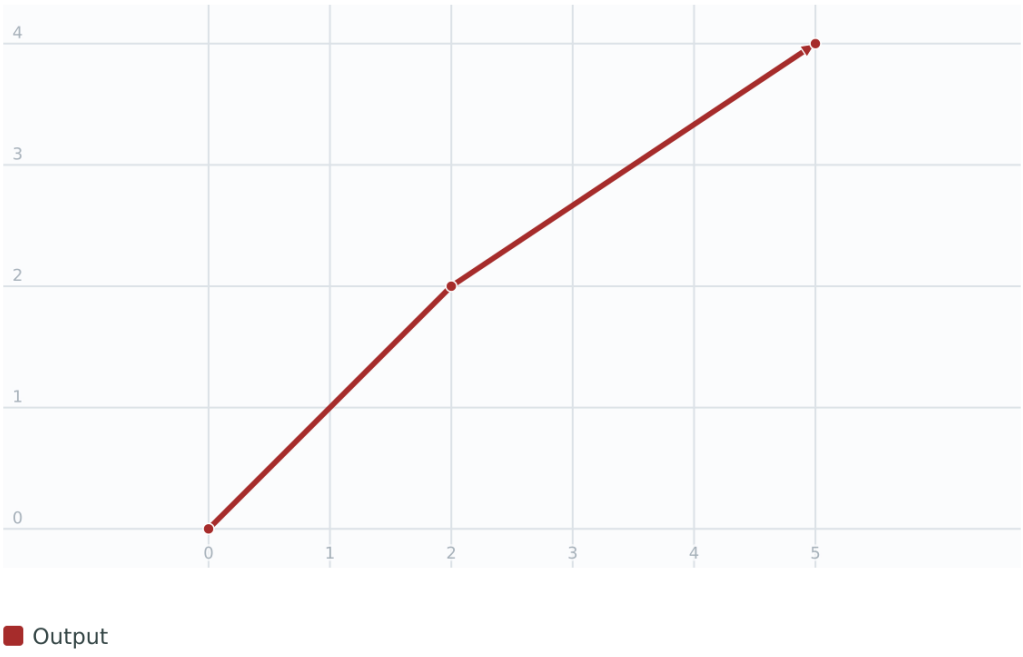
Code

```
SELECT ST_MakeLine(ARRAY[ST_MakePoint(1, 2, 3), ST_MakePoint(3, 4, 5), ST_MakePoint(6, 6, ↩
    6)]);
```

Output

```
LINESTRING(1 2 3,3 4 5,6 6 6)
```

Figure



Aggregate variant.

I detta exempel ställs frågor om tidsbaserade sekvenser av GPS-punkter från en uppsättning spår och en post skapas för varje spår. Resultatgeometrierna är LineStrings som består av GPS-spårpunkterna i den ordning de färdats.

Om du använder aggregatet `ORDER BY` får du en korrekt ordnad LineString.

Code

```
SELECT gps.track_id, ST_MakeLine(gps.geom ORDER BY gps_time) AS geom
FROM gps_points AS gps
GROUP BY gps.track_id;
```

When ordering within the aggregate is not possible, ordering in a subquery can be used. However, sometimes the query plan may not respect the order of the subquery.

Code

```
SELECT gps.track_id, ST_MakeLine(gps.geom) AS geom
FROM (
  SELECT track_id, gps_time, geom
  FROM gps_points
  ORDER BY track_id, gps_time
) AS gps
GROUP BY gps.track_id;
```

Se även

ST_RemoveRepeatedPoints, ST_AsEWKT, ST_AsText, ST_GeomFromText, ST_MakePoint, ST_Point

7.3.5 ST_MakePoint

ST_MakePoint — Skapar en 2D-, 3DZ- eller 4D-punkt.

Synopsis

geometry **ST_MakePoint**(float x, float y);

geometry **ST_MakePoint**(float x, float y, float z);

geometry **ST_MakePoint**(float x, float y, float z, float m);

Beskrivning

Skapar en 2D XY-, 3D XYZ- eller 4D XYZM-punktgeometri. Använd ST_MakePointM för att skapa punkter med XYM-koordinater.

Använd ST_SetSRID för att ange en SRID för den skapade punkten.

`ST_MakePoint` är inte OGC-kompatibelt, men är snabbare än ST_GeomFromText och ST_PointFromText. Det är också lättare att använda för numeriska koordinatvärden.

Notera
För geodetiska koordinater är X longitud och Y latitud

Notera
Funktionerna ST_Point, ST_PointZ, ST_PointM, och ST_PointZM kan användas för att skapa punkter med en given SRID.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

This example creates a point with an unknown SRID.

Code

```
SELECT ST_MakePoint(-71.1043443253471, 42.3150676015829);
```

This example creates a point in the WGS 84 geodetic CRS.

Code

```
SELECT ST_SetSRID(ST_MakePoint(-71.1043443253471, 42.3150676015829), 4326);
```

This example creates a 3D point with altitude.

Code

```
SELECT ST_MakePoint(1, 2, 1.5);
```

This example returns the Z coordinate of a point.

Code

```
SELECT ST_Z(ST_MakePoint(1, 2, 1.5));
```

Output

```
1.5
```

Se även

ST_GeomFromText, ST_PointFromText, ST_SetSRID, ST_MakePointM, ST_Point, ST_PointZ, ST_PointM, ST_PointZM

7.3.6 ST_MakePointM

ST_MakePointM — Skapar en punkt från X-, Y- och M-värden.

Synopsis

geometry **ST_MakePointM**(float x, float y, float m);

Beskrivning

Skapar en punkt med X-, Y- och M-ordinater (mått). Använd ST_MakePoint för att skapa punkter med XY-, XYZ- eller XYZM-koordinater.

Använd ST_SetSRID för att ange en SRID för den skapade punkten.

Notera
För geodetiska koordinater är X longitud och Y latitud

Notera
Funktionerna ST_PointM, och ST_PointZM kan användas för att skapa punkter med ett M-värde och en given SRID.

Exempel

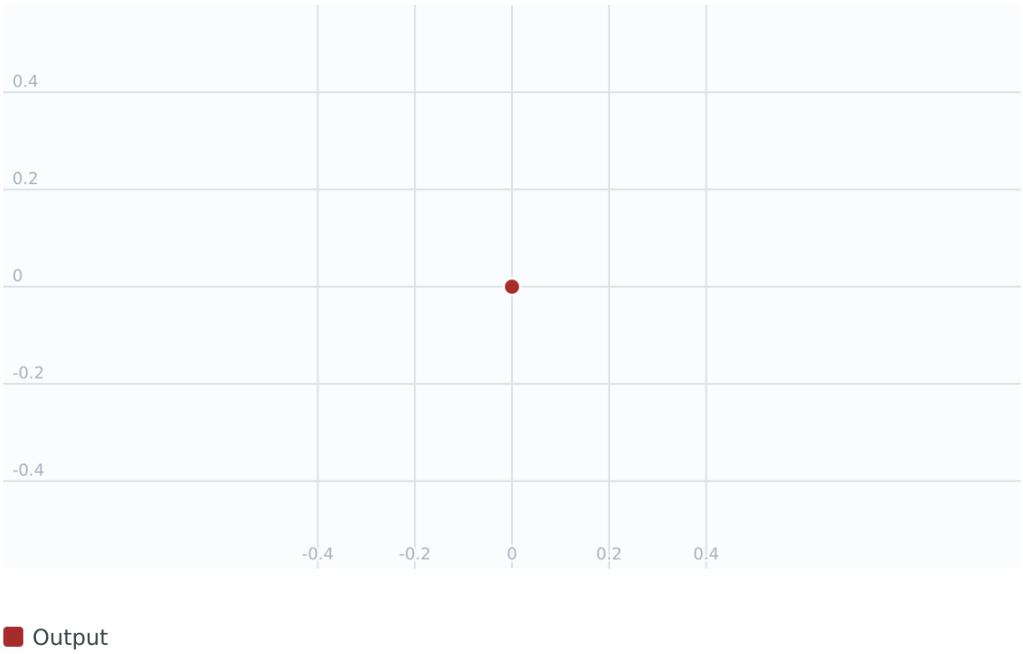
Notera
ST_AsEWKT används för textutdata eftersom ST_AsText inte stöder M-värden.

Skapa punkt med okänd SRID.

Code

```
SELECT ST_MakePointM(-71.1043443253471, 42.3150676015829, 10);
```

Figure

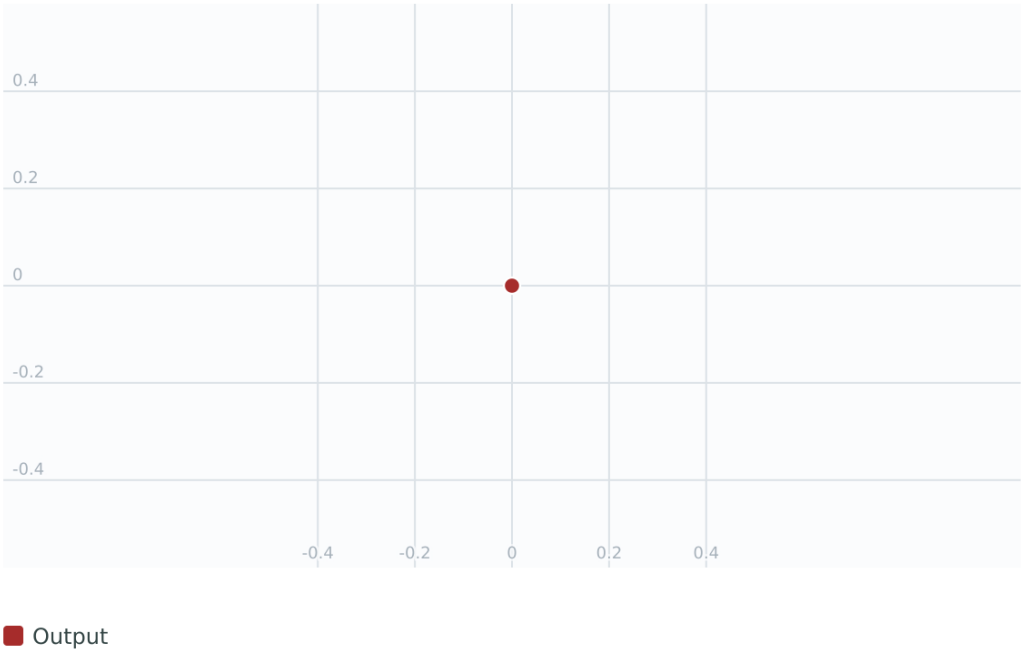


Skapa en punkt med ett mått i det geodetiska koordinatsystemet WGS 84.

Code

```
SELECT ST_SetSRID(ST_MakePointM(-71.104, 42.315, 10), 4326);
```

Figure



Få mått på skapad punkt.

Code

```
SELECT ST_M(ST_MakePointM(-71.104, 42.315, 10));
```

Output

```
10
```

Se även

ST_MakePoint, ST_SetSRID, ST_PointM, ST_PointZM

7.3.7 ST_MakePolygon

ST_MakePolygon — Creates a Polygon or CurvePolygon from a shell and optional list of holes.

Synopsis

geometry **ST_MakePolygon**(geometry linestring);

geometry **ST_MakePolygon**(geometry outerlinestring, geometry[] interiorlinestrings);

Beskrivning

Creates a Polygon or CurvePolygon formed by the given shell and optional array of holes. Input geometries must be closed LineStrings, CircularStrings, CompoundCurves, or NURBSCurves (rings). If any input ring is curved, the result is a CurvePolygon.

Variant 1: Accepts one shell ring.

Variant 2: Accepts a shell ring and an array of inner (hole) rings. A geometry array can be constructed using the PostgreSQL array_agg(), ARRAY[] or ARRAY() constructs.

Notera
Denna funktion accepterar inte MultiLineStrings. Använd ST_LineMerge för att generera en LineString, eller ST_Dump för att extrahera LineStrings.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Enhanced: 3.7.0 - Support for curved input rings was introduced.

Exempel

Single input variant.

Skapa en polygon från en 2D LineString.

Code

```
SELECT ST_MakePolygon('LINESTRING(75 29,77 29,77 29,75 29)');
```

Skapa en polygon från en öppen LineString och använd ST_StartPoint och ST_AddPoint för att stänga den.

Code

```
SELECT ST_MakePolygon(ST_AddPoint(foo.open_line, ST_StartPoint(foo.open_line)) )
FROM (
  SELECT 'LINESTRING(75 29,77 29,77 29,75 29)'::geometry As open_line) As foo;
```

Skapa en polygon från en 3D LineString

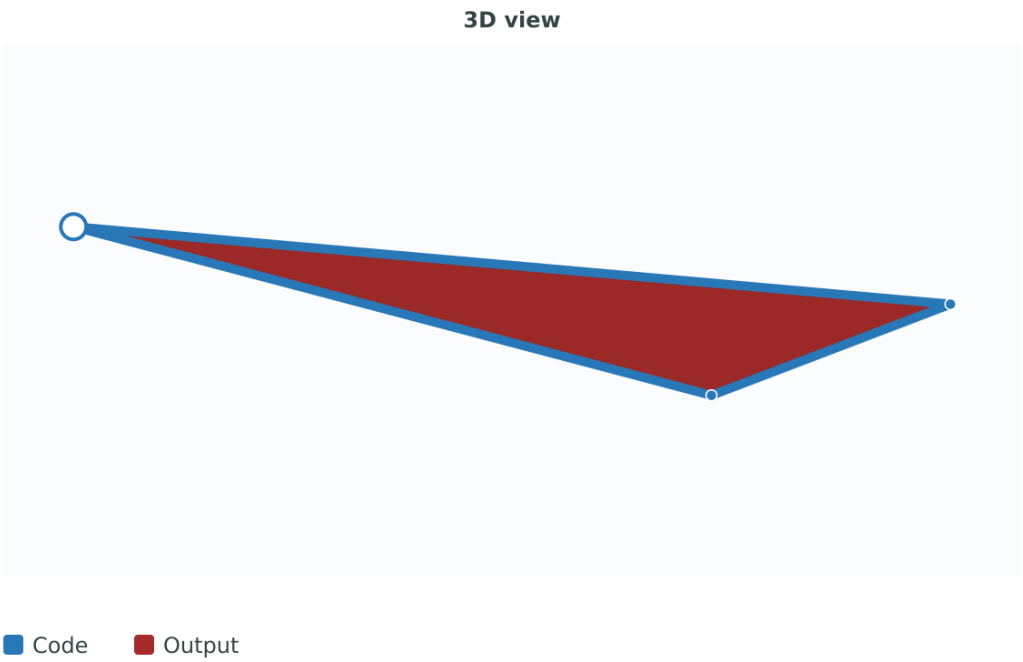
Code

```
SELECT ST_MakePolygon('LINESTRING(75.15 29.53 1,77 29 1,77.6 29.5 1,75.15 29.53 1)');
```

Output

```
POLYGON((75.15 29.53 1,77 29 1,77.6 29.5 1,75.15 29.53 1))
```

Figure

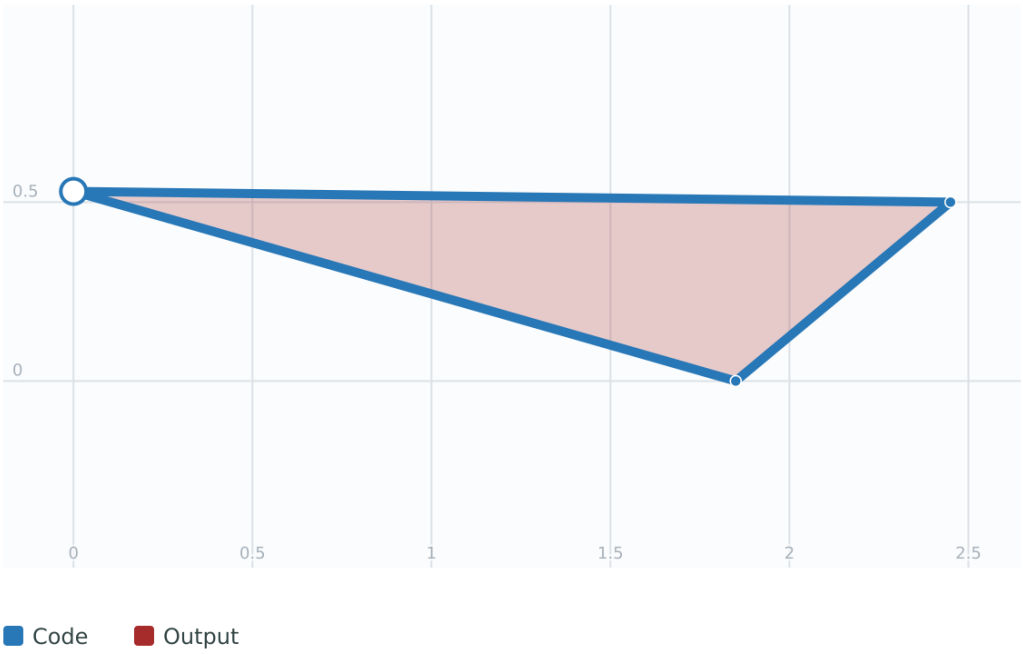


Skapa en polygon från en LineString med mått

Code

```
SELECT ST_MakePolygon('LINESTRINGM(75.15 29.53 1,77 29 1,77.6 29.5 2, 75.15 29.53 2)');
```

Figure



Create a CurvePolygon from a CircularString shell.

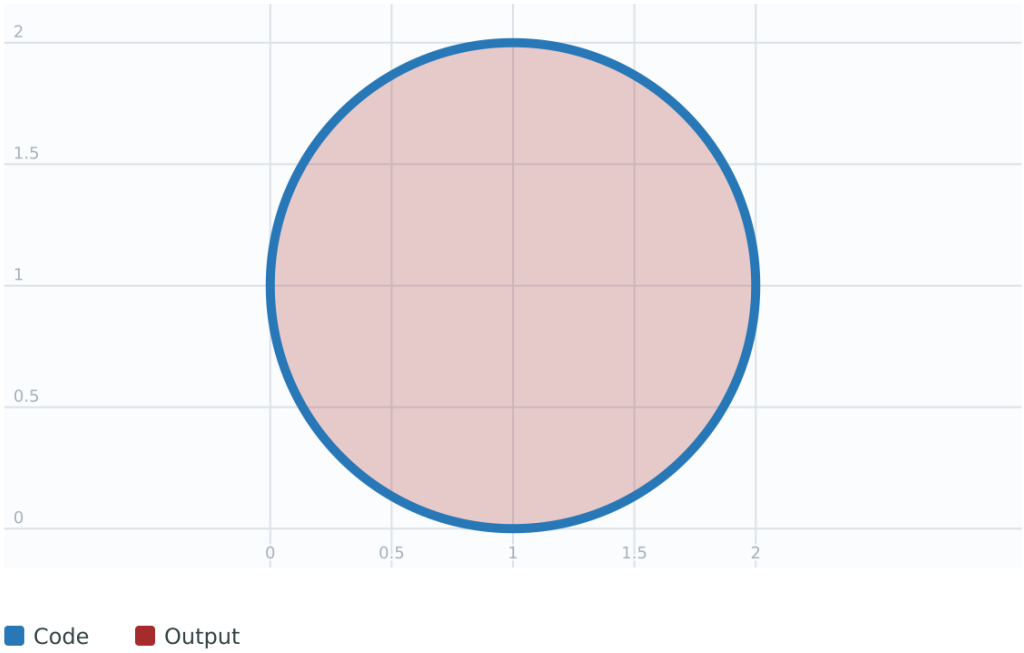
Code

```
SELECT ST_MakePolygon('CIRCULARSTRING(0 0,1 1,2 0,1 -1,0 0)');
```

Output

```
CURVEPOLYGON(CIRCULARSTRING(0 0,1 1,2 0,1 -1,0 0))
```

Figure



Outer shell with inner holes variant.

Skapa en munkpolygon med ett extra hål

Code

```
SELECT ST_MakePolygon(
    ST_ExteriorRing(ST_Buffer(ring.line, 10)),
    ARRAY[  ST_Translate(ring.line, 1, 1),
        ST_ExteriorRing(ST_Buffer(ST_Point(20, 20), 1)) ]
    )
FROM (SELECT ST_ExteriorRing(ST_Buffer(ST_Point(10, 10), 10, 10)) AS line ) AS ring;
```

Skapa en uppsättning provinsgränser med hål som representerar sjöar. Indata är en tabell med provinspolygoner/multipolygoner och en tabell med vattenlinjesträckningar. Linjer som bildar sjöar bestäms med hjälp av ST_IsClosed. Provinsernas linjeföring extraheras med hjälp av ST_Boundary. Som krävs av ST_MakePolygon tvingas gränsen att vara en enda LineString med hjälp av ST_LineMerge. (Observera dock att om en provins har mer än en region eller har öar kommer detta att ge en ogiltig polygon) Genom att använda en LEFT JOIN säkerställs att alla provinser inkluderas även om de inte har några sjöar.

Notera
CASE construct används eftersom en null-array som skickas till ST_MakePolygon resulterar i ett NULL-returvärde.

Code

```
SELECT p.gid, p.province_name,
  CASE
    WHEN array_agg(w.geom) IS NULL THEN p.geom
    ELSE ST_MakePolygon(
      ST_LineMerge(ST_Boundary(p.geom)),
      array_agg(w.geom)
    )
  END AS geom
FROM provinces p
LEFT JOIN waterlines w
  ON ST_Within(w.geom, p.geom)
  AND ST_IsClosed(w.geom)
GROUP BY p.gid, p.province_name, p.geom;
```

En annan teknik är att använda en korrelerad underfråga och ARRAY() constructor som omvandlar en raduppsättning till en array.

Code

```
SELECT p.gid, p.province_name,
  CASE
    WHEN EXISTS (
      SELECT w.geom
      FROM waterlines w
      WHERE ST_Within(w.geom, p.geom)
      AND ST_IsClosed(w.geom)
    )
    THEN ST_MakePolygon(
      ST_LineMerge(ST_Boundary(p.geom)),
      ARRAY(
        SELECT w.geom
        FROM waterlines w
        WHERE ST_Within(w.geom, p.geom)
        AND ST_IsClosed(w.geom)
      )
    )
    ELSE p.geom
  END AS geom
FROM provinces p;
```

Se även

ST_BuildArea ST_Polygon

7.3.8 ST_Point

ST_Point — Skapar en punkt med X-, Y- och SRID-värden.

Synopsis

geometry **ST_Point**(float x, float y);

geometry **ST_Point**(float x, float y, integer srid=unknown);

Beskrivning

Returnerar en Point med de angivna X- och Y-koordinatvärdena. Detta är SQL-MM:s motsvarighet till ST_MakePoint som bara tar X och Y.

Notera
För geodetiska koordinater är X longitud och Y latitud

Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 6.1.2

Exempel

Geometry.

Code

```
SELECT ST_Point(-71.104, 42.315);
```

Skapar en punkt med SRID specificerad:

Code

```
SELECT ST_Point(-71.104, 42.315, 4326);
```

Alternativt sätt att ange SRID:

Code

```
SELECT ST_SetSRID(ST_Point(-71.104, 42.315), 4326);
```

Skapa geografiska punkter med hjälp av syntaxen :: cast:

Code

```
SELECT ST_Point(-71.104, 42.315, 4326)::geography;
```

Kod före PostGIS 3.2, med hjälp av CAST:

Code

```
SELECT CAST(ST_SetSRID(ST_Point(-71.104, 42.315), 4326) AS geography);
```

Om punktkoordinaterna inte är i ett geodetiskt koordinatsystem (t.ex. WGS84) måste de omprojiceras innan de kan läggas in i en geografi. I detta exempel projiceras en punkt i Pennsylvania State Plane feet (SRID 2273) till WGS84 (SRID 4326).

Code

```
SELECT ST_Transform(ST_Point(3637510, 3014852, 2273), 4326)::geography;
```

Se även

ST_MakePoint, ST_PointZ, ST_PointM, ST_PointZM, ST_SetSRID, ST_Transform

7.3.9 ST_PointZ

ST_PointZ — Skapar en punkt med X-, Y-, Z- och SRID-värden.

Synopsis

geometry **ST_PointZ**(float x, float y, float z, integer srid=unknown);

Beskrivning

Returnerar en Point med de angivna koordinatvärdena X, Y och Z, och eventuellt ett SRID-nummer.

Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin.

Exempel

Code

```
SELECT ST_PointZ(-71.104, 42.315, 3.4, 4326)
```

Code

```
SELECT ST_PointZ(-71.104, 42.315, 3.4, srid => 4326)
```

Code

```
SELECT ST_PointZ(-71.104, 42.315, 3.4)
```

Se även

ST_MakePoint, ST_Point, ST_PointM, ST_PointZM

7.3.10 ST_PointM

ST_PointM — Skapar en punkt med X-, Y-, M- och SRID-värden.

Synopsis

geometry **ST_PointM**(float x, float y, float m, integer srid=unknown);

Beskrivning

Returnerar en Point med de angivna koordinatvärdena X, Y och M, och eventuellt ett SRID-nummer.

Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin.

Exempel

Code

```
SELECT ST_PointM(-71.104, 42.315, 3.4, 4326)
```

Code

```
SELECT ST_PointM(-71.104, 42.315, 3.4, srid => 4326)
```

Code

```
SELECT ST_PointM(-71.104, 42.315, 3.4)
```

Se även

ST_MakePoint, ST_Point, ST_PointZ, ST_PointZM

7.3.11 ST_PointZM

ST_PointZM — Skapar en punkt med X-, Y-, Z-, M- och SRID-värden.

Synopsis

geometry **ST_PointZM**(float x, float y, float z, float m, integer srid=unknown);

Beskrivning

Returnerar en Point med de angivna koordinatvärdena X, Y, Z och M, och eventuellt ett SRID-nummer.

Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin.

Exempel

Code

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5, 4326)
```

Code

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5, srid => 4326)
```

Code

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5)
```

Se även

ST_MakePoint, ST_Point, ST_PointM, ST_PointZ, ST_SetSRID

7.3.12 ST_Polygon

ST_Polygon — Skapar en polygon från en LineString med en angiven SRID.

Synopsis

geometry **ST_Polygon**(geometry lineString, integer srid);

Beskrivning

Returnerar en polygon som byggts från den angivna LineString och anger det spatiala referenssystemet från `srid`.

ST_Polygon liknar ST_MakePolygon Variant 1 med tillägget att man kan ställa in SRID.

För att skapa polygoner med hål använder du ST_MakePolygon Variant 2 och sedan ST_SetSRID.

Notera
Denna funktion accepterar inte MultiLineStrings. Använd ST_LineMerge för att generera en LineString, eller ST_Dump för att extrahera LineStrings.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.3.2

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Skapa en 2D-polygon.

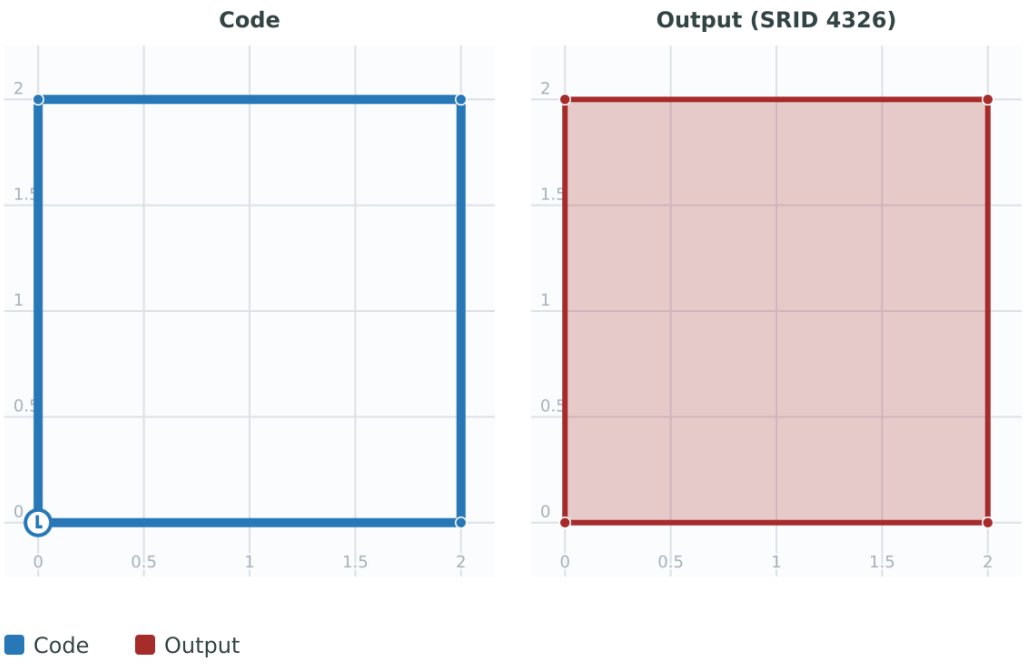
Code

```
SELECT ST_Polygon('LINESTRING(75 29,77 29,77 31,75 31,75 29)'::geometry, 4326);
```

Output

```
POLYGON((75 29,77 29,77 31,75 31,75 29))
```

Figure



Skapa en 3D-polygon.

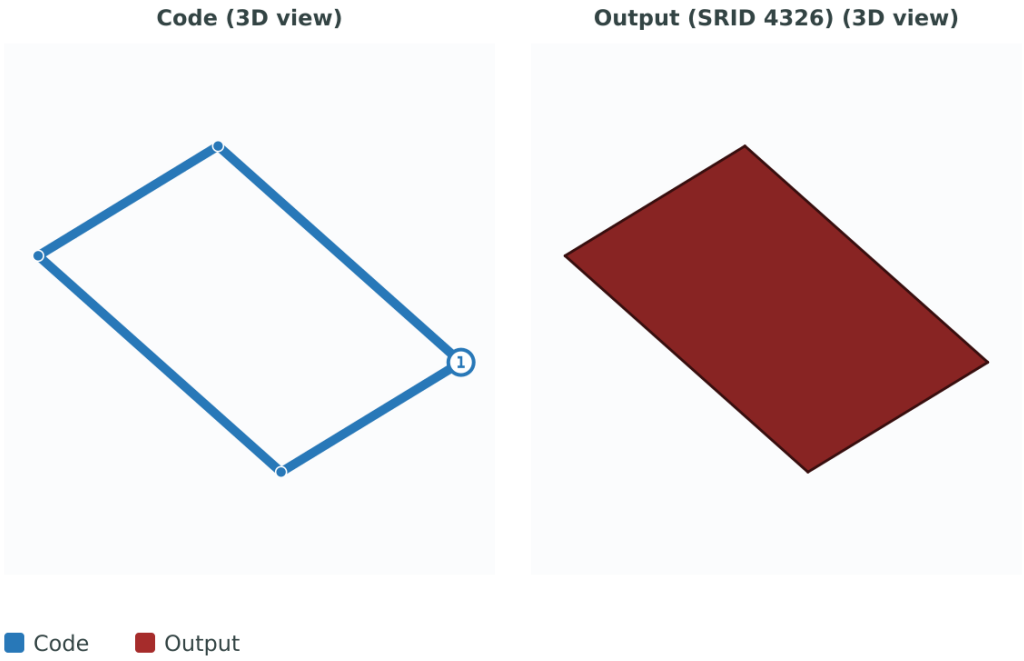
Code

```
SELECT ST_Polygon('LINESTRING(75 29 1,77 29 2,77 31 2,75 31 1,75 29 1)'::geometry, 4326);
```

Output

```
SRID=4326;POLYGON((75 29 1,77 29 2,77 31 2,75 31 1,75 29 1))
```

Figure



Se även

ST_AsEWKT, ST_AsText, ST_GeomFromEWKT, ST_GeomFromText, ST_LineMerge, ST_MakePolygon

7.3.13 ST_TileEnvelope

ST_TileEnvelope — Skapar en rektangulär polygon i Web Mercator (SRID:3857) med hjälp av XYZ-plattsystemet.

Synopsis

geometry **ST_TileEnvelope**(integer tileZoom, integer tileX, integer tileY, geometry bounds=SRID=3857;LI
20037508.342789 -20037508.342789,20037508.342789 20037508.342789), float margin=0.0);

Beskrivning

Skapar en rektangulär polygon som motsvarar utsträckningen av en platta i XYZ-plattsystemet. Plattan specificeras av zoomnivån Z och XY-indexet för plattan i rutnätet på den nivån. Kan användas för att definiera de plattgränser som krävs av ST_AsMVTGeom för att konvertera geometri till MVT-koordinatrymden för plattan.

Som standard är plattans kuvert i koordinatsystemet Web Mercator (SRID:3857) med standardintervallet för systemet Web Mercator (-20037508.342789, 20037508.342789). Detta är det vanligaste koordinatsystemet som används för MVT-plattor. Den valfria parametern `bounds` kan användas för att generera plattor i vilket koordinatsystem som helst. Det är en geometri som har SRID och utsträckning för ”Zoom Level zero-kvadraten inom vilken XYZ-plattsystemet är inskrivet.

Den valfria `marginalparametern` kan användas för att expandera en platta med den angivna procentsatsen. T.ex. `margin=0,125` expanderar plattan med 12,5%, vilket motsvarar buffer=512 när plattans storlek är 4096, som används i ST_AsMVTGeom. Detta är användbart för att skapa en plattbuffert för att inkludera data som ligger utanför plattans synliga område, men vars existens påverkar plattrenderingen. Ett stadsnamn (en punkt) kan t.ex. ligga nära kanten på en platta, så dess etikett bör återges på två plattor, även om punkten ligger i det synliga området på bara en platta. Om du använder expanderade plattor i en fråga kommer stadspunkten att inkluderas i båda plattorna. Använd ett negativt värde för att krympa plattan istället. Värden mindre än -0,5 är förbjudna eftersom det skulle eliminera plattan helt. Ange inte en marginal när du använder `ST_AsMVTGeom`. Se exemplet för ST_AsMVT.

Förbättrad: 3.1.0 Lagt till marginalparameter.

Tillgänglighet: 3.0.0

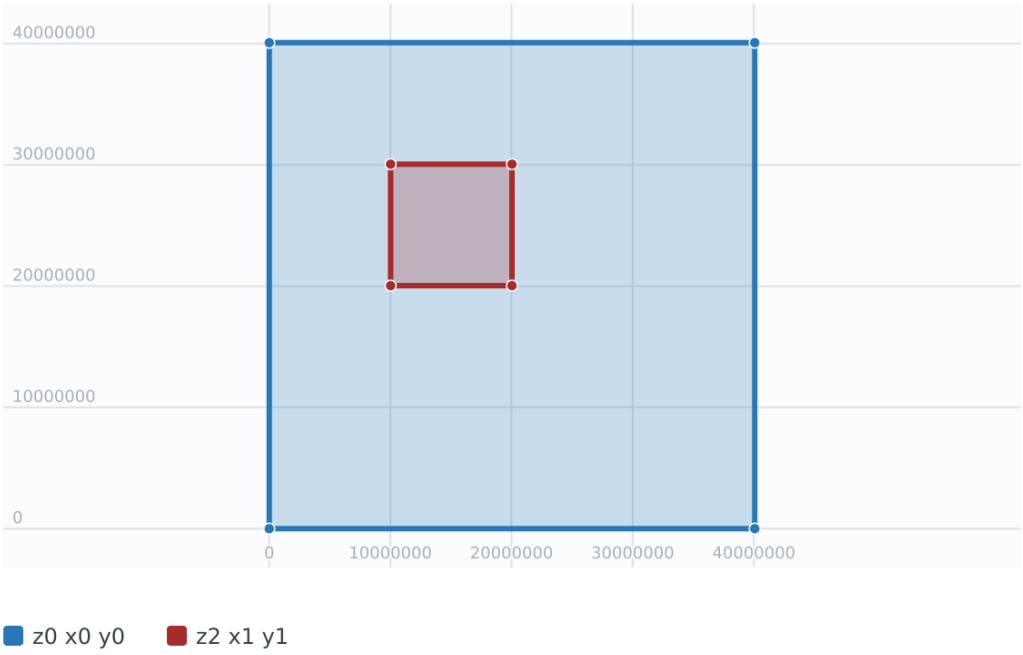
Exempel

Show tile (2, 1, 1) within the root tile (0, 0, 0).

Code

```
SELECT ST_TileEnvelope(0, 0, 0) AS input_z0_x0_y0, ST_TileEnvelope(2, 1, 1) AS z2_x1_y1;
```

Figure



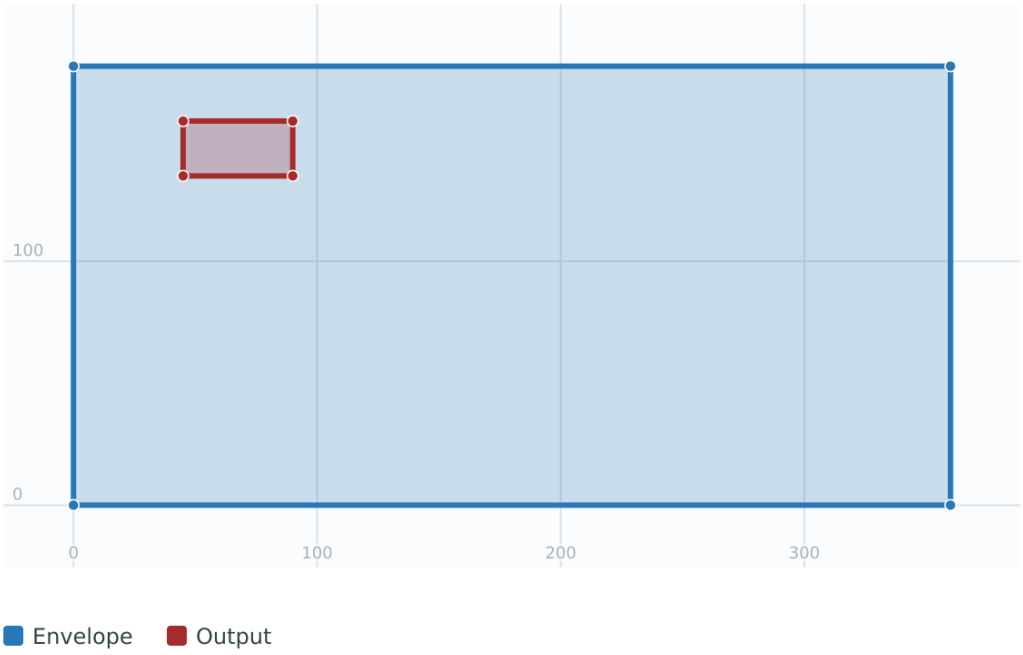
Code

```
SELECT ST_TileEnvelope(3, 1, 1, ST_MakeEnvelope(-180, -90, 180, 90, 4326));
```

Output

```
POLYGON((-135 45,-135 67.5,-90 67.5,-90 45,-135 45))
```

Figure



Se även

ST_MakeEnvelope

7.3.14 ST_HexagonGrid

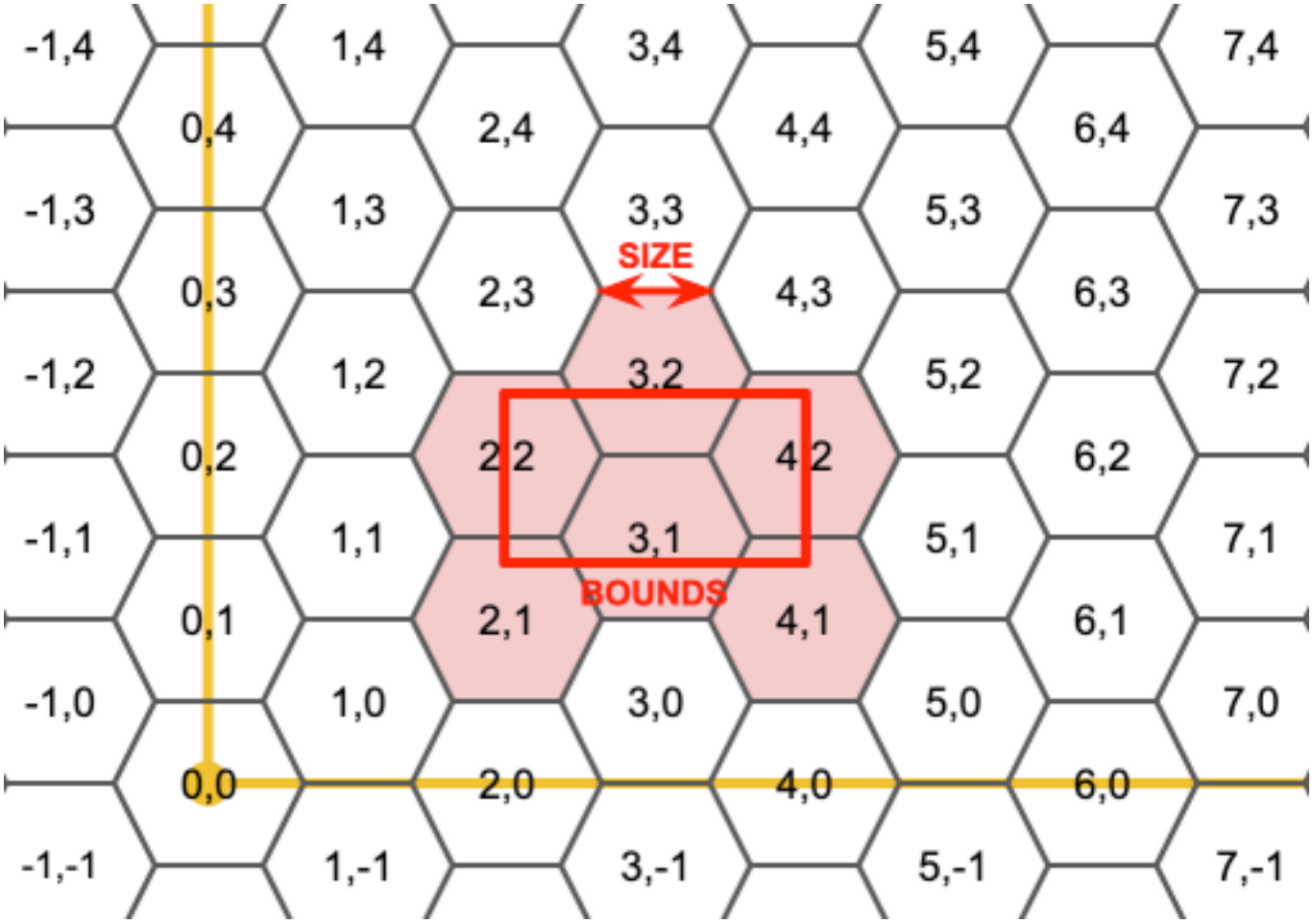
ST_HexagonGrid — Returnerar en uppsättning hexagoner och cellindex som helt täcker gränserna för geometriargumentet.

Synopsis

setof record **ST_HexagonGrid**(float8 size, geometry bounds);

Beskrivning

Starts with the concept of a hexagon tiling of the plane. (Not a hexagon tiling of the globe; for H3 use the h3-pg extension.) For a given planar SRS, and a given edge size, starting at the origin of the SRS, there is one unique hexagonal tiling of the plane, Tiling(SRS, Size). This function answers the question: what hexagons in a given Tiling(SRS, Size) overlap with a given bounds.



SRS för utdatahexagonerna är den SRS som tillhandahålls av begränsningsgeometrin.

Om man dubblar eller tripplar hexagonens kantstorlek genereras en ny parent tiling som passar med origin tilingen. Tyvärr är det inte möjligt att generera överordnade hexagonbeläggningar som barn-beläggningarna passar perfekt i.

Tillgänglighet: 3.1.0

Exempel

Unit hexagons covering rectangular bounds.

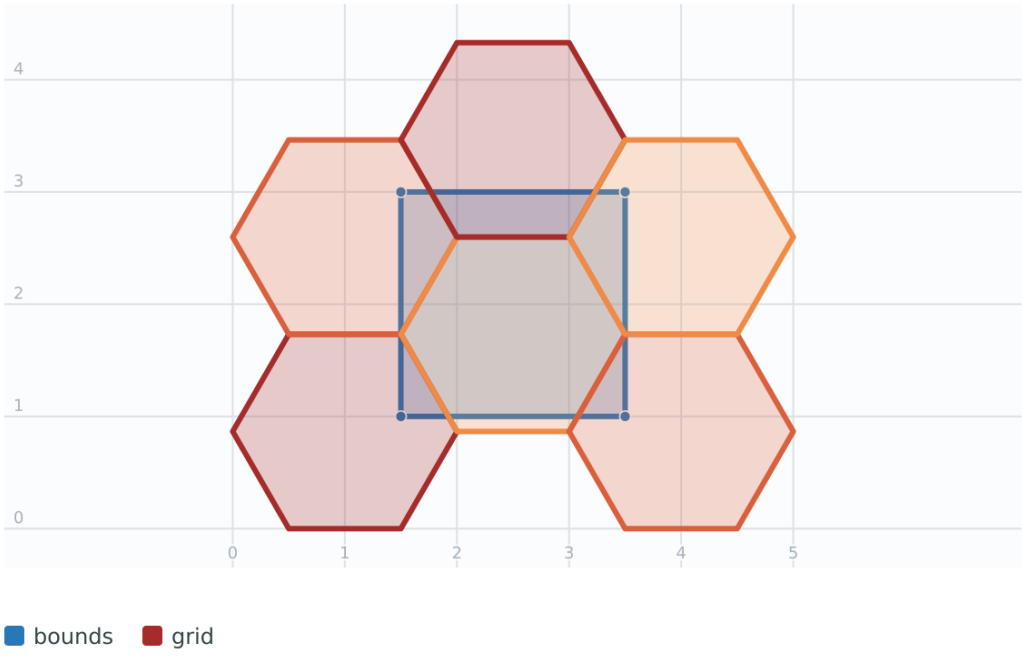
Code

```
WITH bounds AS (
    SELECT ST_MakeEnvelope(2, 1, 4, 3, 3857) AS geom
), grid AS (
    SELECT hex.*
    FROM bounds
    CROSS JOIN LATERAL ST_HexagonGrid(1, geom) AS hex
)
SELECT (SELECT geom FROM bounds) AS input_bounds,
    ST_Collect(geom ORDER BY i, j) AS grid
FROM grid;
```

Output

```
POLYGON((2 1,2 3,4 3,4 1,2 1)) | MULTIPOLYGON(((0.5 0.866,1 0,2 0,2.5 0.866,2 1.732,1 1.732,0.5 0.866)),((0.5 2.598,1 ↩
    1.732,2 1.732,2.5 2.598,2 3.464,1 3.464,0.5 2.598)),((2 1.732,2.5 0.866,3.5 0.866,4 1.732,3.5 2.598,2.5 2.598,2 ↩
    1.732)),((2 3.464,2.5 2.598,3.5 2.598,4 3.464,3.5 4.33,2.5 4.33,2 3.464)),((3.5 0.866,4 0,5 0,5.5 0.866,5 1.732,4 ↩
     1.732,3.5 0.866)),((3.5 2.598,4 1.732,5 1.732,5.5 2.598,5 3.464,4 3.464,3.5 2.598)))
```

Figure



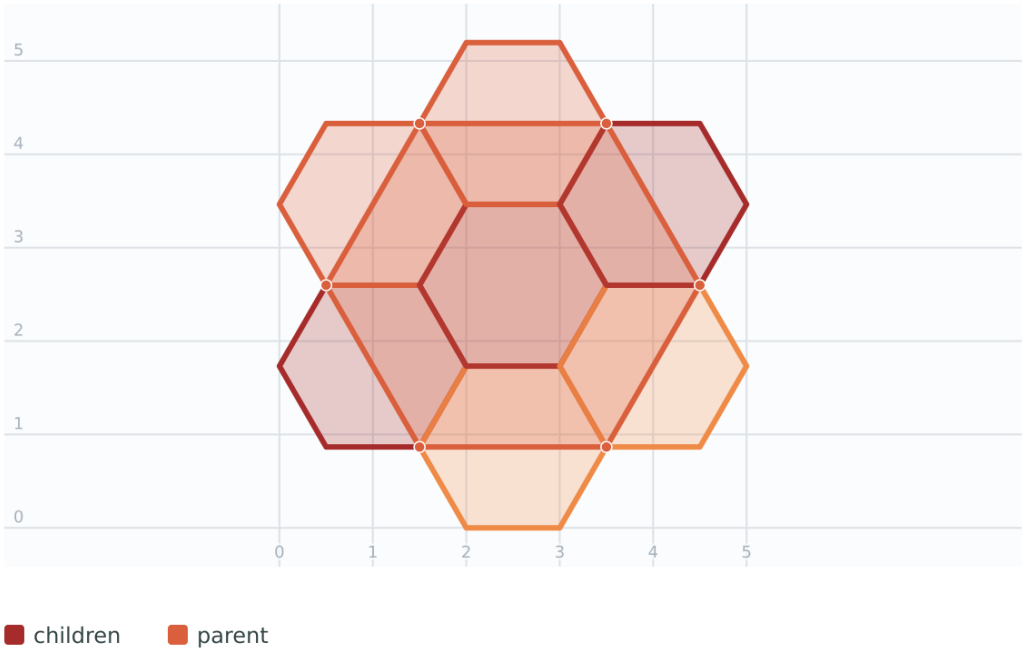
Child tiles that intersect a doubled-size parent hexagon share the tiling origin, but do not nest perfectly inside the parent.

Code

```
WITH parent AS (
    SELECT ST_Hexagon(2, 0, 0) AS geom
), children AS (
    SELECT child.*
```

```
    FROM parent
    CROSS JOIN LATERAL ST_HexagonGrid(1, geom) AS child
    WHERE ST_Area(ST_Intersection(child.geom, parent.geom)) > 0
)
SELECT ST_Collect(geom ORDER BY i, j) AS children,
    (SELECT geom FROM parent) AS parent
FROM children;
```

Figure



Counting points in hexagons.

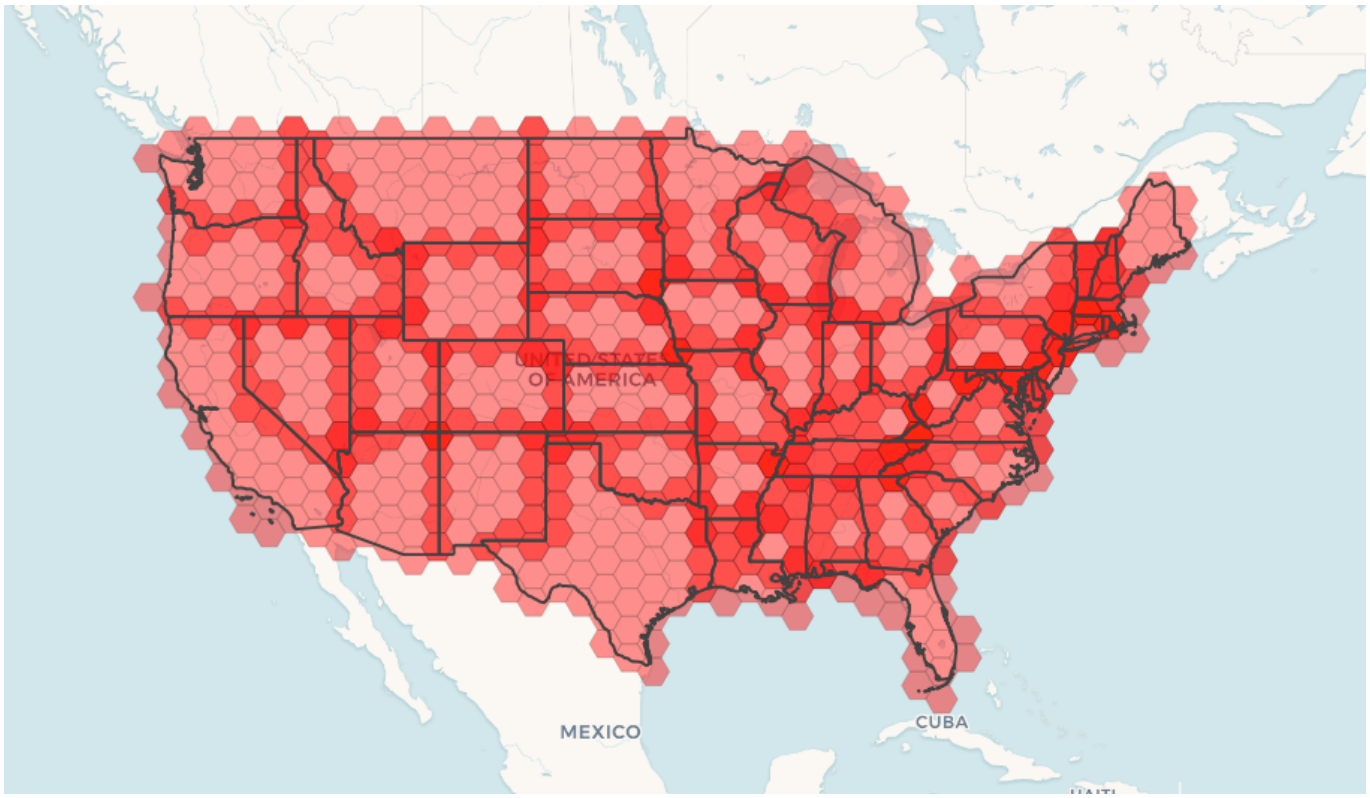
För att göra en punktsammanfattning mot ett hexagonalt rutnät genererar du ett hexagonalt rutnät med punkternas utsträckning som gränser och ansluter sedan spatialt till det rutnätet.

Code

```
SELECT COUNT(*), hexes.geom
FROM ST_HexagonGrid(
  10000,
  ST_SetSRID(ST_EstimatedExtent('pointtable', 'geom'), 3857)
) AS hexes
INNER JOIN pointtable AS pts
  ON ST_Intersects(pts.geom, hexes.geom)
GROUP BY hexes.geom;
```

Generating hex coverage of polygons.

If a grid is generated for each polygon boundary and filtered to intersecting cells, adjacent regions receive overlapping hexagons along their shared border. The build-time figure below uses two simple regions in place of an external administrative-boundary table, so the example is self-contained.



Code

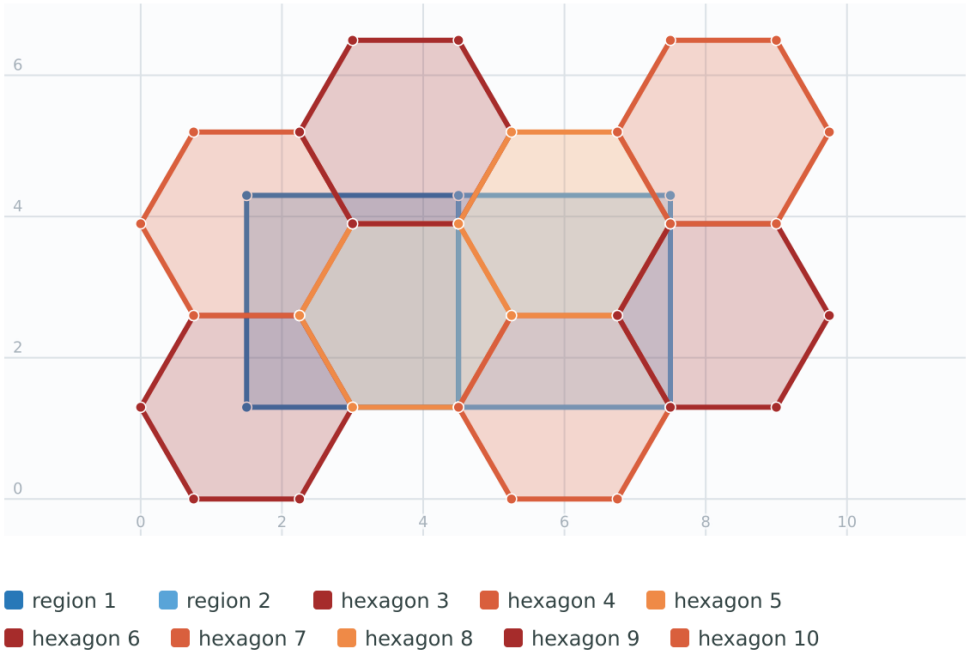
```
WITH admin1(gid, geom) AS (
    VALUES
        (1, ST_MakeEnvelope(0, 0, 3, 3, 3857)),
        (2, ST_MakeEnvelope(3, 0, 6, 3, 3857))
), coverage AS (
    SELECT hex.i, hex.j, hex.geom
    FROM admin1
    CROSS JOIN LATERAL ST_HexagonGrid(1.5, admin1.geom) AS hex
    WHERE ST_Intersects(admin1.geom, hex.geom)
    GROUP BY hex.i, hex.j, hex.geom
), layers AS (
    SELECT 'region' AS kind, gid AS position, geom
    FROM admin1
    UNION ALL
    SELECT
        'hexagon',
        1000 + row_number() OVER (ORDER BY i, j),
        geom
    FROM coverage
)
SELECT
    CASE kind WHEN 'region' THEN geom END AS input_region,
    CASE kind WHEN 'hexagon' THEN geom END AS hexagon
FROM layers
ORDER BY position;
```

Output

```
POLYGON((0 0,0 3,3 3,3 0,0 0)) | null
POLYGON((3 0,3 3,6 3,6 0,3 0)) | null
null | POLYGON((-1.5 0,-0.75 -1.3,0.75 -1.3,1.5 0,0.75 1.3,-0.75 1.3,-1.5 0))
null | POLYGON((-1.5 2.6,-0.75 1.3,0.75 1.3,1.5 2.6,0.75 3.9,-0.75 3.9,-1.5 2.6))
null | POLYGON((0.75 1.3,1.5 0,3 0,3.75 1.3,3 2.6,1.5 2.6,0.75 1.3))
null | POLYGON((0.75 3.9,1.5 2.6,3 2.6,3.75 3.9,3 5.2,1.5 5.2,0.75 3.9))
```

```
null | POLYGON((3 0,3.75 -1.3,5.25 -1.3,6 0,5.25 1.3,3.75 1.3,3 0))
null | POLYGON((3 2.6,3.75 1.3,5.25 1.3,6 2.6,5.25 3.9,3.75 3.9,3 2.6))
null | POLYGON((5.25 1.3,6 0,7.5 0,8.25 1.3,7.5 2.6,6 2.6,5.25 1.3))
null | POLYGON((5.25 3.9,6 2.6,7.5 2.6,8.25 3.9,7.5 5.2,6 5.2,5.25 3.9))
```

Figure



The same pattern applies to a real administrative-boundary table:

Notera
Nyckelordet LATERAL är underförstått för set-returning-funktioner när man hänvisar till en tidigare tabell i FROM-listan. Så CROSS JOIN LATERAL, CROSS JOIN eller helt enkelt , är likvärdiga constructs för det här exemplet.

Code

```
SELECT admin1.gid, hex.geom
FROM
    admin1
    CROSS JOIN
    ST_HexagonGrid(100000, admin1.geom) AS hex
WHERE
    adm0_a3 = 'USA'
    AND
    ST_Intersects(admin1.geom, hex.geom)
```

Se även

ST_EstimatedExtent, ST_SetSRID, ST_SquareGrid, ST_TileEnvelope

7.3.15 ST_Hexagon

ST_Hexagon — Returnerar en enda hexagon med den angivna kantstorleken och cellkoordinaten inom hexagonens rutnätsutrymme.

Synopsis

geometry **ST_Hexagon**(float8 size, integer cell_i, integer cell_j, geometry origin);

Beskrivning

Använder samma hexagontilingkoncept som ST_HexagonGrid, men genererar bara en hexagon vid önskad cellkoordinat. Eventuellt kan du justera ursprungskoordinaten för kaklet, standardkoordinaten är 0,0.

Hexagoner genereras utan att någon SRID har angetts, så använd ST_SetSRID för att ange den SRID du förväntar dig.

Tillgänglighet: 3.1.0

Exempel

Creating a hexagon at the origin.

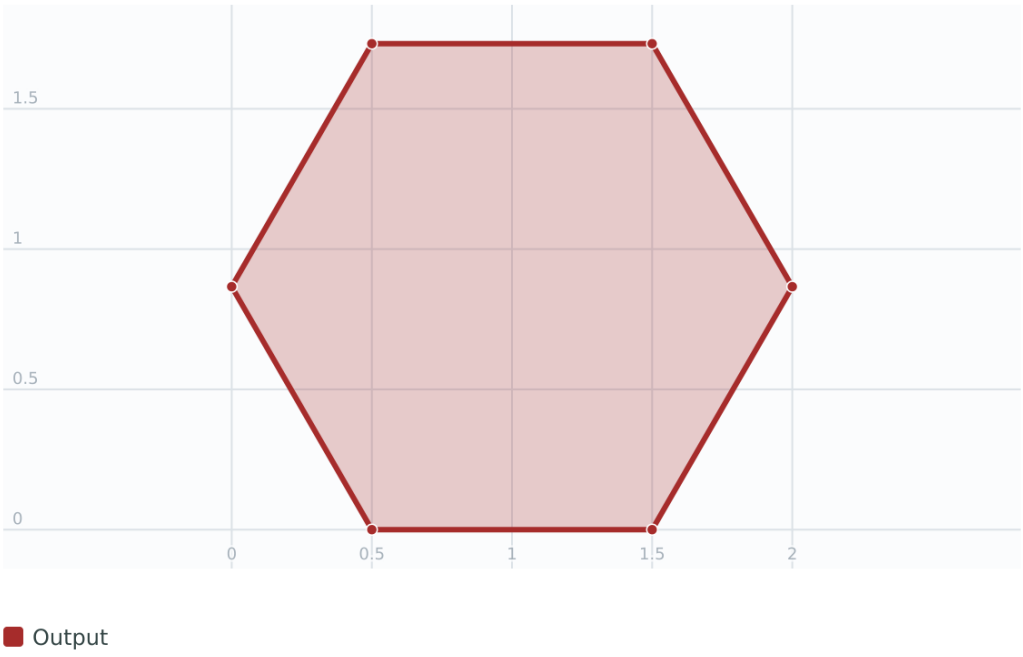
Code

```
SELECT ST_SetSRID(ST_Hexagon(1.0, 0, 0), 3857);
```

Output

```
POLYGON((-1 0,-0.5 -0.8660254037844387,0.5 -0.8660254037844387,1 0,0.5 0.8660254037844387,-0.5 0.8660254037844387,-1 ↩
    0))
```

Figure



Se även

ST_TileEnvelope, ST_HexagonGrid, ST_Square

7.3.16 ST_SquareGrid

ST_SquareGrid — Returnerar en uppsättning rutnätsrutor och cellindex som helt täcker gränserna för geometriargumentet.

Synopsis

setof record **ST_SquareGrid**(float8 size, geometry bounds);

Beskrivning

Börjar med konceptet av en kvadratisk tiling av planet. För en given plan SRS och en given kantstorlek, med början vid SRS:ens ursprung, finns det en unik kvadratisk tiling av planet, Tiling(SRS, Size). Denna funktion svarar på frågan: vilka rutnät i en given Tiling(SRS, Size) överlappar med en given gräns.

SRS för utdatarutorna är den SRS som tillhandahålls av avgränsningsgeometrin.

Dubblering eller kantstorlek av kvadraten genererar en ny överordnad tiling som passar perfekt med den ursprungliga tilingen. Vanliga webbkartor i mercator är bara tvåpotenser av två kvadratiska rutnät i mercatorplanet.

Tillgänglighet: 3.1.0

Exempel

Generating a 1 degree grid for a country.

Rutnätet kommer att fylla hela landets gränser, så om du bara vill ha rutor som berör landet måste du filtrera efteråt med ST_Intersects.

Code

```
WITH grid AS (
SELECT (ST_SquareGrid(1, ST_Transform(geom, 4326))).*
FROM admin0 WHERE name = 'Canada'
)
  SELEcT geom
  FROM grid
```

Counting points in squares (using single chopped grid).

För att göra en punktsammanfattning mot en kvadratisk platta, generera ett kvadratiskt rutnät med hjälp av punkternas utsträckning som gränser och sedan spatialt ansluta till det rutnätet. Observera att den beräknade omfattningen kan skilja sig från den faktiska omfattningen, så var försiktig och se åtminstone till att du har analyserat din tabell.

Code

```
SELECT COUNT(*), squares.geom
FROM pointtable AS pts
INNER JOIN ST_SquareGrid(
  1000,
  ST_SetSRID(ST_EstimatedExtent('pointtable', 'geom'), 3857)
) AS squares
  ON ST_Intersects(pts.geom, squares.geom)
GROUP BY squares.geom;
```

Counting points in squares using set of grid per point.

Detta ger samma resultat som i det första exemplet, men blir långsammare för ett stort antal punkter

Code

```
SELECT COUNT(*), squares.geom
FROM pointtable AS pts
INNER JOIN ST_SquareGrid(
  1000,
  pts.geom
) AS squares
  ON ST_Intersects(pts.geom, squares.geom)
GROUP BY squares.geom;
```

Se även

ST_TileEnvelope, ST_HexagonGrid, ST_EstimatedExtent, ST_SetSRID

7.3.17 ST_Square

ST_Square — Returnerar en enda kvadrat med hjälp av den angivna kantstorleken och cellkoordinaten inom kvadratens rutnätsutrymme.

Synopsis

geometry **ST_Square**(float8 size, integer cell_i, integer cell_j, geometry origin='POINT(0 0)');

Beskrivning

Använder samma koncept med kvadratiska plattor som ST_SquareGrid, men genererar bara en kvadrat i önskad cellkoordinat. Eventuellt kan du justera ursprungskoordinaten för kaklet, standardkoordinaten är 0,0.

Kvadrater genereras med SRID för det givna ursprunget. Använd ST_SetSRID för att ställa in SRID om det angivna ursprunget har ett okänt SRID (vilket är fallet som standard).

Tillgänglighet: 3.1.0

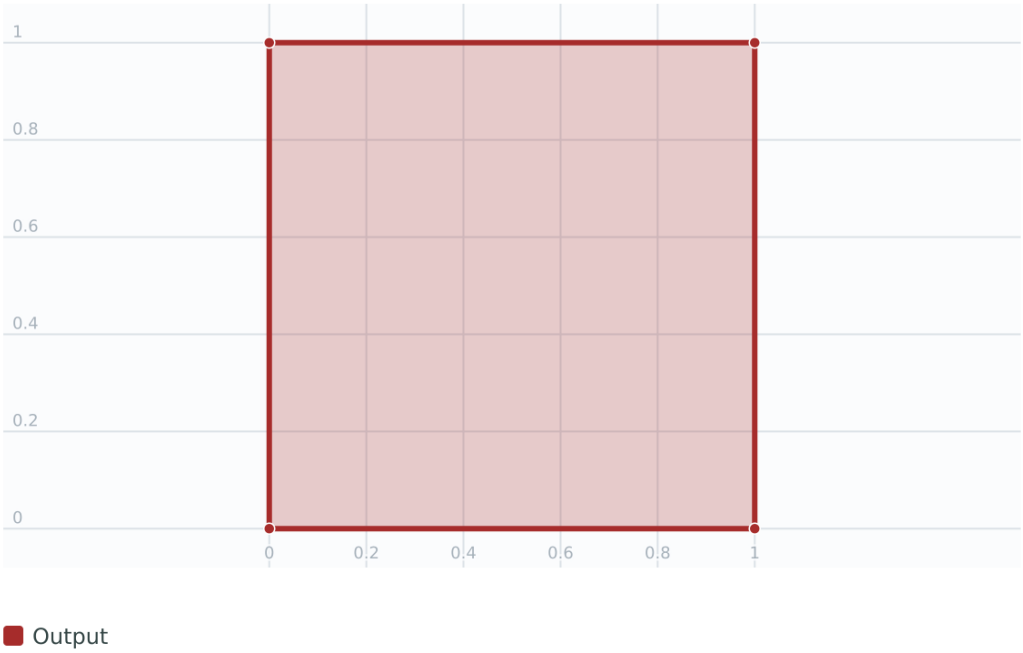
Exempel

Creating a square at the origin.

Code

```
SELECT ST_SetSRID(ST_Square(1.0, 0, 0), 3857);
```

Figure



Se även

ST_TileEnvelope, ST_SquareGrid, ST_Hexagon

7.3.18 ST_Letters

ST_Letters — Returnerar inmatade bokstäver som återges som geometri med en standardstartposition vid origo och en standardtexthöjd på 100.

Synopsis

geometry **ST_Letters**(text letters, json font);

Beskrivning

Använder ett inbyggt teckensnitt för att rendera en sträng som en multipolygongeometri. Standardhöjden för texten är 100,0, avståndet från botten av en nedåtriktad bokstav till toppen av en stor bokstav. Standardstartpositionen placerar början av baslinjen vid origo. Att åsidosätta teckensnittet innebär att en json-karta skickas in, med ett tecken som nyckel och base64-kodade TWKB för teckensnittsformen, med teckensnitt som har en höjd på 1000 enheter från botten av de nedåtgående till toppen av de stora bokstäverna.

Texten genereras som standard i origo, så för att flytta och ändra storlek på texten måste du först använda funktionen `ST_Scale` och sedan funktionen `ST_Translate`.

Tillgänglighet: 3.3.0

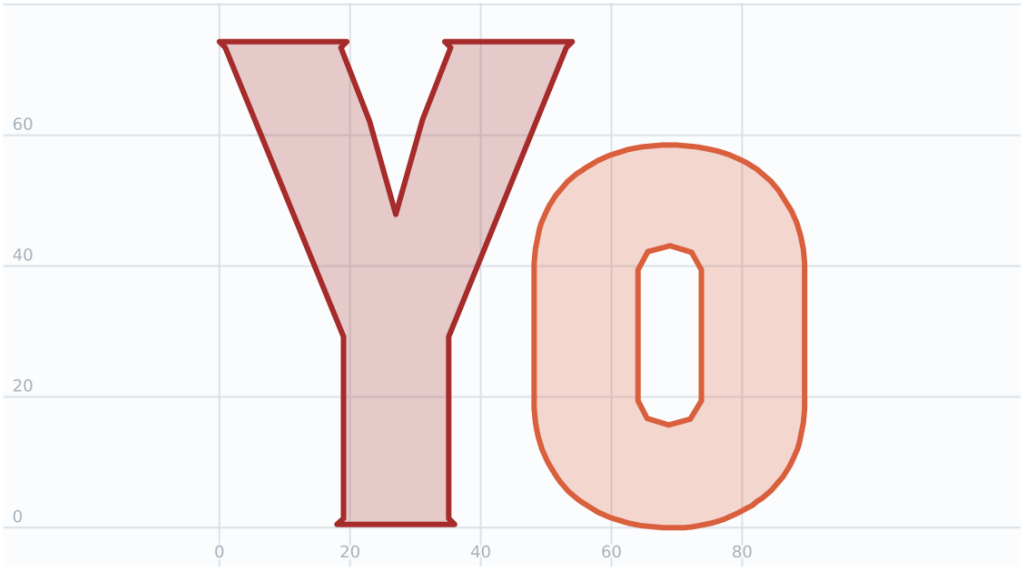
Exempel

Generating the word 'Yo'.

Code

```
SELECT ST_Letters('Yo');
```

Figure



Output

Generating the word 'PostGIS'.

Code

```
SELECT ST_Letters('PostGIS') AS letters;
```

Figure

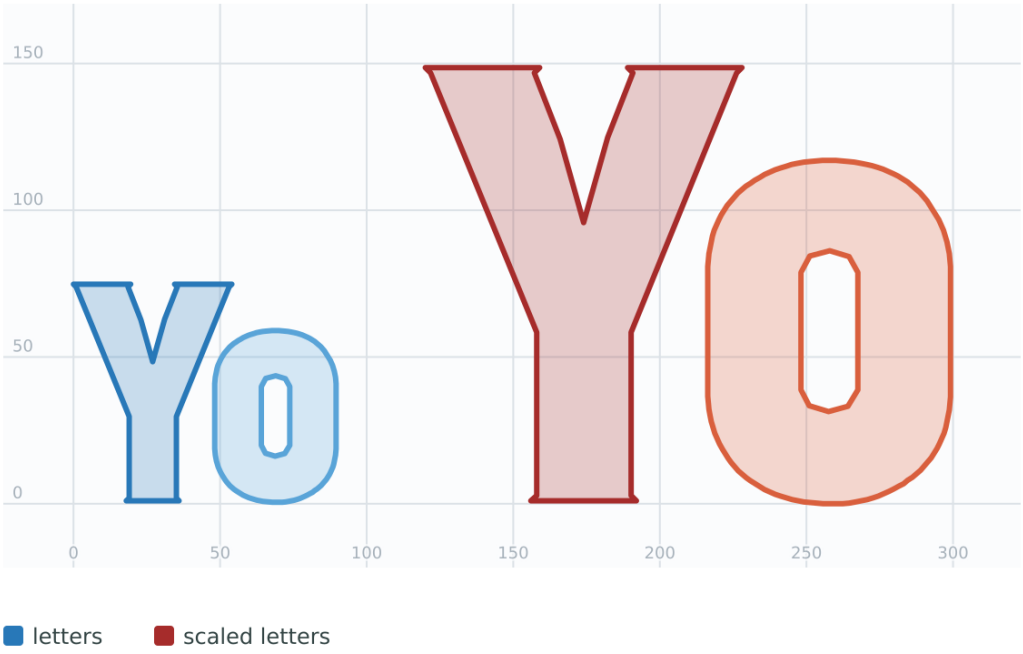
Output

Scaling and moving words.

Code

```
WITH letters AS (
    SELECT ST_Letters('Yo') AS geom
)
SELECT geom AS input_letters,
       ST_Translate(ST_Scale(geom, 2, 2), 120, 0)
           AS scaled_letters
FROM letters;
```

Figure



Se även

ST_AsTWKB, ST_Scale, ST_Translate

7.3.19 ST_MakeNurbsCurve

ST_MakeNurbsCurve — Creates a NURBS (Non-Uniform Rational B-Spline) curve from control points, optional weights, and optional knot vector.

Synopsis

geometry **ST_MakeNurbsCurve**(integer degree, geometry control_points, float8[] weights, float8[] knots);

Beskrivning

Creates a NURBS (Non-Uniform Rational B-Spline) curve geometry from the specified parameters. NURBS curves are smooth parametric curves that provide precise control over curve shape through control points, weights, and knot values.

The `degree` parameter specifies the polynomial degree of the curve (must be between 1 and 10). Higher degrees produce smoother curves.

The `control_points` must be a LINESTRING geometry defining the control polygon. The number of control points must be at least `degree` + 1.

The optional `weights` array assigns a weight to each control point. All weights must be positive. If NULL or omitted, uniform weights (1.0) are used, resulting in a non-rational B-spline.

The optional `knots` array defines the parameter space. If NULL or omitted, a uniform clamped knot vector is automatically generated. If provided, the knot vector must have exactly (`npoints` + `degree` + 1) elements and must be non-decreasing.

Availability: 3.7.0

Exempel

This example creates a degree-2 NURBS curve with uniform weights.

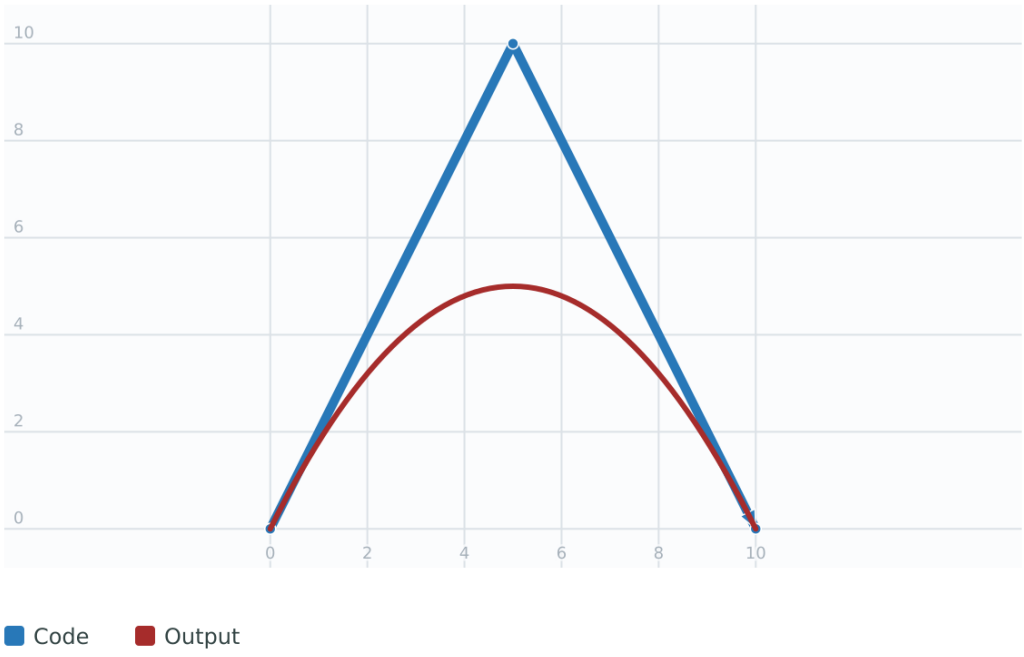
Code

```
SELECT ST_MakeNurbsCurve(
    2,
    'LINESTRING(0 0,5 10,10 0)'::geometry
);
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 10),WEIGHT 1),  ↩
    NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



This example creates a rational NURBS curve with custom weights.

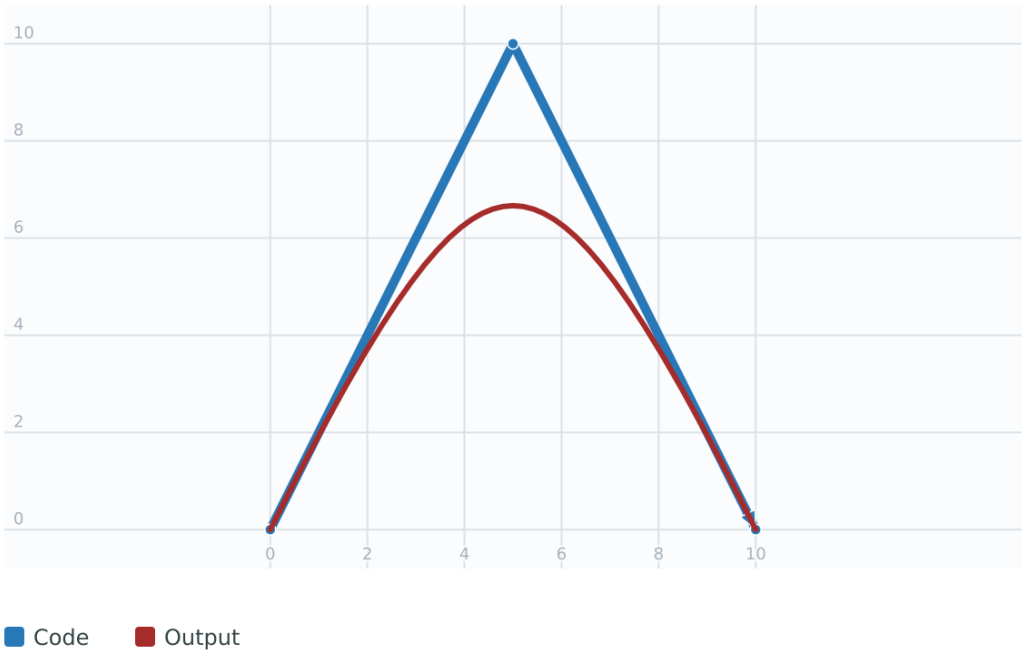
Code

```
SELECT ST_MakeNurbsCurve(
    2,
    'LINESTRING(0 0,5 10,10 0)'::geometry,
    ARRAY[1.0, 2.0, 1.0]
);
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 10),WEIGHT 2),  ↩
    NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



This example creates a NURBS curve with a non-uniform knot vector.

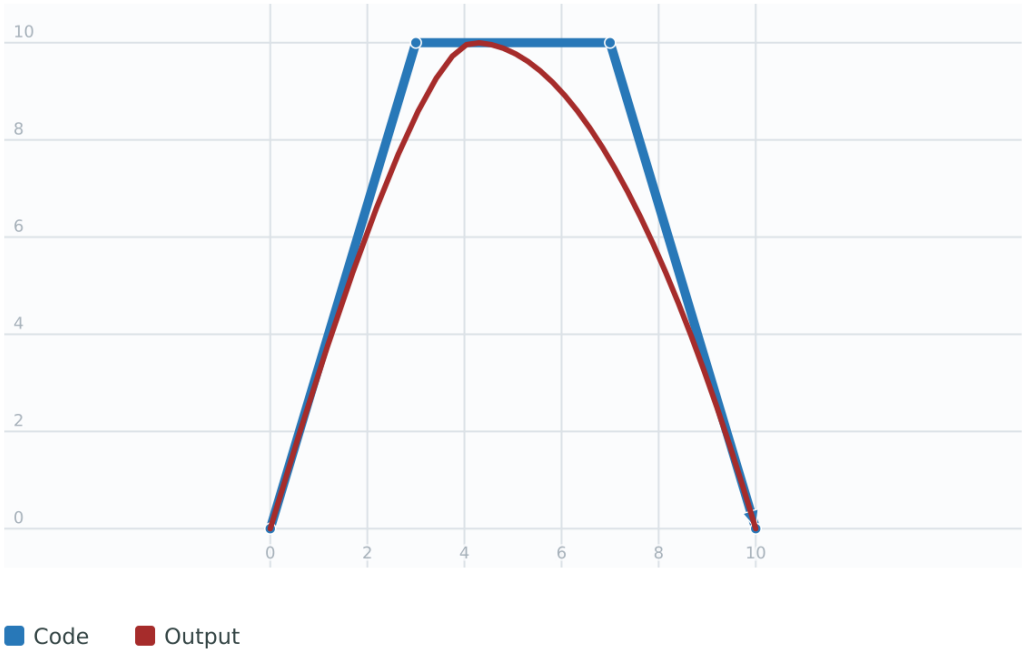
Code

```
SELECT ST_MakeNurbsCurve(
    2,
    'LINESTRING(0 0,3 10,7 10,10 0)'::geometry,
    ARRAY[1.0, 1.0, 1.0, 1.0],
    ARRAY[0.0, 0.0, 0.0, 0.3, 1.0, 1.0, 1.0]
);
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(3 10),WEIGHT 1),  ↩
    NURBSPOINT(WEIGHTEDPOINT(7 10),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(0.3,1) ↩
    , KNOT(1,3)))
```

Figure



Se även

ST_NurbsToLineString, ST_Evaluate, ST_ControlPoints

7.4 Geometriåtkomst

7.4.1 GeometryType

GeometryType — Returnerar typen av geometri som text.

Synopsis

text **GeometryType**(geometry geomA);

Beskrivning

Returnerar geometrins typ som en sträng. T.ex.: LINESTRING”, POLYGON”, MULTIPOINT”, etc.

OGC SPEC s2.1.1.1 - Returnerar namnet på den instantierbara subtypen av Geometry som denna Geometry-instans är medlem av. Namnet på den instansiella subtypen av Geometry returneras som en sträng.

Notera
Denna funktion anger också om geometrin är uppmätt genom att returnera en sträng av formen POINTM”.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

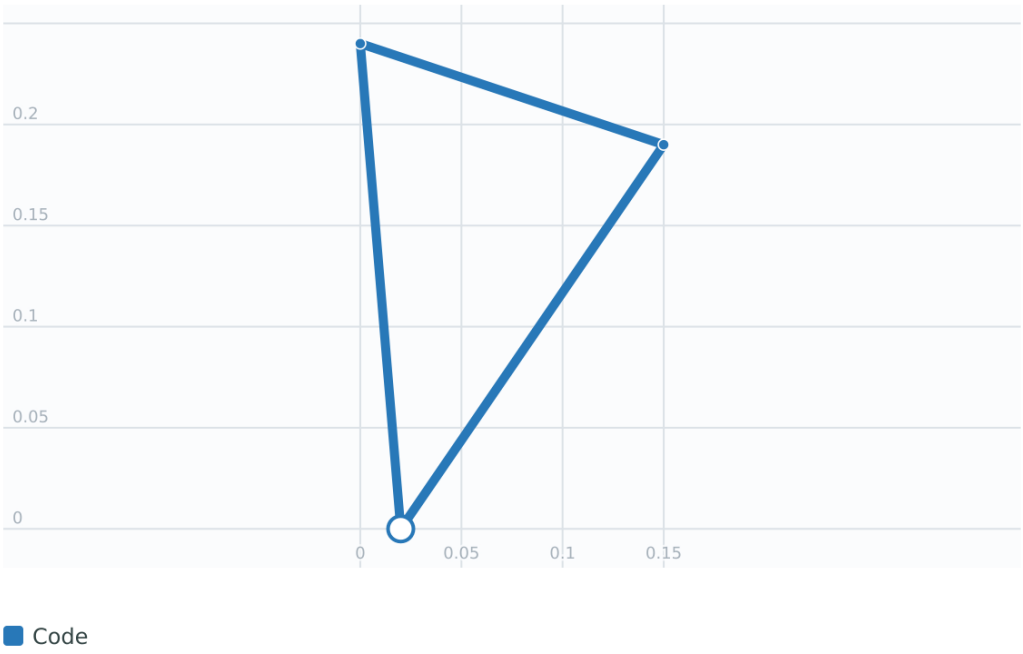
Code

```
SELECT GeometryType('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'::geometry ↩
    );
```

Output

```
LINESTRING
```

Figure



Code

```
SELECT ST_GeometryType('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )');
    POLYHEDRALSURFACE
```

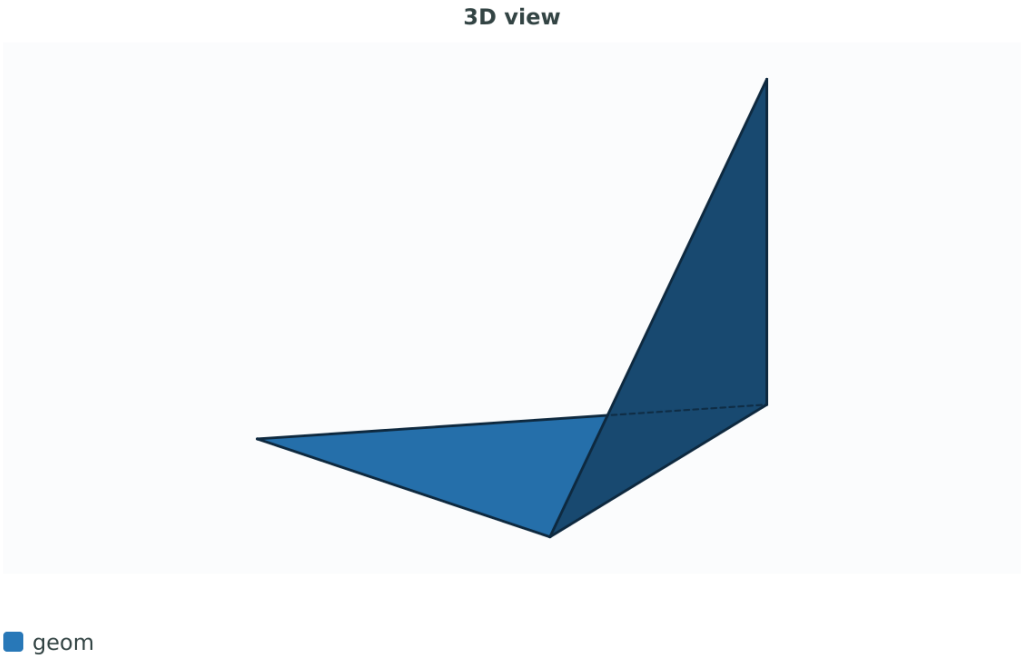
Code

```
SELECT GeometryType(geom) as result
FROM
  (SELECT
     'TIN (((
              0 0 0,
              0 0 1,
              0 1 0,
              0 0 0
          )),((
              0 0 0,
              0 1 0,
              1 1 0,
              0 0 0
          ))
          )'::geometry  AS geom
  ) AS g;
```

Output

```
TIN
```

Figure



Se även

ST_GeometryType

7.4.2 ST_Boundary

ST_Boundary — Returnerar gränsen för en geometri.

Synopsis

geometry **ST_Boundary**(geometry geomA);

Beskrivning

Returnerar stängningen av den kombinatoriska gränsen för denna geometri. Den kombinatoriska gränsen definieras enligt beskrivningen i avsnitt 3.12.3.2 i OGC SPEC. Eftersom resultatet av denna funktion är en slutning, och därmed topologiskt sluten, kan den resulterande gränsen representeras med hjälp av representativa geometriprimitiver som diskuteras i OGC SPEC, avsnitt 3.12.2.

Utförs av GEOS-modulen

Notera
Före 2.0.0 kastar denna funktion ett undantag om den används med `GEOMETRYCOLLECTION`. Från och med 2.0.0 returnerar den istället NULL (indata som inte stöds).

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. OGC SPEC s2.1.1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1.17

Denna funktion stöder 3d och kommer inte att tappa z-index.

Förbättrad: 2.1.0-stöd för Triangle infördes

Ändrat: 3.2.0 stöd för TIN, använder inte geos, lineariserar inte kurvor

Exempel

LineString with boundary points overlaid.

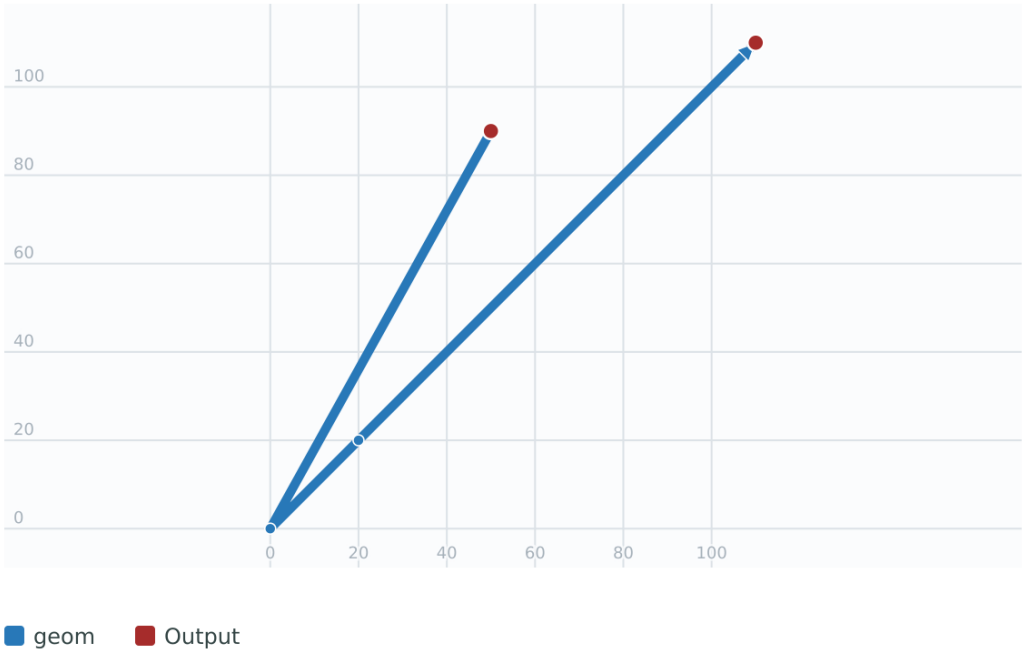
Code

```
SELECT ST_Boundary(geom)
FROM (SELECT 'LINESTRING(100 150,50 60,70 80,160 170)'::geometry As geom) As f;
```

Output

```
MULTIPOINT((100 150),(160 170))
```

Figure



Polygon holes with the boundary MultiLineString overlaid.

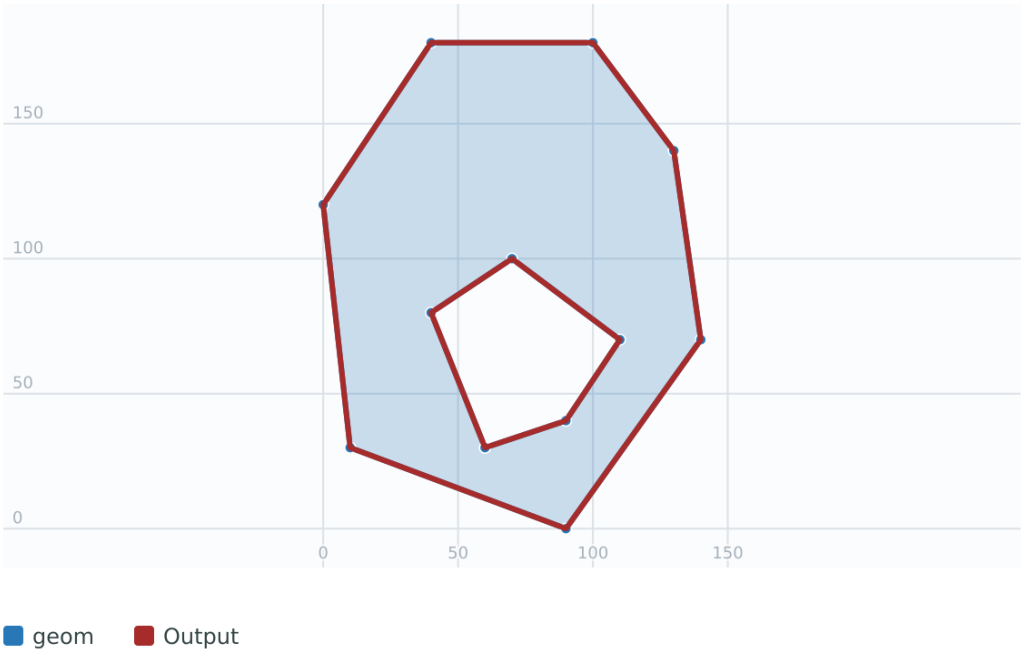
Code

```
SELECT ST_Boundary(geom)
FROM (SELECT
'POLYGON (( 10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130 ),
    ( 70 40,100 50,120 80,80 110,50 90,70 40 ))'::geometry As geom) As f;
```

Output

```
MULTILINESTRING((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130),
       (70 40,100 50,120 80,80 110,50 90,70 40))
```

Figure



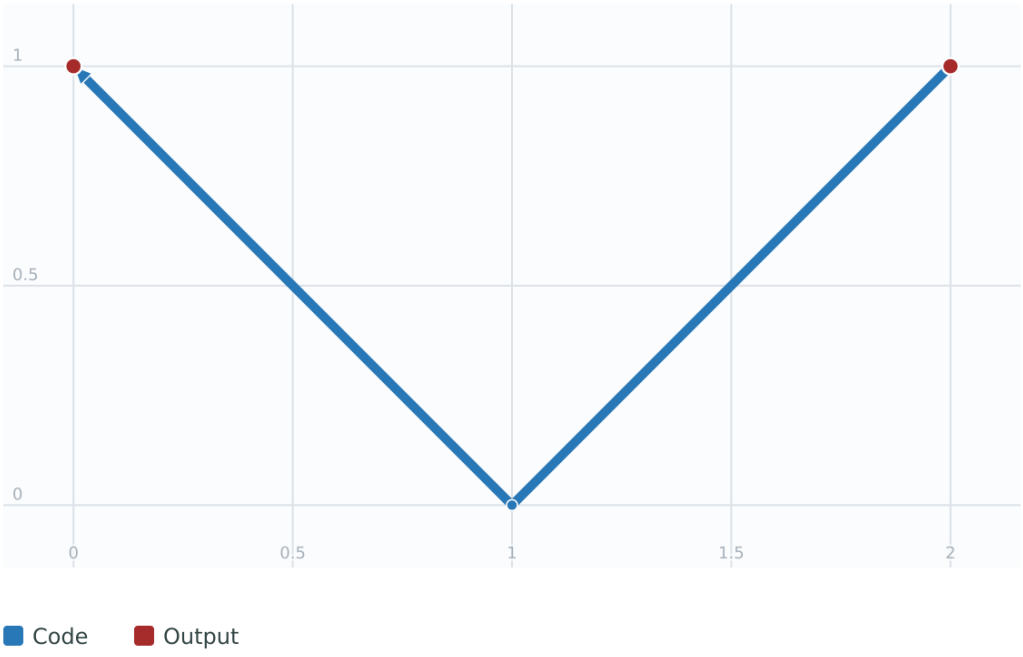
Code

```
SELECT ST_Boundary('LINESTRING(1 1,0 0,-1 1)');
```

Output

```
MULTIPOINT((1 1),(-1 1))
```

Figure



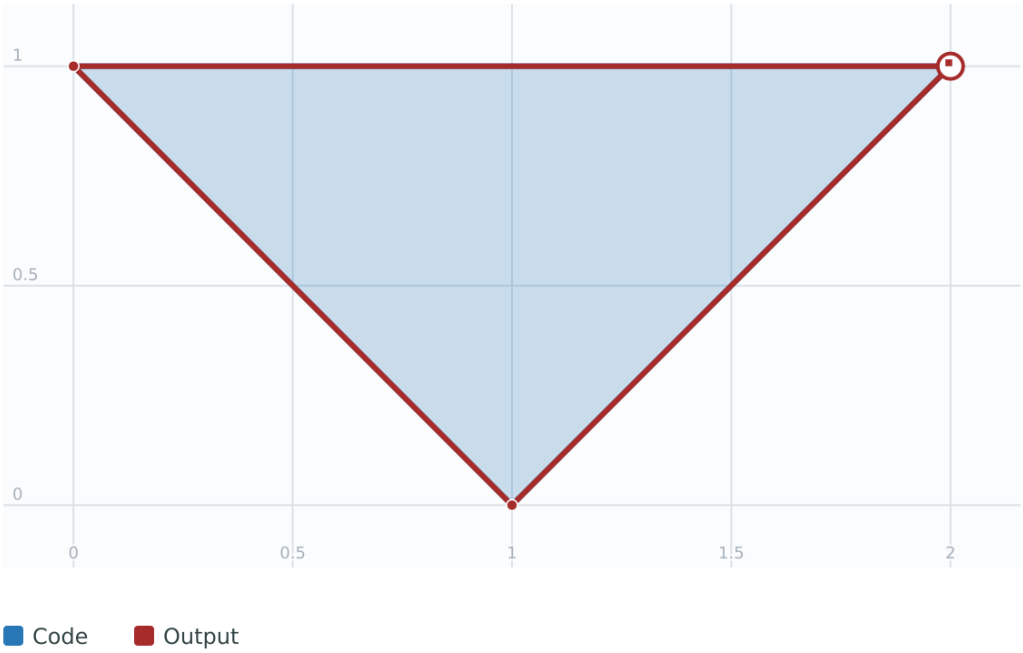
Code

```
SELECT ST_Boundary('POLYGON((1 1,0 0,-1 1,1 1))');
```

Output

```
LINESTRING(1 1,0 0,-1 1,1 1)
```

Figure



This example uses a 3D polygon.

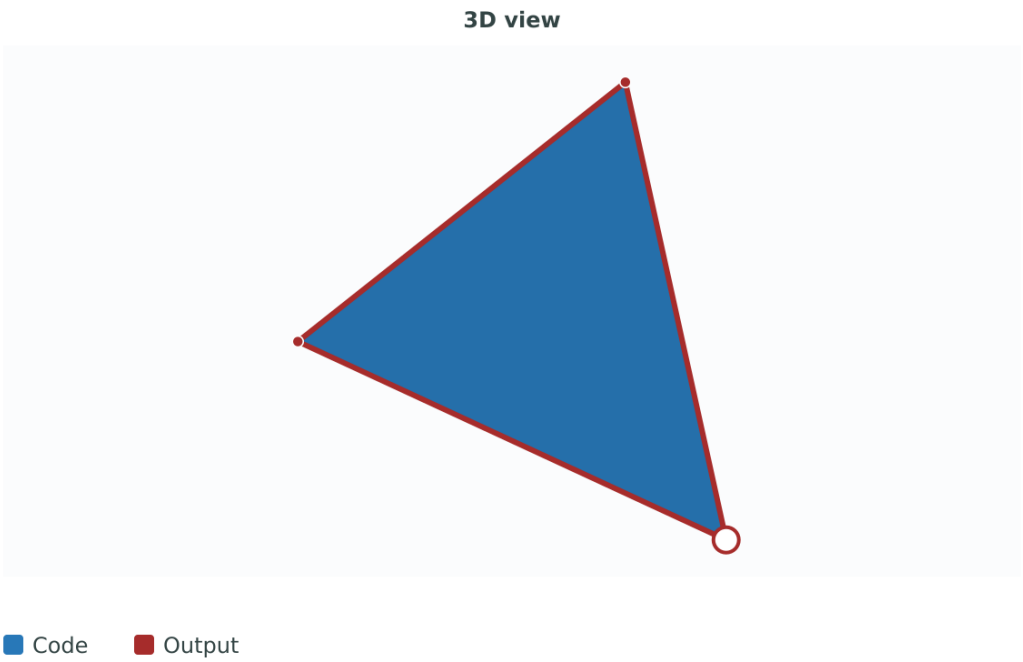
Code

```
SELECT ST_Boundary('POLYGON((1 1 1,0 0 1,-1 1 1,1 1 1))');
```

Output

```
LINESTRING(1 1 1,0 0 1,-1 1 1,1 1 1)
```

Figure



This example uses a 3D MultiLineString.

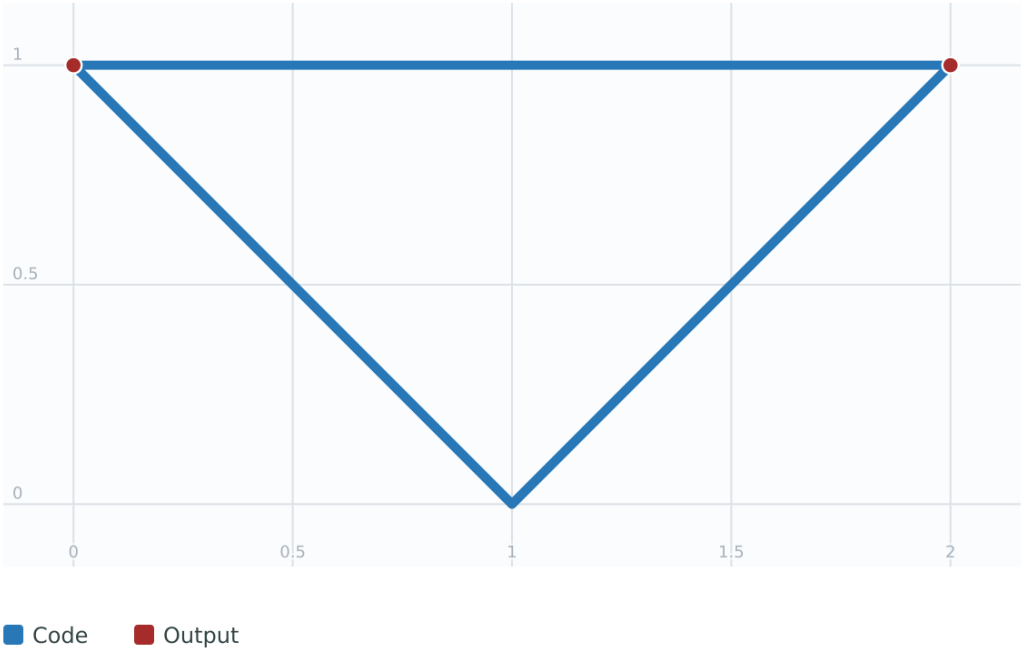
Code

```
SELECT ST_Boundary('MULTILINESTRING((1 1 1,0 0 0.5,-1 1 1),(1 1 0.5,0 0 0.5,-1 1 0.5,1 1 ↩
    0.5) )');
```

Output

```
MULTIPOINT(1 1 1,-1 1 1)
```

Figure



Se även

ST_AsText, ST_ExteriorRing, ST_MakePolygon

7.4.3 ST_BoundingDiagonal

ST_BoundingDiagonal — Returnerar diagonalen i en geometris avgränsande box.

Synopsis

geometry **ST_BoundingDiagonal**(geometry geom, boolean fits=false);

Beskrivning

Returnerar diagonalen i den angivna geometrins avgränsande box som en LineString. Diagonalen är en 2-punkts LineString med minimivärdena för varje dimension i startpunkten och maximivärdena i slutpunkten. Om indatageometrin är tom är diagonalen en LINESTRING EMPTY.

Den valfria parametern `fits` anger om den bästa anpassningen behövs. Om false anges kan diagonalen i en något större avgränsande box accepteras (vilket är snabbare att beräkna för geometrier med många hörn). I båda fallen täcker den returnerade diagonallinjens avgränsningsbox alltid inmatningsgeometrin.

Den returnerade geometrin behåller SRID och dimensionalitet (Z- och M-närvaro) för den inmatade geometrin.

Notera
I degenererade fall (t.ex. en enda vertex i indata) kommer den returnerade linjesträngen att vara formellt ogiltig (inget inre). Resultatet är fortfarande topologiskt giltigt.

Tillgänglighet: 2.2.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Exempel

This example shows a polygon together with the diagonal of its bounding box.

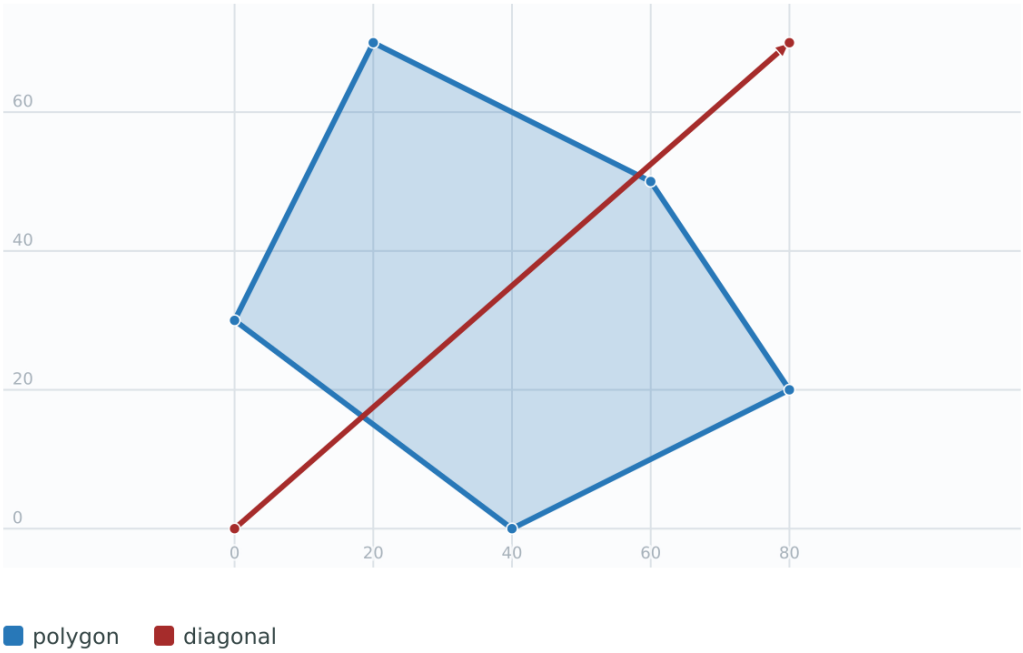
Code

```
WITH input AS (
  SELECT 'POLYGON((10 20,30 60,70 40,90 10,50 -10,10 20))'::geometry AS geom
)
SELECT geom AS input_polygon,
       ST_BoundingDiagonal(geom) AS diagonal
FROM input;
```

Output

```
POLYGON((10 20,30 60,70 40,90 10,50 -10,10 20)) | LINESTRING(10 -10,90 60)
```

Figure



This example returns the minimum X coordinate of a buffer around a point.

Code

```
SELECT ST_X(ST_StartPoint(ST_BoundingDiagonal(
  ST_Buffer(ST_Point(0, 0), 10)
)));
```

Output

```
-10
```

Se även

ST_StartPoint, ST_EndPoint, ST_X, ST_Y, ST_Z, ST_M, ST_Envelope

7.4.4 ST_CoordDim

ST_CoordDim — Returnerar koordinatdimensionen för en geometri.

Synopsis

integer **ST_CoordDim**(geometry geomA);

Beskrivning

Returnerar koordinatdimensionen för ST_Geometry-värdet.

Detta är det MM-kompatibla aliasnamnet för ST_NDims

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.3

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Code

```
SELECT ST_CoordDim('CIRCULARSTRING(1 2 3,1 3 4,5 6 7,8 9 10,11 12 13)');
```

Output

```
3
```

Code

```
SELECT ST_CoordDim(ST_Point(1, 2));
```

Output

```
2
```

Se även

ST_NDims

7.4.5 ST_Dimension

ST_Dimension — Returnerar den topologiska dimensionen för en geometri.

Synopsis

integer **ST_Dimension**(geometry g);

Beskrivning

Returnerar den topologiska dimensionen för detta Geometry-objekt, som måste vara mindre än eller lika med koordinatdimensionen. OGC SPEC s2.1.1.1 - returnerar 0 för `POINT`, 1 för `LINESTRING`, 2 för `POLYGON` och den största dimensionen för komponenterna i en `GEOMETRYCOLLECTION`. Om dimensionen är okänd (t.ex. för en tom `GEOMETRYCOLLECTION`) returneras 0.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.2

Notera
A `TIN`, including a closed TIN surface, is a surface geometry and has dimension 2. Use CG_MakeSolid to make the solid representation explicit; the resulting 3D solid has dimension 3. Solid geometry semantics, such as volume, are handled separately by the SQL/MM 3D solid functions (SQL-MM IEC 13249-3: 9.1).

Förbättrad: 2.0.0 stöd för polyhedrala ytor och TINs infördes. Kastar inte längre ett undantag om en tom geometri ges.

Notera
Före 2.0.0 kastar denna funktion ett undantag om den används med tom geometri.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

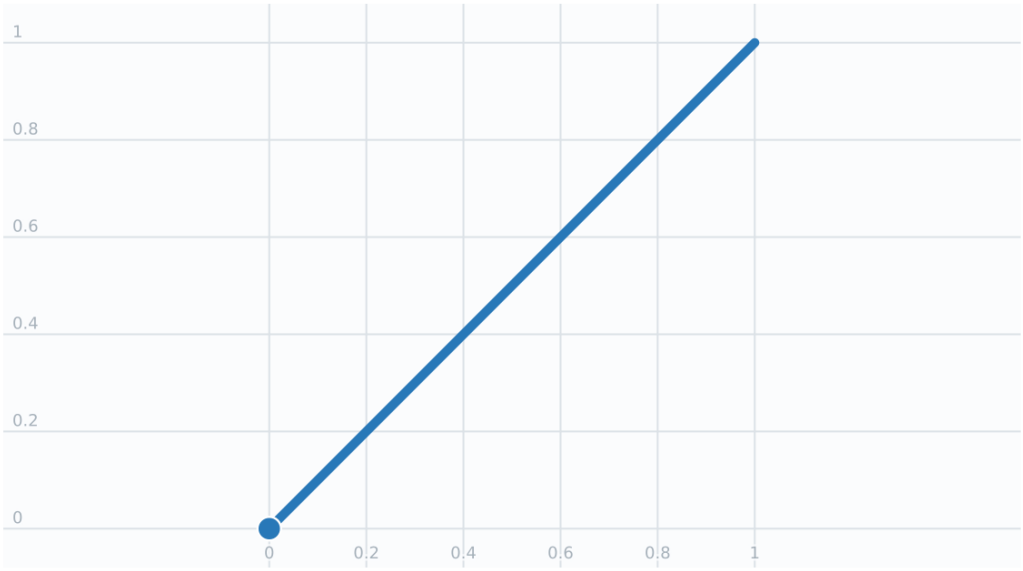
Code

```
SELECT ST_Dimension('GEOMETRYCOLLECTION(LINESTRING(1 1,0 0), POINT(0 0))');
```

Output

```
1
```

Figure



Code

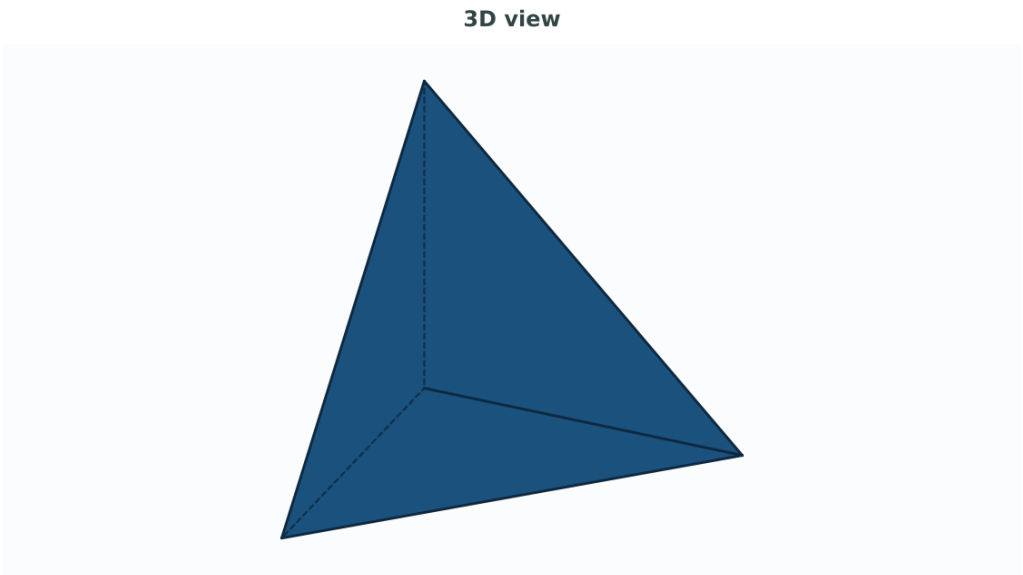
Code

```
SELECT ST_Dimension('TIN Z (((0 0 0,0 0 1,0 1 0,0 0 0)),((0 0 0,0 1 0,1 0 0,0 0 0)),((0 0 ↩
    0,1 0 0,0 0 1,0 0 0)),((1 0 0,0 1 0,0 0 1,1 0 0)))');
```

Output

```
2
```

Figure



Code

Se även

ST_NDims, CG_MakeSolid, CG_Volume

7.4.6 ST_Dump

ST_Dump — Returnerar en uppsättning `geometry_dump-rader` för komponenterna i en geometri.

Synopsis

geometry_dump[] **ST_Dump**(geometry g1);

Beskrivning

En set-returning function (SRF) som extraherar komponenterna i en geometri. Den returnerar en uppsättning geometry_dump -rader som var och en innehåller en geometri (fältet*geom)* och en array med heltal (fältet *path*).

För en atomär geometrityp (POINT,LINESTRING,POLYGON) returneras en enda post med en tom *sökvägsmatris* och indatageometrin som *geom*. För en samling eller multipelgeometri returneras en post för var och en av samlingens komponenter, och *sökvägen* anger komponentens position i samlingen.

ST_Dump är användbart för att expandera geometrier. Det är motsatsen till ST_Collect / GROUP BY, eftersom det skapar nya rader. Det kan t.ex. användas för att expandera MULTIPOLYGONER till POLYGONER.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Tillgänglighet: PostGIS 1.0.0RC1. Kräver PostgreSQL 7.3 eller högre.

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Dump the components of geometries stored in a table.

Code

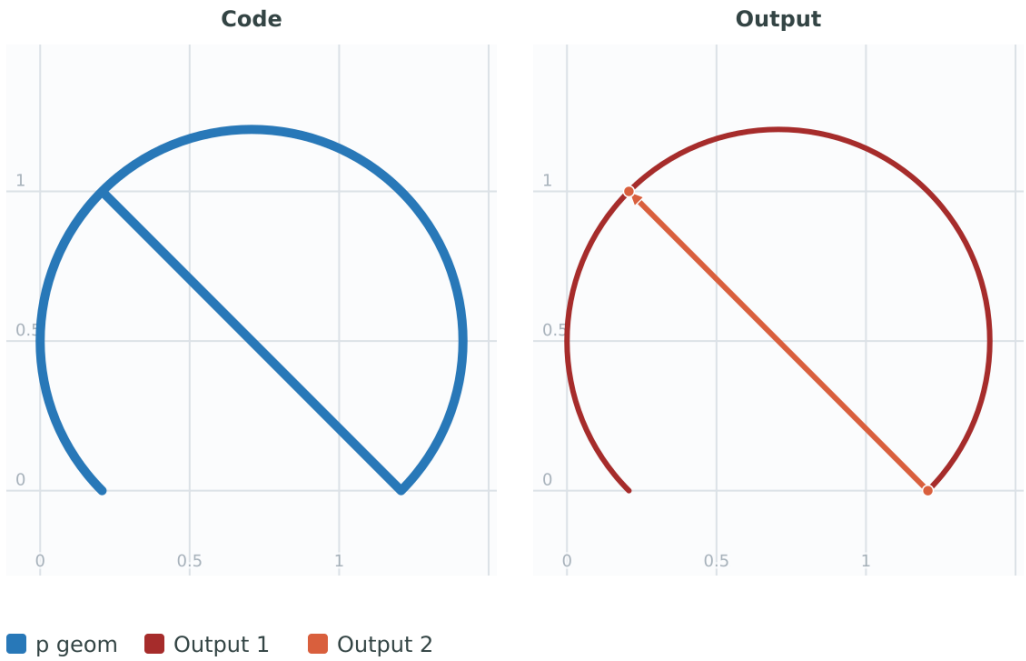
```
SELECT sometable.field1,
       (ST_Dump(sometable.geom)).geom AS geom
FROM sometable;
```

Break a CompoundCurve into its constituent LineStrings and CircularStrings.

Code

```
SELECT a.geom, ST_HasArc(a.geom)
  FROM ( SELECT (ST_Dump(p_geom)).geom AS geom
         FROM (SELECT 'COMPOUNDCURVE(CIRCULARSTRING(0 0,1 1,1 0),(1 0,0 1))'::geometry AS
    p_geom) AS b
        ) AS a;
```

Figure

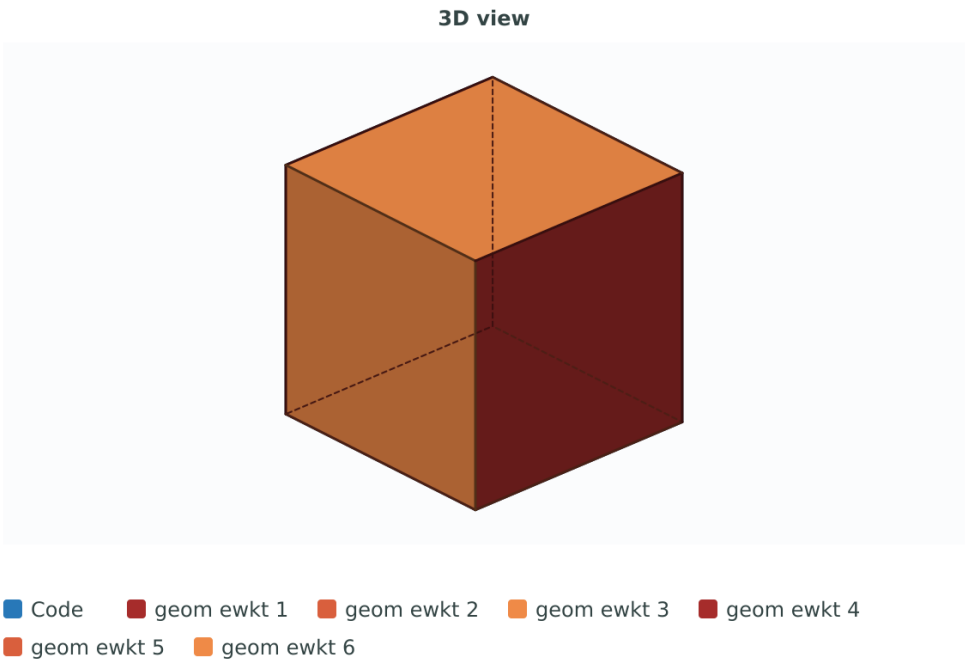


This example breaks a PolyhedralSurface into its faces.

Code

```
SELECT (a.p_geom).path[1] As path,(a.p_geom).geom As geom_ewkt
  FROM (SELECT ST_Dump('POLYHEDRALSURFACE(
((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),((1 1 0,1 1 1,1 0 1,1 0
    0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
)' ) AS p_geom )  AS a;
```

Figure

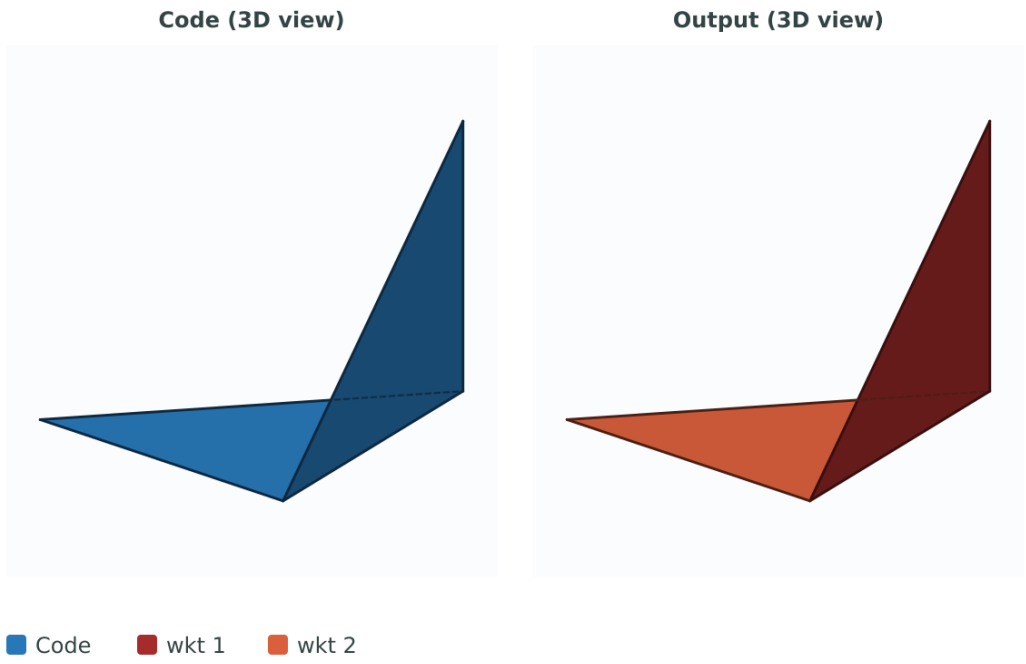


This example uses a TIN geometry.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_Dump('TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
            )),((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
            ))
            )' ) AS gdump
    ) AS g;
```

Figure



Se även

geometry_dump, Avsnitt 13.6, ST_Collect, ST_GeometryN

7.4.7 ST_DumpPoints

ST_DumpPoints — Returnerar en uppsättning `geometry_dump-rader` för koordinaterna i en geometri.

Synopsis

geometry_dump[] **ST_DumpPoints**(geometry geom);

Beskrivning

En set-returning function (SRF) som extraherar koordinaterna (hörnen) för en geometri. Den returnerar en uppsättning geometry_dump -rader, som var och en innehåller en geometri (fältet*geom)* och en array av heltal (fältet *path*).

- *geom-fältet* `POINT`srepresenterar koordinaterna för den medföljande geometrin.
- *path-fältet* (ett `integer[]`) är ett index som räknar upp koordinatpositionerna i elementen i den angivna geometrin. Indexen är 1-baserade. Till exempel för en `LINESTRING` är banorna `{i}` där `i` är den `n:te` koordinaten i `LINESTRING`. För en `POLYGON` är banorna `{i,j}` där `i` är ringnumret (1 är den yttre ringen; de inre ringarna följer) och `j` är koordinatpositionen i ringen.

För att få en enda geometri som innehåller koordinaterna, använd ST_Points.

Förbättrad: 2.1.0 Snabbare hastighet. Återimplementerad som native-C.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Tillgänglighet: 1.5.0

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

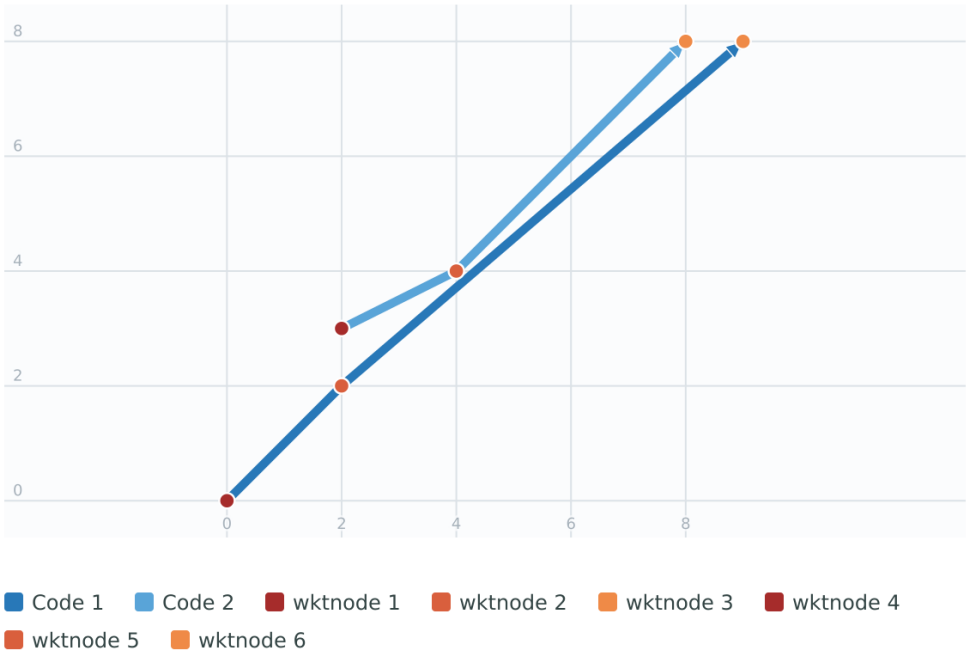
Exempel

Explode a table of LineStrings into their vertices.

Code

```
SELECT edge_id, (dp).path[1] As index,(dp).geom As wktnode
FROM (SELECT 1 As edge_id,
    ST_DumpPoints(ST_GeomFromText('LINESTRING(1 2,3 4,10 10)')) AS dp
     UNION ALL
     SELECT 2 As edge_id,
    ST_DumpPoints(ST_GeomFromText('LINESTRING(3 5,5 6,9 10)')) AS dp
   ) As foo;
```

Figure



Dump every vertex from a mixed GeometryCollection.

Code

```
SELECT path,geom
FROM (
  SELECT (ST_DumpPoints(g.geom)).*
  FROM
```

```
    (SELECT
       'GEOMETRYCOLLECTION(
          POINT ( 0 1 ),
          LINESTRING ( 0 3,3 4 ),
          POLYGON (( 2 0,2 3,0 2,2 0 )),
          POLYGON (( 3 0,3 3,6 3,6 0,3 0 ),
                   ( 5 1,4 2,5 2,5 1 )),
          MULTIPOLYGON (
                  (( 0 5,0 8,4 8,4 5,0 5 ),
                   ( 1 6,3 6,2 7,1 6 )),
                  (( 5 4,5 8,6 7,5 4 ))
          )
        )'::geometry AS geom
    ) AS g
  ) j;
```

Output

```
path    | st_astext
----------+------------
 {1,1}     | POINT(0 1)
 {2,1}     | POINT(0 3)
 {2,2}     | POINT(3 4)
 {3,1,1}   | POINT(2 0)
 {3,1,2}   | POINT(2 3)
 {3,1,3}   | POINT(0 2)
 {3,1,4}   | POINT(2 0)
 {4,1,1}   | POINT(3 0)
 {4,1,2}   | POINT(3 3)
 {4,1,3}   | POINT(6 3)
 {4,1,4}   | POINT(6 0)
 {4,1,5}   | POINT(3 0)
 {4,2,1}   | POINT(5 1)
 {4,2,2}   | POINT(4 2)
 {4,2,3}   | POINT(5 2)
 {4,2,4}   | POINT(5 1)
 {5,1,1,1} | POINT(0 5)
 {5,1,1,2} | POINT(0 8)
 {5,1,1,3} | POINT(4 8)
 {5,1,1,4} | POINT(4 5)
 {5,1,1,5} | POINT(0 5)
 {5,1,2,1} | POINT(1 6)
 {5,1,2,2} | POINT(3 6)
 {5,1,2,3} | POINT(2 7)
 {5,1,2,4} | POINT(1 6)
 {5,2,1,1} | POINT(5 4)
 {5,2,1,2} | POINT(5 8)
 {5,2,1,3} | POINT(6 7)
 {5,2,1,4} | POINT(5 4)
(29 rows)
```

Collect the dumped vertices to visualize them over the source geometry.

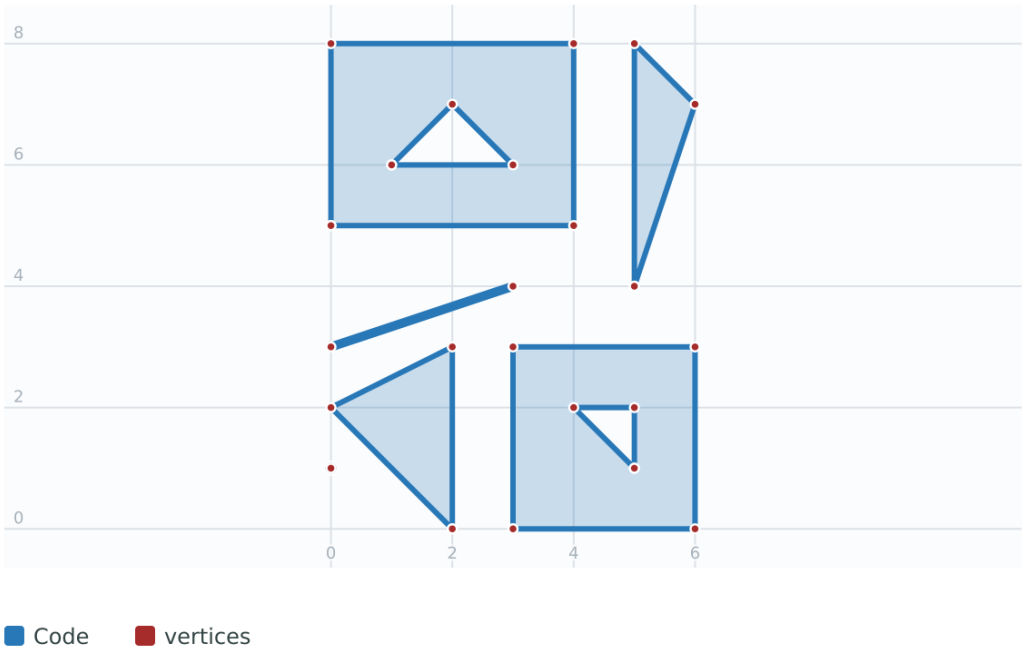
Code

```
WITH source(geom) AS (VALUES (
  'GEOMETRYCOLLECTION(
     POINT (0 1),
     LINESTRING (0 3,3 4),
     POLYGON ((2 0,2 3,0 2,2 0)),
     POLYGON ((3 0,3 3,6 3,6 0,3 0),(5 1,4 2,5 2,5 1)),
     MULTIPOLYGON (((0 5,0 8,4 8,4 5,0 5),(1 6,3 6,2 7,1 6)),
                   ((5 4,5 8,6 7,5 4))))'::geometry
))
SELECT ST_Collect((dp).geom ORDER BY (dp).path) AS vertices
FROM source
CROSS JOIN LATERAL ST_DumpPoints(source.geom) AS dp;
```

Output

```
MULTIPOINT((0 1),(0 3),(3 4),(2 0),(2 3),(0 2),(2 0),(3 0),(3 3),(6 3),(6 0),(3 0),(5 1),(4 2),(5 2),(5 1),(0 5),(0 8) ↩
    ,(4 8),(4 5),(0 5),(1 6),(3 6),(2 7),(1 6),(5 4),(5 8),(6 7),(5 4))
```

Figure



This example uses a cube represented as a PolyhedralSurface.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_DumpPoints('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )' ) AS gdump
    ) AS g;
```

Output

```
path   |     wkt
---------+--------------
 {1,1,1} | POINT(0 0 0)
 {1,1,2} | POINT(0 0 1)
 {1,1,3} | POINT(0 1 1)
 {1,1,4} | POINT(0 1 0)
 {1,1,5} | POINT(0 0 0)
 {2,1,1} | POINT(0 0 0)
 {2,1,2} | POINT(0 1 0)
 {2,1,3} | POINT(1 1 0)
 {2,1,4} | POINT(1 0 0)
 {2,1,5} | POINT(0 0 0)
 {3,1,1} | POINT(0 0 0)
 {3,1,2} | POINT(1 0 0)
 {3,1,3} | POINT(1 0 1)
 {3,1,4} | POINT(0 0 1)
 {3,1,5} | POINT(0 0 0)
 {4,1,1} | POINT(1 1 0)
 {4,1,2} | POINT(1 1 1)
 {4,1,3} | POINT(1 0 1)
 {4,1,4} | POINT(1 0 0)
 {4,1,5} | POINT(1 1 0)
 {5,1,1} | POINT(0 1 0)
```

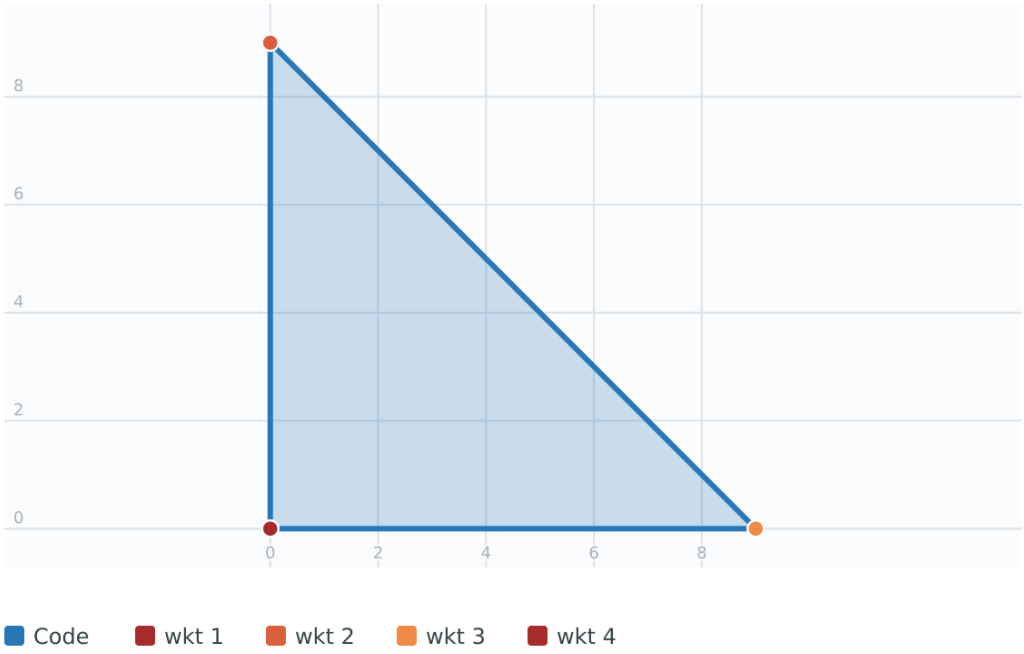
```
 {5,1,2} | POINT(0 1 1)
 {5,1,3} | POINT(1 1 1)
 {5,1,4} | POINT(1 1 0)
 {5,1,5} | POINT(0 1 0)
 {6,1,1} | POINT(0 0 1)
 {6,1,2} | POINT(1 0 1)
 {6,1,3} | POINT(1 1 1)
 {6,1,4} | POINT(0 1 1)
 {6,1,5} | POINT(0 0 1)
(30 rows)
```

This example uses a Triangle geometry.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_DumpPoints('TRIANGLE ((
                0 0,
                0 9,
                9 0,
                0 0
            ))' ) AS gdump
    ) AS g;
```

Figure



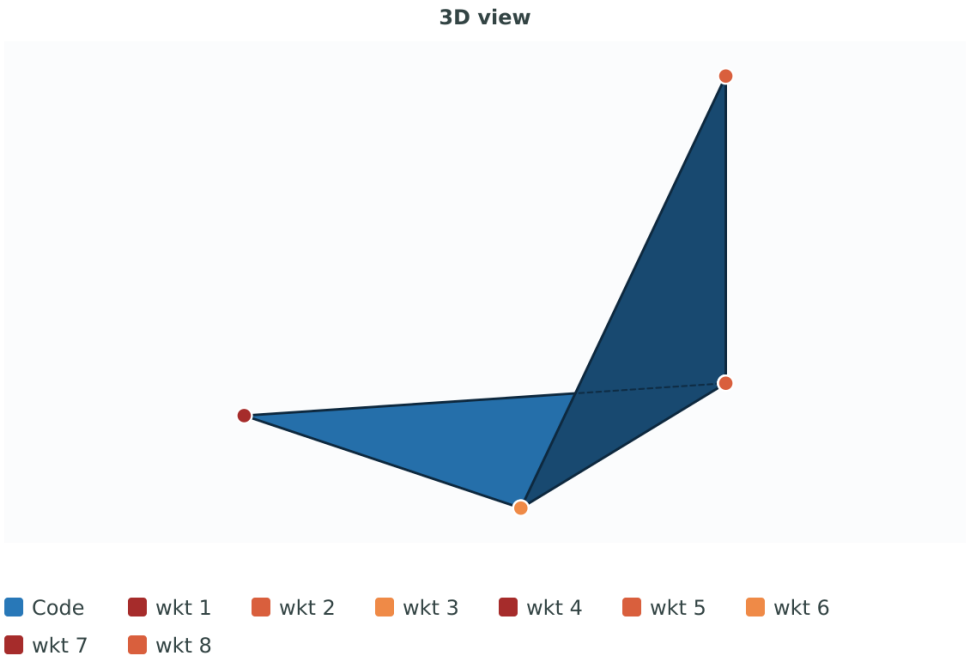
This example uses a TIN geometry.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_DumpPoints('TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
```

```
            )),((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
            ))
            )' ) AS gdump
    ) AS g;
```

Figure



Se även

geometry_dump, Avsnitt 13.6, ST_Dump, ST_DumpRings, ST_Points

7.4.8 ST_DumpSegments

ST_DumpSegments — Returnerar en uppsättning `geometry_dump-rader` för segmenten i en geometri.

Synopsis

geometry_dump[] **ST_DumpSegments**(geometry geom);

Beskrivning

En set-returning function (SRF) som extraherar segmenten i en geometri. Den returnerar en uppsättning geometry_dump rader, var och en innehållande en geometri (fältet*geom)* och en array av heltal (fältet *path*).

- *`geom-fältet`* `LINESTRING`srepresenterar de linjära segmenten i den medföljande geometrin, medan `CIRCULARSTRING`srepresenterar bågsegmenten.

- *path*-fältet (ett `integer[]`) är ett index som räknar upp segmentets startpunktspositioner i elementen i den angivna geometrin. Indexen är 1-baserade. Till exempel för en `LINESTRING` är banorna `{i}` där `i` är den `n:te` startpunkten för segmentet i `LINESTRING`. För en `POLYGON` är banorna `{i,j}` där `i` är ringnumret (1 är yttre; inre ringar följer) och `j` är segmentets startpunktsposition i ringen.

Tillgänglighet: 3.2.0

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

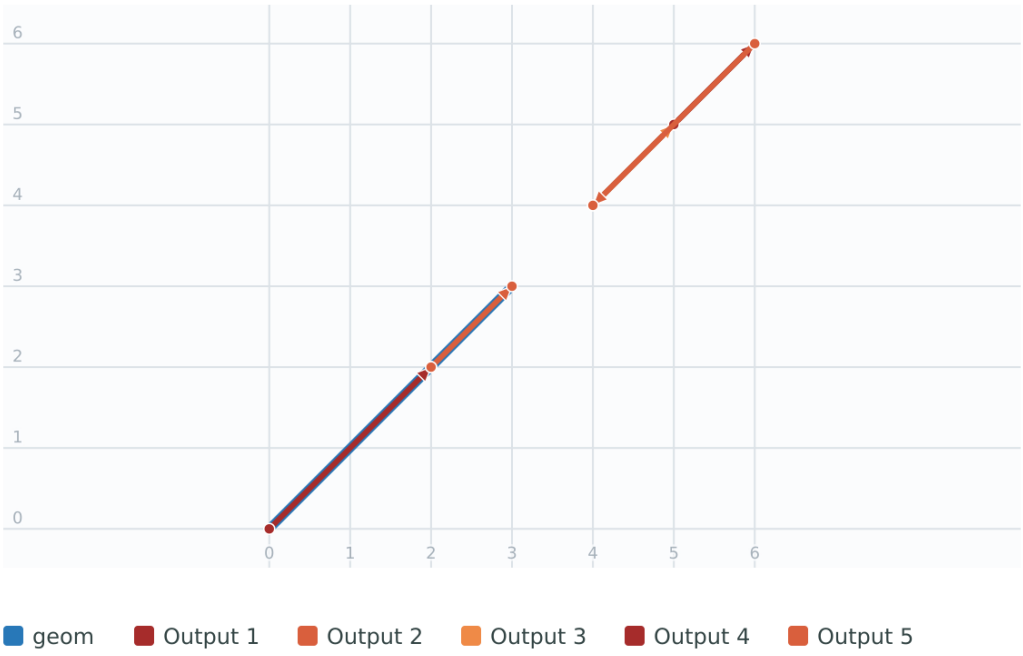
Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel på standardgeometri

Code

```
SELECT path,geom
FROM (
    SELECT (ST_DumpSegments(g.geom)).*
    FROM (SELECT 'GEOMETRYCOLLECTION(
    LINESTRING(1 1,3 3,4 4),
    POLYGON((5 5,6 6,7 7,5 5))
)'::geometry AS geom
        ) AS g
) j;
```

Figure



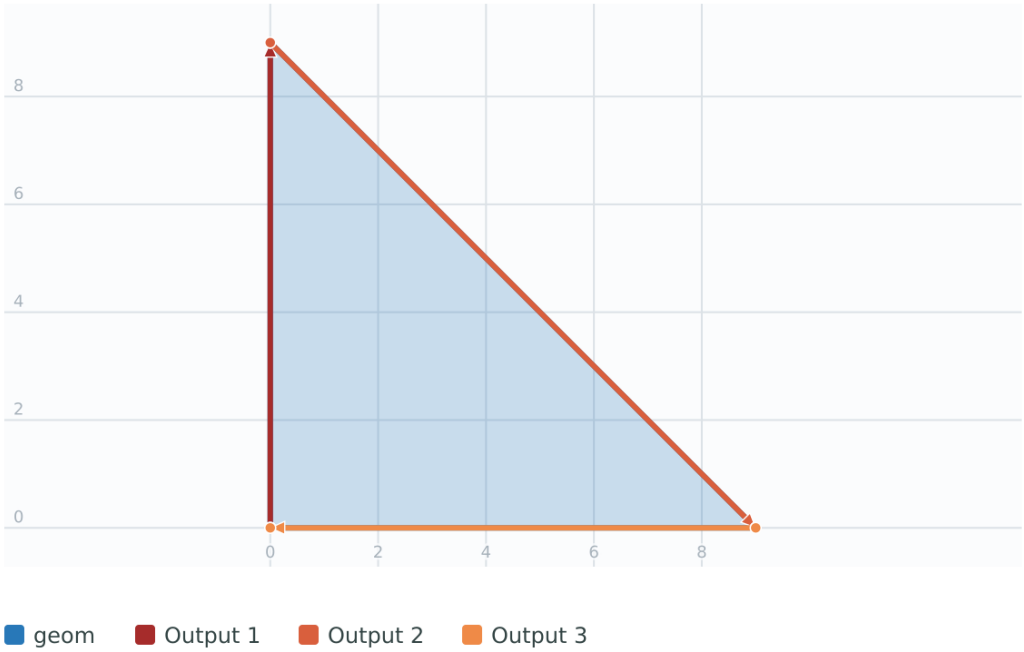
TIN- och triangelexempel

This example uses a Triangle geometry.

Code

```
SELECT path,geom
FROM (
    SELECT (ST_DumpSegments(g.geom)).*
    FROM (SELECT 'TRIANGLE((
        0 0,
        0 9,
        9 0,
        0 0
    ))'::geometry AS geom
        ) AS g
) j;
```

Figure

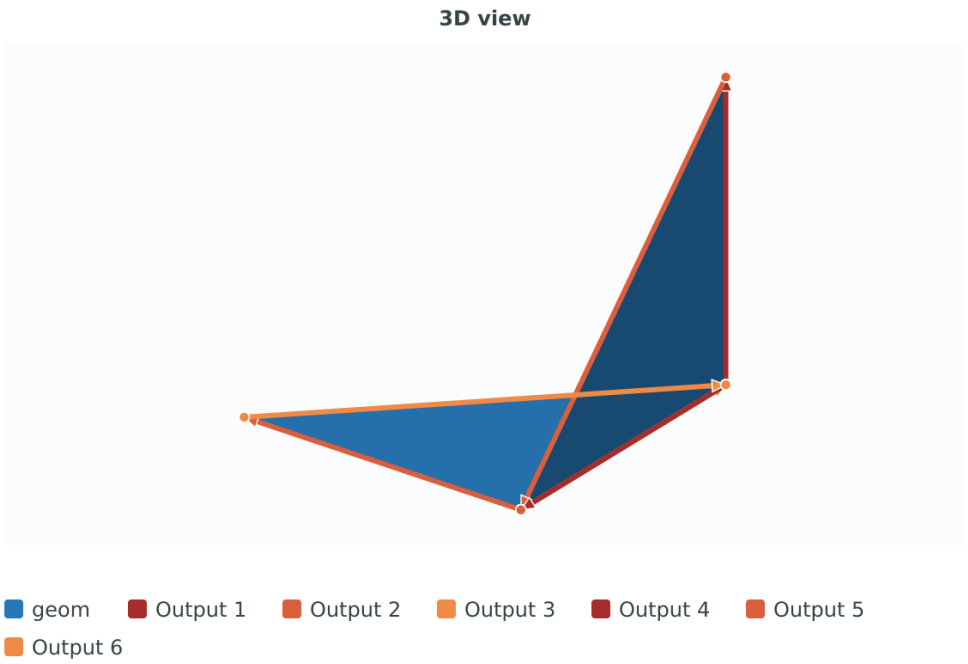


This example uses a TIN geometry.

Code

```
SELECT path,geom
FROM (
    SELECT (ST_DumpSegments(g.geom)).*
    FROM (SELECT 'TIN(((
        0 0 0,
        0 0 1,
        0 1 0,
        0 0 0
    )),((
        0 0 0,
        0 1 0,
        1 1 0,
        0 0 0
    ))
    )'::geometry AS geom
        ) AS g
) j;
```

Figure



Se även

geometry_dump, Avsnitt 13.6, ST_Dump, ST_DumpRings

7.4.9 ST_DumpRings

ST_DumpRings — Returnerar en uppsättning `geometry_dump-rader` för de yttre och inre ringarna i en polygon.

Synopsis

geometry_dump[] **ST_DumpRings**(geometry a_polygon);

Beskrivning

En set-returning function (SRF) som extraherar ringarna i en polygon. Den returnerar en uppsättning geometry_dump rader, var och en innehållande en geometri*`(geom-fält)`* och en array av heltal*`(path-fält)`*.

Fältet *`geom`* innehåller varje ring som en POLYGON. Fältet *`path`* är en heltalsarray med längden 1 som innehåller polygonringens index. Den yttre ringen (skalet) har index 0. De inre ringarna (hålen) har index 1 och högre.

Notera
This only works for POLYGON geometries, not MULTIPOLYGONs. Use ST_Dump to extract polygon elements from polygonal geometries: `ST_DumpRings((ST_Dump(geom)).geom )`

Tillgänglighet: PostGIS 1.1.3. Kräver PostgreSQL 7.3 eller högre.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Extracting all rings as polygons.

Code

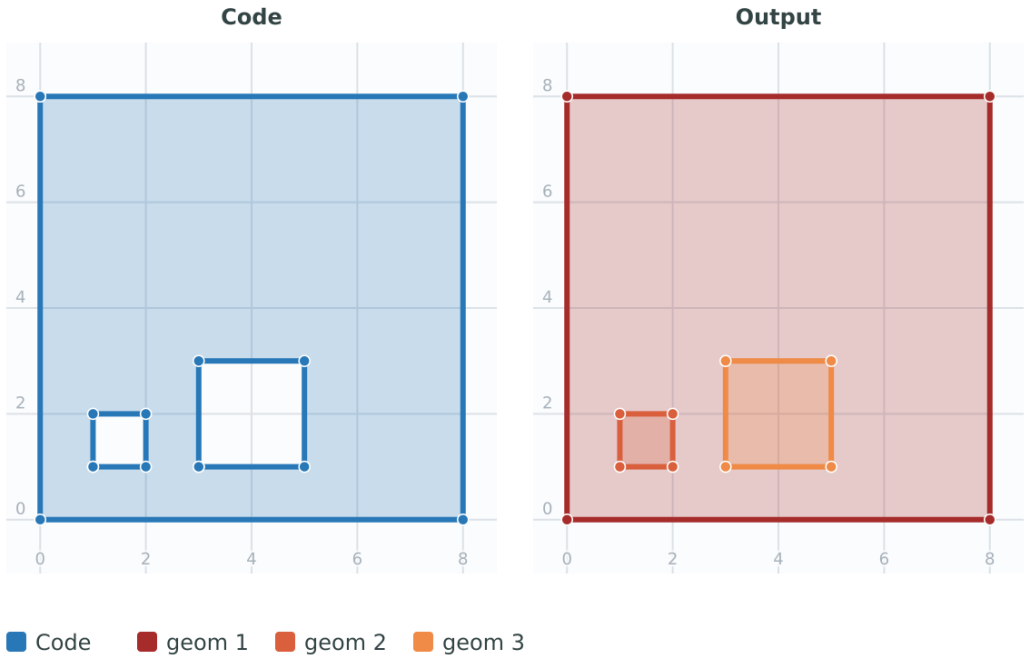
```
SELECT polyTable.field1, polyTable.field1,
      (ST_DumpRings(polyTable.geom)).geom As geom
FROM polyTable
```

Extracting shell and holes from a polygon.

Code

```
SELECT path,geom As geom
FROM ST_DumpRings('POLYGON ((1 9,9 9,9 1,1 1,1 9),(2 2,2 3,3 3,3 2,2 2),(4 2,4 4,6 4,6 2,4 ↩
    2))');
```

Figure



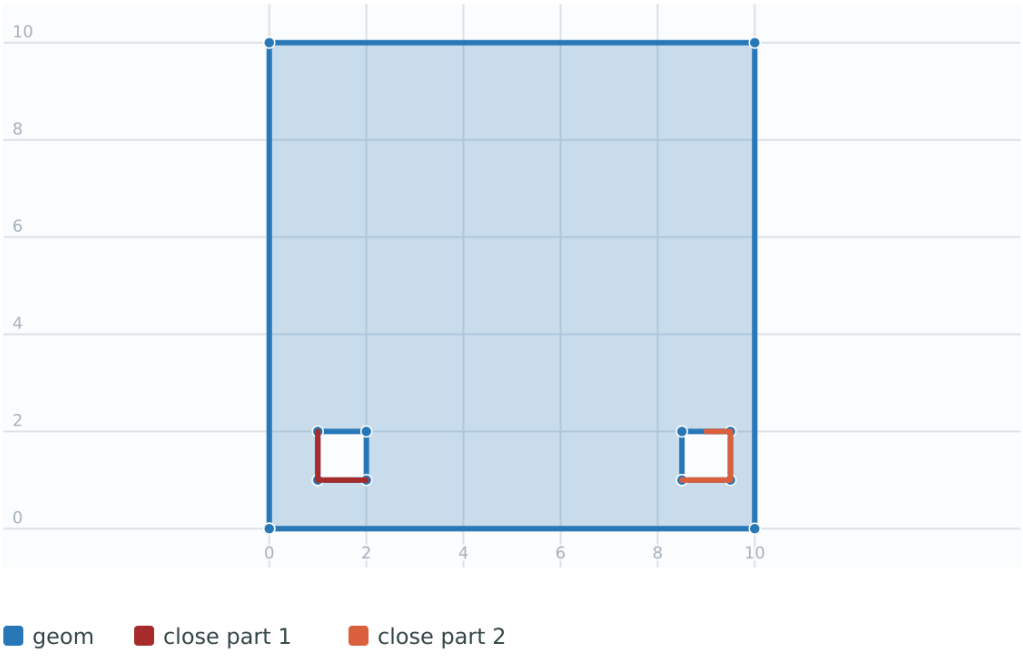
Finding interior rings which lie close to the exterior ring.

Code

```
WITH poly AS (
  SELECT 'POLYGON ((
      0 0, 10 0, 10 10, 0 10, 0 0
    ), (
      1 1, 1 2, 2 2, 2 1, 1 1
    ), (
      8.5 1, 8.5 2, 9.5 2, 9.5 1, 8.5 1
    ))'::geometry AS geom
),
```

```
rings AS (
  SELECT d.path[1] AS ring_no,
         ST_Boundary(d.geom) AS ring_geom
  FROM poly
  CROSS JOIN LATERAL ST_DumpRings(poly.geom) AS d
),
shell AS (
  SELECT ring_geom
  FROM rings
  WHERE ring_no = 0
),
holes AS (
  SELECT ring_no, ring_geom
  FROM rings
  WHERE ring_no > 0
)
SELECT h.ring_no,
       ST_Normalize(ST_Multi(ST_LineMerge(
         ST_Intersection(ST_Buffer(s.ring_geom, 1.0), h.ring_geom)
       ))) AS close_part
FROM shell AS s
JOIN holes AS h
  ON ST_DWithin(s.ring_geom, h.ring_geom, 1.0);
```

Figure



Se även

geometry_dump, Avsnitt 13.6, ST_Boundary, ST_Buffer, ST_Dump, ST_DWithin, ST_ExteriorRing, ST_Interio
ST_Intersection

7.4.10 ST_EndPoint

ST_EndPoint — Returns the last point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_EndPoint**(geometry g);

Beskrivning

Returns the last point of a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry as a `POINT`. Returns `NULL` if the input is not a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE`.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.1.4

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Ändrad: 2.0.0 fungerar inte längre med MultiLineStrings med en geometri. I äldre versioner av PostGIS skulle en MultiLineString med en enda rad fungera med den här funktionen och returnera slutpunkten. I 2.0.0 returnerar den NULL som vilken annan MultiLineString som helst. Det gamla beteendet var en odokumenterad funktion, men personer som antog att de hade sina data lagrade som LINESTRING kan uppleva att dessa returnerar NULL i 2.0.0.

Exempel

Slutpunkt för en LineString

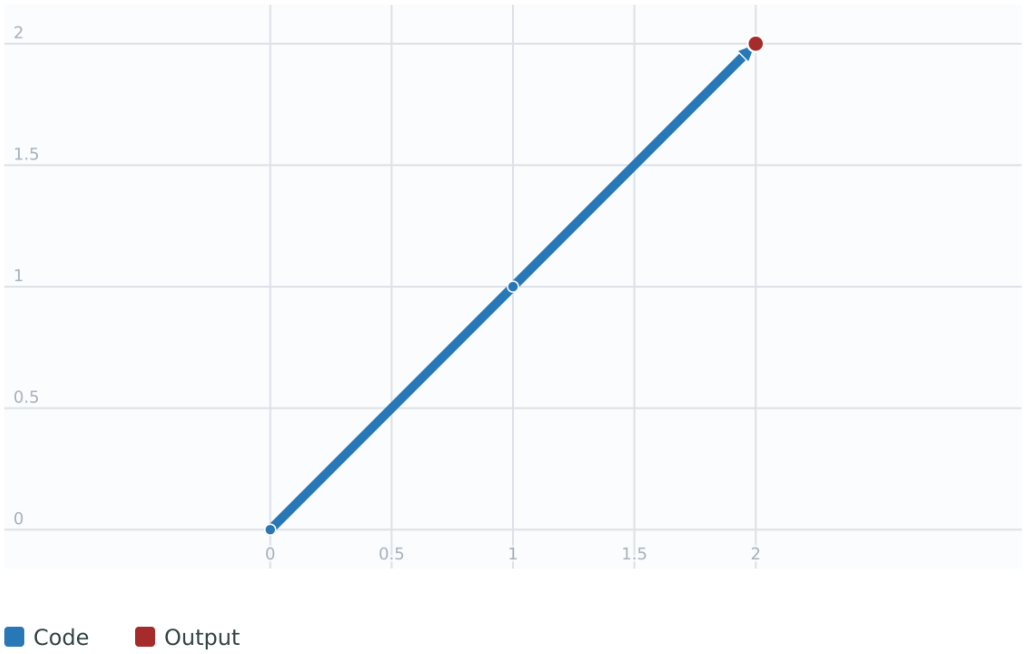
Code

```
SELECT ST_EndPoint('LINESTRING(1 1,2 2,3 3)'::geometry);
```

Output

```
POINT(3 3)
```

Figure



Slutpunkten för en icke-LineString är NULL

Code

```
SELECT ST_EndPoint('POINT(1 1)'::geometry) IS NULL AS is_null;
```

Output

```
t
```

Slutpunkt för en 3D-linjeString

3D endpoint.

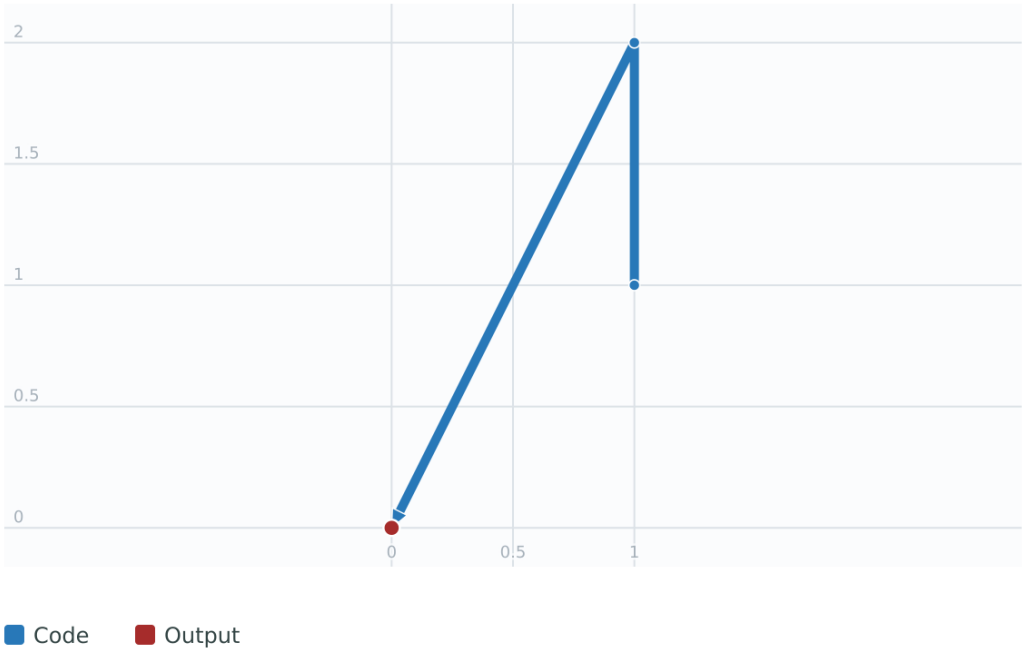
Code

```
SELECT ST_EndPoint('LINESTRING(1 1 2,1 2 3,0 0 5)');
```

Output

```
POINT(0 0 5)
```

Figure



Slutpunkt för en CircularString

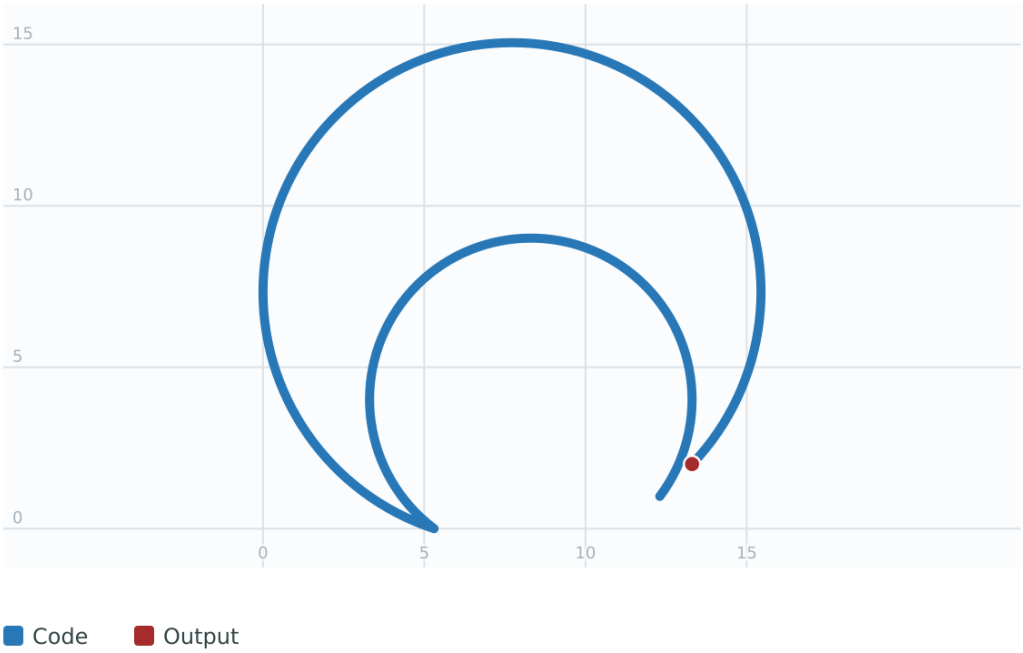
Code

```
SELECT ST_EndPoint('CIRCULARSTRING(5 2,-3 2,-2 1,-4 2,6 3)'::geometry);
```

Output

```
POINT(6 3)
```

Figure



End point of a NURBSCurve

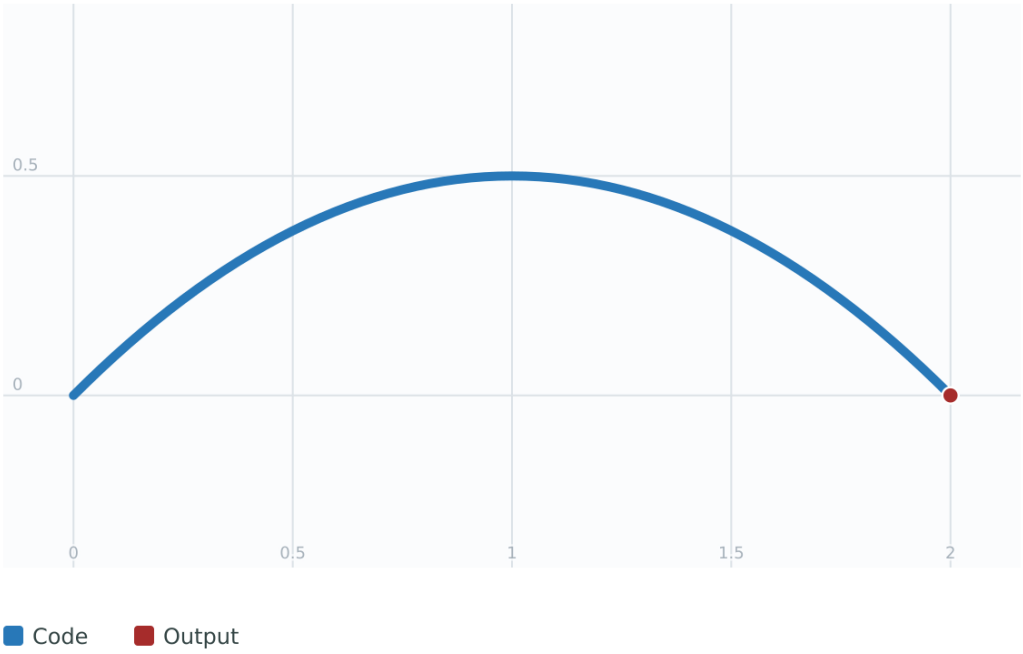
Code

```
SELECT ST_EndPoint('NURBSCURVE(2, (0 0, 1 1, 2 0))'::geometry);
```

Output

```
POINT(2 0)
```

Figure



Se även

ST_PointN, ST_StartPoint

7.4.11 ST_Envelope

ST_Envelope — Returnerar en geometri som representerar en geometris avgränsande box.

Synopsis

geometry **ST_Envelope**(geometry g1);

Beskrivning

Returnerar den minsta avgränsningsboxen med dubbel precision (float8) för den angivna geometrin, som en geometri. Polygonen definieras av hörnpunkterna i begränsningsboxen (((MINX, MINY), (MINX, MAXY),(MAXX, MAXY),(MAXX, MINY), (MINX, MINY)). (PostGIS kommer även att lägga till en ZMIN/ZMAX-koordinat).

Degenererade fall (vertikala linjer, punkter) kommer att returnera en geometri med lägre dimension än POLYGON, dvs. POINT eller LINESTRING.

Tillgänglighet: 1.5.0 ändrat beteende för att mata ut dubbel precision istället för float4

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.19

Exempel

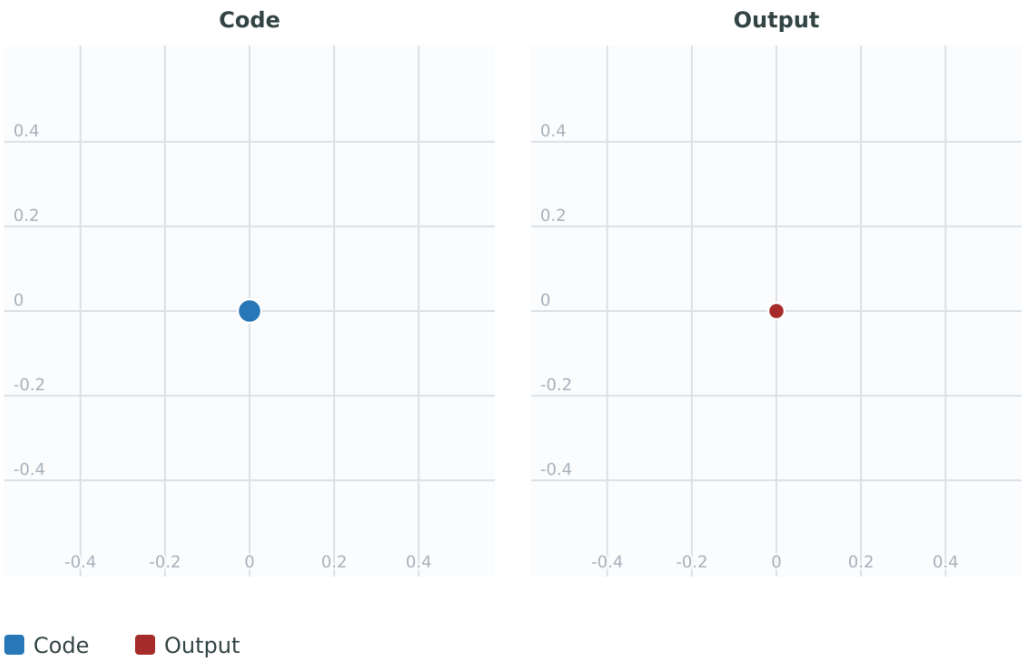
Code

```
SELECT ST_Envelope('POINT(1 3)'::geometry);
```

Output

```
POINT(1 3)
```

Figure



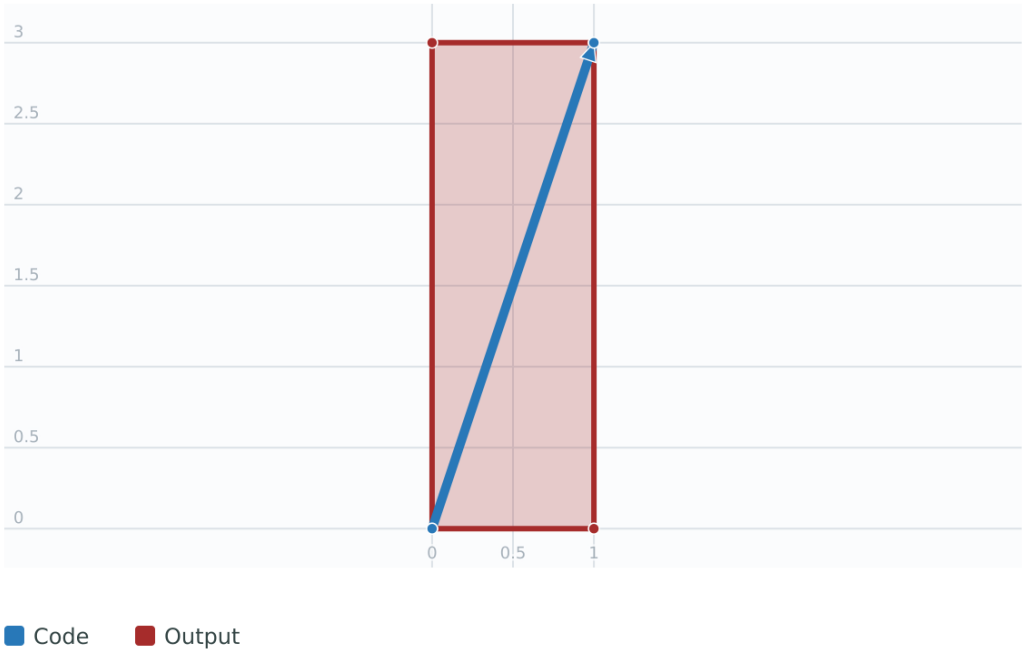
Code

```
SELECT ST_Envelope('LINESTRING(0 0,1 3)'::geometry);
```

Output

```
POLYGON((0 0,0 3,1 3,1 0,0 0))
```

Figure



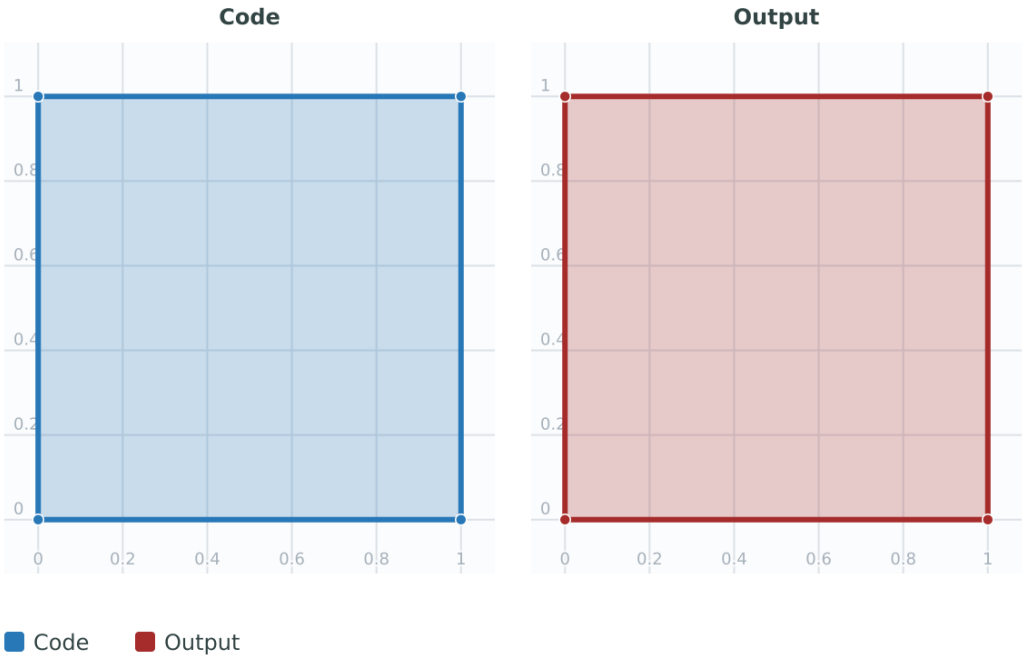
Code

```
SELECT ST_Envelope('POLYGON((0 0,0 1,1.0000001 1,1.0000001 0,0 0))'::geometry);
```

Output

```
POLYGON((0 0,0 1,1.0000001 1,1.0000001 0,0 0))
```

Figure



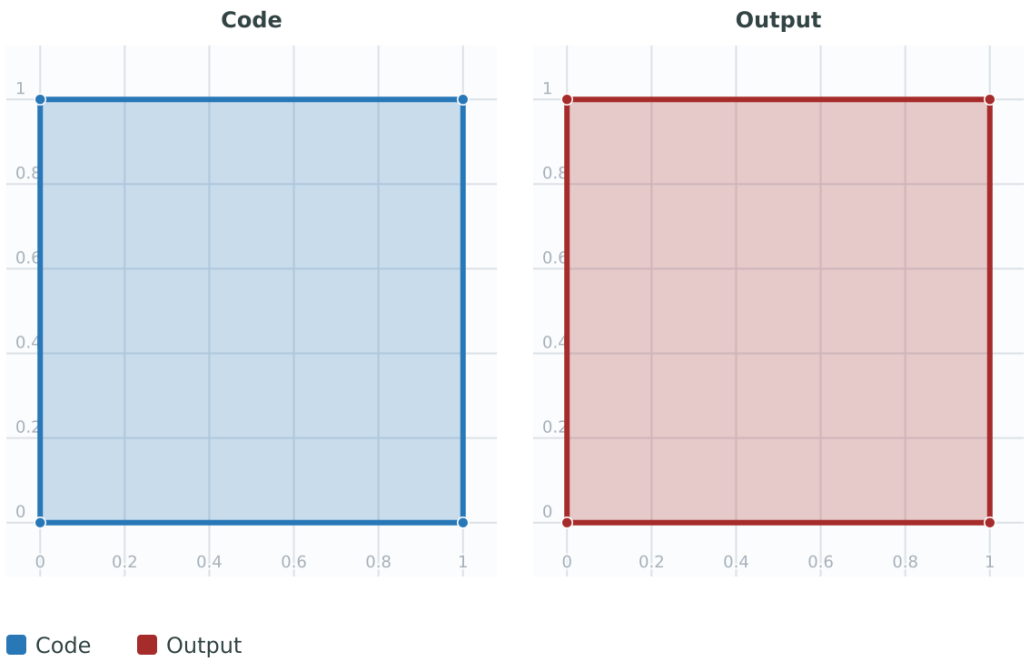
Code

```
SELECT ST_Envelope('POLYGON((0 0,0 1,1.0000000001 1,1.0000000001 0,0 0))'::geometry);
```

Output

```
POLYGON((0 0,0 1,1.0000000001 1,1.0000000001 0,0 0))
```

Figure



Code

```
SELECT Box3D(geom), Box2D(geom), ST_Envelope(geom) As envelopewkt
    FROM (SELECT 'POLYGON((0 0,0 1000012333334.34545678,1.0000001 1,1.0000001 0,0 0))':: ↩
    geometry As geom) As foo;
```

Envelope of a point and LineString.

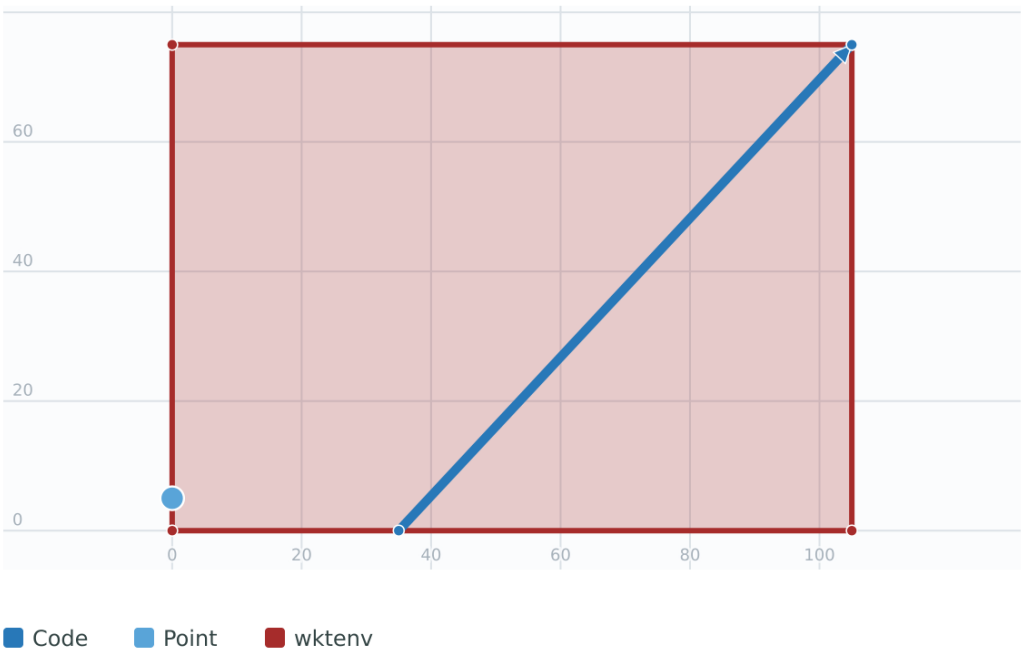
Code

```
SELECT ST_Envelope(ST_Collect(
            'LINESTRING(55 75,125 150)',
            ST_Point(20, 80)
            )) As wktenv;
```

Output

```
POLYGON((20 75,20 150,125 150,125 75,20 75))
```

Figure



Se även

Box2D, Box3D, ST_OrientedEnvelope

7.4.12 ST_ExteriorRing

ST_ExteriorRing — Returnerar en LineString som representerar den yttre ringen av en Polygon.

Synopsis

geometry **ST_ExteriorRing**(geometry a_polygon);

Beskrivning

Returnerar en LINESTRING som representerar den yttre ringen (skalet) av en POLYGON. Returnerar NULL om geometrin inte är en polygon.

Notera
Denna funktion stöder inte MULTIPOLYGON. För MULTIPOLYGON, använd tillsammans med ST_GeometryN eller ST_Dump

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. 2.1.5.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.2.3, 8.3.3

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Extract the exterior ring from a Polygon with a hole.

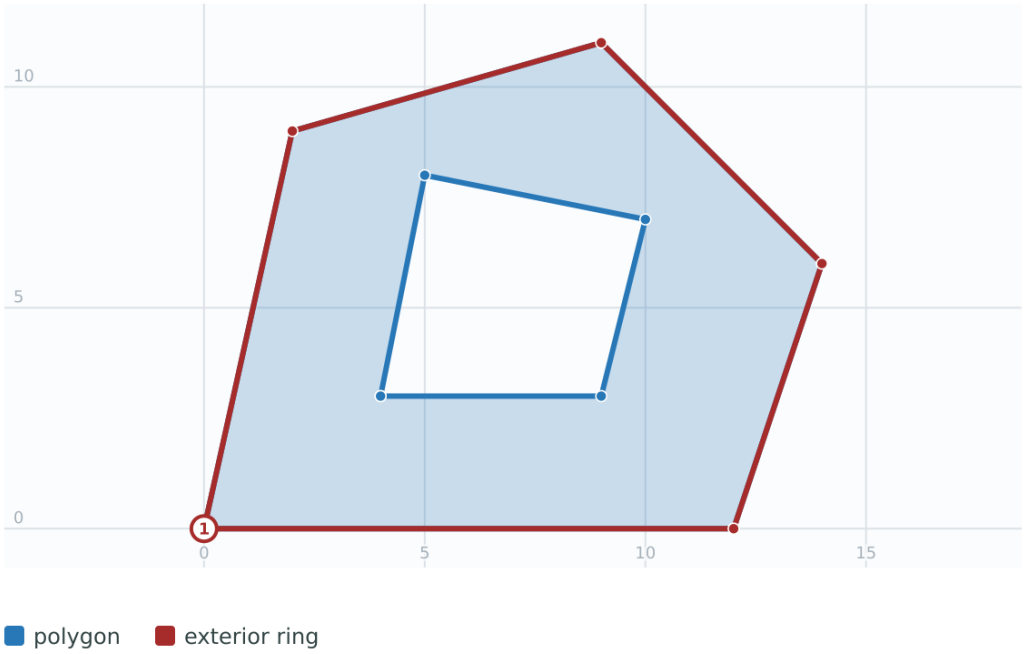
Code

```
WITH input(geom) AS (VALUES (
  'POLYGON((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3))'::geometry
))
SELECT geom AS input_polygon,
       ST_ExteriorRing(geom) AS exterior_ring
FROM input;
```

Output

```
POLYGON((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3)) | LINESTRING(0 0,2 9,9 11,14 6,12 0,0 0)
```

Figure



For a MultiPolygon, dump the component Polygons before extracting their exterior rings. Both component Polygons in this example have holes.

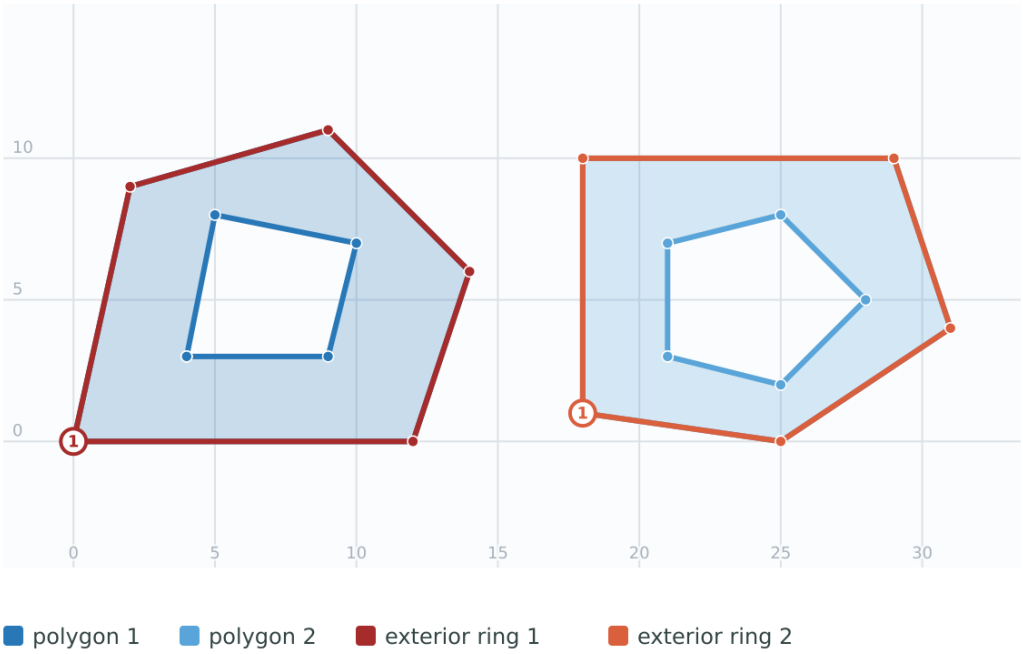
Code

```
WITH input(geom) AS (VALUES (
  'MULTIPOLYGON(
    ((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3)),
    ((18 1,18 10,29 10,31 4,25 0,18 1),(21 3,25 2,28 5,25 8,21 7,21 3))
  )'::geometry
))
SELECT part.geom AS input_polygon,
       ST_ExteriorRing(part.geom) AS exterior_ring
FROM input
CROSS JOIN LATERAL ST_Dump(geom) AS part
ORDER BY part.path;
```

Output

```
POLYGON((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3)) | LINESTRING(0 0,2 9,9 11,14 6,12 0,0 0)
POLYGON((18 1,18 10,29 10,31 4,25 0,18 1),(21 3,25 2,28 5,25 8,21 7,21 3)) | LINESTRING(18 1,18 10,29 10,31 4,25 0,18 ↩
    1)
```

Figure



Extract exterior rings from a table of Polygons.

Code

```
SELECT gid, ST_ExteriorRing(geom) AS ering
FROM sometable;
```

For a table of MultiPolygons, dump the component Polygons and collect their exterior rings into a MultiLineString.

Code

```
SELECT gid, ST_Collect(ST_ExteriorRing(geom)) AS erings
FROM (
  SELECT gid, (ST_Dump(geom)).geom AS geom
  FROM sometable
) AS foo
GROUP BY gid;
```

This example returns the 3D exterior ring of a Polygon.

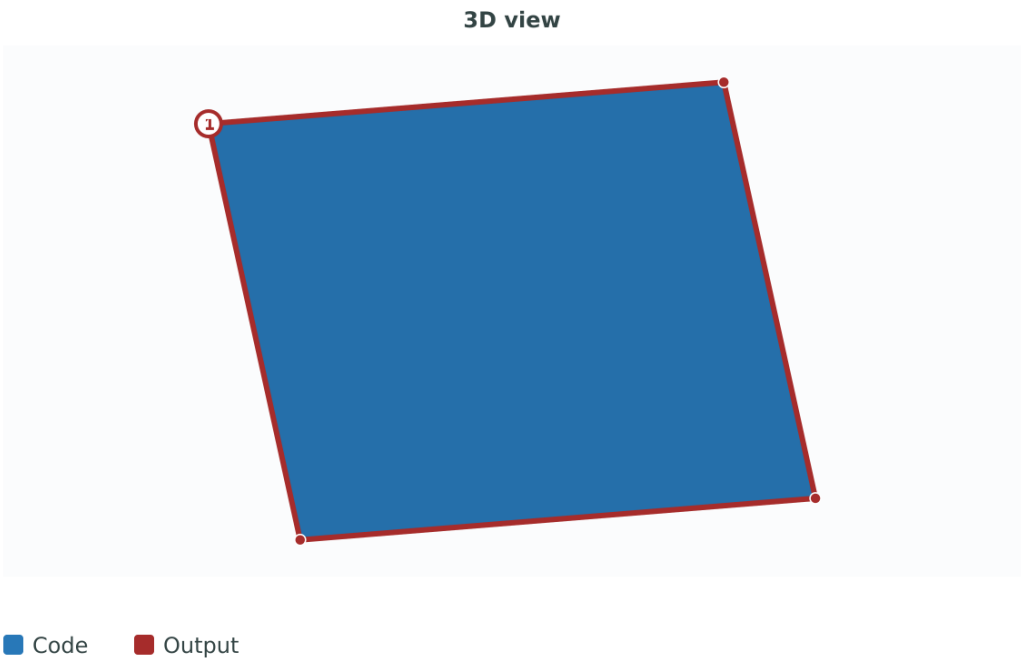
Code

```
SELECT ST_ExteriorRing(
    'POLYGON Z ((0 0 5,4 0 5,4 3 5,0 3 5,0 0 5))'
    );
```

Output

```
LINESTRING(0 0 5,4 0 5,4 3 5,0 3 5,0 0 5)
```

Figure



Se även

ST_InteriorRingN, ST_Boundary, ST_NumInteriorRings

7.4.13 ST_GeometryN

ST_GeometryN — Returnerar ett element i en geometrisamling.

Synopsis

geometry **ST_GeometryN**(geometry geomA, integer n);

Beskrivning

Returns the 1-based Nth element of a GEOMETRYCOLLECTION, MULTIPOINT, MULTILINESTRING, MULTICURVE, or MULTIPOLYGON. For a unitary geometry, including a TIN or PolyhedralSurface, an index of 1 returns the input geometry and any other index returns NULL. Use ST_PatchN and ST_NumPatches to access and count the faces of a TIN or PolyhedralSurface.

Notera
Index är 1-baserat som för OGC-specifikationer sedan version 0.8.0. Tidigare versioner implementerade detta som 0-baserat istället.

Notera
För att extrahera alla element i en geometri är ST_Dump mer effektivt och fungerar för atom-geometrier.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Ändrad: 2.0.0 Tidigare versioner skulle returnera NULL för singulära geometrier. Detta ändrades till att returnera geometrin för ST_GeometryN(..,1) fallet.

Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and are not decomposed into patches; use ST_PatchN to access faces.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 9.1.5

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

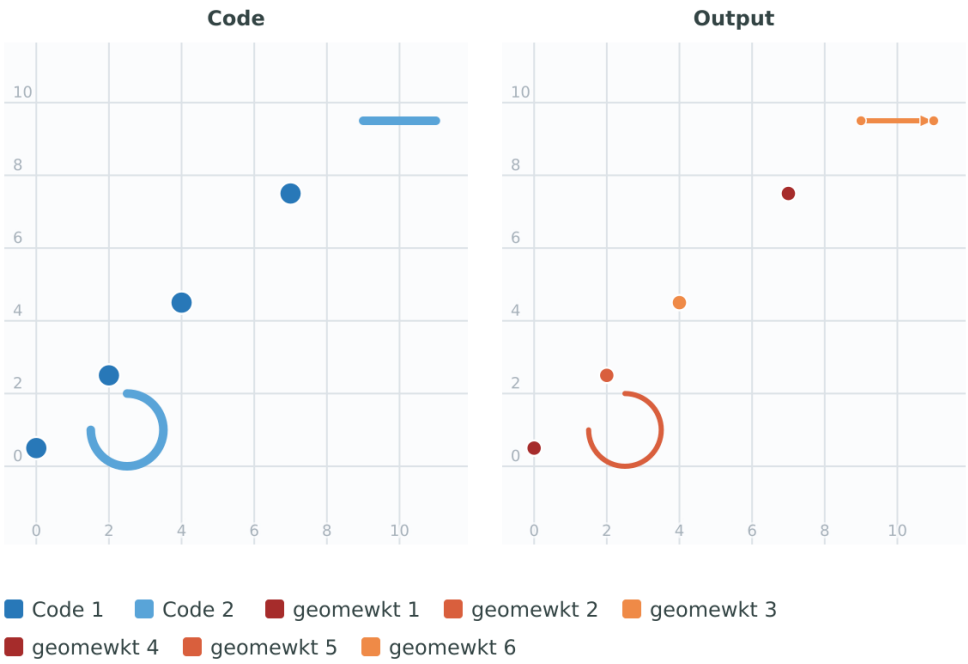
Standardexempel

Extracting a subset of points from a 3D multipoint.

Code

```
SELECT n, ST_GeometryN(geom, n) As geomewkt
FROM (
VALUES
    (1, 'MULTIPOINT((1 2 7), (3 4 7), (5 6 7), (8 9 10))'::geometry ),
    (2, 'MULTICURVE(CIRCULARSTRING(2.5 2.5,4.5 2.5, 3.5 3.5), (10 11, 12 11))'::geometry )
    ) As foo(id, geom)
    CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(geom)
ORDER BY id, n;
```

Figure



Extracting all geometries (useful when you want to assign an id).

Code

```
SELECT gid, n, ST_GeometryN(geom, n)
FROM sometable CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(geom);
```

Polyedriska ytor, TIN- och triangelexempel

A PolyhedralSurface or TIN is treated as a single geometry, not as a collection of faces. `ST_GeometryN` does not decompose the surface into patches, so an index greater than 1 returns `NULL`. Use ST_PatchN and ST_NumPatches to access individual faces.

This example requests the third element of a PolyhedralSurface.

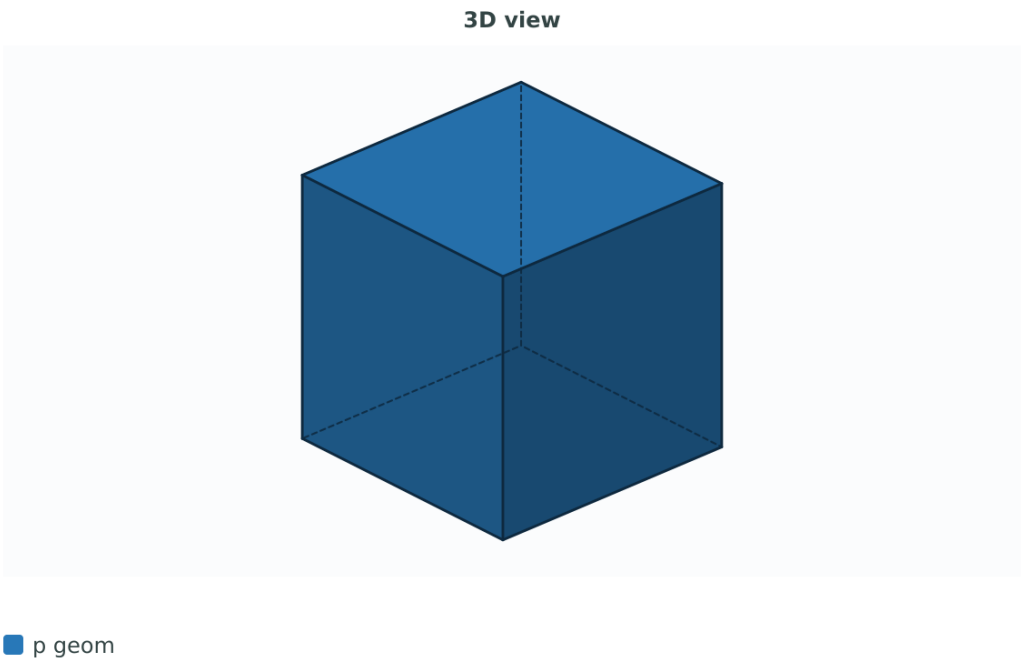
Code

```
SELECT ST_GeometryN(p_geom, 3) As geom_ewkt
  FROM (SELECT 'POLYHEDRALSURFACE(
((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
)'::geometry  AS p_geom )  AS a;
```

Output

```
NULL
```

Figure



This example requests the second element of a TIN.

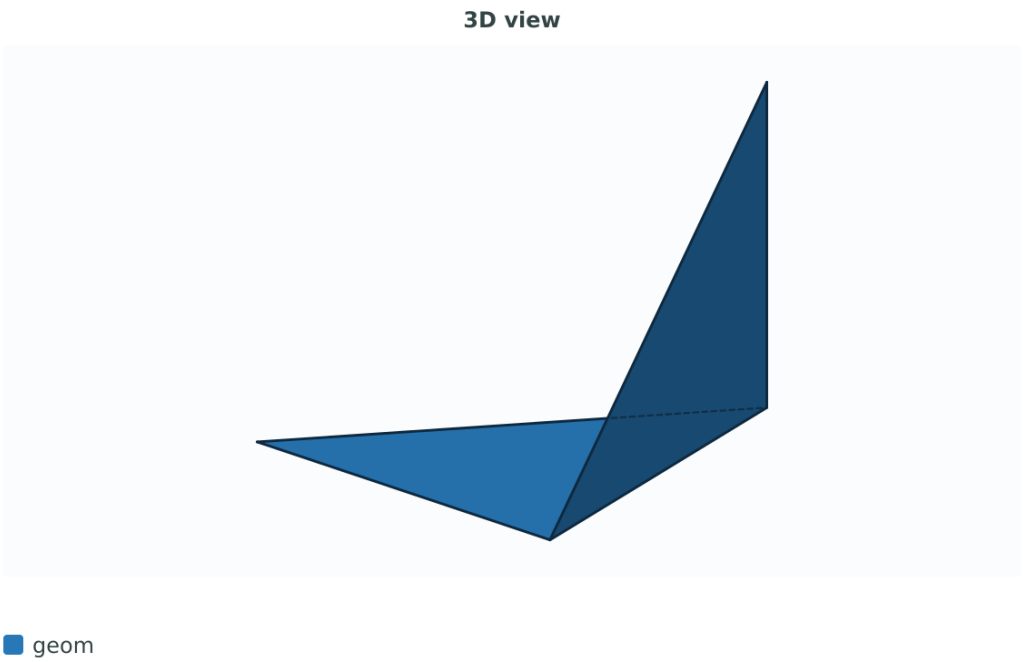
Code

```
SELECT ST_GeometryN(geom, 2) as wkt
  FROM
    (SELECT
       'TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
            )),((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
            ))
            )'::geometry  AS geom
    ) AS g;
```

Output

```
NULL
```

Figure



Se även

ST_Dump, ST_NumGeometries, ST_PatchN, ST_NumPatches

7.4.14 ST_GeometryType

ST_GeometryType — Returnerar SQL-MM-typen för en geometri som text.

Synopsis

text **ST_GeometryType**(geometry g1);

Beskrivning

Returnerar geometrins typ som en sträng. EG: ST_LineString”, ST_Polygon”, ST_MultiPolygon”etc. Denna funktion skiljer sig från GeometryType(geometry) när det gäller strängen och ST framför som returneras, samt det faktum att den inte kommer att ange om geometrin är uppmätt.

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.4

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_GeometryType('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)':: ↩
    geometry);
```

Output

```
ST_LineString
```

Code

```
SELECT ST_GeometryType('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry);
```

Output

```
ST_PolyhedralSurface
```

Code

```
SELECT ST_GeometryType(geom) AS result
FROM
  (SELECT
     'TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
          )),((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
          ))
          )'::geometry AS geom
  ) AS g;
```

Output

```
ST_Tin
```

Se även

GeometryType

7.4.15 ST_HasArc

ST_HasArc — Testar om en geometri innehåller en cirkelbåge

Synopsis

boolean **ST_HasArc**(geometry geomA);

Beskrivning

Returnerar true om en geometri eller geometrisamling innehåller en cirkulär sträng

Tillgänglighet: 1.2.3?

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

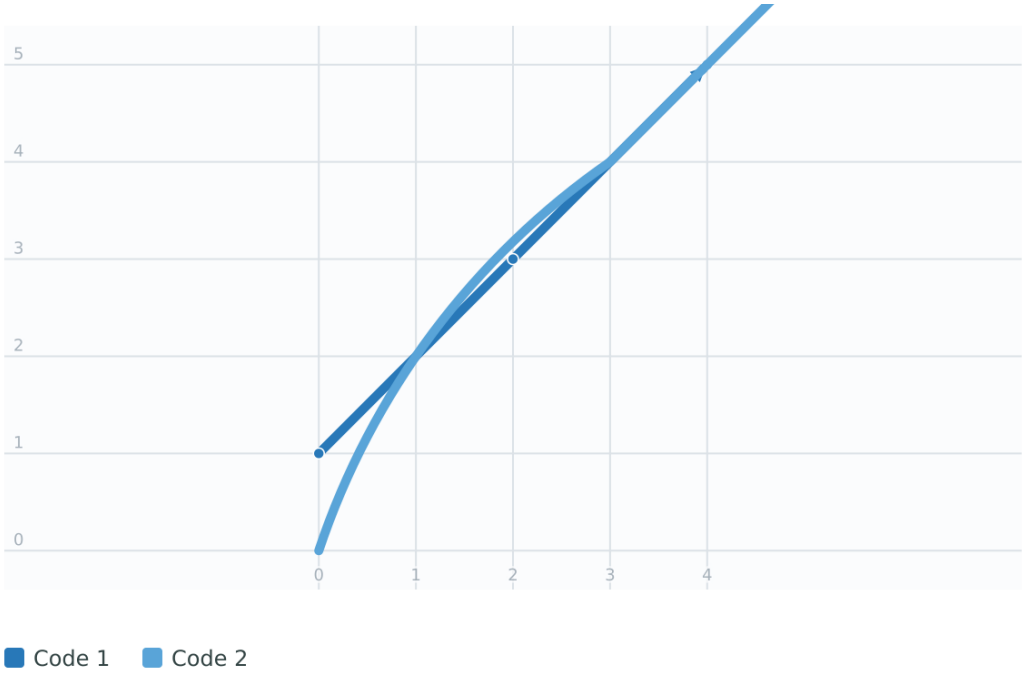
Code

```
SELECT ST_HasArc(ST_Collect('LINESTRING(1 2,3 4,5 6)', 'CIRCULARSTRING(1 1,2 3,4 5,6 7,5 6) ↩
    '));
```

Output

```
t
```

Figure



Se även

ST_CurveToLine, ST_LineToCurve

7.4.16 ST_InteriorRingN

ST_InteriorRingN — Returnerar den N:te inre ringen (hålet) i en polygon.

Synopsis

geometry **ST_InteriorRingN**(geometry a_polygon, integer n);

Beskrivning

Returnerar den N:te inre ringen (hålet) i en POLYGON-geometri som en LINESTRING. Indexet börjar på 1. Returnerar NULL om geometrin inte är en polygon eller om indexet är utanför intervallet.

Notera
Denna funktion stöder inte MULTIPOLYGON. För MULTIPOLYGON, använd tillsammans med ST_GeometryN eller ST_Dump

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.2.6, 8.3.5

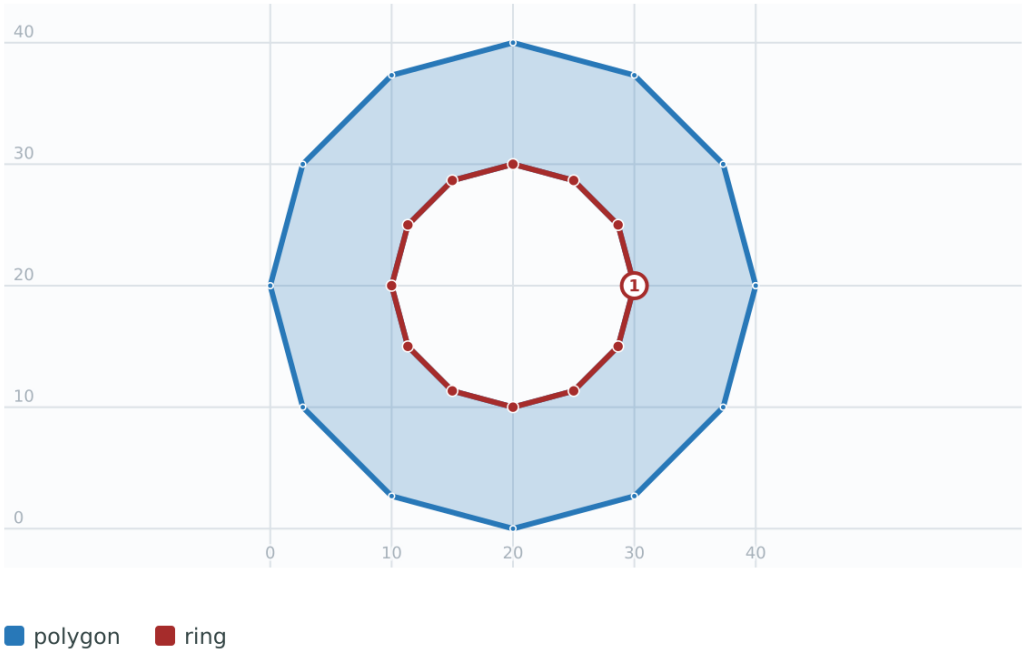
Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
WITH data AS (
  SELECT ST_BuildArea(ST_Collect(
    ST_Buffer(ST_Point(1, 2), 20, 3),
    ST_Buffer(ST_Point(1, 2), 10, 3)
  )) AS geom
)
SELECT geom AS input_polygon,
       ST_InteriorRingN(geom, 1) AS ring
FROM data;
```

Figure



Se även

ST_ExteriorRing, ST_BuildArea, ST_Collect, ST_Dump, ST_NumInteriorRing, ST_NumInteriorRings

7.4.17 ST_NumCurves

ST_NumCurves — Returnerar antalet komponentkurvor i en CompoundCurve.

Synopsis

integer **ST_NumCurves**(geometry a_compoundcurve);

Beskrivning

Returnerar antalet komponentkurvor i en CompoundCurve, noll för en tom CompoundCurve, eller NULL för en icke-CompoundCurve-indata.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.2.6, 8.3.5

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

This CompoundCurve contains three component curves.

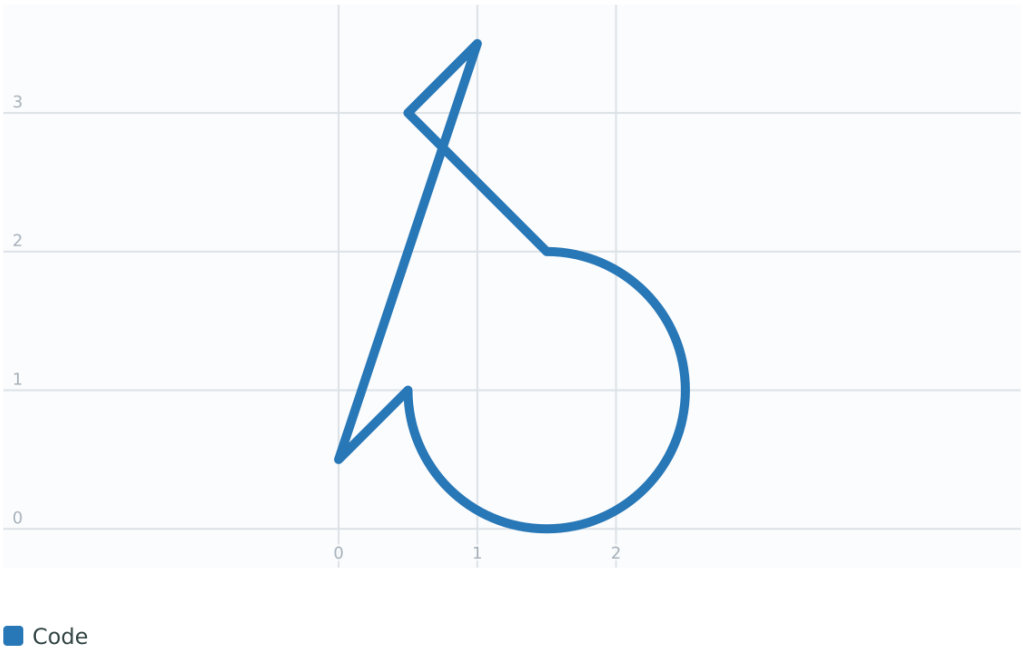
Code

```
SELECT ST_NumCurves('COMPOUNDCURVE(
    (2 2,2.5 2.5),
    CIRCULARSTRING(2.5 2.5,4.5 2.5,3.5 3.5),
    (3.5 3.5,2.5 4.5,3 5,2 2)
  )');
```

Output

```
3
```

Figure



An empty CompoundCurve has no component curves.

Code

```
SELECT ST_NumCurves('COMPOUNDCURVE EMPTY');
```

Output

```
0
```

Se även

ST_CurveN, ST_Dump, ST_ExteriorRing, ST_NumInteriorRings, ST_NumGeometries

7.4.18 ST_CurveN

ST_CurveN — Returnerar den N:te komponentkurvgeometrin för en CompoundCurve.

Synopsis

geometry **ST_CurveN**(geometry a_compoundcurve, integer index);

Beskrivning

Returnerar den N:te komponentkurvgeometrin i en CompoundCurve. Indexet börjar på 1. Returnerar NULL om geometrin inte är en CompoundCurve eller om indexet är utanför intervallet.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.2.6, 8.3.5

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

This example extracts the second component curve, a CircularString, from a CompoundCurve.

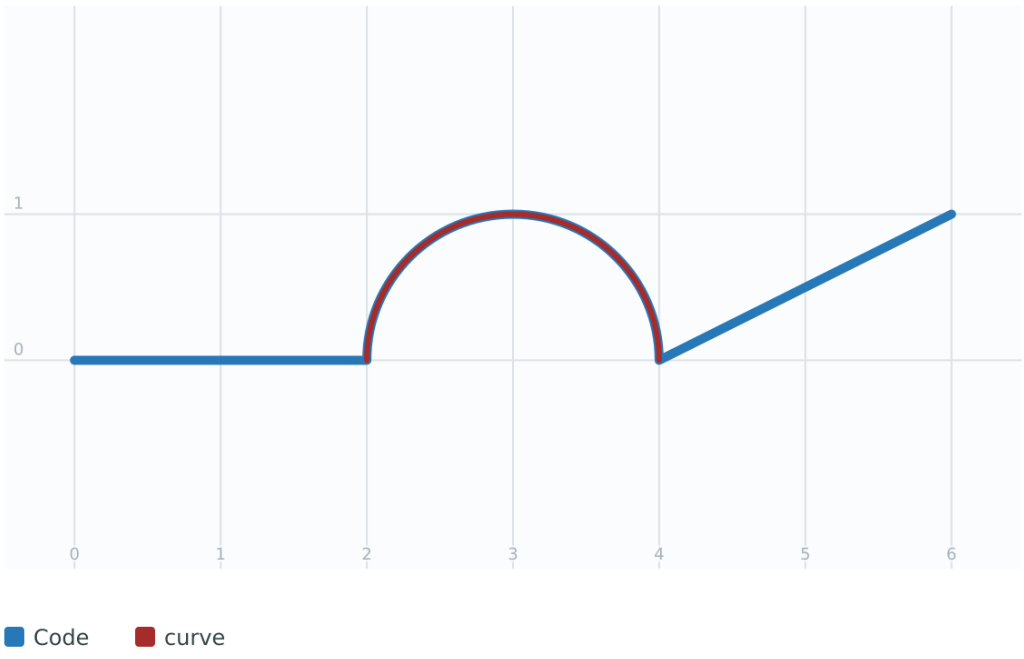
Code

```
SELECT ST_CurveN(
  'COMPOUNDCURVE(
    (0 0,2 0),
    CIRCULARSTRING(2 0,3 1,4 0),
    (4 0,6 1)
  )',
  2) AS curve;
```

Output

```
CIRCULARSTRING(2 0,3 1,4 0)
```

Figure



Se även

ST_NumCurves, ST_Dump, ST_ExteriorRing, ST_NumInteriorRings, ST_NumGeometries

7.4.19 ST_IsClosed

ST_IsClosed — Testar om en LineStrings start- och slutpunkter är sammanfallande. För en Polyhedral-Surface testas om den är sluten (volymetrisk).

Synopsis

boolean **ST_IsClosed**(geometry g);

Beskrivning

Returnerar `TRUE` om `LINESTRINGs` start- och slutpunkter är sammanfallande. För polyedriska ytor rapporteras om ytan är areell (öppen) eller volymetrisk (sluten).

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.1.5, 9.3.3

Notera
SQL-MM definierar att resultatet av `ST_IsClosed(NULL )` ska vara 0, medan PostGIS returnerar `NULL`.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Denna funktion stöder polyedriska ytor.

Linje Sträng och punkt Exempel

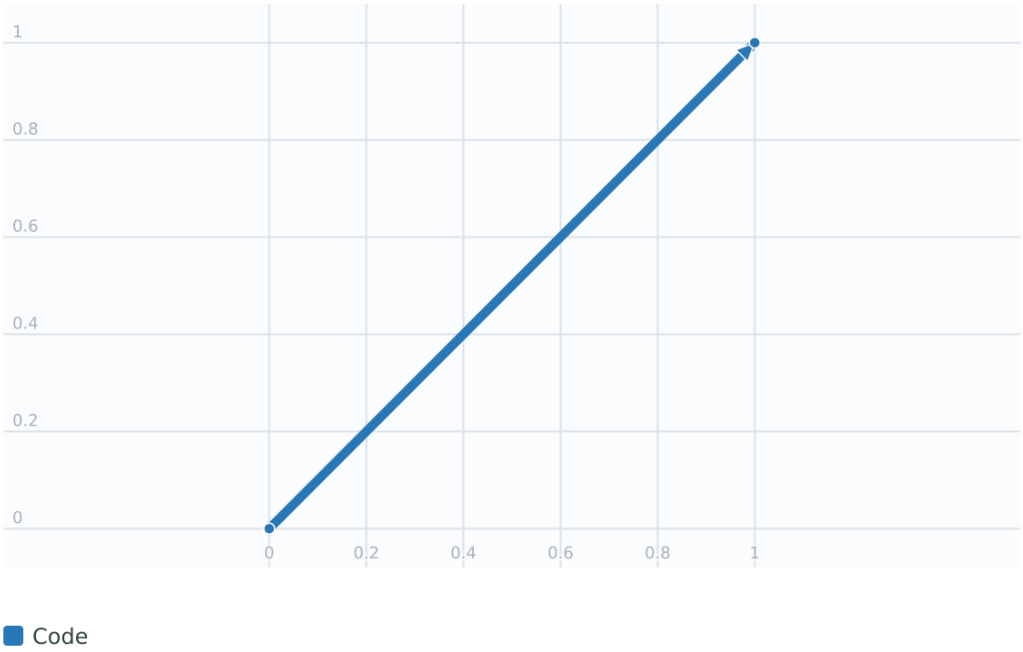
Code

```
SELECT ST_IsClosed('LINESTRING(0 0,1 1)'::geometry);
```

Output

```
f
```

Figure



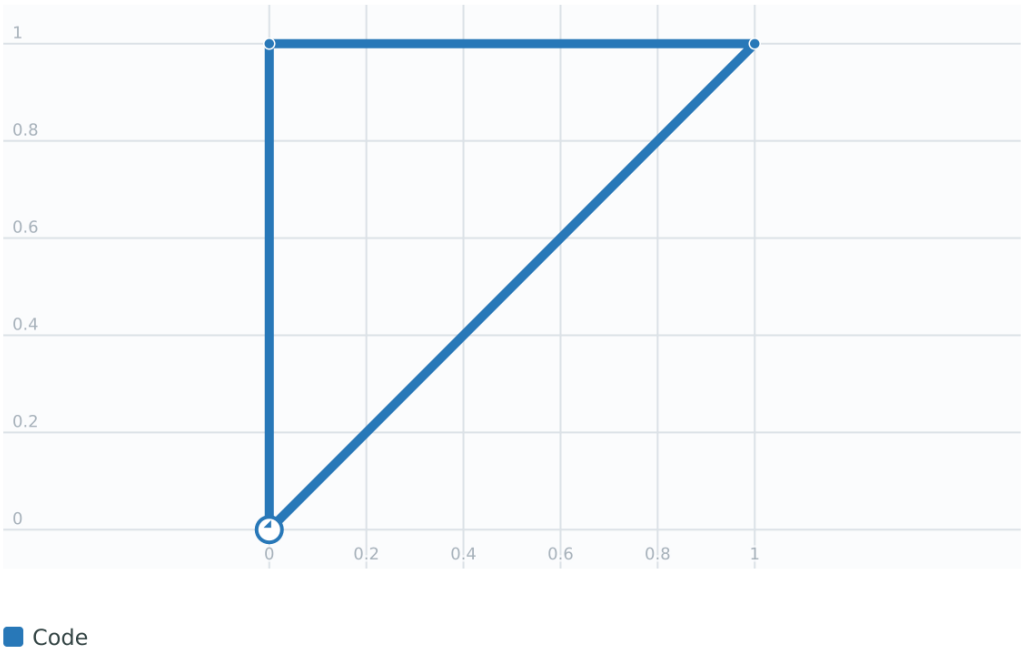
Code

```
SELECT ST_IsClosed('LINESTRING(0 0,0 1,1 1,0 0)'::geometry);
```

Output

```
t
```

Figure



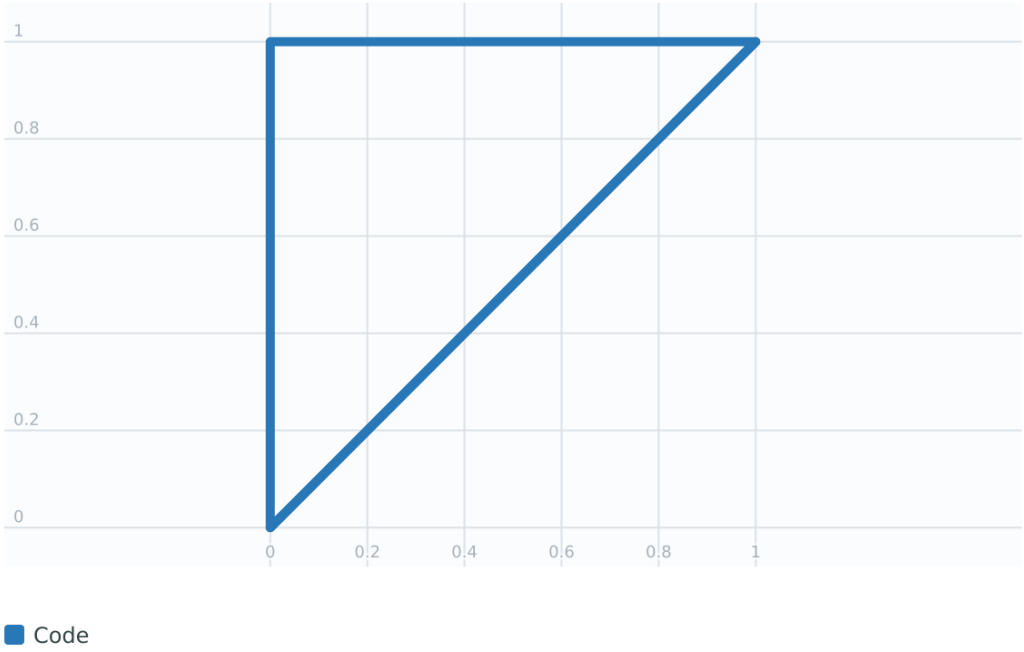
Code

```
SELECT ST_IsClosed('MULTILINESTRING((0 0,0 1,1 1,0 0),(0 0,1 1))'::geometry);
```

Output

```
f
```

Figure



Code

```
SELECT ST_IsClosed('POINT(0 0)'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsClosed('MULTIPOINT((0 0),(1 1))'::geometry);
```

Output

```
t
```

Exempel på polyedrisk yta

This example uses a cube.

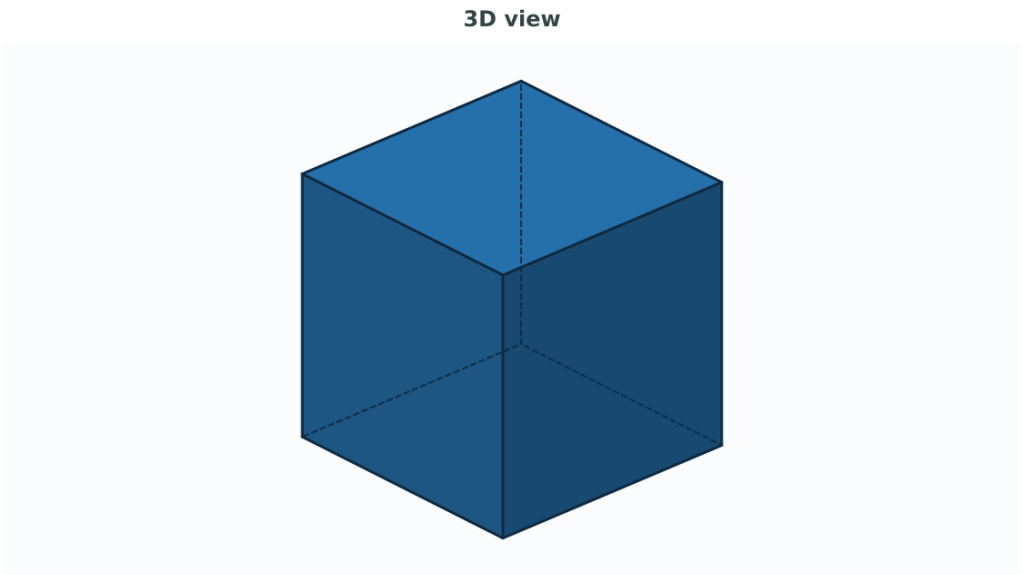
Code

```
SELECT ST_IsClosed('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )');
```

Output

```
t
```

Figure



This example uses the same cube with one side missing.

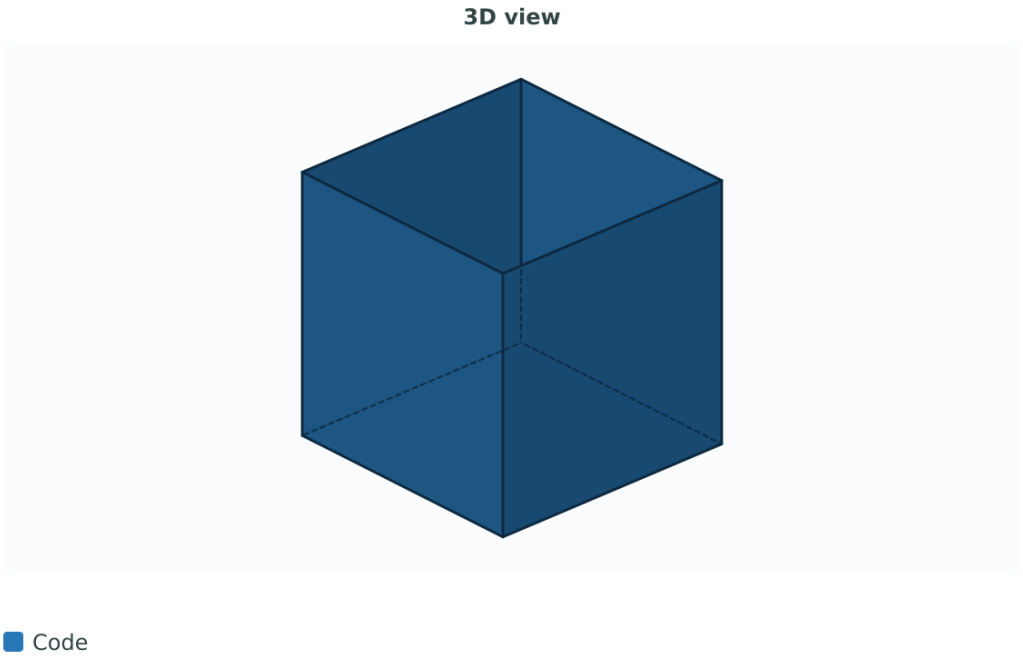
Code

```
SELECT ST_IsClosed('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
       ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
       ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
       ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)) )');
```

Output

```
f
```

Figure



Se även

ST_IsRing

7.4.20 ST_IsCollection

ST_IsCollection — Testar om en geometri är en geometrisamlingstyp.

Synopsis

boolean **ST_IsCollection**(geometry g);

Beskrivning

Returnerar TRUE om argumentets geometrityp är en geometrisamlingstyp. Samlingstyperna är följande:

- GEOMETRYCOLLECTION
- MULTI{POINT,POLYGON,LINESTRING,CURVE,SURFACE}
- COMPOUNDCURVE

Notera
Denna funktion analyserar geometrins typ. Detta innebär att den kommer att returnera TRUE på samlingar som är tomma eller som innehåller ett enda element.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

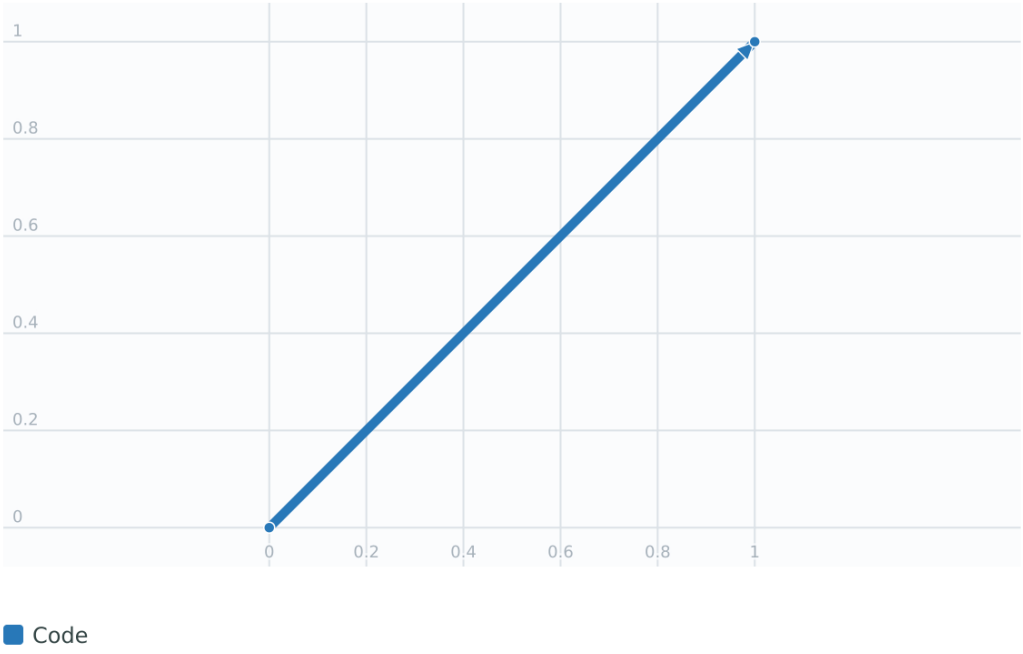
Code

```
SELECT ST_IsCollection('LINESTRING(0 0,1 1)'::geometry);
```

Output

```
f
```

Figure



Code

```
SELECT ST_IsCollection('MULTIPOINT EMPTY'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsCollection('MULTIPOINT((0 0))'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsCollection('MULTIPOINT((0 0),(42 42))'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsCollection('GEOMETRYCOLLECTION(POINT(0 0))'::geometry);
```

Output

```
t
```

Se även

ST_NumGeometries

7.4.21 ST_IsEmpty

ST_IsEmpty — Testar om en geometri är tom.

Synopsis

boolean **ST_IsEmpty**(geometry geomA);

Beskrivning

Returnerar true om denna geometri är en tom geometri. Om true, så representerar denna geometri en tom geometrisamling, polygon, punkt etc.

Notera
SQL-MM definierar att resultatet av ST_IsEmpty(NULL) ska vara 0, medan PostGIS returnerar NULL.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.7

Denna metod stöder cirkulära strängar och kurvor.

Ändrad: 2.0.0 I tidigare versioner av PostGIS var ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)') tillåtet. Detta är nu olagligt i PostGIS 2.0.0 för att bättre överensstämma med SQL/MM-standarder

Exempel

Code

```
SELECT ST_IsEmpty('GEOMETRYCOLLECTION EMPTY'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsEmpty('POLYGON EMPTY'::geometry);
```

Output

```
t
```

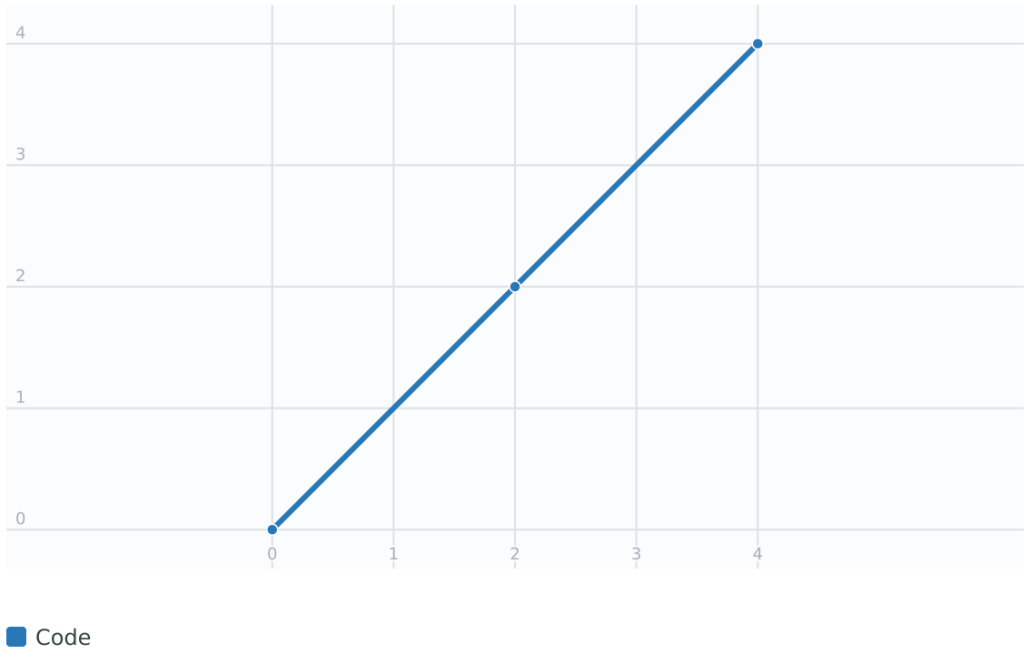
Code

```
SELECT ST_IsEmpty('POLYGON((1 2,3 4,5 6,1 2))'::geometry);
```

Output

```
f
```

Figure



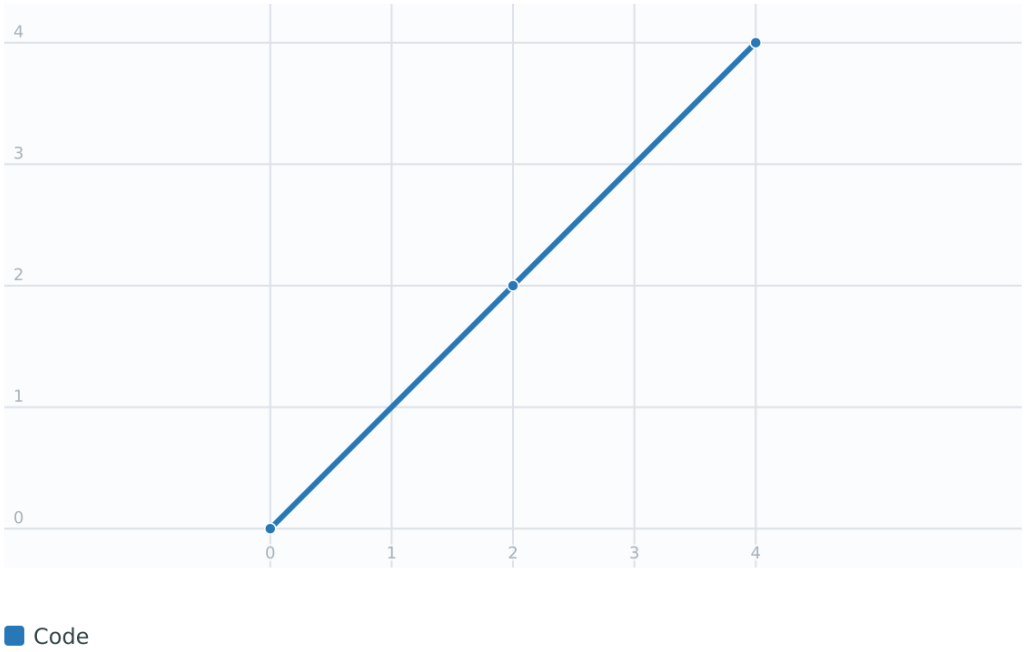
Code

```
SELECT ST_IsEmpty('POLYGON((1 2,3 4,5 6,1 2))'::geometry) = false;
```

Output

```
t
(1 row)
```

Figure



Code

```
SELECT ST_IsEmpty('CIRCULARSTRING EMPTY'::geometry);
```

Output

```
t
```

7.4.22 ST_IsPolygonCCW

ST_IsPolygonCCW — Testar om polygoner har yttre ringar som är orienterade moturs och inre ringar som är orienterade medurs.

Synopsis

boolean **ST_IsPolygonCCW**(geometry geom);

Beskrivning

Returnerar true om alla polygonala komponenter i indatageometrin använder en motsols orientering för sin yttre ring och en medsols riktning för alla inre ringar.

Returnerar true om geometrin inte har några polygonala komponenter.

Notera
Slutna linestrings betraktas inte som polygonala komponenter, så du skulle fortfarande få ett sant resultat genom att skicka en enda sluten linestrings oavsett dess orientering.

Notera
Om en polygonal geometri inte använder omvänd orientering för inre ringar (d.v.s. om en eller flera inre ringar är orienterade i samma riktning som en yttre ring) kommer både ST_IsPolygonCW och ST_IsPolygonCCW att returnera false.

Tillgänglighet: 2.4.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Se även

ST_ForcePolygonCW, ST_ForcePolygonCCW, ST_IsPolygonCW

7.4.23 ST_IsPolygonCW

ST_IsPolygonCW — Testar om polygoner har yttre ringar som är orienterade medurs och inre ringar som är orienterade moturs.

Synopsis

boolean **ST_IsPolygonCW**(geometry geom);

Beskrivning

Returnerar true om alla polygonala komponenter i indatageometrin använder en medurs riktning för sin yttre ring och en moturs riktning för alla inre ringar.

Returnerar true om geometrin inte har några polygonala komponenter.

Notera
Slutna linestrings betraktas inte som polygonala komponenter, så du skulle fortfarande få ett sant resultat genom att skicka en enda sluten linestrings oavsett dess orientering.

Notera
Om en polygonal geometri inte använder omvänd orientering för inre ringar (d.v.s. om en eller flera inre ringar är orienterade i samma riktning som en yttre ring) kommer både ST_IsPolygonCW och ST_IsPolygonCCW att returnera false.

Tillgänglighet: 2.4.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Se även

ST_ForcePolygonCW, ST_ForcePolygonCCW, ST_IsPolygonCW

7.4.24 ST_IsRing

ST_IsRing — Testar om en LineString är sluten och enkel.

Synopsis

boolean **ST_IsRing**(geometry g);

Beskrivning

Returns `TRUE` if this `LINESTRING` is both ST_IsClosed (`ST_StartPoint(g) ~= ST_EndPoint(g)`) and ST_IsSimple (does not self intersect).

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. 2.1.5.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.1.6

Notera
SQL-MM definierar att resultatet av `ST_IsRing(NULL )` ska vara 0, medan PostGIS returnerar `NULL`.

Exempel

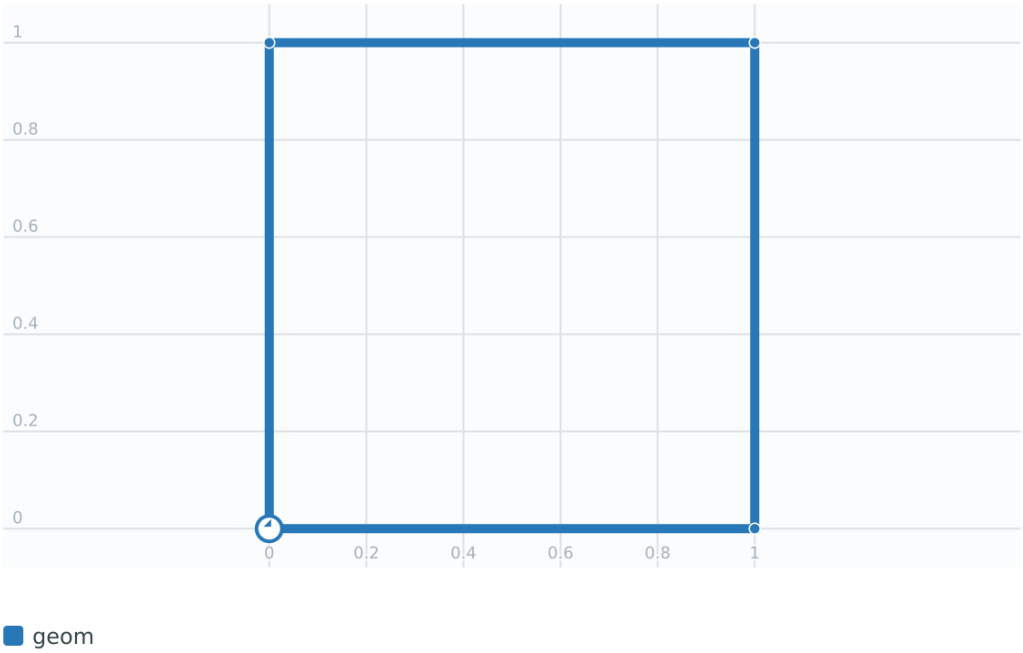
Code

```
SELECT ST_IsRing(geom), ST_IsClosed(geom), ST_IsSimple(geom)
FROM (SELECT 'LINESTRING(0 0,0 1,1 1,1 0,0 0)'::geometry AS geom) AS foo;
```

Output

```
st_isring | st_isclosed | st_issimple
-----------+-------------+-------------
 t         | t           | t
(1 row)
```

Figure



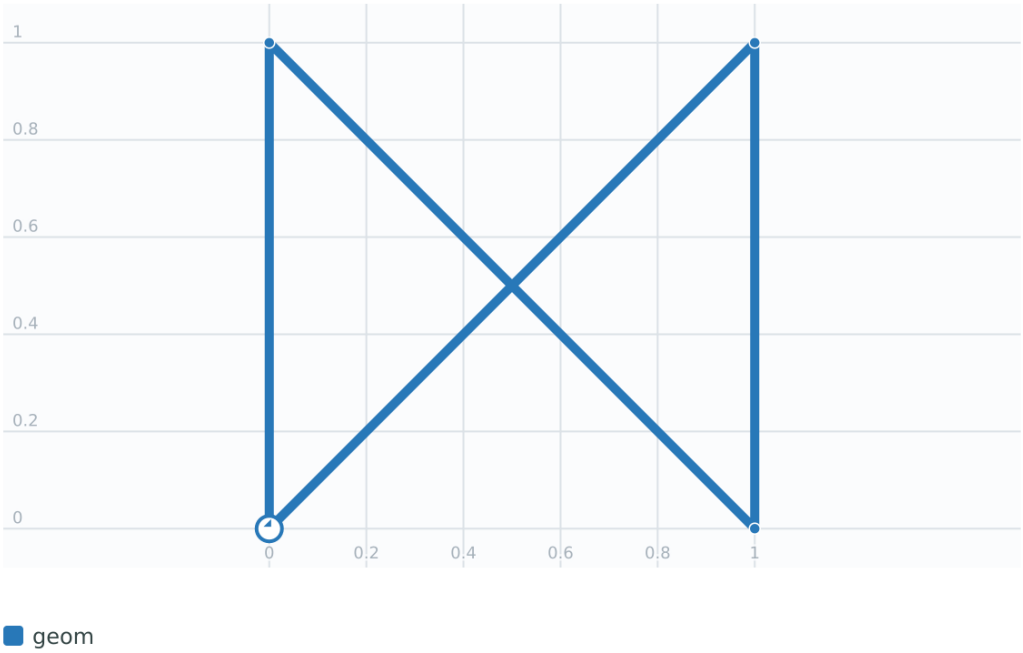
Code

```
SELECT ST_IsRing(geom), ST_IsClosed(geom), ST_IsSimple(geom)
FROM (SELECT 'LINESTRING(0 0,0 1,1 0,1 1,0 0)'::geometry AS geom) AS foo;
```

Output

```
st_isring | st_isclosed | st_issimple
-----------+-------------+-------------
 f         | t           | f
(1 row)
```

Figure



Se även

ST_IsClosed, ST_IsSimple, ST_StartPoint, ST_EndPoint

7.4.25 ST_IsSimple

ST_IsSimple — Testar om en geometri inte har några punkter med självskärning eller självtangentiering.

Synopsis

boolean **ST_IsSimple**(geometry geomA);

Beskrivning

Returnerar true om denna geometri inte har några avvikande geometriska punkter, t.ex. självskärning eller självtangentiering. Mer information om OGC:s definition av geometrins enkelhet och giltighet finns i ”Ensuring OpenGIS compliance of geometries”

Notera
SQL-MM definierar att resultatet av ST_IsSimple(NULL) ska vara 0, medan PostGIS returnerar NULL.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.8

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

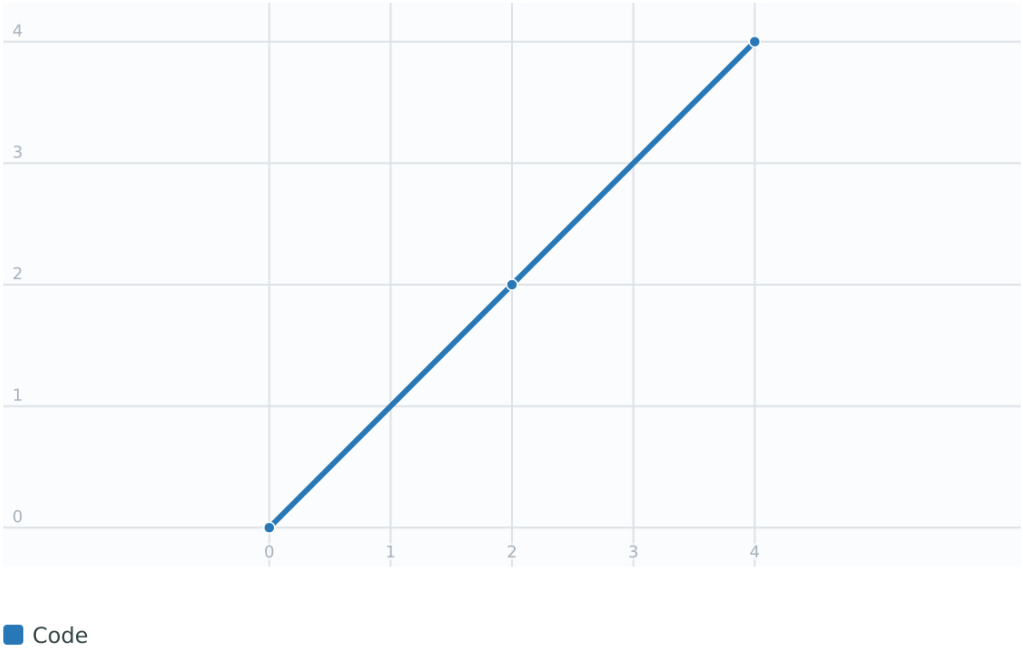
Code

```
SELECT ST_IsSimple('POLYGON((1 2,3 4,5 6,1 2))');
```

Output

```
f
```

Figure



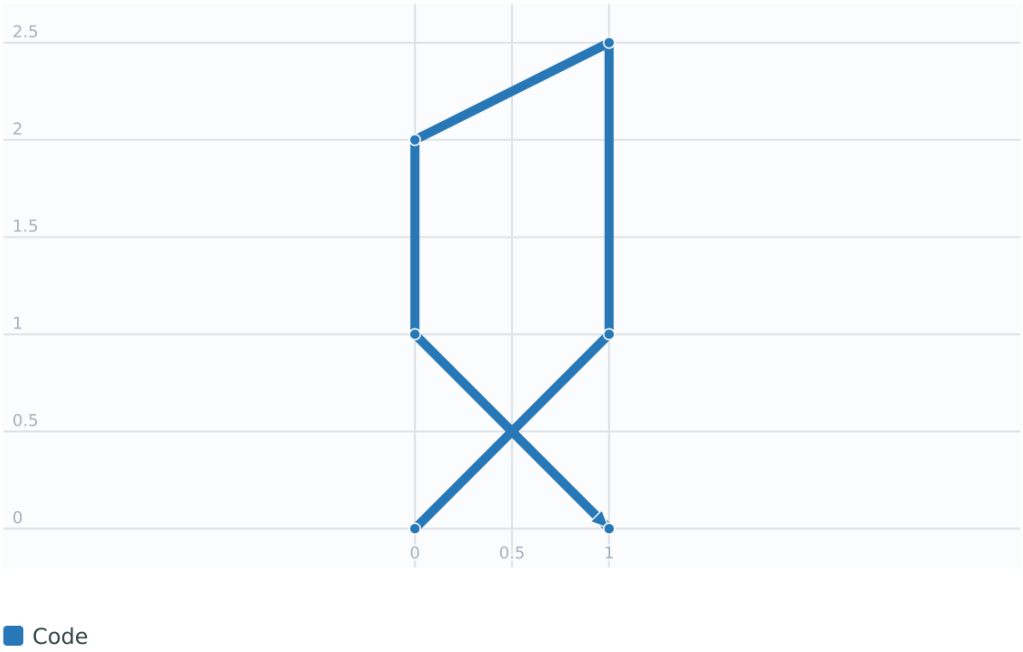
Code

```
SELECT ST_IsSimple('LINESTRING(1 1,2 2,2 3.5,1 3,1 2,2 1)');
```

Output

```
f
```

Figure



Se även

ST_IsValid

7.4.26 ST_M

ST_M — Returnerar M-koordinaten för en punkt.

Synopsis

float **ST_M**(geometry a_point);

Beskrivning

Returnerar M-koordinaten för en Point, eller NULL om den inte är tillgänglig. Indata måste vara en punkt.

Notera
Detta är (ännu) inte en del av OGC-specifikationen, men listas här för att komplettera listan över funktioner för extraktion av punktkoordinater.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_M('POINT(1 2 3 4)');
```

Output

```
4
```

Se även

ST_GeomFromEWKT, ST_X, ST_Y, ST_Z

7.4.27 ST_MemSize

ST_MemSize — Returnerar hur mycket minnesutrymme en geometri tar upp.

Synopsis

integer **ST_MemSize**(geometry geomA);

Beskrivning

Returnerar den mängd minnesutrymme (i byte) som geometrin tar upp.

Detta kompletterar PostgreSQL: s inbyggda databasobjektfunktioner pg_column_size, pg_size_pretty, pg_relation_size, pg_total_relation_size.

Notera
pg_relation_size som ger bytestorleken för en tabell kan returnera en bytestorlek som är lägre än ST_MemSize. Detta beror på att pg_relation_size inte lägger till bidrag från toasted-tabeller och att stora geometrier lagras i TOAST-tabeller.
pg_total_relation_size - inkluderar tabellen, de toastade tabellerna och indexen.
pg_column_size returnerar hur mycket utrymme en geometri skulle ta i en kolumn med hänsyn till komprimering, så det kan vara lägre än ST_MemSize

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Ändrad: 2.2.0 namnet ändrat till ST_MemSize för att följa namngivningskonventionen.

Exempel

Compare the memory used by all geometries in a Massachusetts town dataset with the subset for Boston.

Code

```
SELECT pg_size_pretty(sum(ST_MemSize(geom))) AS totgeomsum,
       pg_size_pretty(sum(CASE WHEN town = 'BOSTON' THEN ST_MemSize(geom) ELSE 0 END)) AS ↩
    bossum,
       cast(
           sum(CASE WHEN town = 'BOSTON' THEN ST_MemSize(geom) ELSE 0 END) * 100.0
           / sum(ST_MemSize(geom)) AS numeric(10, 2)
       ) AS perbos
FROM towns;
```

Output

```
totgeomsum | bossum | perbos
------------+--------+--------
 1522 kB    | 30 kB  |   1.99
(1 row)
```

Return the memory size of a CircularString in bytes.

Code

```
SELECT ST_MemSize('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'::geometry);
```

Output

```
st_memsize
------------
         80
(1 row)
```

Calculate the percentage of a table’s total size used by geometry data.

Code

```
SELECT pg_total_relation_size('public.neighborhoods') AS fulltable_size,
       sum(ST_MemSize(geom)) AS geomsize,
       sum(ST_MemSize(geom)) * 100.0
           / pg_total_relation_size('public.neighborhoods') AS pergeom
FROM neighborhoods;
```

Output

```
fulltable_size | geomsize |         pergeom
---------------+----------+-------------------------
         262144 |    96238 | 36.71188354492187500000
(1 row)
```

7.4.28 ST_NDims

ST_NDims — Returnerar koordinatdimensionen för en geometri.

Synopsis

integer **ST_NDims**(geometry g1);

Beskrivning

Returnerar geometrins koordinatdimension. PostGIS stöder 2 - (x,y) , 3 - (x,y,z) eller 2D med mått - x,y,m, och 4 - 3D med måttrymd x,y,z,m

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_NDims('POINT(1 1)') As d2point,
ST_NDims('POINT(1 1 2)') As d3point,
ST_NDims('POINTM(1 1 0.5)') As d2pointm;
```

Output

```
d2point | d3point | d2pointm
---------+---------+----------
       2 |       3 |        3
```

Se även

ST_CoordDim, ST_Dimension, ST_GeomFromEWKT

7.4.29 ST_NPoints

ST_NPoints — Returnerar antalet punkter (vertices) i en geometri.

Synopsis

integer **ST_NPoints**(geometry g1);

Beskrivning

Returnerar antalet punkter i en geometri. Fungerar för alla geometrier.
Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Count the points in a 2D LineString.

Code

```
SELECT ST_NPoints('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)');
```

Output

```
st_npoints
------------
          4
(1 row)
```

Count the points in a 3D LineString.

Code

```
SELECT ST_NPoints('LINESTRING(77.29 29.07 1,77.42 29.26 0,77.27 29.31 -1,77.29 29.07 3)');
```

Output

```
st_npoints
------------
          4
(1 row)
```

Se även

ST_NumPoints

7.4.30 ST_NRings

ST_NRings — Returnerar antalet ringar i en polygonal geometri.

Synopsis

integer **ST_NRings**(geometry geomA);

Beskrivning

Om geometrin är en polygon eller multipolygon returneras antalet ringar. Till skillnad från NumInteriorRings räknar den även de yttre ringarna.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

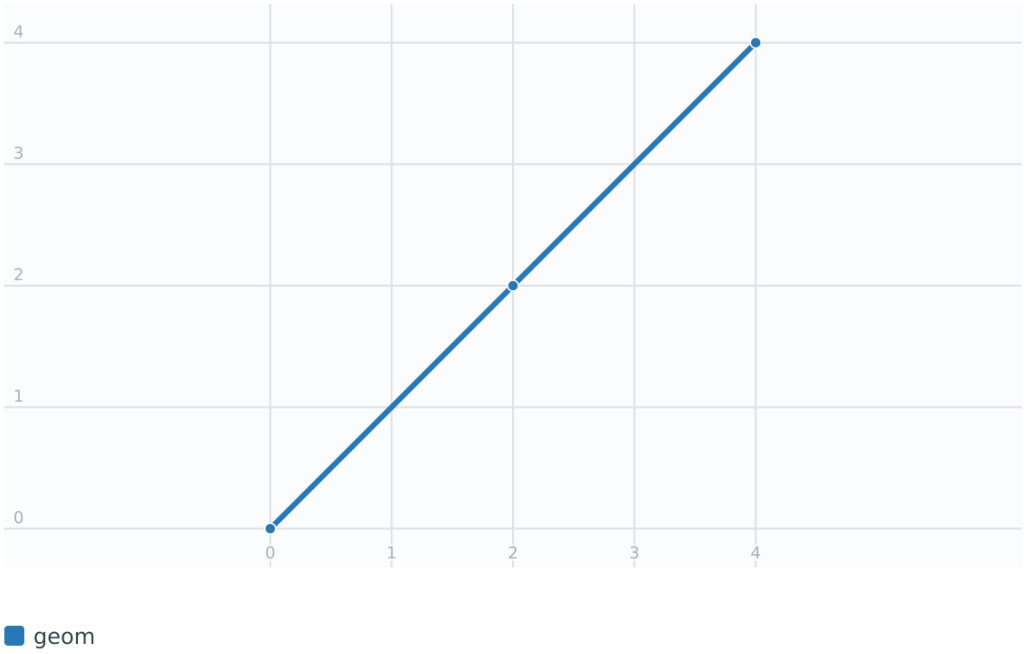
Code

```
SELECT ST_NRings(geom) As Nrings, ST_NumInteriorRings(geom) As ninterrings
FROM (SELECT 'POLYGON((1 2,3 4,5 6,1 2))'::geometry As geom) As foo;
```

Output

```
nrings | ninterrings
--------+-------------
      1 |           0
(1 row)
```

Figure



Se även

ST_NumInteriorRings

7.4.31 ST_NumGeometries

ST_NumGeometries — Returnerar antalet element i en geometrisamling.

Synopsis

integer **ST_NumGeometries**(geometry geom);

Beskrivning

Returnerar antalet element i en geometrisamling (GEOMETRYCOLLECTION eller MULTI*). För icke-tomma atomgeometrier returneras 1. För tomma geometrier returneras 0.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Ändrad: 2.0.0 I tidigare versioner skulle detta returnera NULL om geometrin inte var en samling/MULTI-typ. 2.0.0+ returnerar nu 1 för enskilda geometrier, t.ex. POLYGON, LINESTRING, POINT.

Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and return 1; use ST_NumPatch to count faces.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 9.1.4

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

This example returns 1; prior versions would have returned NULL.

Code

```
SELECT ST_NumGeometries('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)');
```

Output

```
st_numgeometries
-----------------
                1
(1 row)
```

A MultiPoint counts as one geometry within a GeometryCollection.

Code

```
SELECT ST_NumGeometries('GEOMETRYCOLLECTION(MULTIPOINT((-2 3),(-2 2)),
LINESTRING(5 5,10 10),
POLYGON((-7 4.2,-7.1 5,-7.1 4.3,-7 4.2)))');
```

Output

```
st_numgeometries
-----------------
                3
(1 row)
```

Se även

ST_GeometryN, ST_Multi

7.4.32 ST_NumInteriorRings

ST_NumInteriorRings — Returnerar antalet inre ringar (hål) i en polygon.

Synopsis

integer **ST_NumInteriorRings**(geometry a_polygon);

Beskrivning

Returnerar antalet inre ringar i en polygongeometri. Returnerar NULL om geometrin inte är en polygon.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.2.5

Ändrad: 2.0.0 - i tidigare versioner kunde man skicka en MULTIPOLYGON och få tillbaka antalet inre ringar i den första POLYGONEN.

Exempel

Count the interior rings of a Polygon.

Code

```
SELECT gid, field1, field2, ST_NumInteriorRings(geom) AS numholes
FROM sometable;
```

For a MultiPolygon, dump its component Polygons and sum their interior ring counts.

Code

```
SELECT gid, field1, field2, SUM(ST_NumInteriorRings(geom)) AS numholes
FROM (
    SELECT gid, field1, field2, (ST_Dump(geom)).geom AS geom
    FROM sometable
) AS foo
GROUP BY gid, field1, field2;
```

Se även

ST_NumInteriorRing, ST_InteriorRingN

7.4.33 ST_NumInteriorRing

ST_NumInteriorRing — Returns the number of interior rings (holes) of a Polygon. Alias for ST_NumInteriorRi

Synopsis

integer **ST_NumInteriorRing**(geometry a_polygon);

Beskrivning

This function is an alias for ST_NumInteriorRings.

Tillgänglighet: 1.2.2

Se även

ST_NumInteriorRings, ST_InteriorRingN

7.4.34 ST_NumPatches

ST_NumPatches — Returns the number of faces in a PolyhedralSurface or TIN.

Synopsis

integer **ST_NumPatches**(geometry g1);

Beskrivning

Returns the number of faces in a PolyhedralSurface or TIN. Returns NULL for other geometries. Unlike ST_NumGeometries, this function counts individual faces.

Tillgänglighet: 2.0.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM ISO/IEC 13249-3: 8.5

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_NumPatches('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )');
```

Output

```
st_numpatches
---------------
             6
(1 row)
```

Se även

ST_GeomFromEWKT, ST_NumGeometries, ST_PatchN

7.4.35 ST_NumPoints

ST_NumPoints — Returnerar antalet punkter i en LineString eller CircularString.

Synopsis

integer **ST_NumPoints**(geometry g1);

Beskrivning

Returnerar antalet punkter i ett ST_LineString- eller ST_CircularString-värde. Före 1.4 fungerar endast med linestringsar enligt specifikationerna. Från 1.4 och framåt är detta ett alias för ST_NPoints som returnerar antalet hörn för inte bara linestringsar. Överväg att använda ST_NPoints istället som är mångsidigt och fungerar med många geometrityper.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.2.4

Exempel

Code

```
SELECT ST_NumPoints('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)');
```

Output

```
st_numpoints
--------------
           4
(1 row)
```

Se även

ST_NPoints

7.4.36 ST_PatchN

ST_PatchN — Returns the Nth face of a PolyhedralSurface or TIN.

Synopsis

geometry **ST_PatchN**(geometry geomA, integer n);

Beskrivning

Returns the 1-based Nth face of a PolyhedralSurface or TIN. Returns NULL for other geometries or an out-of-range index. Use this function to access faces, because ST_GeometryN treats PolyhedralSurface and TIN as unitary geometries.

Notera
Index är 1-baserat.

Notera
Om du vill extrahera alla element i en geometri är ST_Dump mer effektivt.

Tillgänglighet: 2.0.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM ISO/IEC 13249-3: 8.5

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Extract the 2nd face of the polyhedral surface.

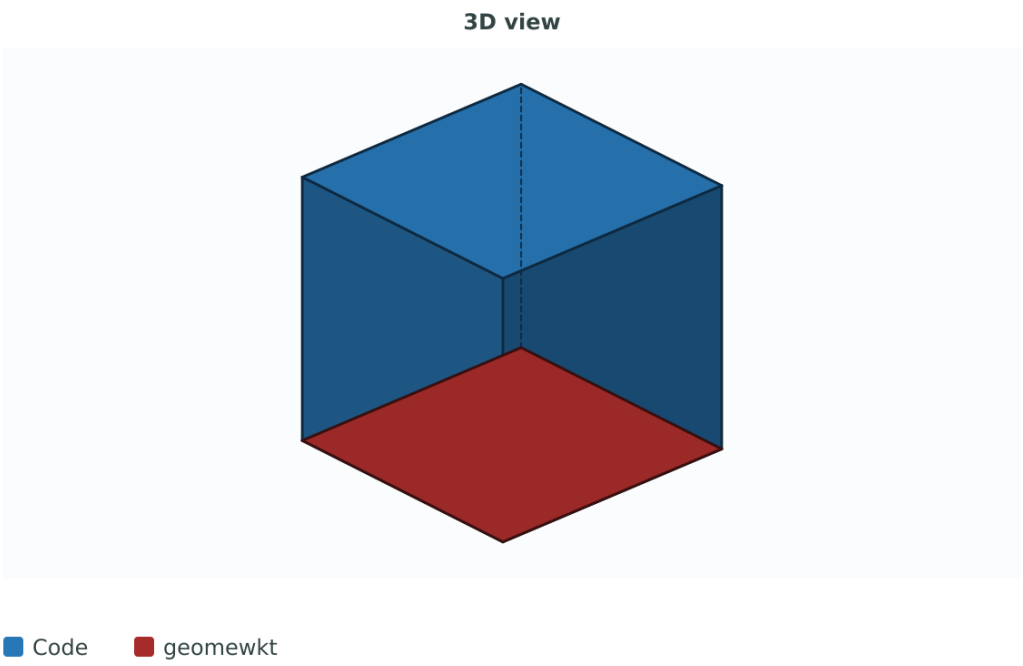
Code

```
SELECT ST_PatchN(geom, 2) As geomewkt
FROM (
VALUES ('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry) ) As ↩
    foo(geom);
```

Output

```
POLYGON((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0))
```

Figure



Se även

ST_AsEWKT, ST_GeomFromEWKT, ST_Dump, ST_GeometryN, ST_NumGeometries, ST_NumPatches

7.4.37 ST_PointN

ST_PointN — Returnerar den N:te punkten i den första LineString eller cirkulära LineString i en geometri.

Synopsis

geometry **ST_PointN**(geometry a_linestring, integer n);

Beskrivning

Returnerar den N:te punkten i en enkel linestrings eller cirkulär linestrings i geometrin. Negativa värden räknas baklänges från slutet av LineString, så att -1 är den sista punkten. Returnerar NULL om det inte finns någon linestrings i geometrin.

Notera
Index is 1-based as for OGC specs since version 0.8.0. Backward indexing (negative index) is not in OGC. Previous versions implemented this as 0-based instead.

Notera
Om du vill få fram den N:te punkten i varje LineString i en MultiLineString, använd tillsammans med ST_Dump

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.2.5, 7.3.5

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Ändrad: 2.0.0 fungerar inte längre med multilinestrings med en geometri. I äldre versioner av PostGIS -- skulle en multilinestring med en enda linje fungera bra med den här funktionen och returnera startpunkten. I 2.0.0 returnerar den bara NULL som vilken annan multilinjesträng som helst.

Ändrad: 2.3.0 : negativ indexering tillgänglig (-1 är sista punkten)

Exempel

This example extracts all POINT geometries from a LineString.

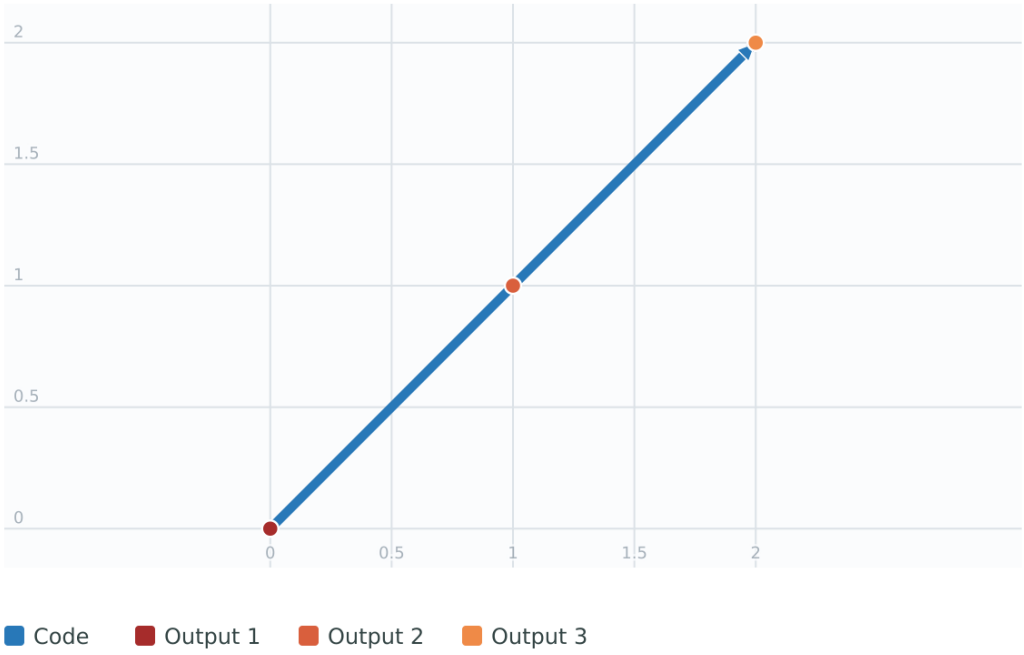
Code

```
SELECT ST_PointN(
      column1,
      generate_series(1, ST_NPoints(column1))
   )
FROM ( VALUES ('LINESTRING(0 0,1 1,2 2)'::geometry) ) AS foo;
```

Output

```
POINT(0 0)
 POINT(1 1)
 POINT(2 2)
(3 rows)
```

Figure



This example uses a CircularString.

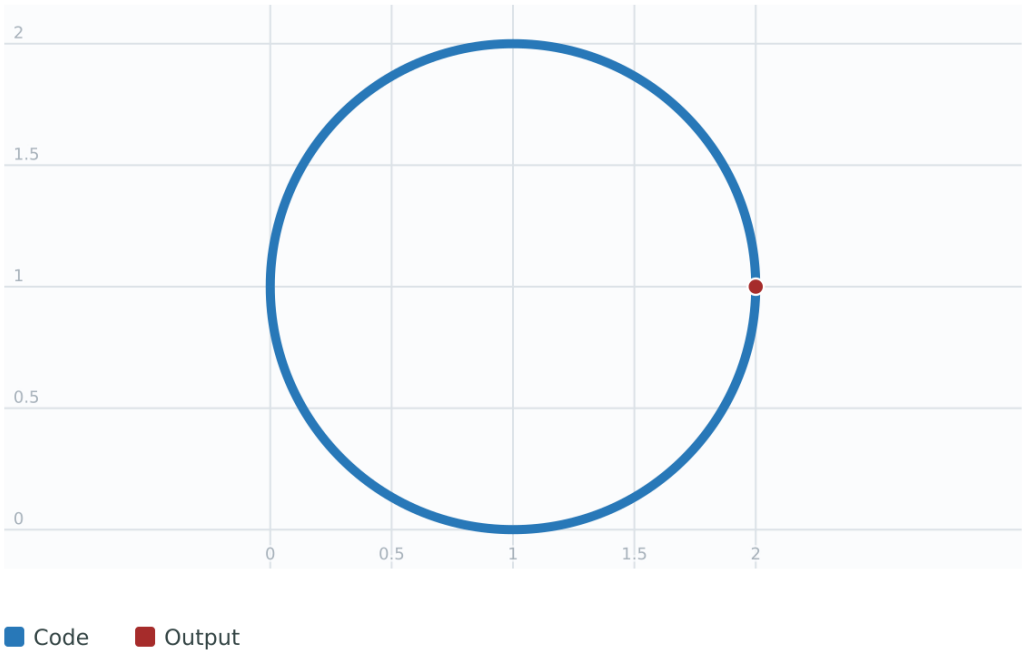
Code

```
SELECT ST_PointN('CIRCULARSTRING(1 2,3 2,1 2)', 2);
```

Output

```
POINT(3 2)
```

Figure



The index is 1-based. A negative index counts backward from the end of the LineString.

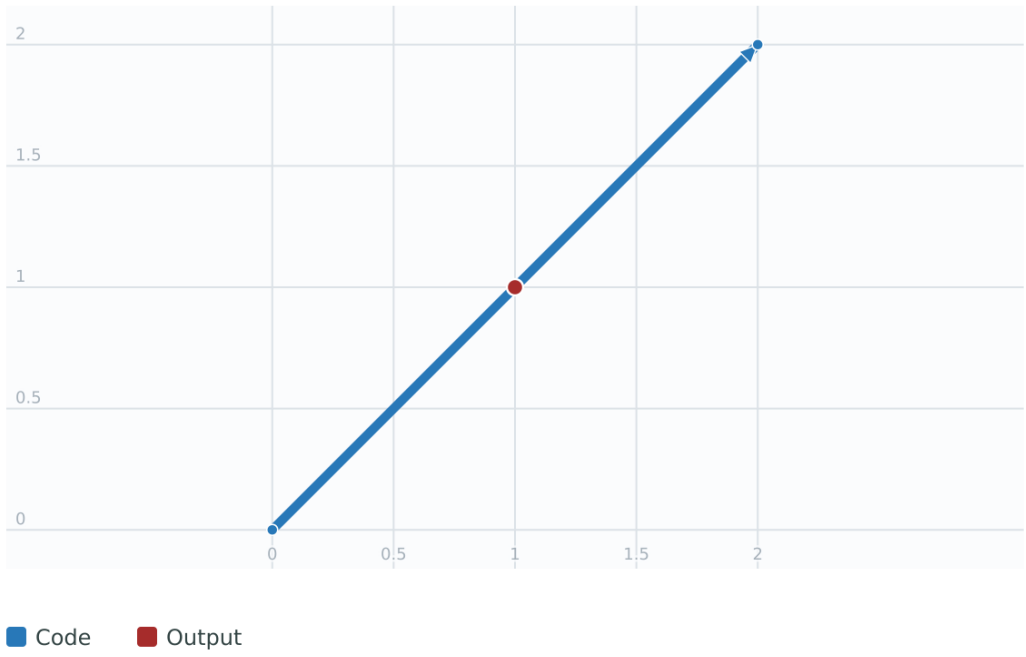
Code

```
SELECT f
FROM ST_GeomFromText('LINESTRING(0 0 0,1 1 1,2 2 2)') AS g
  CROSS JOIN ST_PointN(g, -2) AS f;
```

Output

```
POINT Z (1 1 1)
```

Figure



Se även

ST_NPoints

7.4.38 ST_Points

ST_Points — Returnerar en MultiPoint som innehåller koordinaterna för en geometri.

Synopsis

geometry **ST_Points**(geometry geom);

Beskrivning

Returnerar en MultiPoint som innehåller alla koordinater för en geometri. Duplicerade punkter bevaras, inklusive start- och slutpunkterna för ringgeometrier. (Om så önskas kan duplicerade punkter tas bort genom att anropa ST_RemoveRepeatedPoints på resultatet).

För att få information om varje koordinats position i den överordnade geometrin använder du ST_DumpPoints

M- och Z-koordinater bevaras om de finns.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Tillgänglighet: 2.3.0

Exempel

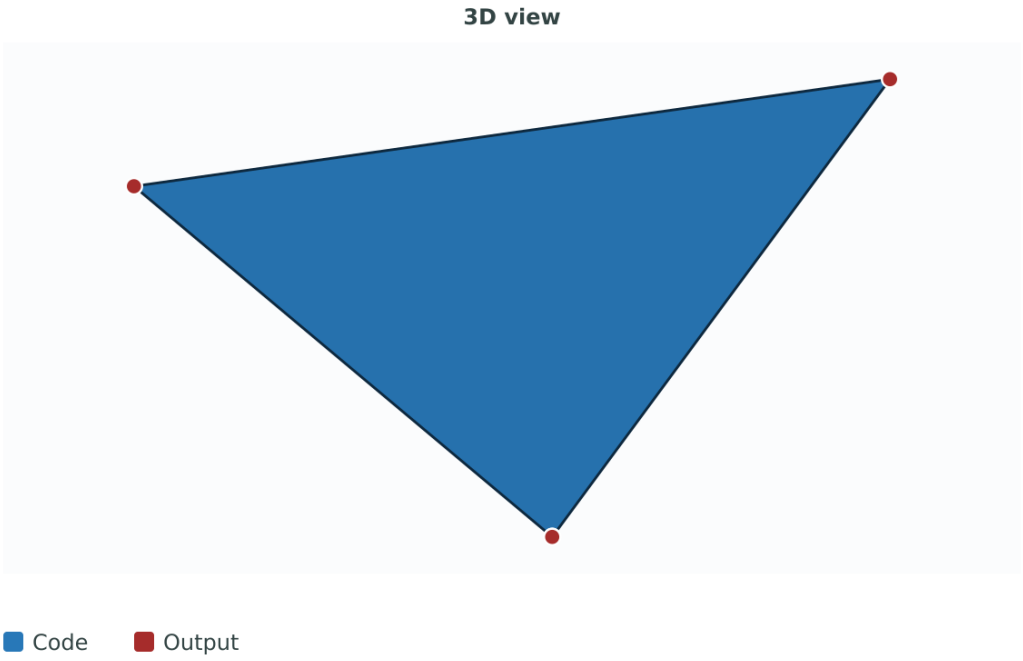
Code

```
SELECT ST_Points('POLYGON Z ((30 10 4,10 30 5,40 40 6,30 10 4))');
```

Output

```
MULTIPOINT Z ((30 10 4),(10 30 5),(40 40 6),(30 10 4))
```

Figure



Se även

ST_RemoveRepeatedPoints, ST_DumpPoints

7.4.39 ST_StartPoint

ST_StartPoint — Returns the first point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_StartPoint**(geometry geomA);

Beskrivning

Returns the first point of a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry as a `POINT`. For other geometries, returns the first point in coordinate order.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.1.3

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Förbättrad: 3.2.0 returnerar en punkt för alla geometrier. Tidigare beteende returnerade NULL om indata inte var en LineString.

Ändrad: 2.0.0 fungerar inte längre med MultiLineStrings med en geometri. I äldre versioner av PostGIS skulle en MultiLineString med en enda rad fungera bra med den här funktionen och returnera startpunkten. I 2.0.0 returnerar den bara NULL som vilken annan MultiLineString som helst. Det gamla beteendet var en odokumenterad funktion, men personer som antog att de hade sina data lagrade som LINESTRING kan uppleva att dessa returnerar NULL i 2.0.0.

Exempel

Startpunkt för en LineString

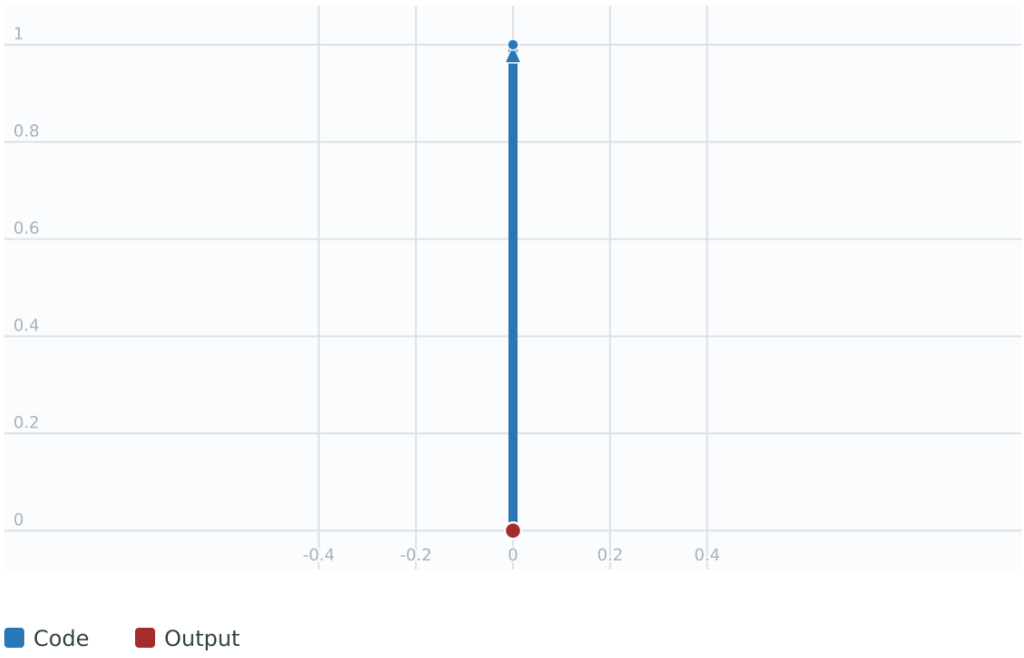
Code

```
SELECT ST_StartPoint('LINESTRING(0 1,0 2)'::geometry);
```

Output

```
POINT(0 1)
```

Figure



Start point of a Point is the Point itself

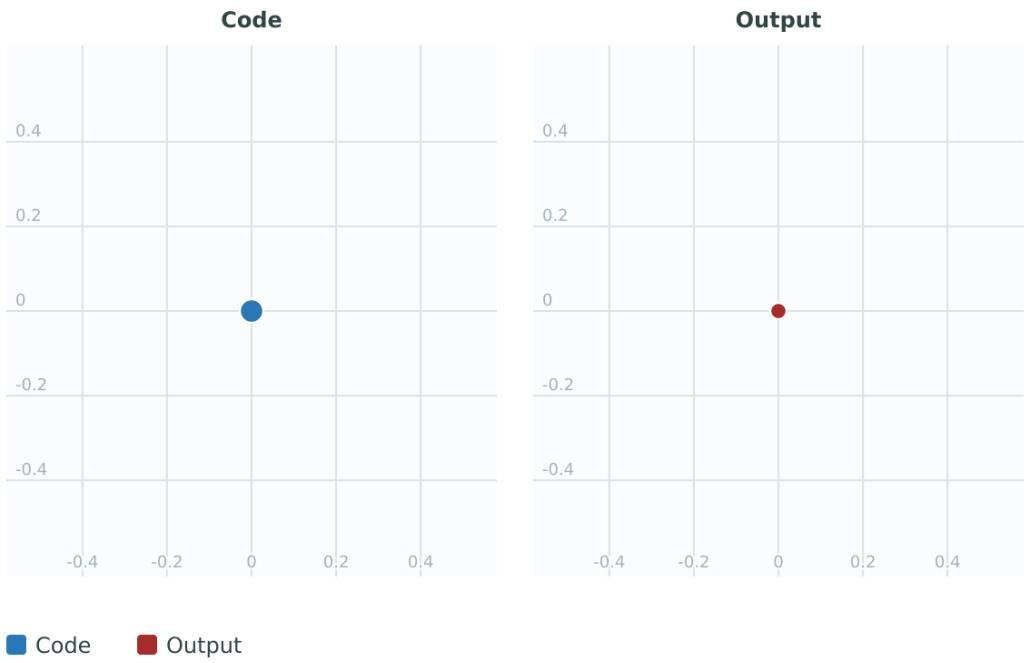
Code

```
SELECT ST_StartPoint('POINT(0 1)'::geometry);
```

Output

```
POINT(0 1)
```

Figure



Startpunkt för en 3D-linjeString

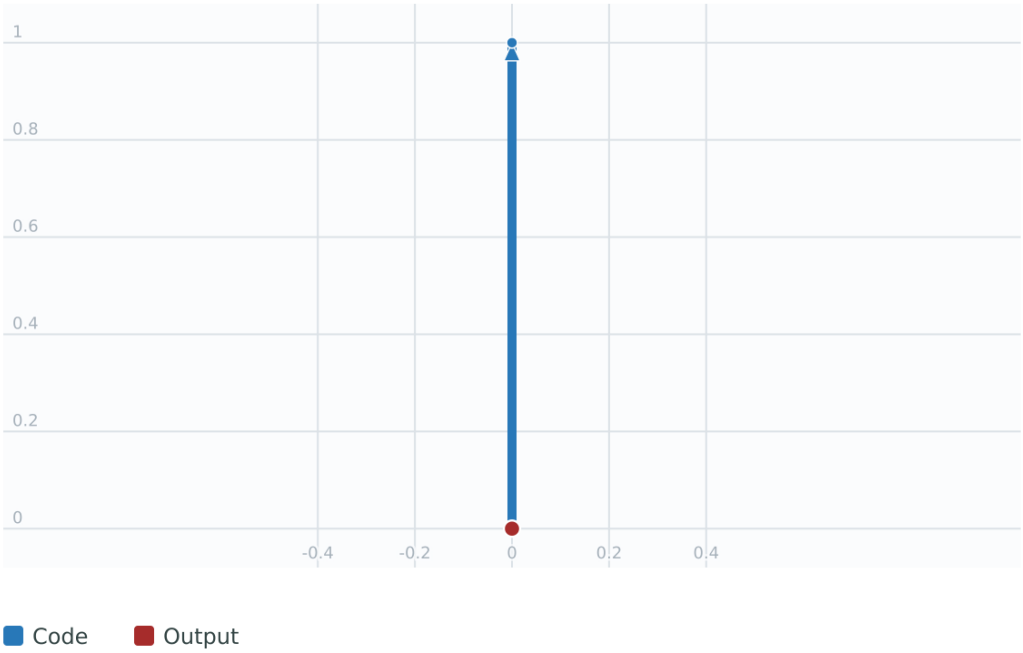
Code

```
SELECT ST_StartPoint('LINESTRING(0 1 1,0 2 2)'::geometry);
```

Output

```
POINT(0 1 1)
```

Figure



Startpunkt för en CircularString

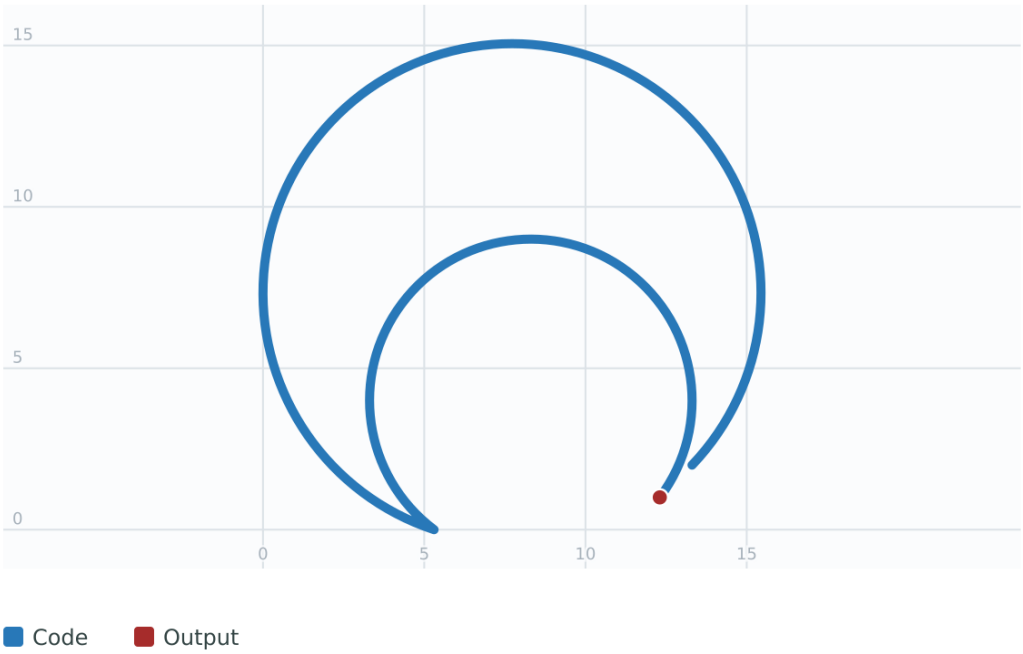
Code

```
SELECT ST_StartPoint('CIRCULARSTRING(5 2,-3 2,-2 1,-4 2,6 3)'::geometry);
```

Output

```
POINT(5 2)
```

Figure



Start point of a NURBSCurve

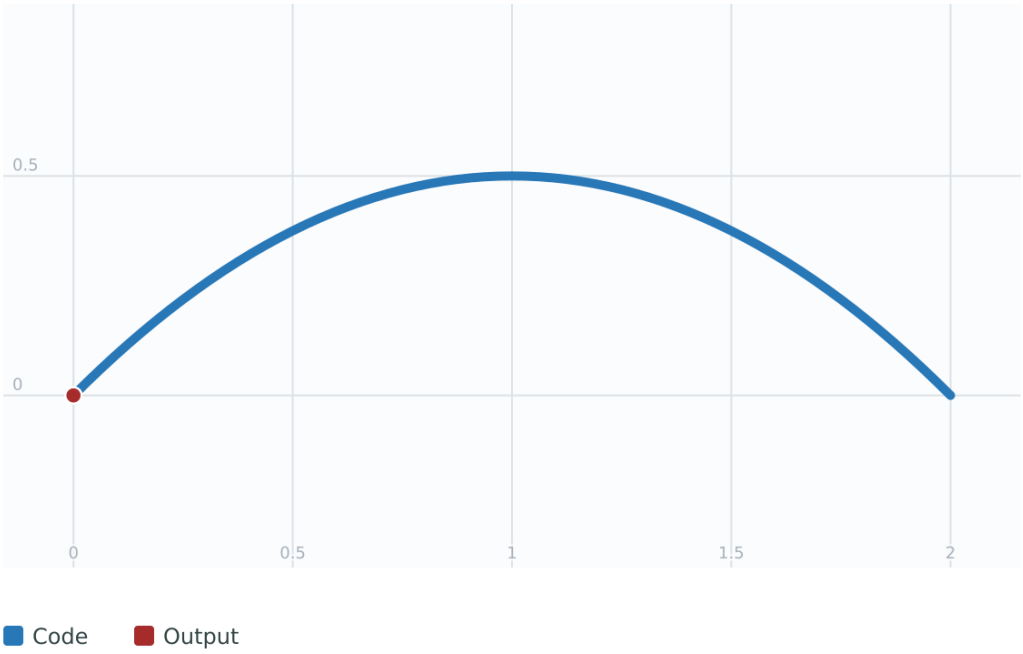
Code

```
SELECT ST_StartPoint('NURBSCURVE(2, (0 0, 1 1, 2 0))'::geometry);
```

Output

```
PUNKT(0 0)
```

Figure



Build edge node references from the start and end points of a LineString table. The example uses exact endpoint equality, so linework with nearly coincident endpoints should be snapped or otherwise cleaned first.

Code

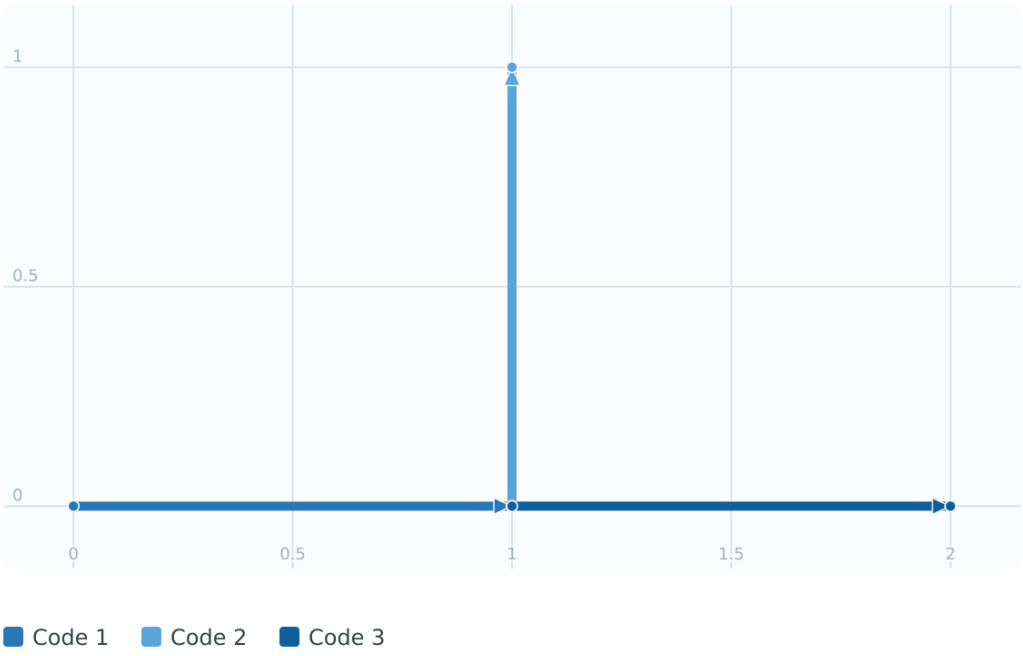
```
WITH roads(road_id, geom) AS (
  VALUES
    (1, 'LINESTRING(0 0, 1 0)'::geometry),
    (2, 'LINESTRING(1 0, 1 1)'::geometry),
    (3, 'LINESTRING(1 0, 2 0)'::geometry)
),
endpoints AS (
  SELECT road_id, 'from' AS end_name, ST_StartPoint(geom) AS geom,
         ST_AsEWKB(ST_StartPoint(geom)) AS endpoint_key
  FROM roads
  UNION ALL
  SELECT road_id, 'to' AS end_name, ST_EndPoint(geom) AS geom,
         ST_AsEWKB(ST_EndPoint(geom)) AS endpoint_key
  FROM roads
),
nodes AS (
  SELECT dense_rank() OVER (ORDER BY ST_X(geom), ST_Y(geom), endpoint_key) AS node_id,
         endpoint_key
  FROM (
    SELECT DISTINCT ON (endpoint_key) endpoint_key, geom
    FROM endpoints
    ORDER BY endpoint_key
```

```
  ) AS distinct_endpoints
)
SELECT e.road_id,
       max(n.node_id) FILTER (WHERE e.end_name = 'from') AS from_node,
       max(n.node_id) FILTER (WHERE e.end_name = 'to') AS to_node
FROM endpoints AS e
JOIN nodes AS n USING (endpoint_key)
GROUP BY e.road_id
ORDER BY e.road_id;
```

Output

```
road_id | from_node | to_node
---------+-----------+---------
     1 |         1 |       2
     2 |         2 |       3
     3 |         2 |       4
```

Figure



Se även

ST_EndPoint, ST_PointN

7.4.40 ST_StartM

ST_StartM — Returns the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

float8 **ST_StartM**(geometry geom);

Beskrivning

Returns the M coordinate of the first point of a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry. Returns `NULL` if the input geometry does not have an M dimension or is not a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE`.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Start M value of a LineString

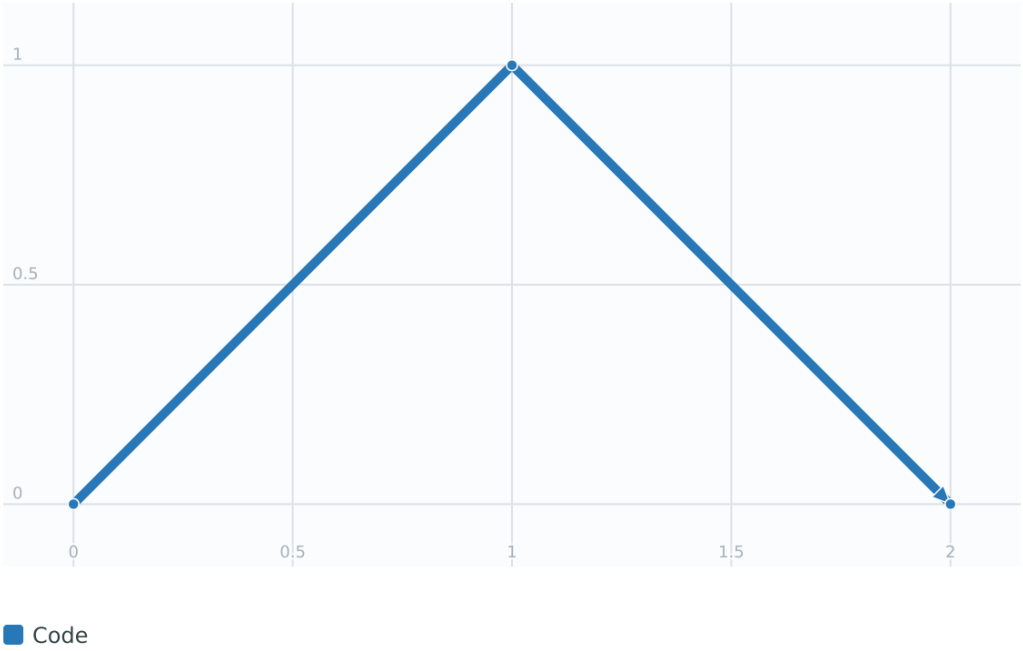
Code

```
SELECT ST_StartM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry);
```

Output

```
10
```

Figure



Start M value of a CircularString

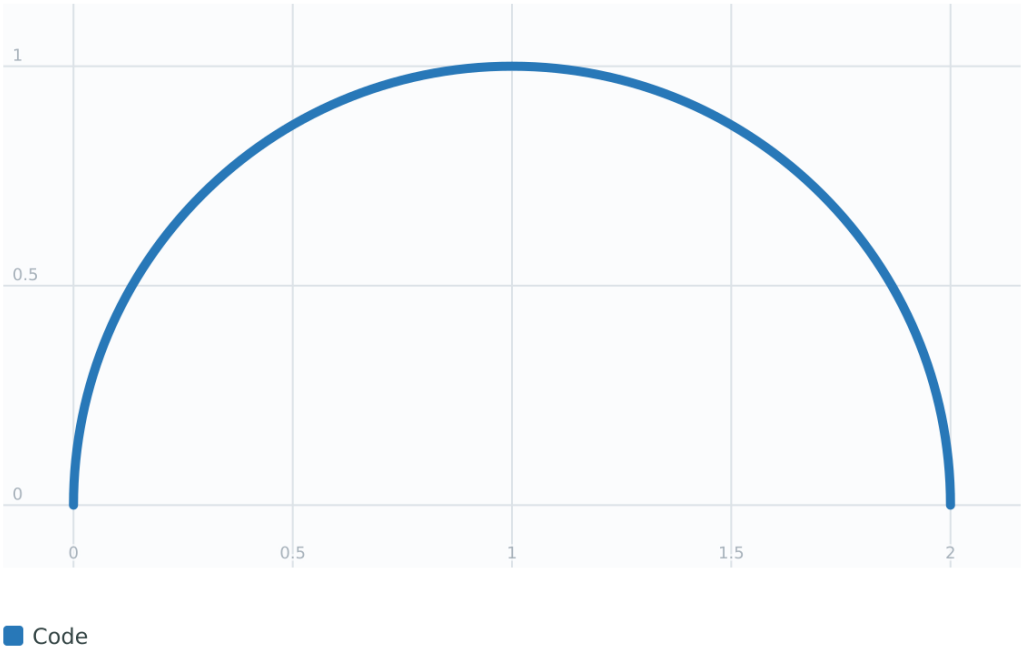
Code

```
SELECT ST_StartM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry);
```

Output

```
5
```

Figure



Start M value of a NURBSCurve

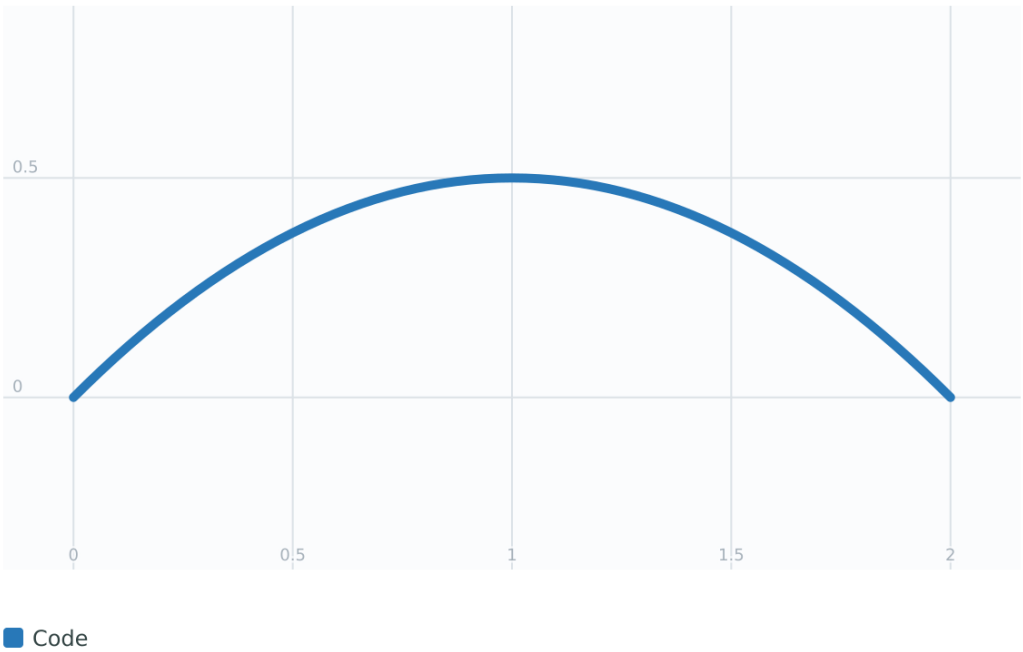
Code

```
SELECT ST_StartM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry);
```

Output

```
100
```

Figure



Geometry without M dimension returns NULL

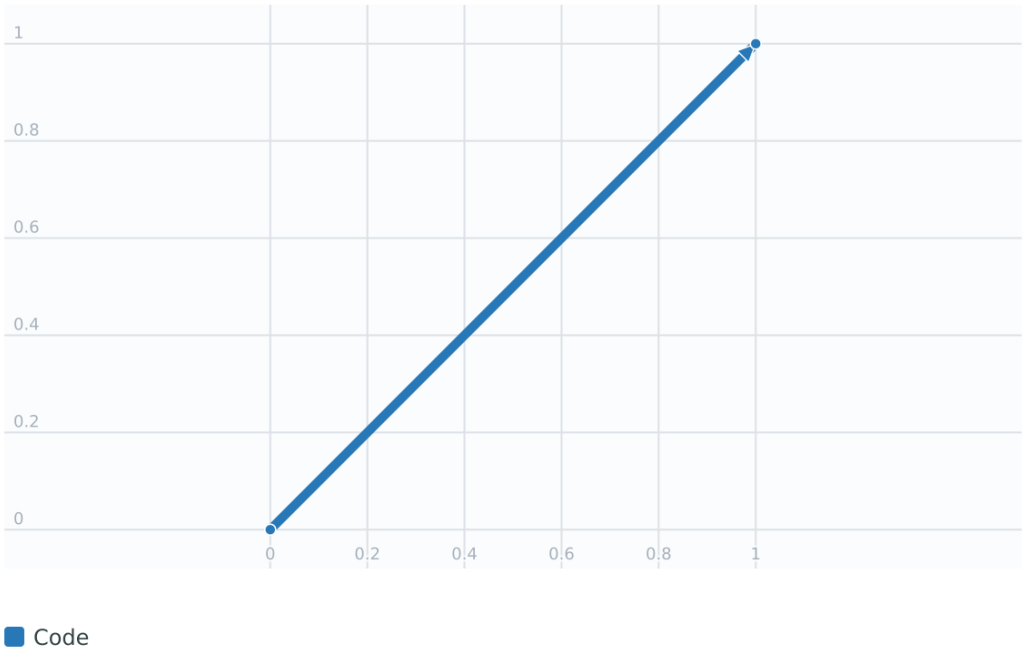
Code

```
SELECT ST_StartM('LINESTRING(0 0,1 1)'::geometry) IS NULL AS is_null;
```

Output

```
t
```

Figure



Se även

ST_EndM, ST_SetStartM, ST_StartPoint

7.4.41 ST_EndM

ST_EndM — Returns the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

float8 **ST_EndM**(geometry geom);

Beskrivning

Returns the M coordinate of the last point of a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry. Returns `NULL` if the input geometry does not have an M dimension or is not a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE`.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

End M value of a LineString

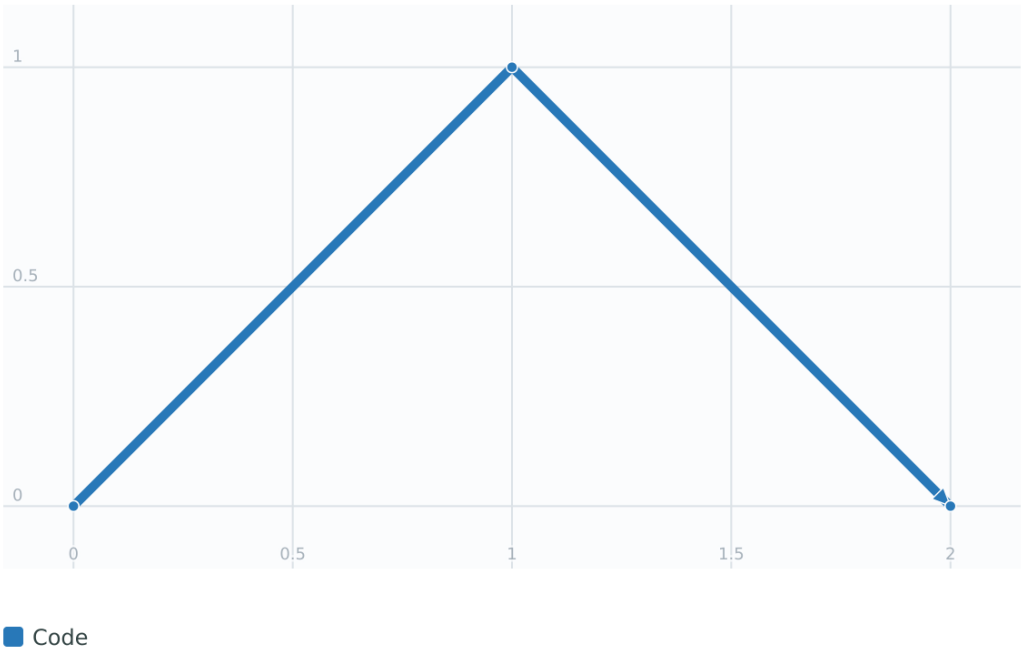
Code

```
SELECT ST_EndM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry);
```

Output

```
30
```

Figure



End M value of a CircularString

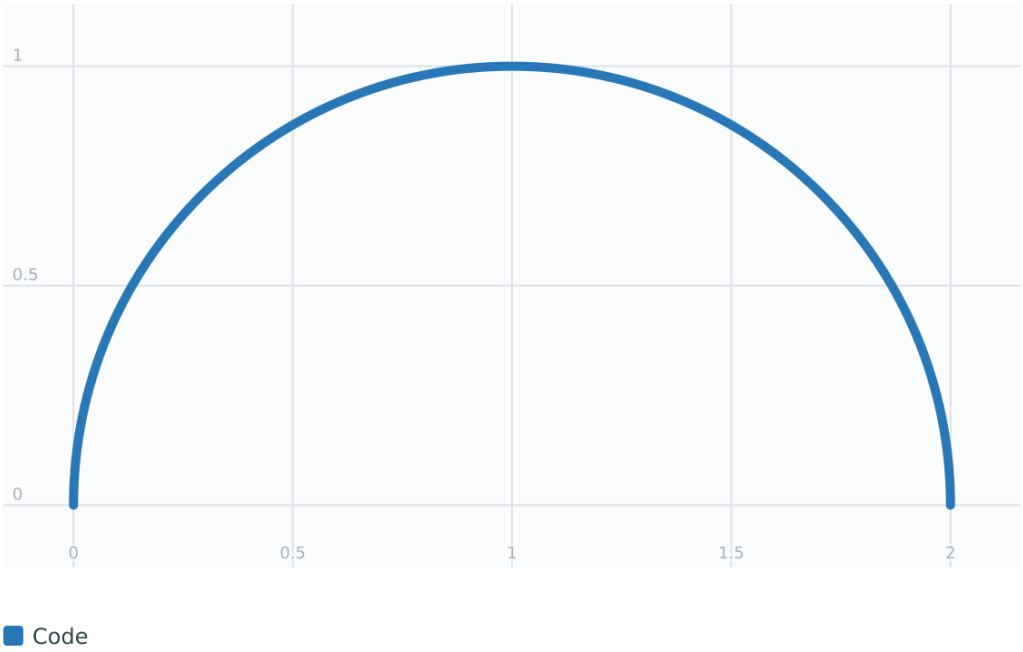
Code

```
SELECT ST_EndM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry);
```

Output

```
15
```

Figure



End M value of a NURBSCurve

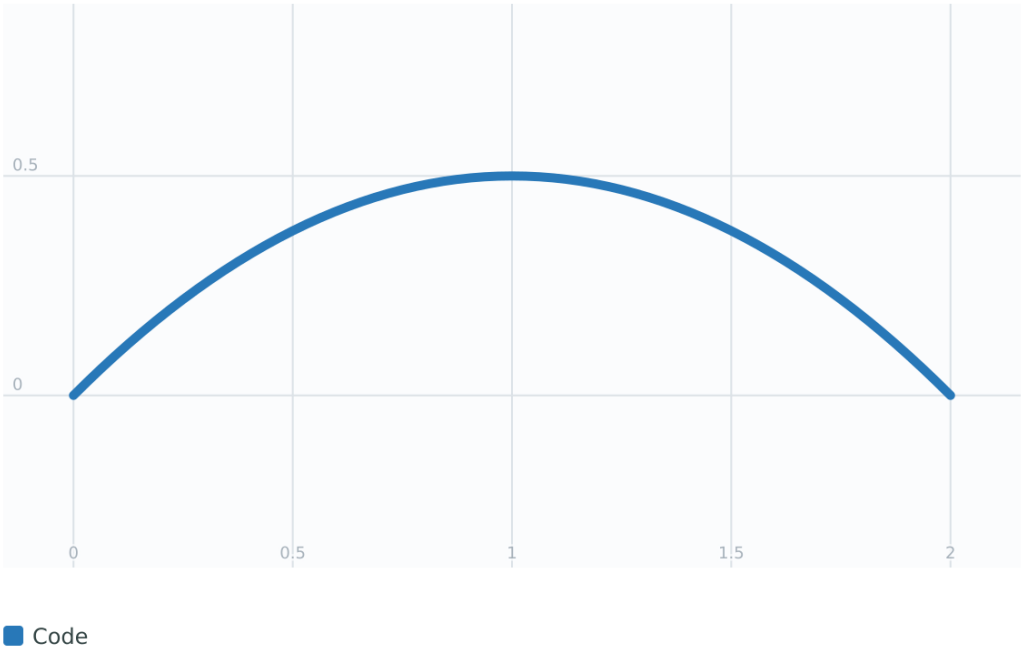
Code

```
SELECT ST_EndM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry);
```

Output

```
300
```

Figure



Geometry without M dimension returns NULL

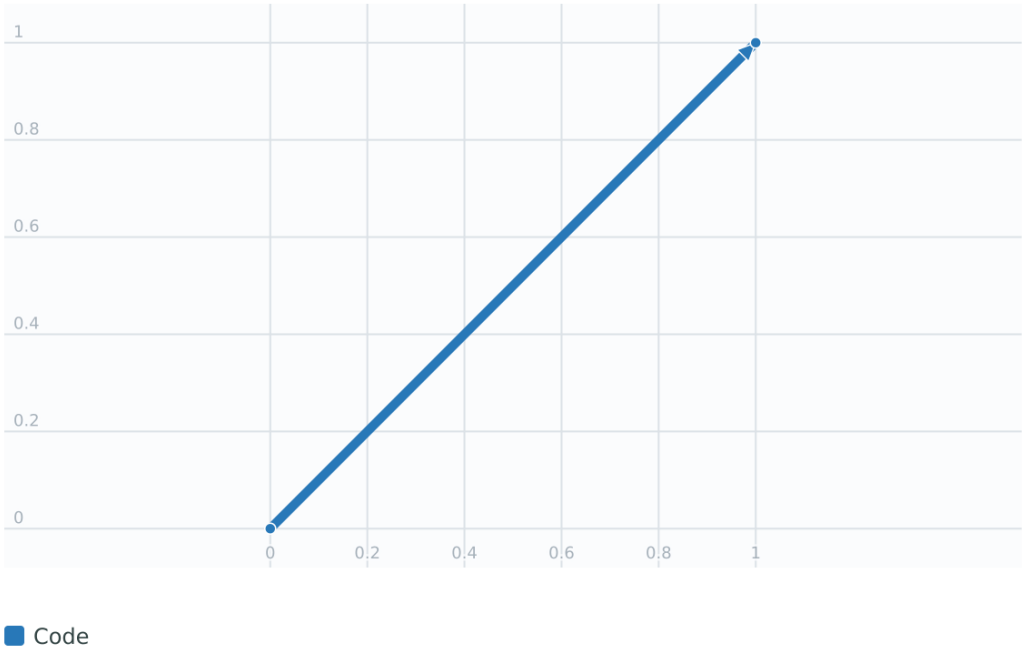
Code

```
SELECT ST_EndM('LINESTRING(0 0,1 1)'::geometry) IS NULL AS is_null;
```

Output

```
t
```

Figure



Se även

ST_StartM, ST_SetEndM, ST_EndPoint

7.4.42 ST_SetStartM

ST_SetStartM — Sets the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_SetStartM**(geometry geom, float8 m);

Beskrivning

Returns a copy of the input `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry with the M coordinate of the first point set to the specified value. If the input geometry does not have an M dimension, it will be added. Returns `NULL` if the input is not a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE`.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Set start M value of a LineString

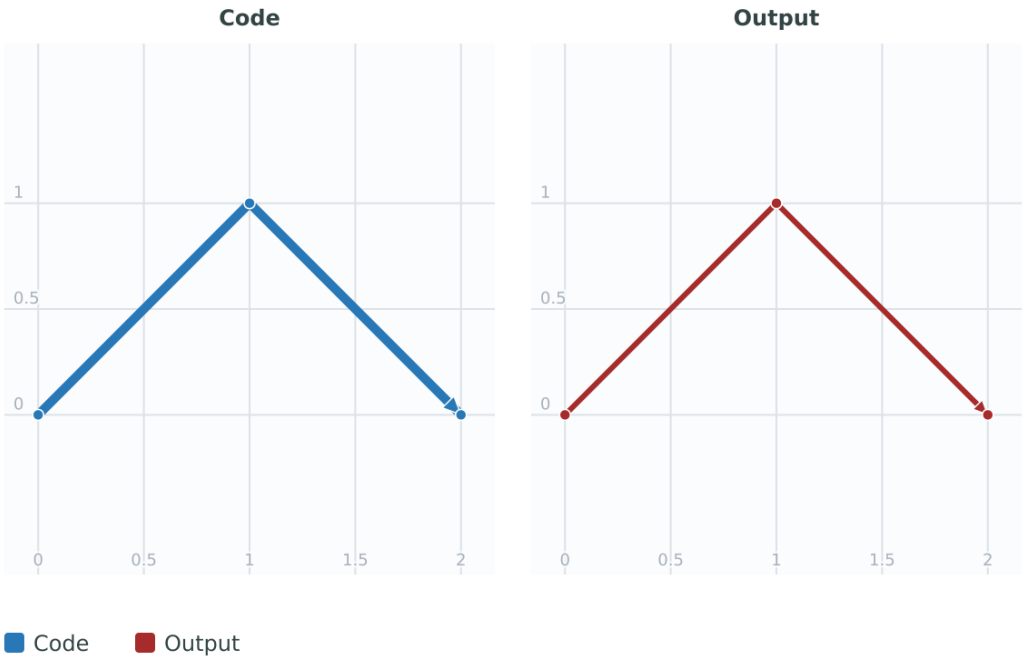
Code

```
SELECT ST_SetStartM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry, 100);
```

Output

```
LINESTRING M (0 0 100,1 1 20,2 0 30)
```

Figure



Add M dimension and set start M value

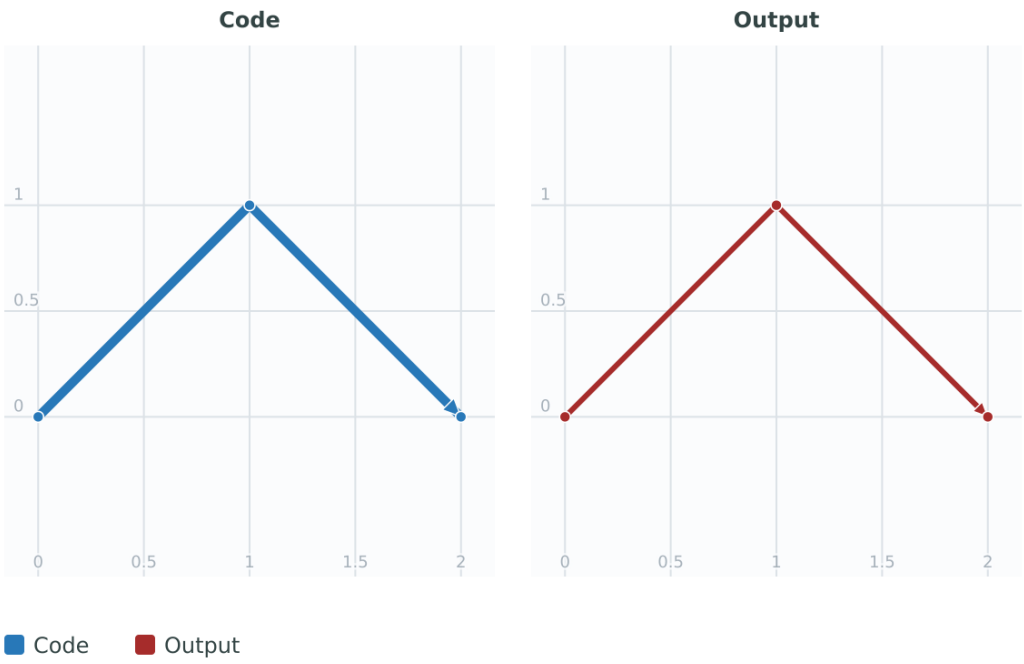
Code

```
SELECT ST_SetStartM('LINESTRING(0 0,1 1,2 0)'::geometry, 50);
```

Output

```
LINESTRING M (0 0 50,1 1 0,2 0 0)
```

Figure



Set start M value of a CircularString

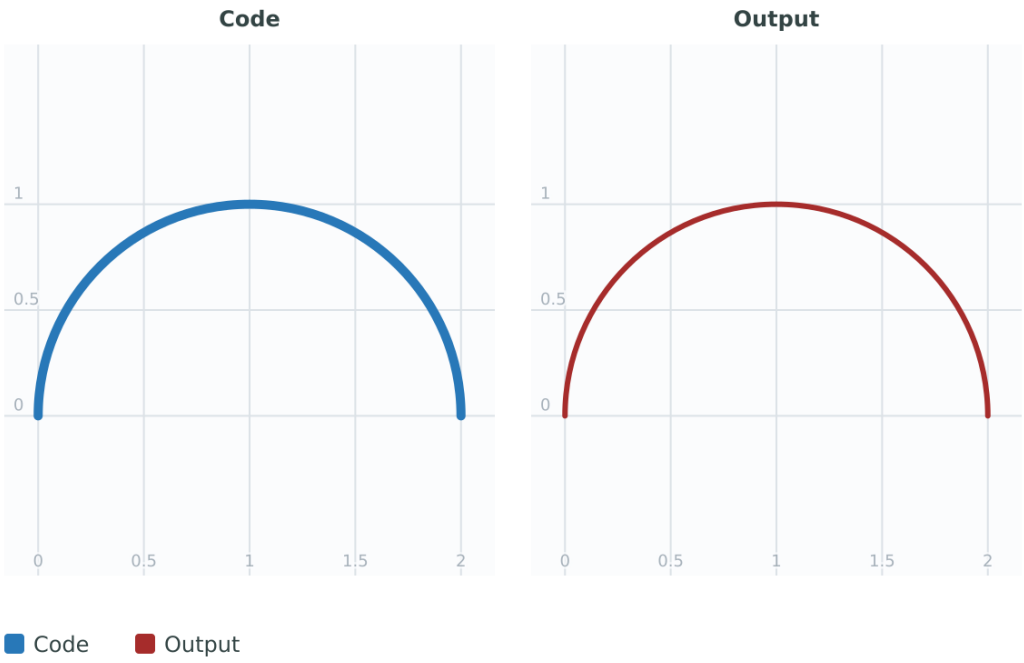
Code

```
SELECT ST_SetStartM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry, 99);
```

Output

```
CIRCULARSTRING M (0 0 99,1 1 10,2 0 15)
```

Figure



Set start M value of a NURBSCurve

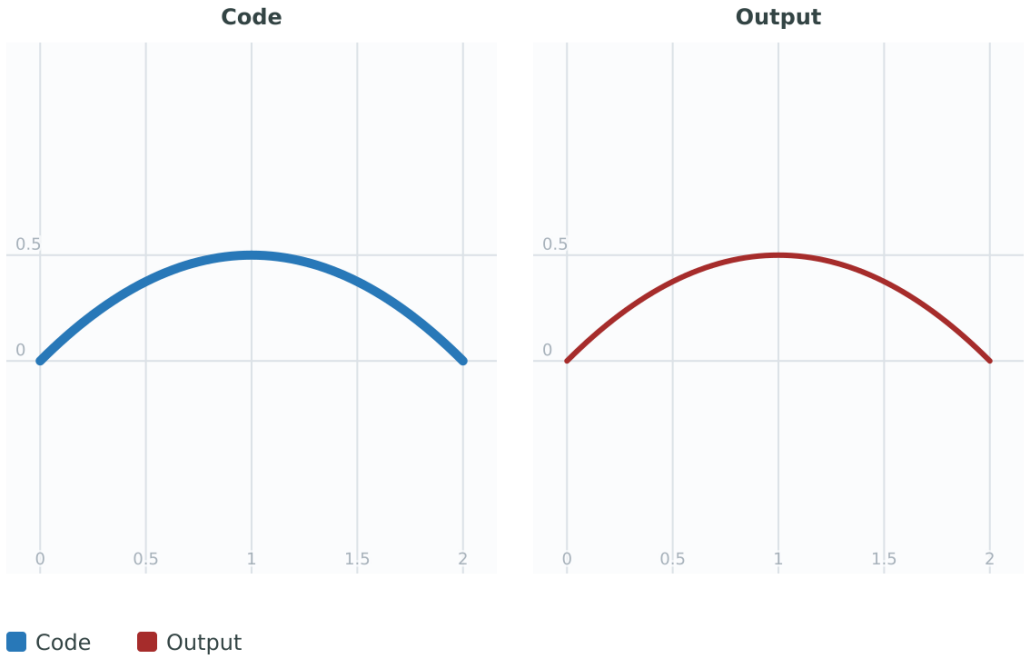
Code

```
SELECT ST_SetStartM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry, 500);
```

Output

```
NURBSCURVE M (DEGREE 2,CONTROLPOINTS M (NURBSPOINT(WEIGHTEDPOINT M (0 0 500),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (1 ↩
    1 200),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (2 0 300),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



Se även

ST_SetEndM, ST_StartM, ST_StartPoint

7.4.43 ST_SetEndM

ST_SetEndM — Sets the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_SetEndM**(geometry geom, float8 m);

Beskrivning

Returns a copy of the input `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry with the M coordinate of the last point set to the specified value. If the input geometry does not have an M dimension, it will be added. Returns `NULL` if the input is not a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE`.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Set end M value of a LineString

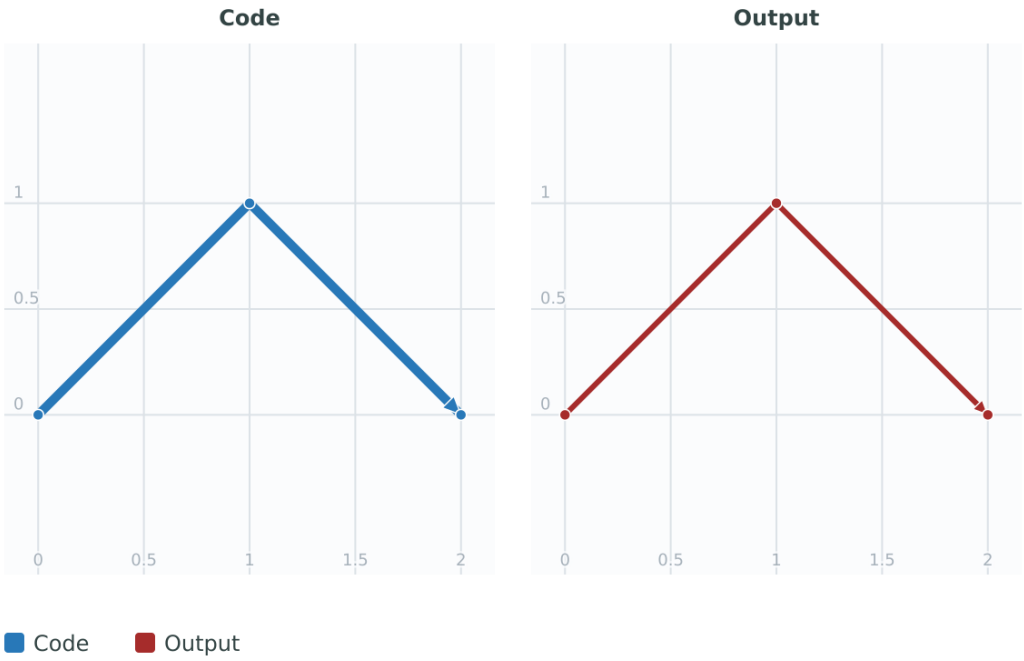
Code

```
SELECT ST_SetEndM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry, 999);
```

Output

```
LINESTRING M (0 0 10,1 1 20,2 0 999)
```

Figure



Add M dimension and set end M value

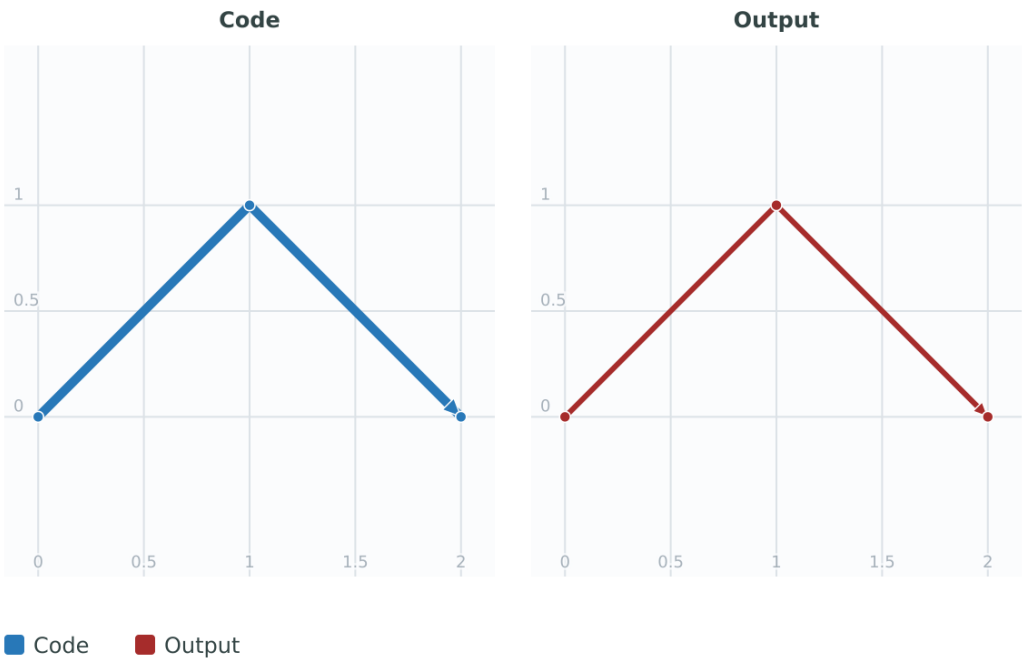
Code

```
SELECT ST_SetEndM('LINESTRING(0 0,1 1,2 0)'::geometry, 75);
```

Output

```
LINESTRING M (0 0 0,1 1 0,2 0 75)
```

Figure



Set end M value of a CircularString

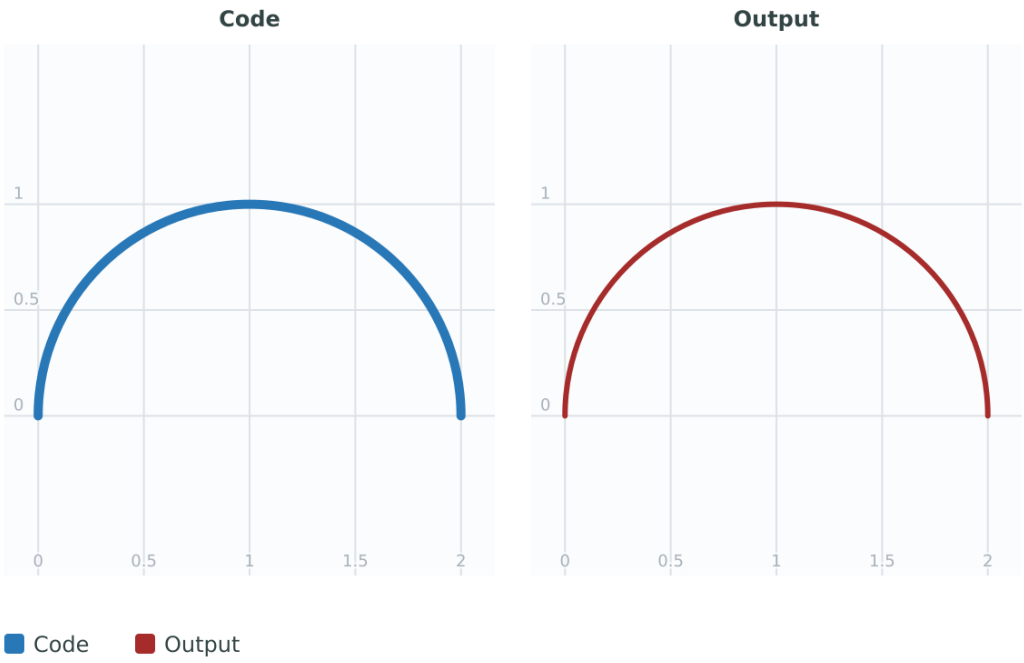
Code

```
SELECT ST_SetEndM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry, 88);
```

Output

```
CIRCULARSTRING M (0 0 5,1 1 10,2 0 88)
```

Figure



Set end M value of a NURBSCurve

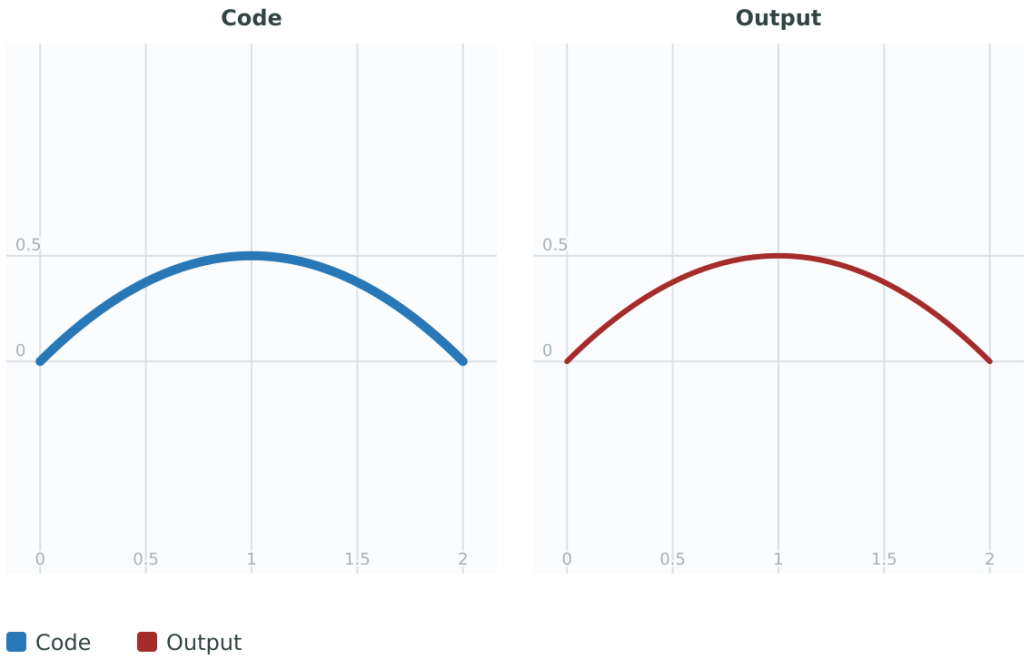
Code

```
SELECT ST_SetEndM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry, 999);
```

Output

```
NURBSCURVE M (DEGREE 2,CONTROLPOINTS M (NURBSPOINT(WEIGHTEDPOINT M (0 0 100),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (1
    1 200),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (2 0 999),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



Se även

ST_SetStartM, ST_EndM, ST_EndPoint

7.4.44 ST_Summary

ST_Summary — Returnerar en textsammanfattning av innehållet i en geometri.

Synopsis

text **ST_Summary**(geometry g);
text **ST_Summary**(geography g);

Beskrivning

Returnerar en textsammanfattning av innehållet i geometrin.

Flaggor som visas inom hakparentes efter geometritypen har följande betydelse:

- M: har M-koordinat

- Z: har Z-koordinat
- B: har en cachad avgränsande box
- G: är geodetisk (geografi)
- S: har ett spatialt referenssystem

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Tillgänglighet: 1.2.2

Förbättrad: 2.0.0 har lagt till stöd för geografi

Utökad: 2.1.0 S-flagga för att ange om det finns ett känt spatialt referenssystem

Förbättrad: 2.2.0 Lagt till stöd för TIN och kurvor

Exempel

Code

```
SELECT ST_Summary('LINESTRING(0 0,1 1)'::geometry) as geom,
ST_Summary('POLYGON((0 0,1 1,1 2,1 1,0 0))'::geography) geog;
```

Output

```
geom              |          geog
------------------------------+--------------------------
 LineString[B] with 2 points | Polygon[BGS] with 1 rings
                              | ring 0 has 5 points
                              :
(1 row)
```

Code

```
SELECT ST_Summary('LINESTRING(0 0 1,1 1 1)'::geography) As geog_line,
ST_Summary('SRID=4326;POLYGON((0 0 1,1 1 2,1 2 3,1 1 1,0 0 1))'::geometry) As geom_poly;
```

Output

```
geog_line             |        geom_poly
------------------------------- +--------------------------
 LineString[ZBGS] with 2 points | Polygon[ZBS] with 1 rings
                                :    ring 0 has 5 points
                                :
(1 row)
```

Se även

PostGIS_DropBBox, PostGIS_AddBBox, ST_Force3DM, ST_Force3DZ, ST_Force2D, geography
ST_IsValid, ST_IsValid, ST_IsValidReason, ST_IsValidDetail

7.4.45 ST_X

ST_X — Returnerar X-koordinaten för en Point.

Synopsis

float **ST_X**(geometry a_point);

Beskrivning

Returnerar punktens X-koordinat, eller NULL om den inte är tillgänglig. Indata måste vara en punkt.

Notera
För att få det minsta och största X-värdet för geometriska koordinater använder du funktionerna ST_XMin och ST_XMax.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 6.1.3

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_X('POINT(1 2 3 4)');
```

Output

```
1
```

Code

```
SELECT ST_Y(ST_Centroid('LINESTRING(1 2 3 4,1 1 1 1)'::geometry));
```

Output

```
1.5
```

Se även

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_XMax, ST_XMin, ST_Y, ST_Z

7.4.46 ST_Y

ST_Y — Returnerar Y-koordinaten för en Point.

Synopsis

float **ST_Y**(geometry a_point);

Beskrivning

Returnerar punktens Y-koordinat, eller NULL om den inte är tillgänglig. Indata måste vara en punkt.

Notera
För att få det lägsta och högsta Y-värdet för geometriska koordinater använder du funktionerna ST_YMin och ST_YMax.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 6.1.4

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_Y('POINT(1 2 3 4)');
```

Output

```
2
```

Code

```
SELECT ST_Y(ST_Centroid('LINESTRING(1 2 3 4,1 1 1 1)'::geometry));
```

Output

```
1.5
```

Se även

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_X, ST_YMax, ST_YMin, ST_Z

7.4.47 ST_Z

ST_Z — Returnerar Z-koordinaten för en Point.

Synopsis

float **ST_Z**(geometry a_point);

Beskrivning

Returnerar punktens Z-koordinat, eller NULL om den inte är tillgänglig. Indata måste vara en punkt.

Notera
För att få det lägsta och högsta Z-värdet för geometriska koordinater använder du funktionerna ST_ZMin och ST_ZMax.

Denna metod implementerar SQL/MM-specifikationen.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_Z('POINT(1 2 3 4)');
```

Output

```
3
```

Se även

ST_GeomFromEWKT, ST_M, ST_X, ST_Y, ST_ZMax, ST_ZMin

7.4.48 ST_ZMFlag

ST_ZMFlag — Returnerar en kod som anger ZM-koordinatdimensionen för en geometri.

Synopsis

smallint **ST_ZMFlag**(geometry geomA);

Beskrivning

Returnerar en kod som anger ZM-koordinatdimensionen för en geometri.
Värdena är: 0 = 2D, 1 = 3D-M, 2 = 3D-Z, 3 = 4D.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_ZMFlag('LINESTRING(1 2,3 4)');
```

Output

```
0
```

Code

```
SELECT ST_ZMFlag('LINESTRINGM(1 2 3, 3 4 3)');
```

Output

```
1
```

Code

```
SELECT ST_ZMFlag('CIRCULARSTRING(1 2 3,3 4 3,5 6 3)');
```

Output

```
2
```

Code

```
SELECT ST_ZMFlag('POINT(1 2 3 4)');
```

Output

```
3
```

Se även

ST_CoordDim, ST_NDims, ST_Dimension

7.4.49 ST_HasZ

ST_HasZ — Kontrollerar om en geometri har en Z-dimension.

Synopsis

boolean **ST_HasZ**(geometry geom);

Beskrivning

Kontrollerar om indatageometrin har en Z-dimension och returnerar ett booleskt värde. Om geometrin har en Z-dimension returneras true, annars returneras false.

Geometriobjekt med en Z-dimension representerar vanligtvis tredimensionella (3D) geometrier, medan de utan Z-dimension är tvådimensionella (2D) geometrier.

Denna funktion är användbar för att avgöra om en geometri har höjdinformation.

Tillgänglighet: 3.5.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Exempel

Code

```
SELECT ST_HasZ('POINT(1 2 3)');
```

Output

```
t
```

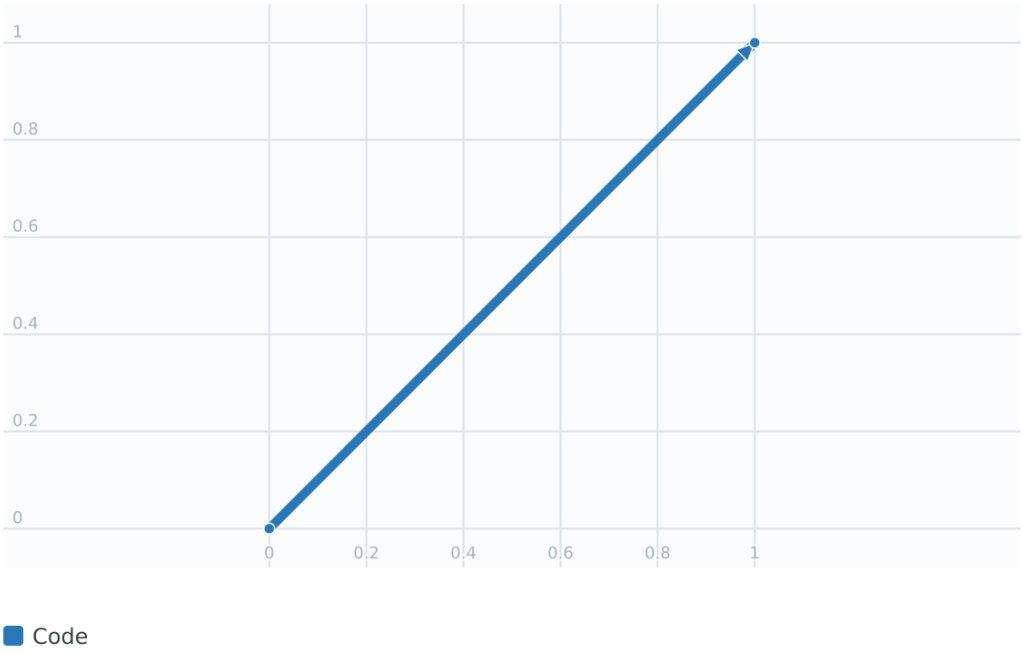
Code

```
SELECT ST_HasZ('LINESTRING(0 0,1 1)');
```

Output

```
f
```

Figure



Se även

ST_ZMFlag
ST_HasM

7.4.50 ST_HasM

ST_HasM — Kontrollerar om en geometri har en M (mått)-dimension.

Synopsis

boolean **ST_HasM**(geometry geom);

Beskrivning

Kontrollerar om indatageometrin har en M-dimension (mått) och returnerar ett booleskt värde. Om geometrin har en M-dimension returnerar den true, annars returnerar den false.

Geometriobjekt med en M-dimension representerar vanligtvis mätningar eller ytterligare data som är associerade med spatiala egenskaper.

Denna funktion är användbar för att avgöra om en geometri innehåller måttinformation.

Tillgänglighet: 3.5.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Exempel

Code

```
SELECT ST_HasM('POINTM(1 2 3)');
```

Output

```
t
```

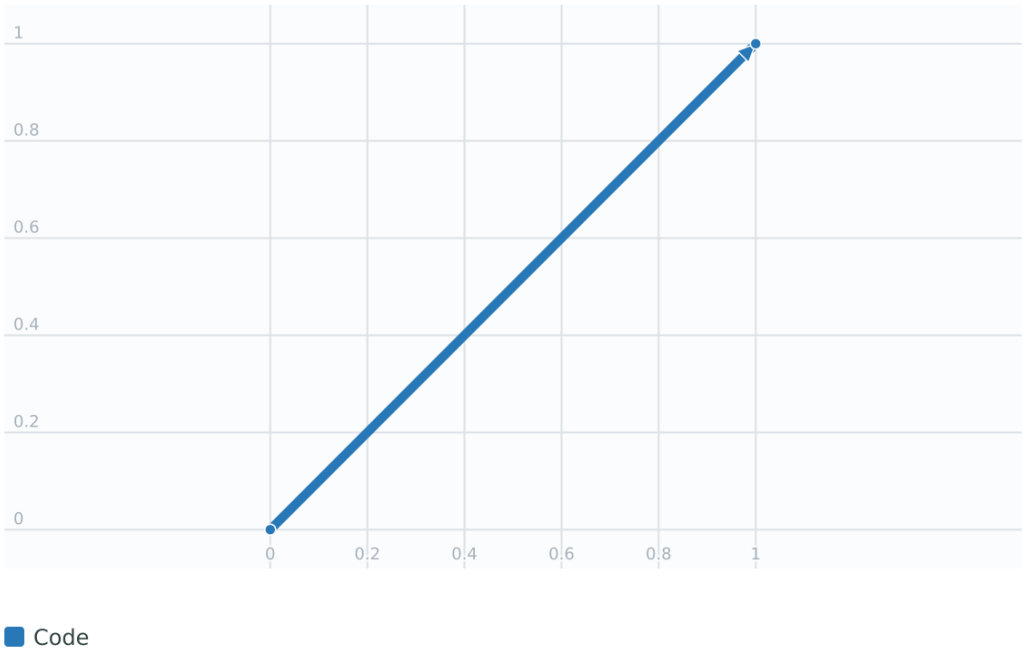
Code

```
SELECT ST_HasM('LINESTRING(0 0,1 1)');
```

Output

```
f
```

Figure



Se även

ST_ZMFlag
ST_HasZ

7.4.51 ST_ControlPoints

ST_ControlPoints — Returns the control points of a NURBS curve as a MULTIPOINT geometry.

Synopsis

geometry **ST_ControlPoints**(geometry nurbscurve);

Beskrivning

Extracts the control points from a NURBS curve and returns them as a MULTIPOINT geometry. Control points define the control polygon that influences the shape of the NURBS curve.

The function raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

Exempel

This example extracts control points from a NURBS curve.

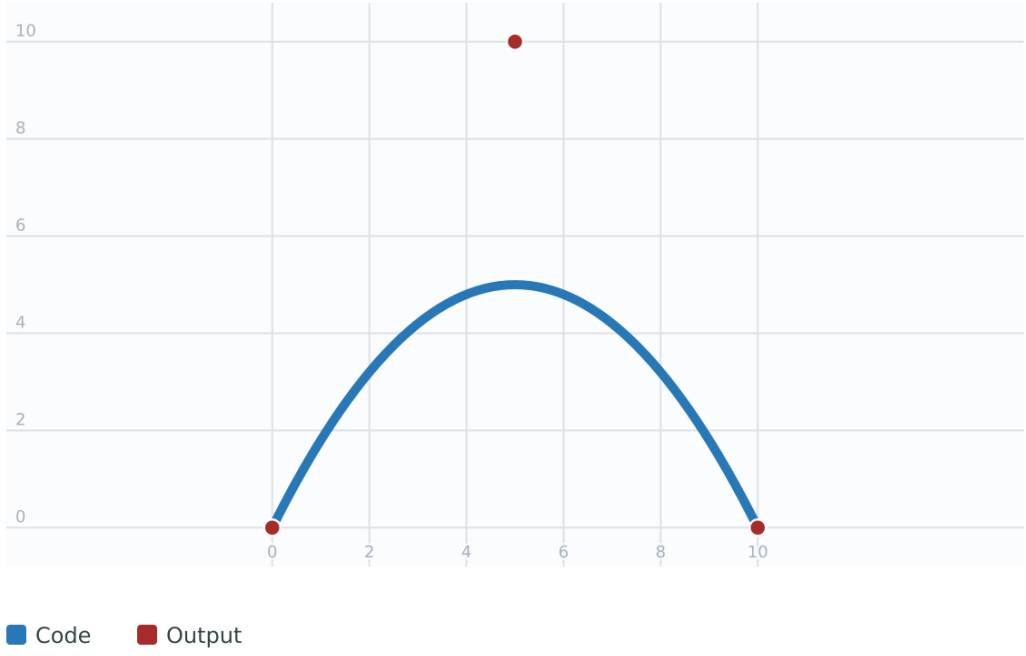
Code

```
SELECT ST_ControlPoints('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
MULTIPOINT((0 0),(5 10),(10 0))
```

Figure



This example counts the control points.

Code

```
SELECT ST_NPoints(ST_ControlPoints('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry));
```

Output

```
3
```

Se även

ST_MakeNurbsCurve, ST_NumControlPoints, ST_Weights

7.4.52 ST_Degree

ST_Degree — Returns the polynomial degree of a NURBS curve.

Synopsis

integer **ST_Degree**(geometry nurbscurve);

Beskrivning

Returns the polynomial degree of a NURBS curve as an integer. The degree determines the curve's smoothness and flexibility. Higher degrees produce smoother curves but require more control points.

The function raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

Exempel

This example returns the degree of a NURBS curve.

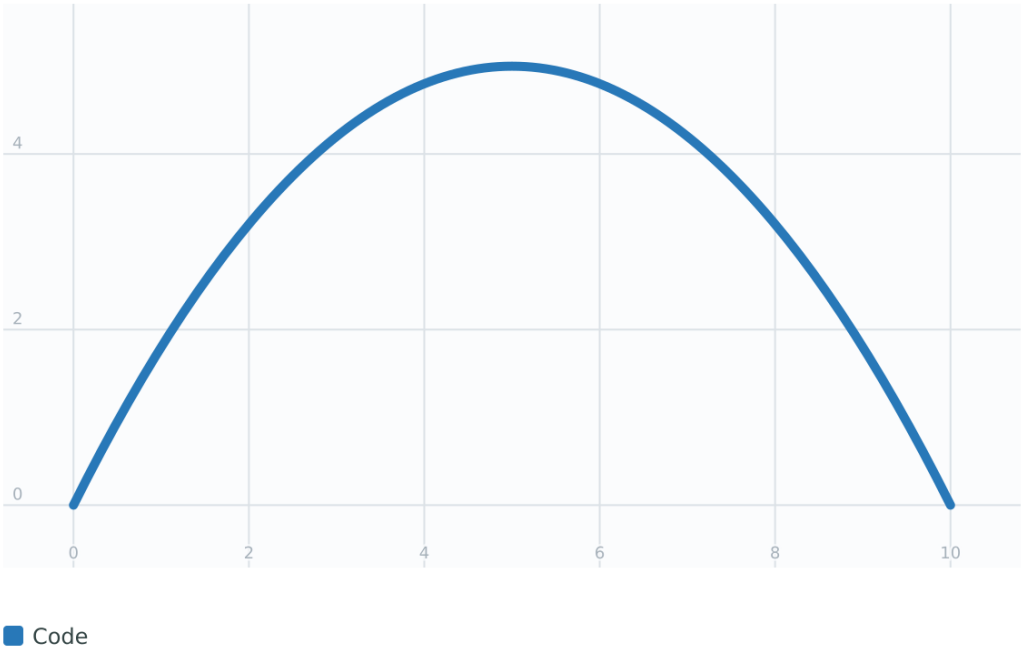
Code

```
SELECT ST_Degree('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
2
```

Figure



This example checks whether a NURBS curve is cubic.

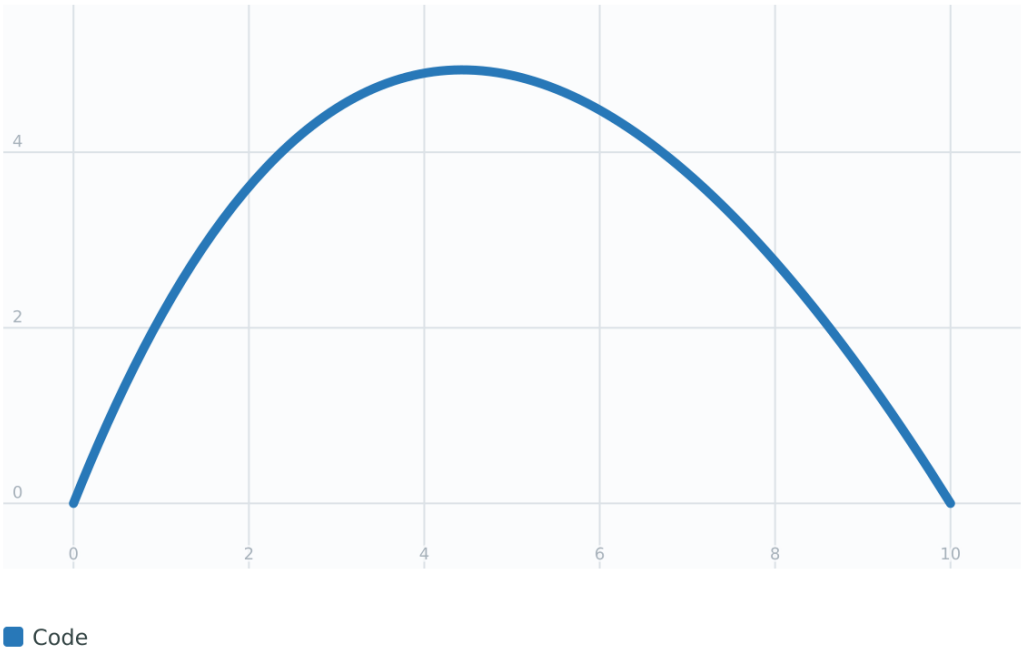
Code

```
SELECT ST_Degree('NURBSCURVE(3, (0 0, 2 5, 5 8, 10 0))'::geometry) = 3;
```

Output

```
t
```

Figure



Se även

ST_MakeNurbsCurve, ST_NumControlPoints

7.4.53 ST_Weights

ST_Weights — Returns the weight array of a rational NURBS curve.

Synopsis

float8[] **ST_Weights**(geometry nurbscurve);

Beskrivning

Returns the weight values of a rational NURBS curve as an array of double precision values. Weights control the influence of each control point on the curve shape. Higher weights pull the curve closer to the corresponding control point.

Returns NULL if the curve has no weights (non-rational B-spline) or if the input is NULL. Raises an error if the input is not a NURBS curve.

Availability: 3.7.0

Exempel

This example returns weights from a rational NURBS curve.

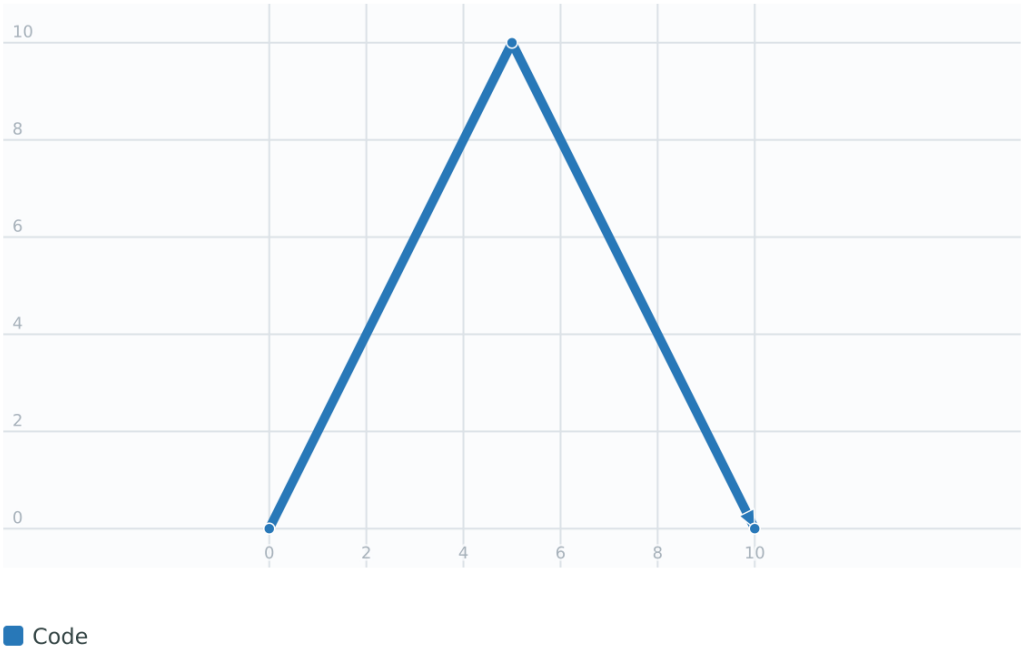
Code

```
SELECT ST_Weights(ST_MakeNurbsCurve(2, 'LINESTRING(0 0,5 10,10 0)'::geometry, ARRAY[1.0,  ↩
    2.0, 1.0]));
```

Output

```
{1,2,1}
```

Figure



This example checks the weights of a non-rational curve.

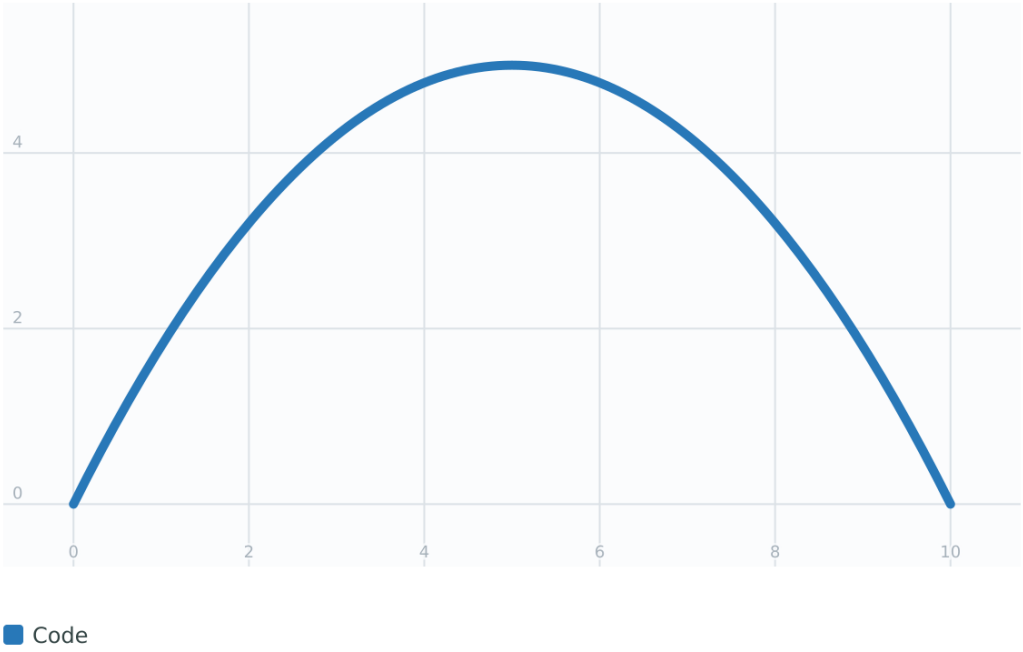
Code

```
SELECT ST_Weights('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
null
```

Figure



Se även

ST_MakeNurbsCurve, ST_NurbsCurveIsRational, ST_ControlPoints

7.4.54 ST_Knots

ST_Knots — Returns the knot vector of a NURBS curve.

Synopsis

float8[] **ST_Knots**(geometry nurbscurve);

Beskrivning

Returns the knot vector of a NURBS curve as an array of double precision values. The knot vector defines the parameter space and controls where and how the control points affect the curve. It must be a non-decreasing sequence with length equal to (number of control points + degree + 1).

Returns NULL if the curve uses the default uniform clamped knot vector or if the input is NULL. Raises an error if the input is not a NURBS curve.

Availability: 3.7.0

Exempel

This example returns knots from a NURBS curve with a custom knot vector.

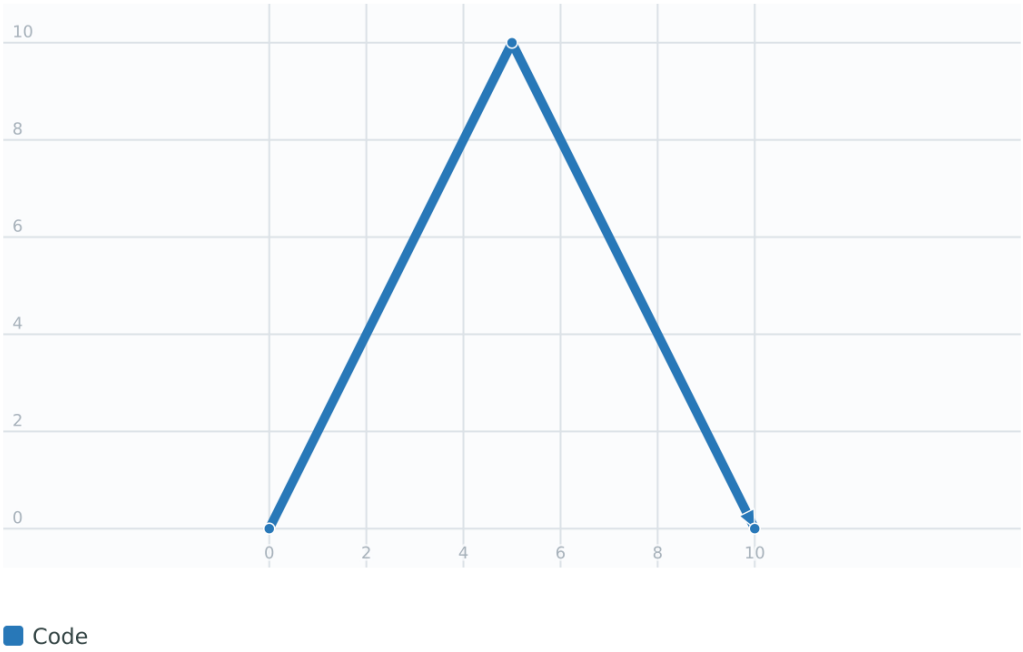
Code

```
SELECT ST_Knots(ST_MakeNurbsCurve(
        2,
        'LINESTRING(0 0,5 10,10 0)'::geometry,
        ARRAY[1.0, 2.0, 1.0],
        ARRAY[0, 0, 0, 1, 1, 1]));
```

Output

```
{0,0,0,1,1,1}
```

Figure



This example checks the knots of a curve using the default knot vector.

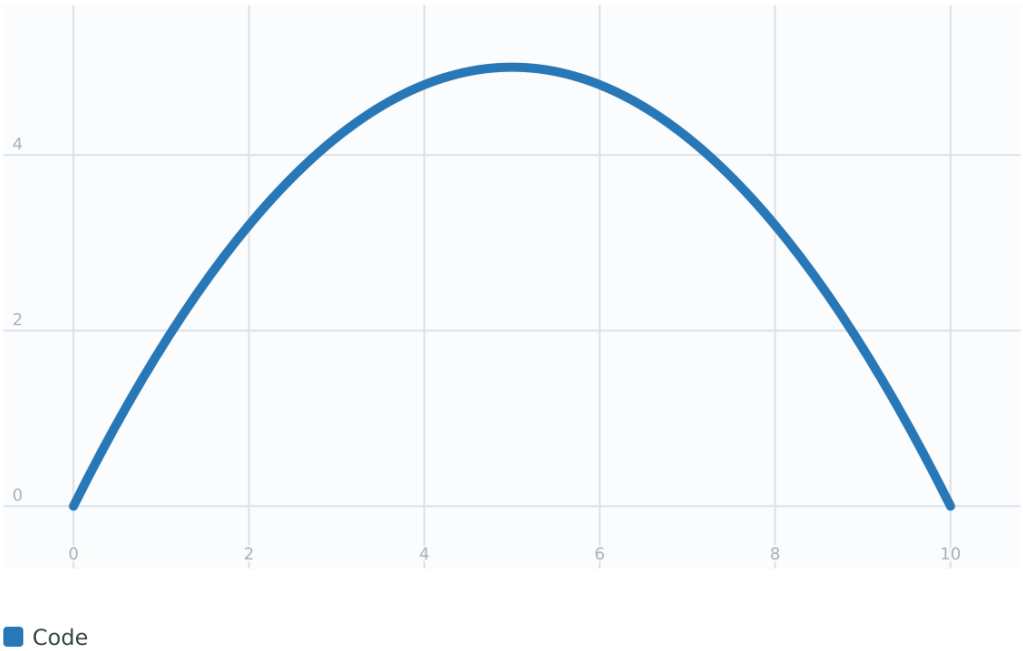
Code

```
SELECT ST_Knots('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
null
```

Figure



Se även

ST_MakeNurbsCurve, ST_Degree, ST_NumControlPoints

7.4.55 ST_NumControlPoints

ST_NumControlPoints — Returns the number of control points in a NURBS curve.

Synopsis

integer **ST_NumControlPoints**(geometry nurbscurve);

Beskrivning

Returns the number of control points in a NURBS curve. The number of control points must be at least (degree + 1) for a valid NURBS curve.

Raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

Exempel

This example returns the number of control points.

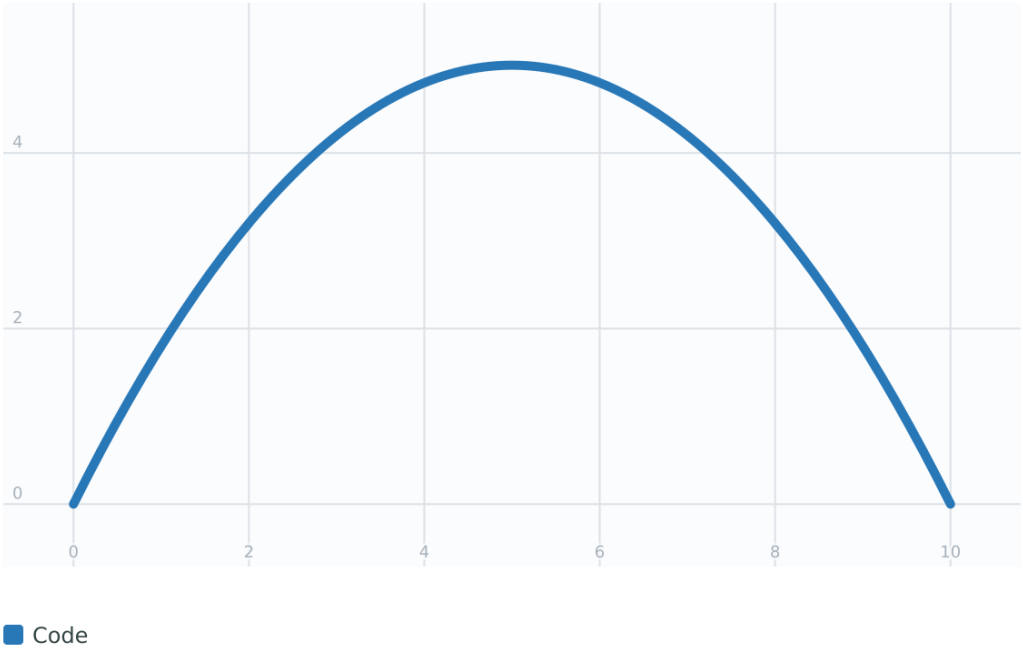
Code

```
SELECT ST_NumControlPoints('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
3
```

Figure



This example verifies that a curve has enough control points for its degree.

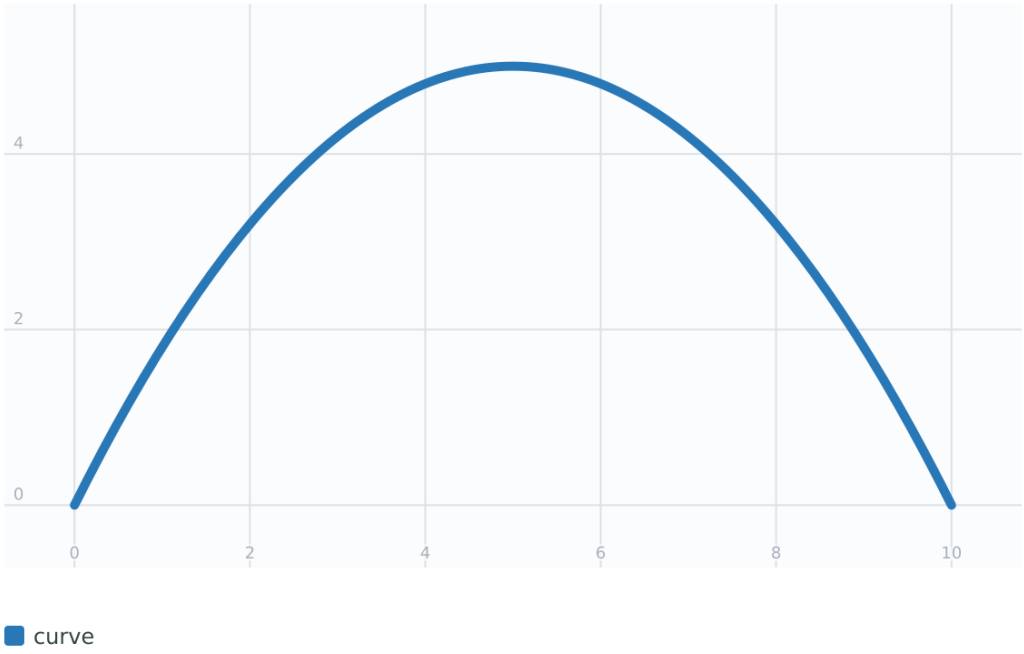
Code

```
SELECT ST_NumControlPoints(curve)
>= ST_Degree(curve) + 1 AS valid
FROM (SELECT 'NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry AS curve) AS t;
```

Output

```
t
```

Figure



Se även

ST_MakeNurbsCurve, ST_ControlPoints, ST_Degree

7.4.56 ST_NurbsCurveIsRational

ST_NurbsCurveIsRational — Checks if a NURBS curve is rational (has weights).

Synopsis

boolean **ST_NurbsCurveIsRational**(geometry nurbscurve);

Beskrivning

Returns TRUE if the NURBS curve has weights assigned to its control points (rational NURBS), FALSE if it uses uniform weights (non-rational B-spline).

Rational NURBS curves provide more flexibility in curve shape control through variable weights. Non-rational B-splines use uniform weights and are a special case of NURBS curves.

Raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

Exempel

This example checks whether a curve is rational.

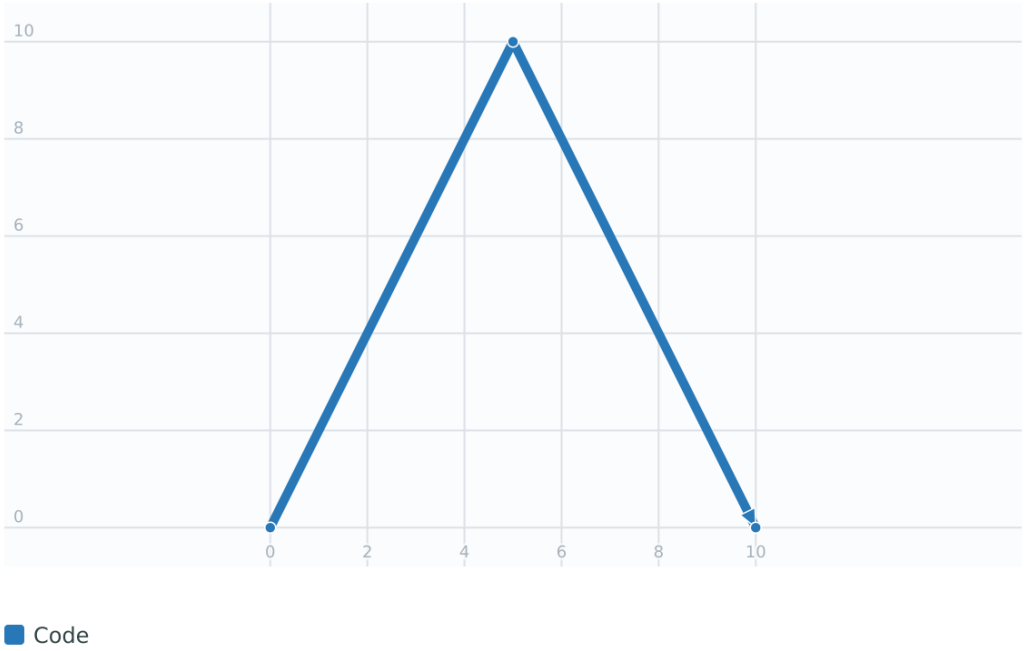
Code

```
SELECT ST_NurbsCurveIsRational(ST_MakeNurbsCurve(
        2,
        'LINESTRING(0 0,5 10,10 0)'::geometry,
        ARRAY[1.0, 2.0, 1.0]));
```

Output

```
t
```

Figure



This example checks a non-rational curve.

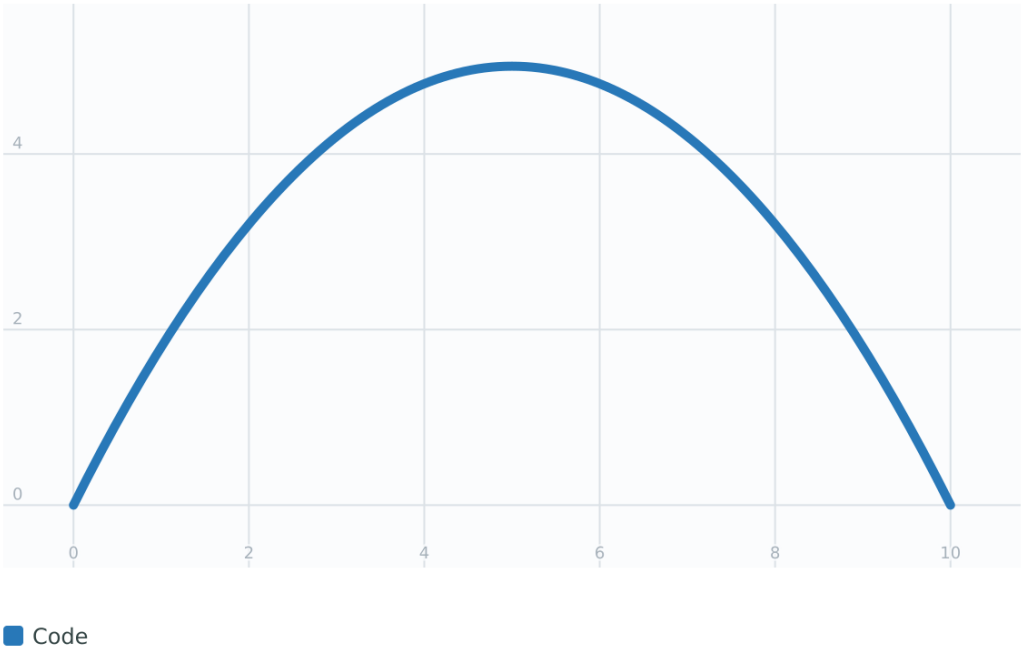
Code

```
SELECT ST_NurbsCurveIsRational('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
f
```

Figure



Se även

ST_MakeNurbsCurve, ST_Weights, ST_IsNurbsCurve

7.4.57 ST_IsNurbsCurve

ST_IsNurbsCurve — Checks if a geometry is a NURBS curve.

Synopsis

boolean **ST_IsNurbsCurve**(geometry geom);

Beskrivning

Returns TRUE if the geometry is a NURBS curve, FALSE otherwise. This function can be used to test geometry types before applying NURBS-specific functions.

Returns NULL if the input is NULL.

Availability: 3.7.0

Exempel

This example checks whether a geometry is a NURBS curve.

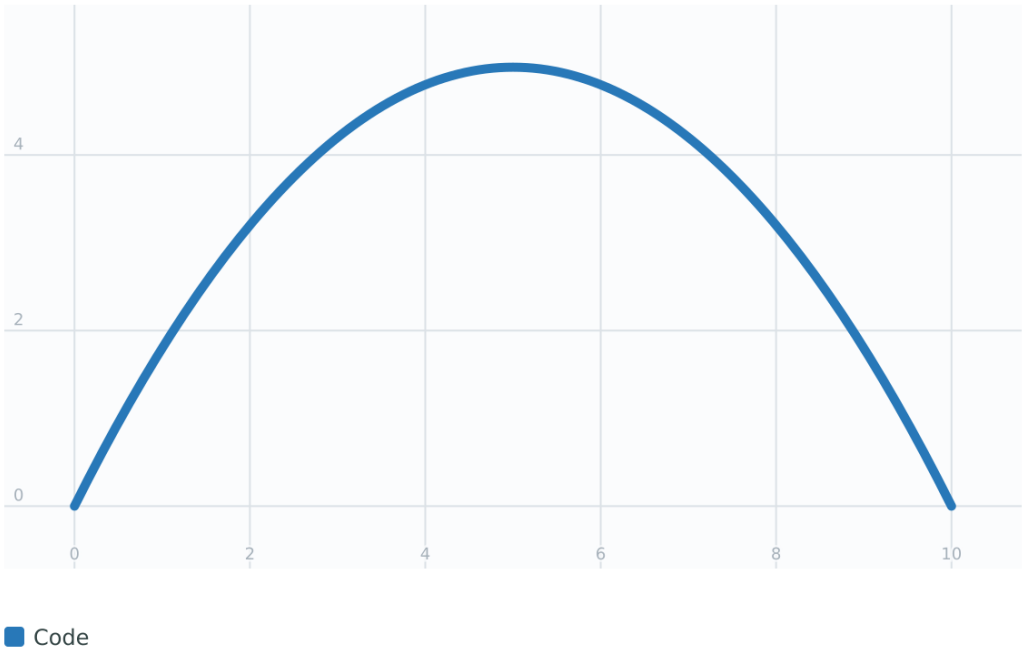
Code

```
SELECT ST_IsNurbsCurve('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
t
```

Figure



This example tests a regular LineString.

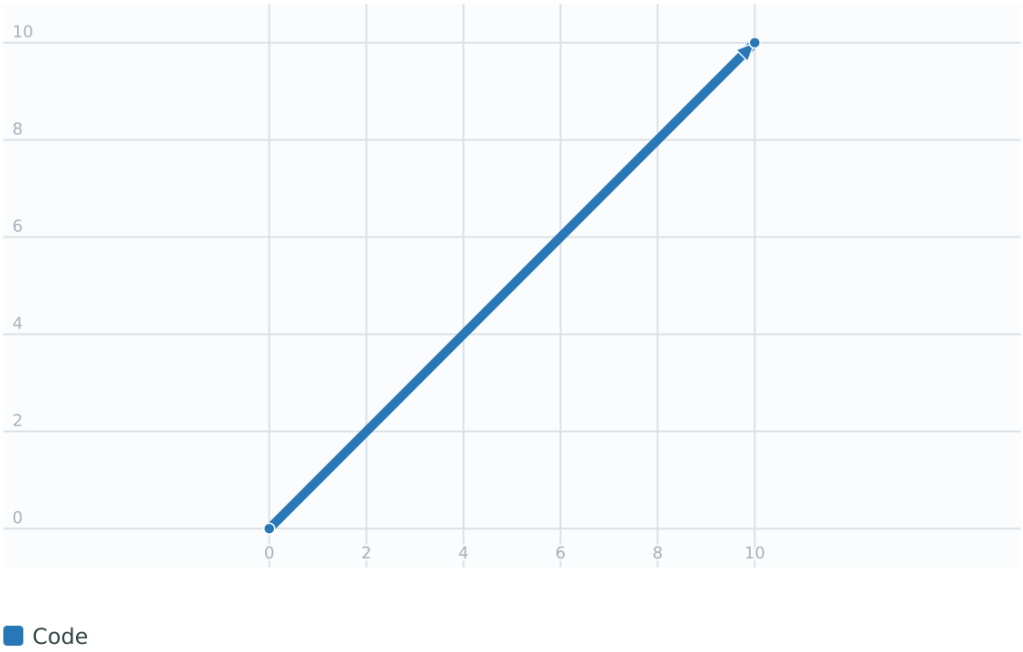
Code

```
SELECT ST_IsNurbsCurve('LINESTRING(0 0,10 10)'::geometry);
```

Output

```
f
```

Figure



Se även

ST_MakeNurbsCurve, ST_GeometryType, ST_NurbsCurveIsRational

7.4.58 ST_Evaluate

ST_Evaluate — Evaluates a NURBS curve at a specific parameter value and returns the resulting point.

Synopsis

geometry **ST_Evaluate**(geometry nurbscurve, float8 parameter);

Beskrivning

Evaluates a NURBS curve at the specified parameter value and returns a POINT geometry representing the position on the curve. The parameter typically ranges from 0.0 (start of curve) to 1.0 (end of curve), though the actual range depends on the knot vector.

Raises an error if the input is not a NURBS curve or if evaluation fails. Returns NULL if either input is NULL.

Availability: 3.7.0

Exempel

This example returns the midpoint of a NURBS curve.

Code

```
SELECT ST_Evaluate('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry, 0.5);
```

This example samples multiple points along a curve.

Code

```
SELECT t, ST_Evaluate(curve, t) AS point
FROM (SELECT 'NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry AS curve) AS c,
         generate_series(0, 1, 0.25) AS t;
```

Se även

ST_MakeNurbsCurve, ST_NurbsToLineString

7.4.59 ST_NurbsToLineString

ST_NurbsToLineString — Converts a NURBS curve to a LINESTRING by uniform sampling.

Synopsis

geometry **ST_NurbsToLineString**(geometry nurbscurve, integer num_segments=32);

Beskrivning

Converts a NURBS curve to a LINESTRING geometry by creating a piecewise linear approximation. The curve is sampled at uniformly distributed parameter values and the resulting points are connected with straight line segments.

The optional `num_segments` parameter specifies the number of line segments in the output (default: 32, range: 2-10000). Higher values produce smoother approximations but increase geometry complexity.

Raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

Exempel

This example converts a NURBS curve to a LineString using the default quality.

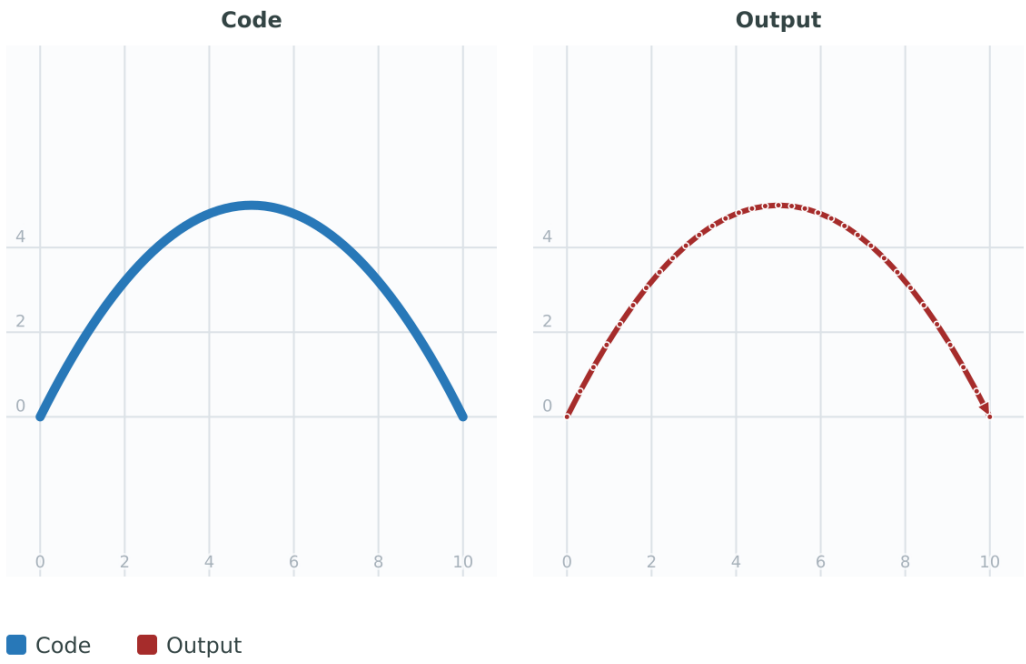
Code

```
SELECT ST_NurbsToLineString('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
LINESTRING(0 0,0.3125 0.60546875,0.625 1.171875,0.9375 1.69921875,1.25 2.1875,1.5625 2.63671875,1.875 3.046875,2.1875 ↩
    3.41796875,2.5 3.75,2.8125 4.04296875,3.125 4.296875,3.4375 4.51171875,3.75 4.6875,4.0625 4.82421875,4.375 ↩
    4.921875,4.6875 4.98046875,5 5,5.3125 4.98046875,5.625 4.921875,5.9375 4.82421875,6.25 4.6875,6.5625 ↩
    4.51171875,6.875 4.296875,7.1875 4.04296875,7.5 3.75,7.8125 3.41796875,8.125 3.046875,8.4375 2.63671875,8.75 ↩
    2.1875,9.0625 1.69921875,9.375 1.171875,9.6875 0.60546875,10 0)
```

Figure

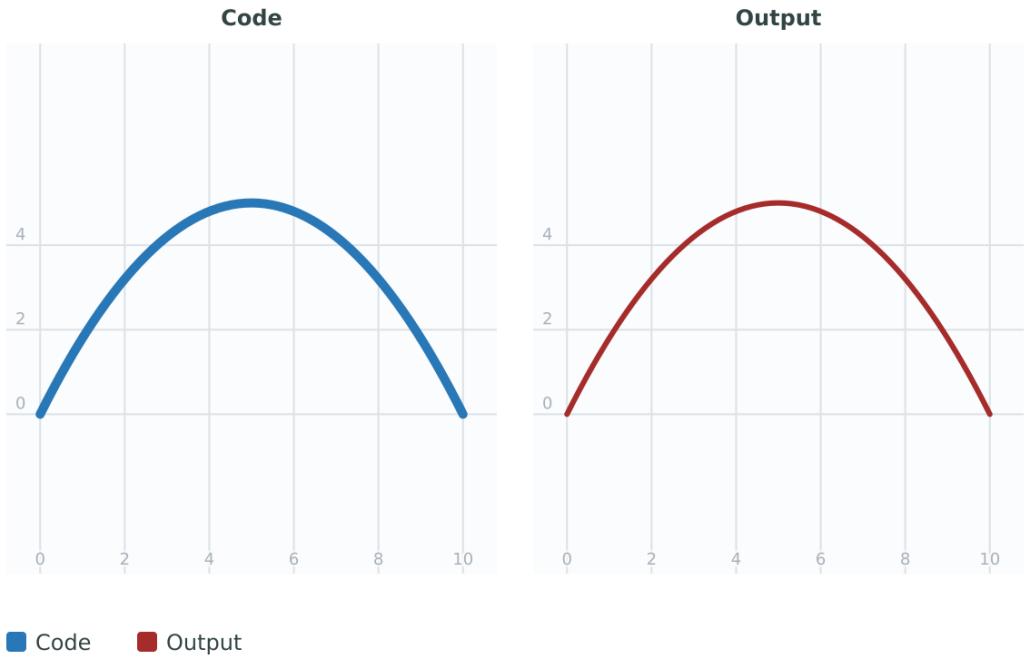


This example creates a high-quality approximation with 64 segments.

Code

```
SELECT ST_NurbsToLineString('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry, 64);
```

Figure



This example creates a lower-quality, faster approximation with 16 segments.

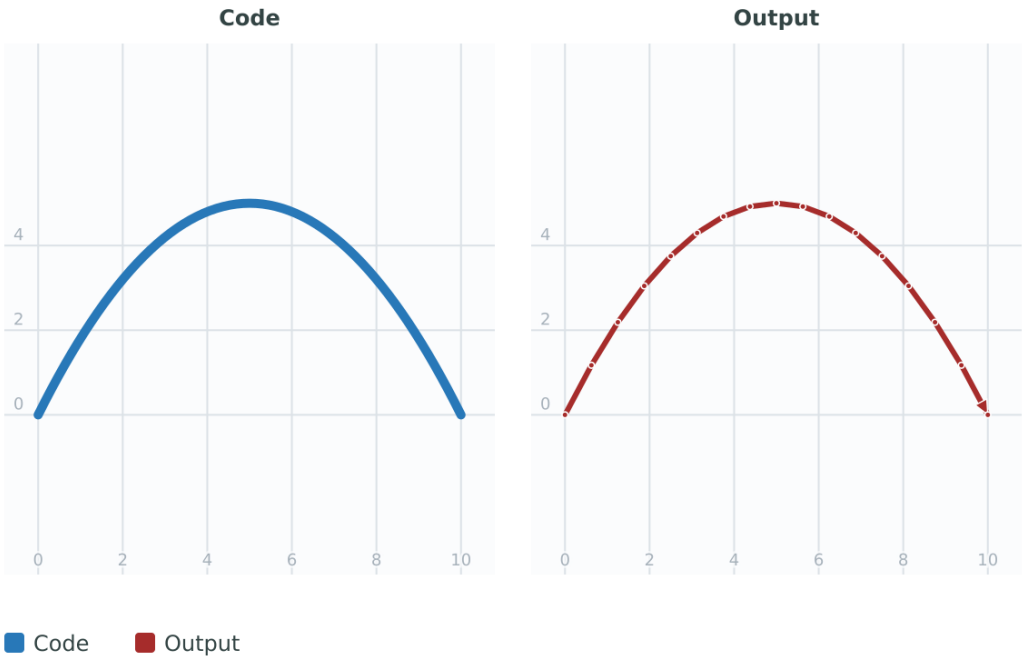
Code

```
SELECT ST_NurbsToLineString('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry, 16);
```

Output

```
LINESTRING(0 0,0.625 1.171875,1.25 2.1875,1.875 3.046875,2.5 3.75,3.125 4.296875,3.75 4.6875,4.375 4.921875,5 5,5.625 ↩
    4.921875,6.25 4.6875,6.875 4.296875,7.5 3.75,8.125 3.046875,8.75 2.1875,9.375 1.171875,10 0)
```

Figure



Se även

ST_MakeNurbsCurve, ST_Evaluate, ST_Segmentize

7.5 Geometriredigerare

7.5.1 ST_AddPoint

ST_AddPoint — Lägg till en punkt i en LineString.

Synopsis

geometry **ST_AddPoint**(geometry linestring, geometry point);

geometry **ST_AddPoint**(geometry linestring, geometry point, integer position = -1);

Beskrivning

Lägger till en punkt till en LineString före *indexpositionen* (med ett 0-baserat index). Om *positionsparameter* utelämnas eller är -1 läggs punkten till i slutet av LineString.

Tillgänglighet: 1.1.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Lägg till en punkt i slutet av en 3D-linje

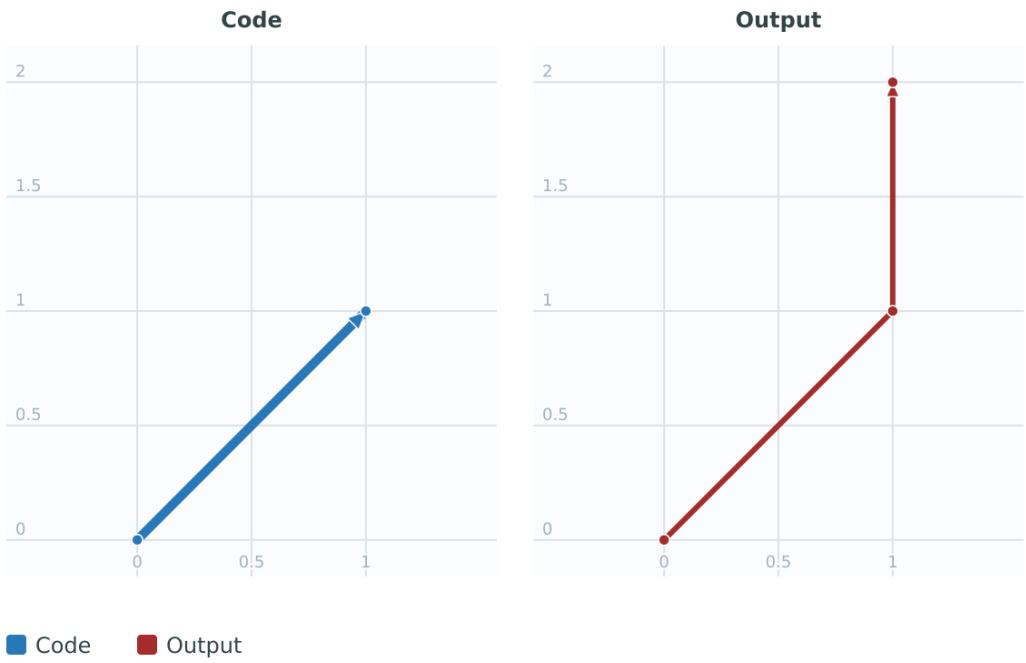
Code

```
SELECT ST_AddPoint('LINESTRING(0 0 1,1 1 1)', ST_MakePoint(1, 2, 3));
```

Output

```
LINESTRING(0 0 1,1 1 1,1 2 3)
```

Figure



Garantera att alla rader i en tabell är slutna genom att lägga till startpunkten för varje rad till slutet av raden endast för de rader som inte är slutna.

Code

```
UPDATE sometable
SET geom = ST_AddPoint(geom, ST_StartPoint(geom))
FROM sometable
WHERE ST_IsClosed(geom) = false;
```

Se även

ST_RemovePoint, ST_SetPoint

7.5.2 ST_CollectionExtract

ST_CollectionExtract — Ger en geometrisamling och returnerar en multigeometri som endast innehåller element av en angiven typ.

Synopsis

geometry **ST_CollectionExtract**(geometry collection);

geometry **ST_CollectionExtract**(geometry collection, integer type);

Beskrivning

Ger en geometrisamling och returnerar en homogen multigeometri.

Om *typen* inte anges returneras en multigeometri som endast innehåller geometrier med den högsta dimensionen. Polygoner är alltså att föredra framför linjer, som i sin tur är att föredra framför punkter.

Om *typen* anges, returneras en multigeometri som endast innehåller den typen. Om det inte finns några subgeometrier av rätt typ returneras en EMPTY geometri. Endast punkter, linjer och polygoner stöds. Typnumren är:

- 1 == PUNKT
- 2 == LINESTRING
- 3 == POLYGON

För inmatningar med atomär geometri returneras geometrin oförändrad om inmatningstypen matchar den begärda typen. I annat fall är resultatet en EMPTY-geometri av den angivna typen. Om så krävs kan dessa konverteras till multi-geometrier med ST_Multi.

Varning
MultiPolygon-resultat kontrolleras inte med avseende på giltighet. Om polygonkomponenterna är angränsande eller överlappande blir resultatet ogiltigt. (Detta kan t.ex. inträffa när denna funktion används på ett resultat från ST_Split.) Denna situation kan kontrolleras med ST_IsValid och repareras med ST_MakeValid.

Tillgänglighet: 1.5.0

Notera
Före 1.5.3 returnerade denna funktion atomära indata oförändrade, oavsett typ. I 1.5.3 returnerade icke-matchande singelgeometrier ett NULL-resultat. I 2.0.0 returnerar icke-matchande enskilda geometrier ett EMPTY-resultat av den begärda typen.

Exempel

Extrahera typ med högsta dimension:

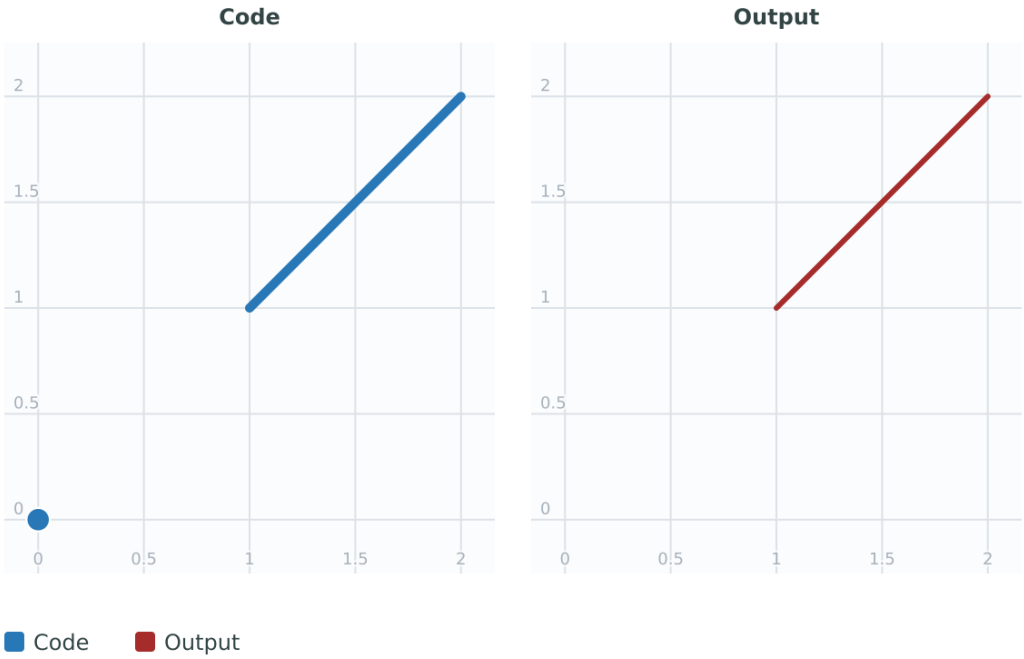
Code

```
SELECT ST_CollectionExtract('GEOMETRYCOLLECTION( POINT(0 0), LINESTRING(1 1,2 2) )');
```

Output

```
MULTILINESTRING((1 1, 2 2))
```

Figure



Extrahera punkter (typ 1 == POINT):

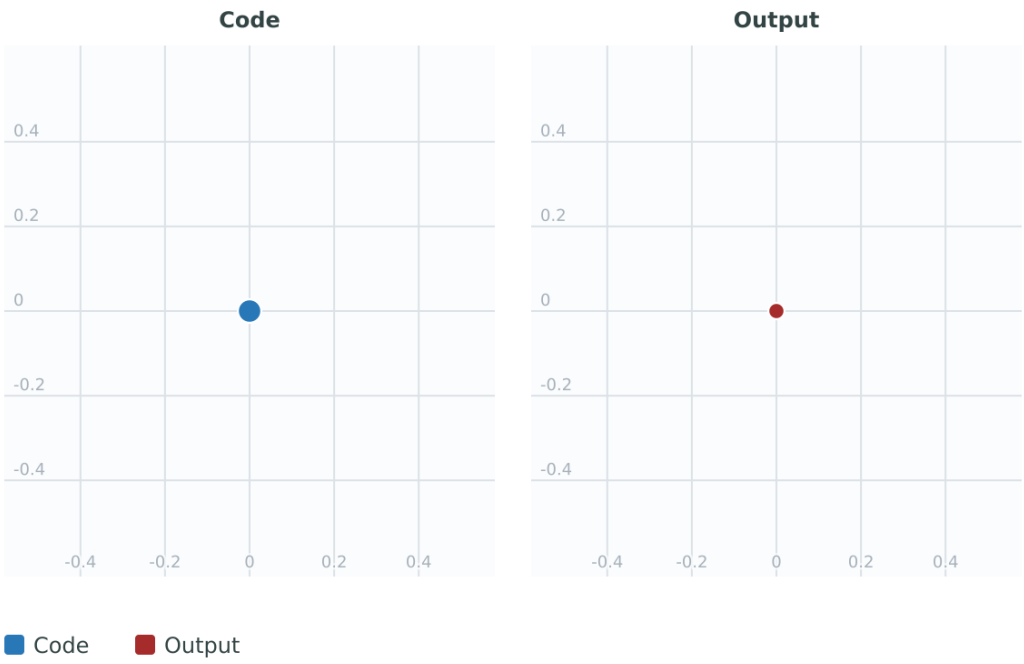
Code

```
SELECT ST_CollectionExtract('GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(POINT(0 0)))',
        1);
```

Output

```
MULTIPOINT((0 0))
```

Figure



Extrahera linjer (typ 2 == LINESTRING):

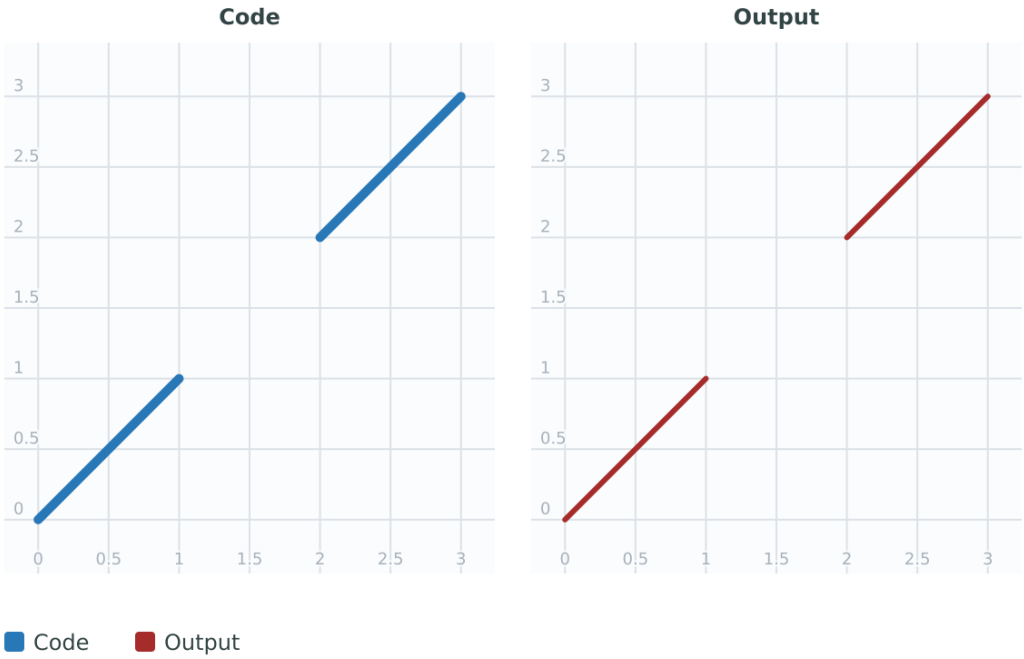
Code

```
SELECT ST_CollectionExtract('GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(LINESTRING(0 0,1 1)),  ↩
    LINESTRING(2 2,3 3))',
        2);
```

Output

```
MULTILINESTRING((0 0, 1 1), (2 2, 3 3))
```

Figure



Se även

ST_CollectionHomogenize, ST_Multi, ST_IsValid, ST_MakeValid

7.5.3 ST_CollectionHomogenize

ST_CollectionHomogenize — Returnerar den enklaste representationen av en geometrisamling.

Synopsis

geometry **ST_CollectionHomogenize**(geometry collection);

Beskrivning

Ger en geometrisamling och returnerar den ”enklasterepresentationen av innehållet.

- Homogena (enhetliga) samlingar returneras som lämplig multigeometri.
- Heterogena (blandade) samlingar plattas ut till en enda GeometryCollection.
- Samlingar som innehåller ett enda atomärt element returneras som det elementet.
- Atomgeometrier returneras oförändrade. Om så krävs kan dessa konverteras till en multigeometri med hjälp av ST_Multi.

Varning
Denna funktion garanterar inte att resultatet är giltigt. I synnerhet kommer en samling som innehåller intilliggande eller överlappande polygoner att skapa en ogiltig MultiPolygon. Denna situation kan kontrolleras med ST_IsValid och repareras med ST_MakeValid.

Tillgänglighet: 2.0.0

Exempel

Samling med ett enda element omvandlad till en atomär geometri

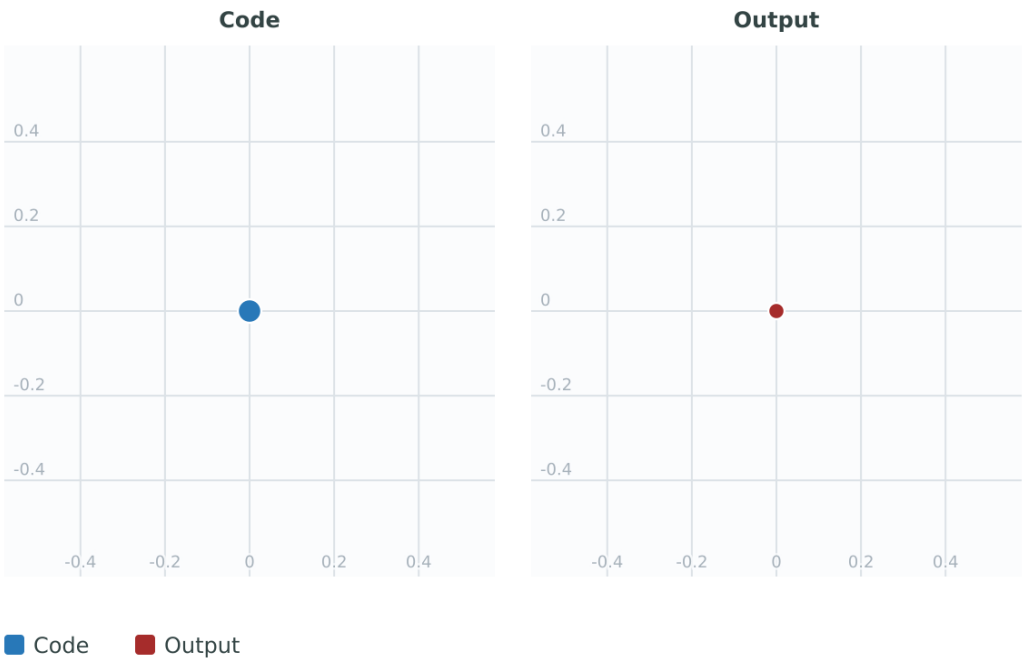
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0))');
```

Output

```
PUNKT(0 0)
```

Figure



Nästlad samling av enskilda element konverterad till en atomär geometri:

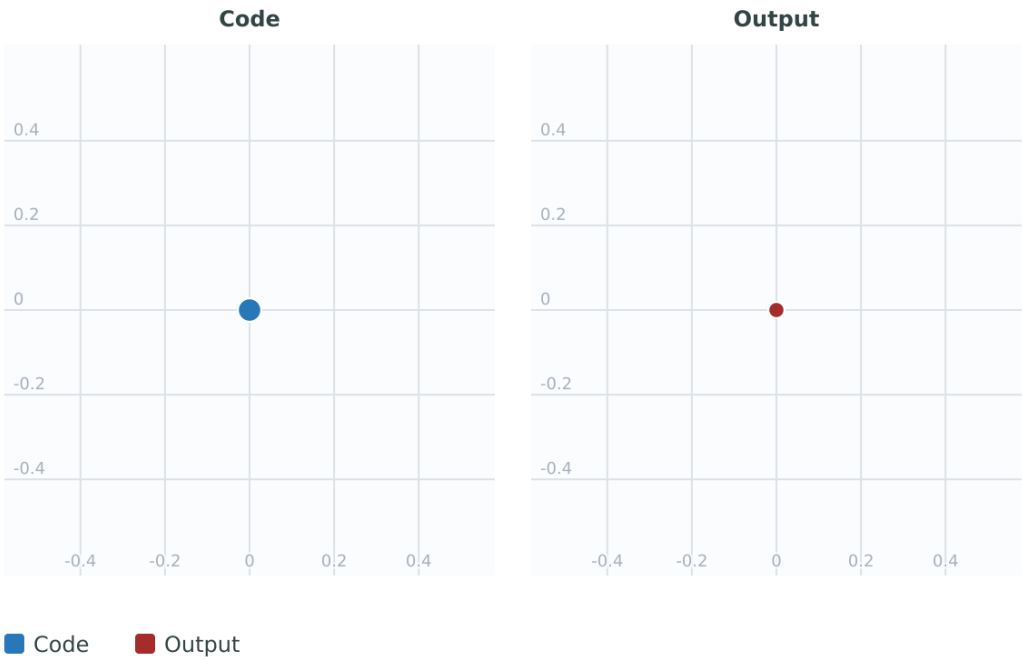
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(MULTIPOINT((0 0)))');
```

Output

```
PUNKT(0 0)
```

Figure



Samling omvandlad till en multigeometri:

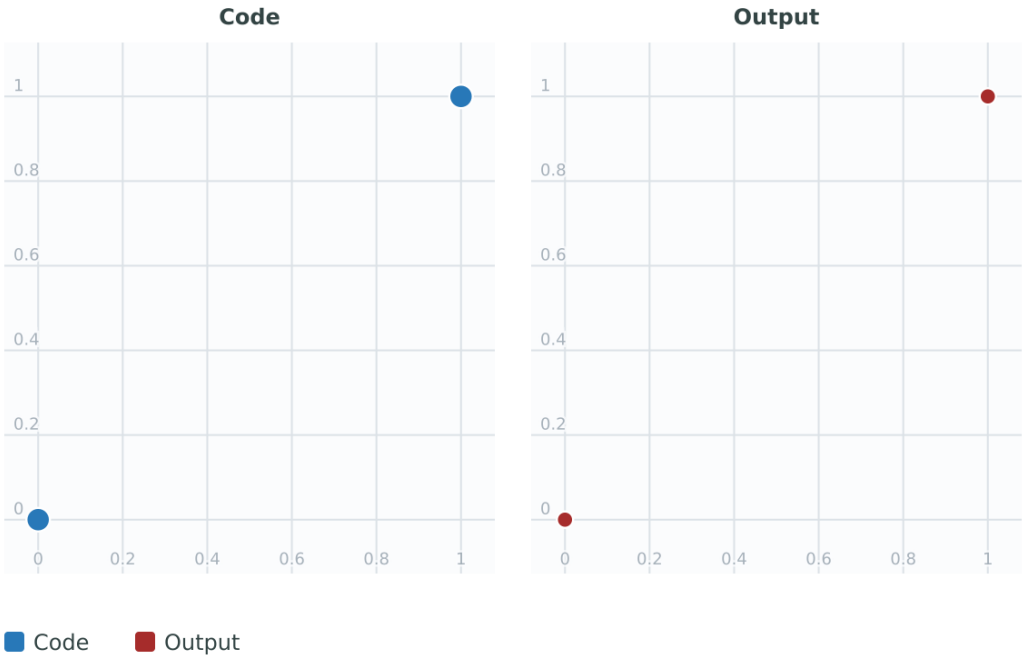
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0), POINT(1 1))');
```

Output

```
MULTIPOINT((0 0),(1 1))
```

Figure



Nästlad heterogen samling som plattats ut till en GeometryCollection:

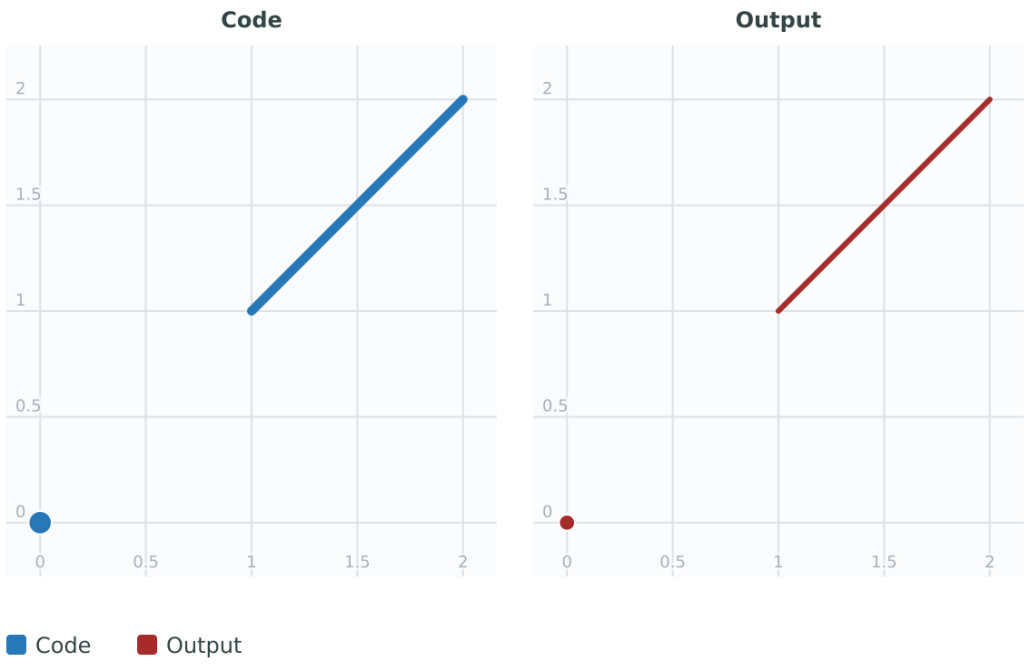
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0), GEOMETRYCOLLECTION(
    LINESTRING(1 1,2 2)))');
```

Output

```
GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(1 1,2 2))
```

Figure



Samling av polygoner som konverterats till en (ogiltig) MultiPolygon:

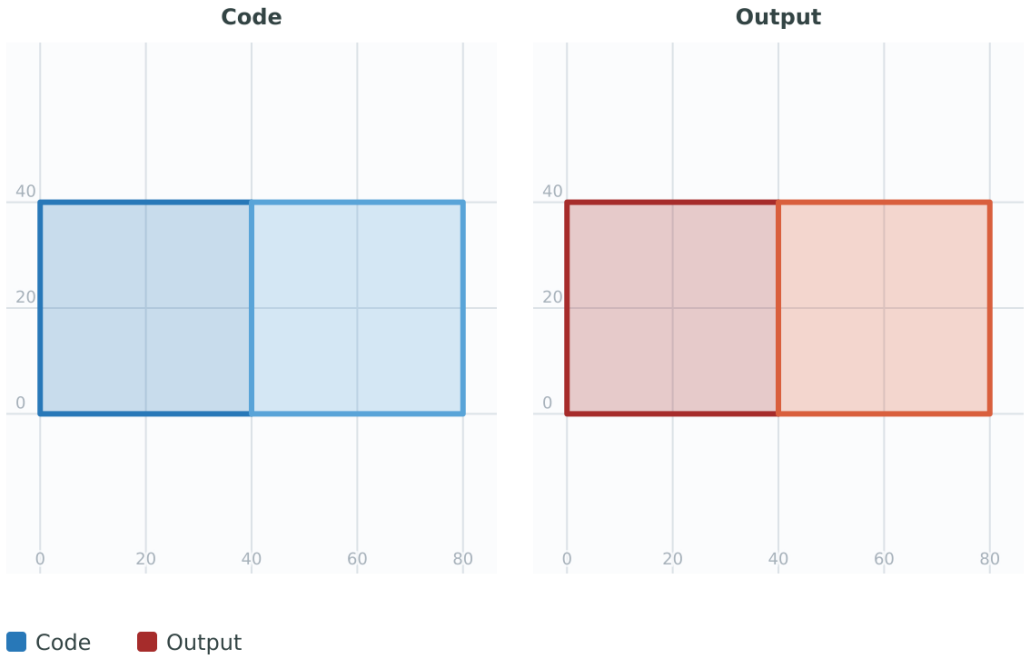
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION (POLYGON ((10 50,50 50,50 10,10 10,10 ↩
    50)),POLYGON ((90 50,90 10,50 10,50 50,90 50)))');
```

Output

```
MULTIPOLYGON(((10 50,50 50,50 10,10 10,10 50)),((90 50,90 10,50 10,50 50,90 50)))
```

Figure



Se även

ST_CollectionExtract, ST_Multi, ST_IsValid, ST_MakeValid

7.5.4 ST_CurveToLine

ST_CurveToLine — Konverterar en geometri som innehåller kurvor till en linjär geometri.

Synopsis

geometry **ST_CurveToLine**(geometry curveGeom, float tolerance, integer tolerance_type, integer flags);

Beskrivning

Konverterar en CIRCULAR STRING till vanlig LINESTRING eller CURVEPOLYGON till POLYGON eller MULTISURFACE till MULTIPOLYGON. Användbart för utdata till enheter som inte stöder geometri-typer av typen CIRCULARSTRING

Konverterar en given geometri till en linjär geometri. Varje krökt geometri eller segment konverteras till en linjär approximation med hjälp av den angivna `toleransen` och alternativen (32 segment per kvadrant och inga alternativ som standard).

Argumentet tolerance_typebestämmer tolkningen av argumentet tolerance”. Det kan anta följande värden:

- 0 (standard): Toleransen är max segment per kvadrant.
- 1: Tolerans är max-avvikelse för linjen från kurvan, i källenheter.
- 2: Toleransen är maxvinkeln, i radianer, mellan genereringsradierna.

Argumentet flags”är ett bitfält. 0 som standard. Bitar som stöds är:

- 1: Symmetrisk (orienteringsoberoende) utdata.
- 2: Behåll vinkeln, undviker att minska vinklar (segmentlängder) när symmetrisk utdata produceras. Har ingen effekt när Symmetric-flaggan är avstängd.

Tillgänglighet: 1.3.0

Förbättrad: I 2.4.0 tillkom stöd för toleranserna max-avvikelse och max-vinkel samt för symmetrisk utdata.

Förbättrad: 3.0.0 implementerade ett minsta antal segment per linjäriserad båge för att förhindra topologisk kollaps.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.1.7

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

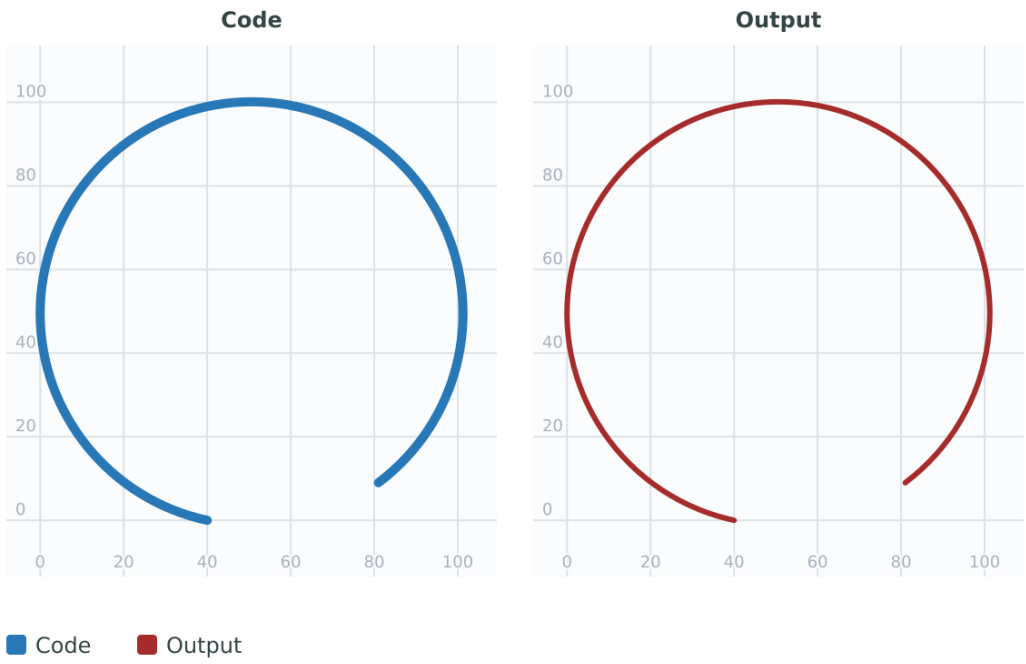
Code

```
SELECT ST_CurveToLine('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)');
```

Output

```
LINESTRING(220268 150415,220269.95064912 150416.539364228,220271.823415575 150418.17258804,220273.613787707 ↩
     150419.895736857,
220275.317452352 150421.704659462,220276.930305234 150423.594998003,220278.448460847 150425.562198489,
220279.868261823 150427.60152176,220281.186287736 150429.708054909,220282.399363347 150431.876723113,
220283.50456625 150434.10230186,220284.499233914 150436.379429536,220285.380970099 150438.702620341,220286.147650624 ↩
     150441.066277505,
220286.797428488 150443.464706771,220287.328738321 150445.892130112,220287.740300149 150448.342699654,
220288.031122486 150450.810511759,220288.200504713 150453.289621251,220288.248038775 150455.77405574,
220288.173610157 150458.257830005,220287.977398166 150460.734960415,220287.659875492 150463.199479347,
220287.221807076 150465.64544956,220286.664248262 150468.066978495,220285.988542259 150470.458232479,220285.196316903 ↩
     150472.81345077,
220284.289480732 150475.126959442,220283.270218395 150477.39318505,220282.140985384 150479.606668057,
220280.90450212 150481.762075989,220279.5637474 150483.85421628,220278.12195122 150485.87804878,
220276.582586992 150487.828697901,220274.949363179 150489.701464356,220273.226214362 150491.491836488,
220271.417291757 150493.195501133,220269.526953216 150494.808354014,220267.559752731 150496.326509628,
220265.520429459 150497.746310603,220263.41389631 150499.064336517,220261.245228106 150500.277412127,
220259.019649359 150501.38261503,220256.742521683 150502.377282695,220254.419330878 150503.259018879,
220252.055673714 150504.025699404,220249.657244448 150504.675477269,220247.229821107 150505.206787101,
220244.779251566 150505.61834893,220242.311439461 150505.909171266,220239.832329968 150506.078553494,
220237.347895479 150506.126087555,220234.864121215 150506.051658938,220232.386990804 150505.855446946,
220229.922471872 150505.537924272,220227.47650166 150505.099855856,220225.054972724 150504.542297043,
220222.663718741 150503.86659104,220220.308500449 150503.074365683,
220217.994991777 150502.167529512,220215.72876617 150501.148267175,
220213.515283163 150500.019034164,220211.35987523 150498.7825509,
220209.267734939 150497.441796181,220207.243902439 150496,
220205.293253319 150494.460635772,220203.420486864 150492.82741196,220201.630114732 150491.104263143,
220199.926450087 150489.295340538,220198.313597205 150487.405001997,220196.795441592 150485.437801511,
220195.375640616 150483.39847824,220194.057614703 150481.291945091,220192.844539092 150479.123276887,220191.739336189 ↩
     150476.89769814,
220190.744668525 150474.620570464,220189.86293234 150472.297379659,220189.096251815 150469.933722495,
220188.446473951 150467.535293229,220187.915164118 150465.107869888,220187.50360229 150462.657300346,
220187.212779953 150460.189488241,220187.043397726 150457.710378749,220186.995863664 150455.22594426,
220187.070292282 150452.742169995,220187.266504273 150450.265039585,220187.584026947 150447.800520653,
220188.022095363 150445.35455044,220188.579654177 150442.933021505,220189.25536018 150440.541767521,
220190.047585536 150438.18654923,220190.954421707 150435.873040558,220191.973684044 150433.60681495,
220193.102917055 150431.393331943,220194.339400319 150429.237924011,220195.680155039 150427.14578372,220197.12195122 ↩
     150425.12195122,
220198.661315447 150423.171302099,220200.29453926 150421.298535644,220202.017688077 150419.508163512,220203.826610682 ↩
     150417.804498867,
220205.716949223 150416.191645986,220207.684149708 150414.673490372,220209.72347298 150413.253689397,220211.830006129 ↩
     150411.935663483,
220213.998674333 150410.722587873,220216.22425308 150409.61738497,220218.501380756 150408.622717305,220220.824571561 ↩
     150407.740981121,
220223.188228725 150406.974300596,220225.586657991 150406.324522731,220227 150406)
```

Figure



This example uses 3D input.

Code

```
SELECT ST_CurveToLine('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)');
```

Output

```
LINESTRING(220268 150415 1,220269.95064912 150416.539364228 1.0181172856673,
220271.823415575 150418.17258804 1.03623457133459,220273.613787707 150419.895736857 1.05435185700189,....AD INFINITUM ↩
    ....
   220225.586657991 150406.324522731 1.32611114201132,220227 150406 3)
```

Use only 2 segments to approximate quarter circle.

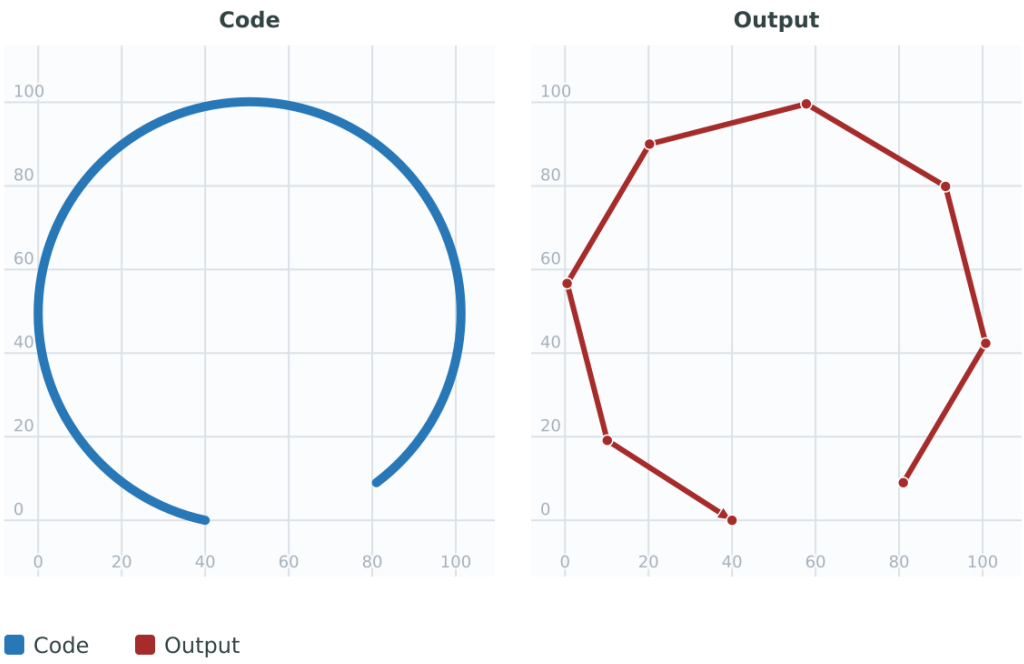
Code

```
SELECT ST_CurveToLine('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)', 2);
```

Output

```
LINESTRING(220268 150415,220287.740300149 150448.342699654,220278.12195122 150485.87804878,
220244.779251566 150505.61834893,220207.243902439 150496,220187.50360229 150462.657300346,
220197.12195122 150425.12195122,220227 150406)
```

Figure



Ensure the approximated line is no more than 20 units away from the original curve, and make the result direction-neutral.

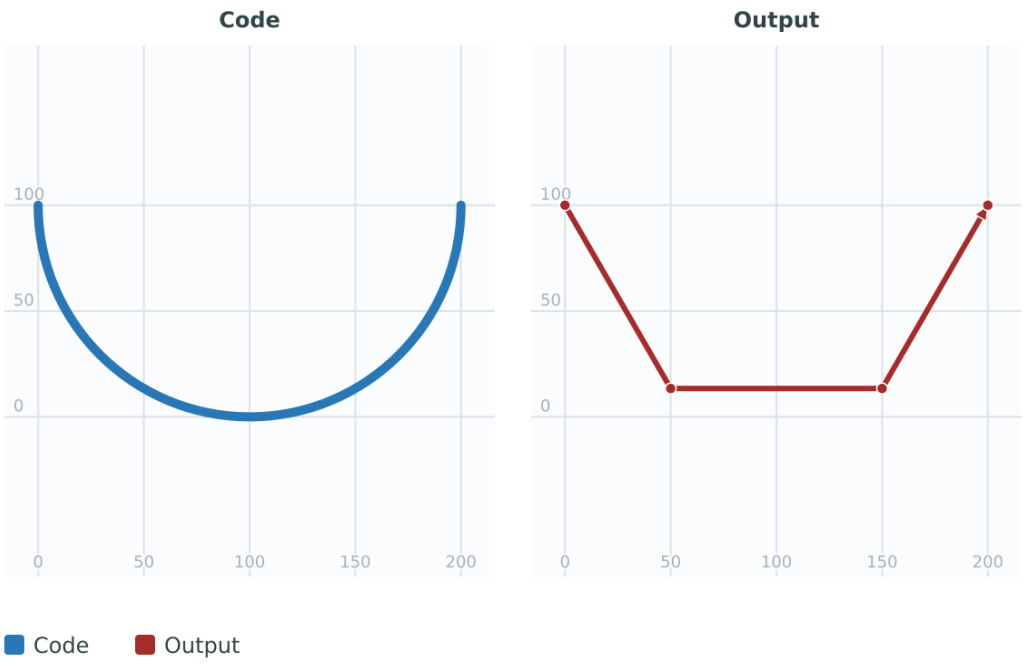
Code

```
SELECT ST_CurveToLine(
    'CIRCULARSTRING(0 0,100 -100,200 0)'::geometry,
    20,
    1,
    1
  );
```

Output

```
LINESTRING(0 0,49.99999999999996 -86.60254037844383,149.99999999999994 -86.6025403784439,200 0)
```

Figure



Se även

ST_LineToCurve

7.5.5 ST_Scroll

ST_Scroll — Ändra startpunkt för en sluten LineString.

Synopsis

geometry **ST_Scroll**(geometry linestring, geometry point);

Beskrivning

Ändrar start-/slutpunkten för en sluten LineString till den angivna *`toppunkten`*.
Tillgänglighet: 3.2.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Exempel

Make a closed line start at its 3rd vertex.

Code

```
SELECT ST_Scroll('SRID=4326;LINESTRING(0 0 0 1,10 0 2 0,5 5 4 2,0 0 0 1)', 'POINT(5 5 4 2)' ↩
    );
```

Output

```
SRID=4326;LINESTRING(5 5 4 2,0 0 0 1,10 0 2 0,5 5 4 2)
```

Figure

Se även

ST_Normalize

7.5.6 ST_FlipCoordinates

ST_FlipCoordinates — Returnerar en version av en geometri med X- och Y-axlarna vända.

Synopsis

geometry **ST_FlipCoordinates**(geometry geom);

Beskrivning

Returnerar en version av den angivna geometrin med X- och Y-axlarna vända. Användbar för att fixa geometrier som innehåller koordinater uttryckta som latitud/longitud (Y,X).

Tillgänglighet: 2.0.0

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Code

```
SELECT ST_FlipCoordinates(GeomFromEWKT('POINT(1 2)'));
```

Output

```
POINT(2 1)
```

Figure

Se även

ST_SwapOrdinates

7.5.7 ST_Force2D

ST_Force2D — Tvinga geometrierna till ett ”2-dimensionellt läge”.

Synopsis

geometry **ST_Force2D**(geometry geomA);

Beskrivning

Tvingar geometrierna till ett ”2-dimensionellt lägeså att alla utdatarepresentationer endast har X- och Y-koordinaterna. Detta är användbart för att tvinga fram OGC-kompatibla utdata (eftersom OGC endast specificerar 2D-geometrier).

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Ändrad: 2.1.0. Fram till 2.0.x kallades detta för ST_Force_2D.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

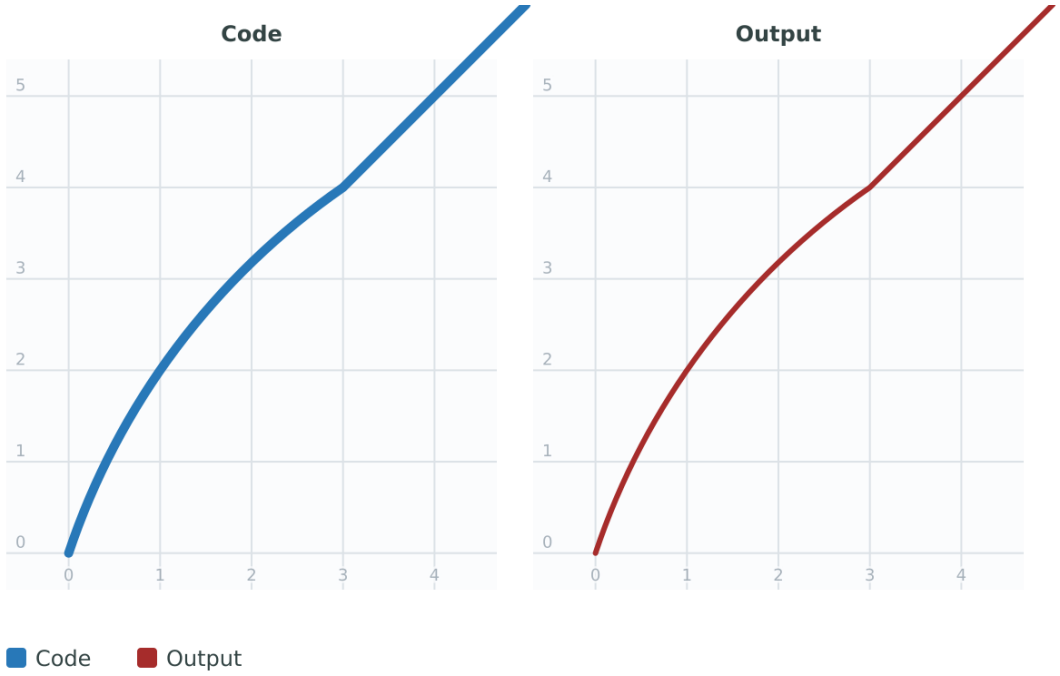
Code

```
SELECT ST_Force2D('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1,2 3,4 5,6 7,5 6)
```

Figure



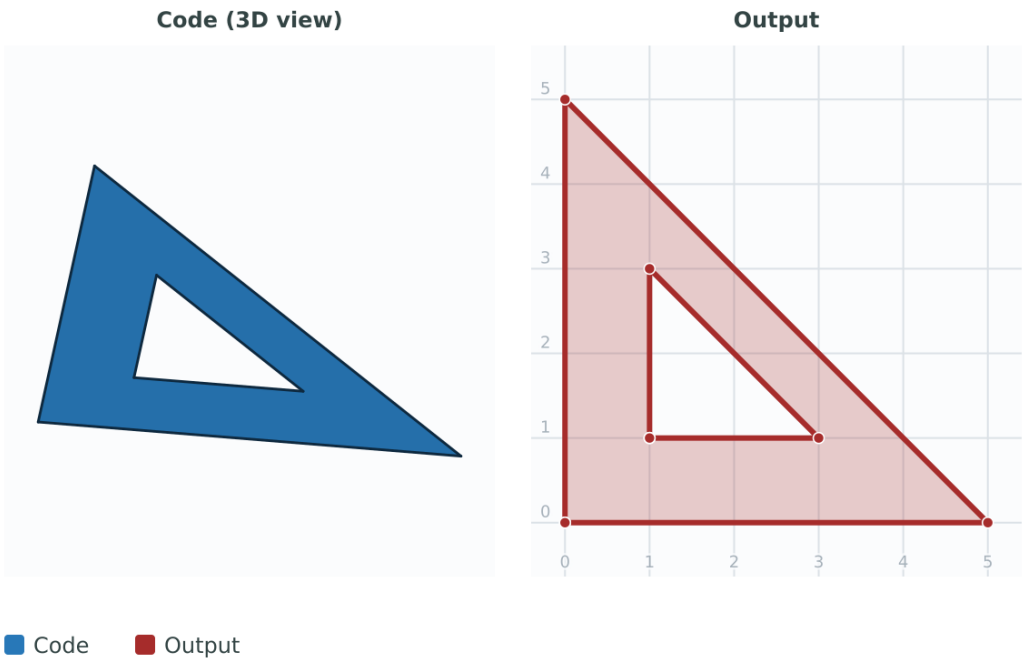
Code

```
SELECT ST_Force2D('POLYGON((0 0 2,0 5 2,5 0 2,0 0 2),(1 1 2,3 1 2,1 3 2,1 1 2))');
```

Output

```
POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))
```

Figure



Se även

ST_Force3D

7.5.8 ST_Force3D

ST_Force3D — Tvingar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ.

Synopsis

geometry **ST_Force3D**(geometry geomA, float Zvalue = 0.0);

Beskrivning

Forcerar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ. Om en geometri inte har någon Z-komponent läggs en Z-koordinat med *Z-värde* till.

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Force_3D.

Ändrad: 3.1.0. Lagt till stöd för att ange ett Z-värde som inte är noll.

Denna funktion stöder polyedriska ytor.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Nothing happens to an already 3D geometry.

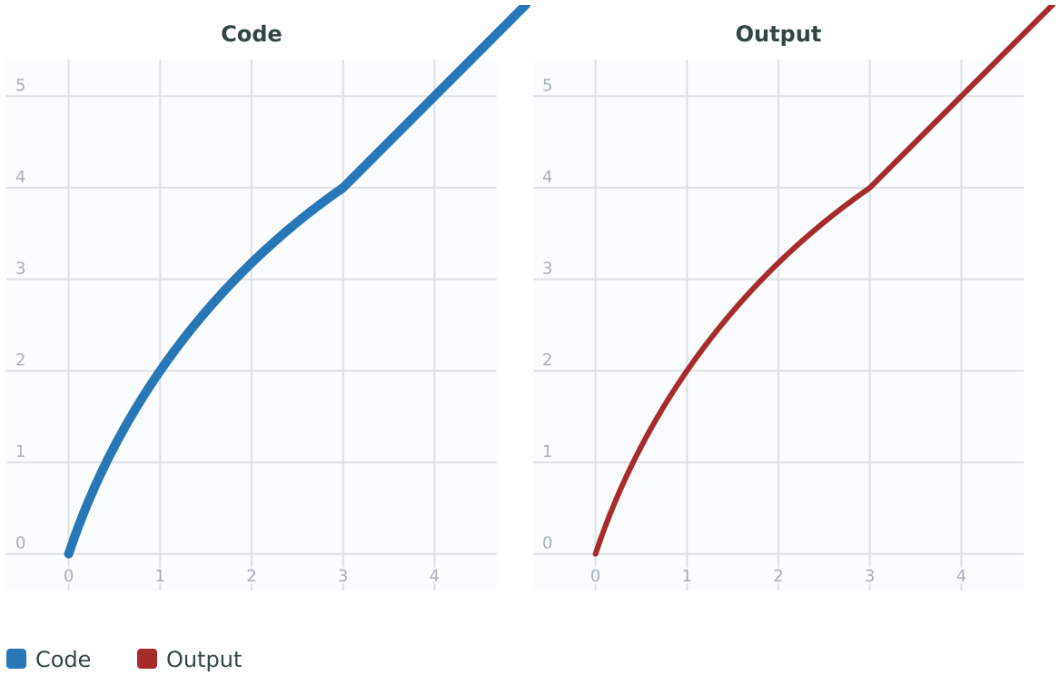
Code

```
SELECT ST_Force3D('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)
```

Figure



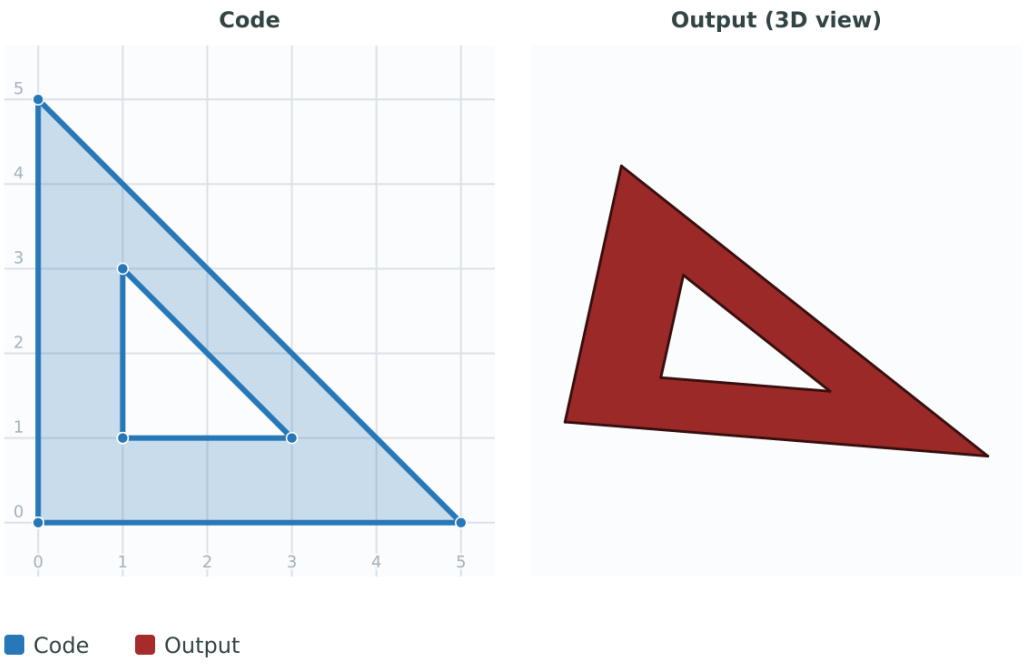
Code

```
SELECT ST_Force3D('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))');
```

Output

```
POLYGON((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0))
```

Figure



Se även

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3DZ

7.5.9 ST_Force3DZ

ST_Force3DZ — Tvinga geometrierna till XYZ-läge.

Synopsis

geometry **ST_Force3DZ**(geometry geomA, float Zvalue = 0.0);

Beskrivning

Tvingar geometrierna till XYZ-läge. Om en geometri inte har någon Z-komponent läggs en Z-koordinat med *Z-värde* till.

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Ändrad: 2.1.0. Fram till 2.0.x kallades detta för ST_Force_3DZ.

Ändrad: 3.1.0. Lagt till stöd för att ange ett Z-värde som inte är noll.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Nothing happens to an already 3D geometry.

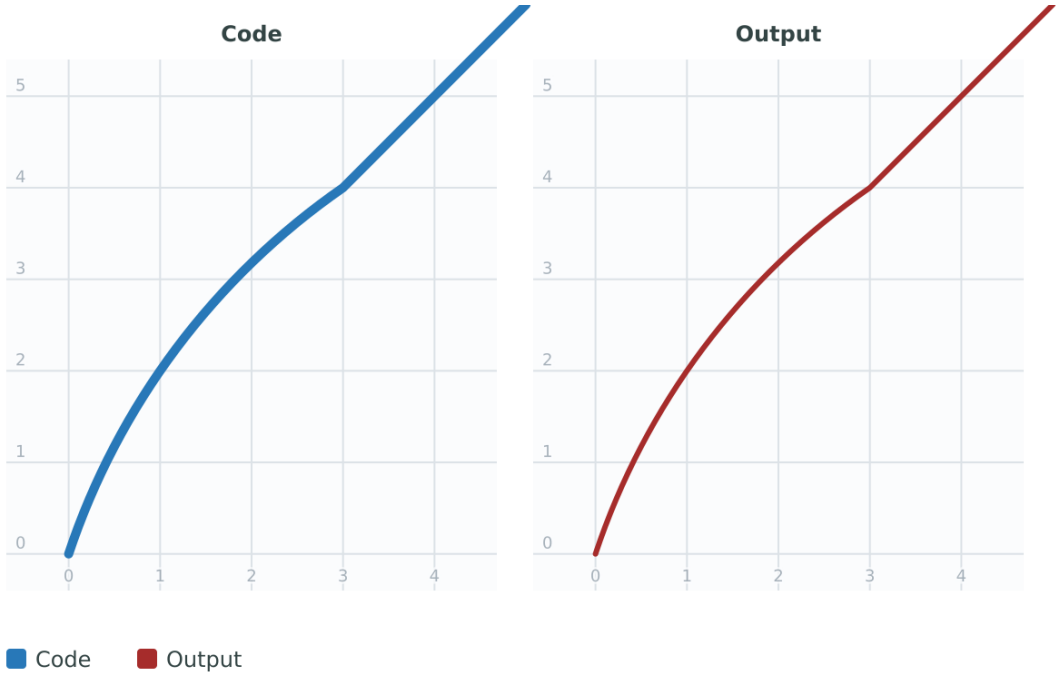
Code

```
SELECT ST_Force3DZ('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)
```

Figure



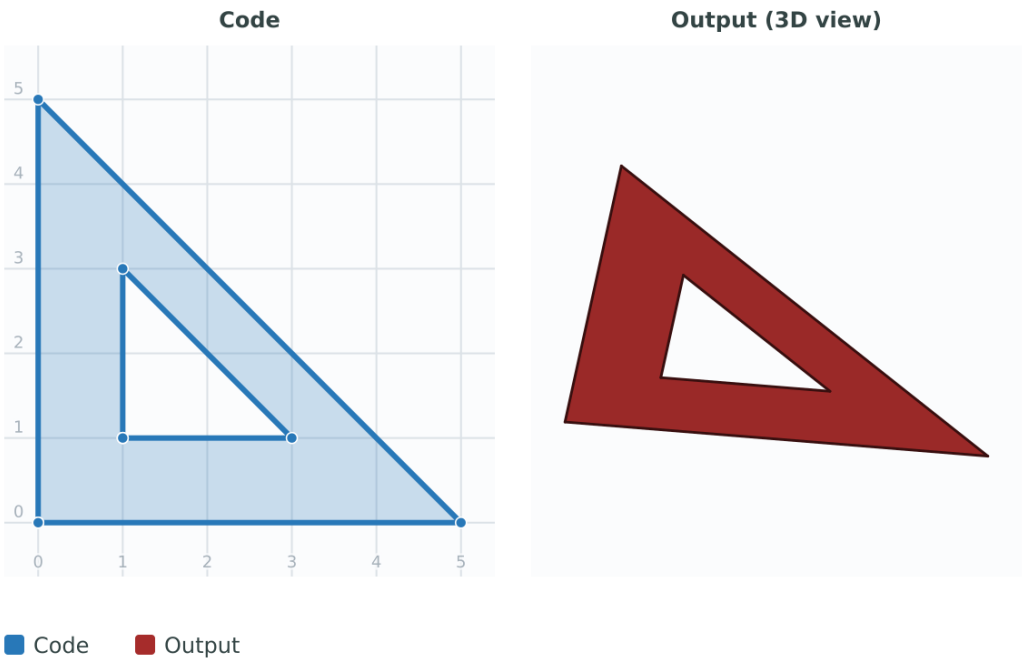
Code

```
SELECT ST_Force3DZ('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))');
```

Output

```
POLYGON((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0))
```

Figure



Se även

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D

7.5.10 ST_Force3DM

ST_Force3DM — Tvinga geometrierna till XYM-läge.

Synopsis

geometry **ST_Force3DM**(geometry geomA, float Mvalue = 0.0);

Beskrivning

Forcerar geometrierna till XYM-läge. Om en geometri inte har någon M-komponent läggs en M-koordinat med *M-värde* till. Om den har en Z-komponent tas Z bort

Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Force_3DM.

Ändrad: 3.1.0. Stöd för att ange ett M-värde som inte är noll har lagts till.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Nothing happens to an already 3D geometry.

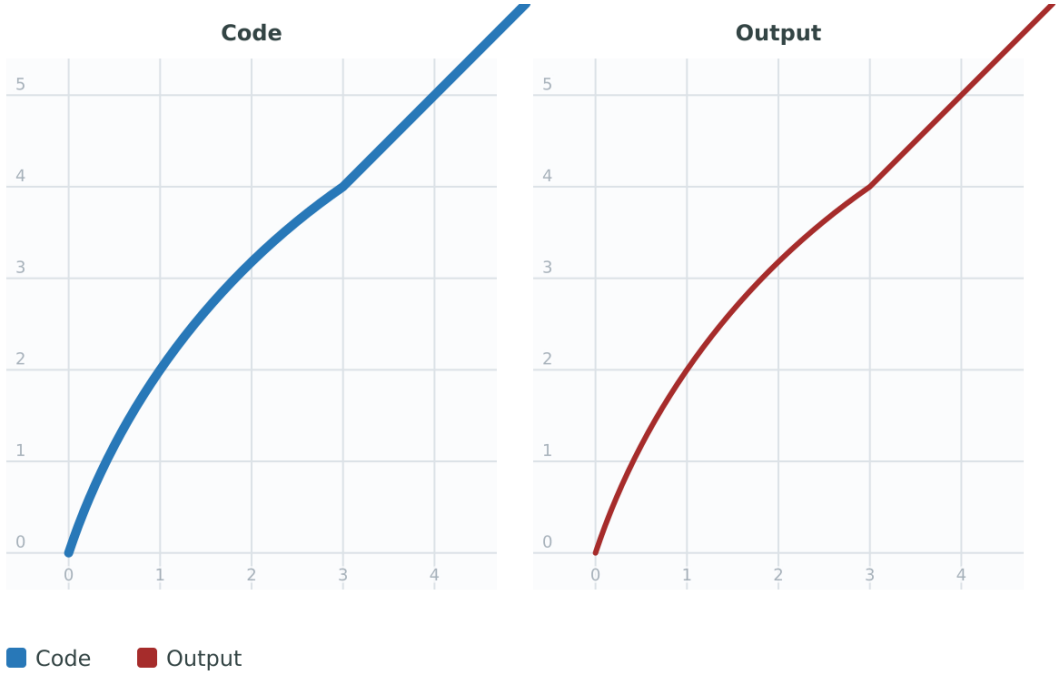
Code

```
SELECT ST_Force3DM('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRINGM(1 1 0,2 3 0,4 5 0,6 7 0,5 6 0)
```

Figure



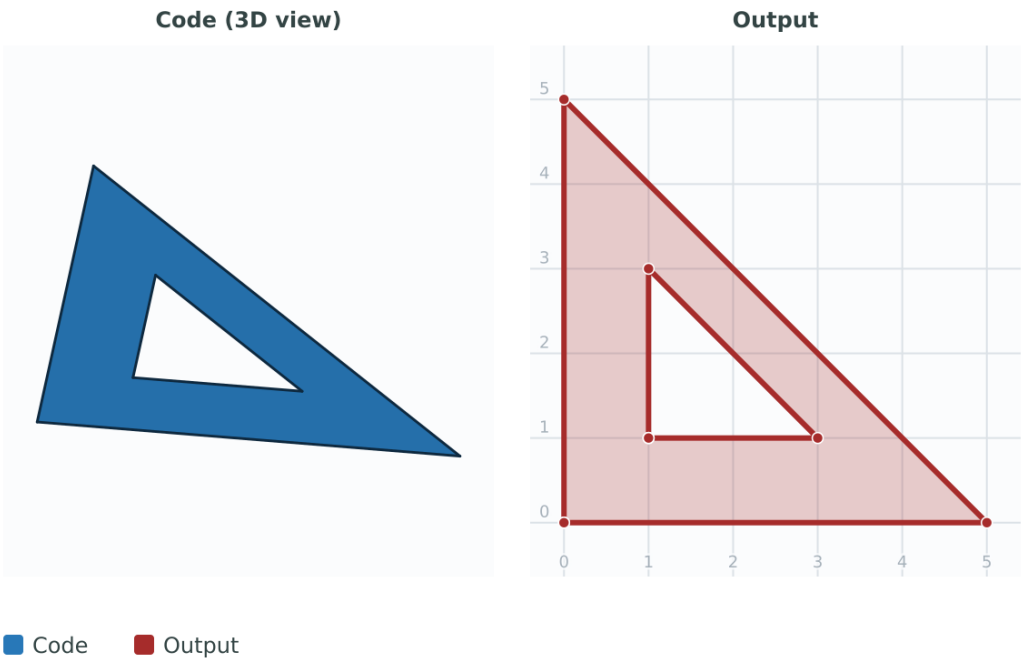
Code

```
SELECT ST_Force3DM('POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))');
```

Output

```
POLYGONM((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0))
```

Figure



Se även

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

7.5.11 ST_Force4D

ST_Force4D — Tvinga geometrierna till XYZM-läge.

Synopsis

geometry **ST_Force4D**(geometry geomA, float Zvalue = 0.0, float Mvalue = 0.0);

Beskrivning

Forcerar geometrierna till XYZM-läge. *`Z-värde`* och *`M-värde`* läggs till för saknade Z- respektive M-dimensioner.

Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Force_4D.

Ändrad: 3.1.0. Stöd för att ange Z- och M-värden som inte är noll har lagts till.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Nothing happens to an already 3D geometry.

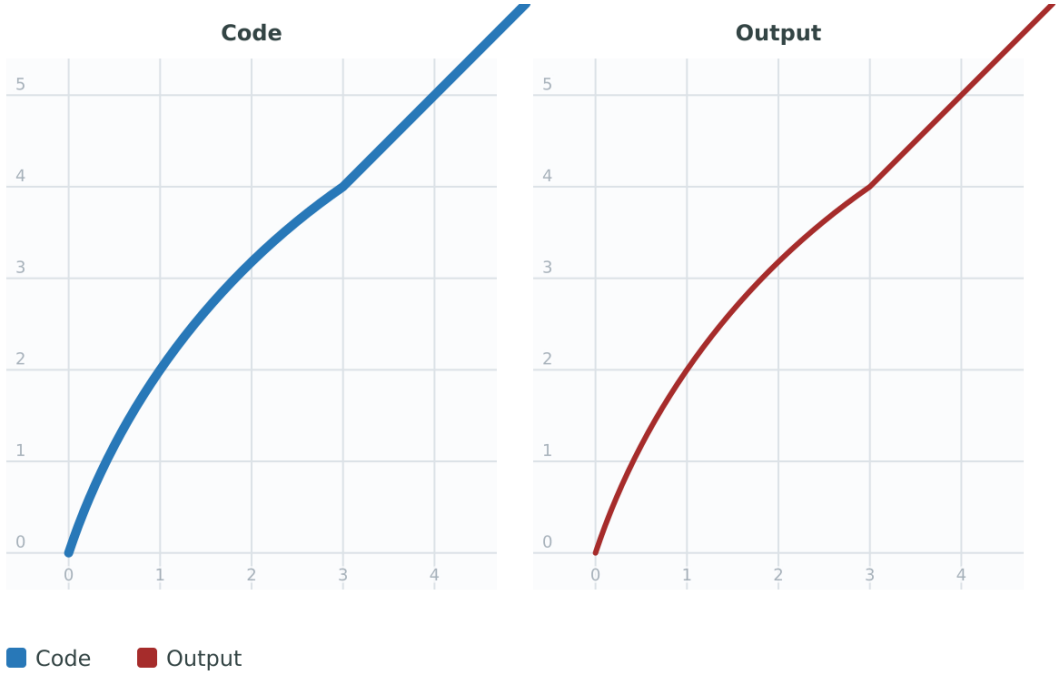
Code

```
SELECT ST_Force4D('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1 2 0,2 3 2 0,4 5 2 0,6 7 2 0,5 6 2 0)
```

Figure



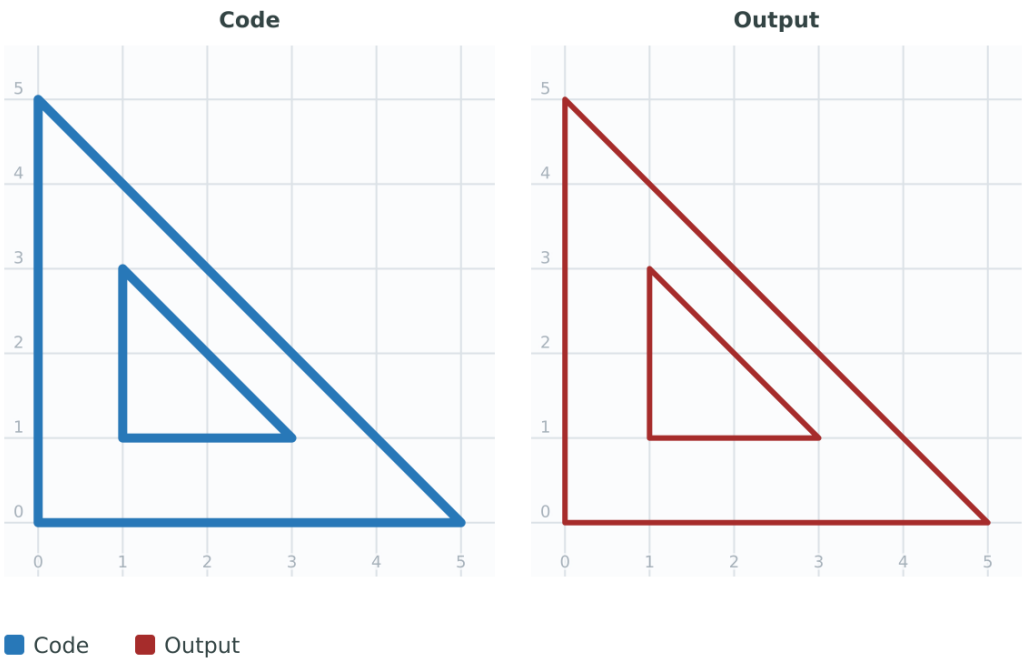
Code

```
SELECT ST_Force4D('MULTILINESTRINGM((0 0 1,0 5 2,5 0 3,0 0 4),(1 1 1,3 1 1,1 3 1,1 1 1))');
```

Output

```
MULTILINESTRING((0 0 0 1,0 5 0 2,5 0 0 3,0 0 0 4),(1 1 0 1,3 1 0 1,1 3 0 1,1 1 0 1))
```

Figure



Se även

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D

7.5.12 ST_ForceCollection

ST_ForceCollection — Konvertera geometrin till en GEOMETRYCOLLECTION.

Synopsis

geometry **ST_ForceCollection**(geometry geomA);

Beskrivning

Konverterar geometrin till en GEOMETRYCOLLECTION. Detta är användbart för att förenkla WKB-representationen.

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Tillgänglighet: 1.2.2, före 1.3.4 kraschar den här funktionen med Curves. Detta är åtgärdat i 1.3.4+

Ändrad: 2.1.0. Fram till 2.0.x hette detta ST_Force_Collection.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

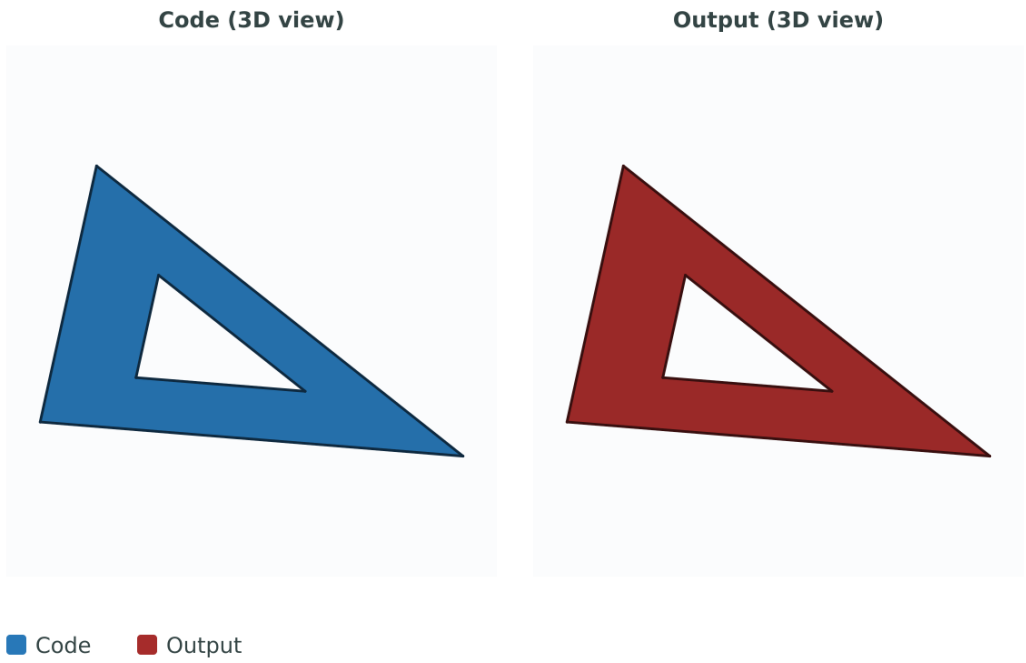
Code

```
SELECT ST_ForceCollection('POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))');
```

Output

```
GEOMETRYCOLLECTION(POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1)))
```

Figure



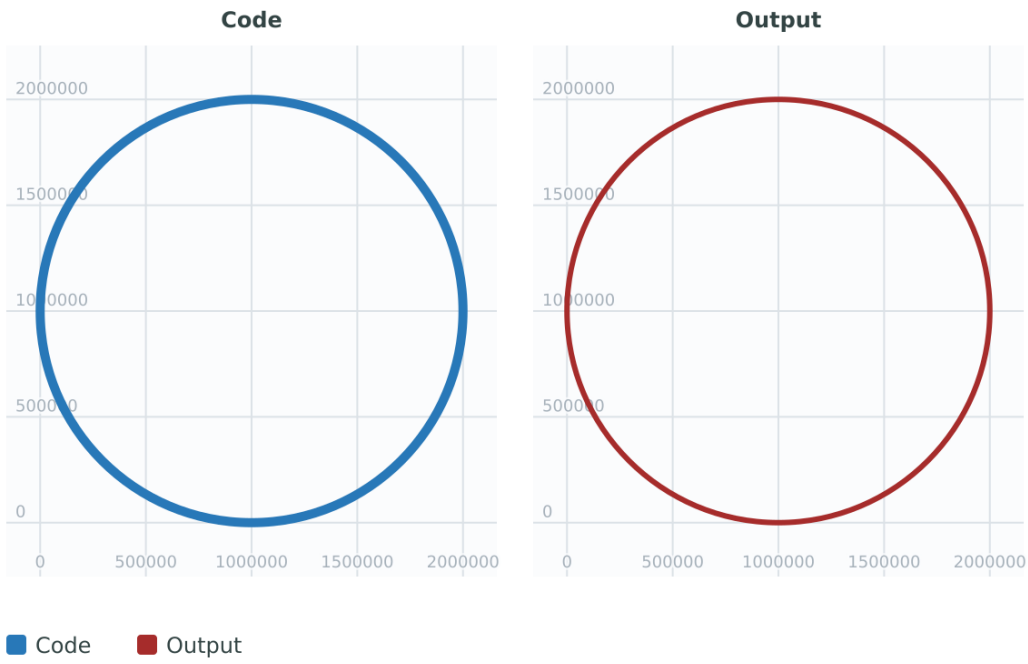
Code

```
SELECT ST_ForceCollection('CIRCULARSTRING(220227 150406,2220227 150407,220227 150406)');
```

Output

```
GEOMETRYCOLLECTION(CIRCULARSTRING(220227 150406,2220227 150407,220227 150406))
```

Figure



This example uses a PolyhedralSurface.

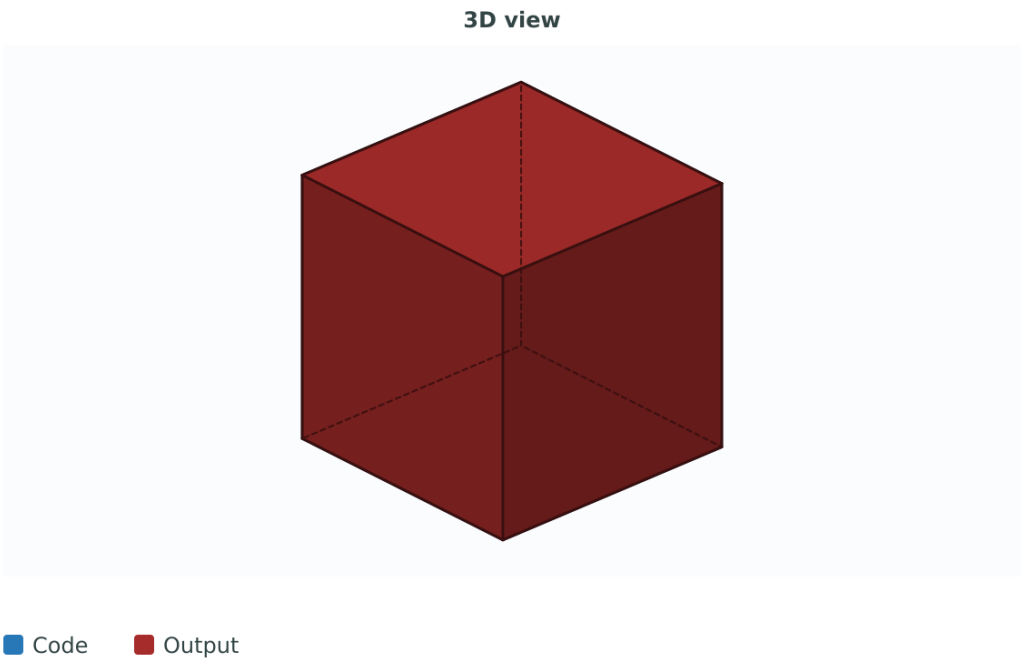
Code

```
SELECT ST_ForceCollection('POLYHEDRALSURFACE(((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
 ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
 ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
 ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
 ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
 ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)))');
```

Output

```
GEOMETRYCOLLECTION(
  POLYGON((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
  POLYGON((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
  POLYGON((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
  POLYGON((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
  POLYGON((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
  POLYGON((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
)
```

Figure



Se även

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

7.5.13 ST_ForceCurve

ST_ForceCurve — Upcasta en geometri till dess kurvade typ, om tillämpligt.

Synopsis

geometry **ST_ForceCurve**(geometry g);

Beskrivning

Omvandlar en geometri till dess krökta representation, om tillämpligt: linjer blir sammansatta kurvor, multilinjer blir multikurvor polygoner blir kurvpolygoner multipolygoner blir multisurfaces. Om den inmatade geometrin redan är en krökt representation returneras samma som inmatningen.

Tillgänglighet: 2.2.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

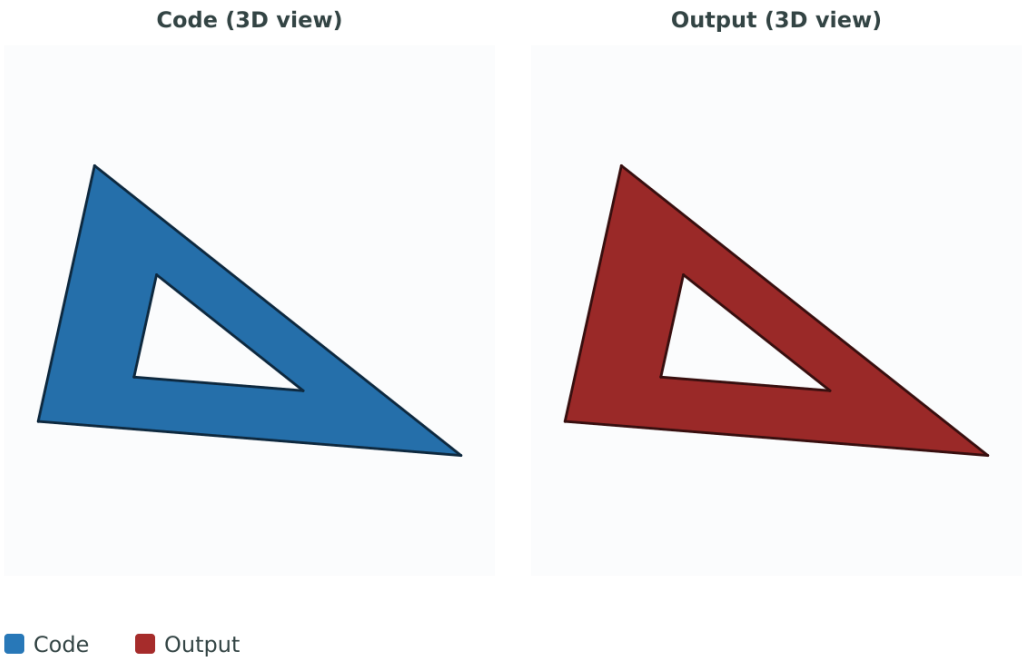
Code

```
SELECT ST_ForceCurve('POLYGON((0 0 2,5 0 2,0 5 2,0 0 2),(1 1 2,1 3 2,3 1 2,1 1 2))':: ↩
    geometry
  );
```

Output

```
CURVEPOLYGON Z ((0 0 2,5 0 2,0 5 2,0 0 2),(1 1 2,1 3 2,3 1 2,1 1 2))
```

Figure



Se även

ST_LineToCurve

7.5.14 ST_ForcePolygonCCW

ST_ForcePolygonCCW — Orienterar alla yttre ringar moturs och alla inre ringar medurs.

Synopsis

geometry **ST_ForcePolygonCCW**(geometry geom);

Beskrivning

Tvingar (multi)polygoner att använda en motsols orientering för sin yttre ring och en medsols orientering för sina inre ringar. Icke-polygonala geometrier returneras oförändrade.

Tillgänglighet: 2.4.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Exempel

Force a polygon and its hole to use counter-clockwise exterior rings and clockwise interior rings.

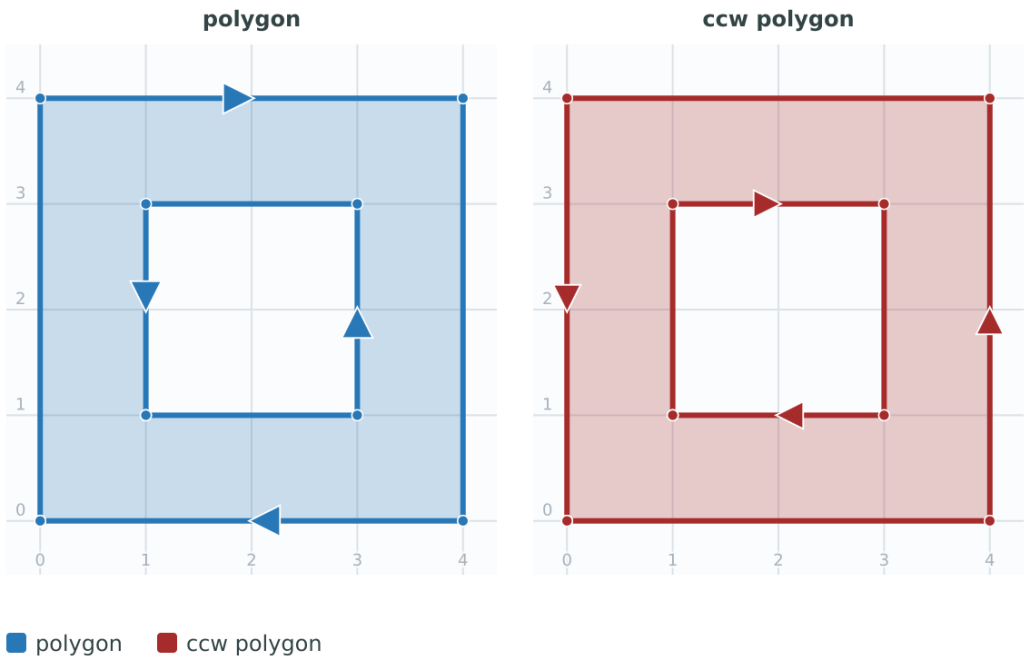
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,0 4,4 4,4 0,0 0),
                  (1 1,3 1,3 3,1 3,1 1))'::geometry AS geom
)
SELECT geom AS input_polygon,
       ST_ForcePolygonCCW(geom) AS ccw_polygon
FROM data;
```

Output

```
POLYGON((0 0,0 4,4 4,4 0,0 0),(1 1,3 1,3 3,1 3,1 1)) | POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,1 3,3 3,3 1,1 1))
```

Figure



Se även

ST_ForcePolygonCW, ST_IsPolygonCCW, ST_IsPolygonCW

7.5.15 ST_ForcePolygonCW

ST_ForcePolygonCW — Orienterar alla yttre ringar medurs och alla inre ringar moturs.

Synopsis

geometry **ST_ForcePolygonCW**(geometry geom);

Beskrivning

Tvingar (Multi)Polygoner att använda en medurs orientering för sin yttre ring och en moturs orientering för sina inre ringar. Icke-polygonala geometrier returneras oförändrade.

Tillgänglighet: 2.4.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Exempel

Force a polygon and its hole to use clockwise exterior rings and counter-clockwise interior rings.

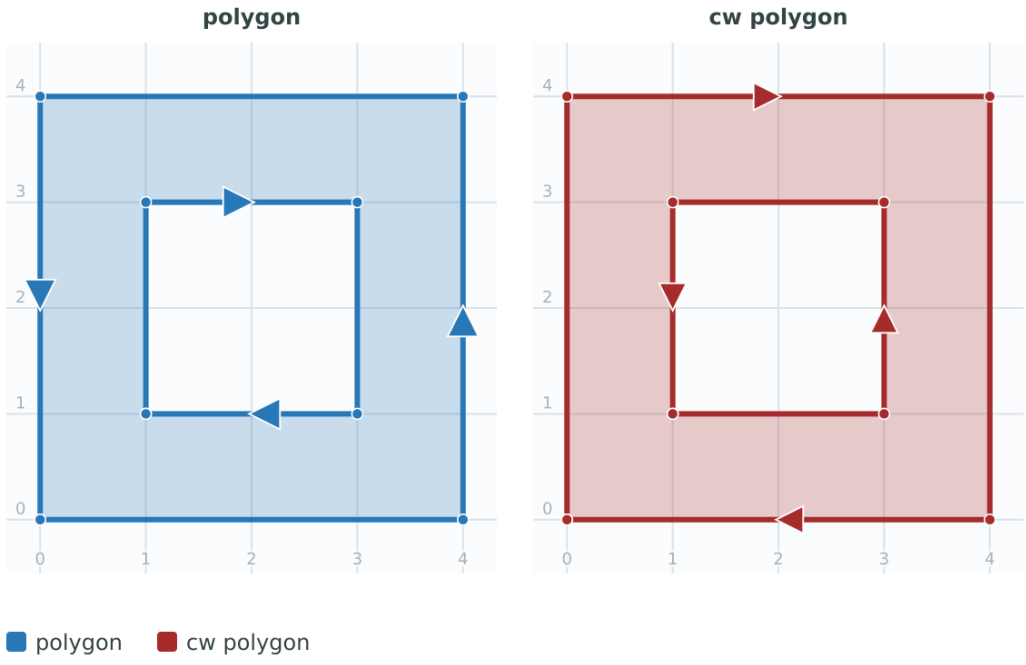
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,4 0,4 4,0 4,0 0),
                  (1 1,1 3,3 3,3 1,1 1))'::geometry AS geom
)
SELECT geom AS input_polygon,
       ST_ForcePolygonCW(geom) AS cw_polygon
FROM data;
```

Output

```
POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,1 3,3 3,3 1,1 1)) | POLYGON((0 0,0 4,4 4,4 0,0 0),(1 1,3 1,3 3,1 3,1 1))
```

Figure



Se även

ST_ForcePolygonCCW, ST_IsPolygonCCW, ST_IsPolygonCW

7.5.16 ST_ForceSFS

ST_ForceSFS — Force geometries to use SFS 1.1 or 1.2 geometry types.

Synopsis

geometry **ST_ForceSFS**(geometry geomA);
geometry **ST_ForceSFS**(geometry geomA, text version);

Beskrivning

Forces a geometry to use the geometry types defined by the requested version of the OGC Simple Features Specification. The default is SFS 1.1. Curved geometries are converted to linear representations, `TRIANGLE` is converted to `POLYGON`, and `TIN` and `POLYHEDRALSURFACE` are converted to `GEOMETRYCOLLECTION` values containing `POLYGON` elements.

If *`version`* begins with `1.2`, curved geometries are still converted to linear representations, but `TRIANGLE`, `TIN`, and `POLYHEDRALSURFACE` are preserved. Other version values use the default SFS 1.1 behavior. Geometry collections are processed recursively, and the SRID and Z/M coordinates are preserved.

Tillgänglighet: 2.1.0

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Compare the geometry types produced for SFS 1.1 (the default) and SFS 1.2. Read each row from the input type in the direction of the result columns.

Code

```
WITH inputs(geom) AS (VALUES
    ('CIRCULARSTRING(0 0, 1 1, 2 0)'::geometry),
    ('TRIANGLE((0 0, 0 2, 2 0, 0 0))'::geometry),
    ('TIN(((0 0, 0 2, 2 0, 0 0)))'::geometry),
    ('POLYHEDRALSURFACE(((0 0, 0 2, 2 0, 0 0)))'::geometry)
)
SELECT ST_GeometryType(geom) || ' b''→b''' AS input,
       ST_GeometryType(ST_ForceSFS(geom)) AS "SFS 1.1 (default)",
       ST_GeometryType(ST_ForceSFS(geom, '1.2')) AS "SFS 1.2"
FROM inputs;
```

Output

```
input          |   SFS 1.1 (default)   |       SFS 1.2
-----------------------+-----------------------+----------------------
 ST_CircularString b''→b''    | ST_LineString         | ST_LineString
 ST_Triangle b''→b''          | ST_Polygon            | ST_Triangle
 ST_Tin b''→b''               | ST_GeometryCollection | ST_Tin
 ST_PolyhedralSurface b''→b'' | ST_GeometryCollection | ST_PolyhedralSurface
(4 rows)
```

Se även

ST_CurveToLine, ST_ForceCurve

7.5.17 ST_ForceRHR

ST_ForceRHR — Tvinga orienteringen av hörnen i en polygon att följa högerhands-regeln.

Synopsis

geometry **ST_ForceRHR**(geometry g);

Beskrivning

Tvingar orienteringen av hörnen i en polygon att följa en högerhandsregel, där det område som avgränsas av polygonen ligger till höger om gränsen. I synnerhet är den yttre ringen orienterad i medurs riktning och de inre ringarna i moturs riktning. Denna funktion är en synonym för ST_ForcePolygonCW

Notera
Ovanstående definition av högerhandsregeln står i konflikt med definitioner som används i andra sammanhang. För att undvika förvirring rekommenderas att man använder ST_ForcePolygonCW.

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Exempel

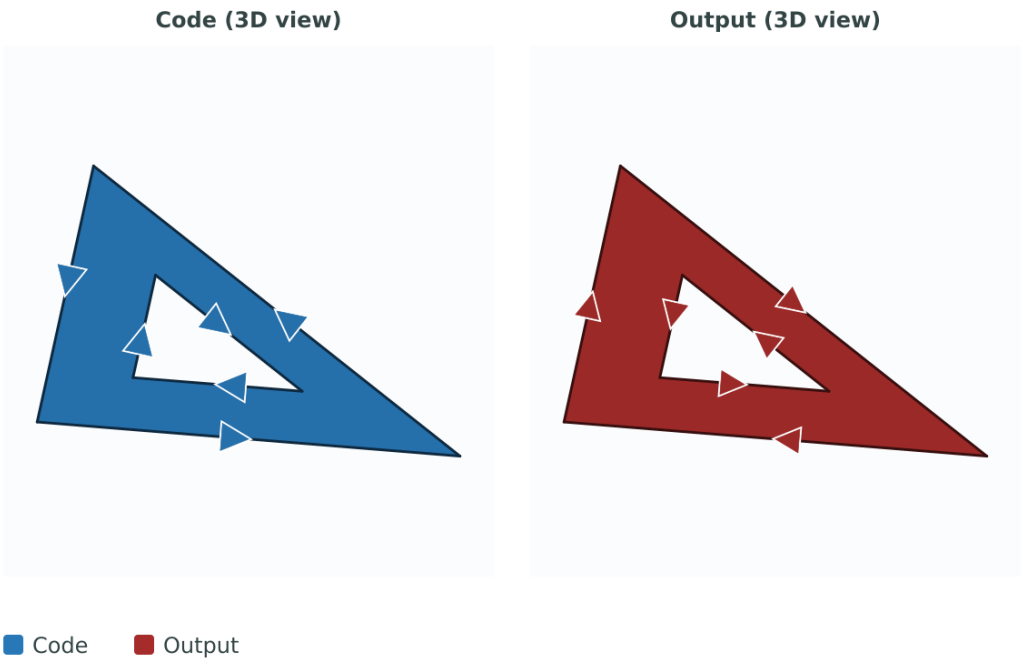
Code

```
SELECT ST_ForceRHR('POLYGON((0 0 2,5 0 2,0 5 2,0 0 2),(1 1 2,1 3 2,3 1 2,1 1 2))'
  );
```

Output

```
POLYGON((0 0 2,0 5 2,5 0 2,0 0 2),(1 1 2,3 1 2,1 3 2,1 1 2))
```

Figure



Se även

ST_ForcePolygonCCW, ST_ForcePolygonCW, ST_IsPolygonCCW, ST_IsPolygonCW, ST_BuildArea, ST_Polygo
ST_Reverse

7.5.18 ST_LineExtend

ST_LineExtend — Returnerar en linje som sträcker sig framåt och bakåt med angivna avstånd.

Synopsis

geometry **ST_LineExtend**(geometry line, float distance_forward, float distance_backward=0.0);

Beskrivning

Returnerar en linje som förlängts framåt och bakåt genom att lägga till nya start- (och slut-) punkter på det eller de angivna avstånden. Ett avstånd på noll innebär inte att en punkt läggs till. Endast icke-negativa avstånd är tillåtna. Riktningen för den/de tillagda punkten/punkterna bestäms av de två första (och sista) distinkta punkterna på linjen. Duplicerade punkter ignoreras.

Tillgänglighet: 3.4.0

Exempel: Förlänger en linje 5 enheter framåt och 6 enheter bakåt

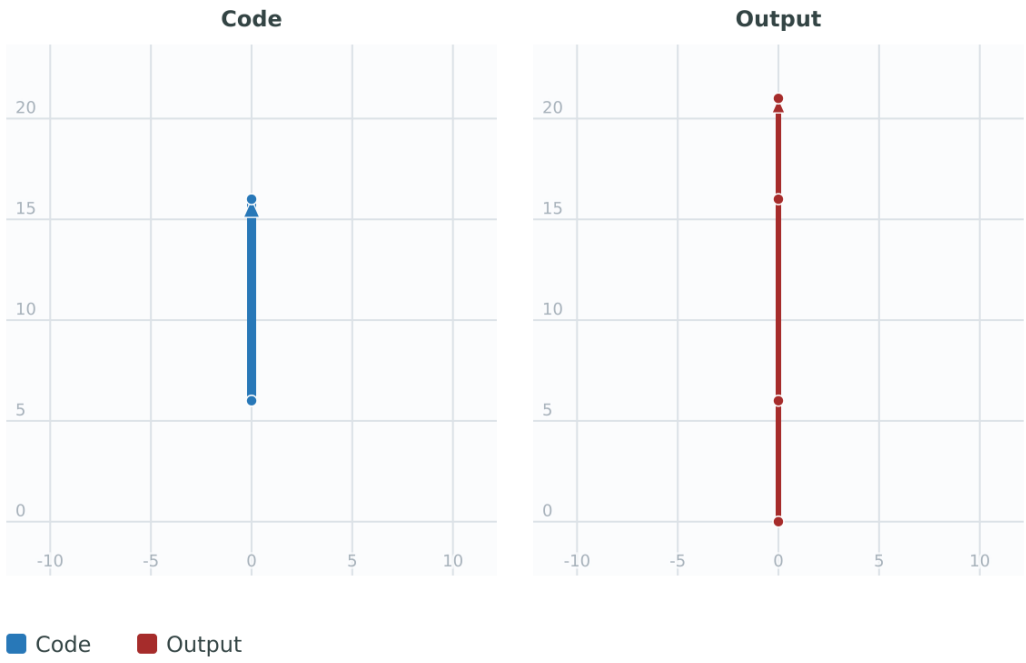
Code

```
SELECT ST_LineExtend('LINESTRING(0 0,0 10)'::geometry, 5, 6);
```

Output

```
LINESTRING(0 -6,0 0,0 10,0 15)
```

Figure



Se även

ST_LineSubstring, ST_LocateAlong, ST_Project

7.5.19 ST_LineToCurve

ST_LineToCurve — Konverterar en linjär geometri till en krökt geometri.

Synopsis

geometry **ST_LineToCurve**(geometry geomANoncircular);

Beskrivning

Konverterar vanlig LINESTRING/POLYGON till CIRKULÄRA STRINGAR och krökta polygoner. Observera att mycket färre punkter behövs för att beskriva den krökta motsvarigheten.

Notera
Om indata LINESTRING/POLYGON inte är tillräckligt krökt för att tydligt representera en kurva, kommer funktionen att returnera samma indatageometri.

Tillgänglighet: 1.3.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

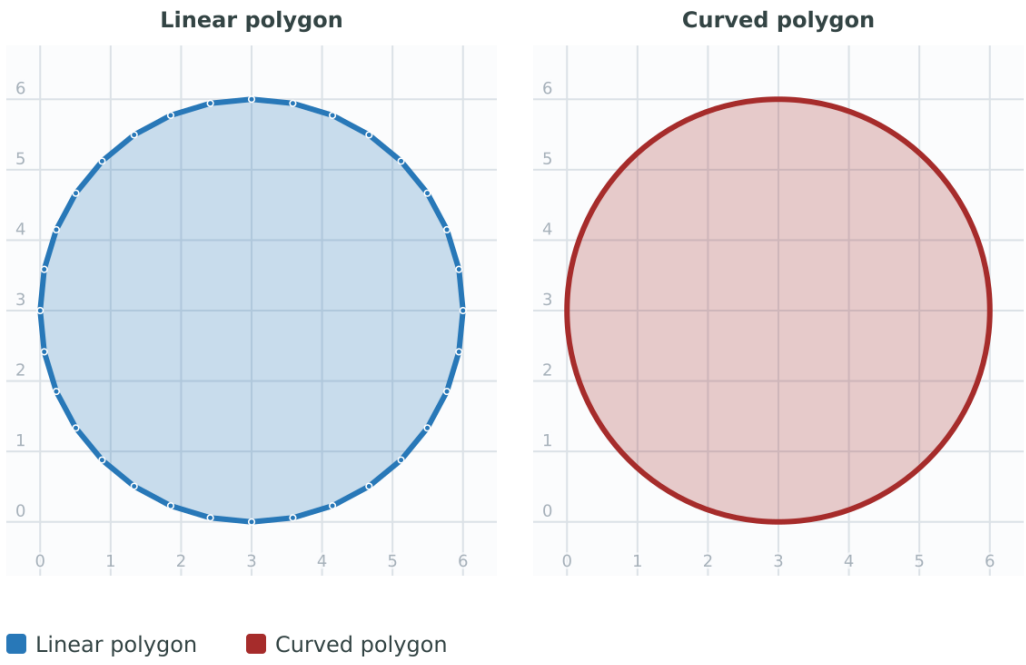
Exempel

This example converts a linear Polygon approximating a circle to a CurvePolygon.

Code

```
WITH data AS (
    SELECT ST_Buffer('POINT(1 3)'::geometry, 3) AS geom
)
SELECT geom AS "input_Linear polygon",
       ST_LineToCurve(geom) AS "Curved polygon"
FROM data;
```

Figure

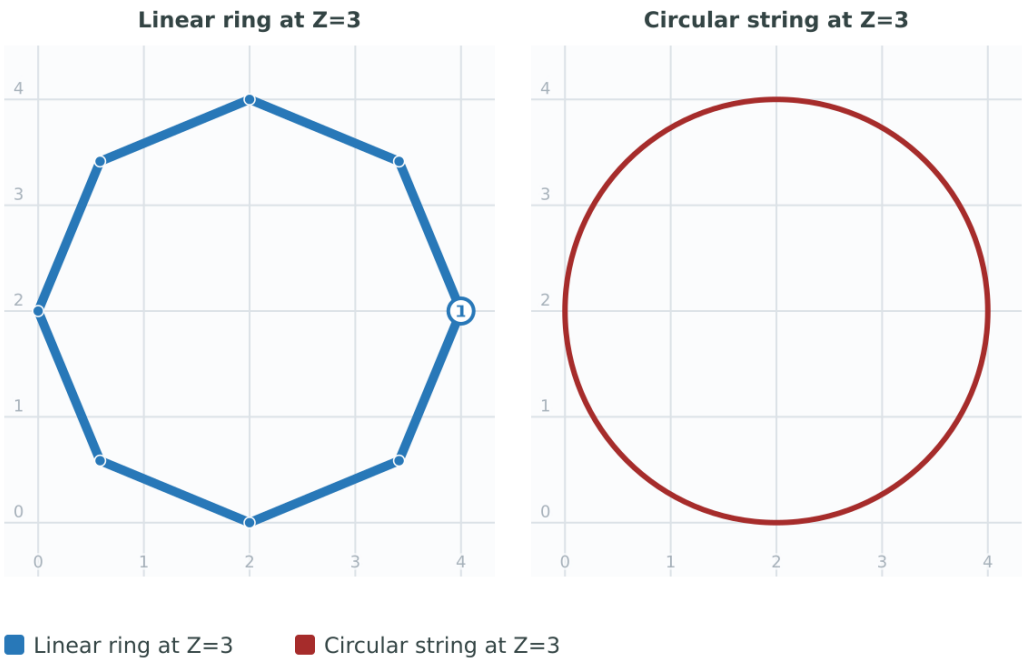


This example converts a 3D linear ring to a CircularString while preserving Z.

Code

```
WITH data AS (
    SELECT ST_Translate(
        ST_Force3D(ST_Boundary(ST_Buffer(ST_Point(1, 3), 2, 2))),
        0, 0, 3
    ) AS geom
)
SELECT geom AS "input_Linear ring at Z=3",
       ST_LineToCurve(geom) AS "Circular string at Z=3"
FROM data;
```

Figure



Se även

ST_CurveToLine

7.5.20 ST_Multi

ST_Multi — Returnera geometrin som en MULTI*-geometri.

Synopsis

geometry **ST_Multi**(geometry geom);

Beskrivning

Returnerar geometrin som en MULTI* geometrisamling. Om geometrin redan är en samling returneras den oförändrad.

Exempel

A Polygon is converted to a single-element MultiPolygon. The coordinates are unchanged.

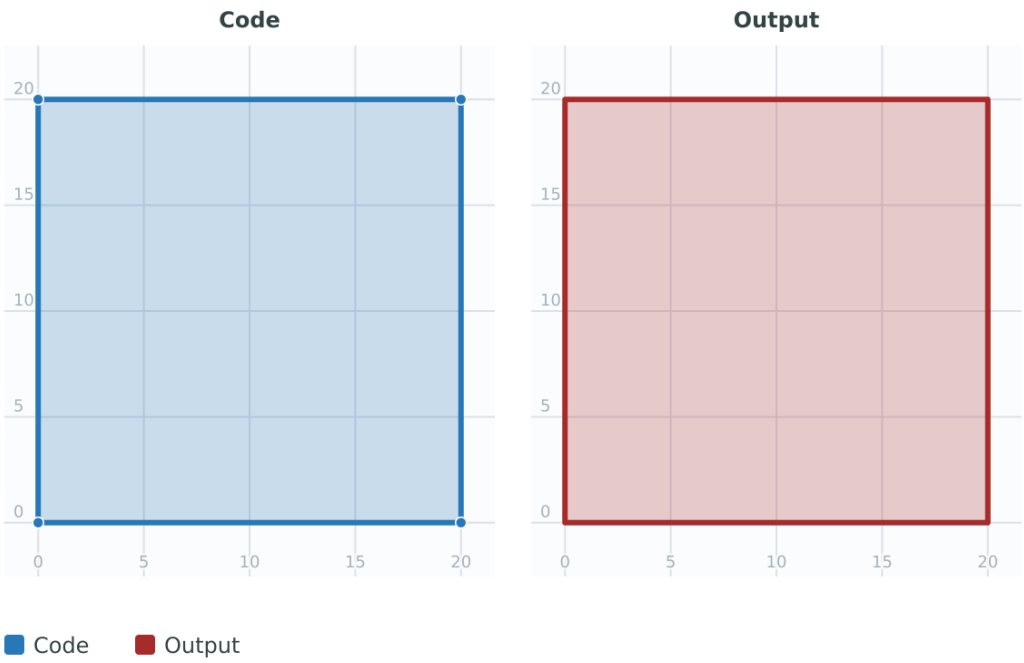
Code

```
SELECT ST_Multi('POLYGON ((10 30,30 30,30 10,10 10,10 30))');
```

Output

```
MULTIPOLYGON(((10 30,30 30,30 10,10 10,10 30)))
```

Figure



Se även

ST_AsText

7.5.21 ST_Normalize

ST_Normalize — Returnera geometrin i dess kanoniska form.

Synopsis

geometry **ST_Normalize**(geometry geom);

Beskrivning

Returnerar geometrin i dess normaliserade/kanoniska form. Kan omordna hörn i polygonringar, ringar i en polygon, element i ett komplex med flera geometrier.

Oftast endast användbar för teständamål (jämförelse mellan förväntade och erhållna resultat).

Tillgänglighet: 2.3.0

Exempel

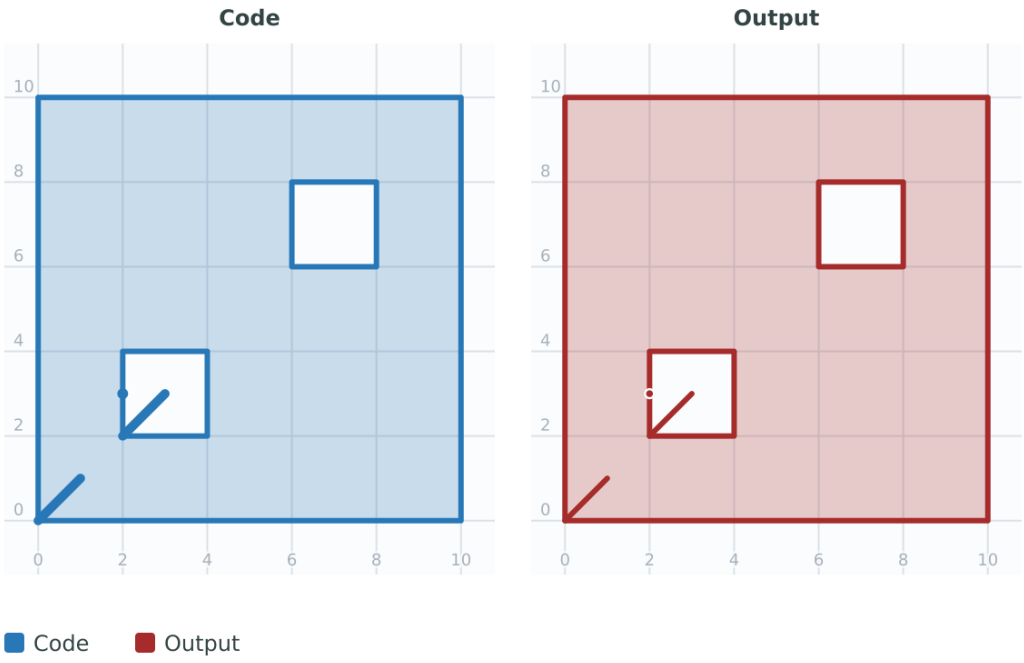
Code

```
SELECT ST_Normalize('GEOMETRYCOLLECTION(
    POINT(2 3),
    MULTILINESTRING((0 0,1 1),(2 2,3 3)),
    POLYGON(
      (0 10,0 0,10 0,10 10,0 10),
      (4 2,2 2,2 4,4 4,4 2),
      (6 8,8 8,8 6,6 6,6 8)
    )
  )');
```

Output

```
GEOMETRYCOLLECTION(POLYGON((0 0,0 10,10 10,10 0,0 0),(6 6,8 6,8 8,6 8,6 6),(2 2,4 2,4 4,2 4,2 2)), MULTILINESTRING((2 ↩
    2,3 3),(0 0,1 1)), POINT(2 3))
```

Figure



Se även

ST_Equals,

7.5.22 ST_Project

ST_Project — Returnerar en punkt som projiceras från en startpunkt med ett avstånd och en bäring (azimut).

Synopsis

geometry **ST_Project**(geometry g1, float distance, float azimuth);
geometry **ST_Project**(geometry g1, geometry g2, float distance);
geography **ST_Project**(geography g1, float distance, float azimuth);
geography **ST_Project**(geography g1, geography g2, float distance);

Beskrivning

Returnerar en punkt som projiceras från en punkt längs en geodetisk linje med hjälp av ett givet avstånd och azimut (bäring). Detta är känt som det direkta geodetiska problemet.

I tvåpunktsversionen används vägen från den första till den andra punkten för att implicit definiera azimut och avståndet används som tidigare.

Avståndet anges i meter. Negativa värden stöds.

Azimuten (även känd som kurs eller bäring) anges i radianer. Den mäts medurs från nordlig riktning.

- Norr är azimut noll (0 grader)
- Öst är azimut π/2 (90 grader)
- Syd är azimut π (180 grader)
- Väst är azimut 3π/2 (270 grader)

Negativa azimutvärden och värden som är större än 2π (360 grader) stöds.

Tillgänglighet: 2.0.0

Förbättrad: 2.4.0 Tillåt negativt avstånd och icke-normaliserad azimut.

Förbättrad: 3.4.0 Tillåt geometriargument och tvåpunktsform som utelämnar azimut.

Exempel: Projicerad punkt på 100.000 meter och bäring 45 grader

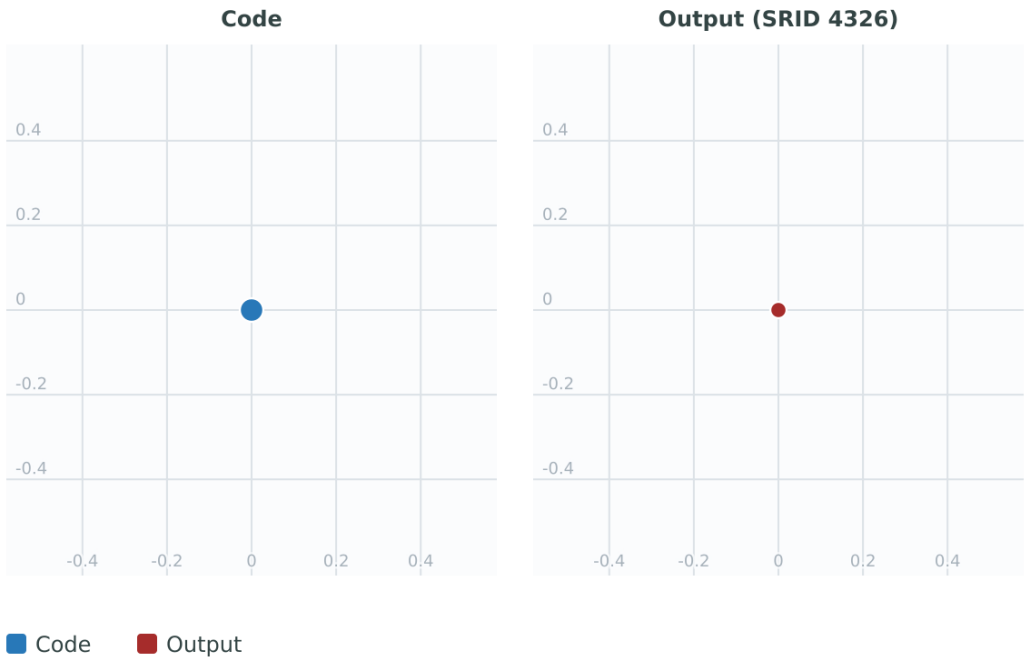
Code

```
SELECT ST_Project('POINT(0 0)'::geography, 100000, radians(45.0));
```

Output

```
POINT(0.635231029125537 0.639472334729198)
```

Figure



Se även

ST_Azimuth, ST_Distance, PostgreSQL-funktion radians()

7.5.23 ST_QuantizeCoordinates

ST_QuantizeCoordinates — Sätter koordinaternas minst signifikanta bitar till noll

Synopsis

geometry **ST_QuantizeCoordinates**(geometry g, int prec_x, int prec_y = NULL, int prec_z = NULL, int prec_m = NULL);

Beskrivning

`ST_QuantizeCoordinates` bestämmer antalet bitar`(N)` som krävs för att representera ett koordinatvärde med ett angivet antal siffror efter decimalpunkten, och sätter sedan alla utom de `N` mest signifikanta bitarna till noll. Det resulterande koordinatvärdet kommer fortfarande att avrundas till det ursprungliga värdet, men har förbättrad komprimerbarhet. Detta kan resultera i en betydande minskning av diskanvändningen, förutsatt att geometrikolumnen använder en komprimerbar lagringstyp. Funktionen tillåter specifikation av olika antal siffror efter decimaltecknet i varje dimension; ospecificerade dimensioner antas ha samma precision som `x-dimensionen`. Negativa siffror tolkas som siffror till vänster om decimaltecknet (t.ex. `prec_x=-2` bevarar koordinatvärden till närmaste 100-tal).

De koordinater som produceras av `ST_QuantizeCoordinates` är oberoende av den geometri som innehåller dessa koordinater och av koordinaternas relativa position inom geometrin. Detta innebär att befintliga topologiska relationer mellan geometrier inte påverkas av användningen av denna funktion. Funktionen kan ge upphov till ogiltig geometri när den anropas med ett antal siffror som är lägre än geometrins egen precision.

Tillgänglighet: 2.5.0

Teknisk bakgrund

PostGIS lagrar alla koordinatvärden som flyttalsintegraler med dubbel precision, som på ett tillförlitligt sätt kan representera 15 signifikanta siffror. PostGIS kan dock användas för att hantera data som i sig har färre än 15 signifikanta siffror. Ett exempel är TIGER-data, som tillhandahålls som geografiska koordinater med sex precisionssiffror efter decimaltecknet (vilket innebär att det endast krävs nio signifikanta siffror för longitud och åtta signifikanta siffror för latitud)

När 15 signifikanta siffror är tillgängliga finns det många möjliga representationer av ett tal med 9 signifikanta siffror. Ett flyttal med dubbel precision använder 52 explicita bitar för att representera koordinatens signifikand (mantissa). Endast 30 bitar behövs för att representera en mantissa med 9 signifikanta siffror, vilket ger 22 obetydliga bitar; vi kan ställa in deras värde till vad vi vill och ändå sluta med ett tal som avrundas till vårt inmatningsvärde. Till exempel kan värdet 100,123456 representeras av de flyttal som ligger närmast 100,123456000000, 100,123456000001 och 100,123456432199. Alla är lika giltiga, eftersom `ST_AsText(geom, 6)` kommer att returnera samma resultat med någon av dessa indata. Eftersom vi kan sätta dessa bitar till vilket värde som helst, sätter `ST_QuantizeCoordinates` de 22 obetydliga bitarna till noll. För en lång koordinatsekvens skapar detta ett mönster av block med på varandra följande nollor som komprimeras av PostgreSQL mer effektivt.

Notera
Endast geometrins storlek på disken påverkas potentiellt av `ST_QuantizeCoordinates`. ST_MemSize, som rapporterar geometrins användning i minnet, kommer att returnera samma värde oavsett hur mycket diskutrymme som används av en geometri.

Exempel

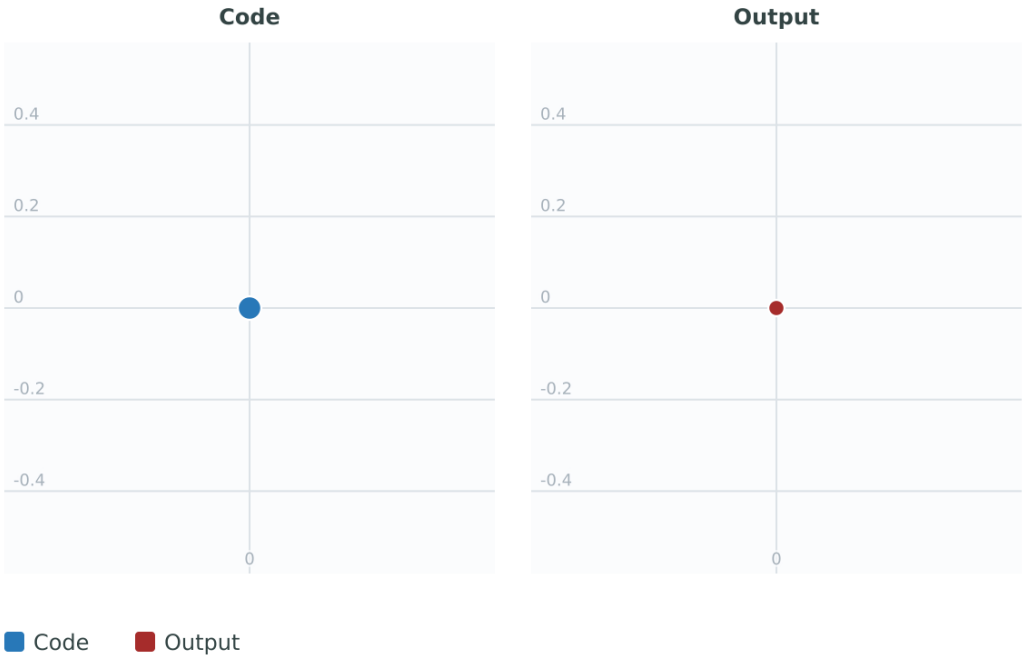
Code

```
SELECT ST_QuantizeCoordinates('POINT (100.123456 0)'::geometry, 4);
```

Output

```
POINT(100.12344360351562 0)
```

Figure



Code

```
WITH test AS (SELECT 'POINT (123.456789123456 123.456789123456)'::geometry AS geom)
SELECT
  digits,
  encode(ST_QuantizeCoordinates(geom, digits), 'hex'),
  ST_QuantizeCoordinates(geom, digits)
FROM test, generate_series(15, -15, -1) AS digits;
```

Output

```
digits |                  encode                  |                 st_astext
--------+------------------------------------------+-------------------------------------------
     15 | 01010000005f9a72083cdd5e405f9a72083cdd5e40 | POINT(123.456789123456 123.456789123456)
     14 | 01010000005f9a72083cdd5e405f9a72083cdd5e40 | POINT(123.456789123456 123.456789123456)
     13 | 01010000005e9a72083cdd5e405e9a72083cdd5e40 | POINT(123.45678912345599 123.45678912345599)
     12 | 0101000000409a72083cdd5e40409a72083cdd5e40 | POINT(123.45678912345556 123.45678912345556)
     11 | 0101000000009a72083cdd5e40009a72083cdd5e40 | POINT(123.45678912345466 123.45678912345466)
     10 | 0101000000009872083cdd5e40009872083cdd5e40 | POINT(123.45678912344738 123.45678912344738)
      9 | 0101000000008072083cdd5e40008072083cdd5e40 | POINT(123.45678912336007 123.45678912336007)
      8 | 0101000000000070083cdd5e40000070083cdd5e40 | POINT(123.45678912103176 123.45678912103176)
      7 | 0101000000000060083cdd5e40000060083cdd5e40 | POINT(123.4567891061306 123.4567891061306)
      6 | 0101000000000000083cdd5e40000000083cdd5e40 | POINT(123.45678901672363 123.45678901672363)
      5 | 0101000000000000003cdd5e40000000003cdd5e40 | POINT(123.456787109375 123.456787109375)
      4 | 0101000000000000003cdd5e40000000003cdd5e40 | POINT(123.456787109375 123.456787109375)
      3 | 0101000000000000038dd5e400000000038dd5e40 | POINT(123.45654296875 123.45654296875)
      2 | 0101000000000000000dd5e400000000000dd5e40 | POINT(123.453125 123.453125)
      1 | 0101000000000000000dc5e400000000000dc5e40 | POINT(123.4375 123.4375)
      0 | 0101000000000000000c05e400000000000c05e40 | POINT(123 123)
     -1 | 010100000000000000000005e4000000000000005e40 | POINT(120 120)
```

```
     -2 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -3 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -4 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -5 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -6 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -7 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -8 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -9 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -10 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -11 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -12 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -13 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -14 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -15 | 0101000000000000000000058400000000000005840 | POINT(96 96)
(31 rows)
```

Se även

ST_SnapToGrid

7.5.24 ST_RemovePoint

ST_RemovePoint — Ta bort en punkt från en linestrings.

Synopsis

geometry **ST_RemovePoint**(geometry linestring, integer offset);

Beskrivning

Tar bort en punkt från en LineString, givet dess index (0-baserat). Användbar för att förvandla en sluten linje (ring) till en öppen linestrings.

Förbättrad: 3.2.0

Tillgänglighet: 1.1.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Garanterar att inga linjer är slutna genom att ta bort slutpunkten för slutna linjer (ringar). Förutsätter att geom är av typen LINESTRING

Code

```
UPDATE sometable
    SET geom = ST_RemovePoint(geom, ST_NPoints(geom) - 1)
    FROM sometable
    WHERE ST_IsClosed(geom);
```

Se även

ST_AddPoint, ST_NPoints, ST_NumPoints

7.5.25 ST_RemoveRepeatedPoints

ST_RemoveRepeatedPoints — Returnerar en version av en geometri där dubbletter av punkter har tagits bort.

Synopsis

geometry **ST_RemoveRepeatedPoints**(geometry geom, float8 tolerance = 0.0);

Beskrivning

Returnerar en version av den angivna geometrin där dubbla punkter har tagits bort. Funktionen bearbetar endast (Multi)LineStrings, (Multi)Polygons och MultiPoints men den kan anropas med alla typer av geometrier. Element i GeometryCollections bearbetas individuellt. Slutpunkterna för LineStrings bevaras.

Om en *`toleransparameter`* som inte är noll anges, anses vertikaler som ligger inom toleransavståndet från varandra vara duplikat. Avståndet beräknas i 2D (XY-planet).

Förbättrad: 3.2.0

Tillgänglighet: 2.2.0

Denna funktion stöder polyedriska ytor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

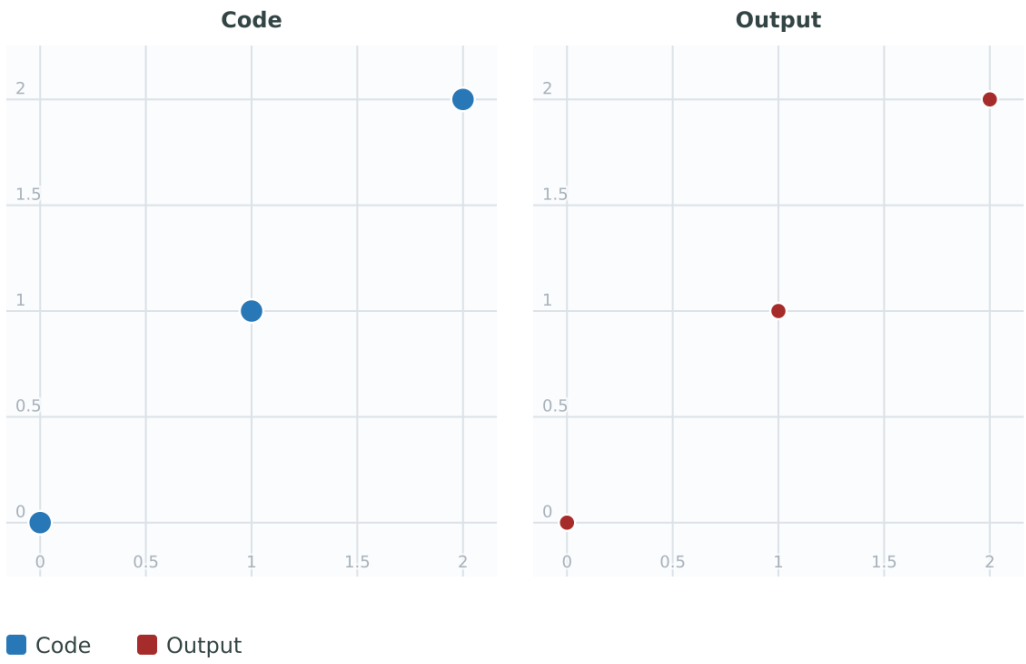
Code

```
SELECT ST_RemoveRepeatedPoints('MULTIPOINT ((1 1),(2 2),(3 3),(2 2))');
```

Output

```
MULTIPOINT((1 1),(2 2),(3 3))
```

Figure



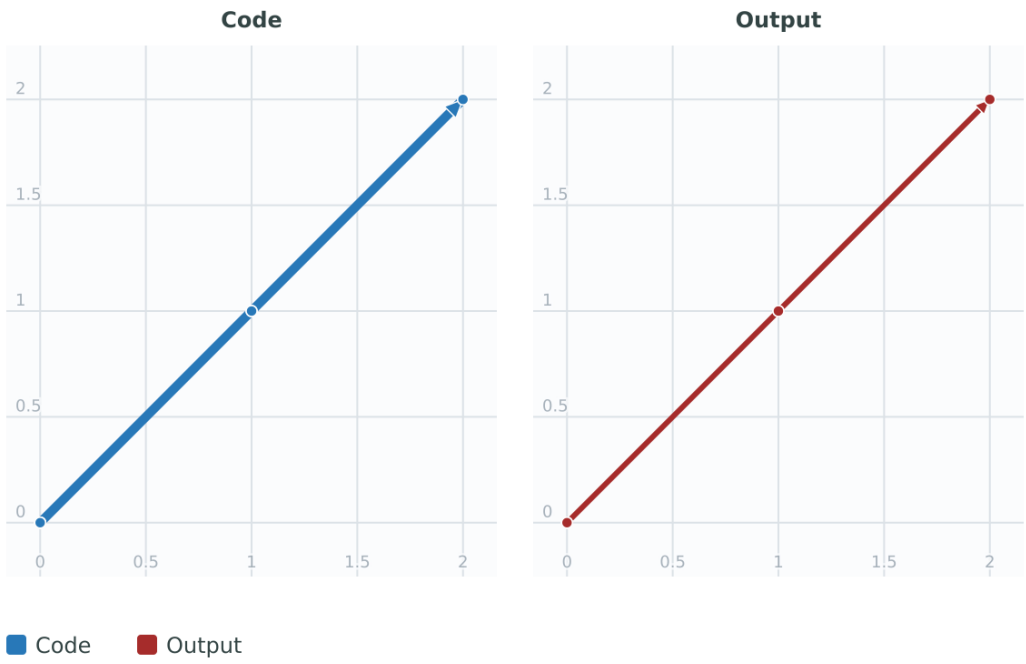
Code

```
SELECT ST_RemoveRepeatedPoints('LINESTRING (0 0,0 0,1 1,0 0,1 1,2 2)');
```

Output

```
LINESTRING(0 0,1 1,0 0,1 1,2 2)
```

Figure



Exempel: Samlingselement bearbetas individuellt.

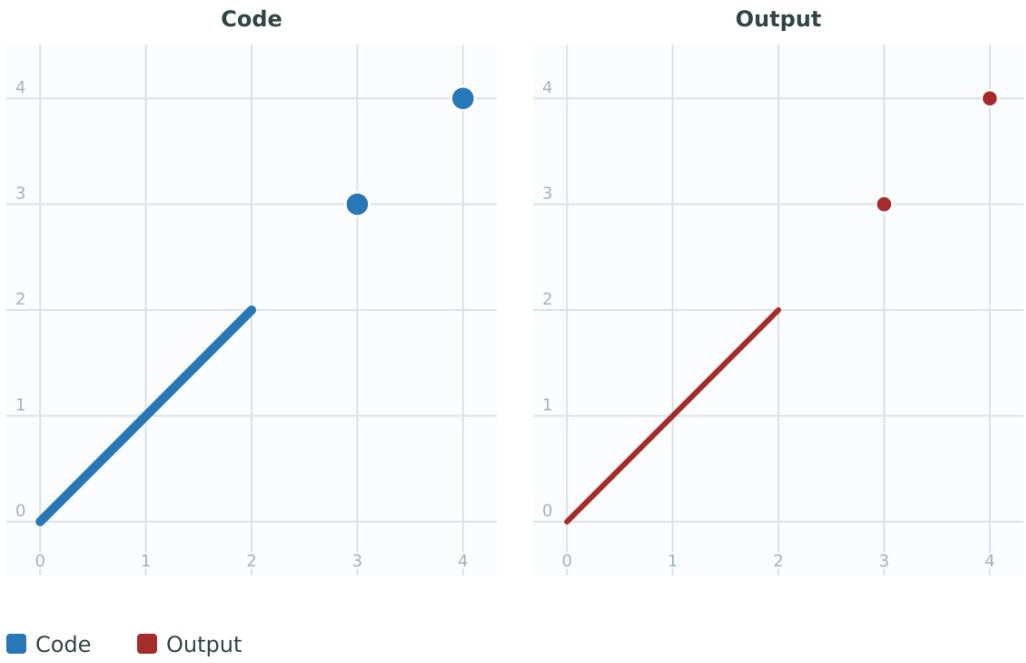
Code

```
SELECT ST_RemoveRepeatedPoints('GEOMETRYCOLLECTION (LINESTRING (1 1,2 2,2 2,3 3),POINT (4 ↩
    4),POINT (4 4),POINT (5 5))');
```

Output

```
GEOMETRYCOLLECTION(LINESTRING(1 1,2 2,3 3), POINT(4 4), POINT(4 4), POINT(5 5))
```

Figure



Exempel: Upprepad punktborttagning med en avståndstolerans.

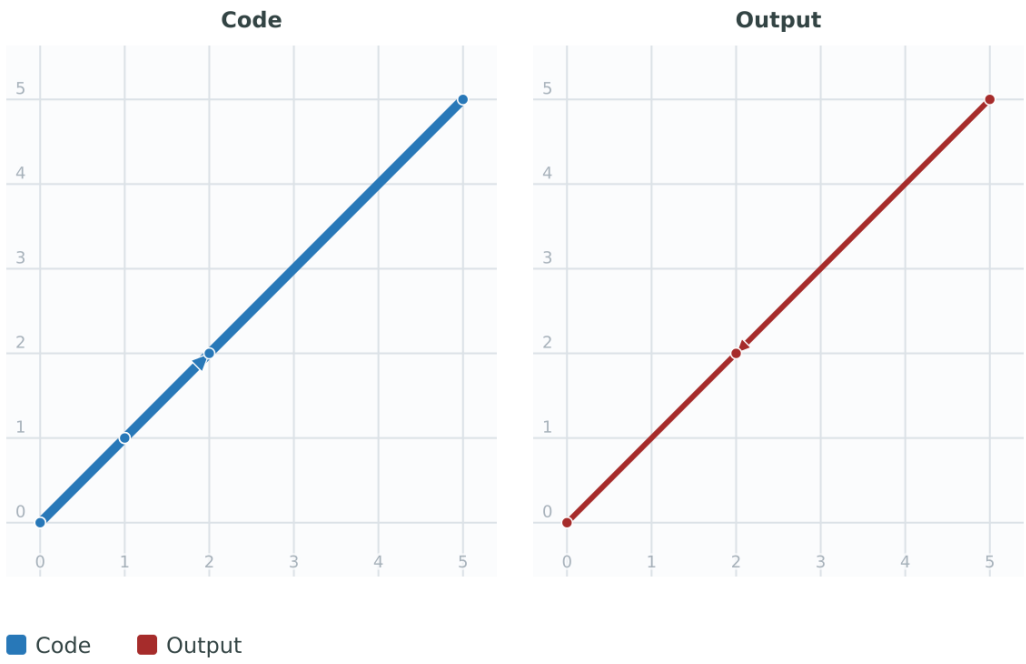
Code

```
SELECT ST_RemoveRepeatedPoints('LINESTRING (0 0,0 0,1 1,5 5,1 1,2 2)', 2);
```

Output

```
LINESTRING(0 0,5 5,2 2)
```

Figure



Se även

ST_Simplify

7.5.26 ST_RemoveIrrelevantPointsForView

ST_RemoveIrrelevantPointsForView — Tar bort punkter som är irrelevanta för rendering av en specifik rektangulär vy av en geometri.

Synopsis

geometry **ST_RemoveIrrelevantPointsForView**(geometry geom, box2d bounds, boolean cartesian_hint = false);

Beskrivning

Returnerar en geometry utan punkter som är irrelevanta för rendering av geometrin inom en given rektangulär vy.

Denna funktion kan användas för att snabbt förbehandla geometrier som endast ska återges inom vissa gränser.

Endast geometrier av typen (MULTI)POLYGON och (MULTI)LINESTRING utvärderas. Övriga geometrier förblir oförändrade.

I motsats till `ST_ClipByBox2D()` är denna funktion

- sorterar ut punkter utan att beräkna nya skärningspunkter, vilket undviker avrundningsfel och vanligtvis ökar prestandan,

- returnerar en geometri med samma eller liknande punktnummer,
- leder till samma renderingsresultat inom den angivna vyn, och
- kan ge upphov till självskärningar som skulle göra den resulterande geometrin ogiltig (se exempel nedan).

Om `cartesian_hint` är satt till `true` tillämpar algoritmen ytterligare optimeringar som involverar cartesisk matematik för att ytterligare minska det resulterande punktantalet. Observera att om du använder det här alternativet kan det leda till renderingsartefakter om de resulterande koordinaterna projiceras till ett annat (icke-kartesiskt) koordinatsystem före rendering.

Varning
För polygoner säkerställer denna funktion för närvarande inte att resultatet är giltigt. Denna situation kan kontrolleras med ST_IsValid och repareras med ST_MakeValid.

Tillgänglighet: 3.5.0

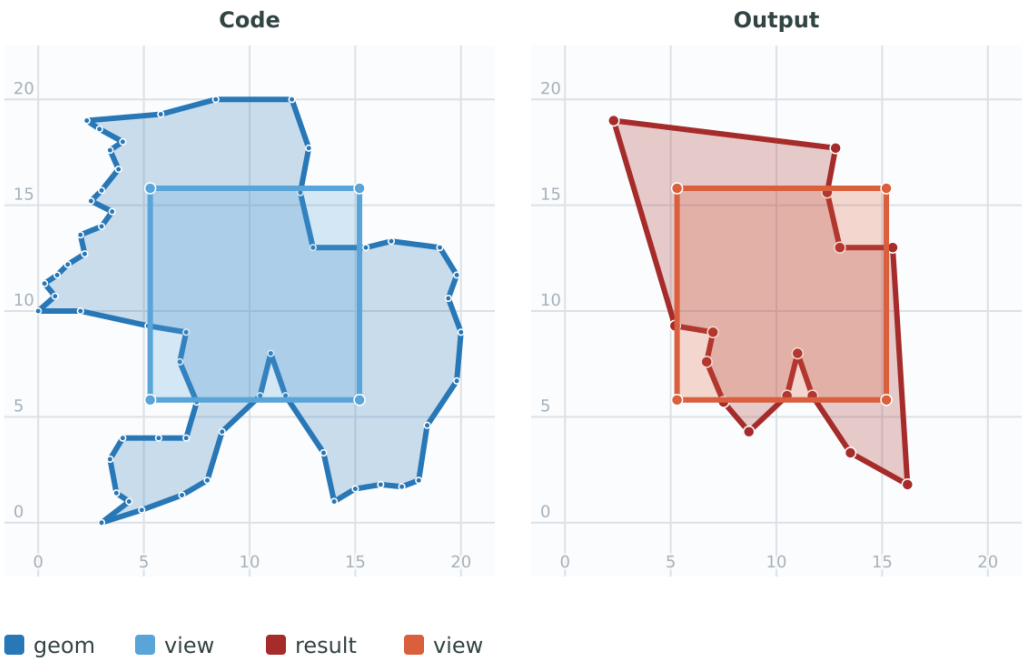
Exempel

The input polygon has 55 points. The result retains 15 points, while rendering the same shape inside the view envelope.

Code

```
WITH data(geom, view) AS (VALUES (
  'POLYGON((3.2 14.2,6.8 14.2,7.6 11.9,7.2 9.8,7.8 7.2,10.3 7.2,11.5 7.5,13.8 7.2,14.6 ↩
   5.9,14.2 4.8,14.8 3.2,14.6 0.9,13.2 -1.2,12.8 -3.8,12 -4.1,11 -4,9.8 -4.2,8.8 -4.8,8.3 ↩
   -2.5,6.5 0.2,5.8 2.2,5.3 0.2,3.5 -1.5,2.8 -3.8,1.6 -4.5,-0.3 -5.2,-2.2 -5.8,-0.9 ↩
   -4.8,-1.5 -4.4,-1.8 -2.8,-1.2 -1.8,0.5 -1.8,1.8 -1.8,2.3 -0.1,1.5 1.8,1.8 3.2,0 3.5,-3.2 ↩
    4.2,-5.2 4.2,-4.4 4.9,-4.9 5.5,-4.3 5.9,-3.8 6.4,-3 6.9,-3.2 7.8,-2.2 8.2,-1.7 8.9,-2.7 ↩
    9.4,-2.2 9.9,-1.4 10.9,-1.8 11.8,-1.2 12.2,-2.3 12.8,-2.9 13.2,0.6 13.5,3.2 14.2))':: ↩
   geometry,
  ST_MakeEnvelope(0.1, 0, 10, 10)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view, true) AS result,
       view AS view
FROM data;
```

Figure

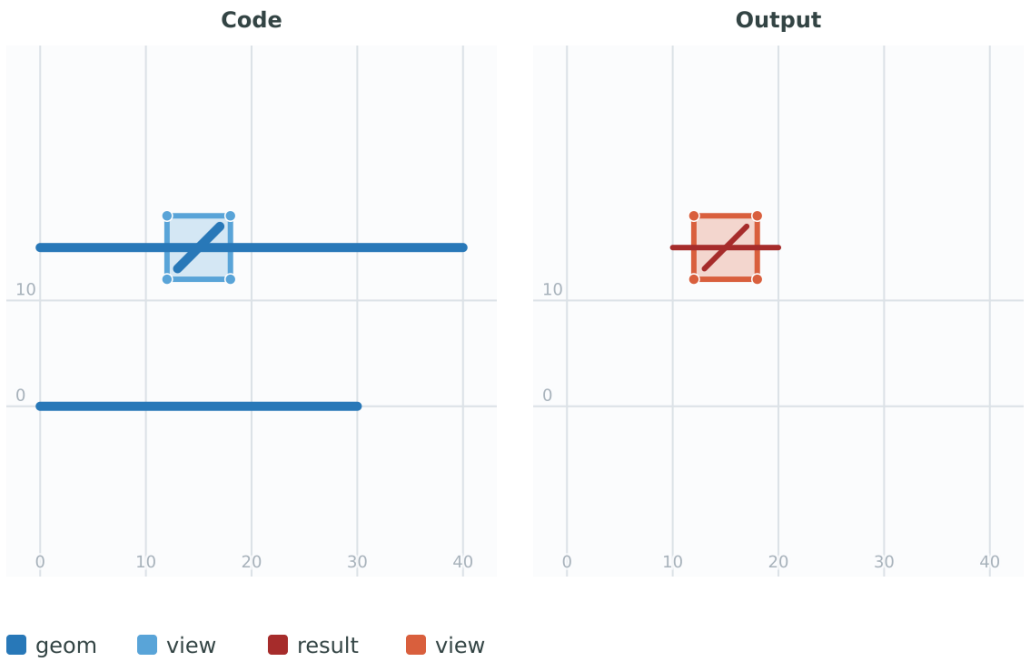


Only linework affecting the view is retained; existing vertices are reused.

Code

```
WITH data(geom, view) AS (VALUES (
  'MULTILINESTRING((0 0,10 0,20 0,30 0),(0 15,5 15,10 15,15 15,20 15,25 15,30 15,40 15),(13 ↩
     13,15 15,17 17))'::geometry,
  ST_MakeEnvelope(12, 12, 18, 18)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view, true) AS result,
       view AS view
FROM data;
```

Figure



A line with no segment relevant to the view becomes empty.

Code

```
SELECT ST_RemoveIrrelevantPointsForView('LINESTRING(0 0,10 0,20 0,30 0)',
ST_MakeEnvelope(12, 12, 18, 18), true);
```

Output

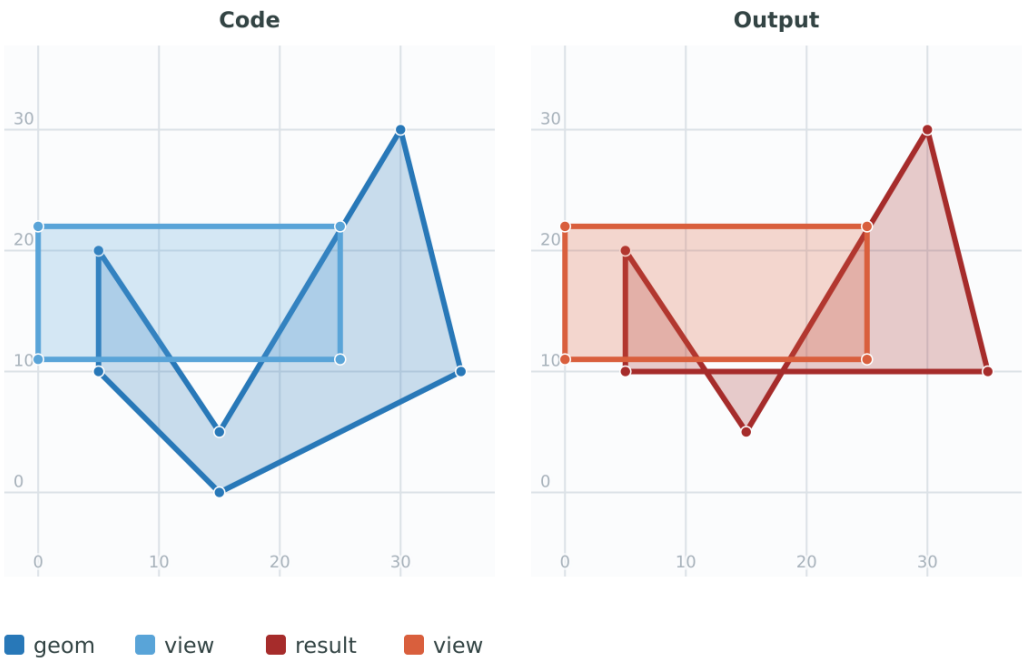
```
LINESTRING TOM
```

Removing vertices without computing intersections can introduce self-intersections in the result.

Code

```
WITH data(geom, view) AS (VALUES (
  'POLYGON((0 10,0 0,10 -10,30 0,25 20,10 -5,0 10))'::geometry,
  ST_MakeEnvelope(-5, 1, 20, 12)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view, true) AS result,
       view AS view
FROM data;
```

Figure

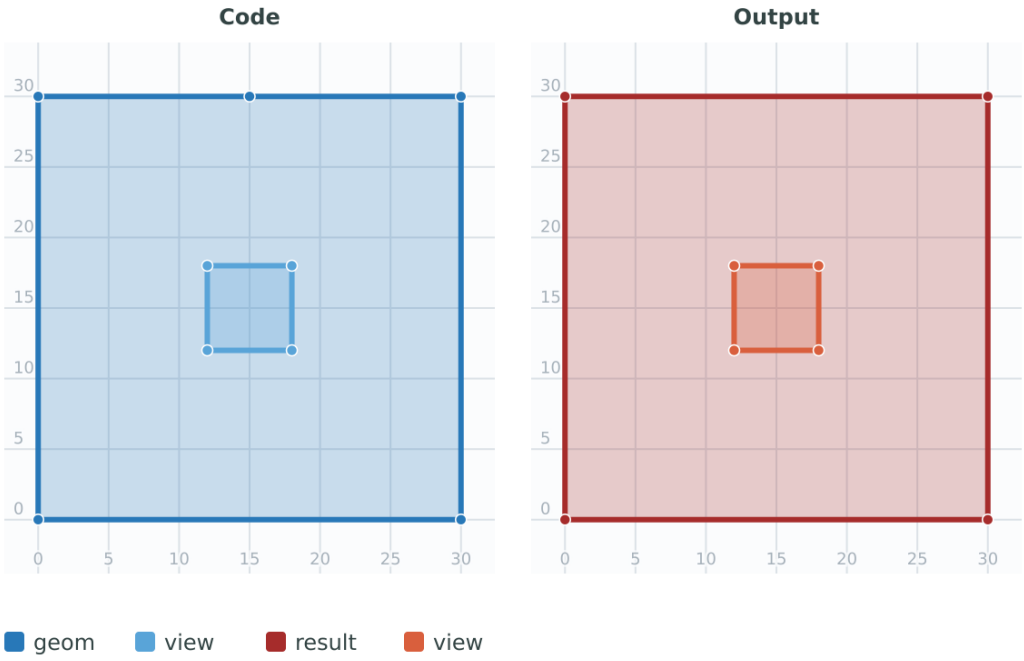


With the default *cartesian_hint*, the function removes only points known to be irrelevant to the view.

Code

```
WITH data(geom, view) AS (VALUES (
  'POLYGON((0 30,15 30,30 30,30 0,0 0,0 30))'::geometry,
  ST_MakeEnvelope(12, 12, 18, 18)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view) AS result,
       view AS view
FROM data;
```

Figure



Se även

ST_ClipByBox2D, ST_Intersection

7.5.27 ST_RemoveSmallParts

ST_RemoveSmallParts — Tar bort små delar (polygonringar eller linestrings) av en geometri.

Synopsis

geometry **ST_RemoveSmallParts**(geometry geom, double precision minSizeX, double precision minSizeY);

Beskrivning

Returnerar en geometry utan små delar (yttre eller inre polygonringar eller linestrings).

Denna funktion kan användas som ett förbehandlingssteg för att skapa förenklade kartor, t.ex. för att ta bort små öar eller hål.

Den utvärderar endast geometrier av typen (MULTI)POLYGON och (MULTI)LINESTRING. Andra geometrier förblir oförändrade.

Om *`minSizeX`* är större än 0 sorteras delar ut om deras bredd är mindre än *`minSizeX`*.

Om *`minSizeY`* är större än 0 sorteras delar ut om deras höjd är mindre än *`minSizeY`*.

Både *`minSizeX`* och *`minSizeY`* mäts i geometrins koordinatsystemenheter.

För polygontyper görs utvärderingen separat för varje ring, vilket kan leda till något av följande resultat:

- den ursprungliga geometrin,
- en POLYGON med alla ringar som har färre toppar,
- en POLYGON med ett reducerat antal inre ringar (som eventuellt har färre toppar),
- en POLYGON EMPTY, eller
- en MULTIPOLYGON med ett reducerat antal polygoner (som eventuellt har färre inre ringar eller hörn), eller
- en MULTIPOLYGON EMPTY.

För linestringstyper görs utvärderingen för varje linestrings, vilket kan leda till något av följande resultat:

- den ursprungliga geometrin,
- en LINESTRING med ett reducerat antal toppar,
- a LINESTRING EMPTY,
- en MULTILINESTRING med ett reducerat antal linestringsar (som eventuellt har färre toppar), eller
- en MULTILINESTRING EMPTY.

Tillgänglighet: 3.5.0

Exempel

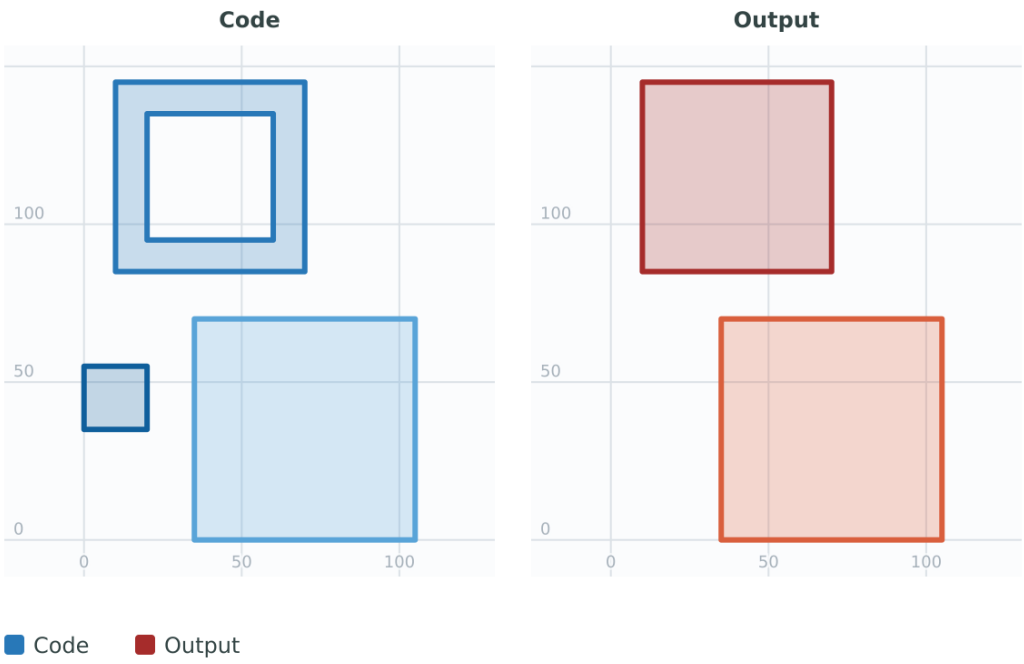
Code

```
SELECT ST_RemoveSmallParts('MULTIPOLYGON(
    ((60 160,120 160,120 220,60 220,60 160),(70 170,70 210,110 210,110 170,70 170)),
    ((85 75,155 75,155 145,85 145,85 75)),
    ((50 110,70 110,70 130,50 130,50 110)))',
    50, 50);
```

Output

```
MULTIPOLYGON(((60 160,120 160,120 220,60 220,60 160)),((85 75,155 75,155 145,85 145,85 75)))
```

Figure



Code

```
SELECT ST_RemoveSmallParts('LINESTRING(10 10,20 20)',
    50, 50);
```

Output

```
LINESTRING TOM
```

7.5.28 ST_Reverse

ST_Reverse — Returnera geometrin med omvänd ordning på topparna.

Synopsis

geometry **ST_Reverse**(geometry g1);

Beskrivning

Kan användas på vilken geometri som helst och vänder på ordningen på hörnen.

Förbättrad: 2.4.0 stöd för kurvor infördes.

Denna funktion stöder 3d och kommer inte att tappa z-index.

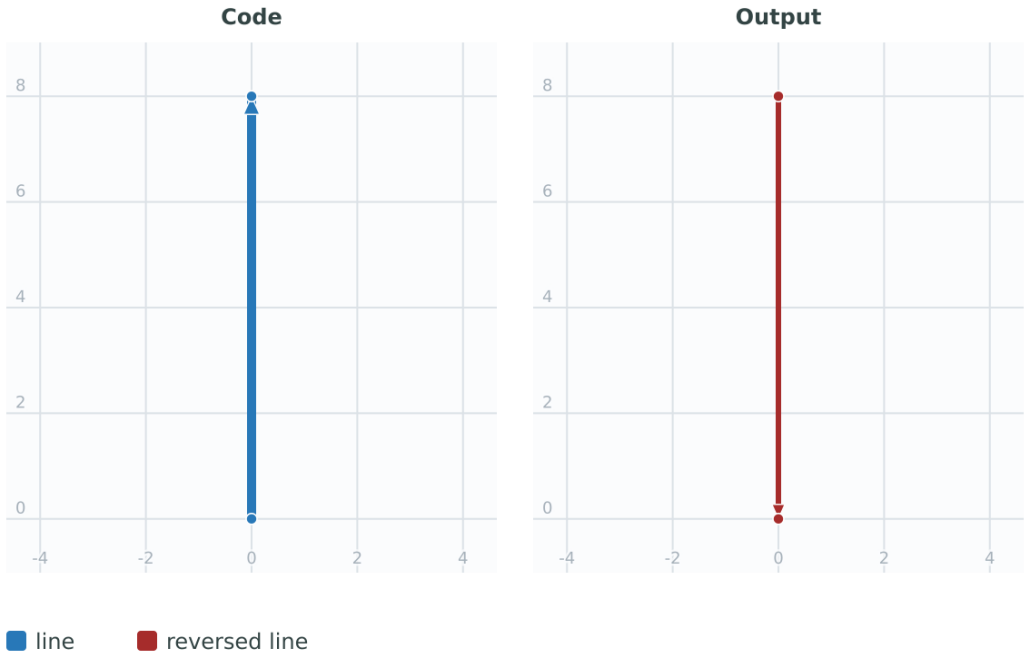
Denna funktion stöder polyedriska ytor.

Exempel

Code

```
WITH data(geom) AS (VALUES (
  ST_MakeLine(ST_Point(1, 2), ST_Point(1, 10))
))
SELECT geom AS input_line,
       ST_Reverse(geom) AS reversed_line
FROM data;
```

Figure



7.5.29 ST_Segmentize

ST_Segmentize — Returnerar en modifierad geometri/geografi som inte har något segment som är längre än ett givet avstånd.

Synopsis

geometry **ST_Segmentize**(geometry geom, float max_segment_length);
geography **ST_Segmentize**(geography geog, float max_segment_length);

Beskrivning

Returnerar en modifierad geometri/geografi som inte har något segment som är längre än `max_segment_leng`
Längden beräknas i 2D. Segmenten delas alltid upp i lika långa undersegment.

- För geometri är den maximala längden i enheterna i det spatiala referenssystemet.
- För geografi är den maximala längden i meter. Avstånden beräknas på sfären. Tillagda hörnpunkter skapas längs de sfäriska storcirkelbågar som definieras av segmentets ändpunkter.

Notera
Detta förkortar endast långa segment. Den förlänger inte segment som är kortare än den maximala längden.

Varning
För indata som innehåller långa segment kan en relativt kort `max_segment_length` leda till att ett mycket stort antal hörnpunkter läggs till. Detta kan hända oavsiktligt om argumentet av misstag anges som ett antal segment i stället för en maximal längd.

Tillgänglighet: 1.2.2

Förbättrad: 3.0.0 Segmentize-geometri producerar nu lika långa undersegment

Förbättrad: 2.3.0 Segmentize-geografin producerar nu lika långa undersegment

Förbättrad: 2.1.0 stöd för geografi infördes.

Ändrad: 2.1.0 Som ett resultat av införandet av geografistöd orsakar användningen `ST_Segmentize('LINEST 2, 3 4)', 0.5)` ett tvetydigt funktionsfel. Indata måste vara korrekt typad som en geometri eller geografi. Använd ST_GeomFromText, ST_GeogFromText eller en cast till den nödvändiga typen (t.ex. `ST_Segmentize('LINESTRING(1 2, 3 4)'::geometry, 0.5) )`

Exempel

Segmentisering av en linje. Långa segment delas upp jämnt, medan korta segment inte delas upp.

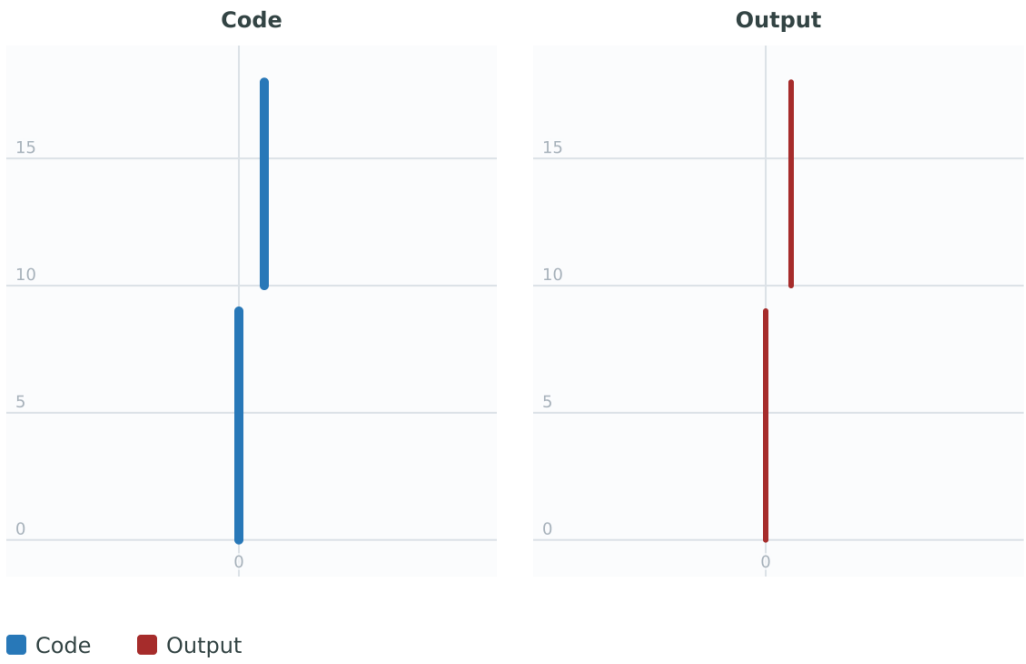
Code

```
SELECT ST_Segmentize('MULTILINESTRING((0 0,0 1,0 9),(1 10,1 18))'::geometry,
5);
```

Output

```
MULTILINESTRING((0 0,0 1,0 5,0 9),(1 10,1 14,1 18))
```

Figure



Segmentisering av en polygon:

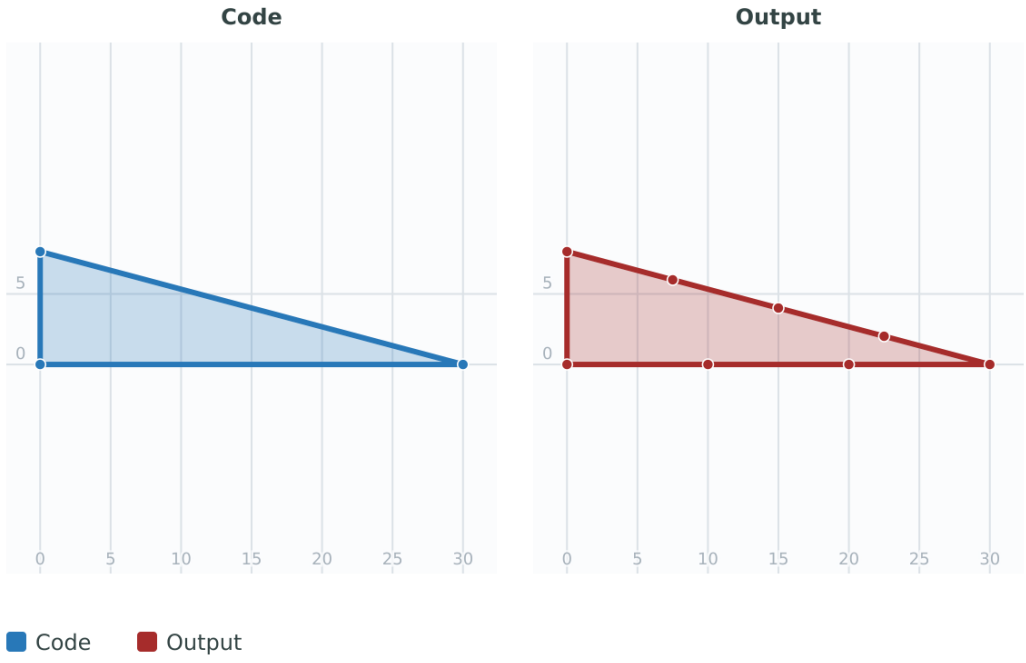
Code

```
SELECT ST_Segmentize('POLYGON((0 0,0 8,30 0,0 0))'::geometry,
        10);
```

Output

```
POLYGON((0 0,0 8,7.5 6,15 4,22.5 2,30 0,20 0,10 0,0 0))
```

Figure



Segmentisering av en geografisk linje med en maximal segmentlängd på 2000 kilometer. Hörnpunkter läggs till längs den storcirkelbåge som förbinder ändpunkterna.

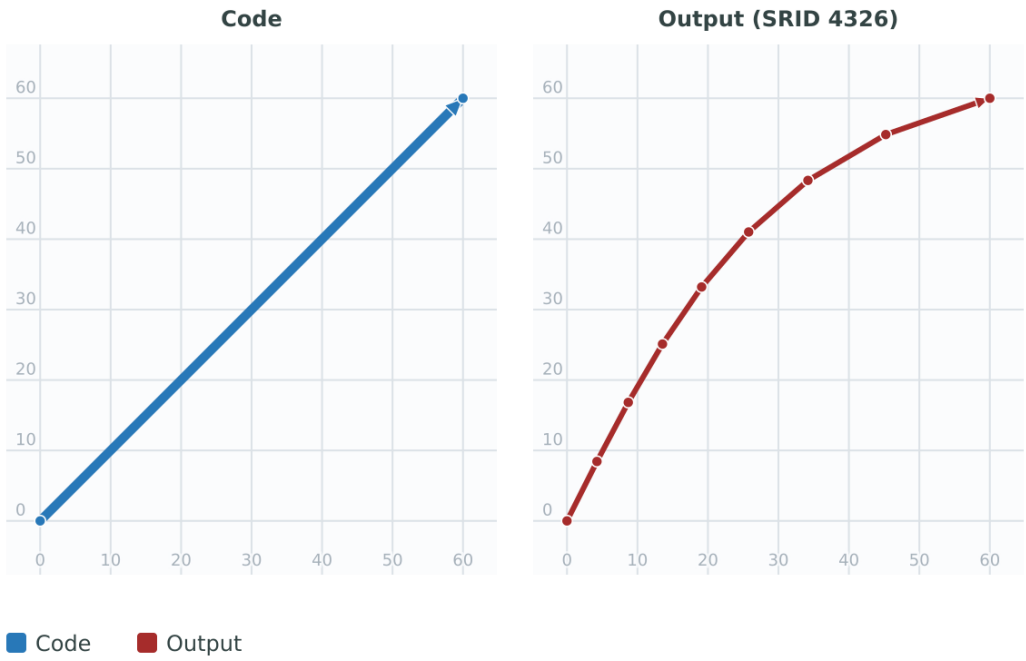
Code

```
SELECT ST_Segmentize(('LINESTRING (0 0,60 60)'::geography), 2000000);
```

Output

```
LINESTRING(0 0,4.252632294621186 8.43596525986862,8.69579947419404 16.824093489701564,13.550465473227048 ↩
    25.107950473646188,19.1066053508691 33.21091076089908,25.779290201459894 41.01711439406505,34.188839517966954 ↩
    48.337222885886,45.238153936612264 54.84733442373889,60 60)
```

Figure



Se även

ST_LineSubstring

7.5.30 ST_SetPoint

ST_SetPoint — Ersätt punkten i en linestrings med en given punkt.

Synopsis

geometry **ST_SetPoint**(geometry linestring, integer zerobasedposition, geometry point);

Beskrivning

Ersätter punkt N i linestrings med given punkt. Index är 0-baserat. Negativa index räknas baklänges, så att -1 är sista punkten. Detta är särskilt användbart i triggers när man försöker bibehålla relationen mellan lederna när en vertex flyttas.

Tillgänglighet: 1.1.0

Uppdaterad 2.3.0: negativ indexering

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Change first point in line string from -1 3 to -1 1.

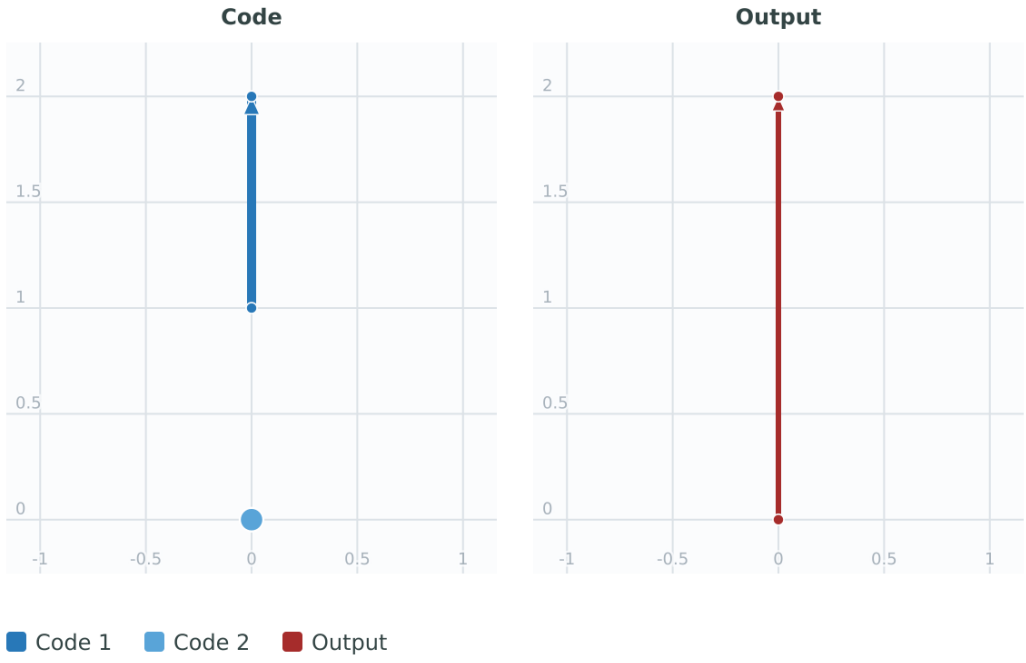
Code

```
SELECT ST_SetPoint('LINESTRING(-1 2,-1 3)', 0, 'POINT(-1 1)');
```

Output

```
LINESTRING(-1 1,-1 3)
```

Figure



Change last point in a line string (let's play with 3D linestring this time)

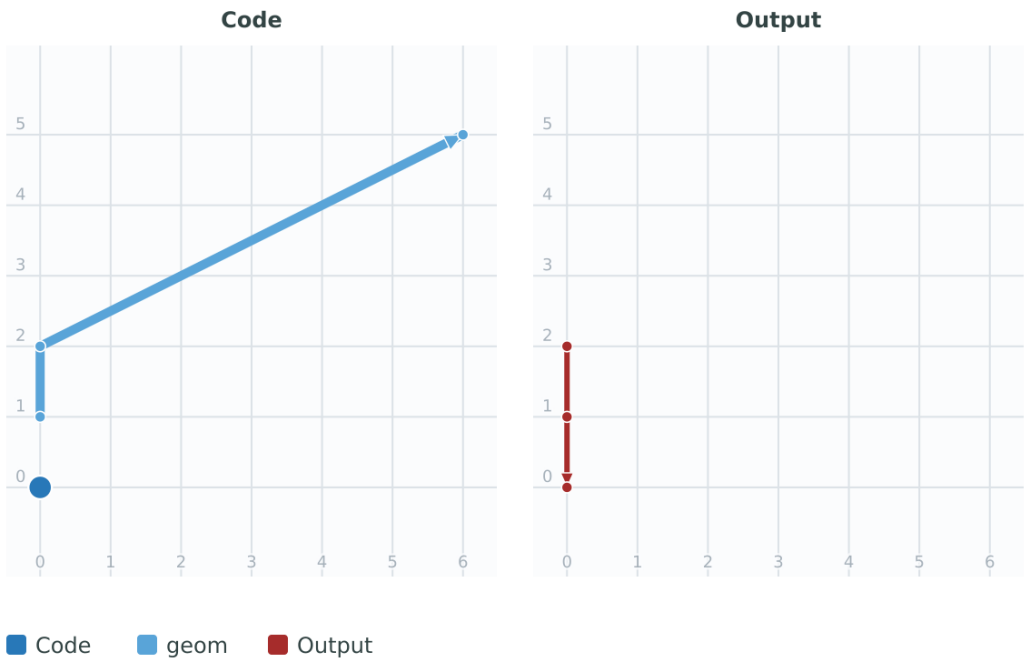
Code

```
SELECT ST_SetPoint(foo.geom, ST_NumPoints(foo.geom) - 1, 'POINT(-1 1 3)')
FROM (SELECT 'LINESTRING(-1 2 3,-1 3 4,5 6 7)'::geometry As geom) As foo;
```

Output

```
LINESTRING(-1 2 3,-1 3 4,-1 1 3)
```

Figure



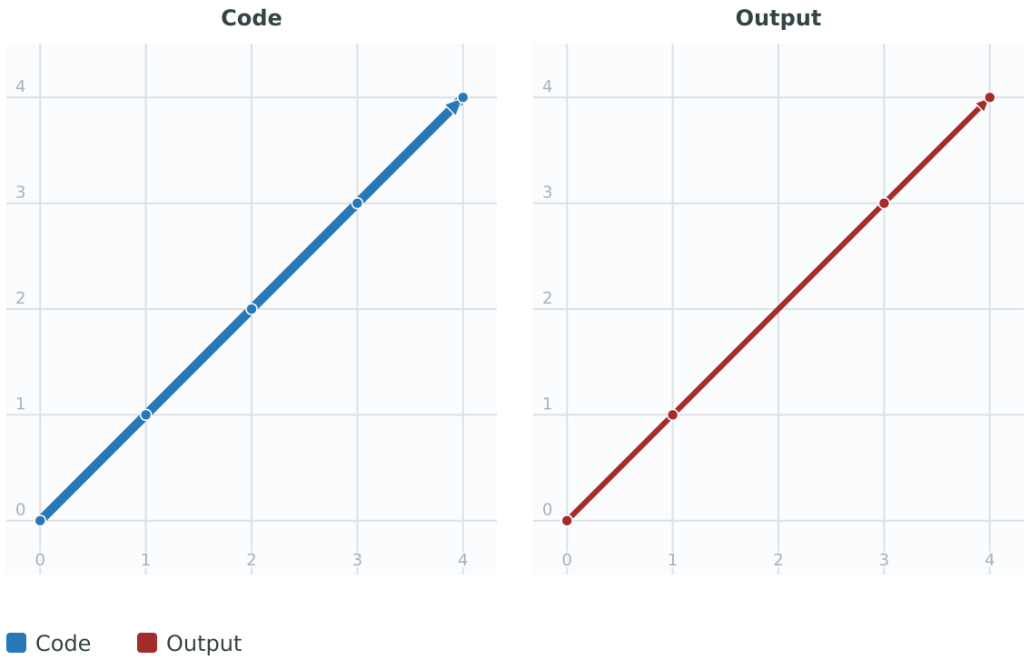
Code

```
SELECT ST_SetPoint(g, -3, p)
FROM ST_GeomFromText('LINESTRING(0 0,1 1,2 2,3 3,4 4)') AS g
    CROSS JOIN ST_PointN(g, 1) as p;
```

Output

```
LINESTRING(0 0,1 1,0 0,3 3,4 4)
```

Figure



Se även

ST_AddPoint, ST_NPoints, ST_NumPoints, ST_PointN, ST_RemovePoint

7.5.31 ST_ShiftLongitude

ST_ShiftLongitude — Flyttar longitudkoordinaterna för en geometri mellan -180..180 och 0..360.

Synopsis

geometry **ST_ShiftLongitude**(geometry geom);

Beskrivning

Läser varje punkt/vertex i en geometri och flyttar dess longitudkoordinat från -180..0 till 180..360 och vice versa om den ligger mellan dessa intervall. Denna funktion är symmetrisk så resultatet är en 0..360-representation av en -180..180-data och en -180..180-representation av en 0..360-data.

Notera
Detta är endast användbart för data med koordinater i longitud/latitud, t.ex. SRID 4326 (WGS 84 geographic)

Varning
Före 1.3.4 hindrade buggen detta från att fungera för MULTIPOINT. 1.3.4+ fungerar även med MULTIPOINT.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes.

OBS: denna funktion döptes om från ST_Shift_Longitude”i 2.2.0

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

This example performs a forward longitude shift on a single point.

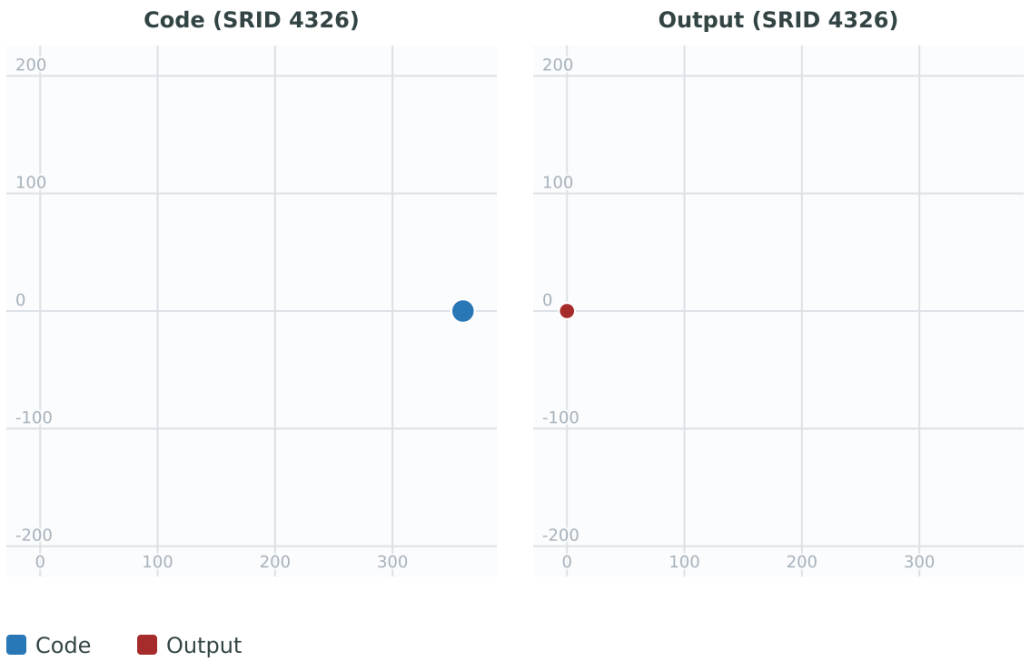
Code

```
SELECT ST_ShiftLongitude('SRID=4326;POINT(270 0)'::geometry)
```

Output

```
POINT(-90 0)
```

Figure



This example performs a reverse longitude shift on a single point.

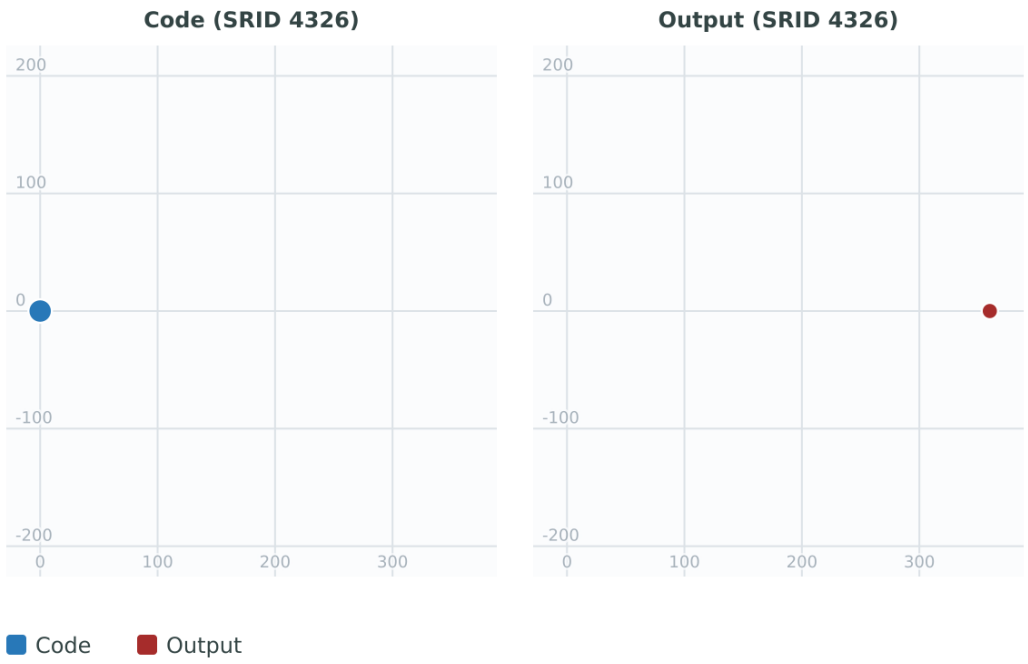
Code

```
SELECT ST_ShiftLongitude('SRID=4326;POINT(-90 0)'::geometry)
```

Output

```
POINT(270 0)
```

Figure



This example shows that LineStrings are shifted only where coordinates require it.

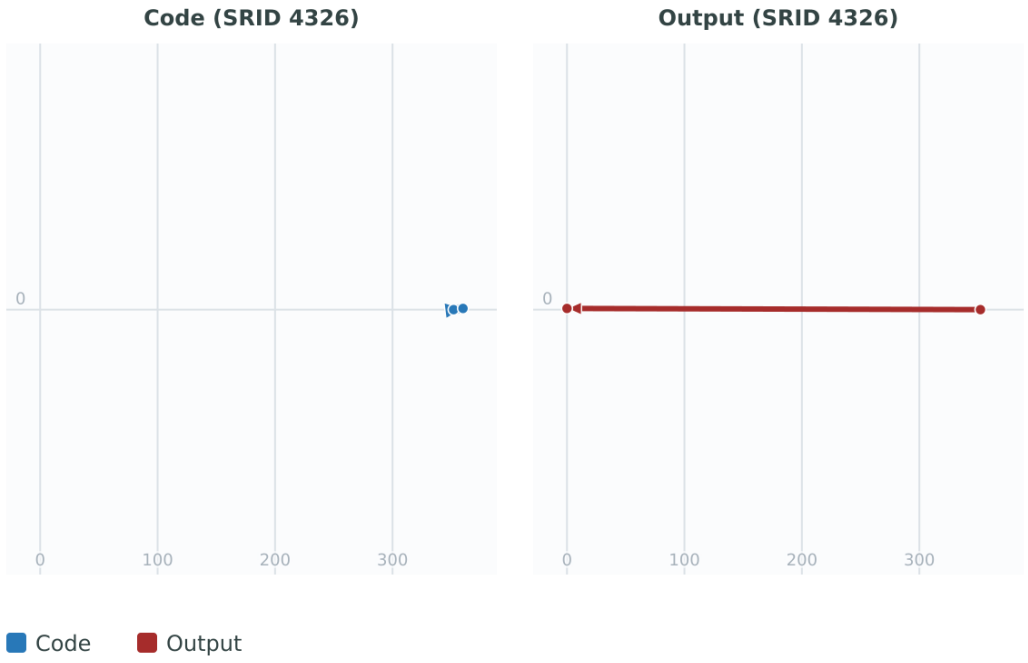
Code

```
SELECT ST_ShiftLongitude('SRID=4326;LINESTRING(174 12,182 13)'::geometry)
```

Output

```
LINESTRING(174 12,-178 13)
```

Figure



Se även

ST_WrapX

7.5.32 ST_WrapX

ST_WrapX — Omsluta en geometri runt ett X-värde.

Synopsis

geometry **ST_WrapX**(geometry geom, float8 wrap, float8 move);

Beskrivning

Denna funktion delar upp inmatningsgeometrierna och flyttar sedan varje resulterande komponent som faller till höger (för negativ move”) eller till vänster (för positiv move”) om den givna årap-linjen i den riktning som anges av move-parametern, och sammanfogar slutligen bitarna igen.

Notera
Detta är användbart för att ömcentreraLong-Lat Input så att intressanta funktioner inte skapas från den ena sidan till den andra.

Tillgänglighet: 2.3.0 kräver GEOS

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Move all components whose bounding boxes fall completely to the left of `x = 0` by +360.

Code

```
SELECT ST_WrapX(geom, 0, 360);
```

Move all components whose bounding boxes fall completely to the left of `x = -30` by +360.

Code

```
SELECT ST_WrapX(geom, -30, 360);
```

Se även

ST_ShiftLongitude

7.5.33 ST_SnapToGrid

ST_SnapToGrid — Fäst alla punkter i indatageometrin i ett regelbundet rutnät.

Synopsis

geometry **ST_SnapToGrid**(geometry geomA, float originX, float originY, float sizeX, float sizeY);
geometry **ST_SnapToGrid**(geometry geomA, float sizeX, float sizeY);
geometry **ST_SnapToGrid**(geometry geomA, float size);
geometry **ST_SnapToGrid**(geometry geomA, geometry pointOrigin, float sizeX, float sizeY, float sizeZ, float sizeM);

Beskrivning

Variant 1,2,3: Fäst alla punkter i indatageometrin i det rutnät som definieras av dess ursprung och cellstorlek. Ta bort på varandra följande punkter som faller på samma cell, och returnera slutligen NULL om utdatapunkterna inte är tillräckliga för att definiera en geometri av den givna typen. Kollapsade geometrier i en samling tas bort från den. Användbart för att minska precisionen.

Variant 4: Introducerad 1.1.0 - Snappa alla punkter i indatageometrin till det rutnät som definieras av dess ursprung (det andra argumentet, måste vara en punkt) och cellstorlekar. Ange 0 som storlek för alla dimensioner som du inte vill fästa i ett rutnät.

Notera
Den återlämnade geometrin kan förlora sin enkelhet (se ST_IsSimple).

Notera
Före release 1.1.0 returnerade denna funktion alltid en 2D-geometri. Från och med 1.1.0 kommer den returnerade geometrin att ha samma dimensionalitet som den inmatade geometrin med högre dimensionsvärden orörda. Använd versionen som tar ett andra geometriargument för att definiera alla rutnätsdimensioner.

Tillgänglighet: 1.0.0RC1

Tillgänglighet: 1.1.0 - Stöd för Z och M

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Snap a geometry column to a precision grid of 10^-3.

Code

```
UPDATE mytable
   SET geom = ST_SnapToGrid(geom, 0.001);
```

Display how the same grid size rounds a LineString.

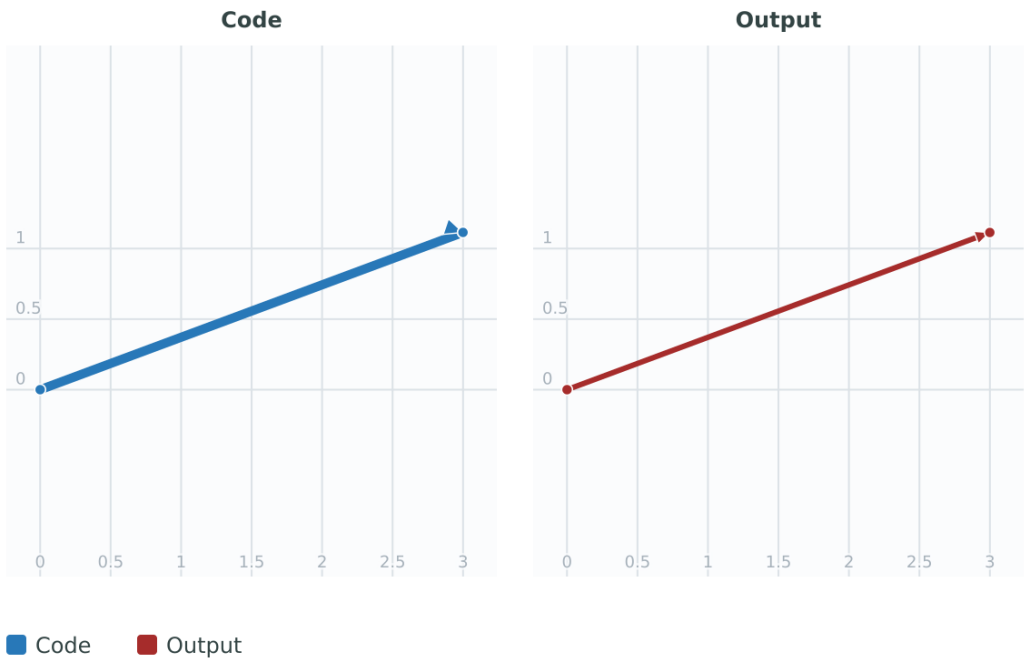
Code

```
SELECT ST_SnapToGrid('LINESTRING(1.1115678 2.123,4.111111 3.2374897,4.11112 3.23748667)':: ↩
    geometry,
        0.001);
```

Output

```
LINESTRING(1.112 2.123,4.111 3.237)
```

Figure



This example snaps a 4D geometry.

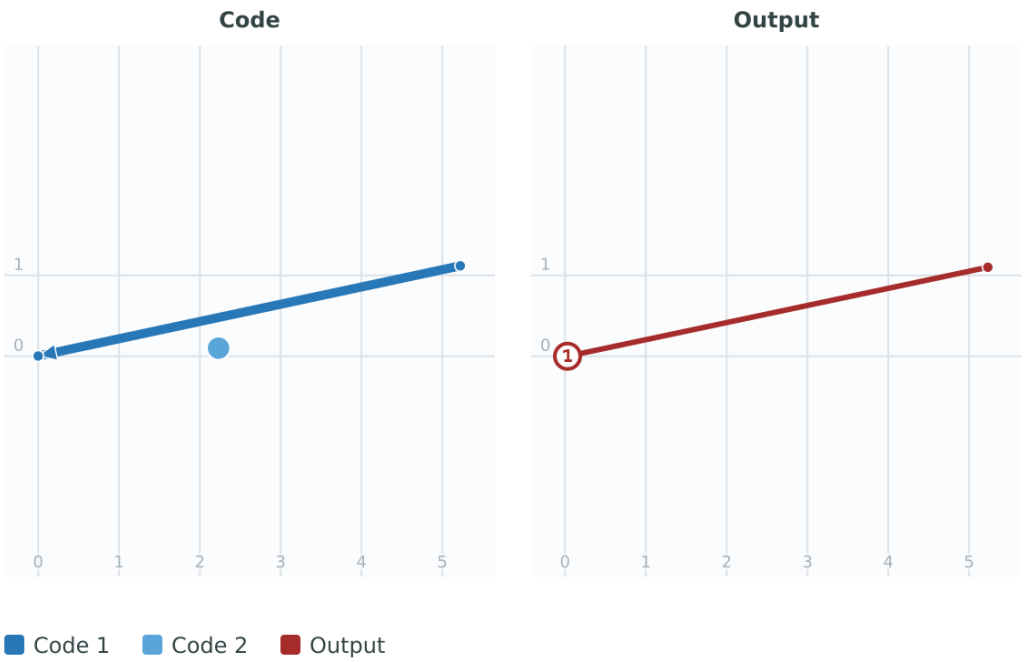
Code

```
SELECT ST_SnapToGrid('LINESTRING(-1.1115678 2.123 2.3456 1.11111,
        4.111111 3.2374897 3.1234 1.1111,-1.11111112 2.123 2.3456 1.1111112)'::geometry,
        'POINT(1.12 2.22 3.2 4.4444)',
        0.1, 0.1, 0.1, 0.01);
```

Output

```
LINESTRING(-1.08 2.12 2.3 1.1144,4.12 3.22 3.1 1.1144,-1.08 2.12 2.3 1.1144)
```

Figure



With a 4D geometry - the ST_SnapToGrid(geom,size) only touches x and y coords but keeps m and z the same.

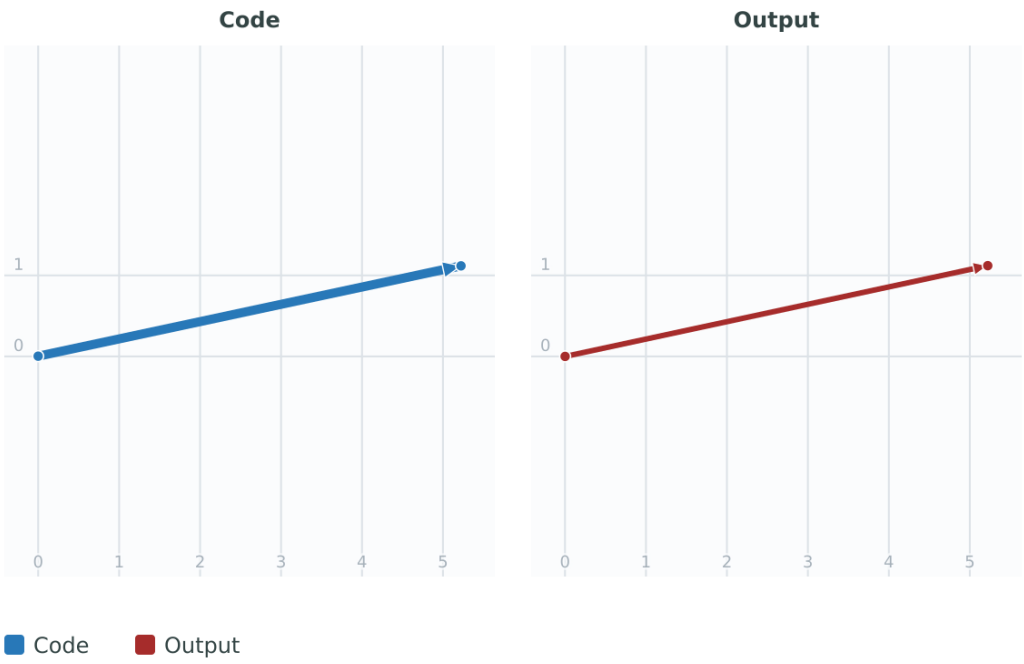
Code

```
SELECT ST_SnapToGrid('LINESTRING(-1.1115678 2.123 3 2.3456,
        4.111111 3.2374897 3.1234 1.1111)'::geometry,
       0.01);
```

Output

```
LINESTRING(-1.11 2.12 3 2.3456,4.11 3.24 3.1234 1.1111)
```

Figure



Se även

ST_Snap, ST_AsEWKT, ST_AsText, ST_GeomFromText, ST_GeomFromEWKT, ST_Simplify

7.5.34 ST_Snap

ST_Snap — Fäst segment och vertikaler i indatageometrin till vertikaler i en referensgeometri.

Synopsis

geometry **ST_Snap**(geometry input, geometry reference, float tolerance);

Beskrivning

Snappar en geometris hörn och segment till en annan geometris hörn. En tolerans för snäppavstånd används för att styra var snäppningen utförs. Resultatgeometrin är indatageometrin med de snäppta topparna. Om ingen snäppning sker returneras indatageometrin oförändrad.

Att snäppa en geometri till en annan kan förbättra robustheten för överlagringsoperationer genom att eliminera nästan sammanfallande kanter (som orsakar problem under nodning och intersektionsberäkning).

För mycket snäppning kan leda till att en ogiltig topologi skapas, så antalet och placeringen av de snäppta topparna bestäms med hjälp av heuristik för att avgöra när det är säkert att snäppa. Detta kan dock leda till att vissa potentiella snäppningar utelämnas.

Notera
Den återlämnade geometrin kan förlora sin enkelhet (se ST_IsSimple) och giltighet (se ST_IsValid).

Utförs av GEOS-modulen.

Tillgänglighet: 2.0.0

Exempel

A MultiPolygon snapped to a LineString at 1.01 times their distance.

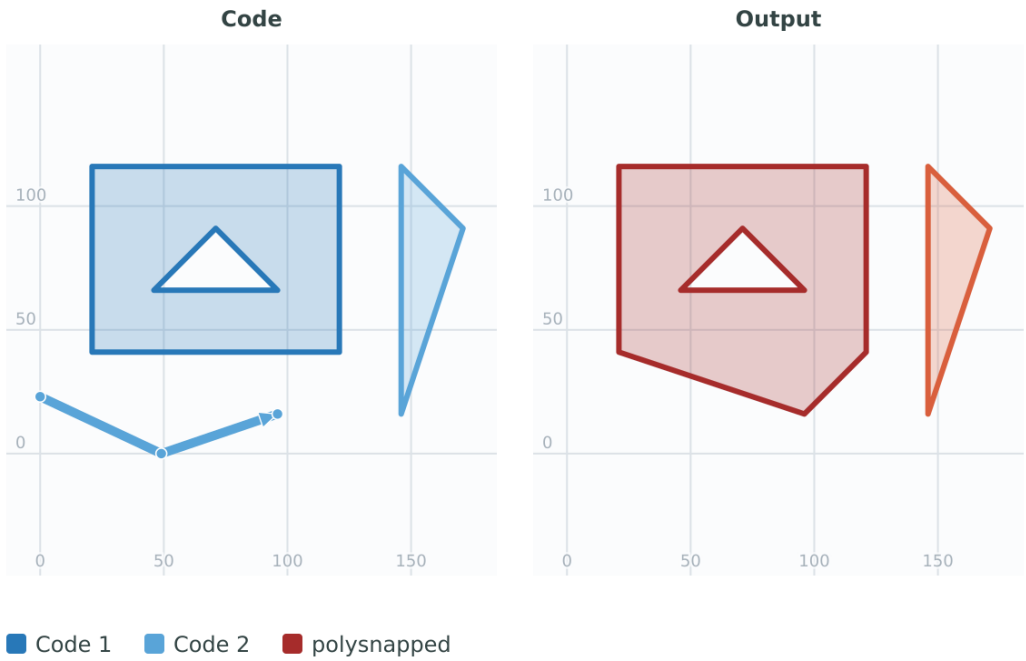
Code

```
SELECT ST_Snap(poly, line, ST_Distance(poly, line)*1.01) AS polysnapped
FROM (SELECT
   ST_GeomFromText('MULTIPOLYGON(
     ((26 125,26 200,126 200,126 125,26 125 ),
      ( 51 150,101 150,76 175,51 150 )),
      (( 151 100,151 200,176 175,151 100 )))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
MULTIPOLYGON(((26 125,26 200,126 200,126 125,101 100,26 125),
(51 150,101 150,76 175,51 150)),((151 100,151 200,176 175,151 100)))
```

Figure



A MultiPolygon snapped to a LineString at 1.25 times their distance.

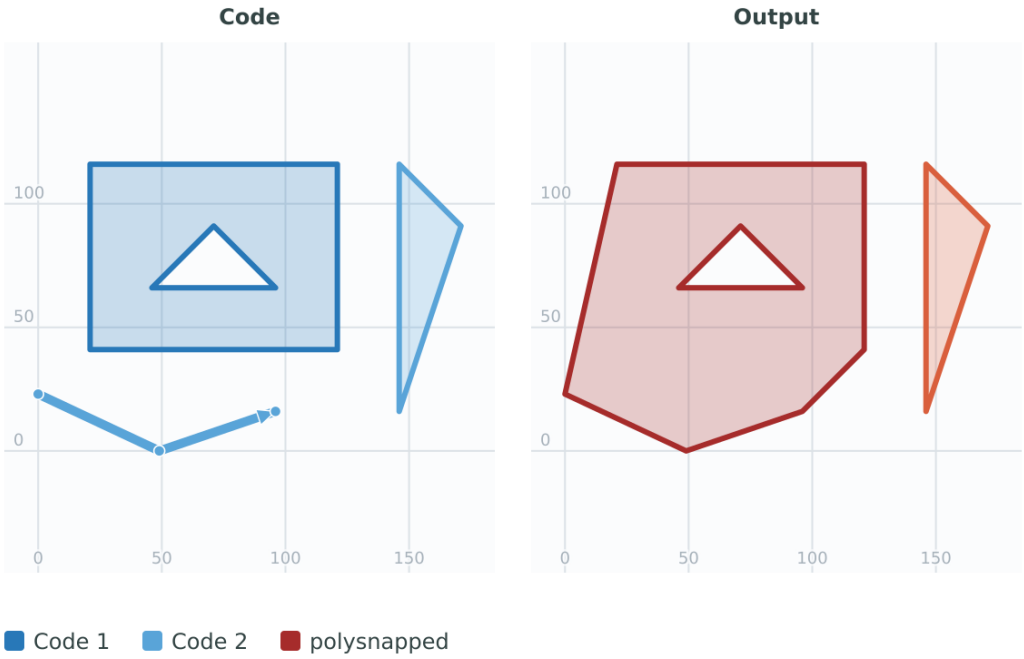
Code

```
SELECT ST_Snap(poly, line, ST_Distance(poly, line)*1.25) AS polysnapped
FROM (SELECT
  ST_GeomFromText('MULTIPOLYGON(
    (( 26 125,26 200,126 200,126 125,26 125 ),
      ( 51 150,101 150,76 175,51 150 )),
      (( 151 100,151 200,176 175,151 100 )))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
MULTIPOLYGON(((5 107,26 200,126 200,126 125,101 100,54 84,5 107),
(51 150,101 150,76 175,51 150)),((151 100,151 200,176 175,151 100)))
```

Figure



The LineString snapped to the original MultiPolygon at 1.01 times their distance.

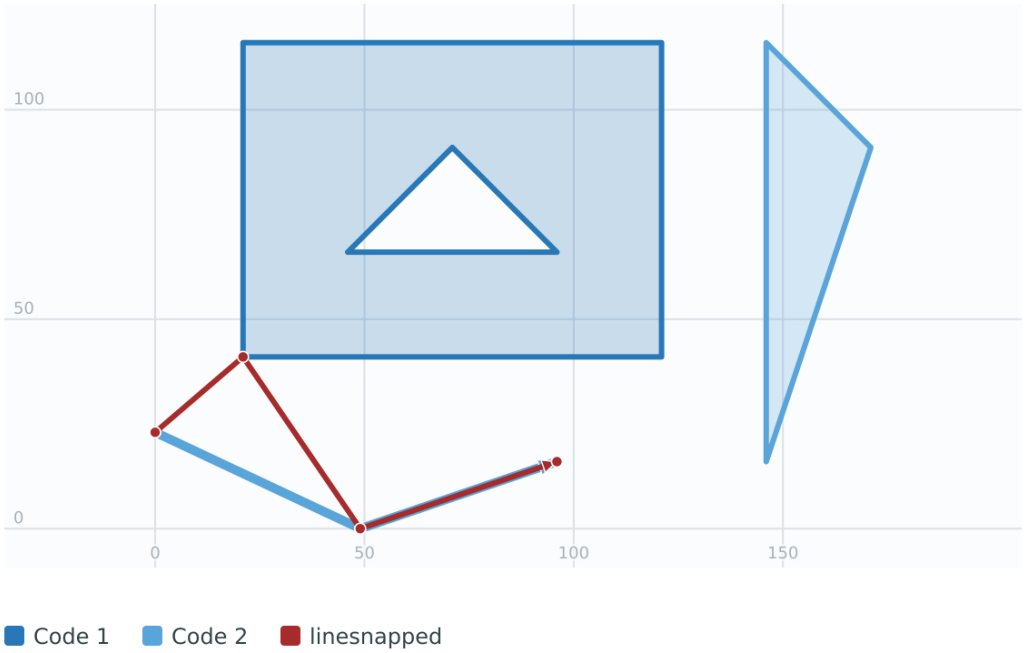
Code

```
SELECT ST_Snap(line, poly, ST_Distance(poly, line)*1.01) AS linesnapped
FROM (SELECT
  ST_GeomFromText('MULTIPOLYGON(
     ((26 125,26 200,126 200,126 125,26 125),
      (51 150,101 150,76 175,51 150 )),
      ((151 100,151 200,176 175,151 100)))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
LINESTRING(5 107,26 125,54 84,101 100)
```

Figure



The LineString snapped to the original MultiPolygon at 1.25 times their distance.

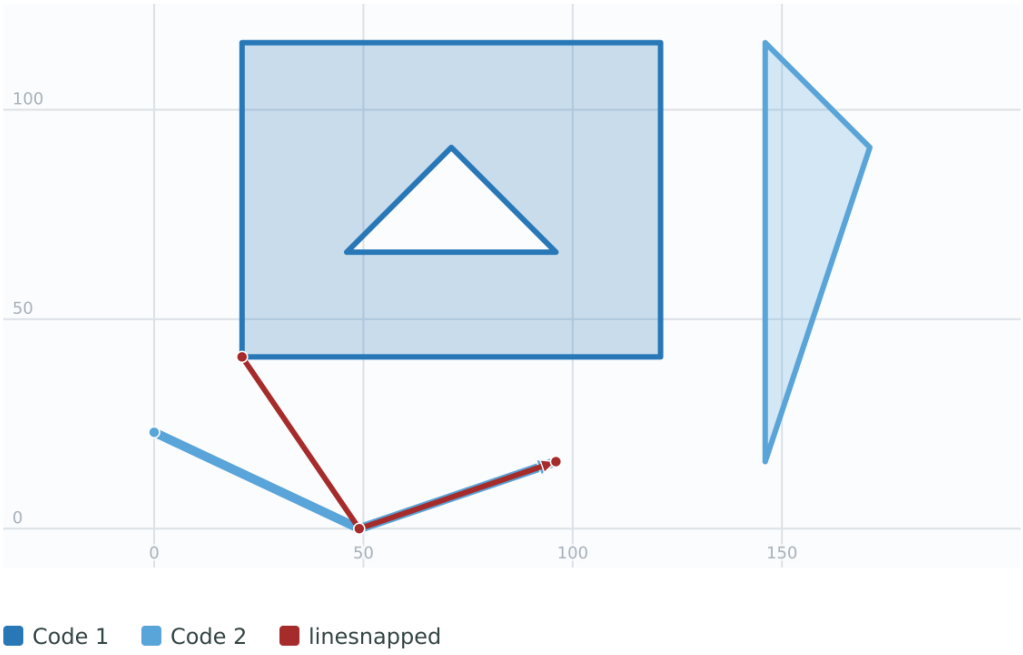
Code

```
SELECT ST_Snap(line, poly, ST_Distance(poly, line)*1.25) AS linesnapped
FROM (SELECT
  ST_GeomFromText('MULTIPOLYGON(
     (( 26 125,26 200,126 200,126 125,26 125 ),
      (51 150,101 150,76 175,51 150 )),
      ((151 100,151 200,176 175,151 100 )))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
LINESTRING(26 125,54 84,101 100)
```

Figure



Se även

ST_SnapToGrid

7.5.35 ST_SwapOrdinates

ST_SwapOrdinates — Returnerar en version av den givna geometrin med givna ordinatvärden ombytta.

Synopsis

geometry **ST_SwapOrdinates**(geometry geom, cstring ords);

Beskrivning

Returnerar en version av den givna geometrin med givna ordinater ombytta.

Parametern `ords` är en sträng med 2 tecken som namnger de ordinater som ska bytas ut. Giltiga namn är: x,y,z och m.

Tillgänglighet: 2.2.0

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

This example scales the M value by 2.

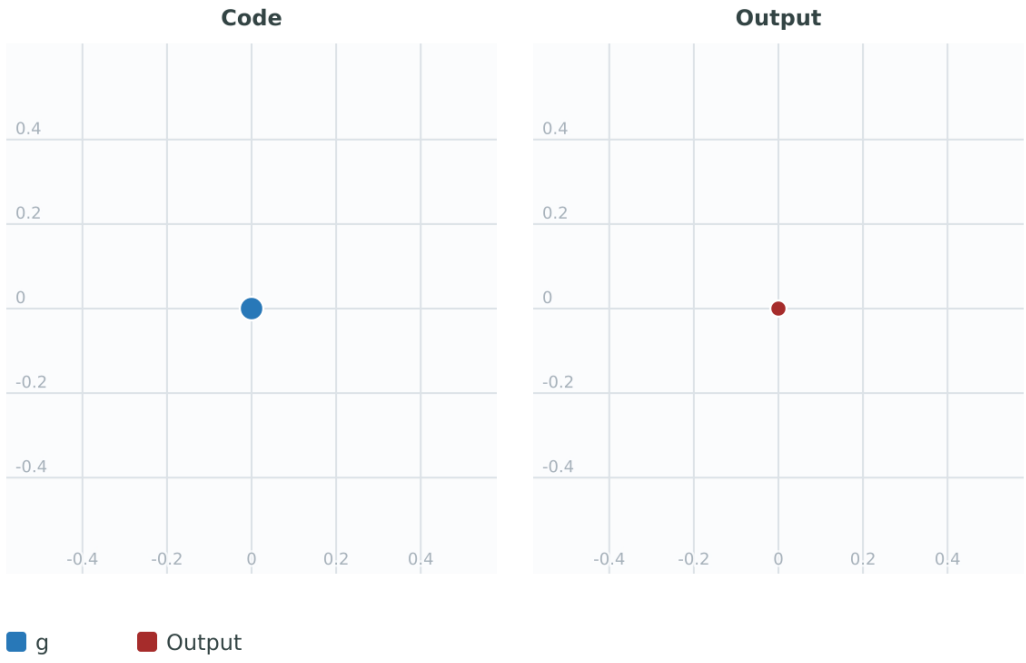
Code

```
SELECT ST_SwapOrdinates(ST_Scale(ST_SwapOrdinates(g, 'xm'),
      2, 1
    ),
  'xm') FROM ( SELECT 'POINT ZM (0 0 0 2)'::geometry g ) foo;
```

Output

```
POINT ZM (0 0 0 4)
```

Figure



Se även

ST_FlipCoordinates

7.6 Validering av geometri

7.6.1 ST_IsValid

ST_IsValid — Testar om en geometri är välformad i 2D.

Synopsis

boolean **ST_IsValid**(geometry g);
boolean **ST_IsValid**(geometry g, integer flags);

Beskrivning

Testar om ett ST_Geometry-värde är välformat och giltigt i 2D enligt OGC:s regler. För geometrier med 3 och 4 dimensioner testas giltigheten fortfarande bara i 2 dimensioner. För geometrier som är ogiltiga skickas en PostgreSQL NOTICE ut med information om varför den inte är giltig.

För versionen med parametern `flags` dokumenteras de värden som stöds i ST_IsValidDetail. Den här versionen skriver inte ut något meddelande som förklarar ogiltigheten.

För mer information om definitionen av geometrisk validitet, se Avsnitt 4.4

Notera
SQL-MM definierar att resultatet av ST_IsValid(NULL) ska vara 0, medan PostGIS returnerar NULL.

Utförs av GEOS-modulen.

Den version som accepterar flaggor är tillgänglig från och med 2.0.0.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.9

Notera
Varken OGC-SFS eller SQL-MM-specifikationerna innehåller ett flaggargument för ST_IsValid. Flaggan är en PostGIS-utökning.

Exempel

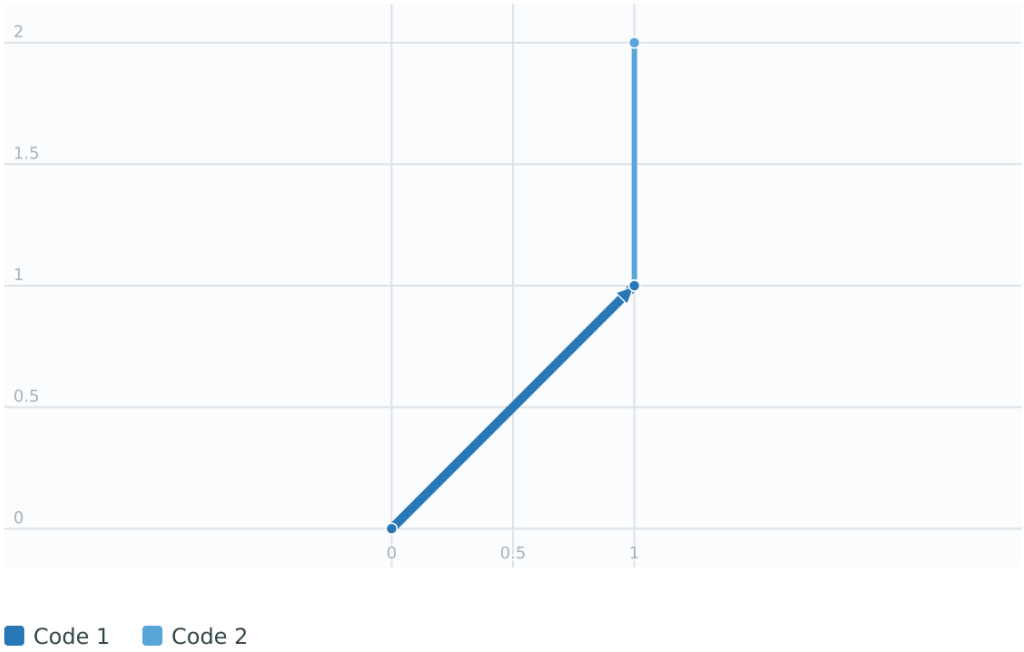
Code

```
SELECT ST_IsValid('LINESTRING(0 0,1 1)') As good_line,
ST_IsValid('POLYGON((0 0,1 1,1 2,1 1,0 0))') As bad_poly;
```

Output

```
NOTICE:  Self-intersection at or near point 0 0
 good_line | bad_poly
-----------+----------
 t         | f
```

Figure



Find tables with invalid geometries in registered geometry columns.

Code

```
CREATE TEMP TABLE invalid_geometry_columns (
    table_name text,
    column_name text,
    invalid_count bigint
);

DO $$
DECLARE
    rec record;
    invalid_count bigint;
BEGIN
    FOR rec IN
        SELECT f_table_schema, f_table_name, f_geometry_column
        FROM geometry_columns
        ORDER BY f_table_schema, f_table_name, f_geometry_column
    LOOP
        EXECUTE format(
            'SELECT count(*) FROM %I.%I WHERE NOT ST_IsValid(%I, 0)',
            rec.f_table_schema,
            rec.f_table_name,
            rec.f_geometry_column
        ) INTO invalid_count;

        IF invalid_count
> 0 THEN
            INSERT INTO invalid_geometry_columns
            VALUES (
                format('%I.%I', rec.f_table_schema, rec.f_table_name),
                rec.f_geometry_column,
                invalid_count
            );
        END IF;
    END LOOP;
```

```
END;
$$;

SELECT * FROM invalid_geometry_columns;
```

Se även

ST_IsSimple, ST_IsValidReason, ST_IsValidDetail,

7.6.2 ST_IsValidDetail

ST_IsValidDetail — Returnerar en `valid_detail-rad` som anger om en geometri är giltig eller om den inte är det, en orsak och en plats.

Synopsis

valid_detail **ST_IsValidDetail**(geometry geom, integer flags);

Beskrivning

Returnerar en `valid_detail-rad` som innehåller en boolean (`valid`) som anger om en geometri är giltig, en varchar (`reason`) som anger varför den är ogiltig och en geometri(`location`) som pekar ut var den är ogiltig.

Användbar för att förbättra kombinationen av ST_IsValid och ST_IsValidReason för att generera en detaljerad rapport om ogiltiga geometrier.

Den valfria parametern `flags` är ett bitfält. Den kan ha följande värden:

- 0: Använd vanlig OGC SFS-validitetssemantik.
- 1: Betrakta vissa typer av självberörande ringar (inverterade skal och exverterade hål) som giltiga. Detta är också känt som ”ESRI-flaggan”, eftersom det är den giltighetsmodell som används av dessa verktyg. Observera att detta är ogiltigt enligt OGC-modellen.

Utförs av GEOS-modulen.

Tillgänglighet: 2.0.0

Exempel

This example returns the first three rejects from a successful quintuplet experiment.

Code

```
SELECT gid, reason(ST_IsValidDetail(geom)), location(ST_IsValidDetail(geom)) as location
FROM
(SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), array_agg(f.line)) As geom, gid
FROM (SELECT ST_Buffer(ST_Point(x1*10,y1), z1) As buff, x1*10 + y1*100 + z1*1000 As gid
    FROM generate_series(-4,6) x1
    CROSS JOIN generate_series(2,5) y1
    CROSS JOIN generate_series(1,8) z1
    WHERE x1
> y1*0.5 AND z1 < x1*y1) As e
```

```
    INNER JOIN (SELECT ST_Translate(ST_ExteriorRing(ST_Buffer(ST_Point(x1*10,y1), z1)),y1 ↩
    *1, z1*2) As line
    FROM generate_series(-3,6) x1
    CROSS JOIN generate_series(2,5) y1
    CROSS JOIN generate_series(1,10) z1
    WHERE x1
> y1*0.75 AND z1 < x1*y1) As f
ON (ST_Area(e.buff)
> 78 AND ST_Contains(e.buff, f.line))
GROUP BY gid, e.buff) As quintuplet_experiment
WHERE ST_IsValid(geom) = false
ORDER BY gid
LIMIT 3;
```

Output

```
 gid  |      reason       |  location
------+-------------------+-------------
 5330 | Self-intersection | POINT(32 5)
 5340 | Self-intersection | POINT(42 5)
 5350 | Self-intersection | POINT(52 5)
```

This example reports a valid LineString.

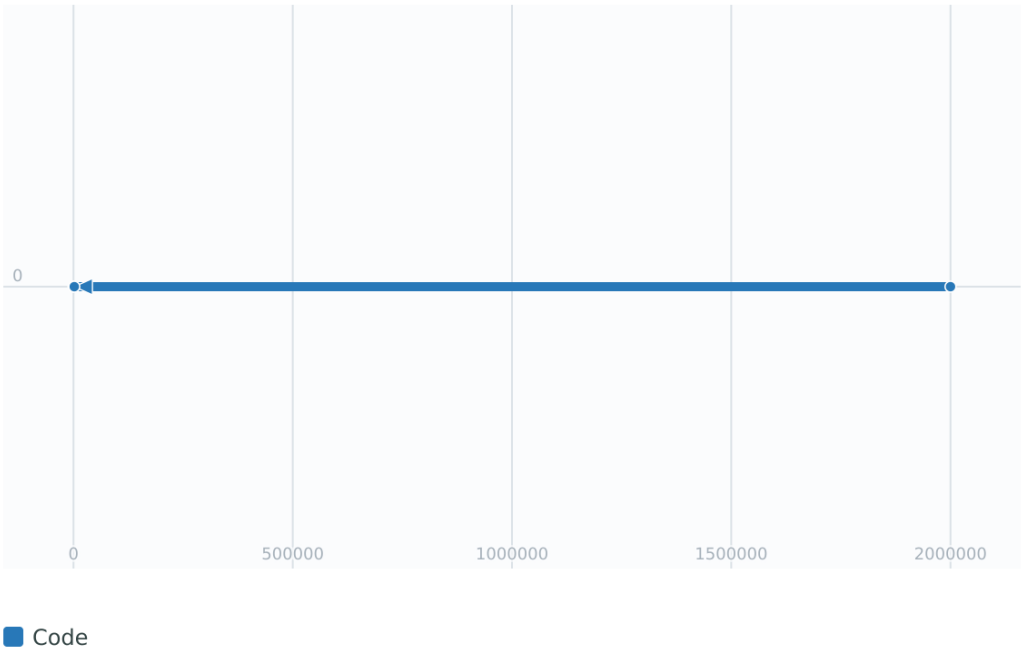
Code

```
SELECT * FROM ST_IsValidDetail('LINESTRING(220227 150406,2220227 150407,222020 150410)');
```

Output

```
 valid | reason | location
-------+--------+----------
 t     |        |
```

Figure



Se även

ST_IsValid, ST_IsValidReason

7.6.3 ST_IsValidReason

ST_IsValidReason — Returnerar text som anger om en geometri är giltig, eller en orsak till ogiltigheten.

Synopsis

text **ST_IsValidReason**(geometry geomA);
text **ST_IsValidReason**(geometry geomA, integer flags);

Beskrivning

Returnerar text som anger om en geometri är giltig, eller om den är ogiltig en orsak till detta.

Används i kombination med ST_IsValid för att generera en detaljerad rapport om ogiltiga geometrier och orsaker.

Tillåtna `flaggor` finns dokumenterade i ST_IsValidDetail.

Utförs av GEOS-modulen.

Tillgänglighet: 1.4

Tillgänglighet: 2.0-versionen tar flaggor.

Exempel

This example uses an invalid bow-tie polygon.

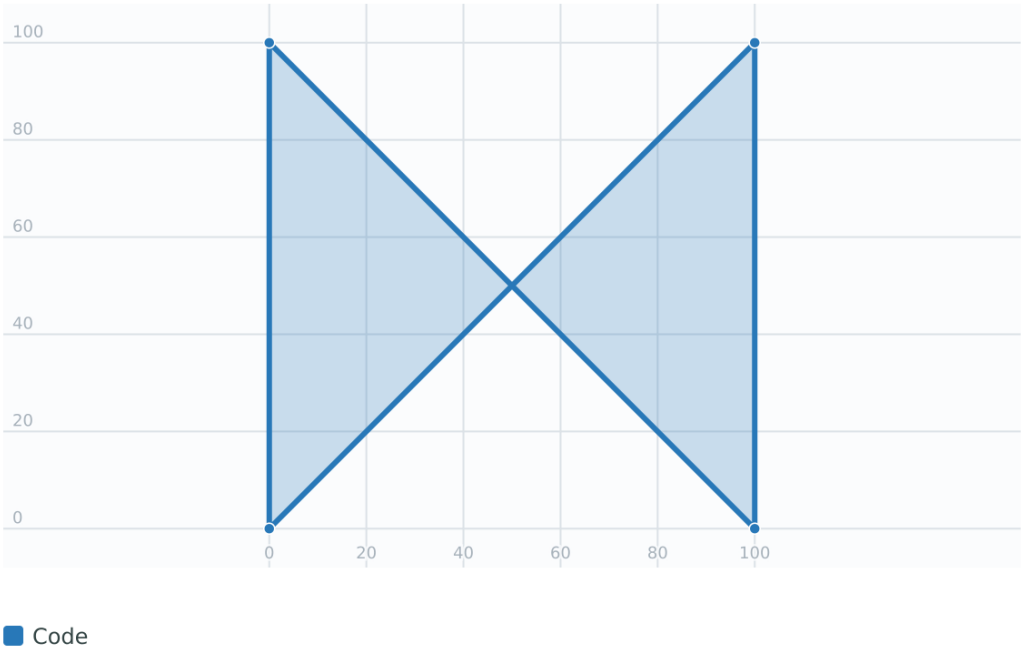
Code

```
SELECT ST_IsValidReason(
    'POLYGON ((100 200,100 100,200 200,
     200 100,100 200))'::geometry) as validity_info;
```

Output

```
Self-intersection[150 150]
```

Figure



This example returns the first three rejects from a successful quintuplet experiment.

Code

```
SELECT gid, ST_IsValidReason(geom) as validity_info
FROM
(SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), array_agg(f.line)) As geom, gid
FROM (SELECT ST_Buffer(ST_Point(x1*10,y1), z1) As buff, x1*10 + y1*100 + z1*1000 As gid
    FROM generate_series(-4,6) x1
    CROSS JOIN generate_series(2,5) y1
    CROSS JOIN generate_series(1,8) z1
    WHERE x1
> y1*0.5 AND z1 < x1*y1) As e
    INNER JOIN (SELECT ST_Translate(ST_ExteriorRing(ST_Buffer(ST_Point(x1*10,y1), z1)),y1 ↩
    *1, z1*2) As line
    FROM generate_series(-3,6) x1
    CROSS JOIN generate_series(2,5) y1
    CROSS JOIN generate_series(1,10) z1
    WHERE x1
> y1*0.75 AND z1 < x1*y1) As f
ON (ST_Area(e.buff)
> 78 AND ST_Contains(e.buff, f.line))
GROUP BY gid, e.buff) As quintuplet_experiment
WHERE ST_IsValid(geom) = false
ORDER BY gid
LIMIT 3;
```

Output

```
 gid  |      validity_info
------+-------------------------
 5330 | Self-intersection [32 5]
 5340 | Self-intersection [42 5]
 5350 | Self-intersection [52 5]
```

Simple example.

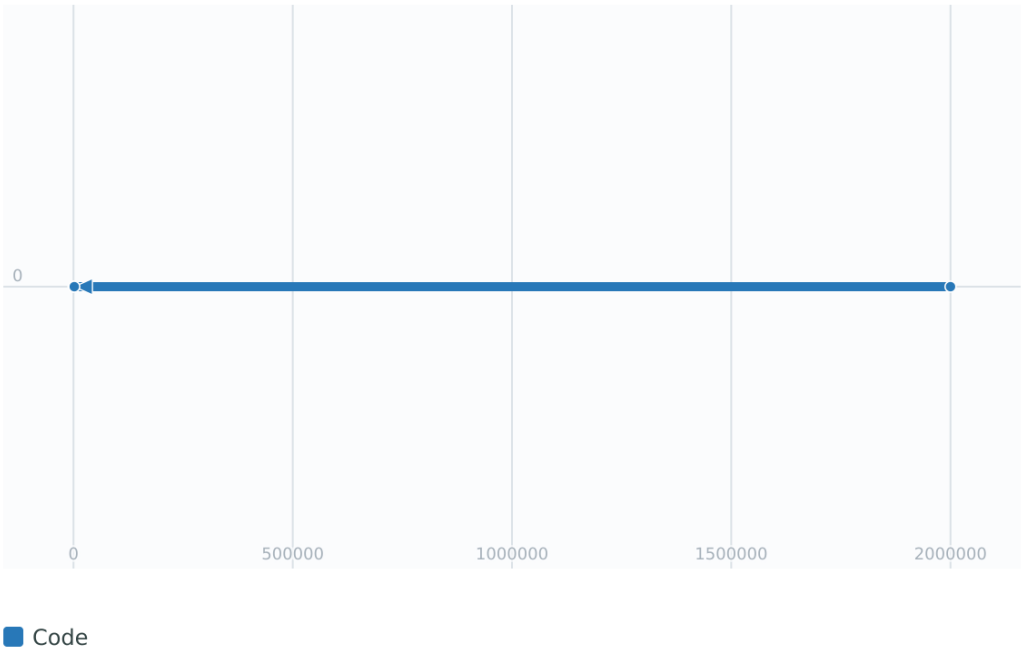
Code

```
SELECT ST_IsValidReason('LINESTRING(220227 150406,2220227 150407,222020 150410)');
```

Output

```
 st_isvalidreason
------------------
 Valid Geometry
```

Figure



Se även

ST_IsValid, ST_Summary

7.6.4 ST_MakeValid

ST_MakeValid — Försöker göra en ogiltig geometri giltig utan att förlora toppar.

Synopsis

geometry **ST_MakeValid**(geometry input);
geometry **ST_MakeValid**(geometry input, text params);

Beskrivning

Funktionen försöker skapa en giltig representation av en given ogiltig geometri utan att förlora någon av de ingående hörnen. Giltiga geometrier returneras oförändrade.

Ingångar som stöds är: POINTS, MULTIPOINTS, LINESTRINGS, MULTILINESTRINGS, POLYGONS, MULTIPOLYGONS och GEOMETRYCOLLECTIONS som innehåller alla blandningar av dem.

Vid fullständig eller partiell dimensionell kollaps kan utdatageometrin vara en samling geometrier med lägre till lika stor dimension, eller en geometri med lägre dimension.

Enstaka polygoner kan bli multigeometrier vid självskärningar.

Argumentet `params` kan användas för att ange en alternativsträng för att välja vilken metod som ska användas för att bygga giltig geometri. Alternativsträngen är i formatet method=linework|structure keepcollapsed=true|false". Om inget params-argument anges kommer linework-algoritmen att användas som standard.

Method-nyckeln har två värden.

- linework"is the default algorithm, and builds valid geometries by first extracting all lines, noding that linework together, then building a value output from the linework. The requirement that no vertices are lost can generate complex collections as outputs.
- structure"is an algorithm that distinguishes between interior and exterior rings, building new geometry by unioning exterior rings, and then differencing all interior rings. The results tend to be more intuitive, where collapsed lines are discarded in the output.

The "keepcollapsed"key is only valid for the structureälgorithm, and takes a value of trueör false".

- Keep collapsed of false"is the default, and means that geometry components that collapse to a lower dimensionality, for example a one-point linestring will be dropped.
- Keep collapsed of truemeans geometry components that collapse to a lower dimensionality will be retained.

Utförs av GEOS-modulen.

Tillgänglighet: 2.0.0

Förbättrad: 2.0.1, hastighetsförbättringar

Förbättrad: 2.1.0, stöd för GEOMETRYCOLLECTION och MULTIPOINT har lagts till.

Förbättrad: 3.1.0, har lagt till borttagning av koordinater med NaN-värden.

Förbättrad: 3.2.0, algoritmalternativen lineworköch structure"har lagts till, vilket kräver GEOS >= 3.10.0.

Denna funktion stöder 3d och kommer inte att tappa z-index.

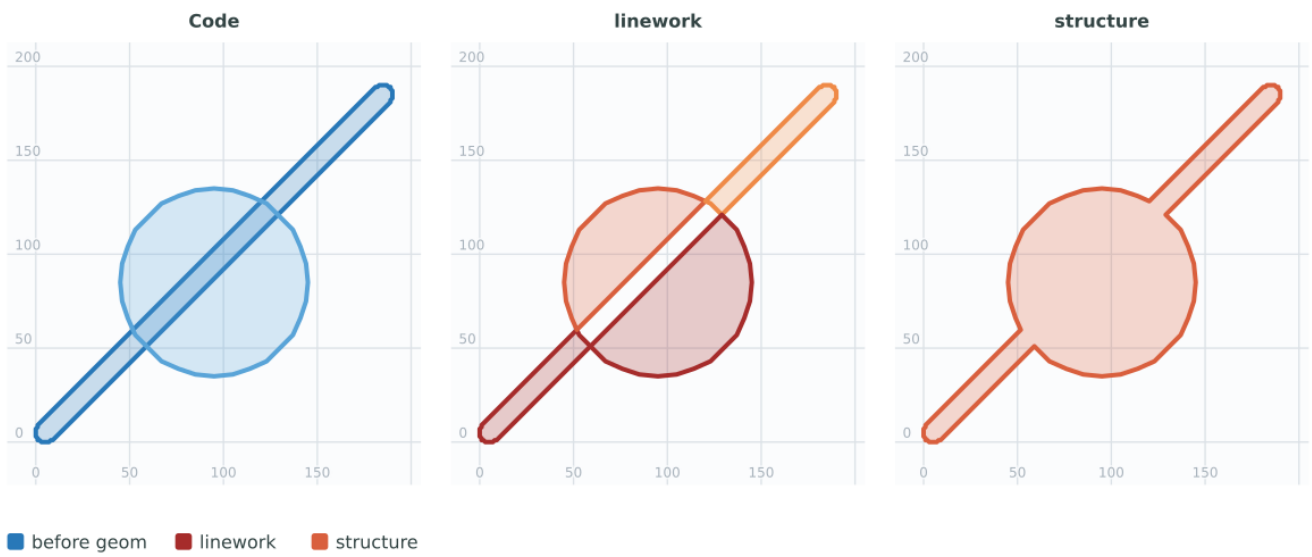
Exempel

Make valid a MultiPolygon of two overlapping polygons using the linework and structure methods.

Code

```
SELECT ST_MakeValid(before_geom) AS linework,
  ST_MakeValid(before_geom, 'method=structure') AS structure
FROM (SELECT 'MULTIPOLYGON(((186 194,187 194,188 195,189 195,190 195,
191 195,192 195,193 194,194 194,194 193,195 192,195 191,
195 190,195 189,195 188,194 187,194 186,14 6,13 6,12 5,11 5,
10 5,9 5,8 5,7 6,6 6,6 7,5 8,5 9,5 10,5 11,5 12,6 13,6 14,186 194)),
((150 90,149 80,146 71,142 62,135 55,128 48,119 44,110 41,100 40,
90 41,81 44,72 48,65 55,58 62,54 71,51 80,50 90,51 100,
54 109,58 118,65 125,72 132,81 136,90 139,100 140,110 139,
119 136,128 132,135 125,142 118,146 109,149 100,150 90)))'::geometry AS before_geom) AS f;
```

Figure

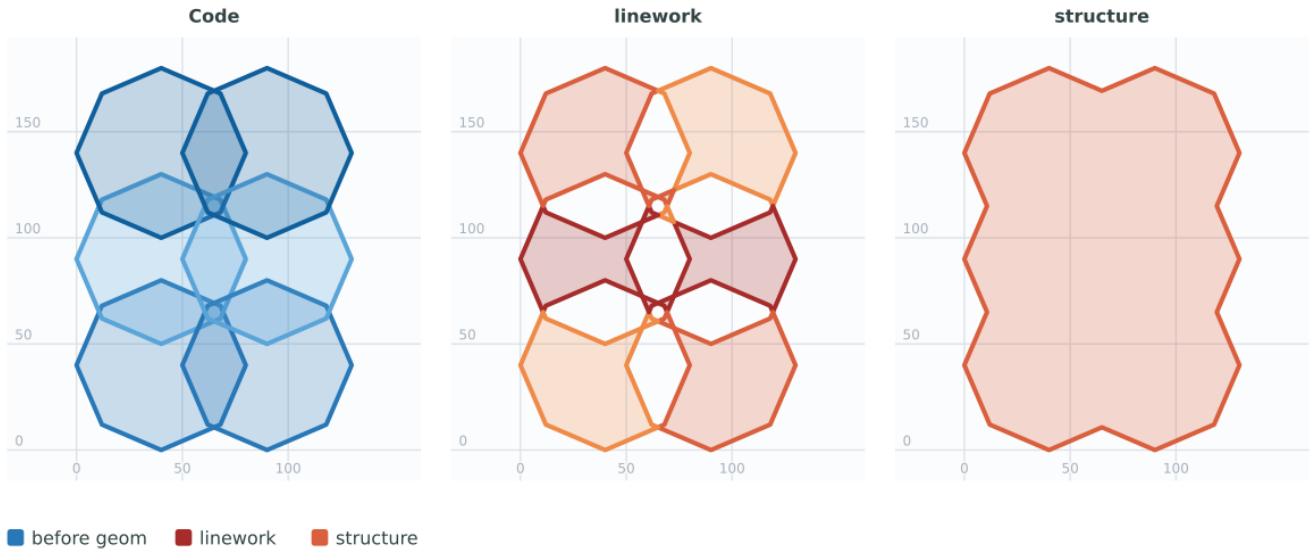


Make valid a MultiPolygon of six overlapping polygons using the linework and structure methods.

Code

```
SELECT ST_MakeValid(before_geom) AS linework,
  ST_MakeValid(before_geom, 'method=structure') AS structure
FROM (SELECT 'MULTIPOLYGON(((91 50,79 22,51 10,23 22,11 50,23 78,51 90,79 78,91 50)),
      ((91 100,79 72,51 60,23 72,11 100,23 128,51 140,79 128,91 100)),
      ((91 150,79 122,51 110,23 122,11 150,23 178,51 190,79 178,91 150)),
      ((141 50,129 22,101 10,73 22,61 50,73 78,101 90,129 78,141 50)),
      ((141 100,129 72,101 60,73 72,61 100,73 128,101 140,129 128,141 100)),
      ((141 150,129 122,101 110,73 122,61 150,73 178,101 190,129 178,141 150)))'::geometry ↩
    AS before_geom) AS c;
```

Figure



Collapsed Geometry Examples

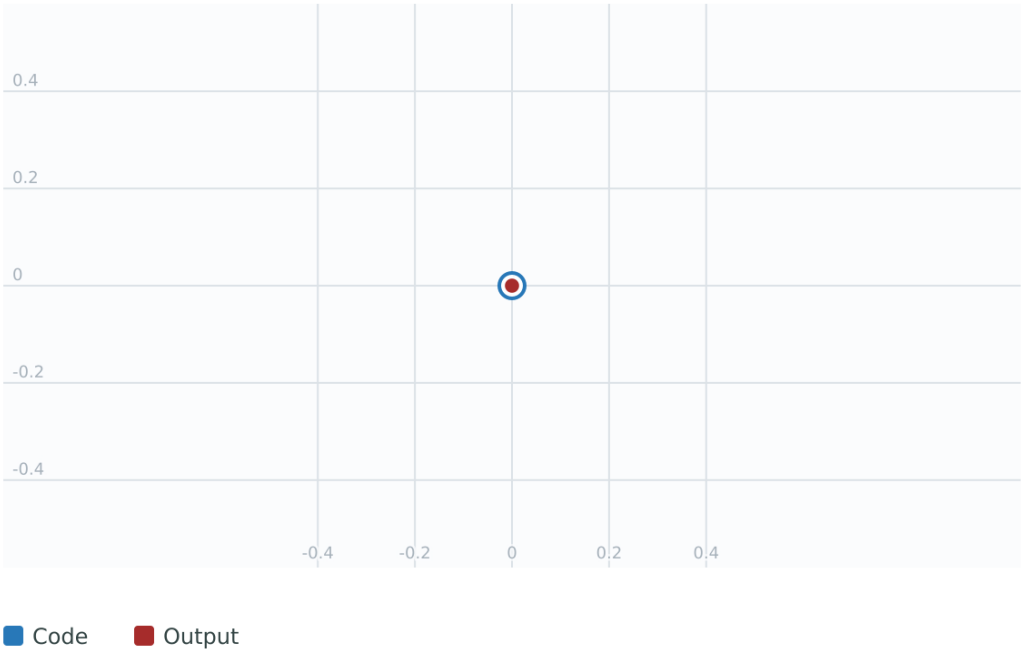
Code

```
SELECT ST_MakeValid(
'LINESTRING(0 0,0 0)',
'method=structure keepcollapsed=true'
);
```

Output

```
PUNKT(0 0)
```

Figure



Code

```
SELECT ST_MakeValid(
'LINESTRING(0 0,0 0)',
'method=structure keepcollapsed=false'
);
```

Output

```
LINESTRING TOM
```

Se även

ST_IsValid, ST_Collect, ST_CollectionExtract

7.7 Funktioner för spatialt referenssystem

7.7.1 ST_InverseTransformPipeline

ST_InverseTransformPipeline — Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem med hjälp av inversen av en definierad pipeline för koordinattransformation.

Synopsis

geometry **ST_InverseTransformPipeline**(geometry geom, text pipeline, integer to_srid);

Beskrivning

Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem med hjälp av en definierad pipeline för koordinattransformation i motsatt riktning.

Se ST_TransformPipeline för mer information om hur du skriver en transformationspipeline.

Tillgänglighet: 3.4.0

SRID för indatageometrin ignoreras och SRID för utdatageometrin sätts till noll om inte ett värde anges via den valfria parametern `to_srid`. När ST_TransformPipeline används körs pipelinen i framåtriktad riktning. Om du använder `ST_InverseTransformPipeline()` körs pipelinen i invers riktning.

Transformationer som använder pipelines är en specialiserad version av ST_Transform. I de flesta fall väljer `ST_Transform` rätt operationer för att konvertera mellan koordinatsystem, och bör därför föredras.

Exempel

Ändra WGS 84 long lat till UTM 31N med hjälp av EPSG:16031-omvandlingen

This example transforms in the inverse direction.

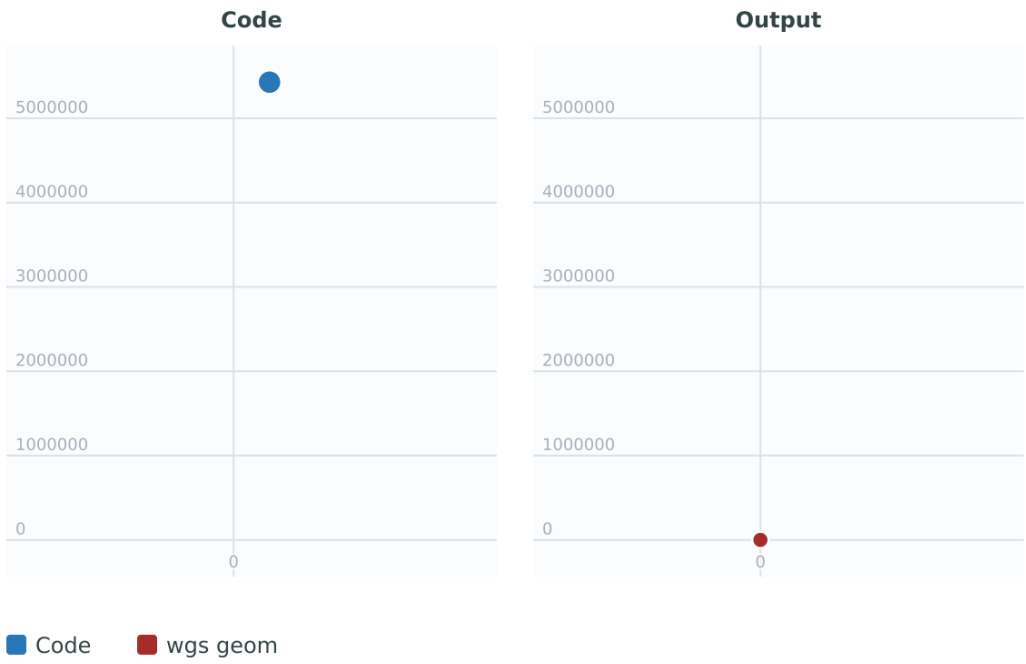
Code

```
SELECT ST_InverseTransformPipeline(
        'POINT(426857.9877165967 5427937.523342293)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::16031'
    ) AS wgs_geom;
```

Output

```
POINT(2 48.99999999999999)
```

Figure



GDA2020 exempel.

This example uses ST_Transform with automatic selection of a conversion pipeline.

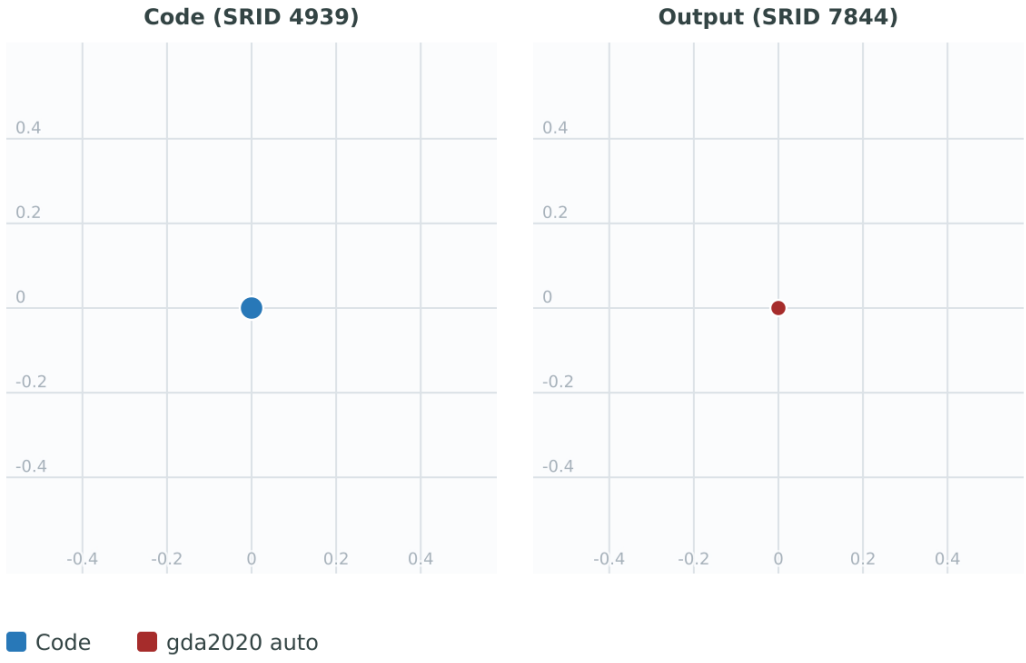
Code

```
SELECT ST_Transform('SRID=4939;POINT(143.0 -37.0)'::geometry, 7844) AS gda2020_auto;
```

Output

```
POINT(143.00000635638918 -36.999986706128176)
```

Figure



Se även

ST_Transform, ST_TransformPipeline

7.7.2 ST_SetSRID

ST_SetSRID — Ställ in SRID på en geometri.

Synopsis

geometry **ST_SetSRID**(geometry geom, integer srid);

Beskrivning

Ställer in SRID på en geometri till ett visst heltalsvärde. Användbart för att konstruera avgränsande rutor för frågor.

Notera
Den här funktionen omvandlar inte geometrikoordinaterna på något sätt - den ställer bara in metadata som definierar det spatiala referenssystem som geometrin antas vara i. Använd ST_Transform om du vill omvandla geometrin till en ny projektion.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Mark a point as WGS 84 longitude/latitude.

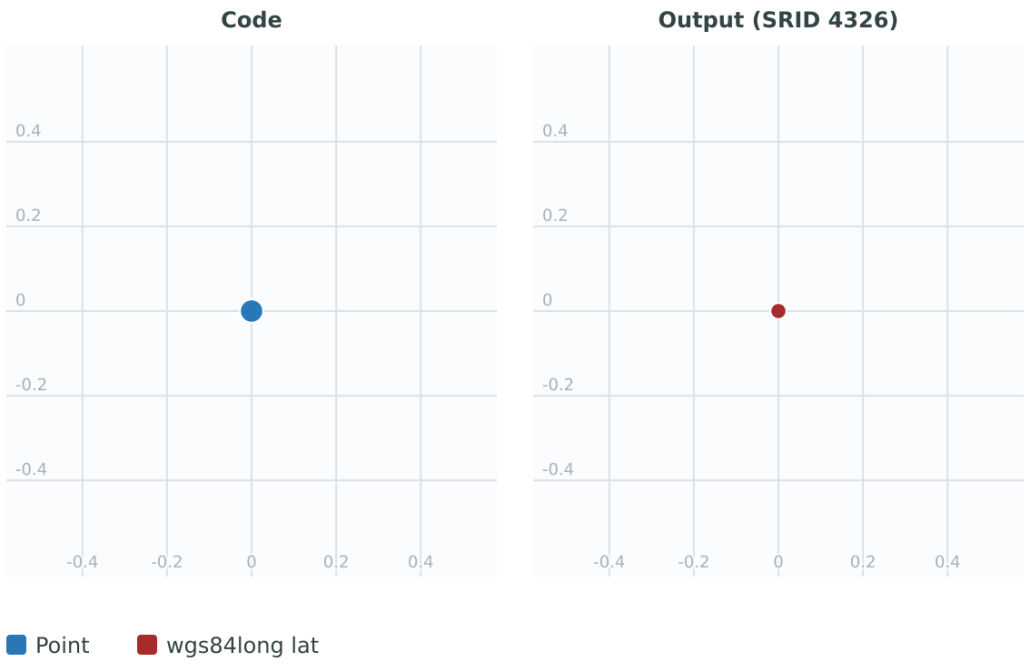
Code

```
SELECT ST_SetSRID(ST_Point(-123.365556, 48.428611), 4326) AS wgs84long_lat;
```

Output

```
SRID=4326;POINT(-123.365556 48.428611)
```

Figure



Mark a point as WGS 84 longitude/latitude and then transform it to Web Mercator.

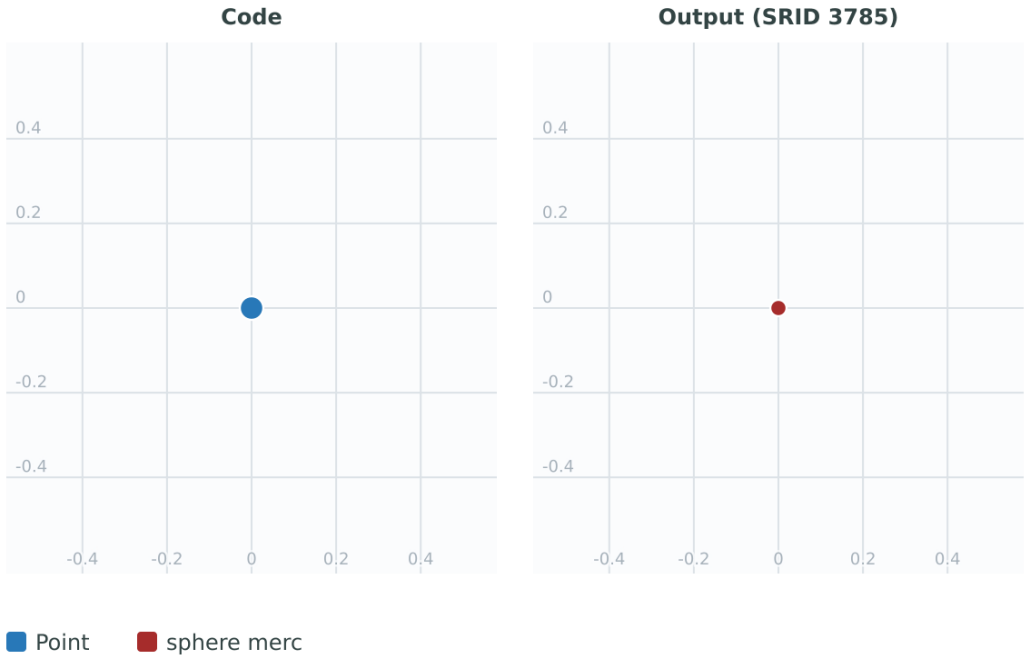
Code

```
SELECT ST_Transform(ST_SetSRID(ST_Point(-123.365556, 48.428611), 4326), 3785) AS ↩
    sphere_merc;
```

Output

```
SRID=3785;POINT(-13732990.8753491 6178458.96425423)
```

Figure



Se även

Avsnitt 4.5, ST_SRID, ST_Transform, UpdateGeometrySRID

7.7.3 ST_SRID

ST_SRID — Returnerar den spatiala referensidentifieraren för en geometri.

Synopsis

integer **ST_SRID**(geometry g1);

Beskrivning

Returnerar den spatiala referensidentifieraren för ST_Geometry enligt definitionen i tabellen spatial_ref_sys. Avsnitt 4.5

Notera
spatial_ref_sys-tabellen är en tabell som katalogiserar alla spatiala referenssystem som är kända för PostGIS och används för transformationer från ett spatialt referenssystem till ett annat. Det är därför viktigt att verifiera att du har rätt identifierare för det spatiala referenssystemet om du planerar att transformera dina geometrier.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.5

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)', 4326));
```

Output

```
st_srid
---------
    4326
(1 row)
```

Se även

Avsnitt 4.5, ST_SetSRID, ST_Transform, ST_SRID, ST_SRID

7.7.4 ST_Transform

ST_Transform — Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem.

Synopsis

geometry **ST_Transform**(geometry g1, integer srid);
geometry **ST_Transform**(geometry geom, text to_proj);
geometry **ST_Transform**(geometry geom, text from_proj, text to_proj);
geometry **ST_Transform**(geometry geom, text from_proj, integer to_srid);

Beskrivning

Returnerar en ny geometri vars koordinater har omvandlats till ett annat spatialt referenssystem. Den spatiala destinationsreferensen `to_srid` kan identifieras med en giltig SRID heltalsparameter (dvs. den måste finnas i tabellen `spatial_ref_sys`). Alternativt kan en spatial referens som definieras som en PROJ.4-sträng användas för `to_proj` och/eller `from_proj`, men dessa metoder är inte optimerade. Om det spatiala referenssystemet för destinationen uttrycks med en PROJ.4-sträng i stället för en SRID, kommer SRID för utdatageometrin att sättas till noll. Med undantag för funktioner med `from_proj` måste indatageometrier ha en definierad SRID.

ST_Transform förväxlas ofta med ST_SetSRID. ST_Transform ändrar faktiskt koordinaterna för en geometri från ett spatialt referenssystem till ett annat, medan ST_SetSRID() helt enkelt ändrar SRID-identifieraren för geometrin.

ST_Transform väljer automatiskt en lämplig konverteringspipeline med tanke på källans och målets spatiala referenssystem. För att använda en specifik konverteringsmetod, använd ST_TransformPipeline.

Notera
Kräver att PostGIS är kompilerat med PROJ-stöd. Använd PostGIS_Full_Version för att bekräfta att du har PROJ-stöd kompilerat.

Notera
Om du använder mer än en transformation är det bra att ha ett funktionellt index på de vanligaste transformationerna för att dra nytta av indexanvändningen.

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Förbättrad: 2.3.0 stöd för direkt PROJ.4-text infördes.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.6

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Ändra Massachusetts statsplan US feet geometri till WGS 84 long lat

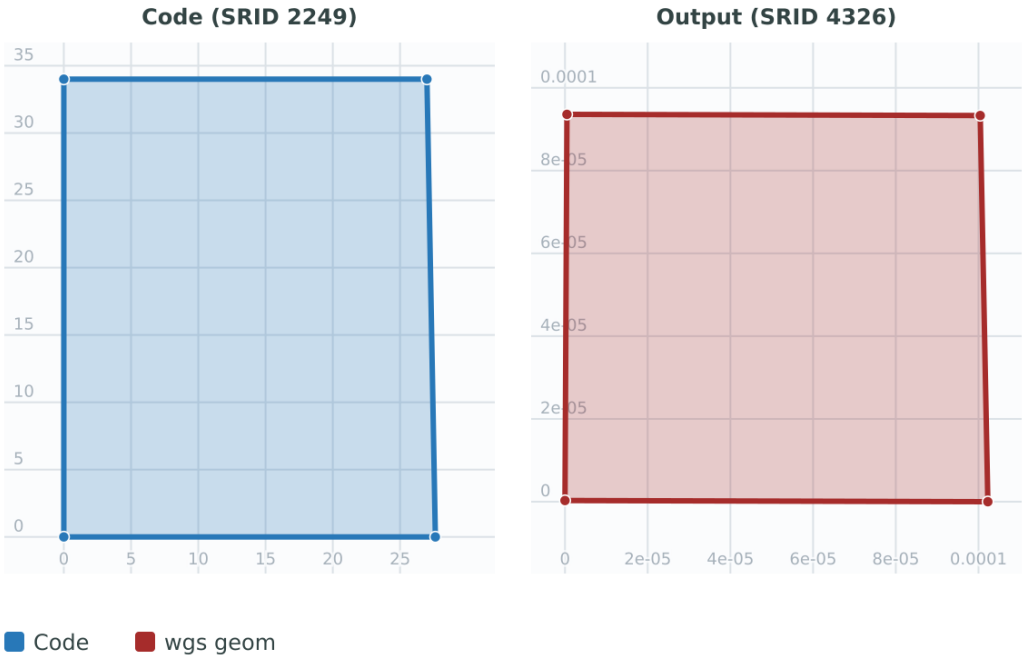
Code

```
SELECT ST_Transform(
  'SRID=2249;POLYGON((743238 2967416,743238 2967450,
    743265 2967450,743265.625 2967416,743238 2967416))'::geometry,
  4326) AS wgs_geom;
```

Output

```
SRID=4326;POLYGON((-71.1776848522251 42.3902896512903,-71.17768437663261 42.390382947801015,-71.17758443054647 ↩
    42.390382667791854,-71.17758259272304 42.390289364798825,-71.1776848522251 42.3902896512903))
```

Figure



3D Circular String example.

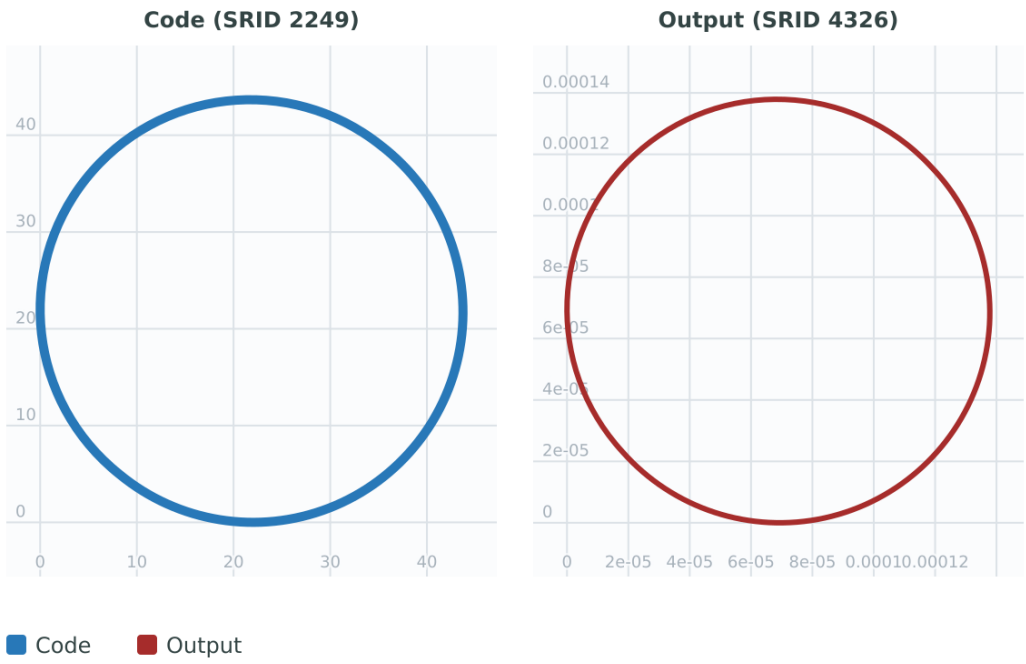
Code

```
SELECT ST_Transform('SRID=2249;CIRCULARSTRING(743238 2967416 1,743238 2967450 2,743265 ↩
    2967450 3,743265.625 2967416 3,743238 2967416 4)'::geometry,
    4326);
```

Output

```
SRID=4326;CIRCULARSTRING(-71.1776848522251 42.3902896512903 1,-71.17768437663261 42.390382947801015 2,
-71.17758443054647 42.390382667791854 3,
-71.17758259272304 42.390289364798825 3,-71.1776848522251 42.3902896512903 4)
```

Figure



Exempel på hur man skapar ett partiellt funktionellt index. För tabeller där du inte är säker på att alla geometrier kommer att fyllas i är det bäst att använda ett partiellt index som utelämnar nollgeometrier, vilket både sparar utrymme och gör ditt index mindre och mer effektivt.

Code

```
CREATE INDEX idx_geom_26986_parcels
  ON parcels
  USING gist
  (ST_Transform(geom, 26986))
  WHERE geom IS NOT NULL;
```

Exempel på användning av PROJ.4-text för att transformera med anpassade spatiala referenser.

Find intersection of two polygons near the North pole, using a custom Gnomic projection. See http://boundles
the-peel/.

Code

```
WITH data AS (
  SELECT
    ST_GeomFromText('POLYGON((170 50,170 72,-130 72,-130 50,170 50))', 4326) AS p1,
    ST_GeomFromText('POLYGON((-170 68,-170 90,-141 90,-141 68,-170 68))', 4326) AS p2,
    '+proj=gnom +ellps=WGS84 +lat_0=70 +lon_0=-160 +no_defs'::text AS gnom
)
SELECT ST_Transform(ST_Intersection(ST_Transform(p1, gnom), ST_Transform(p2, gnom)),
  gnom, 4326)
FROM data;
```

Output

```
POLYGON((-140.9999840290591 73.42768865981984,-141 68,-170 68,-170.0000026915361 74.05492179637055,-140.9999840290591 ↩
    73.42768865981984))
```

Figure

Konfigurera transformationsbeteende

Ibland kan koordinattransformation som involverar en rutnätsförskjutning misslyckas, t.ex. om PROJ.4 inte har byggts med rutnätsförskjutningsfiler eller om koordinaten inte ligger inom det område för vilket rutnätsförskjutningen är definierad. Som standard kommer PostGIS att ge ett felmeddelande om det inte finns någon gridskiftfil, men detta beteende kan konfigureras per SRID antingen genom att testa olika `to_proj-värden` för PROJ.4-text eller genom att ändra `proj4text-värdet` i tabellen `spatial_ref_sys`.

Till exempel är proj4text-parametern +datum=NAD87 en kortform för följande +nadgrids-parameter:

Code

```
+nadgrids=@conus, @alaska, @ntv2_0.gsb, @ntv1_can.dat
```

Prefixet @ innebär att inget fel rapporteras om filerna inte finns, men om slutet av listan nås utan att någon fil har varit lämplig (dvs. hittad och överlappande) så utfärdas ett fel.

Om du däremot vill säkerställa att åtminstone standardfilerna finns, men att en nolltransformation tillämpas om alla filer skannas utan träff, kan du använda:

Code

```
+nadgrids=@conus, @alaska, @ntv2_0.gsb, @ntv1_can.dat, null
```

Null grid shift-filen är en giltig grid shift-fil som täcker hela världen och inte tillämpar någon förskjutning. Så för ett fullständigt exempel, om du ville ändra PostGIS så att transformationer till SRID 4267 som inte låg inom rätt intervall inte kastade ett ERROR, skulle du använda följande:

Code

```
UPDATE spatial_ref_sys SET proj4text = '+proj=longlat +ellps=clrk66 +nadgrids=@conus, ↩
    @alaska,@ntv2_0.gsb,@ntv1_can.dat,null +no_defs' WHERE srid = 4267;
```

Se även

Avsnitt 4.5, ST_SetSRID, ST_SRID, UpdateGeometrySRID, ST_TransformPipeline

7.7.5 ST_TransformPipeline

ST_TransformPipeline — Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem med hjälp av en definierad pipeline för koordinattransformation.

Synopsis

geometry **ST_TransformPipeline**(geometry g1, text pipeline, integer to_srid);

Beskrivning

Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem med hjälp av en definierad pipeline för koordinattransformation.

Transformationspipelines definieras med hjälp av något av följande strängformat:

- `urn:ogc:def:coordinateOperation:AUTHORITY::CODE`. Observera att en enkel `EPSG:CODE` -sträng inte identifierar en koordinatoperation på ett unikt sätt: samma EPSG-kod kan användas för en CRS-definition.
- En PROJ-pipeline-sträng av formen: `+proj=pipeline ...`. Automatisk axelnormalisering kommer inte att tillämpas och vid behov måste den som ringer lägga till ytterligare ett pipelinesteg eller ta bort `axelbytessteg`.
- Sammanlänkade operationer av formen: `urn:ogc:def:coordinateOperation,coordinateOperation:EPS`

Tillgänglighet: 3.4.0

SRID för inmatningsgeometrin ignoreras och SRID för utdatageometrin kommer att sättas till noll om inte ett värde anges via den valfria parametern `to_srid`. När du använder `ST_TransformPipeline()` körs pipelinen i framåtriktad riktning. Om du använder ST_InverseTransformPipeline körs pipelinen i omvänd riktning.

Transformationer som använder pipelines är en specialiserad version av ST_Transform. I de flesta fall väljer `ST_Transform` rätt operationer för att konvertera mellan koordinatsystem, och bör därför föredras.

Exempel

Ändra WGS 84 long lat till UTM 31N med hjälp av EPSG:16031-omvandlingen

This example transforms in the forward direction.

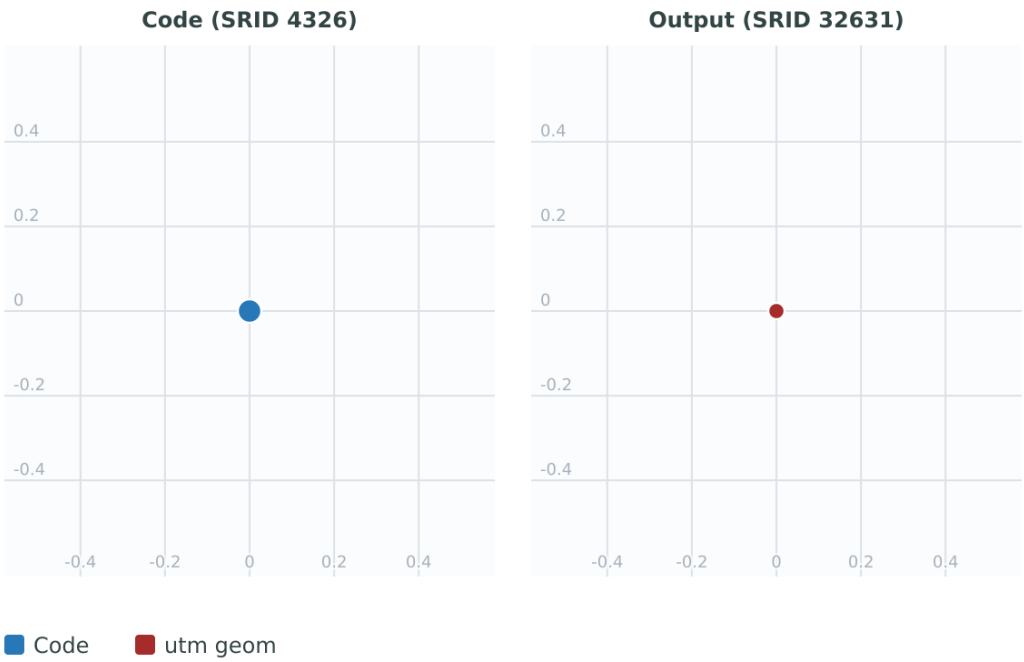
Code

```
SELECT
    ST_SetSRID(ST_TransformPipeline(
        'SRID=4326;POINT(2 49)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::16031'
    ), 32631) AS utm_geom;
```

Output

```
SRID=32631;POINT(426857.9877165967 5427937.523342293)
```

Figure



This example transforms in the inverse direction.

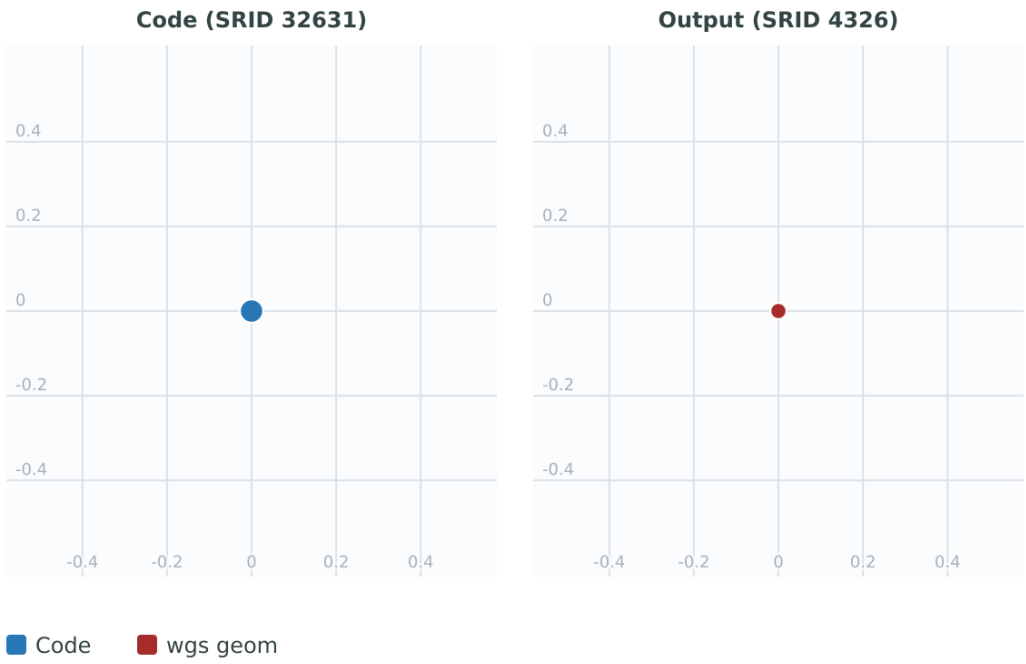
Code

```
SELECT
    ST_SetSRID(ST_InverseTransformPipeline(
        'SRID=32631;POINT(426857.9877165967 5427937.523342293)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::16031'
    ), 4326) AS wgs_geom;
```

Output

```
SRID=4326;POINT(2 48.99999999999999)
```

Figure



GDA2020 exempel.

This example uses ST_Transform with automatic selection of a conversion pipeline.

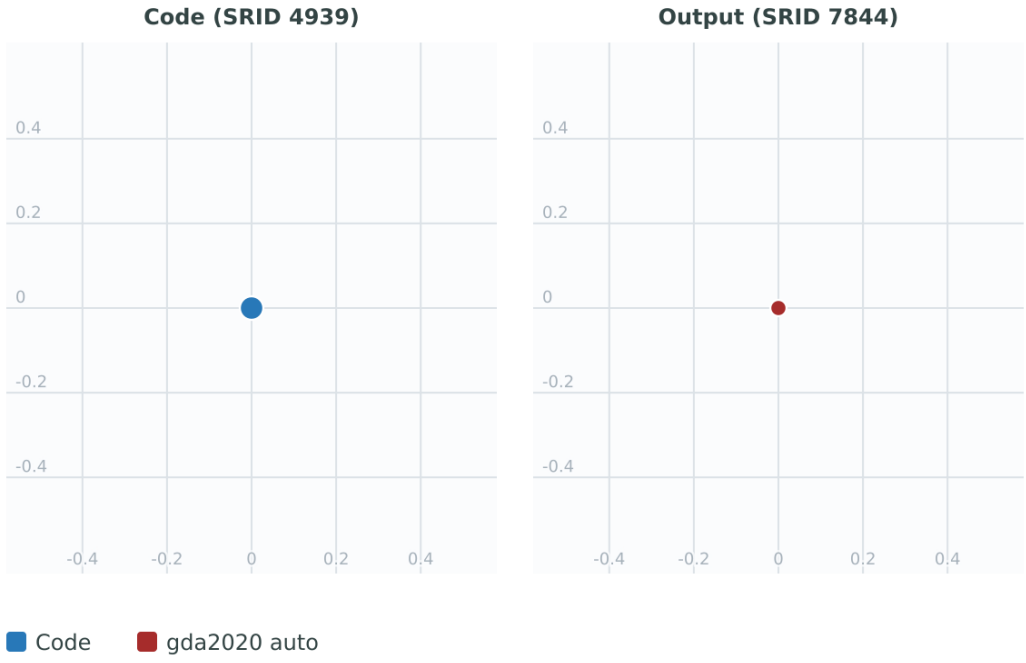
Code

```
SELECT ST_Transform('SRID=4939;POINT(143.0 -37.0)'::geometry, 7844) AS gda2020_auto;
```

Output

```
POINT(143.00000635638918 -36.999986706128176)
```

Figure



Using a defined conversion (EPSG:8447)

Code

```
SELECT
    ST_TransformPipeline(
        'SRID=4939;POINT(143.0 -37.0)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::8447'
    ) AS gda2020_code;
```

Output

```
POINT(143.0000063280214 -36.999986718287545)
```

This example uses a PROJ pipeline definition matching EPSG:8447, as returned from projinfo. Any axisswap steps must be removed.

Code

```
SELECT
    ST_TransformPipeline(
        'SRID=4939;POINT(143.0 -37.0)'::geometry,
        '+proj=pipeline
   +step +proj=unitconvert +xy_in=deg +xy_out=rad
   +step +proj=hgridshift +grids=au_icsm_GDA94_GDA2020_conformal_and_distortion.tif
   +step +proj=unitconvert +xy_in=rad +xy_out=deg'
    ) AS gda2020_pipeline;
```

Output

```
POINT(143.0000063280214 -36.999986718287545)
```

Se även

ST_Transform, ST_InverseTransformPipeline

7.7.6 postgis_srs_codes

postgis_srs_codes — Returnerar listan över SRS-koder som är associerade med den angivna myndigheten.

Synopsis

setof text **postgis_srs_codes**(text auth_name);

Beskrivning

Returnerar en uppsättning av alla `auth_srid` för det angivna `auth_name`.

Tillgänglighet: 3.4.0

Proj version 6+

Exempel

Ange de tio första koderna som är kopplade till myndigheten EPSG.

Code

```
SELECT * FROM postgis_srs_codes('EPSG') ORDER BY 1 LIMIT 10;
```

Output

```
10156
10157
10158
10160
10162
10163
10164
10165
10166
10167
```

Se även

postgis_srs, postgis_srs_all, postgis_srs_search

7.7.7 postgis_srs

postgis_srs — Returnerar en metadatapost för den begärda myndigheten och srid.

Synopsis

setof record **postgis_srs**(text auth_name, text auth_srid);

Beskrivning

Returnerar en metadatapost för den begärda `auth_srid` för det angivna `auth_name`. Posten kommer att innehålla `auth_name`, `auth_srid`, `srname`, `srtext`, `proj4text` och hörnen av användningsområdet, `point_sw` och `point_ne`.

Tillgänglighet: 3.4.0

Proj version 6+

Exempel

Hämta metadata för EPSG:3005.

Code

```
SELECT auth_name, auth_srid, srname,point_sw AS point_sw,point_ne AS point_ne
FROM postgis_srs('EPSG', '3005');
```

Output

```
-[ RECORD 1 ]-------------------
auth_name | EPSG
auth_srid | 3005
srname    | NAD83 / BC Albers
point_sw  | POINT(-139.04 48.25)
point_ne  | POINT(-114.08 60.01)
```

Se även

postgis_srs_codes, postgis_srs_all, postgis_srs_search

7.7.8 postgis_srs_all

postgis_srs_all — Returnera metadataposter för varje spatialt referenssystem i den underliggande Proj-databasen.

Synopsis

postgis_srs_all(void);

Beskrivning

Returnerar en uppsättning av alla metadataposter i den underliggande Proj-databasen. Posterna kommer att ha `auth_name`, `auth_srid`, `srname`, `srtext`, `proj4text` och hörnen på användningsområdet, `point_sw` och `point_ne`.

Tillgänglighet: 3.4.0

Proj version 6+

Exempel

Hämta de 10 första metadataposterna från databasen Proj.

Code

```
SELECT auth_name, auth_srid, srname FROM postgis_srs_all() ORDER BY auth_name, auth_srid ↩
    LIMIT 10;
```

Output

```
auth_name | auth_srid |                                srname
-----------+-----------+-----------------------------------------------------------------------
 EPSG      | 10156     | ETRS89 + MSL UK & Ireland VORF08 depth
 EPSG      | 10157     | ETRS89 + CD UK & Ireland VORF08 depth
 EPSG      | 10158     | S34J-IRF
 EPSG      | 10160     | S34J reconstruction east-orientated
 EPSG      | 10162     | JGD2011 / Japan Plane Rectangular CS I + JGD2011 (vertical) height
 EPSG      | 10163     | JGD2011 / Japan Plane Rectangular CS II + JGD2011 (vertical) height
 EPSG      | 10164     | JGD2011 / Japan Plane Rectangular CS III + JGD2011 (vertical) height
 EPSG      | 10165     | JGD2011 / Japan Plane Rectangular CS IV + JGD2011 (vertical) height
 EPSG      | 10166     | JGD2011 / Japan Plane Rectangular CS V + JGD2011 (vertical) height
 EPSG      | 10167     | JGD2011 / Japan Plane Rectangular CS VI + JGD2011 (vertical) height
```

Se även

postgis_srs_codes, postgis_srs, postgis_srs_search

7.7.9 postgis_srs_search

postgis_srs_search — Returnera metadataposter för projicerade koordinatsystem som har användningsområden som helt innehåller parametern bounds.

Synopsis

setof record **postgis_srs_search**(geometry bounds, text auth_name=EPSG);

Beskrivning

Returnerar en uppsättning metadataposter för projicerade koordinatsystem som har användningsområden som helt innehåller parametern bounds. Varje post kommer att innehålla `auth_name`, `auth_srid`, `srname`, `srtext`, `proj4text` och användningsområdets hörn, `point_sw` och `point_ne`.

Sökningen letar endast efter projicerade koordinatsystem och är avsedd för användare att utforska de möjliga system som fungerar för omfattningen av deras data.

Tillgänglighet: 3.4.0

Proj version 6+

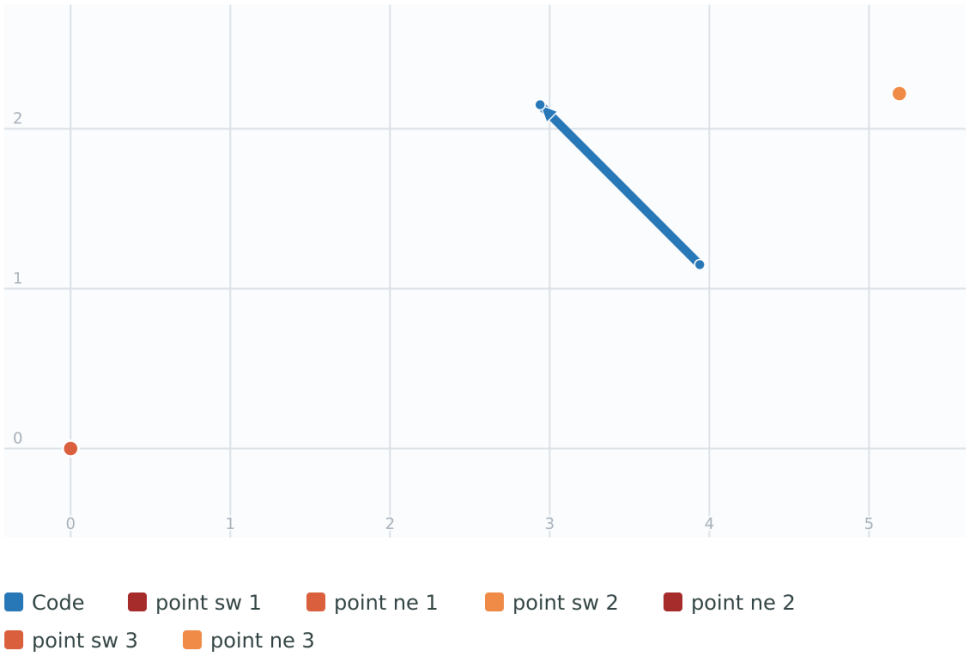
Exempel

Sök efter projicerade koordinatsystem i Louisiana.

Code

```
SELECT auth_name, auth_srid, srname,point_sw AS point_sw,point_ne AS point_ne
FROM postgis_srs_search('SRID=4326;LINESTRING(-90 30,-91 31)')
WHERE auth_srid IN ('2801', '3452', '3457')
ORDER BY auth_name, auth_srid::integer
```

Figure



Skanna en tabell för maximal utsträckning och hitta projicerade koordinatsystem som kan passa.

Code

```
WITH ext AS (
  SELECT ST_Extent(geom) AS geom, Max(ST_SRID(geom)) AS srid
  FROM foo
)
SELECT auth_name, auth_srid, srname,
  point_sw AS point_sw,
  point_ne AS point_ne
FROM ext
CROSS JOIN postgis_srs_search(ST_SetSRID(ext.geom, ext.srid))
LIMIT 3;
```

Se även

postgis_srs_codes, postgis_srs_all, postgis_srs

7.8 Inmatning av geometri

7.8.1 Well-Known Text (WKT)

7.8.1.1 ST_BdPolyFromText

ST_BdPolyFromText — Konstruera en polygon givet en godtycklig samling av slutna linestrings som en MultiLineString Well-Known textrepresentation.

Synopsis

geometry **ST_BdPolyFromText**(text WKT, integer srid);

Beskrivning

Konstruera en polygon givet en godtycklig samling av slutna linestrings som en MultiLineString Well-Known textrepresentation.

Notera
Kastar ett fel om WKT inte är en MULTILINESTRING. Kastar ett fel om utdata är en MULTIPOLYGON; använd ST_BdMPolyFromText i så fall, eller se ST_BuildArea() för en Postgis-specifik metod.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Utförs av GEOS-modulen.

Tillgänglighet: 1.1.0

Se även

ST_BuildArea, ST_BdMPolyFromText

7.8.1.2 ST_BdMPolyFromText

ST_BdMPolyFromText — Konstruera en MultiPolygon givet en godtycklig samling av slutna linestrings som en MultiLineString text representation Well-Known text representation.

Synopsis

geometry **ST_BdMPolyFromText**(text WKT, integer srid);

Beskrivning

Konstruera en polygon givet en godtycklig samling av slutna linestrings, polygoner, MultiLineStrings som Well-Known text representation.

Notera
Ger ett felmeddelande om WKT inte är en MULTILINESTRING. Tvingar fram MULTIPOLYGON-utdata även om resultatet egentligen bara består av en enda POLYGON; använd ST_BdPolyFromText om du är säker på att en enda POLYGON blir resultatet av operationen, eller se ST_BuildArea() för en Postgis-specifik metod.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Utförs av GEOS-modulen.

Tillgänglighet: 1.1.0

Se även

ST_BuildArea, ST_BdPolyFromText

7.8.1.3 ST_GeogFromText

ST_GeogFromText — Returnera ett angivet geografiskt värde från Well-Known Text representation or extended (WKT).

Synopsis

geography **ST_GeogFromText**(text EWKT);

Beskrivning

Returnerar ett geografiskt objekt från den välkända texten eller den utökade välkända representationen. SRID 4326 antas om den inte är specificerad. Detta är ett alias för ST_GeographyFromText. Punkter uttrycks alltid i lång lat-form.

Exempel

Add a WGS 84 geography point column to a table.

Code

```
ALTER TABLE sometable ADD COLUMN geog geography(POINT, 4326);
```

Populate the geography column from longitude and latitude columns.

Code

```
UPDATE sometable
SET geog = ST_GeogFromText('SRID=4326;POINT(' || lon || ' ' || lat || ')');
```

Specify a geography point using EPSG:4267, NAD27.

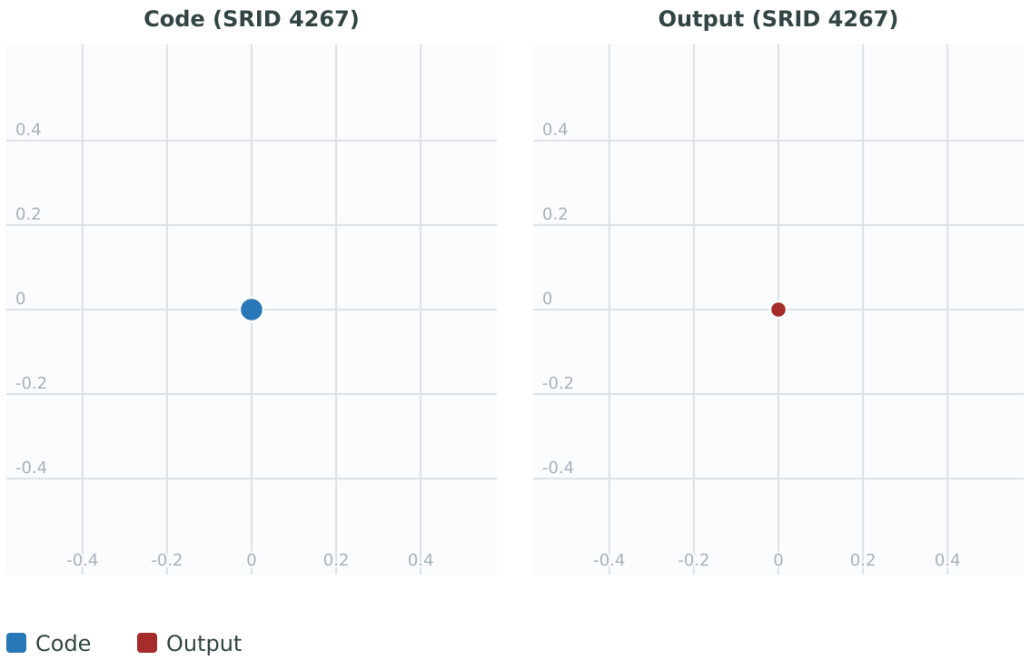
Code

```
SELECT ST_GeogFromText('SRID=4267;POINT(-77.0092 38.889588)');
```

Output

```
SRID=4267;POINT(-77.0092 38.889588)
```

Figure



Se även

ST_AsText, ST_GeographyFromText

7.8.1.4 ST_GeographyFromText

ST_GeographyFromText — Returnera ett angivet geografiskt värde från Well-Known Text representation or extended (WKT).

Synopsis

geography **ST_GeographyFromText**(text EWKT);

Beskrivning

Returnerar ett geografiskt objekt från den välkända textrepresentationen. SRID 4326 antas om den inte är specificerad.

Se även

ST_GeogFromText, ST_AsText

7.8.1.5 ST_GeomCollFromText

ST_GeomCollFromText — Skapar en geometrisk samling från samlingen WKT med angiven SRID. Om SRID inte anges är standardvärdet 0.

Synopsis

geometry **ST_GeomCollFromText**(text WKT, integer srid);
geometry **ST_GeomCollFromText**(text WKT);

Beskrivning

Skapar en geometrisk samling från WKT-representationen (Well-Known-Text) med angiven SRID. Om SRID inte anges är standardvärdet 0.

OGC SPEC 3.2.6.2 - alternativet SRID är från överensstämmelsesviten

Returnerar null om WKT inte är en GEOMETRYCOLLECTION

Notera
Om du är helt säker på att alla dina WKT-geometrier är samlingar ska du inte använda den här funktionen. Den är långsammare än ST_GeomFromText eftersom den lägger till ytterligare ett valideringssteg.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Denna metod implementerar SQL/MM-specifikationen.

Exempel

Code

```
SELECT ST_GeomCollFromText('GEOMETRYCOLLECTION(POINT(1 2), LINESTRING(1 2,3 4))');
```

Se även

ST_GeomFromText, ST_SRID

7.8.1.6 ST_GeomFromEWKT

ST_GeomFromEWKT — Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Text representation (EWKT).

Synopsis

geometry **ST_GeomFromEWKT**(text EWKT);

Beskrivning

Konstruerar ett PostGIS ST_Geometry-objekt från OGC Extended Well-Known text (EWKT)-representationen

Notera
EWKT-formatet är inte en OGC-standard, utan ett PostGIS-specifikt format som inkluderar identifieraren för det spatiala referenssystemet (SRID)

Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Create a LineString with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;LINESTRING(-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932)') AS geom;
```

Output

```
geometry_type | srid
--------------+------
 ST_LineString | 4269
(1 row)
```

Create a MultiLineString with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;MULTILINESTRING((-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932))') AS geom;
```

Output

```
geometry_type    | srid
--------------------+------
 ST_MultiLineString  | 4269
(1 row)
```

Create a Point with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;POINT(-71.064544 42.28787)') AS geom;
```

Output

```
geometry_type | srid
--------------+------
 ST_Point      | 4269
(1 row)
```

Create a Polygon with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;POLYGON((-71.1776585052917 ↩
    42.3902909739571,-71.1776820268866 42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))') AS geom;
```

Output

```
geometry_type | srid
--------------+------
 ST_Polygon    | 4269
(1 row)
```

Create a MultiPolygon with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;MULTIPOLYGON(((-71.1031880899493 42.3152774590236,
-71.1031627617667 42.3152960829043,-71.102923838298 42.3149156848307,
-71.1023097974109 42.3151969047397,-71.1019285062273 42.3147384934248,
-71.102505233663 42.3144722937587,-71.10277487471 42.3141658254797,
-71.103113945163 42.3142739188902,-71.10324876416 42.31402489987,
-71.1033002961013 42.3140393340215,-71.1033488797549 42.3139495090772,
-71.103396240451 42.3138632439557,-71.1041521907712 42.3141153348029,
-71.1041411411543 42.3141545014533,-71.1041287795912 42.3142114839058,
-71.1041188134329 42.3142693656241,-71.1041112482575 42.3143272556118,
-71.1041072845732 42.3143851580048,-71.1041057218871 42.3144430686681,
-71.1041065602059 42.3145009876017,-71.1041097995362 42.3145589148055,
-71.1041166403905 42.3146168544148,-71.1041258822717 42.3146748022936,
-71.1041375307579 42.3147318674446,-71.1041492906949 42.3147711126569,
-71.1041598612795 42.314808571739,-71.1042515013869 42.3151287620809,
-71.1041173835118 42.3150739481917,-71.1040809891419 42.3151344119048,
-71.1040438678912 42.3151191367447,-71.1040194562988 42.3151832057859,
-71.1038734225584 42.3151140942995,-71.1038446938243 42.3151006300338,
-71.1038315271889 42.315094347535,-71.1037393329282 42.315054824985,
-71.1035447555574 42.3152608696313,-71.1033436658644 42.3151648370544,
-71.1032580383161 42.3152269126061,-71.103223066939 42.3152517403219,
-71.1031880899493 42.3152774590236)),
((-71.1043632495873 42.315113108546,-71.1043583974082 42.3151211109857,
-71.1043443253471 42.3150676015829,-71.1043850704575 42.3150793250568,-71.1043632495873 ↩
    42.315113108546)))') AS geom;
```

Output

```
geometry_type   | srid
------------------+------
 ST_MultiPolygon   | 4269
(1 row)
```

Create a 3D CircularString without an SRID.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)') AS ↩
    geom;
```

Output

```
geometry_type    | srid
-------------------+------
 ST_CircularString  | 0
(1 row)
```

Create a PolyhedralSurface without an SRID.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('POLYHEDRALSURFACE(
    ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
    ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
    ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
)') AS geom;
```

Output

```
geometry_type       | srid
-------------------------+------
 ST_PolyhedralSurface     | 0
(1 row)
```

Se även

ST_AsEWKT, ST_GeomFromText

7.8.1.7 ST_GeomFromMARC21

ST_GeomFromMARC21 — Tar MARC21/XML-geografiska data som indata och returnerar ett PostGIS-geometriobjekt.

Synopsis

geometry **ST_GeomFromMARC21** (text marcxml);

Beskrivning

Denna funktion skapar en PostGIS-geometri från en MARC21/XML-post, som kan innehålla en `POINT` eller en `POLYGON`. Om det finns flera geografiska dataposter i samma MARC21/XML-post returneras en `MULTIPOINT` eller `MULTIPOLYGON`. Om posten innehåller blandade geometrityper returneras en `GEOMETRYCOLLECTION`. Den returnerar NULL om MARC21/XML-posten inte innehåller några geografiska data (datafield:034).

Stöd för LOC MARC21/XML-versioner:

- MARC21/XML 1.1

Tillgänglighet: 3.3.0, kräver libxml2 2.6+

Notera
MARC21/XML Coded Cartographic Mathematical Data tillhandahåller för närvarande inte något sätt att beskriva det spatiala referenssystemet för de kodade koordinaterna, så denna funktion returnerar alltid en geometri med `SRID 0`.

Notera
Återlämnade `POLYGON-geometrier` kommer alltid att vara orienterade medurs.

Exempel

Konvertering av MARC21/XML geografiska data som innehåller en enda `POINT` kodad som `hddd.ddddddd`

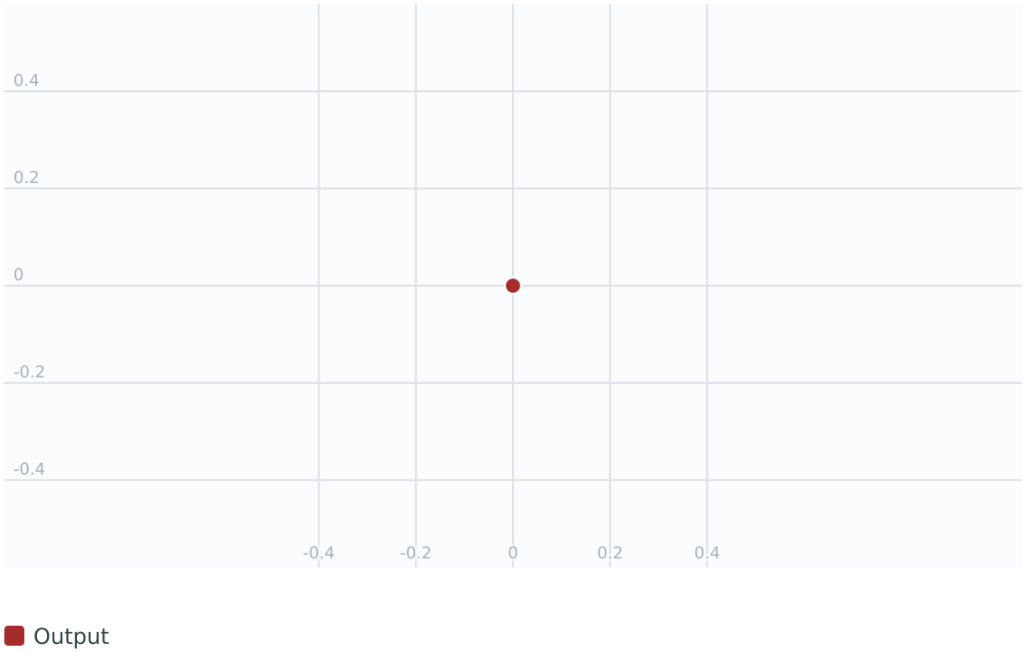
Code

```
SELECT ST_GeomFromMARC21(
'<record xmlns="http://www.loc.gov/MARC21/slim">
    <leader
>00000nz a2200000nc 4500</leader>
    <controlfield tag="001"
>040277569</controlfield>
    <datafield tag="034" ind1=" " ind2=" ">
        <subfield code="d"
>W004.500000</subfield>
        <subfield code="e"
>W004.500000</subfield>
        <subfield code="f"
>N054.250000</subfield>
        <subfield code="g"
>N054.250000</subfield>
    </datafield>
</record
>'
);
```

Output

```
POINT(-4.5 54.25)
```

Figure



Konvertering av MARC21/XML geografiska data som innehåller en enda `POLYGON` kodad som `hdddmmss`

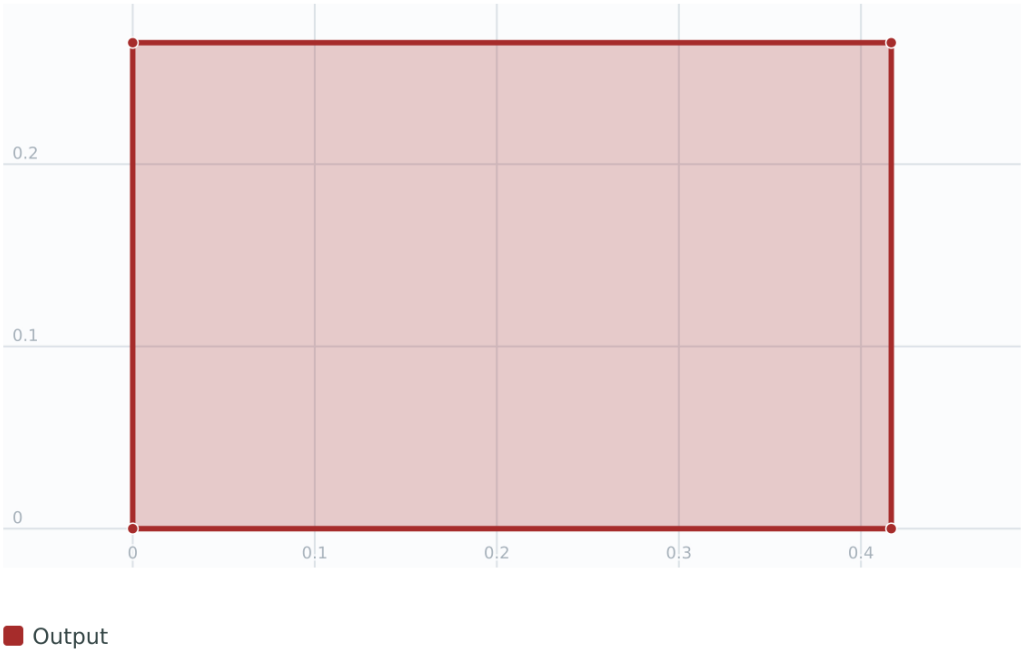
Code

```
SELECT ST_GeomFromMARC21(
'<record xmlns="http://www.loc.gov/MARC21/slim">
    <leader
>01062cem a2200241 a 4500</leader>
    <controlfield tag="001"
>   84696781 </controlfield>
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="b"
>50000</subfield>
        <subfield code="d"
>E0130600</subfield>
        <subfield code="e"
>E0133100</subfield>
        <subfield code="f"
>N0523900</subfield>
        <subfield code="g"
>N0522300</subfield>
    </datafield>
</record
>'
);
```

Output

```
POLYGON((13.1 52.65,13.516666666666667 52.65,13.516666666666667 52.38333333333333,13.1 52.38333333333333,13.1 52.65))
```

Figure



Konvertering av geografiska data i MARC21/XML som innehåller en `POLYGON` och en `POINT`:

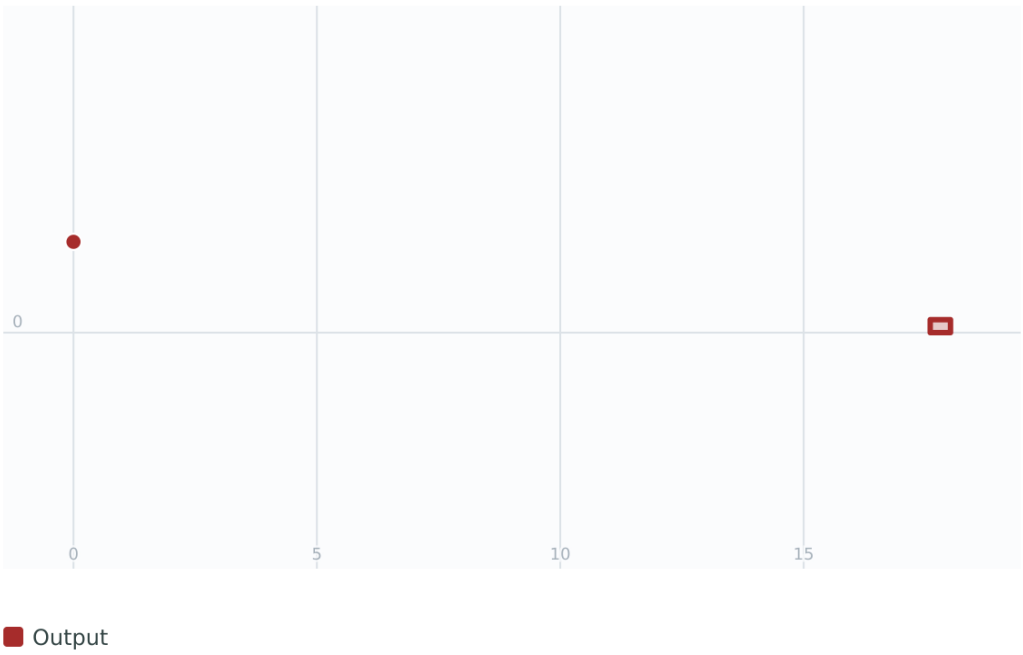
Code

```
SELECT ST_GeomFromMARC21(
'<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="b"
>50000</subfield>
        <subfield code="d"
>E0130600</subfield>
        <subfield code="e"
>E0133100</subfield>
        <subfield code="f"
>N0523900</subfield>
        <subfield code="g"
>N0522300</subfield>
    </datafield>
    <datafield tag="034" ind1=" " ind2=" ">
        <subfield code="d"
>W004.500000</subfield>
        <subfield code="e"
>W004.500000</subfield>
        <subfield code="f"
>N054.250000</subfield>
        <subfield code="g"
>N054.250000</subfield>
    </datafield>
</record
>'
);
```

Output

```
GEOMETRYCOLLECTION(POLYGON((13.1 52.65,13.516666666666667 52.65,13.516666666666667 52.38333333333333,13.1 ↩
    52.38333333333333,13.1 52.65)), POINT(-4.5 54.25))
```

Figure



Se även

ST_AsMARC21

7.8.1.8 ST_GeometryFromText

ST_GeometryFromText — Returns a geometry from Well-Known Text (WKT). Alias for ST_GeomFromText.

Synopsis

geometry **ST_GeometryFromText**(text WKT);
geometry **ST_GeometryFromText**(text WKT, integer srid);

Beskrivning

This function is an alias for ST_GeomFromText and provides the same signatures.

Tillgänglighet: 1.2.2

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.40

Se även

ST_GeomFromText

7.8.1.9 ST_GeomFromText

ST_GeomFromText — Returnera ett specificerat ST_Geometry-värde från Well-Known Text representation (WKT).

Synopsis

geometry **ST_GeomFromText**(text WKT);
geometry **ST_GeomFromText**(text WKT, integer srid);

Beskrivning

Konstruerar ett PostGIS ST_Geometry-objekt från OGC Well-Known-textrepresentationen.

Notera
Det finns två varianter av funktionen ST_GeomFromText. Den första tar ingen SRID och returnerar en geometri utan definierat spatialt referenssystem (SRID=0). Den andra tar en SRID som andra argument och returnerar en geometri som innehåller denna SRID som en del av sina metadata.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2 - alternativet SRID är från överensstämmelsesviten.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.40

Denna metod stöder cirkulära strängar och kurvor.

Notera
Även om det inte är OGC-kompatibelt är ST_MakePoint snabbare än ST_GeomFromText och ST_PointFromText. Det är också lättare att använda för numeriska koordinatvärden. ST_Point är ett annat alternativ som liknar ST_MakePoint i hastighet och är OGC-kompatibelt, men stöder inte annat än 2D-punkter.

Ändrad: 2.0.0 I tidigare versioner av PostGIS var ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)') tillåtet. Detta är nu olagligt i PostGIS 2.0.0 för att bättre överensstämma med SQL/MM-standarder. Detta ska nu skrivas som ST_GeomFromText('GEOMETRYCOLLECTION EMPTY')

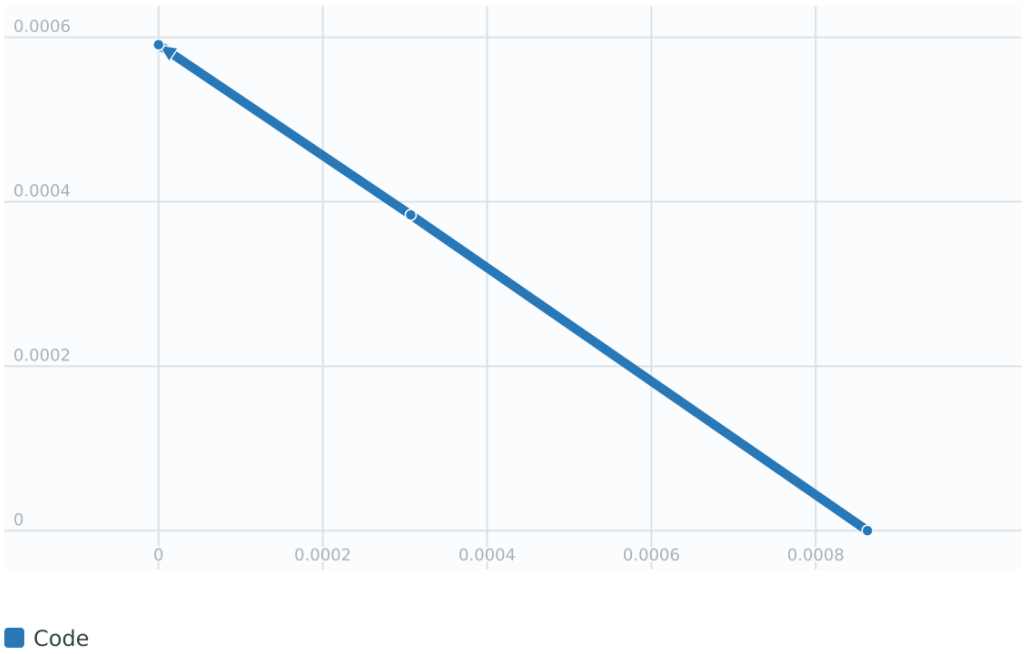
Exempel

Create a LineString with SRID 0.

Code

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932)'));
```

Figure

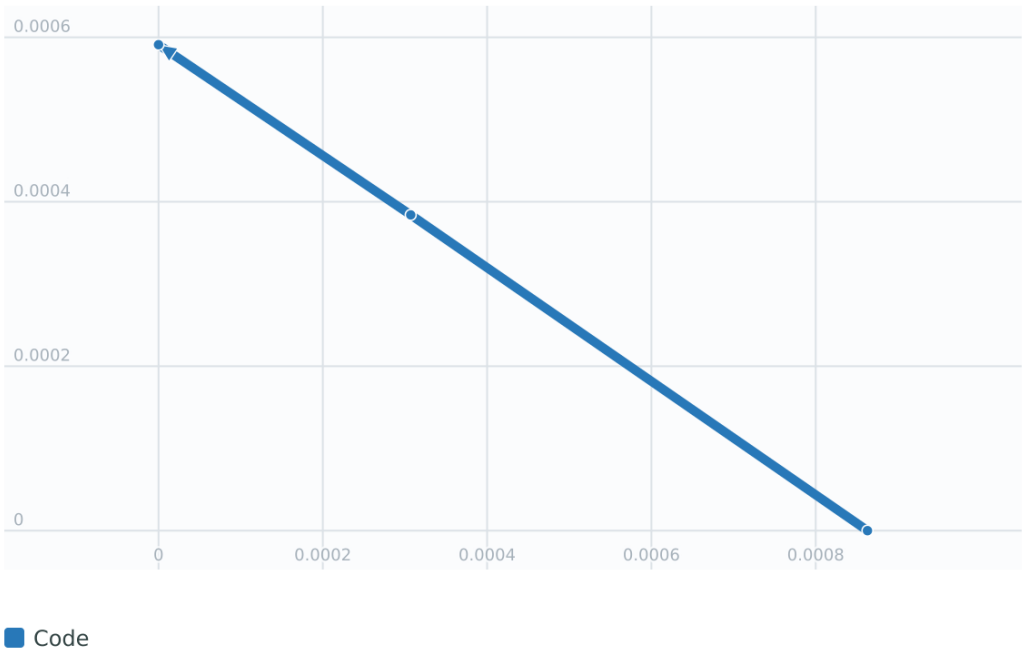


Create the same LineString with SRID 4269.

Code

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932)', 4269));
```

Figure

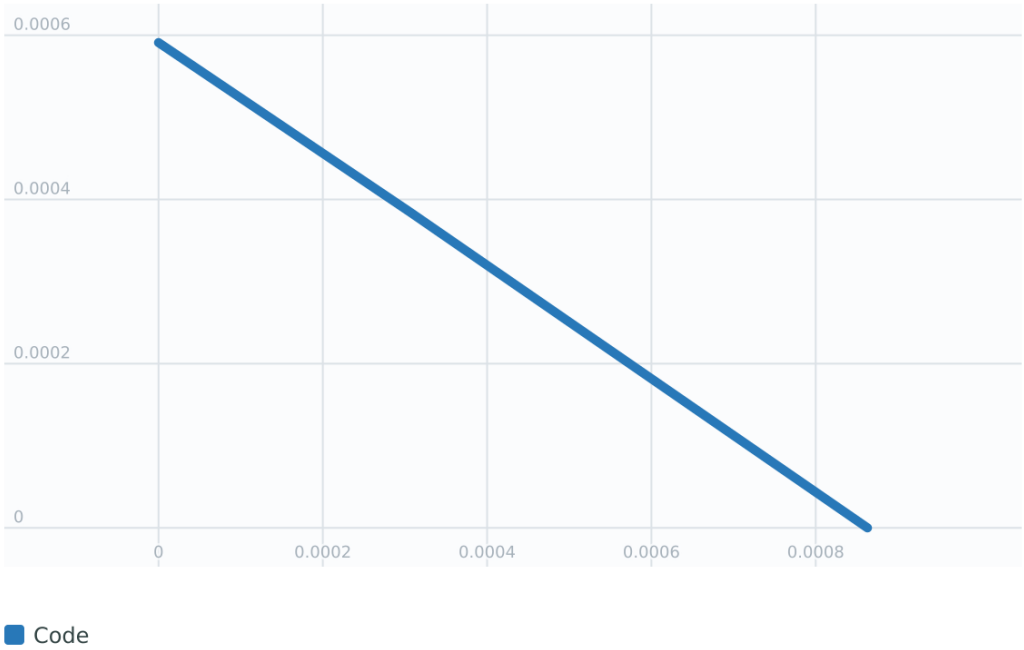


Create a MultiLineString.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('MULTILINESTRING((-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932))'));
```

Figure



Create a Point.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('POINT(-71.064544 42.28787)'));
```

Output

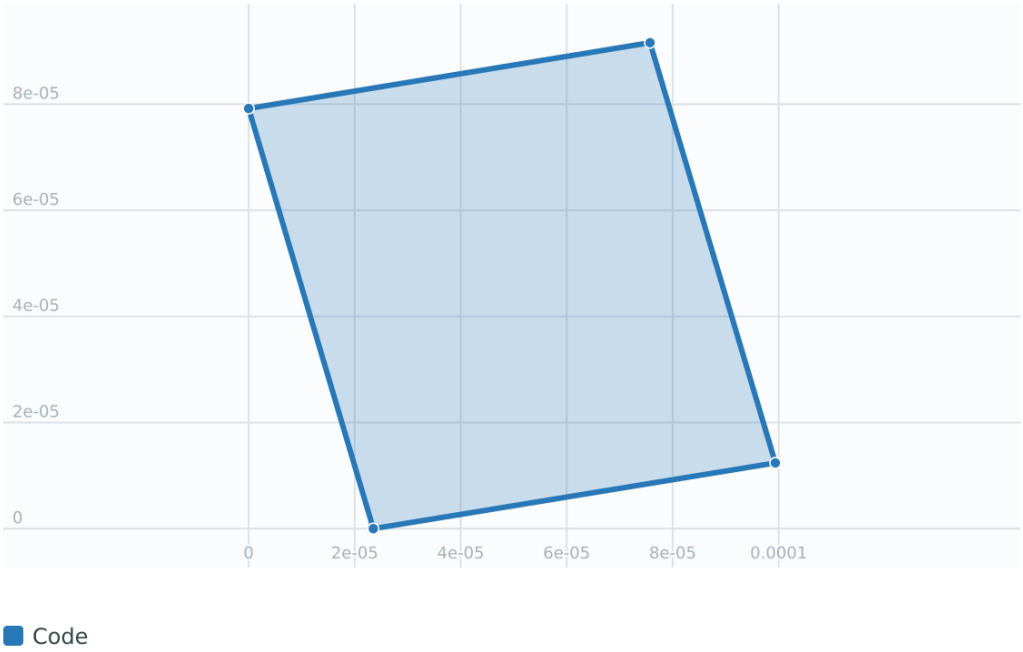
```
ST_Point
```

Create a Polygon.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('POLYGON((-71.1776585052917 ↩
    42.3902909739571,-71.1776820268866 42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))'));
```

Figure

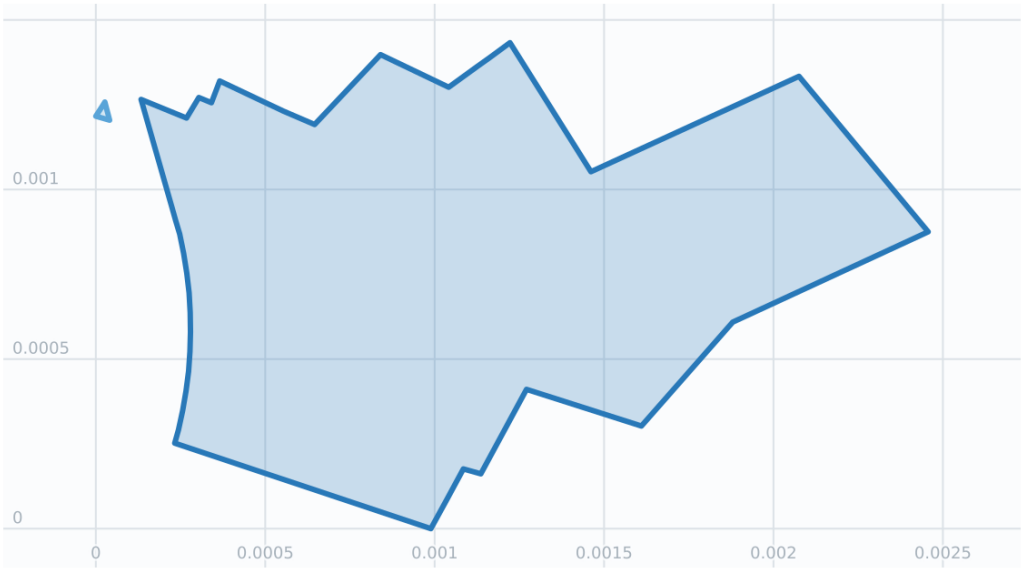


Create a MultiPolygon with SRID 4326.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('MULTIPOLYGON(((-71.1031880899493 42.3152774590236,
-71.1031627617667 42.3152960829043,-71.102923838298 42.3149156848307,
-71.1023097974109 42.3151969047397,-71.1019285062273 42.3147384934248,
-71.102505233663 42.3144722937587,-71.10277487471 42.3141658254797,
-71.103113945163 42.3142739188902,-71.10324876416 42.31402489987,
-71.1033002961013 42.3140393340215,-71.1033488797549 42.3139495090772,
-71.103396240451 42.3138632439557,-71.1041521907712 42.3141153348029,
-71.1041411411543 42.3141545014533,-71.1041287795912 42.3142114839058,
-71.1041188134329 42.3142693656241,-71.1041112482575 42.3143272556118,
-71.1041072845732 42.3143851580048,-71.1041057218871 42.3144430686681,
-71.1041065602059 42.3145009876017,-71.1041097995362 42.3145589148055,
-71.1041166403905 42.3146168544148,-71.1041258822717 42.3146748022936,
-71.1041375307579 42.3147318674446,-71.1041492906949 42.3147711126569,
-71.1041598612795 42.314808571739,-71.1042515013869 42.3151287620809,
-71.1041173835118 42.3150739481917,-71.1040809891419 42.3151344119048,
-71.1040438678912 42.3151191367447,-71.1040194562988 42.3151832057859,
-71.1038734225584 42.3151140942995,-71.1038446938243 42.3151006300338,
-71.1038315271889 42.315094347535,-71.1037393329282 42.315054824985,
-71.1035447555574 42.3152608696313,-71.1033436658644 42.3151648370544,
-71.1032580383161 42.3152269126061,-71.103223066939 42.3152517403219,
-71.1031880899493 42.3152774590236)),
((-71.1043632495873 42.315113108546,-71.1043583974082 42.3151211109857,
-71.1043443253471 42.3150676015829,-71.1043850704575 42.3150793250568,-71.1043632495873 ↩
    42.315113108546)))', 4326));
```

Figure

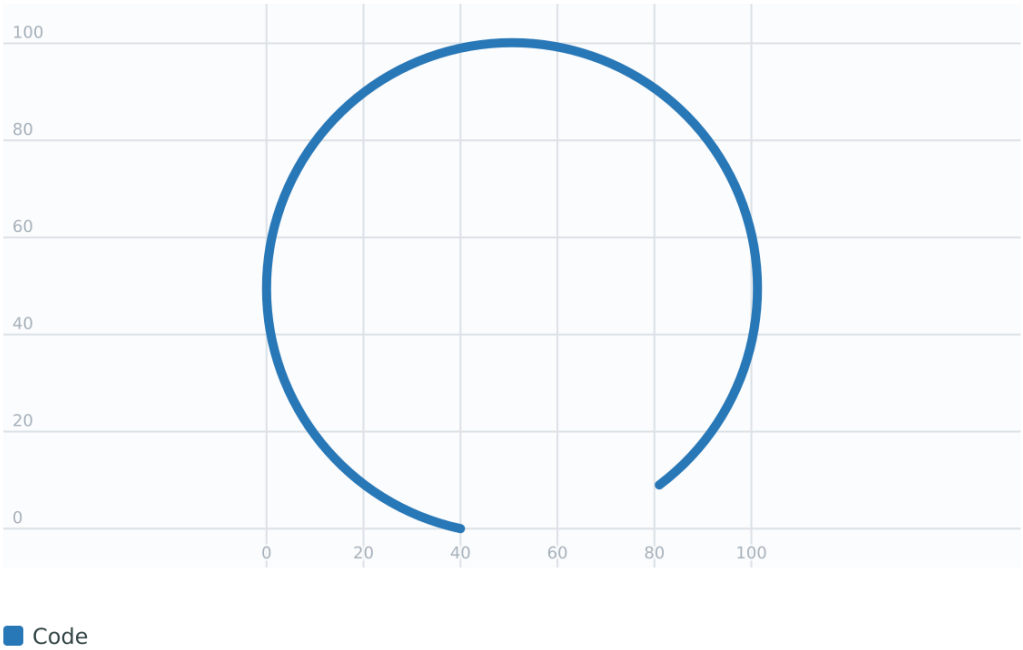


Create a CircularString.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 ↩
    150406)'));
```

Figure



Se även

ST_GeomFromEWKT, ST_GeomFromWKB, ST_SRID

7.8.1.10 ST_LineFromText

ST_LineFromText — Skapar en geometri från en WKT-representation med angiven SRID. Om SRID inte anges är standardvärdet 0.

Synopsis

geometry **ST_LineFromText**(text WKT);
geometry **ST_LineFromText**(text WKT, integer srid);

Beskrivning

Skapar en geometri från WKT med angiven SRID. Om SRID inte anges är standardvärdet 0. Om WKT som skickas in inte är en LINESTRING returneras null.

Notera
OGC SPEC 3.2.6.2 - valfri SRID är från överensstämmelsesviten.

Notera
Om du vet att alla dina geometrier är LINESTRINGS är det mer effektivt att bara använda ST_GeomFromText. Detta anropar bara ST_GeomFromText och lägger till ytterligare validering att den returnerar en linestrings.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.2.8

Exempel

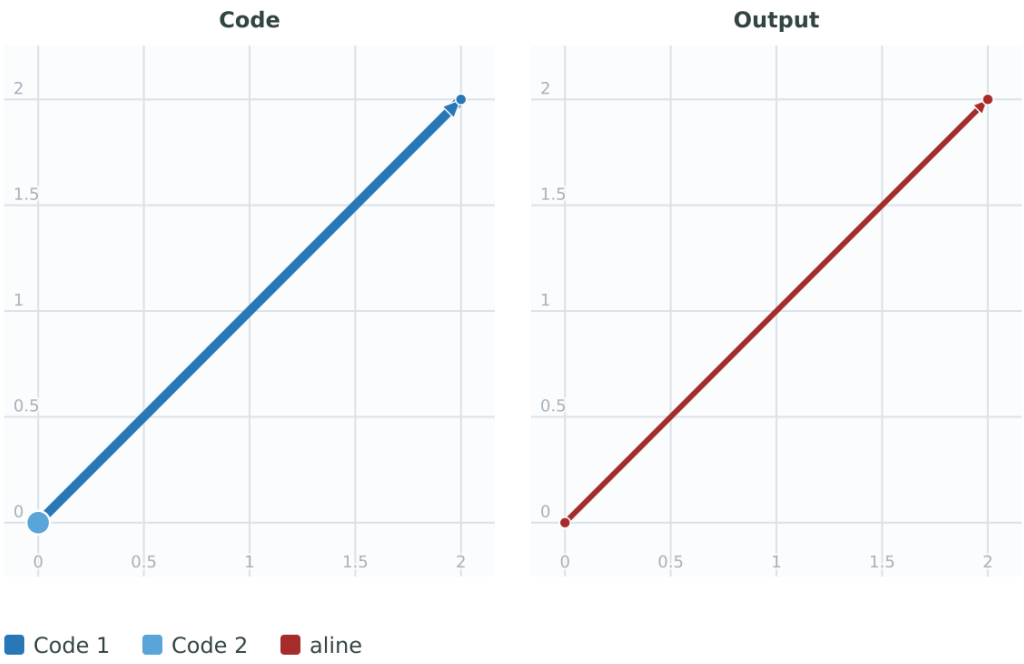
Code

```
SELECT ST_LineFromText('LINESTRING(1 2,3 4)') AS aline,
ST_LineFromText('POINT(1 2)') IS NULL AS null_return;
```

Output

```
aline                | null_return
-----------------------------------
LINESTRING(1 2,3 4)  | t
```

Figure



Se även

ST_GeomFromText

7.8.1.11 ST_MLineFromText

ST_MLineFromText — Returnera ett specificerat ST_MultiLineString-värde från WKT-representation.

Synopsis

geometry **ST_MLineFromText**(text WKT, integer srid);
geometry **ST_MLineFromText**(text WKT);

Beskrivning

Skapar en geometri från Well-Known-Text (WKT) med angiven SRID. Om SRID inte anges är standard-värdet 0.

OGC SPEC 3.2.6.2 - alternativet SRID är från överensstämmelsesviten

Returnerar null om WKT inte är en MULTILINESTRING

Notera
Om du är helt säker på att alla dina WKT-geometrier är punkter ska du inte använda den här funktionen. Den är långsammare än ST_GeomFromText eftersom den lägger till ytterligare ett valideringssteg.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 9.4.4

Exempel

Code

```
SELECT ST_MLineFromText('MULTILINESTRING((1 2,3 4),(4 5,6 7))');
```

Se även

ST_GeomFromText

7.8.1.12 ST_MPointFromText

ST_MPointFromText — Skapar en geometri från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.

Synopsis

geometry **ST_MPointFromText**(text WKT, integer srid);
geometry **ST_MPointFromText**(text WKT);

Beskrivning

Skapar en geometri från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.

OGC SPEC 3.2.6.2 - alternativet SRID är från överensstämmelsesviten

Returnerar null om WKT inte är en MULTIPOINT

Notera
Om du är helt säker på att alla dina WKT-geometrier är punkter ska du inte använda den här funktionen. Den är långsammare än ST_GeomFromText eftersom den lägger till ytterligare ett valideringssteg.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. 3.2.6.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 9.2.4

Exempel

Code

```
SELECT ST_MPointFromText('MULTIPOINT((1 2),(3 4))');
SELECT ST_MPointFromText('MULTIPOINT((-70.9590 42.1180),(-70.9611 42.1223))', 4326);
```

Se även

ST_GeomFromText

7.8.1.13 ST_MPolyFromText

ST_MPolyFromText — Skapar en MultiPolygon Geometry från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.

Synopsis

geometry **ST_MPolyFromText**(text WKT, integer srid);
geometry **ST_MPolyFromText**(text WKT);

Beskrivning

Skapar en MultiPolygon från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.

OGC SPEC 3.2.6.2 - alternativet SRID är från överensstämmelsesviten

Kastar ett fel om WKT inte är en MULTIPOLYGON

Notera
Om du är helt säker på att alla dina WKT-geometrier är multipolygoner ska du inte använda den här funktionen. Den är långsammare än ST_GeomFromText eftersom den lägger till ytterligare ett valideringssteg.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 9.6.4

Exempel

Create a 3D MultiPolygon with an interior ring and SRID 0.

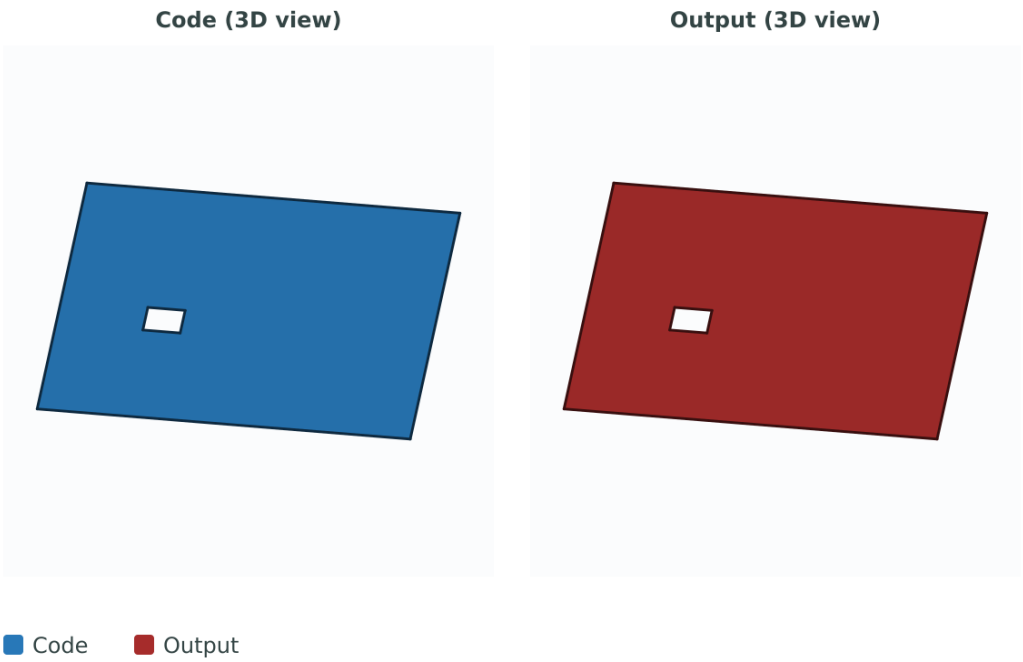
Code

```
SELECT ST_MPolyFromText(
    'MULTIPOLYGON(((0 0 1,20 0 1,20 20 1,0 20 1,0 0 1),
                   (5 5 3,5 7 3,7 7 3,7 5 3,5 5 3)))'
);
```

Output

```
MULTIPOLYGON(((0 0 1,20 0 1,20 20 1,0 20 1,0 0 1),(5 5 3,5 7 3,7 7 3,7 5 3,5 5 3)))
```

Figure

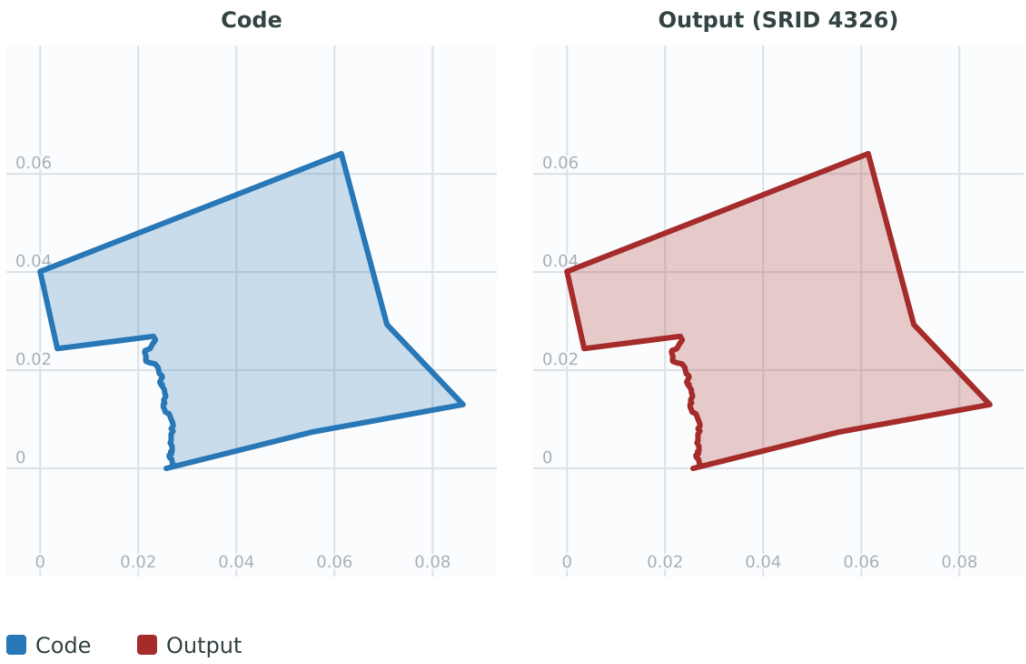


Create a MultiPolygon with SRID 4326.

Code

```
SELECT ST_MPolyFromText(
    'MULTIPOLYGON(((-70.916 42.1002,-70.9468 42.0946,-70.9765 42.0872,
      -70.9754 42.0875,-70.9749 42.0879,-70.9752 42.0881,-70.9754 42.0891,
      -70.9758 42.0894,-70.9759 42.0897,-70.9759 42.0899,-70.9754 42.0902,
      -70.9756 42.0906,-70.9753 42.0907,-70.9753 42.0917,-70.9757 42.0924,
      -70.9755 42.0928,-70.9755 42.0942,-70.9751 42.0948,-70.9755 42.0953,
      -70.9751 42.0958,-70.9751 42.0962,-70.9759 42.0983,-70.9767 42.0987,
      -70.9768 42.0991,-70.9771 42.0997,-70.9771 42.1003,-70.9768 42.1005,
      -70.977 42.1011,-70.9766 42.1019,-70.9768 42.1026,-70.9769 42.1033,
      -70.9775 42.1042,-70.9773 42.1043,-70.9776 42.1043,-70.9778 42.1048,
      -70.9773 42.1058,-70.9774 42.1061,-70.9779 42.1065,-70.9782 42.1078,
      -70.9788 42.1085,-70.9798 42.1087,-70.9806 42.109,-70.9807 42.1093,
      -70.9806 42.1099,-70.9809 42.1109,-70.9808 42.1112,-70.9798 42.1116,
      -70.9792 42.1127,-70.979 42.1129,-70.9787 42.1134,-70.979 42.1139,
      -70.9791 42.1141,-70.9987 42.1116,-71.0022 42.1273,-70.9408 42.1513,
      -70.9315 42.1165,-70.916 42.1002)))',
    4326
);
```

Figure



Se även

ST_GeomFromText, ST_SRID

7.8.1.14 ST_PointFromText

ST_PointFromText — Skapar en punktgeometri från WKT med angiven SRID. Om SRID inte anges är standardvärdet okänt.

Synopsis

geometry **ST_PointFromText**(text WKT);
geometry **ST_PointFromText**(text WKT, integer srid);

Beskrivning

Konstruerar ett PostGIS ST_Geometry-punktobjekt från OGC Well-Known-textrepresentationen. Om SRID inte anges är standardvärdet okänt (för närvarande 0). Om geometrin inte är en WKT-punktrepresentat returneras null. Om WKT är helt ogiltig, kastas ett fel.

Notera
Det finns 2 varianter av ST_PointFromText-funktionen, den första tar ingen SRID och returnerar en geometri utan definierat spatialt referenssystem. Den andra tar ett id för spatial referens som det andra argumentet och returnerar en ST_Geometry som innehåller denna srid som en del av dess metadata. SRID:en måste vara definierad i tabellen spatial_ref_sys.

Notera
Om du är helt säker på att alla dina WKT-geometrier är punkter ska du inte använda den här funktionen. Den är långsammare än ST_GeomFromText eftersom den lägger till ytterligare ett valideringssteg. Om du bygger punkter från långa lat-koordinater och bryr dig mer om prestanda och noggrannhet än OGC-överensstämmelse, använd ST_MakePoint eller OGC-kompatibelt alias ST_Point.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2 - alternativet SRID är från överensstämmelsesviten.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 6.1.8

Exempel

Create a point with the default SRID 0.

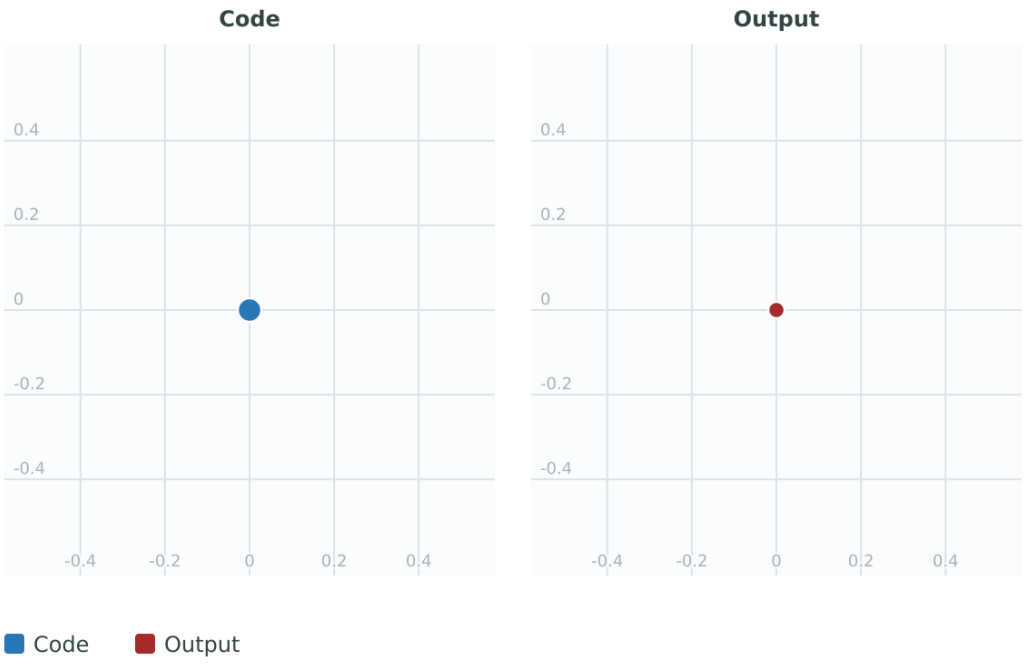
Code

```
SELECT ST_PointFromText('POINT(-71.064544 42.28787)');
```

Output

```
POINT(-71.064544 42.28787)
```

Figure



Create a point with SRID 4326.

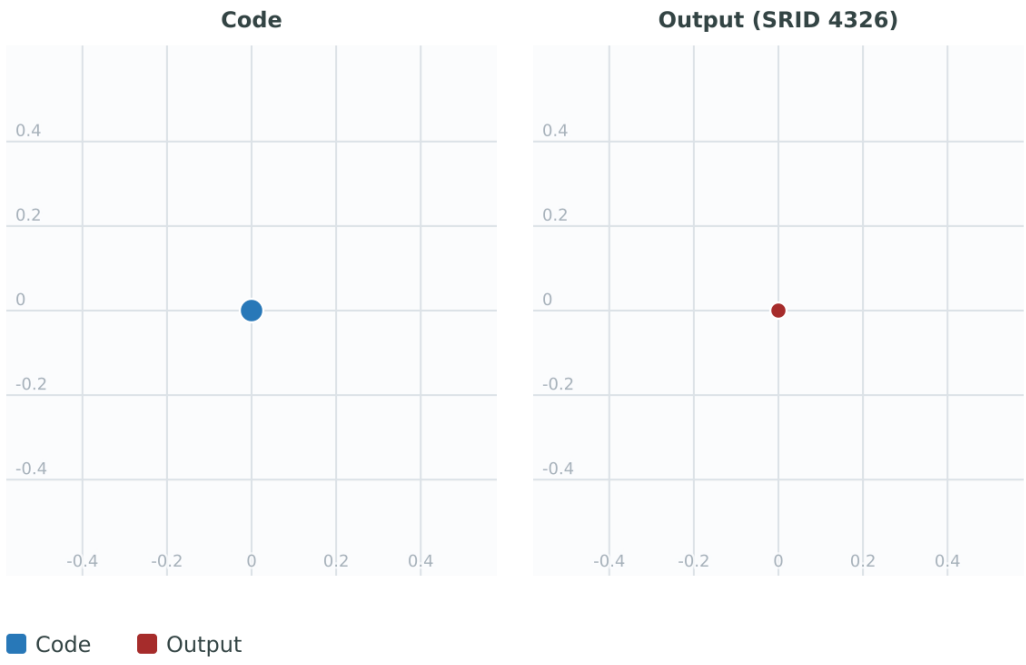
Code

```
SELECT ST_PointFromText('POINT(-71.064544 42.28787)', 4326);
```

Output

```
SRID=4326;POINT(-71.064544 42.28787)
```

Figure



Se även

ST_GeomFromText, ST_MakePoint, ST_Point, ST_SRID

7.8.1.15 ST_PolygonFromText

ST_PolygonFromText — Skapar en geometri från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.

Synopsis

geometry **ST_PolygonFromText**(text WKT);
geometry **ST_PolygonFromText**(text WKT, integer srid);

Beskrivning

Skapar en geometri från WKT med angiven SRID. Om SRID inte anges är standardvärdet 0. Returnerar null om WKT inte är en polygon.

OGC SPEC 3.2.6.2 - alternativet SRID är från överensstämmelsesviten

Notera
Om du är helt säker på att alla dina WKT-geometrier är polygoner ska du inte använda den här funktionen. Den är långsammare än ST_GeomFromText eftersom den lägger till ytterligare ett valideringssteg.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.3.6

Exempel

The result is returned as a native geometry. The documentation test runner compares that value directly with the readable EWKT shown below and uses the same geometry for the generated figure.

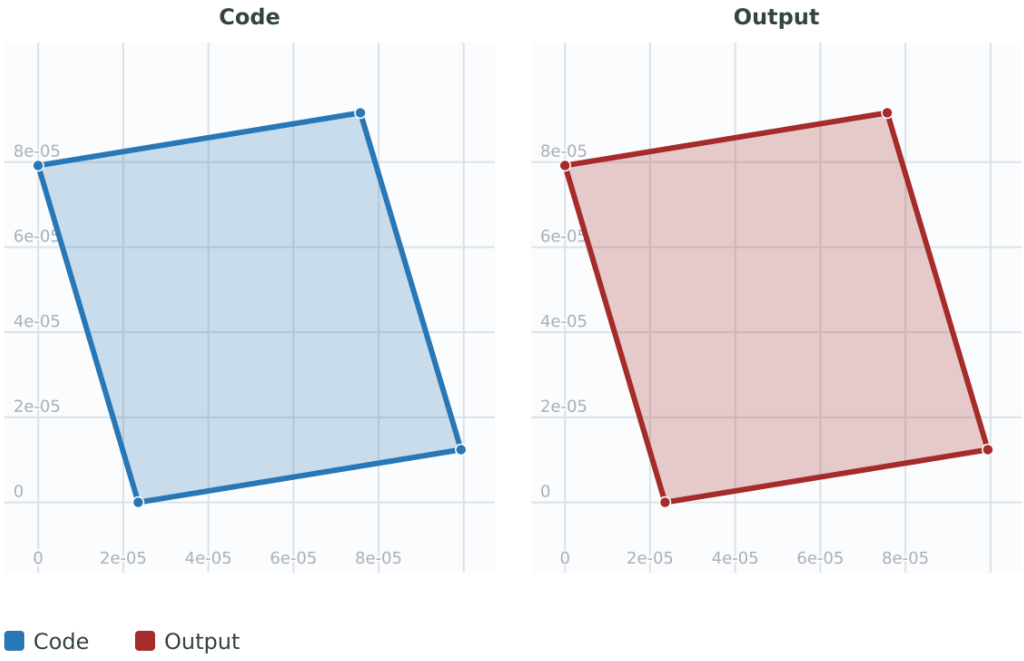
Code

```
SELECT ST_PolygonFromText('POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 ↩
    42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))');
```

Output

```
POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 42.3903701743239,-71.1776063012595 ↩
    42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 42.3902909739571))
```

Figure



Code

```
SELECT ST_PolygonFromText('POINT(1 2)') IS NULL AS point_is_notpoly;
```

Output

```
t
```

Se även

ST_GeomFromText

7.8.1.16 ST_WKTToSQL

ST_WKTToSQL — Returns a geometry from Well-Known Text (WKT). SQL/MM alias for ST_GeomFromText.

Synopsis

geometry **ST_WKTToSQL**(text WKT);

Beskrivning

This function is the SQL/MM alias for ST_GeomFromText with no SRID argument.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.34

Se även

ST_GeomFromText

7.8.2 Välkänd binär (WKB)

7.8.2.1 ST_GeogFromWKB

ST_GeogFromWKB — Skapar en geografisk instans från en geometrisk representation av Well-Known Binary (WKB) eller utökad Well Known Binary (EWKB).

Synopsis

geography **ST_GeogFromWKB**(bytea wkb);

Beskrivning

Funktionen `ST_GeogFromWKB` tar en välkänd binär representation (WKB) av en geometri eller PostGIS Extended WKB och skapar en instans av lämplig geograftyp. Denna funktion spelar samma roll som Geometry Factory i SQL.

Om SRID inte anges är standardvärdet 4326 (WGS 84 lång lat).

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Although bytea rep contains single \, these need to be escaped when inserting into a table.

Code

```
SELECT ST_GeogFromWKB(E'\\001\\002\\000\\000\\000\\002\\000\\000\\000\\037\\205\\353Q ↩
    \\270~\\\\\\300\\323Mb\\020X\\231C@\\020X9\\264\\310~\\\\\\300)\\\\\\217\\302\\365\\230 ↩
    C@');
```

Output

```
LINESTRING(-113.98 39.198,-113.981 39.195)
```

Figure

Se även

ST_GeogFromText, ST_AsBinary

7.8.2.2 ST_GeomFromEWKB

ST_GeomFromEWKB — Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Binary representation (EWKB).

Synopsis

geometry **ST_GeomFromEWKB**(bytea EWKB);

Beskrivning

Constructs a PostGIS geometry from an Extended Well-Known Binary (EWKB) representation.

Notera
EWKB-formatet är inte en OGC-standard, utan ett PostGIS-specifikt format som inkluderar identifieraren för det spatiala referenssystemet (SRID)

Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

This example constructs `LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.259` in NAD83 longitude/latitude (SRID 4269) from its EWKB representation.

PostgreSQL escape string syntax (`E'...'`) always interprets backslash escapes. Each backslash therefore has to be doubled so that the bytea input parser receives the octal escapes.

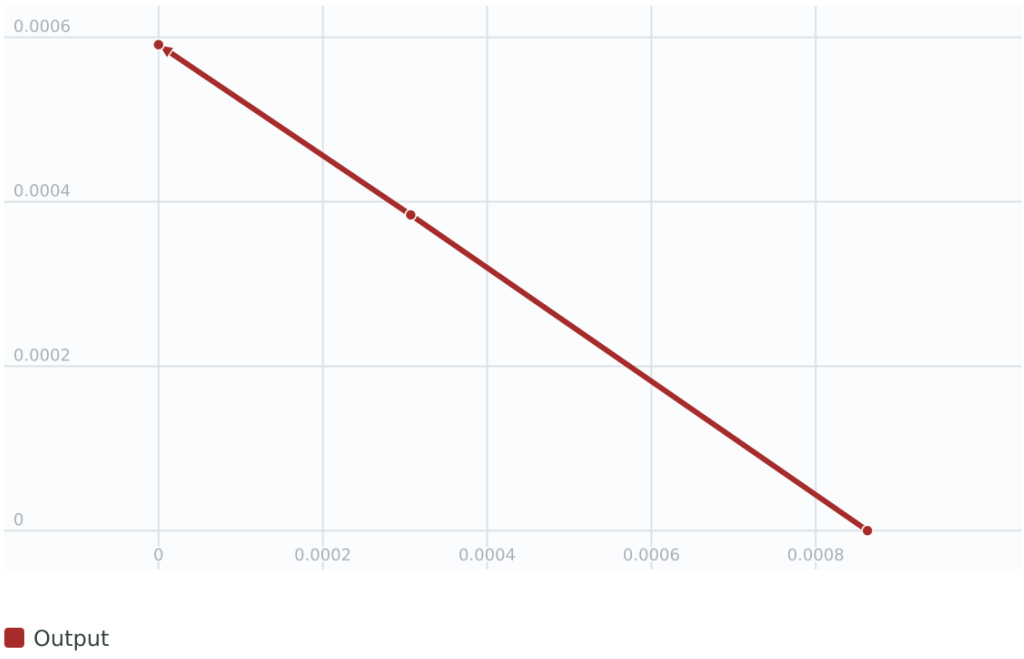
Code

```
SELECT ST_GeomFromEWKB((
    E'\\001\\002\\000\\000\\040\\255\\020\\000\\000\\003\\000\\000\\000\\344\\112\\075' ||
    E'\\013\\102\\312\\121\\300\\156\\303\\050\\010\\036\\041\\105\\100\\047\\277\\105' ||
    E'\\047\\113\\312\\121\\300\\366\\173\\142\\235\\052\\041\\105\\100\\225\\174\\354' ||
    E'\\056\\120\\312\\121\\300\\160\\231\\323\\145\\061\\041\\105\\100'
)::bytea);
```

Output

```
0102000020 ↩
    AD10000003000000E44A3D0B42CA51C06EC328081E21454027BF45274BCA51C0F67B629D2A214540957CEC2E50CA51C07099D36531214540
```

Figure



With `standard_conforming_strings` enabled (the default), ordinary string literals preserve backslashes. The same EWKB can therefore be written with a single backslash before each octal byte value.

Code

```
SELECT ST_GeomFromEWKB((
    '\001\002\000\000\040\255\020\000\000\003\000\000\000\344\112\075' ||
    '\013\102\312\121\300\156\303\050\010\036\041\105\100\047\277\105' ||
    '\047\113\312\121\300\366\173\142\235\052\041\105\100\225\174\354' ||
    '\056\120\312\121\300\160\231\323\145\061\041\105\100'
)::bytea);
```

Output

```
0102000020 ↩
    AD10000003000000E44A3D0B42CA51C06EC328081E21454027BF45274BCA51C0F67B629D2A214540957CEC2E50CA51C0
```

Se även

ST_AsBinary, ST_AsEWKB, ST_GeomFromWKB

7.8.2.3 ST_GeomFromWKB

ST_GeomFromWKB — Skapar en geometriinstans från en Well-Known Binary geometrirepresentation (WKB) och valfri SRID.

Synopsis

geometry **ST_GeomFromWKB**(bytea geom);
geometry **ST_GeomFromWKB**(bytea geom, integer srid);

Beskrivning

Funktionen `ST_GeomFromWKB` tar en välkänd binär representation av en geometri och ett`SRID`(Spatial Reference System ID) och skapar en instans av lämplig geometrityp. Denna funktion spelar samma roll som Geometry Factory i SQL. Detta är ett alternativt namn för ST_WKBToSQL.

Om SRID inte anges är standardvärdet 0 (Okänd).

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.7.2 - den valfria SRID:en är från överensstämmelsesviten

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.41

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Although bytea rep contains single \, these need to be escaped when inserting into a table. Unless standard_conforming_strings is set to on.

Code

```
SELECT ST_GeomFromWKB(E'\\001\\002\\000\\000\\000\\002\\000\\000\\000\\037\\205\\353Q ↩
    \\270~\\\\\\300\\323Mb\\020X\\231C@\\020X9\\264\\310~\\\\\\300)\\\\\\217\\302\\365\\230 ↩
    C@', 4326);
```

Output

```
SRID=4326;LINESTRING(-113.98 39.198,-113.981 39.195)
```

Figure

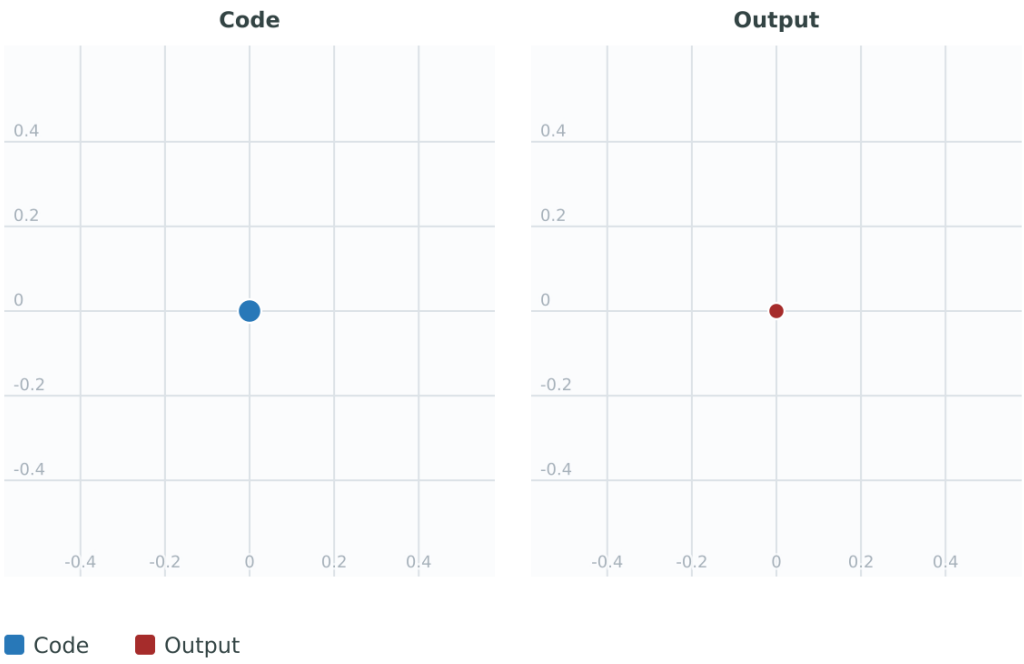
Code

```
SELECT ST_GeomFromWKB(ST_AsEWKB('POINT(2 5)'::geometry)
  );
```

Output

```
POINT(2 5)
```

Figure



The PostgreSQL `decode` function converts a hexadecimal WKB string to the bytea value accepted by `ST_GeomFromWKB`.

Code

```
SELECT ST_AsEWKT(ST_GeomFromWKB(
    decode('0101000000e5d022dbf93e2e40dbf97e6abc743540', 'hex'),
    4326
)) AS geom;
```

Output

```
SRID=4326;POINT(15.123 21.456)
```

Se även

ST_WKBToSQL, ST_AsBinary, ST_AsEWKT, ST_GeomFromEWKB

7.8.2.4 ST_LineFromWKB

ST_LineFromWKB — Creates a `LINESTRING` from a WKB representation, optionally using a given SRID.

Synopsis

geometry **ST_LineFromWKB**(bytea WKB);
geometry **ST_LineFromWKB**(bytea WKB, integer srid);

Beskrivning

The `ST_LineFromWKB` function takes a Well-Known Binary (WKB) representation of a geometry and an optional Spatial Reference System ID (`SRID`), and creates a `LINESTRING` geometry. It is a type-specific geometry factory in SQL.

If an SRID is not specified, it defaults to 0. `NULL` is returned if the input `bytea` does not represent a `LINESTRING` geometry.

Notera
OGC SPEC 3.2.6.2 - valfri SRID är från överensstämmelsesviten.

Notera
If you know all your geometries are `LINESTRING`s, it is more efficient to use ST_GeomFromWKB directly. This function calls ST_GeomFromWKB and adds validation that the result is a `LINESTRING`.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.2.9

Exempel

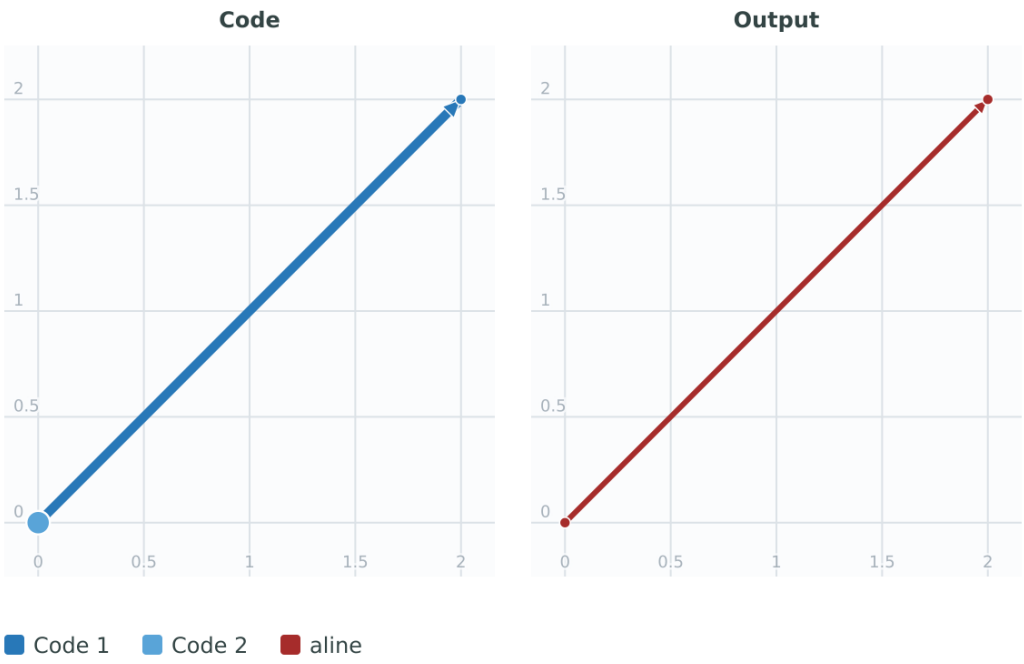
Code

```
SELECT ST_LineFromWKB(
    ST_AsBinary(ST_GeomFromText('LINESTRING(1 2,3 4)'))
) AS aline,
ST_LineFromWKB(
    ST_AsBinary(ST_GeomFromText('POINT(1 2)'))
) IS NULL AS null_return;
```

Output

```
aline                | null_return
-------------------------------------
LINESTRING(1 2,3 4)  | t
```

Figure



Se även

ST_GeomFromWKB, ST_LinestringFromWKB

7.8.2.5 ST_LinestringFromWKB

ST_LinestringFromWKB — Creates a `LINESTRING` from a WKB representation, optionally using a given SRID.

Synopsis

geometry **ST_LinestringFromWKB**(bytea WKB);
geometry **ST_LinestringFromWKB**(bytea WKB, integer srid);

Beskrivning

The `ST_LinestringFromWKB` function takes a Well-Known Binary (WKB) representation of a geometry and an optional Spatial Reference System ID (`SRID`), and creates a `LINESTRING` geometry. It is an alias for ST_LineFromWKB.

If an SRID is not specified, it defaults to 0. `NULL` is returned if the input `bytea` does not represent a `LINESTRING` geometry.

Notera
OGC SPEC 3.2.6.2 - valfri SRID är från överensstämmelsesviten.

Notera
If you know all your geometries are LINESTRINGs, it is more efficient to use ST_GeomFromWKB directly. This function calls ST_GeomFromWKB and adds validation that the result is a LINESTRING.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.6.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.2.9

Exempel

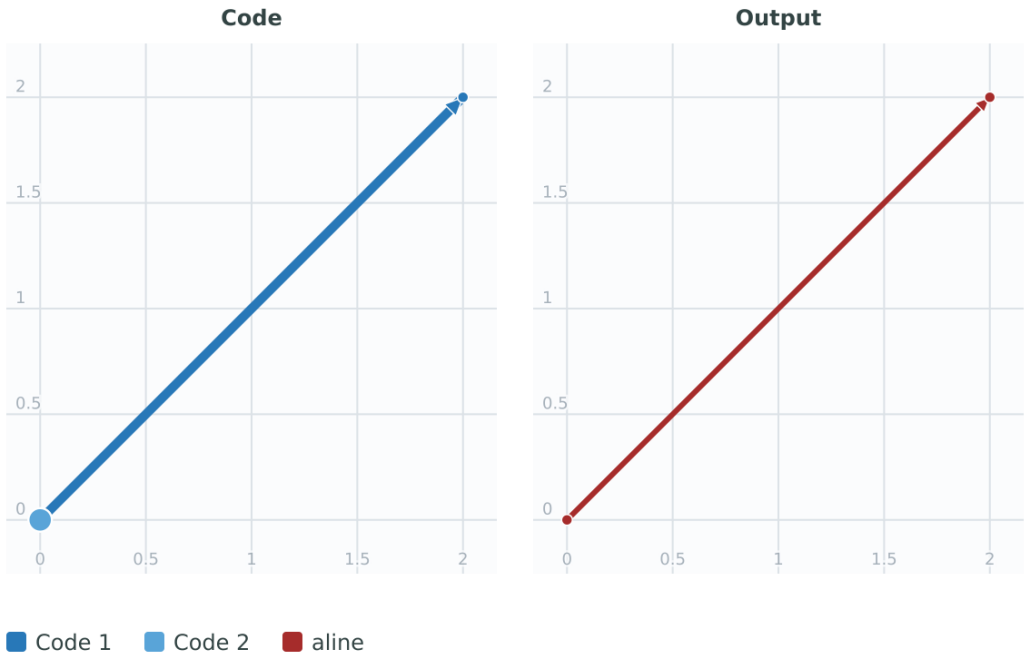
Code

```
SELECT ST_LinestringFromWKB(
    ST_AsBinary(ST_GeomFromText('LINESTRING(1 2,3 4)'))
) AS aline,
ST_LinestringFromWKB(
    ST_AsBinary(ST_GeomFromText('POINT(1 2)'))
) IS NULL AS null_return;
```

Output

```
aline                | null_return
-------------------------------------
LINESTRING(1 2,3 4)  | t
```

Figure



Se även

ST_GeomFromWKB, ST_LineFromWKB

7.8.2.6 ST_PointFromWKB

ST_PointFromWKB — Skapar en geometri från WKB med den angivna SRID

Synopsis

geometry **ST_GeomFromWKB**(bytea geom);
geometry **ST_GeomFromWKB**(bytea geom, integer srid);

Beskrivning

Funktionen `ST_PointFromWKB` tar en välkänd binär representation av geometri och ett`SRID`(Spatial Reference System ID) och skapar en instans av lämplig geometrityp - i detta fall en `POINT-geometri`. Denna funktion har samma roll som Geometry Factory i SQL.

Om en SRID inte anges är standardvärdet 0. `NULL` returneras om `indatabytea` inte representerar en `POINT-geometri`.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.7.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 6.1.9

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

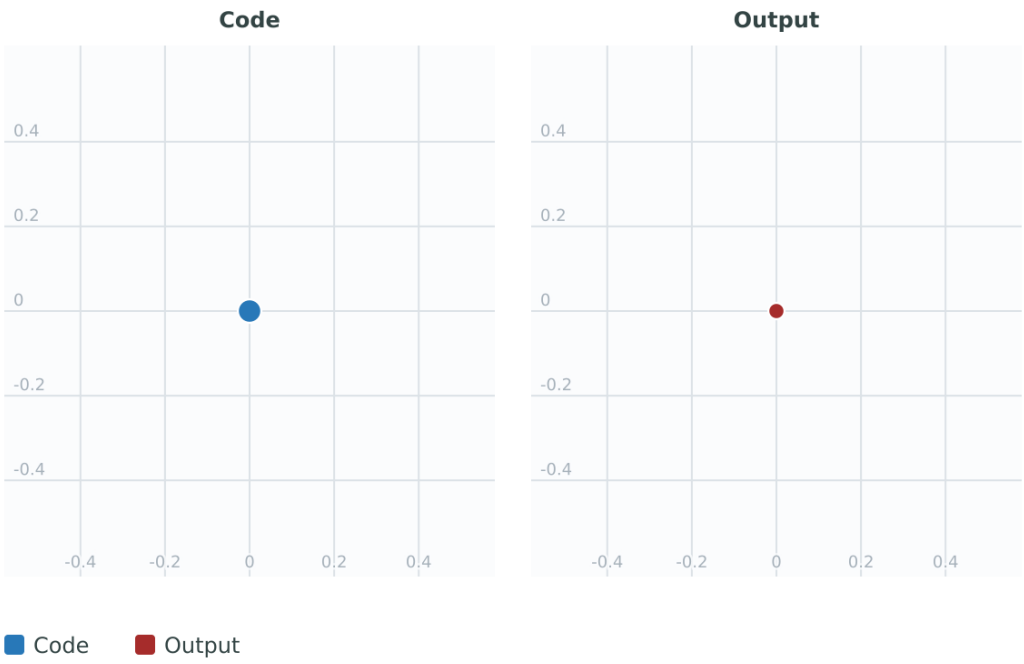
Code

```
SELECT ST_PointFromWKB(ST_AsEWKB('POINT(2 5)'::geometry));
```

Output

```
POINT(2 5)
```

Figure



Code

```
SELECT ST_PointFromWKB(ST_AsEWKB('LINESTRING(2 5,2 6)'::geometry));
```

Output

```
null
```

Se även

ST_GeomFromWKB, ST_LineFromWKB

7.8.2.7 ST_WKBToSQL

ST_WKBToSQL — Returns a geometry from Well-Known Binary (WKB). SQL/MM alias for ST_GeomFromWKB without an SRID argument.

Synopsis

geometry **ST_WKBToSQL**(bytea WKB);

Beskrivning

This function is the SQL/MM alias for ST_GeomFromWKB with no SRID argument.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.36

Se även

ST_GeomFromWKB

7.8.3 Andra format

7.8.3.1 ST_Box2dFromGeoHash

ST_Box2dFromGeoHash — Returnerar en BOX2D från en GeoHash-sträng.

Synopsis

box2d **ST_Box2dFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

Beskrivning

Returnerar en BOX2D från en GeoHash-sträng.

Om ingen `precision` anges returnerar ST_Box2dFromGeoHash en BOX2D baserad på full precision för den inmatade GeoHash-strängen.

Om `precision` anges kommer ST_Box2dFromGeoHash att använda så många tecken från GeoHash för att skapa BOX2D. Lägre precisionsvärden resulterar i större BOX2D och större värden ökar precisionen.

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0');
```

Output

```
BOX(-115.172816 36.114646,-115.172816 36.114646)
```

Code

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 0);
```

Output

```
BOX(-180 -90,180 90)
```

Code

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 10);
```

Output

```
BOX(-115.17282128334 36.1146408319473,-115.172810554504 36.1146461963654)
```

Se även

ST_GeoHash, ST_GeomFromGeoHash, ST_PointFromGeoHash

7.8.3.2 ST_GeomFromGeoHash

ST_GeomFromGeoHash — Returnerar en geometri från en GeoHash-sträng.

Synopsis

geometry **ST_GeomFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

Beskrivning

Returnerar en geometri från en GeoHash-sträng. Geometrin kommer att vara en polygon som representerar GeoHash-gränserna.

Om ingen `precision` anges returnerar ST_GeomFromGeoHash en polygon baserad på full precision för den inmatade GeoHash-strängen.

Om `precision` anges kommer ST_GeomFromGeoHash att använda så många tecken från GeoHash för att skapa polygonen.

Tillgänglighet: 2.1.0

Exempel

The cells are returned together so their relative sizes remain visible in a shared coordinate frame.

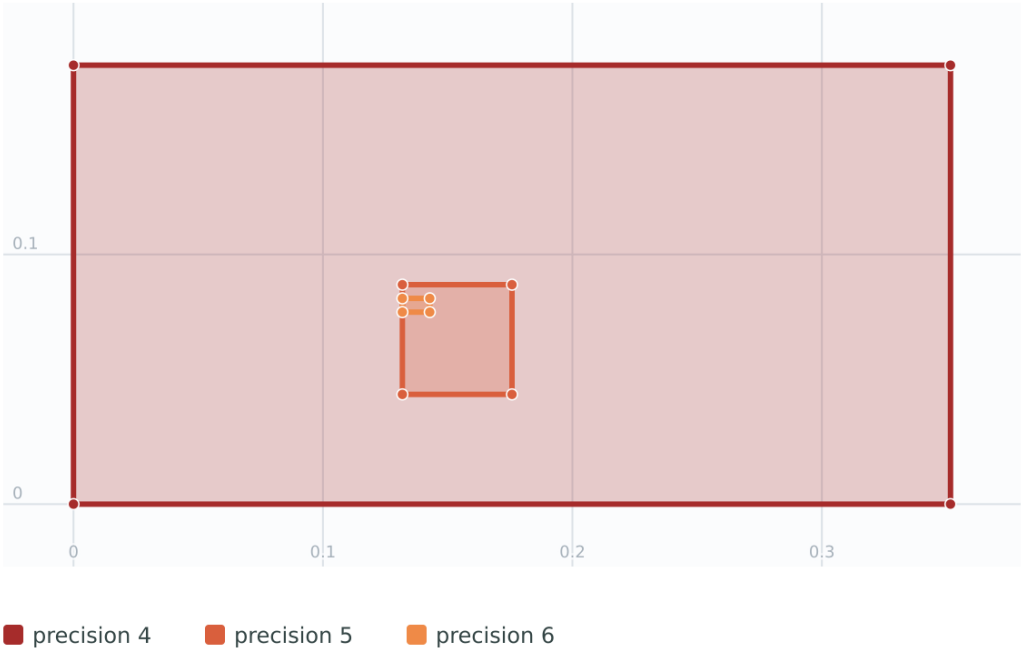
Code

```
SELECT ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 4) AS "precision 4", ST_GeomFromGeoHash ↩
    ('9qqj7nmxncgyy4d0dbxqz0', 5) AS "precision 5", ST_GeomFromGeoHash('9 ↩
    qqj7nmxncgyy4d0dbxqz0', 6) AS "precision 6";
```

Output

```
-[ RECORD 1 ]----------
precision 4 | POLYGON((-115.3125 36.03515625,-115.3125 36.2109375,-114.9609375 36.2109375,-114.9609375  ↩
     36.03515625,-115.3125 36.03515625))
precision 5 | POLYGON((-115.1806640625 36.0791015625,-115.1806640625 36.123046875,-115.13671875  ↩
     36.123046875,-115.13671875 36.0791015625,-115.1806640625 36.0791015625))
precision 6 | POLYGON((-115.1806640625 36.112060546875,-115.1806640625 36.1175537109375,-115.169677734375  ↩
     36.1175537109375,-115.169677734375 36.112060546875,-115.1806640625 36.112060546875))
```

Figure



Se även

ST_GeoHash,ST_Box2dFromGeoHash, ST_PointFromGeoHash

7.8.3.3 ST_GeomFromGML

ST_GeomFromGML — Tar som indata GML-representation av geometri och matar ut ett PostGIS-geometriobjekt

Synopsis

geometry **ST_GeomFromGML**(text geomgml);
geometry **ST_GeomFromGML**(text geomgml, integer srid);

Beskrivning

Konstruerar ett PostGIS ST_Geometry-objekt från OGC GML-representationen.

ST_GeomFromGML fungerar endast för GML Geometry-fragment. Den ger ett felmeddelande om du försöker använda den på ett helt GML-dokument.

OGC GML-versioner stöds:

- GML 3.2.1 namnrymd
- GML 3.1.1 Simple Features profile SF-2 (med GML 3.1.0 och 3.0.0 bakåtkompatibilitet)
- GML 2.1.2

OGC GML-standarder, se: http://www.opengeospatial.org/standards/gml::

Tillgänglighet: 1.5, kräver libxml2 1.6+

Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes.

Förbättrad: 2.0.0 standard srid valfri parameter tillagd.

Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

GML tillåter blandade dimensioner (t.ex. 2D och 3D inom samma MultiGeometry). Eftersom PostGIS-geometrier inte gör det konverterar ST_GeomFromGML hela geometrin till 2D om en saknad Z-dimension hittas en gång.

GML stöder blandade SRS inom samma MultiGeometry. Eftersom PostGIS-geometrier inte gör det, omprojicerar ST_GeomFromGML i detta fall alla subgeometrier till SRS-rotnoden. Om inget srsName-attribut finns tillgängligt för GML-rotnoden kastar funktionen ett fel.

ST_GeomFromGML-funktionen är inte pedantisk när det gäller en explicit GML-namnrymd. Du kan undvika att nämna det uttryckligen för vanliga användningar. Men du behöver det om du vill använda XLink-funktionen i GML.

Exempel

A single geometry with srsName.

Code

```
        SELECT ST_AsEWKT(ST_GeomFromGML($$
            <gml:LineString xmlns:gml="http://www.opengis.net/gml"
                    srsName="EPSG:4269">
                <gml:coordinates>
            -71.16028,42.258729 -71.160837,42.259112 -71.161143,42.25932
                </gml:coordinates>
            </gml:LineString>
        $$));
```

Output

```
SRID=4269;LINESTRING(-71.16028 42.258729,-71.160837 42.259112,-71.161143 42.25932)
```

Figure

XLink usage.

Code

```
        SELECT ST_AsEWKT(ST_GeomFromGML($$
            <gml:LineString xmlns:gml="http://www.opengis.net/gml"
                    xmlns:xlink="http://www.w3.org/1999/xlink"
                    srsName="EPSG:4269">
        <gml:pointProperty>
            <gml:Point gml:id="p1"
><gml:pos
>-71.16028 42.258729</gml:pos
></gml:Point>
        </gml:pointProperty>
        <gml:pos
>-71.160837 42.259112</gml:pos>
        <gml:pos
>-71.161143 42.258729</gml:pos>
                <gml:pointProperty>
```

```
                    <gml:Point xlink:type="simple" xlink:href="#p1"/>
                </gml:pointProperty>
            </gml:LineString>
        $$));
```

Output

```
SRID=4269;LINESTRING(-71.16028 42.258729,-71.160837 42.259112,-71.161143 42.258729,-71.16028 42.258729)
```

Figure

Polyhedral Surface.

Code

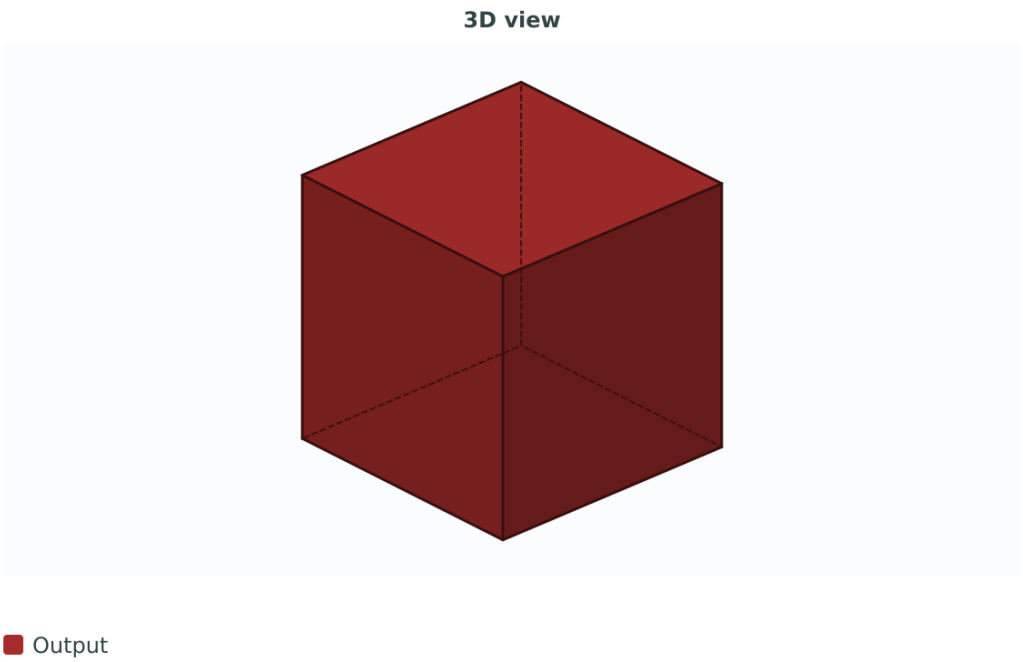
```
SELECT ST_AsEWKT(ST_GeomFromGML('
<gml:PolyhedralSurface xmlns:gml="http://www.opengis.net/gml">
<gml:polygonPatches>
  <gml:PolygonPatch>
    <gml:exterior>
      <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 0 0 1 0 1 1 0 1 0 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 0 1 0 1 1 0 1 0 0 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
```

```
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 1 0 0 1 0 1 0 0 1 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>1 1 0 1 1 1 1 0 1 1 0 0 1 1 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 1 0 0 1 1 1 1 1 1 0 0 1 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 1 1 0 1 1 1 1 0 1 1 0 0 1</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
</gml:polygonPatches>
</gml:PolyhedralSurface
>'));
```

Output

```
POLYHEDRALSURFACE(((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)))
```

Figure



Se även

Avsnitt 2.2.3, ST_AsGML, ST_GMLToSQL

7.8.3.4 ST_GeomFromGeoJSON

ST_GeomFromGeoJSON — Tar som indata en geojson-representation av en geometri och matar ut ett PostGIS-geometriobjekt

Synopsis

geometry **ST_GeomFromGeoJSON**(text geomjson);
geometry **ST_GeomFromGeoJSON**(json geomjson);
geometry **ST_GeomFromGeoJSON**(jsonb geomjson);

Beskrivning

Konstruerar ett PostGIS-geometriobjekt från GeoJSON-representationen.

ST_GeomFromGeoJSON fungerar endast för JSON Geometry-fragment. Den ger ett felmeddelande om du försöker använda den på ett helt JSON-dokument.

Enhanced: 3.0.0 parsad geometri har SRID=4326 som standard om inget annat anges.

Förbättrad: 2.5.0 kan nu acceptera json och jsonb som indata.

Tillgänglighet: 2.0.0 kräver - JSON-C >= 0.9

Notera
Om du inte har aktiverat JSON-C kommer du att få ett felmeddelande i stället för att se en utdata. Aktivera JSON-C genom att köra configure --with-jsondir=/path/to/json-c. Se Avsnitt 2.2.3 för mer information.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

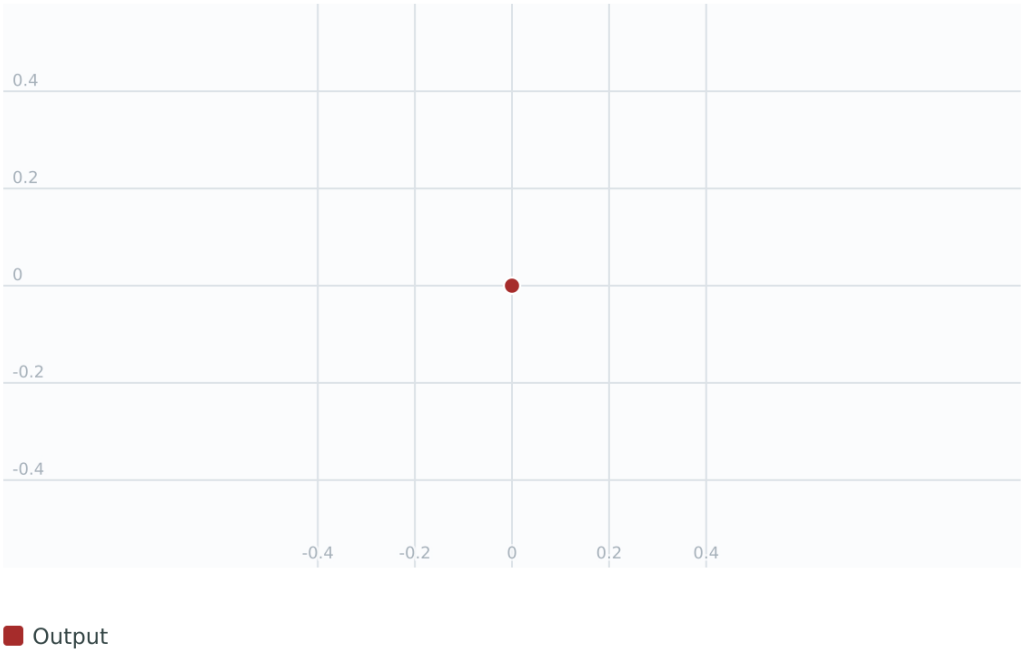
Code

```
SELECT ST_GeomFromGeoJSON('{"type":"Point","coordinates":[-48.23456,20.12345]}') AS wkt;
```

Output

```
POINT(-48.23456 20.12345)
```

Figure



This example creates a 3D LineString.

Code

```
SELECT ST_GeomFromGeoJSON('{"type":"LineString","coordinates":[[1,2,3],[4,5,6],[7,8,9]]}')
    AS wkt;
```

Output

```
LINESTRING Z (1 2 3,4 5 6,7 8 9)
```

Figure

Se även

ST_AsText, ST_AsGeoJSON, Avsnitt 2.2.3

7.8.3.5 ST_GeomFromKML

ST_GeomFromKML — Tar som indata KML-representation av geometri och matar ut ett PostGIS-geometriobjekt

Synopsis

geometry **ST_GeomFromKML**(text geomkml);

Beskrivning

Constructs a PostGIS geometry from an OGC KML representation.

ST_GeomFromKML fungerar endast för KML Geometry-fragment. Den ger ett felmeddelande om du försöker använda den på ett helt KML-dokument.

OGC KML-versioner stöds:

- KML 2.2.0 Namnrymd

OGC KML-standarder, se: http://www.opengeospatial.org/standards/kml:.

Tillgänglighet: 1.5, kräver libxml2 2.6+

Denna funktion stöder 3d och kommer inte att tappa z-index.

Notera
ST_GeomFromKML does not support SQL/MM curve geometries.

Exempel

A single LineString geometry.

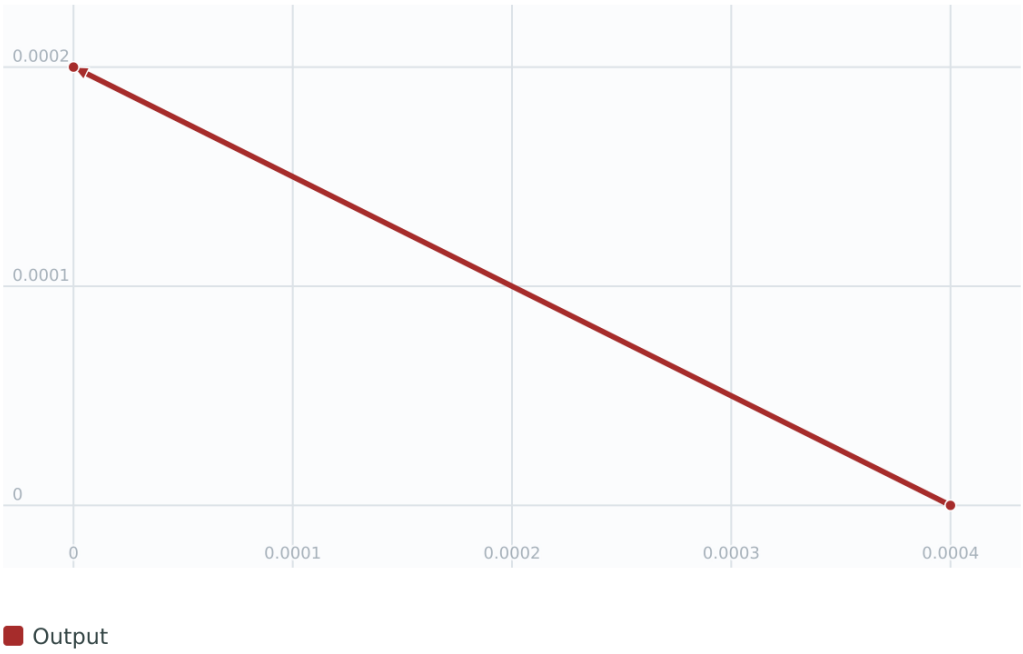
Code

```
        SELECT ST_AsEWKT(ST_GeomFromKML($$
            <LineString>
                <coordinates
>-71.1663, 42.2614
                    -71.1667, 42.2616</coordinates>
            </LineString>
        $$));
```

Output

```
SRID=4326;LINESTRING(-71.1663 42.2614,-71.1667 42.2616)
```

Figure



Se även

Avsnitt 2.2.3, ST_AsKML

7.8.3.6 ST_GeomFromTWKB

ST_GeomFromTWKB — Skapar en geometriinstans från en TWKB (Tiny Well-Known Binary")-geometrirepres

Synopsis

geometry **ST_GeomFromTWKB**(bytea twkb);

Beskrivning

Funktionen `ST_GeomFromTWKB` tar en TWKB (Tiny Well-Known Binary") geometrirepresentation (WKB) och skapar en instans av lämplig geometrityp.

Exempel

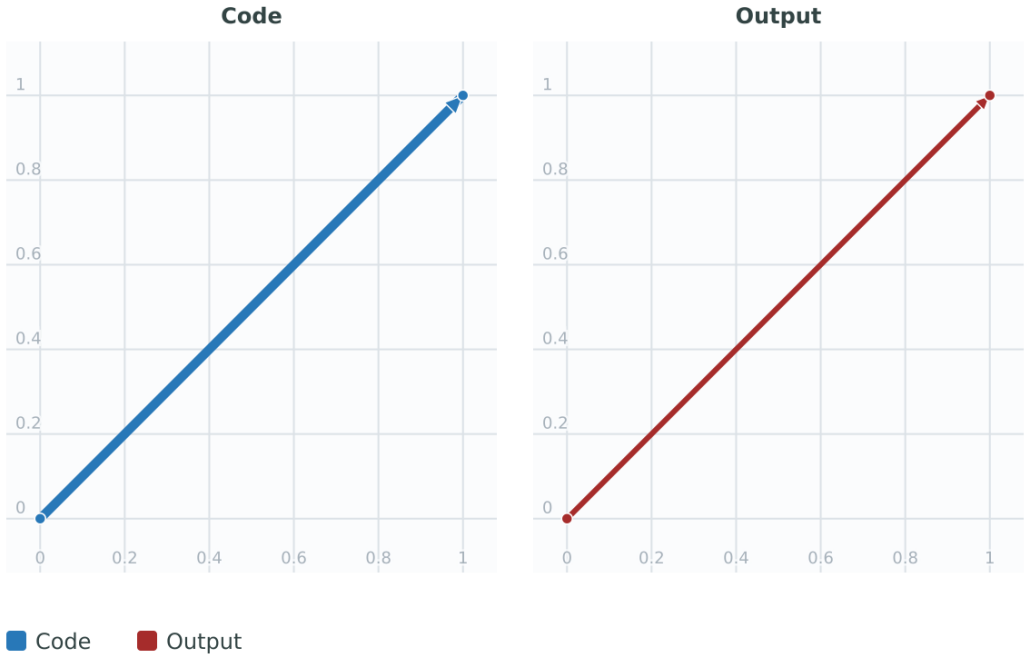
Code

```
SELECT ST_GeomFromTWKB(ST_AsTWKB('LINESTRING(126 34,127 35)'::geometry));
```

Output

```
LINESTRING(126 34, 127 35)
```

Figure



Code

```
SELECT ST_GeomFromTWKB(E'\\x620002f7f40dbce4040105');
```

Output

```
LINESTRING(-113.98 39.198,-113.981 39.195)
```

Figure

Se även

ST_AsTWKB

7.8.3.7 ST_GMLToSQL

ST_GMLToSQL — Returns a geometry from GML. Alias for ST_GeomFromGML.

Synopsis

geometry **ST_GMLToSQL**(text geomgml);
geometry **ST_GMLToSQL**(text geomgml, integer srid);

Beskrivning

This function is an alias for ST_GeomFromGML and provides the same signatures.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.50.

Tillgänglighet: 1.5, kräver libxml2 1.6+

Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes.

Förbättrad: 2.0.0 standard srid valfri parameter tillagd.

Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced.

Se även

Avsnitt 2.2.3, ST_GeomFromGML, ST_AsGML

7.8.3.8 ST_LineFromEncodedPolyline

ST_LineFromEncodedPolyline — Skapar en LineString från en kodad polylinje.

Synopsis

geometry **ST_LineFromEncodedPolyline**(text polyline, integer precision=5);

Beskrivning

Skapar en LineString från en kodad Polyline-sträng.

Valfri `precision` anger hur många decimaler som ska bevaras i Encoded Polyline. Värdet bör vara detsamma vid kodning och avkodning, annars blir koordinaterna felaktiga.

Se http://developers.google.com/maps/documentation/utilities/polylinealgorithm

Tillgänglighet: 2.2.0

Exempel

This example creates a LineString from a polyline.

Code

```
SELECT ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@');
```

Output

```
SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)
```

Figure

This example selects a different precision than the one used for polyline encoding.

Code

```
SELECT ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@', 6);
```

Output

```
SRID=4326;LINESTRING(-12.02 3.85,-12.095 4.07,-12.6453 4.3252)
```

Figure

Se även

ST_AsEncodedPolyline

7.8.3.9 ST_PointFromGeoHash

ST_PointFromGeoHash — Returnerar en punkt från en GeoHash-sträng.

Synopsis

point **ST_PointFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

Beskrivning

Returnerar en punkt från en GeoHash-sträng. Punkten representerar mittpunkten i GeoHash.

Om ingen `precision` anges returnerar ST_PointFromGeoHash en punkt baserad på full precision för den inmatade GeoHash-strängen.

Om `precision` anges kommer ST_PointFromGeoHash att använda så många tecken från GeoHash för att skapa punkten.

Tillgänglighet: 2.1.0

Exempel

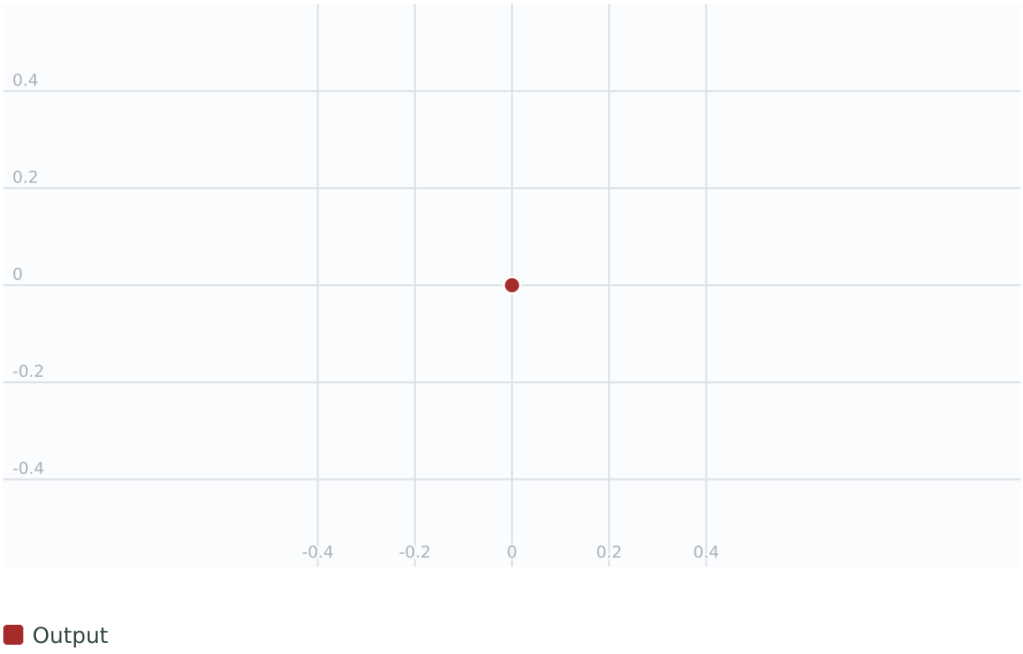
Code

```
SELECT ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0');
```

Output

```
POINT(-115.172816 36.114646)
```

Figure



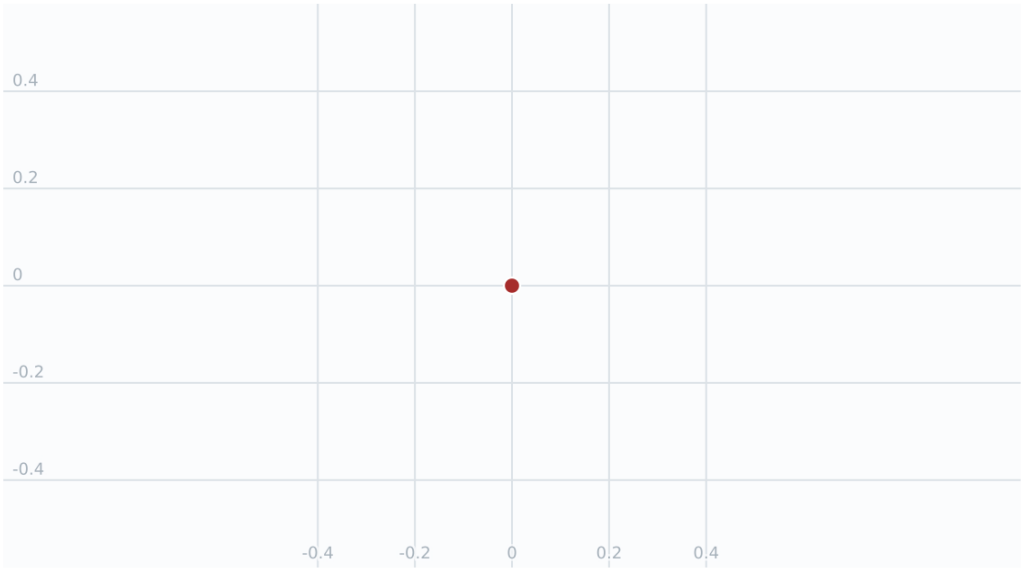
Code

```
SELECT ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 4);
```

Output

```
POINT(-115.13671875 36.123046875)
```

Figure



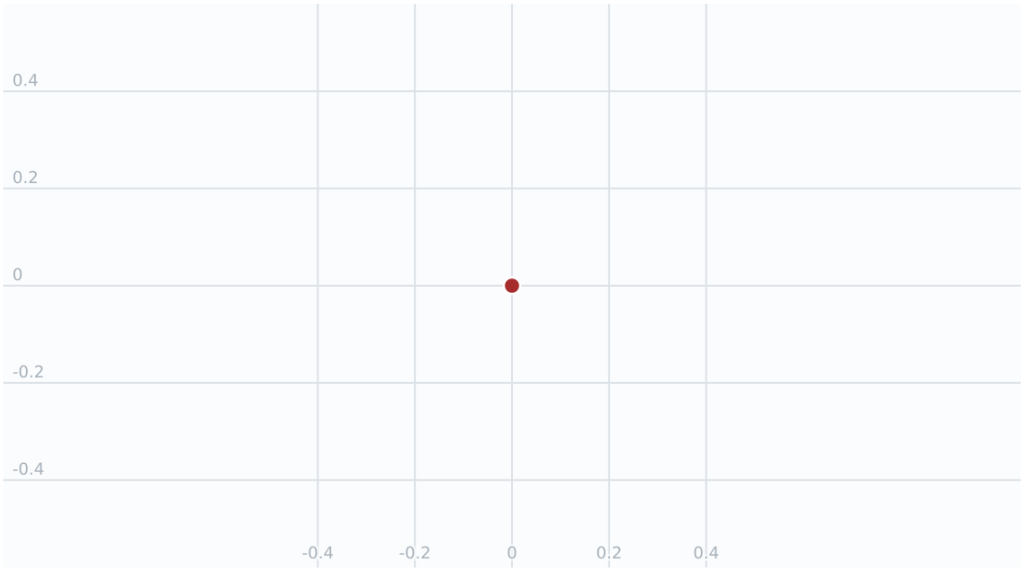
Code

```
SELECT ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 10);
```

Output

```
POINT(-115.172815918922 36.1146435141563)
```

Figure



Se även

ST_GeoHash, ST_Box2dFromGeoHash, ST_GeomFromGeoHash

7.8.3.10 ST_FromFlatGeobufToTable

ST_FromFlatGeobufToTable — Skapar en tabell baserad på strukturen i FlatGeobuf-data.

Synopsis

void **ST_FromFlatGeobufToTable**(text schemaname, text tablename, bytea FlatGeobuf input data);

Beskrivning

Skapar en tabell baserad på strukturen i FlatGeobuf-data.(http://flatgeobuf.org).
`schema` Schemanamn.
`table` Tabellnamn.
`data` Ingång FlatGeobuf data.
Tillgänglighet: 3.2.0

7.8.3.11 ST_FromFlatGeobuf

ST_FromFlatGeobuf — Läser FlatGeobuf-data.

Synopsis

setof anyelement **ST_FromFlatGeobuf**(anyelement Table reference, bytea FlatGeobuf input data);

Beskrivning

Läser FlatGeobuf-data(http://flatgeobuf.org). OBS: PostgreSQL bytea kan inte överstiga 1 GB.
`tabletype` referens till en tabelltyp.
`datainmatning` FlatGeobuf-data.
Tillgänglighet: 3.2.0

7.9 Geometriutdata

7.9.1 Well-Known Text (WKT)

7.9.1.1 ST_AsEWKT

ST_AsEWKT — Returnera WKT-representationen (Well-Known Text) av geometrin med SRID-metadata.

Synopsis

text **ST_AsEWKT**(geometry g1);
text **ST_AsEWKT**(geometry g1, integer maxdecimaldigits=15);
text **ST_AsEWKT**(geography g1);
text **ST_AsEWKT**(geography g1, integer maxdecimaldigits=15);

Beskrivning

Returnerar Well-Known Text-representationen av geometrin med SRID som prefix. Det valfria argumentet *maxdecimaldigits* kan användas för att minska det maximala antalet decimalsiffror efter flyttal som används i utdata (standard är 15).

För att utföra den omvända konverteringen av EWKT-representation till PostGIS-geometri använder du ST_GeomFromEWKT.

Varning
Om parametern *maxdecimaldigits* används kan det leda till att utdatageometrin blir ogiltig. Undvik detta genom att först använda ST_ReducePrecision med en lämplig gridstorlek.

Notera
WKT-specifikationen innehåller inte SRID. För att få OGC WKT-formatet använd ST_AsText.

Varning
WKT-formatet upprätthåller inte precisionen, så för att förhindra flytande trunkering ska du använda ST_AsBinary eller ST_AsEWKB för transport.

Förbättrad: 3.1.0 stöd för valfri precisionsparameter.

Förbättrad: 2.0.0 stöd för Geography, Polyhedral surfaces, Triangles och TIN infördes.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

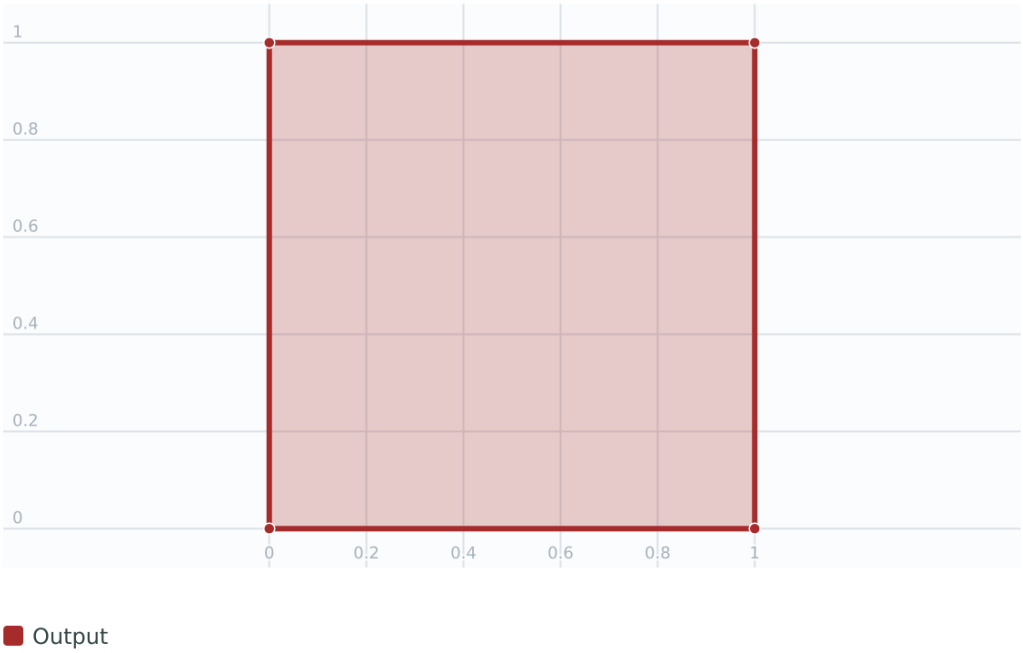
Code

```
SELECT ST_AsEWKT('0103000020 ↩
    e610000001000000050000000000000000000000000000000000000000000000000000000000000000000000000000000000f03f00000000
    '::geometry);
```

Output

```
SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
```

Figure



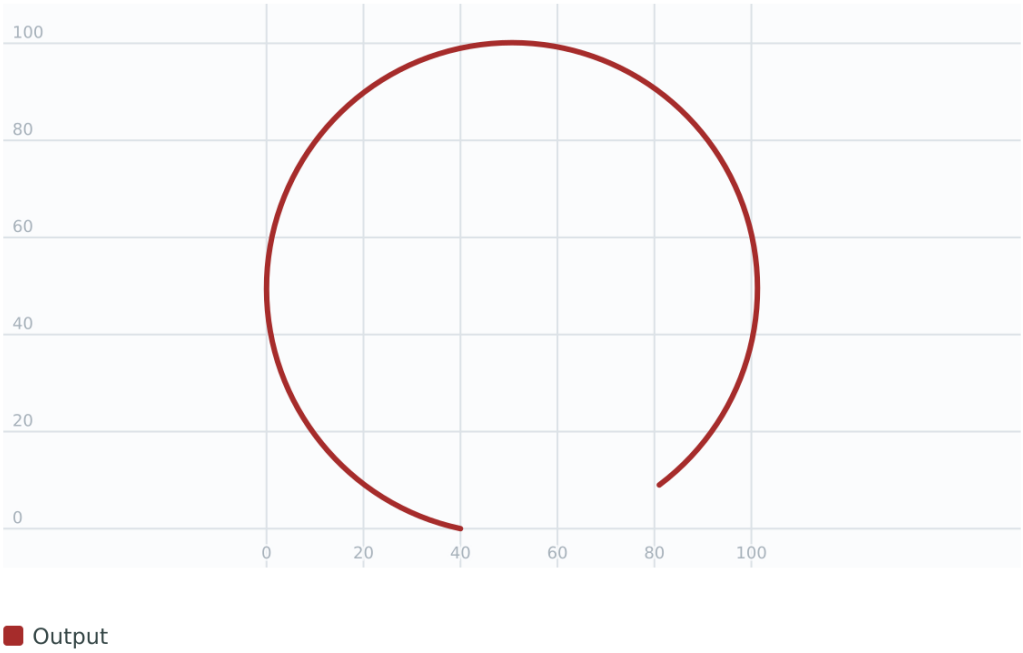
Code

```
SELECT ST_AsEWKT('0108000080030000000000000060 ↩
    E30A4100000000785C0241000000000000F03F0000000018E20A4100000000485F024100000000000000004000000000018
    ');
```

Output

```
CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)
```

Figure



Se även

ST_AsBinary, ST_AsEWKB, ST_AsText, ST_GeomFromEWKT

7.9.1.2 ST_AsText

ST_AsText — Returnera WKT-representationen (Well-Known Text) av geometrin/geografin utan SRID-metadata.

Synopsis

text **ST_AsText**(geometry g1);
text **ST_AsText**(geometry g1, integer maxdecimaldigits = 15);
text **ST_AsText**(geography g1);
text **ST_AsText**(geography g1, integer maxdecimaldigits = 15);

Beskrivning

Returnerar OGC Well-Known Text (WKT)-representationen av geometrin/geografin. Det valfria argumentet *`maxdecimaldigits`* kan användas för att begränsa antalet siffror efter decimaltecknet i utmatade ordinater (standard är 15).

För att utföra den omvända konverteringen av WKT-representation till PostGIS-geometri använder du ST_GeomFromText.

Notera
Den standardiserade OGC WKT-representationen innehåller inte SRID. Om du vill inkludera SRID som en del av utdatarepresentationen använder du den icke-standardiserade PostGIS-funktionen ST_AsEWKT

Varning
Den textuella representationen av tal i WKT kanske inte upprätthåller full flyttalsprecision. För att säkerställa full noggrannhet vid lagring eller transport av data är det bäst att använda Well-Known Binary (WKB)-format (se ST_AsBinary och *maxdecimaldigits*).

Varning
Om parametern *maxdecimaldigits* används kan det leda till att utdatageometrin blir ogiltig. Undvik detta genom att först använda ST_ReducePrecision med en lämplig gridstorlek.

Tillgänglighet: 1.5 - stöd för geografi infördes.

Förbättrad: 2.5 - valfri parameterprecision infördes.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.25

Denna metod stöder cirkulära strängar och kurvor.

Exempel

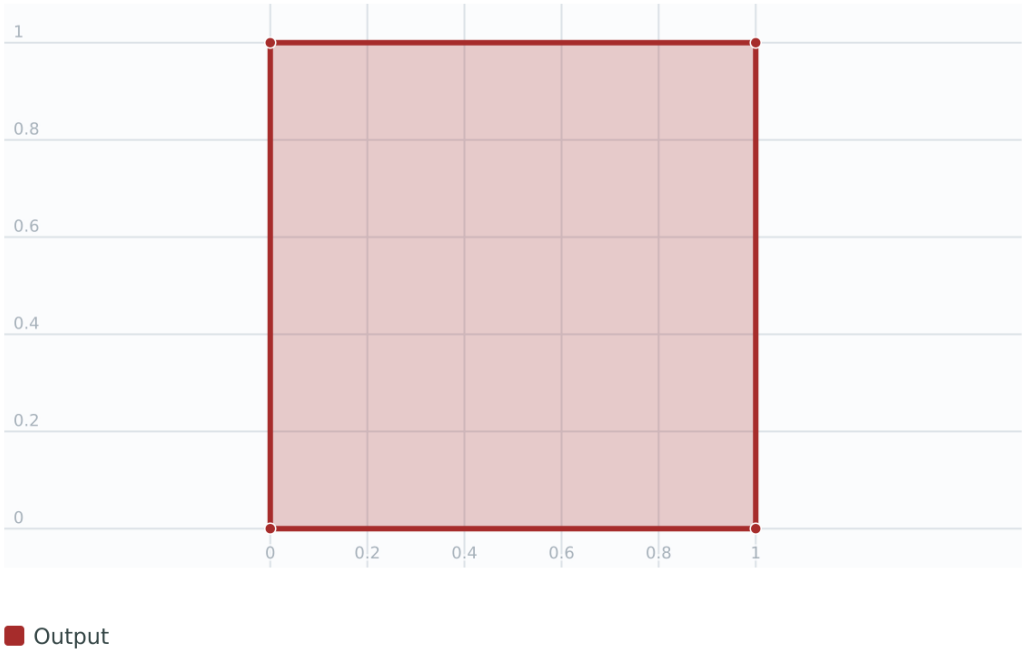
Code

```
SELECT ST_AsText(' ↩
    0103000000010000000500000000000000000000000000000000000000000000000000000000000000000000000000000000 ↩
    f03f000000000000f03f000000000000f03f000000000000f03f000000000000000000000000000000000000000000000000
    ');
```

Output

```
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

Figure



Fullständig precisionsutdata är standard.

Code

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'::geometry);
```

Output

```
POINT(111.1111111 1.1111111)
```

Argumentet *maxdecimaldigits* kan användas för att begränsa precisionen i utdata.

Code

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'::geometry, 2);
```

Output

```
POINT(111.11 1.11)
```

Se även

ST_AsBinary, ST_AsEWKB, ST_AsEWKT, ST_GeomFromText

7.9.2 Välkänd binär (WKB)

7.9.2.1 ST_AsBinary

ST_AsBinary — Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.

Synopsis

bytea **ST_AsBinary**(geometry g1);
bytea **ST_AsBinary**(geometry g1, text NDR_or_XDR);
bytea **ST_AsBinary**(geography g1);
bytea **ST_AsBinary**(geography g1, text NDR_or_XDR);

Beskrivning

Returnerar OGC/ISO Well-Known Binary (WKB)-representationen av geometrin. Den första funktionsvarianten använder som standard kodning med servermaskinens endian. Den andra funktionsvarianten tar ett textargument som anger endian-kodningen: antingen 'NDR' för little-endian eller 'XDR' för big-endian. Om du anger okända argument kommer utdata att resultera i little-endian.

WKB-formatet är användbart för att läsa geometridata från databasen och bibehålla full numerisk precision. På så sätt undviks den precisionsavrundning som kan förekomma med textformat som WKT.

För att utföra den omvända konverteringen av WKB till PostGIS-geometri använder du ST_GeomFromWKB.

Notera
OGC/ISO WKB-formatet innehåller inte SRID. För att få EWKB-formatet som innehåller SRID använder du ST_AsEWKB

Notera
PostgreSQL outputs bytea values in hex encoding by default. If your GUI tools require the old behavior, then SET `bytea_output='escape'` in your database.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Förbättrad: 2.0.0 stöd för högre koordinatdimensioner infördes.

Förbättrad: 2.0.0 stöd för att ange endian med geografi infördes.

Tillgänglighet: 1.5.0 stöd för geografi infördes.

Changed: 2.0.0 Inputs to this function can not be unknown -- must be geometry. Constructs such as `ST_AsBinary('POINT(1 2)')` are no longer valid and you will get an `n ST_AsBinary(unknown) is not unique error`. Code like that needs to be changed to `ST_AsBinary('POINT(1 2)'::geometry);`. If that is not possible, then install `legacy.sql`.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.37

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
\x0103000000010000000500000000000000000000000000000000000000000000000000000000000000000000000000000000
000000f03f000000000000f03f000000000000f03f000000000000f03f000000000000000000000000
000000000000000000000000
```

Code

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326), 'XDR');
```

Output

```
\x0000000003000000010000000500000000000000000000000000000000000000000000000000000000000000000003ff000
00000000003ff0000000000000003ff0000000000000003ff00000000000000000000000000000000000
000000000000000000000000
```

Use the PostgreSQL encode function to produce a hexadecimal WKB string without the \x prefix used for bytea display. Specifying NDR makes the byte order deterministic.

Code

```
SELECT encode(
    ST_AsBinary('POINT(15.123 21.456)'::geometry, 'NDR'),
    'hex'
) AS wkb_hex;
```

Output

```
0101000000e5d022dbf93e2e40dbf97e6abc743540
```

Se även

ST_GeomFromWKB, ST_AsEWKB, ST_AsTWKB, ST_AsText,

7.9.2.2 ST_AsEWKB

ST_AsEWKB — Returnerar EWKB-representationen (Extended Well-Known Binary) av geometrin med SRID-metadata.

Synopsis

bytea **ST_AsEWKB**(geometry g1);
bytea **ST_AsEWKB**(geometry g1, text NDR_or_XDR);

Beskrivning

Returnerar EWKB-representationen (Extended Well-Known Binary) av geometrin med SRID-metadata. Den första funktionsvarianten använder som standard kodning med servermaskinens endian. Den andra funktionsvarianten tar ett textargument som anger endian-kodningen: antingen 'NDR' för little-endian eller 'XDR' för big-endian. Om du anger okända argument kommer utdata att resultera i little-endian.

WKB-formatet är användbart för att läsa geometridata från databasen och bibehålla full numerisk precision. På så sätt undviks den precisionsavrundning som kan förekomma med textformat som WKT.

För att utföra den inversa konverteringen av EWKB till PostGIS-geometri använd ST_GeomFromEWKB.

Notera
För att få OGC/ISO WKB-formatet använd ST_AsBinary. Observera att OGC/ISO WKB-formatet inte innehåller SRID.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Code

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
\x0103000020e6100000010000000500000000000000000000000000000000000000000000000000000000000000
0000000000000f03f000000000000f03f000000000000f03f000000000000f03f00000000000000
0000000000000000000000000000000000
```

Code

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326), 'XDR');
```

Output

```
\x00200000003000010e60000000100000005000000000000000000000000000000000000000000000000000000
003ff0000000000003ff0000000000003ff0000000000003ff0000000000000000000000000000
0000000000000000000000000000000000
```

Se även

ST_AsBinary, ST_GeomFromEWKB, ST_SRID

7.9.2.3 ST_AsHEXEWKB

ST_AsHEXEWKB — Returnerar en geometri i HEXEWKB-format (som text) med antingen little-endian (NDR) eller big-endian (XDR) kodning.

Synopsis

text **ST_AsHEXEWKB**(geometry g1, text NDRorXDR);
text **ST_AsHEXEWKB**(geometry g1);

Beskrivning

Returnerar en geometri i HEXEWKB-format (som text) med antingen little-endian- (NDR) eller big-endian- (XDR) kodning. Om ingen kodning anges används NDR.

Tillgänglighet: 1.2.2

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_AsHEXEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
0103000020E6100000010000000500
0000000000000000000000000000000000000000
0000000000000000000000000000000000F03F
000000000000F03F000000000000F03F000000000000F03
F000000000000000000000000000000000000000000000000000000000
```

Which gives the same answer as:

Code

```
SELECT ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326)::text;
```

7.9.3 Andra format

7.9.3.1 ST_AsEncodedPolyline

ST_AsEncodedPolyline — Returnerar en kodad polylinje från en LineString-geometri.

Synopsis

text **ST_AsEncodedPolyline**(geometry geom, integer precision=5);

Beskrivning

Returns the geometry in the Encoded Polyline Algorithm Format. The format stores latitude and longitude deltas as integers after multiplying coordinates by 10 raised to the requested `precision`.

The informal names *polyline5* and *polyline6* refer to the same encoding with precision 5 and 6. The precision must match when encoding and decoding; otherwise the coordinates are scaled incorrectly.

The input geometry must use SRID 4326, with longitude as X and latitude as Y. Use ST_Transform to transform geometries in other coordinate systems before encoding.

The OSRM API returns precision 5 for `geometries=polyline` and precision 6 for `geometries=polyline6`. Decode the resulting `route.geometry` with ST_LineFromEncodedPolyline using the matching precision.

Tillgänglighet: 2.2.0

Exempel

Encode the same WGS 84 LineString using precision 5 and 6.

Code

```
WITH route AS (
  SELECT ST_GeomFromEWKT(
    'SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)'
  ) AS geom
)
SELECT ST_AsEncodedPolyline(geom, 5) AS polyline5,
       ST_AsEncodedPolyline(geom, 6) AS polyline6
FROM route;
```

Output

```
polyline5          |           polyline6
---------------------------+--------------------------------
 _p~iF~ps|U_ulLnnqC_mqNvxq`@ | _izlhA~rlgdF_{geC~ywl@_kwzCn`{nI
```

Decode an OSRM `route.geometry` returned with `geometries=polyline6`.

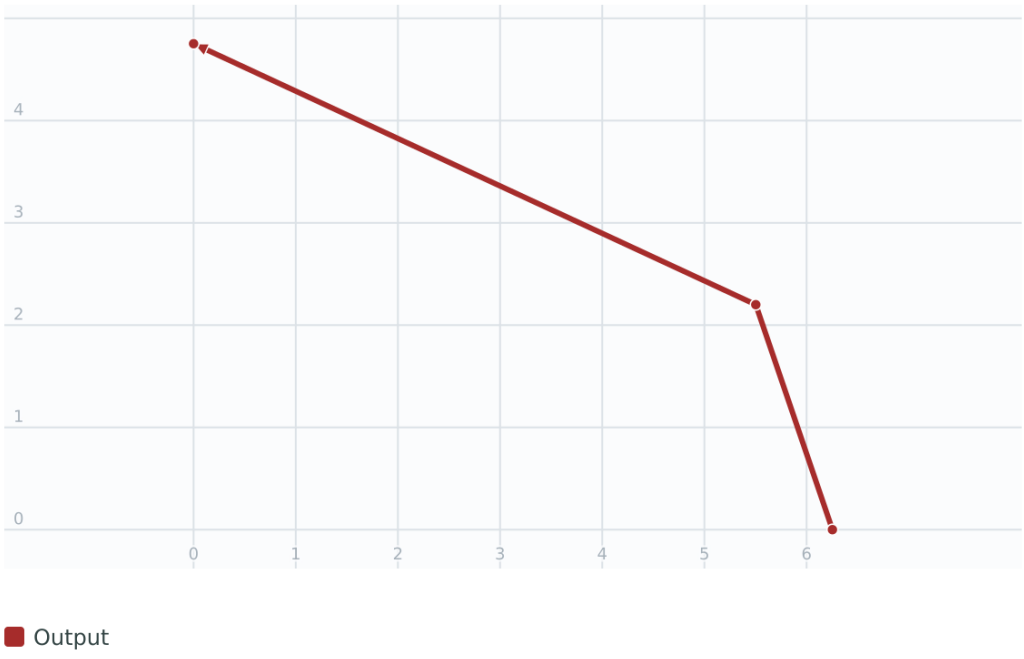
Code

```
SELECT ST_AsEWKT(ST_LineFromEncodedPolyline(
  '_izlhA~rlgdF_{geC~ywl@_kwzCn`{nI',
  6
)) AS route;
```

Output

```
SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)
```

Figure



The Google Maps JavaScript Geometry library decodes precision-5 strings. This browser example assumes that `map` is an initialized Google map.

Code

```
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&libraries=geometry"
></script>
<script>
  const encodedPath = "_p~iF~ps|U_ulLnnqC_mqNvxq`@";
  const flightPath = new google.maps.Polyline({
    path: google.maps.geometry.encoding.decodePath(encodedPath),
    map: map,
    strokeColor: "#0000CC",
    strokeOpacity: 1.0,
    strokeWeight: 4
  });
</script>
```

Se även

ST_LineFromEncodedPolyline, ST_Transform

7.9.3.2 ST_AsFlatGeobuf

ST_AsFlatGeobuf — Returnerar en FlatGeobuf-representation av en uppsättning rader.

Synopsis

bytea **ST_AsFlatGeobuf**(anyelement set row);
bytea **ST_AsFlatGeobuf**(anyelement row, bool index);
bytea **ST_AsFlatGeobuf**(anyelement row, bool index, text geom_name);

Beskrivning

Returnera en FlatGeobuf-representation(http://flatgeobuf.org) av en uppsättning rader som motsvarar en FeatureCollection. OBS: PostgreSQL bytea kan inte överstiga 1 GB.

`rad` raddata med minst en geometrikolumn.

`index` växlar spatialt indexskapande. Standard är false.

`geom_name` är namnet på geometrikolumnen i raddata. Om NULL används som standard den första geometrikolumnen som hittas.

Tillgänglighet: 3.2.0

7.9.3.3 ST_AsGeobuf

ST_AsGeobuf — Returnerar en Geobuf-representation av en uppsättning rader.

Synopsis

bytea **ST_AsGeobuf**(anyelement set row);
bytea **ST_AsGeobuf**(anyelement row, text geom_name);

Beskrivning

Returnerar en Geobuf-representation(https://github.com/mapbox/geobuf) av en uppsättning rader som motsvarar en FeatureCollection. Varje inmatad geometri analyseras för att bestämma maximal precision för optimal lagring. Observera att Geobuf i sin nuvarande form inte kan strömmas, så hela utdata kommer att samlas i minnet.

`rad` raddata med minst en geometrikolumn.

`geom_name` är namnet på geometrikolumnen i raddata. Om NULL används som standard den första geometrikolumnen som hittas.

Tillgänglighet: 2.4.0

Exempel

Code

```
SELECT encode(ST_AsGeobuf(q, 'geom'), 'base64')
FROM (SELECT 'POLYGON((0 0,0 1,1 1,1 0,0 0))'::geometry AS geom) AS q;
```

Output

```
GAAiEAoOCgwIBBoIAAAAAgIAAAE=
```

7.9.3.4 ST_AsGeoJSON

ST_AsGeoJSON — Returnerar en geometri eller funktion i GeoJSON-format.

Synopsis

text **ST_AsGeoJSON**(record feature, text geom_column=, integer maxdecimaldigits=9, boolean pretty_bool=false, text id_column='');
text **ST_AsGeoJSON**(geometry geom, integer maxdecimaldigits=9, integer options=8);
text **ST_AsGeoJSON**(geography geog, integer maxdecimaldigits=9, integer options=0);

Beskrivning

Returnerar en geometri som ett GeoJSON geometry-objekt eller en rad som ett GeoJSON feature-objekt.

De resulterande GeoJSON-geometri- och funktionsrepresentationerna överensstämmer med GeoJSON-specifikationerna RFC 7946, utom när de analyserade geometrierna refereras med en annan CRS än WGS84-longitud och latitud(EPSG:4326, urn:ogc:def:crs:OGC::CRS84); GeoJSON-geometriobjektet kommer då att ha en kort CRS SRID-identifierare bifogad som standard. både 2D- och 3D-geometrier stöds. GeoJSON stöder endast SFS 1.1 geometrityper (inget stöd för kurvor t.ex.).

Parametern `geom_column` används för att skilja mellan flera geometrikolumner. Om den utelämnas kommer den första geometrikolumnen i posten att bestämmas. Omvänt sparar man kolumntypssökningar om man skickar parametern.

Argumentet `maxdecimaldigits` kan användas för att minska det maximala antalet decimaler som används i utdata (standard är 9). Om du använder EPSG:4326 och endast matar ut geometrin för visning, kan `maxdecimaldigits=6` vara ett bra val för många kartor.

Varning
Om parametern *maxdecimaldigits* används kan det leda till att utdatageometrin blir ogiltig. Undvik detta genom att först använda ST_ReducePrecision med en lämplig gridstorlek.

Argumentet `options` kan användas för att lägga till BBOX eller CRS i GeoJSON-utdata:

- 0: innebär inget alternativ
- 1: GeoJSON BBOX
- 2: GeoJSON Short CRS (t.ex. EPSG:4326)
- 4: GeoJSON Long CRS (t.ex. urn:ogc:def:crs:EPSG::4326)
- 8: GeoJSON Short CRS om inte EPSG:4326 (standard)

Parametern `id_column` används för att ställa in medlemmen ”id”i de returnerade GeoJSON-funktionerna. Enligt GeoJSON RFC BÖR detta användas när en funktion har en allmänt använd identifierare, t.ex. en primärnyckel. Om den inte anges kommer de producerade funktionerna inte att få någon ”id-medlem och alla andra kolumner än geometrin, inklusive eventuella nycklar, kommer bara att hamna i funktionens properties-medlem.

When the input record includes duplicate column names, the resulting propertiesöbject will contain repeated keys. `ST_AsGeoJSON` raises a warning in that case because PostgreSQL jsonb keeps only the last value for duplicate keys.

The GeoJSON specification states that polygons are oriented using the Right-Hand Rule, and some clients require this orientation. This can be ensured by using ST_ForcePolygonCCW. The specification also requires that geometry be in the WGS84 coordinate system (SRID = 4326). If necessary geometry can be projected into WGS84 using ST_Transform: `ST_Transform(geom, 4326 )`.

GeoJSON kan testas och ses online på geojson.io och geojsonlint.com. Det stöds i stor utsträckning av ramverk för webbkartläggning:

- OpenLayers GeoJSON Exempel
- Exempel på GeoJSON för broschyrer
- Mapbox GL GeoJSON Exempel

Tillgänglighet: 1.3.4

Tillgänglighet: 1.5.0 stöd för geografi infördes.

Ändrad: 2.0.0 stödjer standard args och namngivna args.

Ändrat: 3.0.0 stödjer poster som indata

Ändrat: 3.0.0 mata ut SRID om inte EPSG:4326.

Ändrad: 3.5.0 gör det möjligt att ange kolumnen som innehåller funktionens id

Changed: 3.7.0 added warning about duplicate keys

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Generera en FeatureCollection:

Code

```
SELECT json_build_object(
    'type', 'FeatureCollection',
    'features', json_agg(ST_AsGeoJSON(t.*, id_column =
> 'id')::json)
    )
FROM ( VALUES (1, 'one', 'POINT(1 1)'::geometry),
              (2, 'two', 'POINT(2 2)'),
              (3, 'three', 'POINT(3 3)')
     ) as t(id, name, geom);
```

Output

```
{"type" : "FeatureCollection", "features" : [{"type": "Feature", "geometry": {"type":"Point","coordinates":[1,1]}, "id ↩
     ": 1, "properties": {"name": "one"}}, {"type": "Feature", "geometry": {"type":"Point","coordinates":[2,2]}, "id": ↩
      2, "properties": {"name": "two"}}, {"type": "Feature", "geometry": {"type":"Point","coordinates":[3,3]}, "id":  ↩
     3, "properties": {"name": "three"}}]}
```

Skapa en funktion:

Code

```
SELECT ST_AsGeoJSON(t.*, id_column =
> 'id')
FROM (VALUES (1, 'one', 'POINT(1 1)'::geometry)) AS t(id, name, geom);
```

Output

```
{"type": "Feature", "geometry": {"type":"Point","coordinates":[1,1]}, "id": 1, "properties": {"name": "one"}}
```

Glöm inte att omvandla dina data till WGS84-longitud, latitud för att överensstämma med GeoJSON-specifikationen:

Code

```
SELECT ST_AsGeoJSON(ST_Transform(geom, 4326)) FROM fe_edges LIMIT 1;
```

Output

```
{"type":"MultiLineString","coordinates":[[[-89.734634999999997,31.492072000000000],
[-89.734955999999997,31.492237999999997]]]}
```

stöd för 3D-geometrier:

Code

```
SELECT ST_AsGeoJSON('LINESTRING(1 2 3,4 5 6)');
```

Output

```
{"type":"LineString","coordinates":[[1,2,3],[4,5,6]]}
```

Options-argumentet kan användas för att lägga till BBOX och CRS i GeoJSON-utdata:

Code

```
SELECT ST_AsGeoJSON(ST_SetSRID('POINT(1 1)'::geometry, 4326), 9, 4|1);
```

Output

```
{"type":"Point","crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:EPSG::4326"}},"bbox ↩
    ":[1.000000000,1.000000000,1.000000000,1.000000000],"coordinates":[1,1]}
```

Se även

ST_GeomFromGeoJSON, ST_ForcePolygonCCW, ST_Transform

7.9.3.5 ST_AsGML

ST_AsGML — Returnera geometrin som ett GML-element version 2 eller 3.

Synopsis

text **ST_AsGML**(geometry geom, integer maxdecimaldigits=15, integer options=0);
text **ST_AsGML**(geography geog, integer maxdecimaldigits=15, integer options=0, text nprefix=null, text id=null);
text **ST_AsGML**(integer version, geometry geom, integer maxdecimaldigits=15, integer options=0, text nprefix=null, text id=null);
text **ST_AsGML**(integer version, geography geog, integer maxdecimaldigits=15, integer options=0, text nprefix=null, text id=null);

Beskrivning

Returnerar geometrin som ett GML-element (Geography Markup Language). Versionsparametern, om den anges, kan vara antingen 2 eller 3. Om ingen versionsparameter anges antas standardvärdet vara 2. Argumentet `maxdecimaldigits` kan användas för att minska det maximala antalet decimaler som används i utdata (standard är 15).

Varning
Om parametern *`maxdecimaldigits`* används kan det leda till att utdatageometrin blir ogiltig. Undvik detta genom att först använda ST_ReducePrecision med en lämplig gridstorlek.

GML 2 hänvisar till version 2.1.2, GML 3 till version 3.1.1

Argumentet öptions”är ett bitfält. Det kan användas för att definiera CRS-utdatatyp i GML-utdata och för att deklarera data som lat/lon:

- 0: GML Short CRS (t.ex. EPSG:4326), standardvärde
- 1: GML Long CRS (t.ex. urn:ogc:def:crs:EPSG::4326)
- 2: Endast för GML 3, ta bort srsDimension-attributet från utdata.
- 4: Endast för GML 3, använd <LineString> i stället för <Curve> tagg för linjer.
- 16: Ange att data är lat/lon (t.ex. srid=4326). Standard är att anta att data är planarer. Detta alternativ är endast användbart för GML 3.1.1-utdata, relaterat till axelordning. Så om du anger det kommer koordinaterna att bytas så att ordningen blir lat lon istället för databas lon lat.
- 32: Utdata av geometrins box (kuvert).

Argumentet namespace prefix”kan användas för att ange ett eget namnrymdsprefix eller inget prefix (om det är tomt). Om null eller utelämnat används gml-prefixet

Tillgänglighet: 1.3.2

Tillgänglighet: 1.5.0 stöd för geografi infördes.

Förbättrad: Stöd för prefix 2.0.0 har införts. Alternativ 4 för GML3 infördes för att tillåta användning av LineString istället för Curve-tagg för linjer. GML3-stöd för polyedriska ytor och TINS infördes. Alternativ 32 infördes för att mata ut boxen.

Ändrad: 2.0.0 använder standardnamn för args

Förbättrad: 2.1.0 id-stöd infördes för GML 3.

Notera
Endast version 3+ av ST_AsGML stöder polyhedrala ytor och TINS.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 17.2

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Version 2.

Code

```
SELECT ST_AsGML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
<gml:Polygon srsName="EPSG:4326"
><gml:outerBoundaryIs
><gml:LinearRing
><gml:coordinates
>0,0 0,1 1,1 1,0 0,0</gml:coordinates
></gml:LinearRing
></gml:outerBoundaryIs
></gml:Polygon
>
```

Version 3.

Flip coordinates and output extended EPSG (16 | 1)

Code

```
SELECT ST_AsGML(3, ST_GeomFromText('POINT(5.234234233242 6.34534534534)', 4326), 5, 17);
```

Output

```
<gml:Point srsName="urn:ogc:def:crs:EPSG::4326"
><gml:pos srsDimension="2"
>6.34535 5.23423</gml:pos
></gml:Point
>
```

Output the envelope (32).

Code

```
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 32);
```

Output

```
<gml:Envelope srsName="EPSG:4326" srsDimension="2"
><gml:lowerCorner
>1 2</gml:lowerCorner
><gml:upperCorner
>10 20</gml:upperCorner
></gml:Envelope
>
```

Output the envelope (32), reverse (lat lon instead of lon lat) (16), long srs (1) = 32 | 16 | 1 = 49.

Code

```
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 49);
```

Output

```
<gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326" srsDimension="2"
><gml:lowerCorner
>2 1</gml:lowerCorner
><gml:upperCorner
>20 10</gml:upperCorner
></gml:Envelope
>
```

This example uses a PolyhedralSurface.

Code

```
SELECT ST_AsGML(
3,
'POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )'::geometry);
```

Output

```
 <gml:PolyhedralSurface>
<gml:polygonPatches>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 0 0 0 0 1 0 1 1 0 1 0 0 0 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 0 0 0 1 0 1 1 0 1 0 0 0 0 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 0 0 1 0 0 1 0 1 0 0 1 0 0 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>1 1 0 1 1 1 1 0 1 1 0 0 1 1 0</gml:posList>
```

```
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 1 0 0 1 1 1 1 1 1 0 0 1 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 0 1 1 0 1 1 1 1 0 1 1 0 0 1</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
</gml:polygonPatches>
</gml:PolyhedralSurface
>
```

Se även

ST_GeomFromGML

7.9.3.6 ST_AsKML

ST_AsKML — Returnera geometrin som ett KML-element.

Synopsis

text **ST_AsKML**(geometry geom, integer maxdecimaldigits=15, text nprefix=NULL);
text **ST_AsKML**(geography geog, integer maxdecimaldigits=15, text nprefix=NULL);

Beskrivning

Returnera geometrin som ett Keyhole Markup Language (KML)-element. standard maximalt antal decimaler är 15, standard namnrymd är inget prefix.

Varning
Om parametern *`maxdecimaldigits`* används kan det leda till att utdatageometrin blir ogiltig. Undvik detta genom att först använda ST_ReducePrecision med en lämplig gridstorlek.

Tillgänglighet: 1.2.2 - senare varianter som innehåller version param kom i 1.3.2

Förbättrad: 2.0.0 - Lägg till prefixnamnrymd, använd standard- och namngivna args

Ändrad: 3.0.0 - Variantsignaturen ”versioned”har tagits bort

Notera
Kräver att PostGIS är kompilerat med Proj-stöd. Använd PostGIS_Full_Version för att bekräfta att du har proj-stöd kompilerat.

Notera
AsKML-utdata fungerar inte med geometrier som inte har en SRID

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_AsKML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
<Polygon
><outerBoundaryIs
><LinearRing
><coordinates
>0,0 0,1 1,1 1,0 0,0</coordinates
></LinearRing
></outerBoundaryIs
></Polygon
>
```

This example outputs a 3D LineString.

Code

```
SELECT ST_AsKML('SRID=4326;LINESTRING(1 2 3,4 5 6)');
```

Output

```
<LineString
><coordinates
>1,2,3 4,5,6</coordinates
></LineString
>
```

Se även

ST_AsSVG, ST_AsGML

7.9.3.7 ST_AsLatLonText

ST_AsLatLonText — Returnerar grader, minuter och sekunder för den angivna punkten.

Synopsis

text **ST_AsLatLonText**(geometry pt, text format='');

Beskrivning

Returnerar grader, minuter och sekunder för punkten.

Notera
Det antas att punkten befinner sig i en lat/lon-projektion. Koordinaterna X (lon) och Y (lat) normaliseras i utdata till det normala”intervallet (-180 till +180 för lon, -90 till +90 för lat).

Parametern text är en formatsträng som innehåller formatet för den resulterande texten, på samma sätt som en datumformatsträng. Giltiga symboler är Dför grader, Mför minuter, Sför sekunder och ”Cför kardinalriktning (NSEW). DMS-tokens kan upprepas för att ange önskad bredd och precision (SSS.SSSSbetyder ”1,0023”).

M”, Söch ”C”är valfria. Om ”C”utelämnas visas grader med ett --tecken om det är söder eller väster. Om S”utelämnas visas minuter som decimaltal med så många precisionssiffror som du anger. Om Mockså utelämnas visas grader som decimaltal med så många siffrors precision som du anger.

Om formatsträngen utelämnas (eller är noll-lång) kommer ett standardformat att användas.

Tillgänglighet: 2.0

Exempel

Standardformat.

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)'));
```

Output

```
2°19'29.928"S 3°14'3.243"W
```

Tillhandahåller ett format (samma som standard).

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D°M''S.SSS"C'));
```

Output

```
2°19'29.928"S 3°14'3.243"W
```

Andra tecken än D, M, S, C och . passeras bara igenom.

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D degrees, M minutes, S seconds to ↩
    the C'));
```

Output

```
2 degrees, 19 minutes, 30 seconds to the S 3 degrees, 14 minutes, 3 seconds to the W
```

Signerade grader istället för kardinalriktningar.

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D°M''S.SSS"'));
```

Output

```
-2°19'29.928" -3°14'3.243"
```

Decimala grader.

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D.DDDD degrees C'));
```

Output

```
2.3250 degrees S 3.2342 degrees W
```

To convert parsed degree, minute, and second components to decimal degrees, apply the cardinal-direction sign explicitly.

Code

```
WITH dms(label, degrees, minutes, seconds, hemisphere) AS (
  VALUES
    ('latitude', 43::numeric, 0::numeric, 50.60::numeric, 'S'),
    ('longitude', 147::numeric, 12::numeric, 18.20::numeric, 'E')
)
SELECT label,
       round(
         (CASE WHEN hemisphere IN ('S', 'W') THEN -1 ELSE 1 END)
         * (degrees + minutes / 60 + seconds / 3600),
         9
       ) AS decimal_degrees
FROM dms;
```

Output

```
   label   | decimal_degrees
-----------+-----------------
 latitude  |   -43.014055556
 longitude |   147.205055556
```

Alltför stora värden normaliseras.

Code

```
SELECT (ST_AsLatLonText('POINT (-302.2342342 -792.32498)'));
```

Output

```
72°19'29.928"S 57°45'56.757"E
```

7.9.3.8 ST_AsMARC21

ST_AsMARC21 — Returnerar geometri som en MARC21/XML-post med ett geografiskt datafält (034).

Synopsis

text **ST_AsMARC21** (geometry geom , text format='hdddmmss');

Beskrivning

Denna funktion returnerar en MARC21/XML-post med kodade kartografiska matematiska data som representerar den begränsande boxen för en given geometri. `Formatparametern` gör det möjligt att koda koordinaterna i underfälten `$d,$e,$f` och `$g` i alla format som stöds av MARC21/XML-standarden. Giltiga format är:

- kardinalriktning, grader, minuter och sekunder (standard): `hdddmmss`
- decimalgrader med kardinalriktning: `hddd.dddddd`
- decimalgrader utan kardinalriktning: `ddd.dddddddd`
- decimalminuter med kardinalriktning: `hdddmm.mmmm`
- decimalminuter utan kardinalriktning: `dddmm.mmmm`
- decimalsekunder med kardinalriktning: `hdddmmss.sss`

Decimaltecknet kan också vara ett kommatecken, t.ex. `hdddmm,mmmm`..

Precisionen i decimalformat kan begränsas av antalet tecken efter decimaltecknet, t.ex. `hdddmm.mm` för decimala minuter med en precision på två decimaler.

Denna funktion ignorerar Z- och M-dimensionerna.

Stöd för LOC MARC21/XML-versioner:

- MARC21/XML 1.1

Tillgänglighet: 3.3.0

Notera
Denna funktion stöder inte geometrier som inte är lon/lat, eftersom de inte stöds av MARC21/XML-standarden (Coded Cartographic Mathematical Data).

Notera
MARC21/XML-standarden tillhandahåller inte något sätt att kommentera det spatiala referenssystemet för kodade kartografiska matematiska data, vilket innebär att denna information kommer att gå förlorad efter konvertering till MARC21/XML.

Exempel

Konvertera en `POINT` till MARC21/XML formaterad som hdddmmss (standard)

Code

```
SELECT ST_AsMARC21('SRID=4326;POINT(-4.504289 54.253312)'::geometry);
```

Output

```
                                st_asmarc21
--------------------------------------------------
<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>W0043015</subfield>
```

```
        <subfield code="e"
>W0043015</subfield>
        <subfield code="f"
>N0541512</subfield>
        <subfield code="g"
>N0541512</subfield>
    </datafield>
</record>
```

Konvertering av en POLYGON till MARC21/XML formaterad i decimalgrader

Code

```
SELECT ST_AsMARC21('SRID=4326;POLYGON((-4.5792388916015625 ↩
    54.18172660239091,-4.56756591796875 54.196993557130355,-4.546623229980469 ↩
    54.18313300502024,-4.5792388916015625 54.18172660239091))'::geometry,'hddd.dddd');

<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>W004.5792</subfield>
        <subfield code="e"
>W004.5466</subfield>
        <subfield code="f"
>N054.1970</subfield>
        <subfield code="g"
>N054.1817</subfield>
    </datafield>
</record>
```

Konvertering av en GEOMETRYCOLLECTION till MARC21/XML formaterad i decimalminuter. Geometrierna ordning i MARC21/XML-utdata motsvarar deras ordning i samlingen.

Code

```
SELECT ST_AsMARC21('SRID=4326;GEOMETRYCOLLECTION(POLYGON((13.1 52.65,13.516666666666667 ↩
    52.65,13.516666666666667 52.38333333333333,13.1 52.38333333333333,13.1 52.65)),POINT ↩
    (-4.5 54.25))'::geometry,'hdddmm.mmmm');
```

Output

```
                                st_asmarc21
-----------------------------------------------------
<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>E01307.0000</subfield>
        <subfield code="e"
>E01331.0000</subfield>
        <subfield code="f"
>N05240.0000</subfield>
        <subfield code="g"
>N05224.0000</subfield>
    </datafield>
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>W00430.0000</subfield>
        <subfield code="e"
>W00430.0000</subfield>
        <subfield code="f"
>N05415.0000</subfield>
```

```
        <subfield code="g"
>N05415.0000</subfield>
    </datafield>
</record>
```

Se även

ST_GeomFromMARC21

7.9.3.9 ST_AsMVTGeom

ST_AsMVTGeom — Transformerar en geometri till koordinatrymden för en MVT-platta.

Synopsis

geometry **ST_AsMVTGeom**(geometry geom, box2d bounds, integer extent=4096, integer buffer=256, boolean clip_geom=true);

Beskrivning

Transformerar en geometri till koordinatrymden för en MVT-karta(Mapbox Vector Tile) och klipper den till kartans gränser om så krävs. Geometrin måste vara i målkartans koordinatsystem (med hjälp av ST_Transform om det behövs). Vanligtvis är detta Web Mercator (SRID:3857).

Funktionen försöker bevara geometrins validitet och korrigerar den vid behov. Detta kan leda till att resultatgeometrin kollapsar till en lägre dimension.

The function performs geometry simplification appropriate to the integer tile coordinate space. After transforming to tile coordinates it snaps coordinates to the integer grid, removes repeated points, and removes points that lie on straight lines. It does not perform general-purpose, scale-dependent simplification such as ST_Simplify. To reduce detail before tile conversion, simplify the geometry in map coordinates before calling `ST_AsMVTGeom`.

De rektangulära gränserna för plattan i målkartans koordinatrum måste anges, så att geometrin kan transformeras och klippas vid behov. Gränserna kan genereras med hjälp av ST_TileEnvelope.

Denna funktion används för att konvertera geometrin till det koordinatutrymme för plattor som krävs av ST_AsMVT.

`geom` är den geometri som ska transformeras, i målkartans koordinatsystem.

`bounds` är de rektangulära gränserna för plattan i kartkoordinatrymden, utan buffert.

`extent` är storleken på plattans extent i plattans koordinatutrymme enligt definitionen i MVT-specifikationen. Standardvärdet är 4096.

`buffer` är buffertstorleken i plattkoordinatrymden för geometriklippning. Standardvärdet är 256.

`clip_geom` är en boolean som styr om geometrier ska klippas eller kodas som de är. Standardvärdet är true.

Tillgänglighet: 2.4.0

Notera
Från och med 3.0 kan Wagyu väljas vid konfigurationstillfället för att klippa och validera MVT-polygoner. Det här biblioteket är snabbare och ger mer korrekta resultat än GEOS standard, men det kan tappa små polygoner.

Exempel

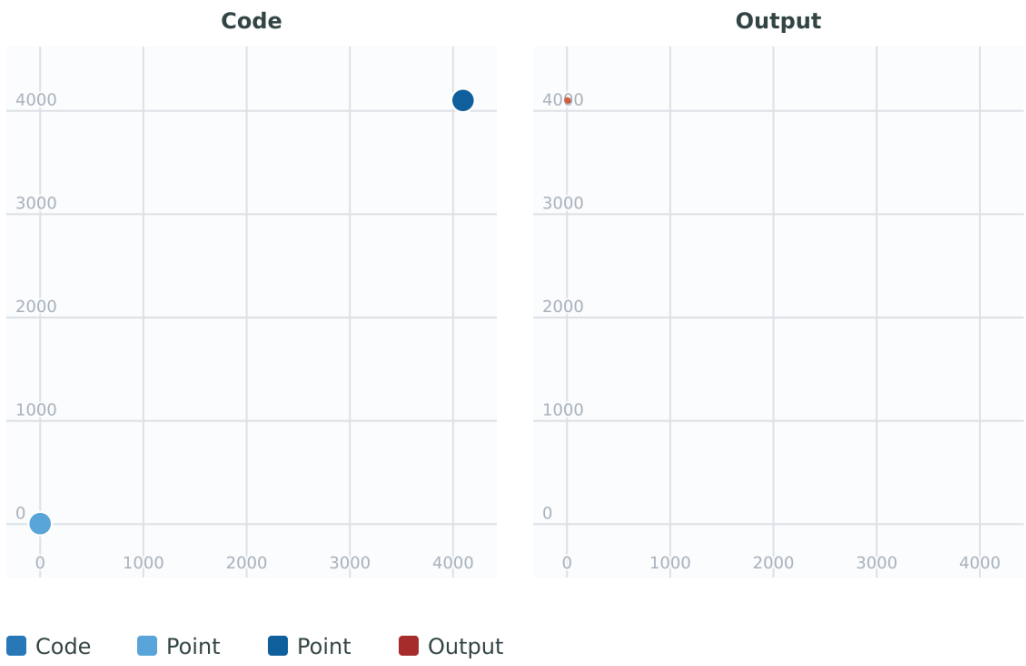
Code

```
SELECT ST_AsMVTGeom('POLYGON ((0 0,10 0,10 5,0 -5,0 0))',
    ST_MakeBox2D(ST_Point(0, 0), ST_Point(4096, 4096)),
    4096, 0, false);
```

Output

```
MULTIPOLYGON(((5 4096,10 4091,10 4096,5 4096)),((5 4096,0 4101,0 4096,5 4096)))
```

Figure



Kanoniskt exempel för en Web Mercator-platta som använder en beräknad plattgräns för att fråga och klippa geometri. Detta förutsätter att data.geom-kolumnen har srid på 4326.

Code

```
SELECT ST_AsMVTGeom(ST_Transform(geom, 3857 ),
            ST_TileEnvelope(12, 513, 412),
            extent =
> 4096,
            buffer =
> 64) AS geom
  FROM data
  WHERE geom && ST_Transform(ST_TileEnvelope(12, 513, 412, margin =
> (64.0 / 4096)), 4326)
```

Se även

ST_AsMVT, ST_TileEnvelope, PostGIS_Wagyu_Version

7.9.3.10 ST_AsMVT

ST_AsMVT — Aggregatfunktion som returnerar en MVT-representation av en uppsättning rader.

Synopsis

bytea **ST_AsMVT**(anyelement set row);
bytea **ST_AsMVT**(anyelement row, text name);
bytea **ST_AsMVT**(anyelement row, text name, integer extent);
bytea **ST_AsMVT**(anyelement row, text name, integer extent, text geom_name);
bytea **ST_AsMVT**(anyelement row, text name, integer extent, text geom_name, text feature_id_name);

Beskrivning

En aggregerad funktion som returnerar en binär Mapbox Vector Tile-representation av en uppsättning rader som motsvarar ett plattlager. Raderna måste innehålla en geometrikolumn som kommer att kodas som en funktionsgeometri. Geometrin måste vara i plattkoordinatrymden och giltig enligt MVT-specifikationen. ST_AsMVTGeom kan användas för att omvandla geometrin till plattkoordinatrymden. Övriga kolumner i raden kodas som attribut för objektet.

Mapbox Vector Tile-format kan lagra funktioner med varierande uppsättningar attribut. För att använda den här funktionen måste du ange en JSONB-kolumn i raddata som innehåller Json-objekt på en nivås djup. Nycklarna och värdena i JSONB-värdena kommer att kodas som funktionsattribut.

Plattor med flera lager kan skapas genom att sammanfoga flera anrop till denna funktion med hjälp av `||` eller `STRING_AGG`.

Viktigt
Anropa inte med en `GEOMETRYCOLLECTION` som ett element i raden. Du kan dock använda ST_AsMVTGeom för att förbereda en geometrisamling för inkludering.

`rad` raddata med minst en geometrikolumn.

`name` är namnet på lagret. Default är strängen default”.

`extent` är plattans utsträckning i skärmutrymme enligt specifikationens definition. Standardvärdet är 4096.

`geom_name` är namnet på geometrikolumnen i raddata. Standard är den första geometrikolumnen. Observera att PostgreSQL som standard automatiskt viker icke-citerade identifierare till gemener, vilket innebär att om inte geometrikolumnen är citerad, t.ex. `MyMVTGeom`”, måste denna parameter tillhandahållas som gemener.

`feature_id_name` är namnet på kolumnen Feature ID i raddata. Om NULL eller negativt anges inte Feature ID. Den första kolumnen som matchar namn och giltig typ (smallint, integer, bigint) kommer att användas som Feature ID, och alla efterföljande kolumner kommer att läggas till som en egenskap. JSON-egenskaper stöds inte.

Förbättrad: 3.0 - stöd för Feature ID har lagts till.

Förbättrad: 2.5.0 - stöd för parallella frågor har lagts till.

Tillgänglighet: 2.4.0

Exempel

Code

```
WITH mvtgeom AS
(
  SELECT ST_AsMVTGeom(geom, ST_TileEnvelope(12, 513, 412), extent =
> 4096, buffer =
```

```
> 64) AS geom, name, description
  FROM points_of_interest
  WHERE geom && ST_TileEnvelope(12, 513, 412, margin =
> (64.0 / 4096))
)
SELECT ST_AsMVT(mvtgeom.*)
FROM mvtgeom;
```

Se även

ST_AsMVTGeom, ST_TileEnvelope

7.9.3.11 ST_AsSVG

ST_AsSVG — Returnerar SVG-banedata för en geometri.

Synopsis

text **ST_AsSVG**(geometry geom, integer rel=0, integer maxdecimaldigits=15);
text **ST_AsSVG**(geography geog, integer rel=0, integer maxdecimaldigits=15);

Beskrivning

Returnerar geometrin som SVG-banedata (Scalar Vector Graphics). Använd 1 som andra argument om du vill att sökvägsdata ska implementeras i form av relativa förflyttningar, standard (eller 0) använder absoluta förflyttningar. Det tredje argumentet kan användas för att minska det maximala antalet decimaler som används i utdata (standard är 15). Punktgeometrier kommer att återges som cx/cy när 'rel'-argumentet är 0, x/y när 'rel' är 1. Multipoint-geometrier avgränsas med kommatecken (","), GeometryCollection-geometrier avgränsas med semikolon (";").

För att arbeta med PostGIS SVG-grafik, kolla in biblioteket pg_svg som innehåller plpgsql-funktioner för att arbeta med utdata från ST_AsSVG.

Förbättrad: 3.4.0 för stöd för alla kurvtyper

Ändrad: 2.0.0 för att använda standardargs och stödja namngivna args

Tillgänglighet: 1.2.2. Tillgänglighet: 1.4.0 Ändrat i PostGIS 1.4.0 för att inkludera L-kommandot i absolut sökväg för att överensstämma med http://www.w3.org/TR/SVG/paths.html#PathDataBNF

Denna metod stöder cirkulära strängar och kurvor.

Exempel

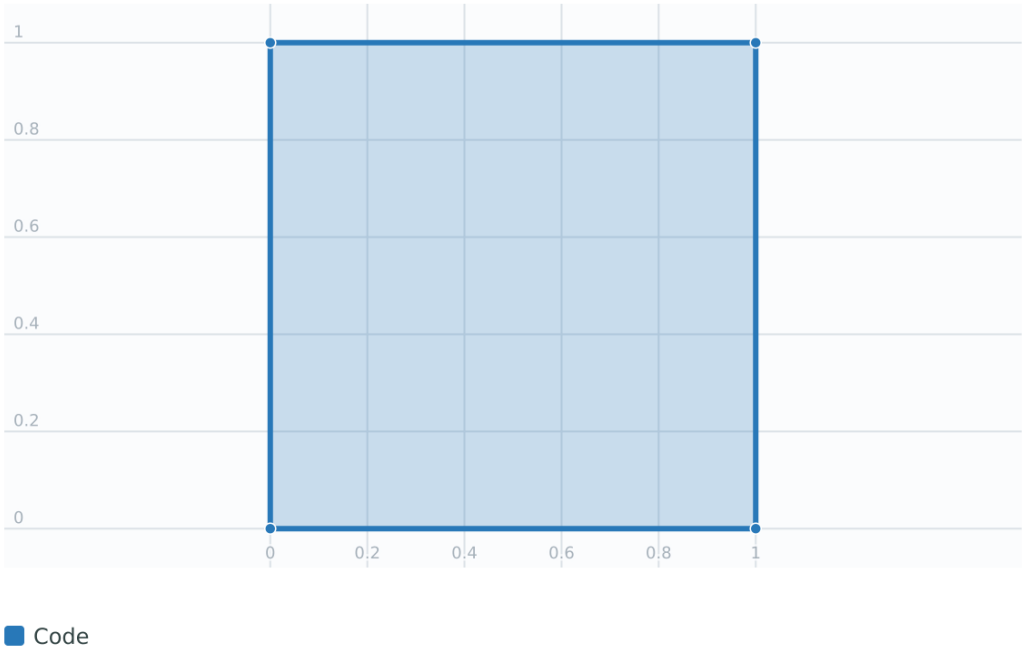
Code

```
SELECT ST_AsSVG('POLYGON((0 0,0 1,1 1,1 0,0 0))'::geometry);
```

Output

```
M 0 0 L 0 -1 1 -1 1 0 Z
```

Figure



Cirkulär sträng

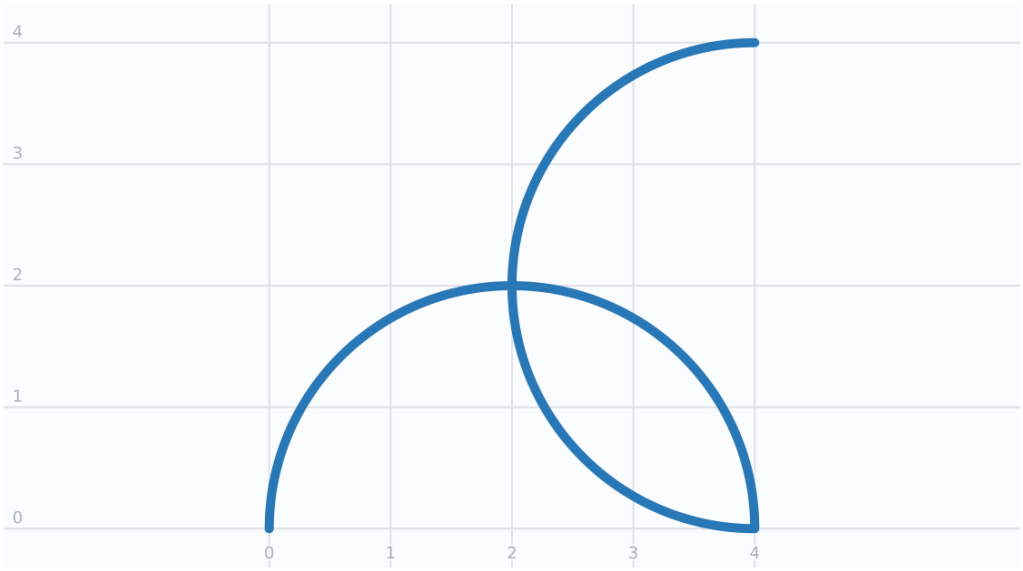
Code

```
SELECT ST_AsSVG('CIRCULARSTRING(-2 0,0 2,2 0,0 2,2 4)'::geometry);
```

Output

```
M -2 0 A 2 2 0 0 1 2 0 A 2 2 0 0 1 2 -4
```

Figure



Multi-kurva

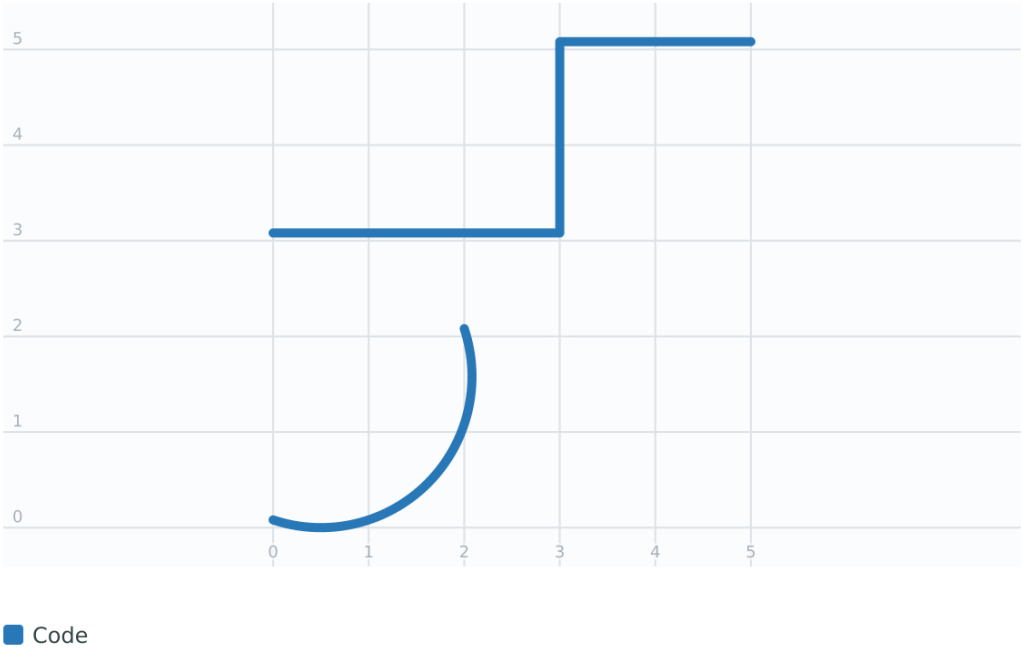
Code

```
SELECT ST_AsSVG('MULTICURVE((5 5,3 5,3 3,0 3),
CIRCULARSTRING(0 0,2 1,2 2))'::geometry, 0, 0);
```

Output

```
M 5 -5 L 3 -5 3 -3 0 -3 M 0 0 A 2 2 0 0 0 2 -2
```

Figure



Flera ytor

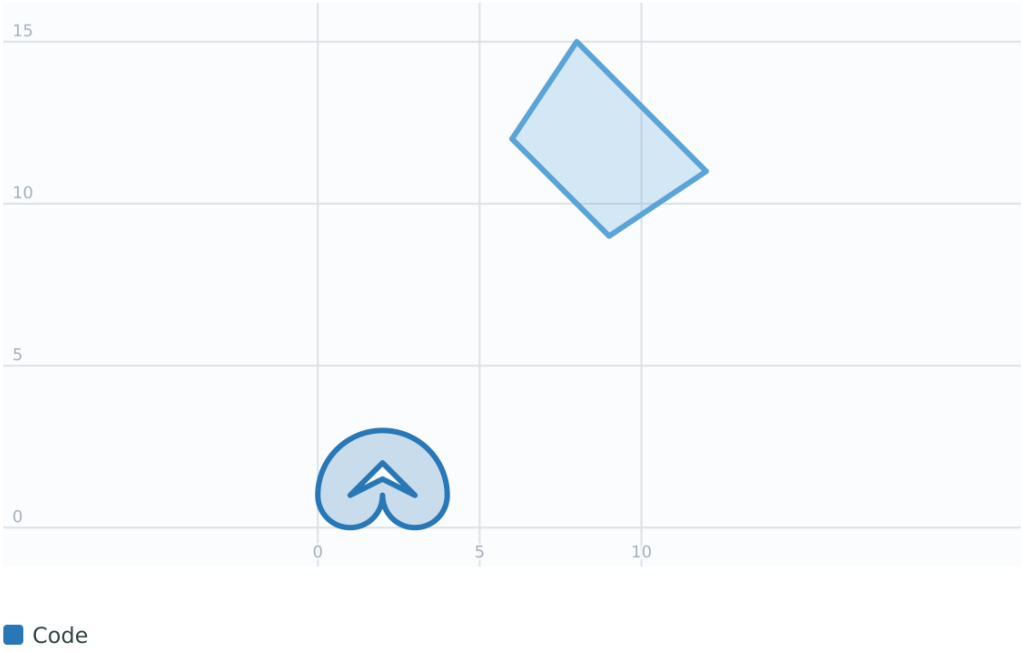
Code

```
SELECT ST_AsSVG('MULTISURFACE(
CURVEPOLYGON(CIRCULARSTRING(-2 0,-1 -1,0 0,1 -1,2 0,0 2,-2 0),
    (-1 0,0 0.5,1 0,0 1,-1 0)),
((7 8,10 10,6 14,4 11,7 8)))'::geometry, 0, 2);
```

Output

```
M -2 0 A 1 1 0 0 0 0 0 A 1 1 0 0 0 2 0 A 2 2 0 0 0 -2 0 Z
M -1 0 L 0 -0.5 1 0 0 -1 -1 0 Z
M 7 -8 L 10 -10 6 -14 4 -11 Z
```

Figure



7.9.3.12 ST_AsTWKB

ST_AsTWKB — Returnerar geometrin som TWKB, aka Tiny Well-Known Binary”

Synopsis

bytea **ST_AsTWKB**(geometry geom, integer prec=0, integer prec_z=0, integer prec_m=0, boolean with_sizes=false, boolean with_boxes=false);
bytea **ST_AsTWKB**(geometry[] geom, bigint[] ids, integer prec=0, integer prec_z=0, integer prec_m=0, boolean with_sizes=false, boolean with_boxes=false);

Beskrivning

Returnerar geometrin i TWKB-format (Tiny Well-Known Binary). TWKB är ett komprimerat binärt format med fokus på att minimera storleken på utdata.

Parametrarna för decimalsiffror styr hur mycket precision som lagras i utdata. Som standard avrundas värden till närmaste enhet före kodning. Om du vill överföra mer precision ökar du antalet. Ett värde på 1 innebär t.ex. att den första siffran till höger om decimaltecknet bevaras.

Parametrarna sizes och bounding boxes styr om valfri information om objektets kodade längd och objektets gränser ska inkluderas i utdata. Som standard är de inte det. Aktivera dem inte om inte din klientprogramvara har användning för dem, eftersom de bara tar upp utrymme (och att spara utrymme är själva poängen med TWKB).

Funktionens array-input-form används för att konvertera en samling geometrier och unika identifierare till en TWKB-samling som bevarar identifierarna. Detta är användbart för klienter som förväntar sig att packa upp en samling och sedan få tillgång till ytterligare information om objekten inuti. Du kan skapa matriserna med hjälp av funktionen array_agg. De andra parametrarna fungerar på samma sätt som för den enkla formen av funktionen.

Notera
Formatspecifikationen finns tillgänglig online på https://github.com/TWKB/Specification, och kod för att bygga en JavaScript-klient finns på https://github.com/TWKB/twkb.js..

Förbättrad: 2.4.0 minnes- och hastighetsförbättringar.

Tillgänglighet: 2.2.0

Exempel

Code

```
SELECT ST_AsTWKB('LINESTRING(1 1,5 5)'::geometry);
```

Output

```
\x02000202020808
```

För att skapa ett aggregerat TWKB-objekt inklusive identifierare aggregerar du först de önskade geometrierna och objekten med hjälp av ärray_agg()och anropar sedan lämplig TWKB-funktion.

Code

```
SELECT ST_AsTWKB(array_agg(geom), array_agg(gid)) FROM mytable;
```

Output

```
\x040402020400000202
```

Se även

ST_GeomFromTWKB, ST_AsBinary, ST_AsEWKB, ST_AsEWKT, ST_GeomFromText

7.9.3.13 ST_AsX3D

ST_AsX3D — Returnerar en geometri i X3D xml-nodelementformat: ISO-IEC-19776-1.2-X3DEncodings-XML

Synopsis

text **ST_AsX3D**(geometry g1, integer maxdecimaldigits=15, integer options=0);

Beskrivning

Returnerar en geometri som ett X3D xml-formaterat nodelement http://www.web3d.org/standards/-number/19776-1. Om `maxdecimaldigits` (precision) inte anges är standardvärdet 15.

Notera
Det finns olika alternativ för att översätta PostGIS-geometrier till X3D eftersom X3D-geometrityper inte mappar direkt till PostGIS-geometrityper och några nyare X3D-typer som kan vara bättre mappningar som vi har undvikit eftersom de flesta renderingsverktyg för närvarande inte stöder dem. Det här är de mappningar vi har bestämt oss för. Skicka gärna en buggbiljett om du har tankar om idén eller sätt som vi kan låta människor ange sina föredragna mappningar.
Nedan visas hur vi för närvarande mappar PostGIS 2D/3D-typer till X3D-typer

The 'options' argument is a bitfield that denotes whether to represent coordinates with the X3D GeoCoordinates geospatial node and whether to flip the x/y axis. By default, `ST_AsX3D` outputs in database form (long,lat or X,Y), but the X3D default of lat/lon, y/x may be preferred.

- 0: X/Y i databasordning (t.ex. long/lat = X,Y är standard databasordning), standardvärde och icke-spatiala koordinater (bara vanliga gamla Coordinate-taggar).
- 1: Vänd X och Y. Om det används tillsammans med alternativomkopplaren GeoCoordinate kommer utdata att vara standard latitude_firstöch koordinaterna kommer också att vändas.
- 2: Utdata av koordinater i GeoSpatial GeoCoordinates. Detta alternativ kommer att ge ett fel om geometrierna inte är i WGS 84 long lat (srid: 4326). Detta är för närvarande den enda GeoCoordinate-typen som stöds. Se X3D-specifikationerna som anger ett spatialt referenssystem. Standardutdata kommer att vara `GeoCoordinate geoSystem='GDWElongitude_first"'`. Om du föredrar X3D-standarden `GeoCoordinate geoSystem='GDWElatitude_first" '` använder du `(2 + 1) = 3`

PostGIS-typ	2D X3D Typ	3D X3D Typ
LINESTRING	LineSet (coordinates are promoted to 3D)	Linjeuppsättning
MULTILINESTRING	IndexedLineSet (coordinates are promoted to 3D)	IndexeradLineSet
MULTIPOINT	Polypunkt2D	PointSet
PUNKT	matar ut de utrymmesavgränsade koordinaterna	matar ut de utrymmesavgränsade koordinaterna
(MULTI) POLYGON, POLYHEDRALSYTA	IndexedFaceSet (coordinates are promoted to 3D; inner rings currently output as another faceset)	IndexedFaceSet (inre ringar som för närvarande matas ut som en annan faceset)
TIN	TriangleSet2D (ännu ej implementerad)	IndexeradTriangeluppsättning
CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, MULTICURVE, MULTISURFACE	linearized, then emitted with the linear geometry mapping	linearized, then emitted with the linear geometry mapping

PostGIS-typ	2D X3D Typ	3D X3D Typ
NURBSCURVE	NurbsCurve with controlPoint, knot, weight, and order fields	NurbsCurve with controlPoint, knot, weight, and order fields

Notera
stöd för 2D-geometri ännu inte komplett. Inre ringar ritas för närvarande bara som separata polygoner. Vi arbetar på dessa.

M coordinates are not represented in X3D output.

Många framsteg sker inom 3D-rymden, särskilt med X3D Integration med HTML5

Det finns också en trevlig X3D-visare med öppen källkod som du kan använda för att visa renderade geometrier. Gratis Wrl http://freewrl.sourceforge.net/ binärfiler tillgängliga för Mac, Linux och Windows. Använd FreeWRL_Launcher-paketet för att visa geometrierna.

Kolla också in PostGIS minimalistiska X3D-visningsprogram som använder den här funktionen och x3dDom html/js open source toolkit.

Tillgänglighet: 2.0.0: ISO-IEC-19776-1.2-X3DEncodings-XML

Förbättrad: 2.2.0: Stöd för GeoCoordinates och vändning av axlar (x/y, long/lat). Titta på alternativ för detaljer.

Enhanced: 3.7.0: Support for curve geometries. NURBSCURVE is emitted as X3D NurbsCurve; other curve types are linearized before output.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Create a complete X3D document containing a cube that is viewable in FreeWRL and other X3D viewers.

Code

```
SELECT '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d ↩
    -3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
       <Appearance>
            <Material emissiveColor=''0 0 1''/>
       </Appearance
> ' ||
       ST_AsX3D(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )')) ||
      '</Shape>
    </Transform>
  </Scene>
</X3D
>' As x3ddoc;
```

Output

```
    x3ddoc
    --------
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
       <Appearance>
            <Material emissiveColor='0 0 1'/>
       </Appearance>
       <IndexedFaceSet  coordIndex='0 1 2 3 -1 4 5 6 7 -1 8 9 10 11 -1 12 13 14 15 -1 16 17 18 19 -1 20 21 22 23'>
            <Coordinate point='0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0 ↩
     1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1' />
      </IndexedFaceSet>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

PostGIS buildings.

Kopiera och klistra in resultatet av den här frågan i x3d scene viewer och klicka på Show

Code

```
SELECT string_agg('<Shape
>' || ST_AsX3D(CG_Extrude(geom, 0,0, i*0.5)) ||
    '<Appearance>
          <Material diffuseColor="' || (0.01*i)::text || ' 0.8 0.2" specularColor="' ||  ↩
    (0.05*i)::text || ' 0 0.5"/>
        </Appearance>
    </Shape
>', '')
FROM ST_Subdivide(ST_Letters('PostGIS'),20) WITH ORDINALITY AS f(geom,i);
```

The following preview shows the extruded geometry generated from a simplified version of the same source lettering. It is a geometry preview, not X3D markup; use the preceding `ST_AsX3D` query output in an X3D viewer to inspect the native X3D scene.

Code

```
SELECT
    CG_Extrude(
        ST_SimplifyPreserveTopology(ST_Letters('PostGIS'), 2),
        0, 0, 10
    ) AS buildings;
```

Figure

An Octagon elevated 3 Units and decimal precision of 6.

Code

```
SELECT ST_AsX3D(ST_Translate(ST_Force3D(ST_Buffer(ST_Point(10,10),5, 'quad_segs=2')), 0,0,
    3),
  6) As x3dfrag;
```

Output

```
x3dfrag
--------
<IndexedFaceSet  convex='false' coordIndex='0 1 2 3 4 5 6 7'
><Coordinate point='15 10 3 13.535534 6.464466 3 10 5 3 6.464466 6.464466 3 5 10 3 6.464466 13.535534 3 10 15 3 ↩
     13.535534 13.535534 3 ' /></IndexedFaceSet>
```

TIN.

Code

```
SELECT ST_AsX3D(ST_GeomFromEWKT('TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
            )), ((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
            ))
            )')) As x3dfrag;
```

Output

```
    x3dfrag
    --------
<IndexedTriangleSet  index='0 1 2 3 4 5'
><Coordinate point='0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0'/></IndexedTriangleSet>
```

Closed multilinestring (the boundary of a polygon with holes).

Code

```
SELECT ST_AsX3D(ST_GeomFromEWKT('MULTILINESTRING((20 0 10,16 -12 10,0 -16 10,-12 -12 10,-20 ↩
     0 10,-12 16 10,0 24 10,16 16 10,20 0 10),
  (12 0 10,8 8 10,0 12 10,-8 8 10,-8 0 10,-8 -4 10,0 -8 10,8 -4 10,12 0 10))')
) As x3dfrag;
```

Output

```
    x3dfrag
    --------
<IndexedLineSet  coordIndex='0 1 2 3 4 5 6 7 0 -1 8 9 10 11 12 13 14 15 8'>
    <Coordinate point='20 0 10 16 -12 10 0 -16 10 -12 -12 10 -20 0 10 -12 16 10 0 24 10 16 16 10 12 0 10 8 8 10 0 12  ↩
     10 -8 8 10 -8 0 10 -8 -4 10 0 -8 10 8 -4 10 ' />
 </IndexedLineSet>
```

7.9.3.14 ST_GeoHash

ST_GeoHash — Returnerar en GeoHash-representation av geometrin.

Synopsis

text **ST_GeoHash**(geometry geom, integer maxchars=full_precision_of_point);

Beskrivning

Beräknar en GeoHash-representation av en geometri. En GeoHash kodar en geografisk punkt till en textform som är sorterbar och sökbar baserat på prefix. En kortare GeoHash är en mindre exakt representation av en punkt. Den kan betraktas som en låda som innehåller punkten.

Icke-punktgeometriska värden med en utsträckning som inte är noll kan också mappas till GeoHash-koder. Kodens precision beror på geometrins geografiska utbredning.

Om `maxchars` inte anges är den returnerade GeoHash-koden för den minsta cell som innehåller indatageometrin. Punkter returnerar en GeoHash med 20 teckens precision (ungefär tillräckligt för att rymma indata med full dubbel precision). Andra geometriska typer kan returnera en GeoHash med lägre precision, beroende på geometrins omfattning. Större geometrier representeras med mindre precision, mindre med mer precision. Den ruta som bestäms av GeoHash-koden innehåller alltid indatafunktionen.

Om `maxchars` anges har den returnerade GeoHash-koden högst så många tecken. Den mappas till en (eventuellt) lägre precisionsrepresentation av indatageometrin. För icke-punkter är beräkningens startpunkt centrum för geometrins avgränsande box.

Tillgänglighet: 1.4.0

Notera
ST_GeoHash kräver att indatageometrin är i geografiska koordinater (lon/lat).

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_GeoHash(ST_Point(-126, 48));
```

Output

```
c0w3hf1s70w3hf1s70w3
```

Code

```
SELECT ST_GeoHash(ST_Point(-126, 48), 5);
```

Output

```
c0w3h
```

This example shows that the GeoHash for a line containing a point is a prefix of the point code.

Code

```
SELECT ST_GeoHash('LINESTRING(-126 48,-126.1 48.1)'::geometry);
```

Output

```
c0w3
```

Se även

ST_GeomFromGeoHash, ST_PointFromGeoHash, ST_Box2dFromGeoHash

7.10 Operatorer

7.10.1 Operatorer för avgränsande box

7.10.1.1 &&

&& — Returnerar TRUE om A:s 2D-begränsningsbox skär B:s 2D-begränsningsbox.

Synopsis

boolean **&&**(geometry A , geometry B);
boolean **&&**(geography A , geography B);

Beskrivning

Operatorn && returnerar TRUE om den 2D-begränsande boxen för geometri A skär den 2D-begränsande boxen för geometri B.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes.

Tillgänglighet: 1.5.0 stöd för geografi infördes.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

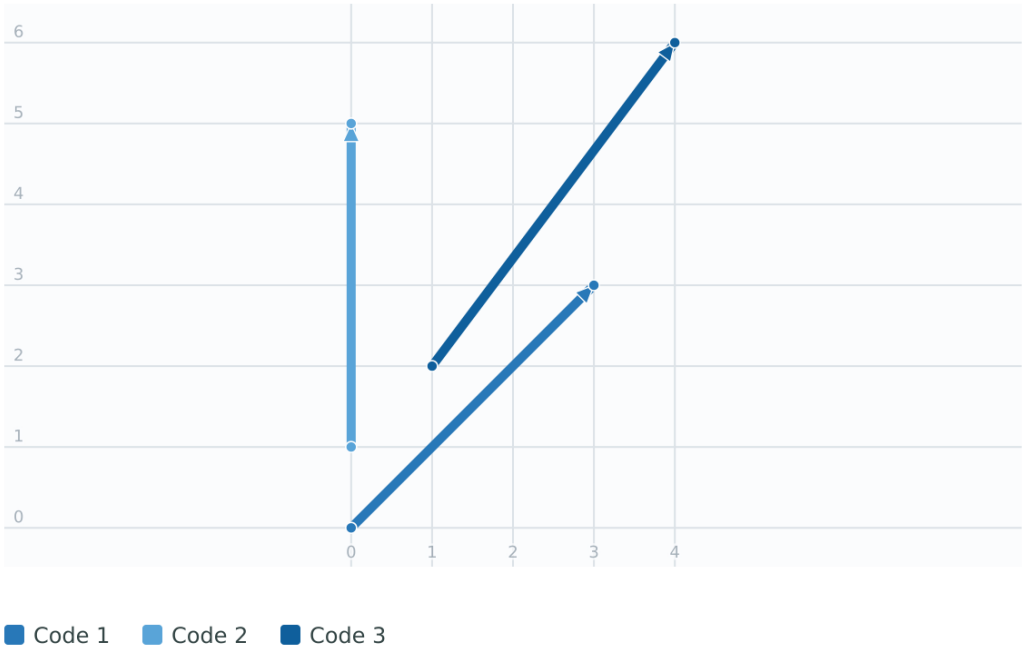
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 && tbl2.column2 AS overlaps
FROM ( VALUES
    (1, 'LINESTRING(0 0, 3 3)'::geometry),
    (2, 'LINESTRING(0 1, 0 5)'::geometry)) AS tbl1,
( VALUES
    (3, 'LINESTRING(1 2, 4 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overlaps
---------+---------+----------
       1 |       3 | t
       2 |       3 | f
(2 rows)
```

Figure



Se även

ST_Intersects, ST_Extent, |&>, &>, &<|, &<, ~, @

7.10.1.2 &&(geometry,box2df)

&&(geometry,box2df) — Returnerar TRUE om en geometris (cachelagrade) 2D-begränsningsbox skär en 2D-begränsningsbox med flytande precision (BOX2DF).

Synopsis

boolean **&&**(geometry A , box2df B);

Beskrivning

Operatorn && returnerar TRUE om den cachade 2D-begränsningsrutan för geometri A skär 2D-begränsningsru B, med float-precision. Detta innebär att om B är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-begränsningsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_Point(1,1) && ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) AS overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

Se även

&&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.3 &&(box2df,geometry)

&&(box2df,geometry) — Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) skär en geometris (cachelagrade) 2D-gränsbox.

Synopsis

boolean **&&**(box2df A , geometry B);

Beskrivning

Operatorn && returnerar TRUE om 2D-begränsningsrutan A skär den cachade 2D-begränsningsrutan för geometri B, med float-precision. Detta innebär att om A är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_Point(1,1) AS overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

Se även

&&(geometry,box2df), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.4 &&(box2df,box2df)

&&(box2df,box2df) — Returnerar TRUE om två 2D-begränsningsboxar med flytande precision (BOX2DF) skär varandra.

Synopsis

boolean **&&**(box2df A , box2df B);

Beskrivning

Operatorn && returnerar TRUE om två 2D-begränsningsrutor A och B skär varandra, med float-precision. Detta innebär att om A (eller B) är en box2d (dubbel precision), konverteras den internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operator är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_MakeBox2D(ST_Point(1,1), ST_Point ↩
    (3,3)) AS overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

Se även

&&(geometry,box2df), &&(box2df,geometry), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.5 &&&

&&& — Returnerar TRUE om A:s n-D-gränsbox skär B:s n-D-gränsbox.

Synopsis

boolean **&&&**(geometry A , geometry B);

Beskrivning

Operatorn &&& returnerar TRUE om n-D-gränsboxen för geometri A skär n-D-gränsboxen för geometri B.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Tillgänglighet: 2.0.0

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

3D LineStrings.

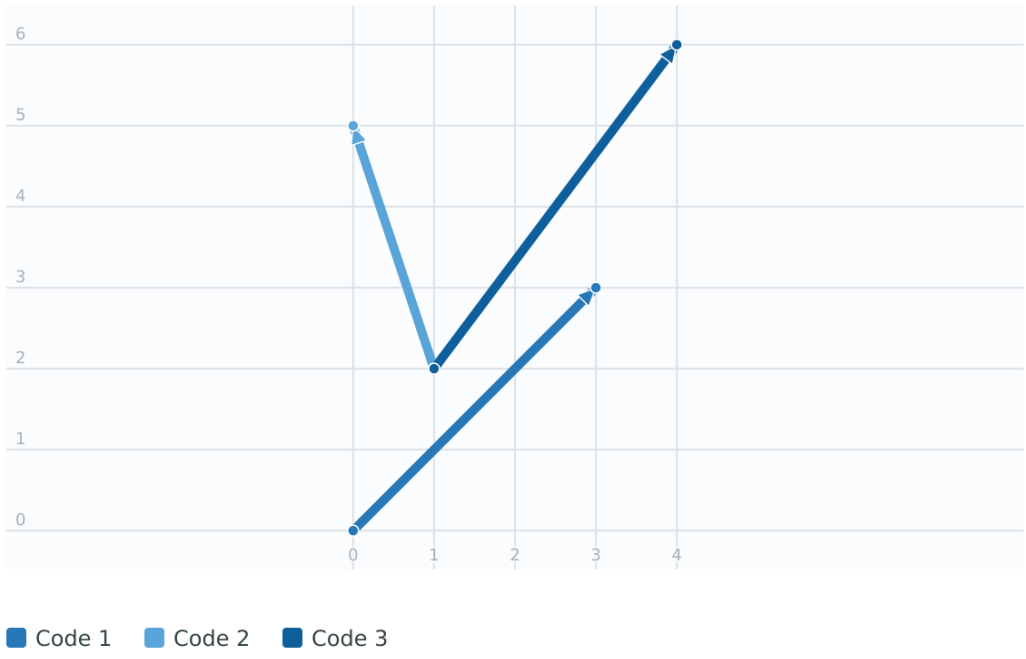
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &&& tbl2.column2 AS overlaps_3d,
                        tbl1.column2 && tbl2.column2 AS overlaps_2d
FROM ( VALUES
    (1, 'LINESTRING Z(0 0 1, 3 3 2)'::geometry),
    (2, 'LINESTRING Z(1 2 0, 0 5 -1)'::geometry)) AS tbl1,
( VALUES
    (3, 'LINESTRING Z(1 2 1, 4 6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overlaps_3d | overlaps_2d
---------+---------+-------------+-------------
       1 |       3 | t           | t
       2 |       3 | f           | t
```

Figure



3M LineStrings.

Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &&& tbl2.column2 AS overlaps_3zm,
                       tbl1.column2 && tbl2.column2 AS overlaps_2d
FROM ( VALUES
    (1, 'LINESTRING M(0 0 1, 3 3 2)'::geometry),
    (2, 'LINESTRING M(1 2 0, 0 5 -1)'::geometry)) AS tbl1,
( VALUES
    (3, 'LINESTRING M(1 2 1, 4 6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overlaps_3zm | overlaps_2d
---------+---------+--------------+-------------
       1 |       3 | t            | t
       2 |       3 | f            | t
```

Figure

Se även

&&

7.10.1.6 &&&(geometry,gidx)

&&&(geometry,gidx) — Returnerar TRUE om en geometris (cachade) n-D-begränsningsbox skär en n-D-begränsningsbox med flytande precision (GIDX).

Synopsis

boolean **&&&**(geometry A , gidx B);

Beskrivning

Operatorn &&& returnerar TRUE om den cachade n-D-begränsningsrutan för geometri A skär n-D-begränsningsrutan B, med float-precision. Detta innebär att om B är en (dubbel precision) box3d, kommer den att konverteras internt till en 3D-begränsningsbox med flytande precision (GIDX)

Notera
Denna operator är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_MakePoint(1,1,1) &&& ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) AS ↩
    overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

Se även

&&&(gidx,geometry), &&&(gidx,gidx)

7.10.1.7 &&&(gidx,geometry)

&&&(gidx,geometry) — Returnerar TRUE om en n-D avgränsningsbox med flytande precision (GIDX) skär en geometris (cachelagrade) n-D avgränsningsbox.

Synopsis

boolean **&&&**(gidx A , geometry B);

Beskrivning

Operatorn &&& returnerar TRUE om n-D-begränsningsrutan A skär den cachade n-D-begränsningsrutan för geometri B, med float-precision. Detta innebär att om A är en box3d (dubbel precision) kommer den att konverteras internt till en 3D-gränsbox med flytande precision (GIDX)

Notera
Denna operator är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_MakePoint(1,1,1) AS ↩
    overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

Se även

&&&(geometry,gidx), &&&(gidx,gidx)

7.10.1.8 &&&(gidx,gidx)

&&&(gidx,gidx) — Returnerar TRUE om två gränsboxar (GIDX) med n-D floatprecision skär varandra.

Synopsis

boolean **&&&**(gidx A , gidx B);

Beskrivning

Operatorn &&& returnerar TRUE om två n-D-begränsningsrutor A och B skär varandra, med float-precision. Detta innebär att om A (eller B) är en (dubbel precision) box3d, konverteras den internt till en 3D-gränsbox med flytande precision (GIDX)

Notera
Denna operator är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_3DMakeBox(ST_MakePoint ↩
    (1,1,1), ST_MakePoint(3,3,3)) AS overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

Se även

&&&(geometry,gidx), &&&(gidx,geometry)

7.10.1.9 &<

&< — Returnerar TRUE om A:s avgränsande box överlappar eller ligger till vänster om B:s.

Synopsis

boolean **&<**(geometry A , geometry B);

Beskrivning

Operatorn &< returnerar TRUE om geometri A:s avgränsande box överlappar eller är till vänster om geometri B:s avgränsande box, eller mer exakt, överlappar eller INTE är till höger om geometri B:s avgränsande box.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

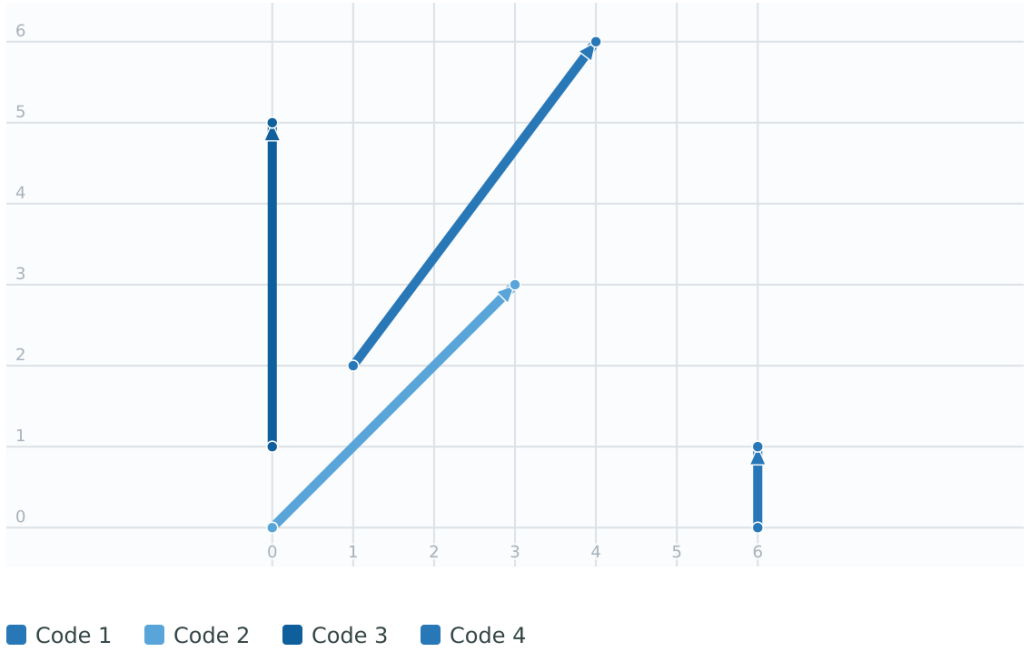
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &< tbl2.column2 AS overleft
FROM
  ( VALUES
    (1, 'LINESTRING(1 2, 4 6)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)'::geometry),
    (3, 'LINESTRING(0 1, 0 5)'::geometry),
    (4, 'LINESTRING(6 0, 6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overleft
---------+---------+----------
       1 |       2 | f
       1 |       3 | f
       1 |       4 | t
(3 rows)
```

Figure



Se även

&&, |&>, &>, &<|

7.10.1.10 &<|

&<| — Returnerar TRUE om A:s avgränsande box överlappar eller ligger under B:s.

Synopsis

boolean **&<|**(geometry A , geometry B);

Beskrivning

Operatorn &<| returnerar TRUE om begränsningsrutan för geometri A överlappar eller ligger under begränsningsrutan för geometri B, eller mer exakt, överlappar eller INTE ligger över begränsningsrutan för geometri B.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

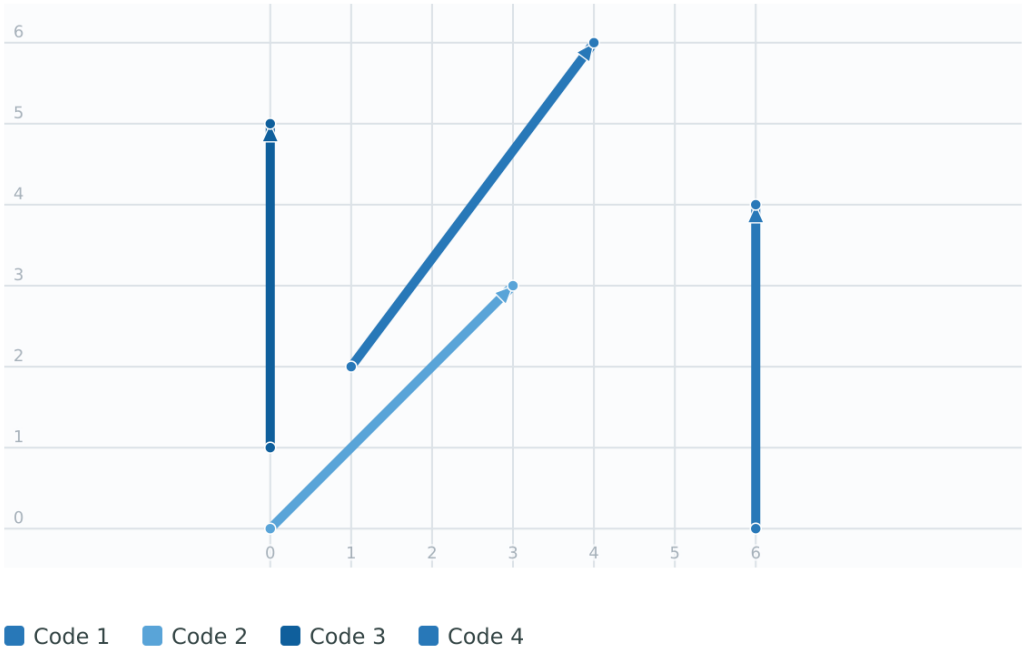
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &<| tbl2.column2 AS overbelow
FROM
  ( VALUES
    (1, 'LINESTRING(6 0, 6 4)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)'::geometry),
    (3, 'LINESTRING(0 1, 0 5)'::geometry),
    (4, 'LINESTRING(1 2, 4 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overbelow
---------+---------+-----------
       1 |       2 | f
       1 |       3 | t
       1 |       4 | t
(3 rows)
```

Figure



Se även

&&, |&>, &>, &<

7.10.1.11 &>

&> — Returnerar TRUE om A:s avgränsande box överlappar eller är till höger om B:s.

Synopsis

boolean **&>**(geometry A , geometry B);

Beskrivning

Operatorn &> returnerar TRUE om geometri A:s avgränsande box överlappar eller ligger till höger om geometri B:s avgränsande box, eller mer exakt, överlappar eller INTE ligger till vänster om geometri B:s avgränsande box.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

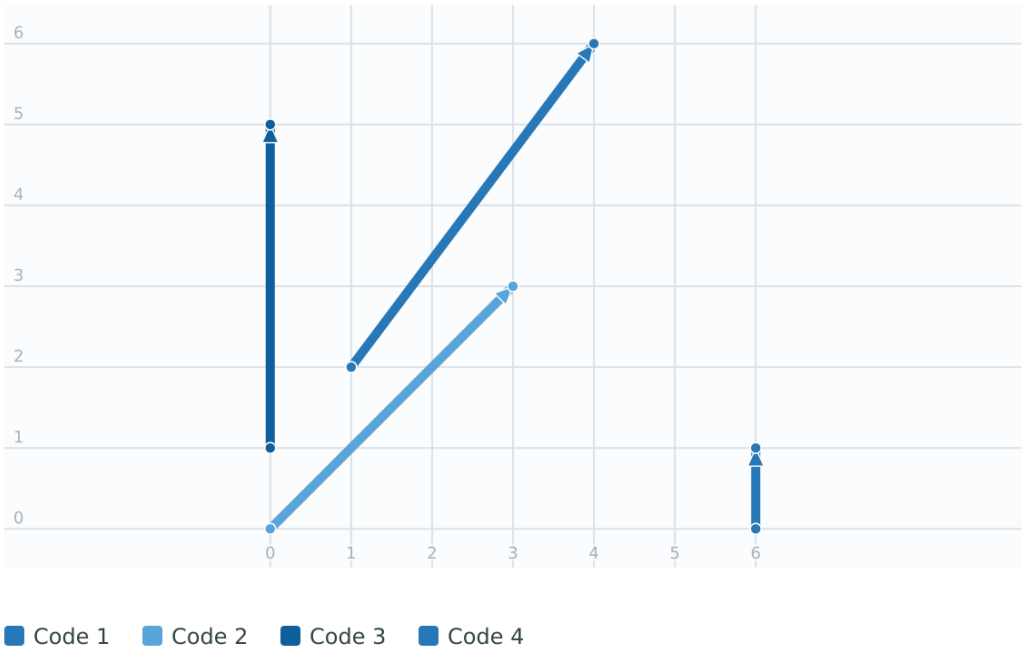
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &
> tbl2.column2 AS overright
FROM
  ( VALUES
    (1, 'LINESTRING(1 2,4 6)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0,3 3)'::geometry),
    (3, 'LINESTRING(0 1,0 5)'::geometry),
    (4, 'LINESTRING(6 0,6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overright
---------+---------+-----------
       1 |       2 | t
       1 |       3 | t
       1 |       4 | f
(3 rows)
```

Figure



Se även

&&, |&>, &<|, &<

7.10.1.12 <<

<< — Returnerar TRUE om A:s avgränsande box ligger strikt till vänster om B:s.

Synopsis

boolean **<<**(geometry A , geometry B);

Beskrivning

Operatorn << returnerar TRUE om geometri A:s begränsningsbox ligger strikt till vänster om geometri B:s begränsningsbox.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

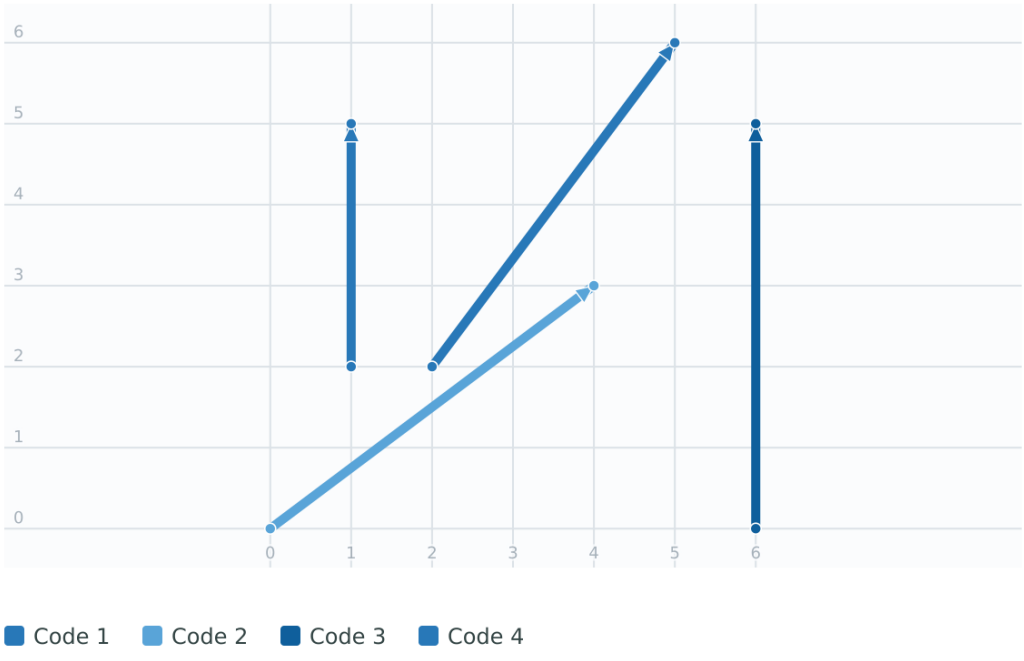
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 << tbl2.column2 AS left
FROM
  ( VALUES
    (1, 'LINESTRING (1 2, 1 5)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0, 4 3)'::geometry),
    (3, 'LINESTRING (6 0, 6 5)'::geometry),
    (4, 'LINESTRING (2 2, 5 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | left
---------+---------+------
       1 |       2 | f
       1 |       3 | t
       1 |       4 | t
(3 rows)
```

Figure



Se även

>>, |>>, <<|

7.10.1.13 <<|

<<| — Returnerar TRUE om A:s avgränsande box är strikt under B:s.

Synopsis

boolean **<<|**(geometry A , geometry B);

Beskrivning

Operatorn <<| returnerar TRUE om geometri A:s begränsningsbox ligger strikt under geometri B:s begränsningsbox.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

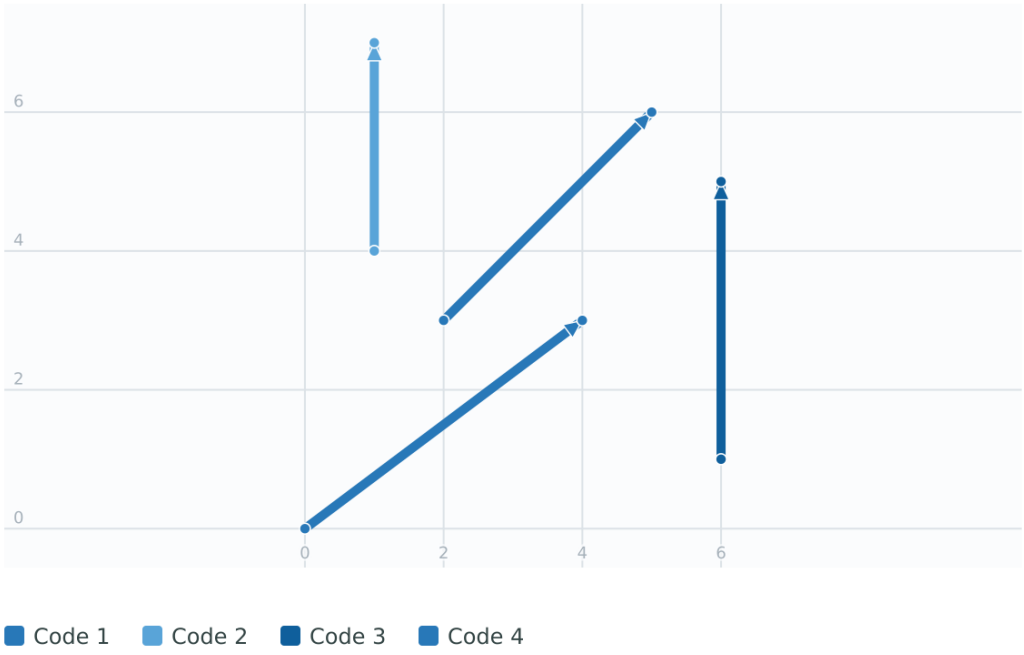
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 <<| tbl2.column2 AS below
FROM
  ( VALUES
    (1, 'LINESTRING (0 0, 4 3)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (1 4, 1 7)'::geometry),
    (3, 'LINESTRING (6 1, 6 5)'::geometry),
    (4, 'LINESTRING (2 3, 5 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | below
---------+---------+-------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



Se även

<<, >>, |>>

7.10.1.14 =

= — Returnerar TRUE om koordinaterna och koordinatordningen för geometri/geografi A är samma som koordinaterna och koordinatordningen för geometri/geografi B.

Synopsis

boolean =(geometry A , geometry B);
boolean =(geography A , geography B);

Beskrivning

Operatorn = returnerar SANT om koordinaterna och koordinatordningen geometri / geografi A är desamma som koordinaterna och koordinatordningen för geometri / geografi B. PostgreSQL använder operatorerna =, < och > som definieras för geometrier för att utföra interna ordningar och jämförelse av geometrier (dvs. i en GROUP BY- eller ORDER BY-klausul).

Notera
Endast geometri/geografi som är exakt lika i alla avseenden, med samma koordinater, i samma ordning, betraktas som lika av denna operator. För spatial likhet", som ignorerar saker som koordinatordning och kan upptäcka funktioner som täcker samma spatiala område med olika representationer, använd ST_OrderingEquals eller ST_Equals

Observera
Detta operand kommer INTE att använda några index som kan finnas tillgängliga på geometrierna. För ett indexassisterat exakt likhetstest, kombinera = med &&.

Ändrad: 2.4.0, i tidigare versioner var detta bounding box-likhet inte en geometrisk likhet. Om du behöver bounding box-likhet, använd ~= istället.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

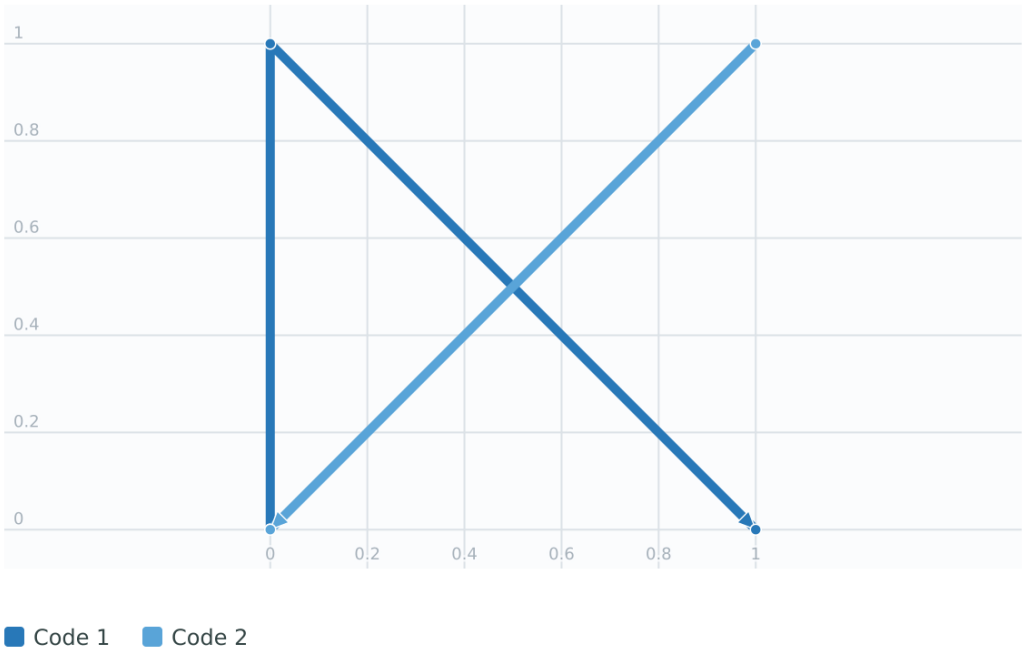
Code

```
SELECT 'LINESTRING(0 0,0 1,1 0)'::geometry = 'LINESTRING(1 1,0 0)'::geometry;
```

Output

```
 ?column?
----------
 f
(1 row)
```

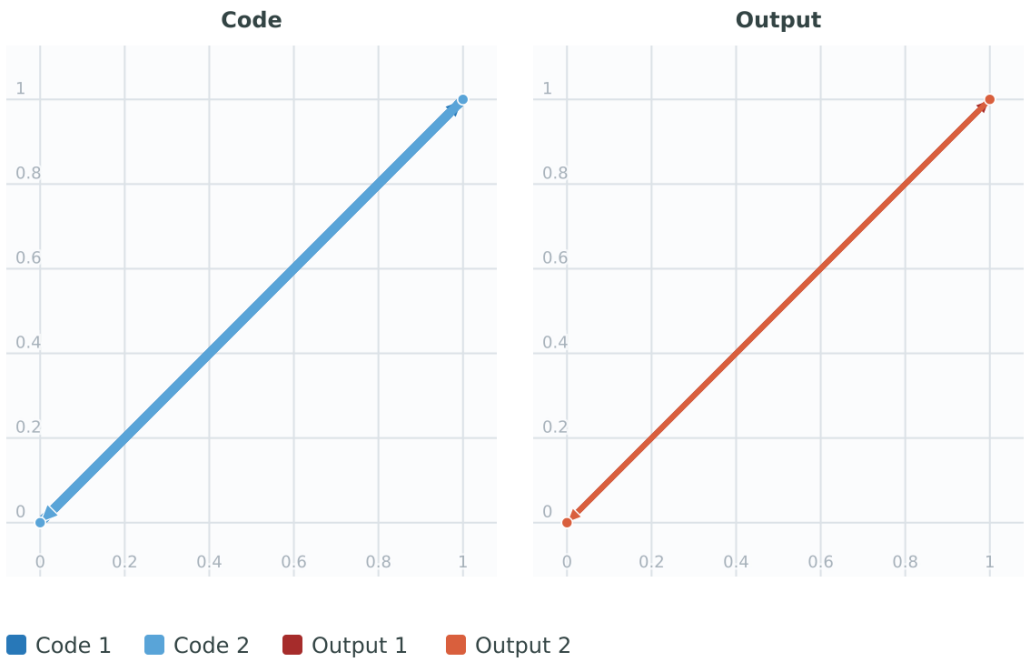
Figure



Code

```
SELECT column1
FROM ( VALUES
    ('LINESTRING(0 0, 1 1)'::geometry),
    ('LINESTRING(1 1, 0 0)'::geometry)) AS foo;
```

Figure

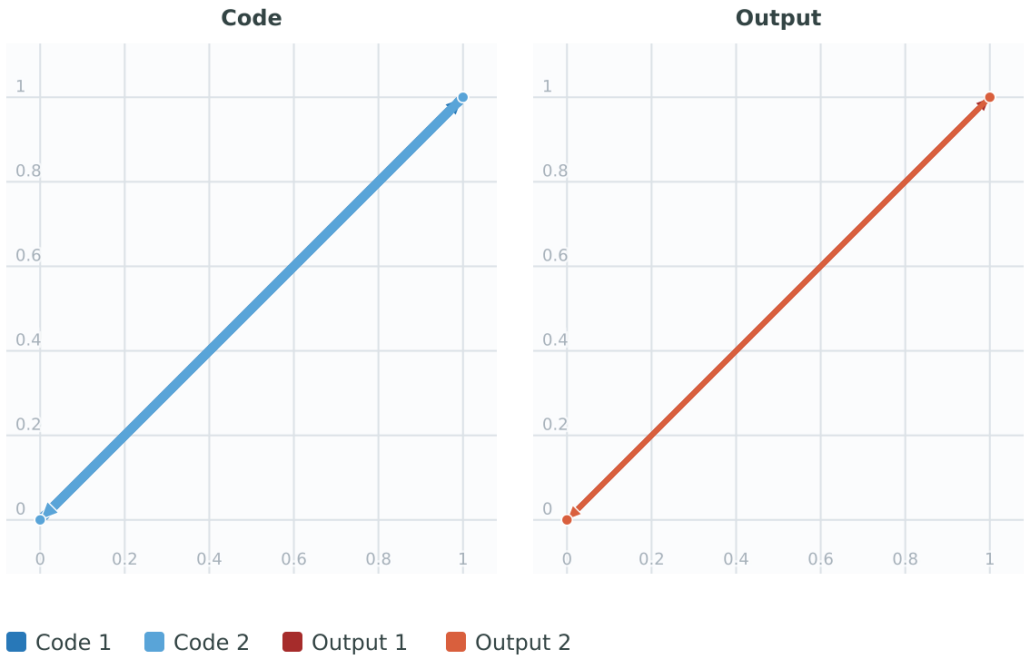


Note: the GROUP BY uses the -to compare for geometry equivalency.

Code

```
SELECT column1
FROM ( VALUES
    ('LINESTRING(0 0, 1 1)'::geometry),
    ('LINESTRING(1 1, 0 0)'::geometry)) AS foo
GROUP BY column1;
```

Figure



In versions prior to 2.0, this used to return true.

Code

```
SELECT 'POINT(1707296.37 4820536.77)'::geometry =
   'POINT(1707296.27 4820536.87)'::geometry As pt_intersect;
```

Output

```
 pt_intersect
--------------
 f
(1 row)
```

Se även

ST_Equals, ST_OrderingEquals, ~=

7.10.1.15 >>

>> — Returnerar TRUE om A:s avgränsande box ligger strikt till höger om B:s.

Synopsis

boolean **>>**(geometry A , geometry B);

Beskrivning

Operatorn >> returnerar TRUE om geometri A:s avgränsningsbox ligger strikt till höger om geometri B:s avgränsningsbox.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

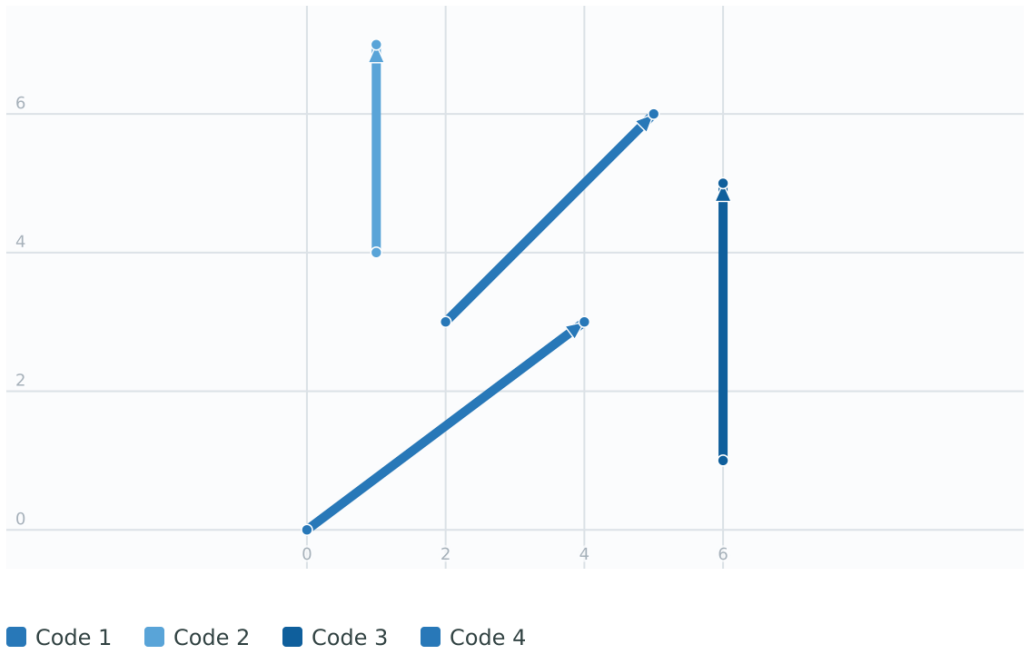
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2
>
> tbl2.column2 AS right
FROM
  ( VALUES
    (1, 'LINESTRING (2 3,5 6)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (1 4,1 7)'::geometry),
    (3, 'LINESTRING (6 1,6 5)'::geometry),
    (4, 'LINESTRING (0 0,4 3)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | right
---------+---------+-------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



Se även

<<, |>>, <<|

7.10.1.16 @

@ — Returnerar TRUE om A:s avgränsande box ingår i B:s.

Synopsis

boolean @(geometry A , geometry B);

Beskrivning

Operatorn @ returnerar TRUE om geometri A:s begränsningsbox är helt innesluten i geometri B:s begränsningsbox.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

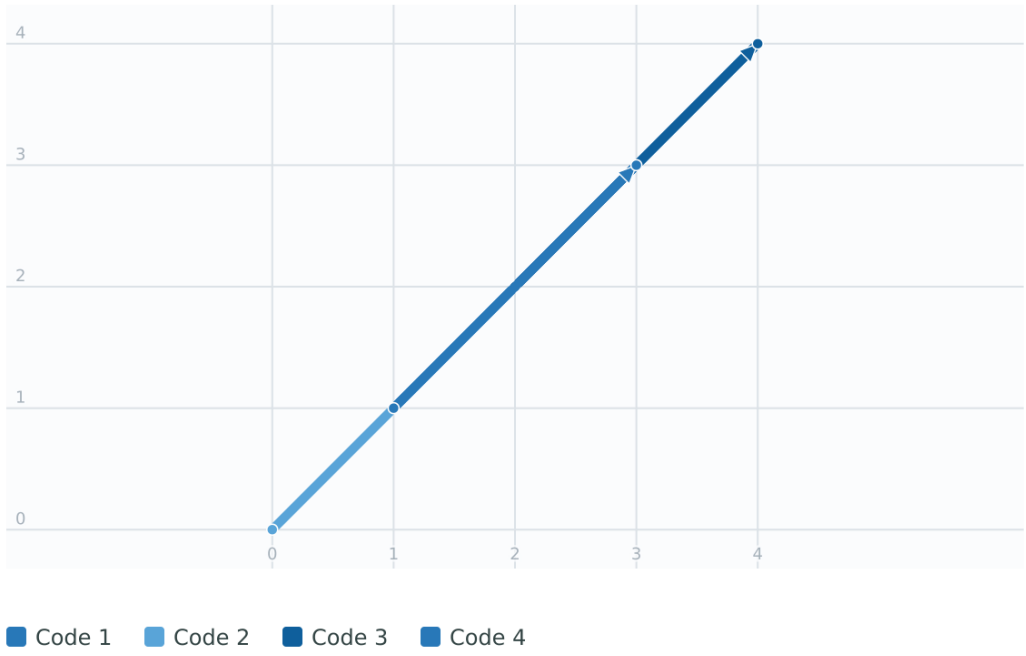
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 @ tbl2.column2 AS contained
FROM
  ( VALUES
    (1, 'LINESTRING (1 1, 3 3)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0, 4 4)'::geometry),
    (3, 'LINESTRING (2 2, 4 4)'::geometry),
    (4, 'LINESTRING (1 1, 3 3)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | contained
---------+---------+-----------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | t
(3 rows)
```

Figure



Se även

~, &&

7.10.1.17 @(geometry,box2df)

@(geometry,box2df) — Returnerar TRUE om en geometris 2D-begränsningsbox ingår i en 2D-begränsningsbo
med flytande precision (BOX2DF).

Synopsis

boolean @(geometry A , box2df B);

Beskrivning

Operatorn @ returnerar TRUE om A-geometrins 2D-begränsningsbox ingår i 2D-begränsningsboxen B, med float-precision. Detta innebär att om B är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_Buffer('POINT(2 2)'::geometry, 1) @ ST_MakeBox2D(ST_Point(0, 0), ST_Point(5, 5))  ↩
    AS is_contained;
```

Output

```
t
```

Se även

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry) ~(box2df,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.18 @(box2df,geometry)

@(box2df,geometry) — Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) ingår i en geometris 2D-gränsbox.

Synopsis

boolean @(box2df A , geometry B);

Beskrivning

Operatorn @ returnerar TRUE om 2D-gränsboxen A ingår i B-geometrins 2D-gränsbox, med float-precision. Detta innebär att om B är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_MakeBox2D(ST_Point(2, 2), ST_Point(3, 3)) @ ST_Buffer('POINT(1 1)'::geometry, 10) ↩
    AS is_contained;
```

Output

```
t
```

Se även

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry) ~(box2df,box2df), @(geometry,box2df), @(box2df,box2df)

7.10.1.19 @(box2df,box2df)

@(box2df,box2df) — Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) ingår i en annan avgränsande 2D-box med flytande precision.

Synopsis

boolean @(box2df A , box2df B);

Beskrivning

Operatorn @ returnerar TRUE om 2D-begränsningsrutan A ingår i 2D-begränsningsrutan B, med float-precision. Detta innebär att om A (eller B) är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_MakeBox2D(ST_Point(2, 2), ST_Point(3, 3)) @ ST_MakeBox2D(ST_Point(0, 0), ST_Point ↩
    (5, 5)) AS is_contained;
```

Output

```
t
```

Se även

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry) ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry)

7.10.1.20 |&>

|&> — Returnerar TRUE om A:s avgränsande box överlappar eller ligger ovanför B:s.

Synopsis

boolean **|&>**(geometry A , geometry B);

Beskrivning

Operatorn |&> returnerar TRUE om geometri A:s begränsningsbox överlappar eller ligger ovanför geometri B:s begränsningsbox, eller mer exakt, överlappar eller INTE ligger under geometri B:s begränsningsbox.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

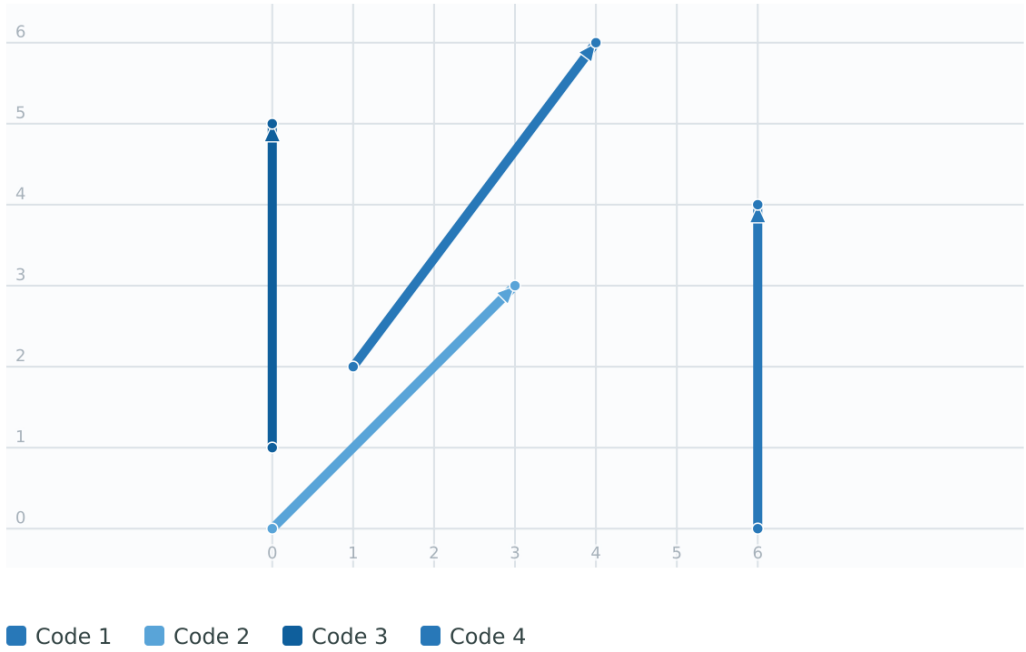
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 |&
> tbl2.column2 AS overabove
FROM
  ( VALUES
    (1, 'LINESTRING(6 0,6 4)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0,3 3)'::geometry),
    (3, 'LINESTRING(0 1,0 5)'::geometry),
    (4, 'LINESTRING(1 2,4 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overabove
---------+---------+-----------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



Se även

&&, &>, &<|, &<

7.10.1.21 |>>

|>> — Returnerar TRUE om A:s avgränsande box är strikt över B:s.

Synopsis

boolean **|>>**(geometry A , geometry B);

Beskrivning

Operatorn |>> returnerar TRUE om geometri A:s begränsningsbox ligger strikt ovanför geometri B:s begränsningsbox.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

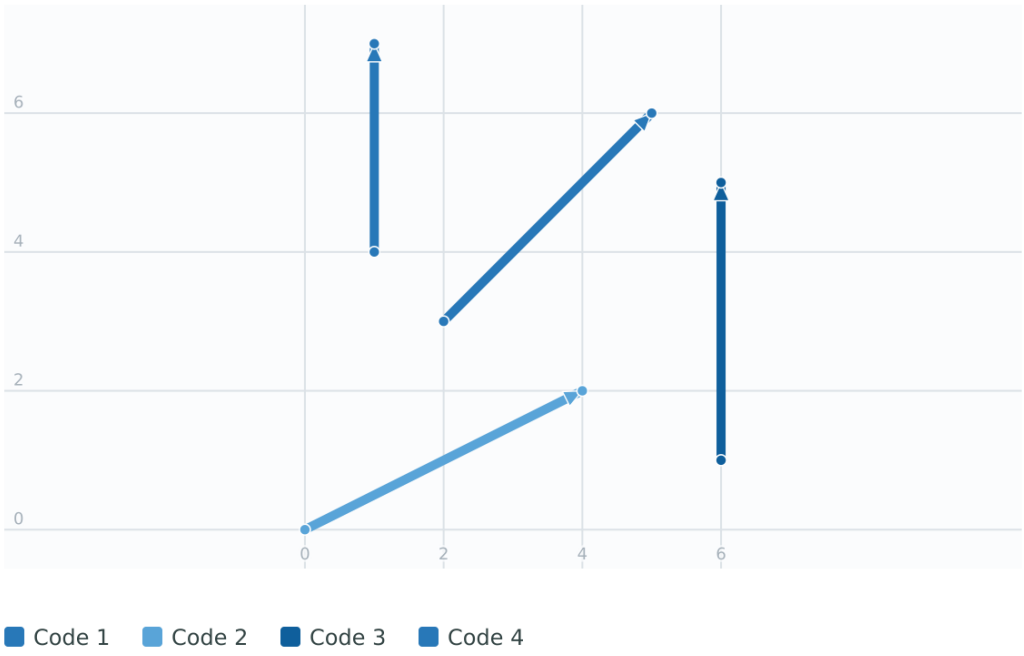
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 |>> tbl2.column2 AS above
FROM
  ( VALUES
    (1, 'LINESTRING (1 4,1 7)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0,4 2)'::geometry),
    (3, 'LINESTRING (6 1,6 5)'::geometry),
    (4, 'LINESTRING (2 3,5 6)'::geometry)) AS tbl2;
```

Output

```
column1 | column1 | above
---------+---------+-------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



Se även

<<, >>, <<|

7.10.1.22 ~

~ — Returnerar TRUE om A:s avgränsande box innehåller B:s.

Synopsis

boolean ~(geometry A , geometry B);

Beskrivning

Operatorn ~ returnerar TRUE om den avgränsande boxen för geometri A helt innehåller den avgränsande boxen för geometri B.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

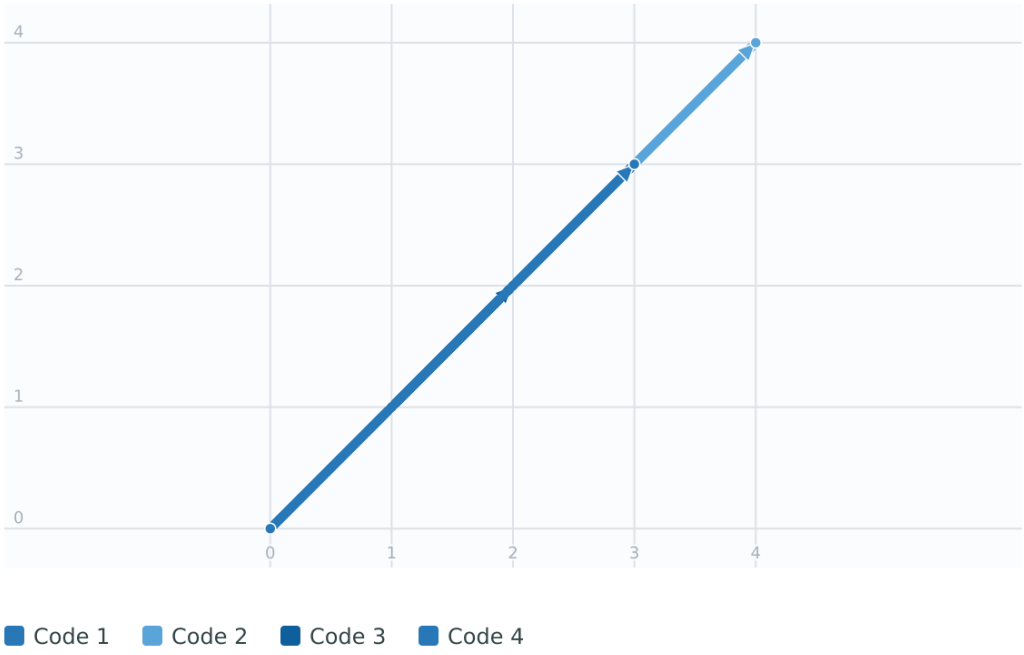
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 ~ tbl2.column2 AS contains
FROM
  ( VALUES
    (1, 'LINESTRING (0 0,3 3)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0,4 4)'::geometry),
    (3, 'LINESTRING (1 1,2 2)'::geometry),
    (4, 'LINESTRING (0 0,3 3)'::geometry)) AS tbl2;
```

Output

```
column1 | column1 | contains
---------+---------+----------
       1 |       2 | f
       1 |       3 | t
       1 |       4 | t
(3 rows)
```

Figure



Se även

@, &&

7.10.1.23 ~(geometry,box2df)

~(geometry,box2df) — Returnerar TRUE om en geometris 2D-bindningsbox innehåller en 2D-bindningsbox med floatprecision (GIDX).

Synopsis

boolean ~(geometry A , box2df B);

Beskrivning

Operatorn ~ returnerar TRUE om 2D-begränsningsrutan för en geometri A innehåller 2D-begränsningsrutan B, med float-precision. Detta innebär att om B är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_Buffer('POINT(1 1)'::geometry, 10) ~ ST_MakeBox2D(ST_Point(0, 0), ST_Point(2, 2)) ↩
     AS contains;
```

Output

```
t
```

Se även

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.24 ~(box2df,geometry)

~(box2df,geometry) — Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) innehåller en geometris 2D-bondingbox.

Synopsis

boolean ~(box2df A , geometry B);

Beskrivning

Operatorn ~ returnerar TRUE om 2D-gränsboxen A innehåller B-geometrins gränsbox, med float-precision. Detta innebär att om A är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_MakeBox2D(ST_Point(0, 0), ST_Point(5, 5)) ~ ST_Buffer('POINT(2 2)'::geometry, 1)  ↩
    AS contains;
```

Output

```
t
```

Se även

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.25 ~(box2df,box2df)

~(box2df,box2df) — Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) innehåller en annan avgränsande 2D-box med flytande precision (BOX2DF).

Synopsis

boolean ~(box2df A , box2df B);

Beskrivning

Operatorn ~ returnerar TRUE om 2D-begränsningsrutan A innehåller 2D-begränsningsrutan B, med float-precision. Detta innebär att om A är en (dubbel precision) box2d, kommer den att konverteras internt till en 2D-gränsbox med flytande precision (BOX2DF)

Notera
Denna operand är avsedd att användas internt av BRIN-index, mer än av användare.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Exempel

Code

```
SELECT ST_MakeBox2D(ST_Point(0, 0), ST_Point(5, 5)) ~ ST_MakeBox2D(ST_Point(2, 2), ST_Point ↩
    (3, 3)) AS contains;
```

Output

```
t
```

Se även

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry) @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.26 ~=

~= — Returnerar TRUE om A:s avgränsande box är densamma som B:s.

Synopsis

boolean **~=**(geometry A , geometry B);

Beskrivning

Operatorn ~= returnerar TRUE om avgränsningsrutan för geometri/geografi A är densamma som avgränsningsrutan för geometri/geografi B.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Tillgänglighet: 1.5.0 ändrat beteende

Denna funktion stöder polyedriska ytor.

Varning
Denna operator har ändrat beteende i PostGIS 1.5 från att testa för faktisk geometrisk likhet till att endast kontrollera för bounding box likhet. För att komplicera saker och ting beror det också på om du har gjort en hård eller mjuk uppgradering vilket beteende din databas har. För att ta reda på vilket beteende din databas har kan du köra frågan nedan. För att kontrollera sann jämlikhet använder du ST_OrderingEquals eller ST_Equals.

Exempel

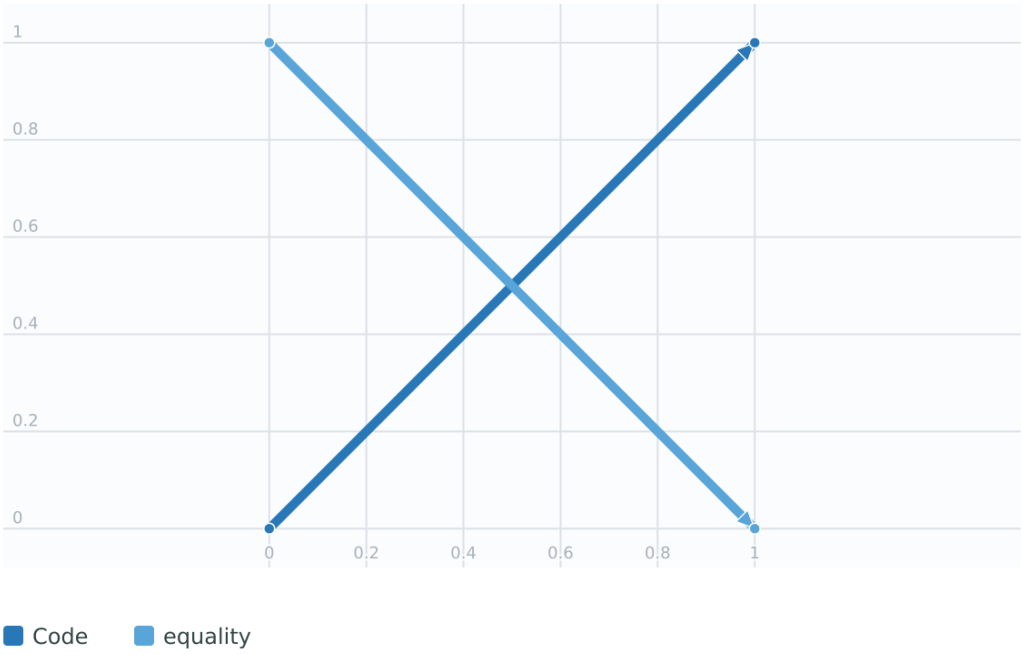
Code

```
SELECT 'LINESTRING(0 0,1 1)'::geometry ~= 'LINESTRING(0 1,1 0)'::geometry AS equality;
```

Output

```
t
```

Figure



Se även

ST_Equals, ST_OrderingEquals, =

7.10.2 Avståndsoperatorer

7.10.2.1 <->

<-> — Returnerar 2D-avståndet mellan A och B.

Synopsis

double precision **<->**(geometry A , geometry B);
double precision **<->**(geography A , geography B);

Beskrivning

The <-> operator returns the 2D distance between two geometries. Used in the ÖRDER BY”clause provides index-assisted nearest-neighbor result sets with true KNN distance search for geometries, and distance on the sphere for geographies.

Notera
Denna operand använder 2D GiST-index som kan finnas tillgängliga för geometrierna. Den skiljer sig från andra operatorer som använder spatiala index genom att det spatiala indexet endast används när operatorn är i ORDER BY-satsen.

Notera
Indexet börjar bara gälla om en av geometrierna är en konstant (inte i en underfråga). t.ex. 'SRID=3005;POINT(1011102 450541)'::geometry istället för a.geom

Se PostGIS workshop: Nearest-Neighbor Searching för ett detaljerat exempel.

Enhanced: 2.2.0 -- True KNN ("K nearest neighbor") behavior for geometry and geography. Note for geography KNN is based on sphere rather than spheroid.

Changed: 2.2.0 -- Old hybrid-syntax workarounds may be slower once true KNN is available. See examples below.

Availability: 2.0.0 -- Weak KNN provides nearest neighbors based on geometry centroid distances instead of true distances. Exact results for points, inexact for all other types.

Exempel

Code

```
SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d, edabbr, vaabbr
FROM va2005
ORDER BY d limit 10;
```

Output

```
d         | edabbr | vaabbr
------------------+--------+--------
                0 | ALQ    | 128
 5541.57712511724 | ALQ    | 129A
 5579.67450712005 | ALQ    | 001
  6083.4207708641 | ALQ    | 131
  7691.2205404848 | ALQ    | 003
 7900.75451037313 | ALQ    | 122
 8694.20710669982 | ALQ    | 129B
 9564.24289057111 | ALQ    | 130
  12089.665931705 | ALQ    | 127
 18472.5531479404 | ALQ    | 002
(10 rows)
```

Sedan KNN rå svar:

Code

```
SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d,edabbr, vaabbr
FROM va2005
ORDER BY geom <-> 'SRID=3005;POINT(1011102 450541)'::geometry limit 10;
```

Output

```
        d         | edabbr | vaabbr
------------------+--------+--------
                0 | ALQ    | 128
 5541.57712511724 | ALQ    | 129A
 5579.67450712005 | ALQ    | 001
  6083.4207708641 | ALQ    | 131
  7691.2205404848 | ALQ    | 003
 7900.75451037313 | ALQ    | 122
 8694.20710669982 | ALQ    | 129B
 9564.24289057111 | ALQ    | 130
```

```
  12089.665931705 | ALQ    | 127
 18472.5531479404 | ALQ    | 002
(10 rows)
```

Om du kör ”EXPLAIN ANALYZEpå de två frågorna skulle du se en prestandaförbättring för den andra.

When a plan cannot produce exact distances directly, use a hybrid query to find the true nearest neighbors. First a CTE query using the index-assisted KNN, then an exact query to get correct ordering:

Code

```
WITH index_query AS (
  SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d,edabbr, vaabbr
    FROM va2005
  ORDER BY geom <-> 'SRID=3005;POINT(1011102 450541)'::geometry LIMIT 100)
  SELECT *
    FROM index_query
  ORDER BY d limit 10;
```

Output

```
        d         | edabbr | vaabbr
------------------+--------+--------
                0 | ALQ    | 128
 5541.57712511724 | ALQ    | 129A
 5579.67450712005 | ALQ    | 001
  6083.4207708641 | ALQ    | 131
  7691.2205404848 | ALQ    | 003
 7900.75451037313 | ALQ    | 122
 8694.20710669982 | ALQ    | 129B
 9564.24289057111 | ALQ    | 130
  12089.665931705 | ALQ    | 127
 18472.5531479404 | ALQ    | 002
(10 rows)
```

Se även

ST_DWithin, ST_Distance, <#>

7.10.2.2 |=|

|=| — Returnerar avståndet mellan A- och B-banorna vid deras närmaste inflygningspunkt.

Synopsis

double precision **|=|**(geometry A , geometry B);

Beskrivning

The |=| operator returns the 3D distance between two trajectories (See ST_IsValidTrajectory). This is the same as ST_DistanceCPA but as an operator it can be used for doing nearest neighbor searches using an N-dimensional index.

Notera
Denna operand använder ND GiST-index som kan finnas tillgängliga på geometrierna. Den skiljer sig från andra operatorer som använder spatiala index genom att det spatiala indexet endast används när operatorn är i ORDER BY-satsen.

Notera
Index startar bara om en av geometrierna är en konstant (inte i en underfråga/cte). t.ex. 'SRID=3005;LINESTRINGM(0 0 0,0 0 1)'::geometry istället för a.geom

Availability: 2.2.0.

Exempel

This example orders stored trajectories by their distance from a literal query trajectory.

Code

```
\set qt 'ST_AddMeasure(ST_MakeLine(ST_MakePointM(-350, 300, 0), ST_MakePointM(-410, 490, 0) ↩
    ), 10, 20)'

SELECT track_id, dist FROM (
  SELECT track_id,
    ST_DistanceCPA(tr, :qt) dist
  FROM trajectories
  ORDER BY tr |=| :qt
  LIMIT 5
) foo;
```

Output

```
track_id        dist
----------+------------------
      395 | 0.576496831518066
      380 |  5.06797130410151
      390 |  7.72262293958322
      385 |   9.8004461358071
      405 |  10.9534397988433
(5 rows)
```

Se även

ST_DistanceCPA, ST_ClosestPointOfApproach, ST_IsValidTrajectory

7.10.2.3 <#>

<#> — Returnerar 2D-avståndet mellan A- och B-begränsningsrutorna.

Synopsis

double precision **<#>**(geometry A , geometry B);

Beskrivning

The <#> operator returns distance between two floating point bounding boxes, possibly reading them from a spatial index. Useful for doing nearest neighbor **approximate** distance ordering.

Notera
Denna operand kommer att använda alla index som kan finnas tillgängliga på geometrierna. Den skiljer sig från andra operatorer som använder spatiala index genom att det spatiala indexet endast används när operatorn är i ORDER BY-satsen.

Notera
Indexet fungerar bara om en av geometrierna är en konstant, t.ex. ORDER BY (ST_GeomFromText('POINT(1 2)') <#> geom) istället för g1.geom <#>.

Availability: 2.0.0.

Exempel

Code

```
SELECT *
FROM (
SELECT b.tlid, b.mtfcc,
    b.geom <#
> ST_GeomFromText('LINESTRING(746149 2948672,745954 2948576,
        745787 2948499,745740 2948468,745712 2948438,
        745690 2948384,745677 2948319)', 2249) As b_dist,
        ST_Distance(b.geom, ST_GeomFromText('LINESTRING(746149 2948672,745954 2948576,
        745787 2948499,745740 2948468,745712 2948438,
        745690 2948384,745677 2948319)', 2249)) As act_dist
    FROM bos_roads As b
    ORDER BY b_dist, b.tlid
    LIMIT 100) As foo
    ORDER BY act_dist, tlid LIMIT 10;
```

Output

```
   tlid    | mtfcc |      b_dist      |     act_dist
-----------+-------+------------------+------------------
  85732027 | S1400 |                0 |                0
  85732029 | S1400 |                0 |                0
  85732031 | S1400 |                0 |                0
  85734335 | S1400 |                0 |                0
  85736037 | S1400 |                0 |                0
 624683742 | S1400 |                0 | 128.528874268666
  85719343 | S1400 | 260.839270432962 | 260.839270432962
  85741826 | S1400 | 164.759294123275 | 260.839270432962
  85732032 | S1400 |           277.75 | 311.830282365264
  85735592 | S1400 |           222.25 | 311.830282365264
(10 rows)
```

Se även

ST_DWithin, ST_Distance, <->

7.10.2.4 <<->>

<<->> — Returnerar n-D-avståndet mellan geometrierna eller begränsningsrutorna A och B

Synopsis

double precision **<<->>**(geometry A , geometry B);

Beskrivning

Operatorn <<->> returnerar n-D (euklidiskt) avstånd mellan mittpunkterna i de avgränsande boxarna för två geometrier. Användbar för att göra närmaste granne **ungefärlig** avståndsordning.

Notera
Denna operand använder n-D GiST-index som kan finnas tillgängliga för geometrierna. Den skiljer sig från andra operatorer som använder spatiala index genom att det spatiala indexet endast används när operatorn är i ORDER BY-satsen.

Notera
Indexet börjar bara gälla om en av geometrierna är en konstant (inte i en underfråga). t.ex. 'SRID=3005;POINT(1011102 450541)'::geometry istället för a.geom

Availability: 2.2.0.

Se även

<->

7.11 Spatiala relationer

7.11.1 Topologiska relationer

7.11.1.1 ST_3DIntersects

ST_3DIntersects — Testar om två geometrier korsar varandra spatialt i 3D - endast för punkter, linestrings, polygoner, polyedrisk yta (area)

Synopsis

boolean **ST_3DIntersects**(geometry geomA , geometry geomB);

Beskrivning

Overlaps, Touches, Within innebär alla spatial skärning. Om någon av de ovan nämnda returnerar true, så skär geometrierna varandra även spatialt. Disjoint implicerar false för spatial intersektion.

Notera
Denna funktion inkluderar automatiskt en jämförelse av begränsningsrutor som använder alla spatiala index som finns tillgängliga för geometrierna.

Notera
På grund av brister i den flytande robustheten korsar geometrier inte alltid varandra på det sätt som man förväntar sig efter geometrisk bearbetning. Till exempel kanske den närmaste punkten på en linestrings till en geometri inte ligger på linestringsen. För den här typen av problem, där ett avstånd på en centimeter bara ska betraktas som en skärningspunkt, använder du ST_3DDWithin.

Ändrad: 3.0.0 SFCGAL backend borttagen, GEOS backend stöder TINs.

Tillgänglighet: 2.0.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1

Exempel på geometri

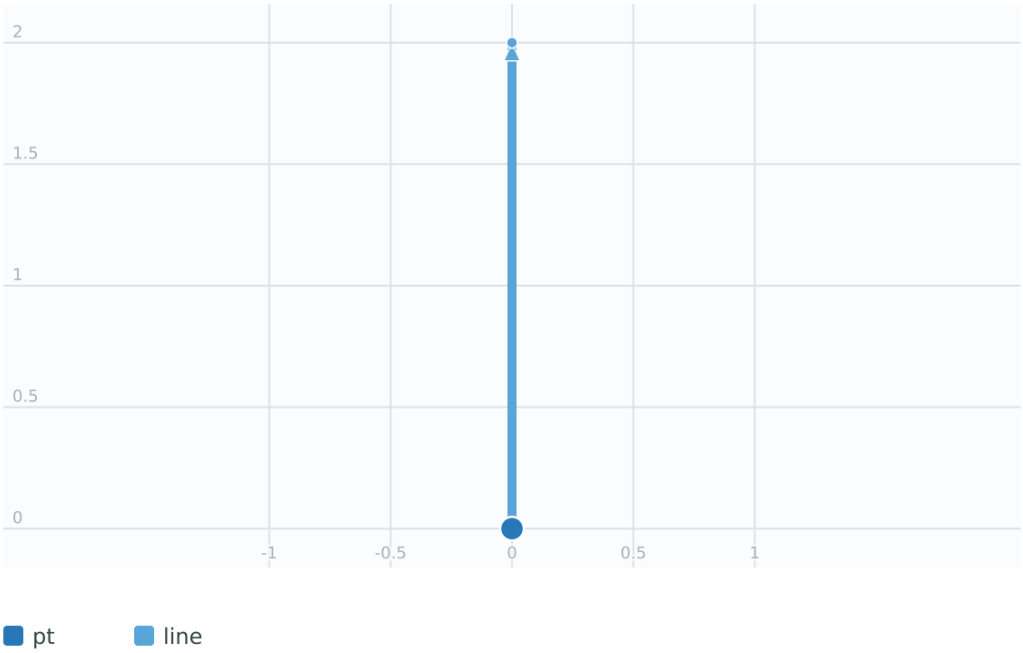
Code

```
SELECT ST_3DIntersects(pt, line), ST_Intersects(pt, line)
FROM (SELECT 'POINT(0 0 2)'::geometry As pt, 'LINESTRING (0 0 1,0 2 3)'::geometry As line)  ↩
    As foo;
```

Output

```
st_3dintersects | st_intersects
----------------+---------------
 f              | t
(1 row)
```

Figure



Exempel på TIN

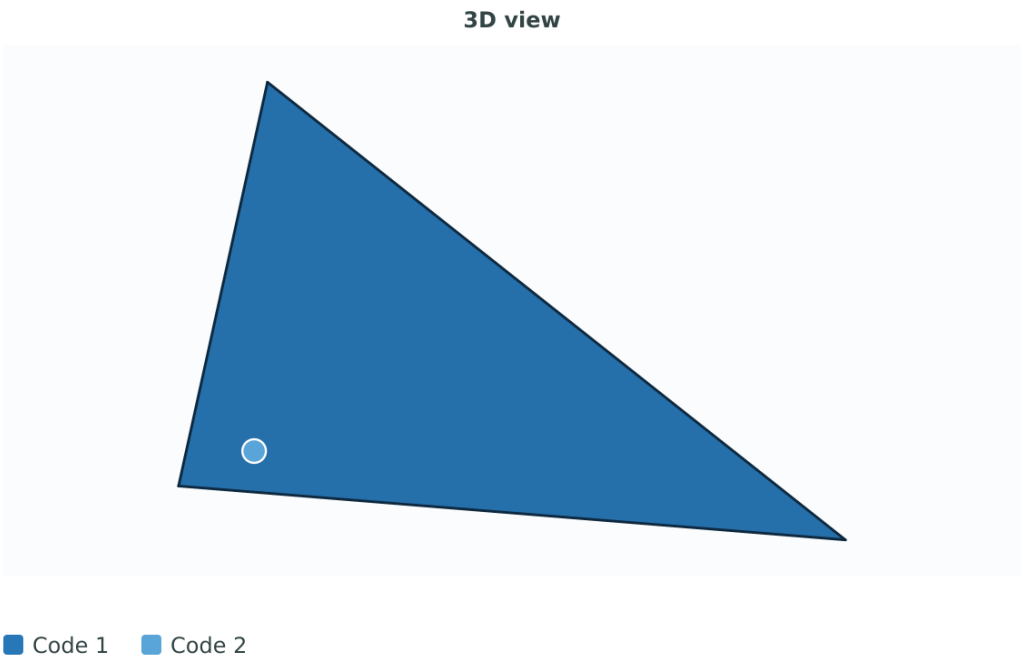
Code

```
SELECT ST_3DIntersects('TIN(((0 0 0,1 0 0,0 1 0,0 0 0)))'::geometry, 'POINT(.1 .1 0)':: ↩
    geometry);
```

Output

```
t
```

Figure



Se även

ST_3DDWithin, ST_Intersects

7.11.1.2 ST_Contains

ST_Contains — Testar om varje punkt i B ligger i A, och deras interiörer har en gemensam punkt

Synopsis

boolean **ST_Contains**(geometry geomA, geometry geomB);

Beskrivning

Returnerar TRUE om geometri A innehåller geometri B. A innehåller B om och endast om alla punkter i B ligger inuti (dvs. i det inre eller på gränsen till) A (eller motsvarande, inga punkter i B ligger i det yttre av A), och det inre av A och B har minst en punkt gemensamt.

I matematiska termer: *ST_Contains(A, B) ⇔ (A ∩ B = B) ∧ (Int(A) ∩ Int(B) ≠ ∅)*

Förhållandet contains är reflexivt: varje geometri innehåller sig själv. (I predikatet ST_ContainsProperly innehåller däremot en geometri *inte* sig själv på rätt sätt.) Relationen är antisymmetrisk: om `ST_Contains(A,B) = true` och `ST_Contains(B,A) = true`, måste de två geometrierna vara topologiskt lika (`ST_Equals(A,B) = true`).

ST_Contains är motsatsen till ST_Within. Så `ST_Contains(A,B) = ST_Within(B,A)`..

Notera
Eftersom interiörerna måste ha en gemensam punkt är en finess i definitionen att polygoner och linjer *inte* innehåller linjer och punkter som ligger helt inom deras gräns. För ytterligare information se Subtiliteter i OGC Covers, Contains, Within. Predikatet ST_Covers ger en mer inkluderande relation.

Notera
&index_aware; För att undvika indexanvändning, använd funktionen `_ST_Contains`.

Utförs av GEOS-modulen

Förbättrad: 2.3.0 Förbättring av PIP-kortslutning utökad till att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon.

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Viktigt
Använd inte denna funktion med ogiltiga geometrier. Du kommer att få oväntade resultat.

OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2 // s2.1.13.3 - samma som inom(geometri B, geometri A)

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.31

Exempel

`ST_Contains` returnerar `TRUE` i följande situationer:

A LineString contains a MultiPoint whose points lie in its interior or boundary.

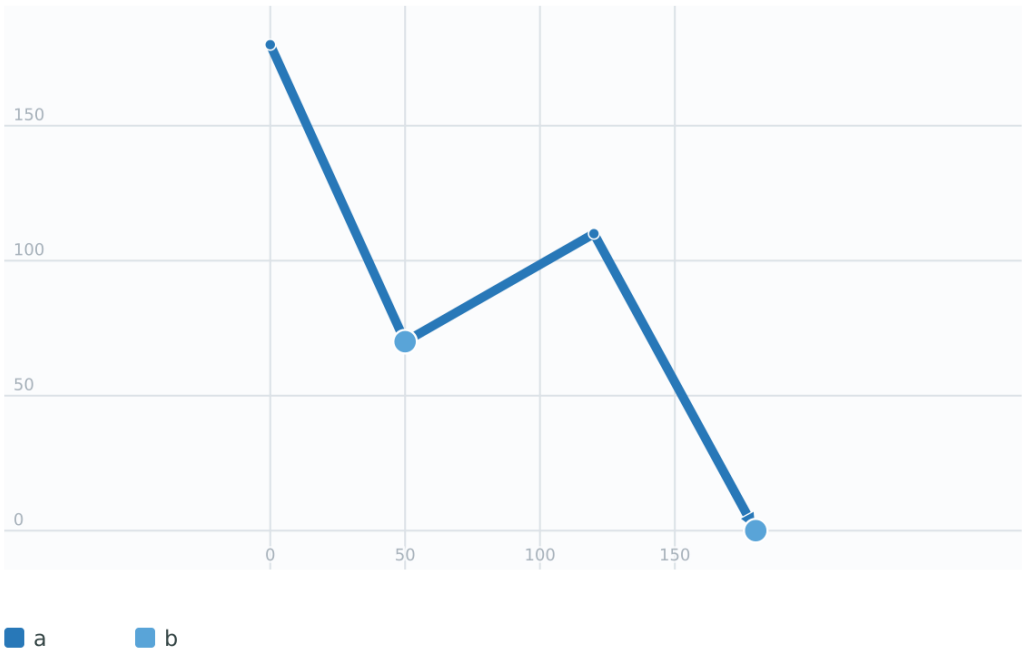
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'MULTIPOINT (60 80,190 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon contains a Point in its interior.

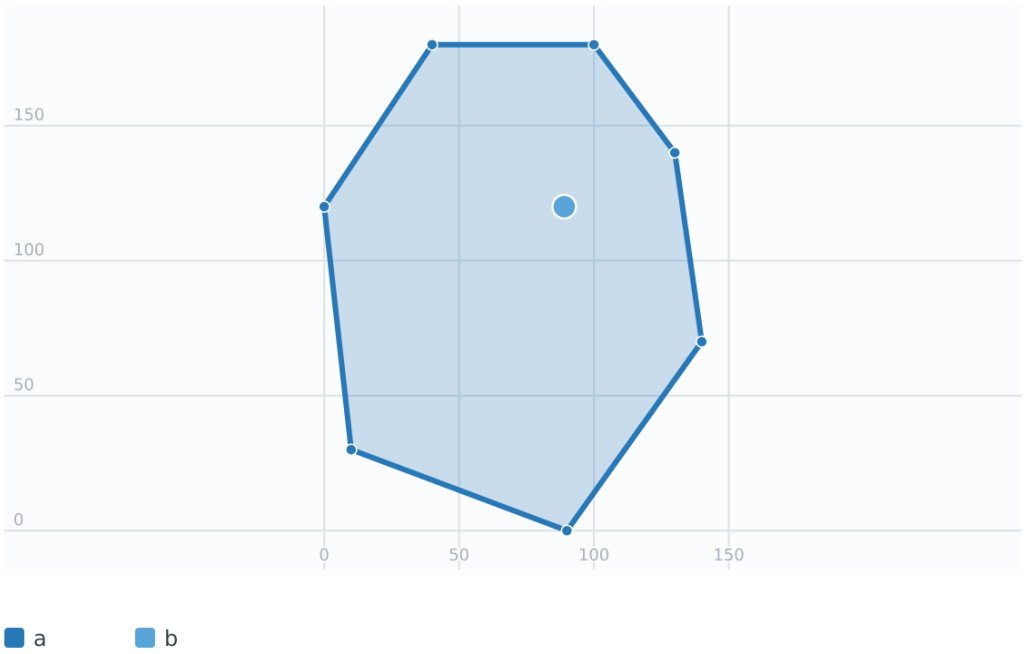
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'POINT (99 130)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon contains a LineString with interior points inside the Polygon.

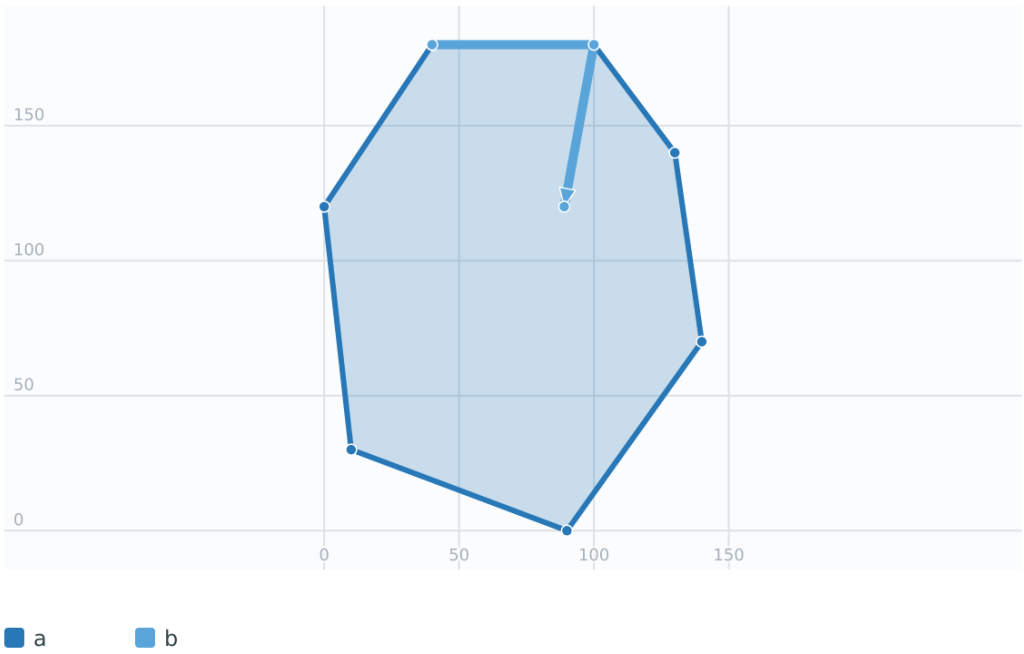
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'LINESTRING (50 190,110 190,99 130)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon contains another Polygon in its interior.

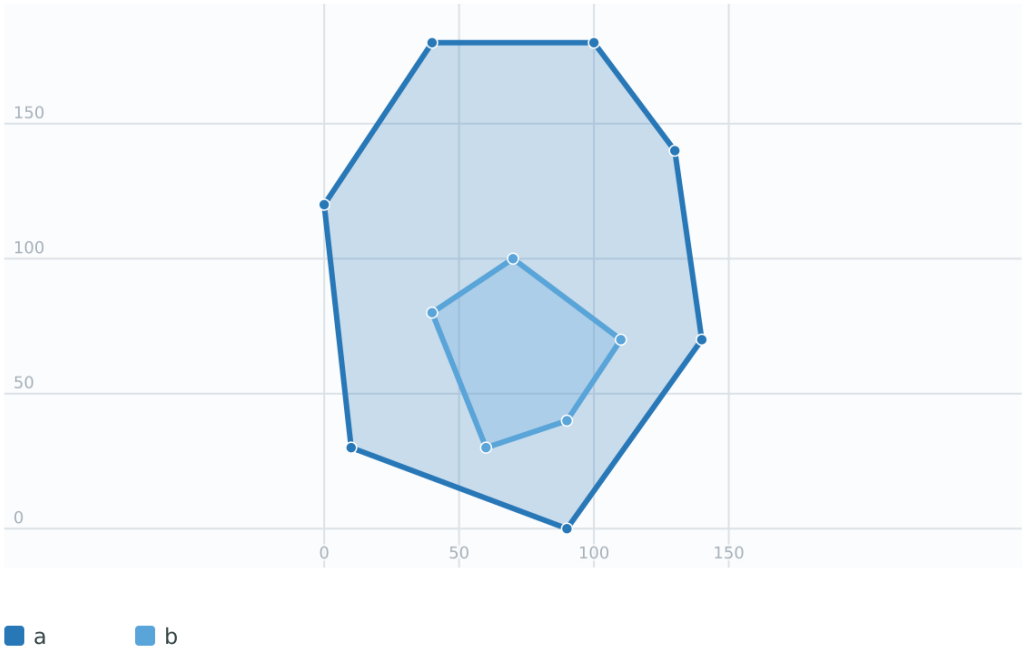
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'POLYGON ((70 40,100 50,120 80,80 110,50 90,70 40))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



`ST_Contains` returnerar `FALSE` i följande situationer:

A Polygon does not contain a MultiPoint when one point lies in a hole.

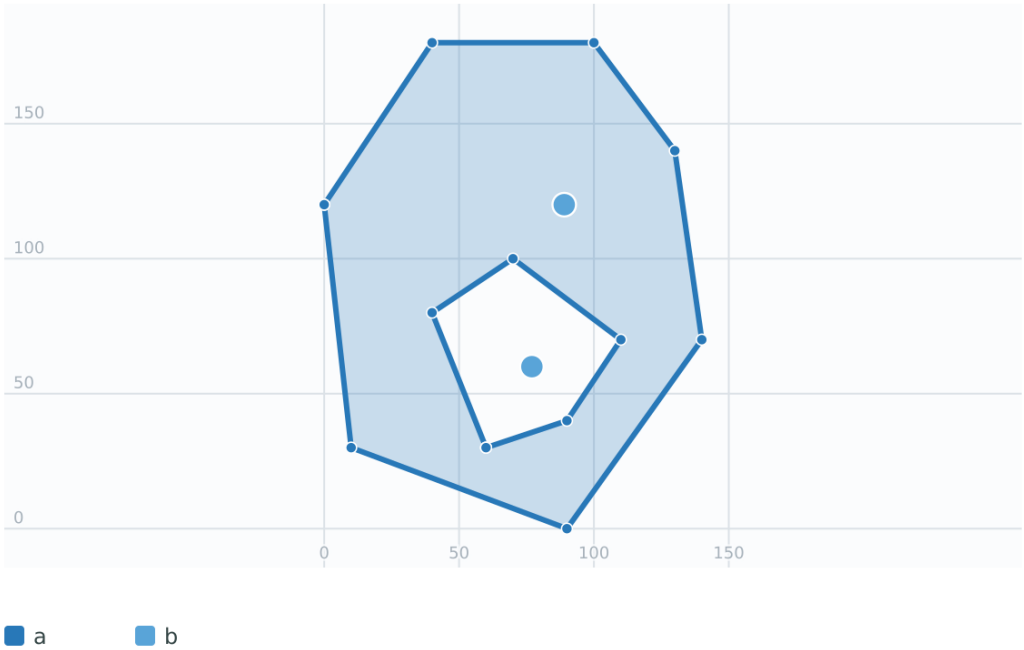
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130),(70 40,100 50,120 ↩
    80,80 110,50 90,70 40))'::geometry AS a,
  'MULTIPOINT (99 130,87 70)'::geometry AS b
) AS example;
```

Output

```
f
```

Figure



A Polygon does not contain a LineString extending outside it.

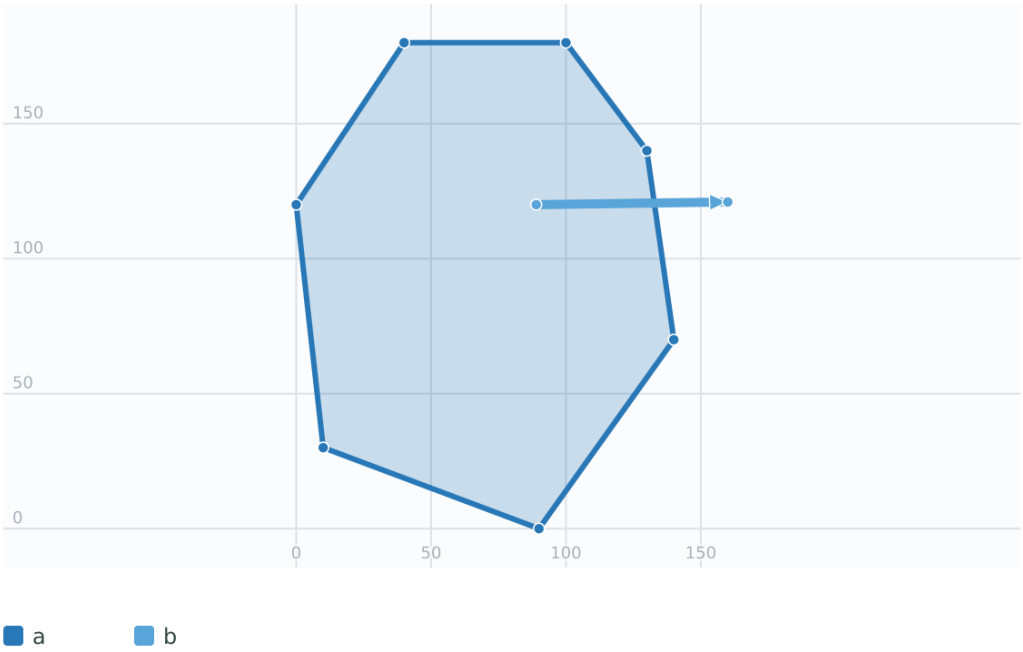
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'LINESTRING (99 130,170 131)'::geometry AS b
) AS example;
```

Output

```
f
```

Figure



På grund av villkoret för inre skärning returnerar `ST_Contains` `FALSE` i följande situationer (medan `ST_Covers` returnerar `TRUE`):

A LineString does not contain a Point lying only on its boundary.

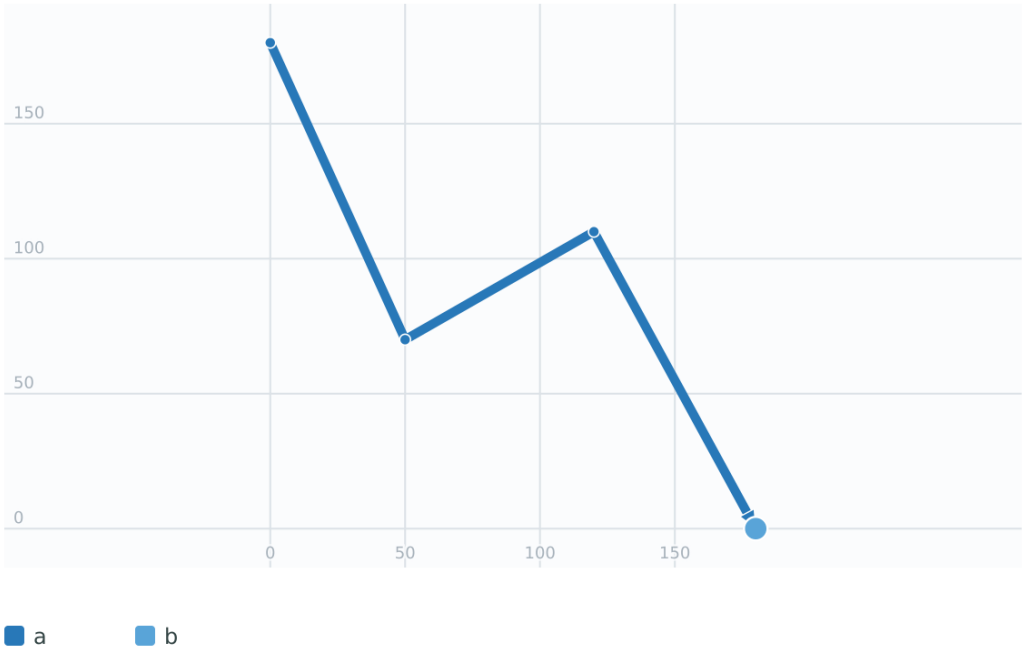
Code

```
SELECT ST_Contains(a, b), ST_Covers(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'POINT (190 10)'::geometry AS b
) AS example;
```

Output

```
f | t
```

Figure



A Polygon does not contain a LineString lying entirely on its boundary.

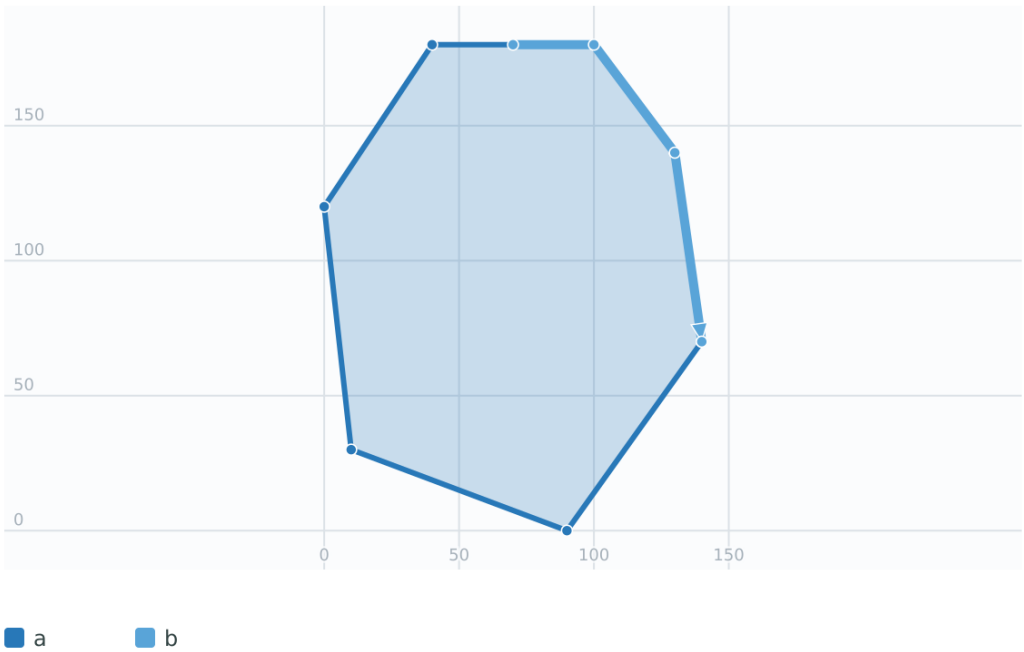
Code

```
SELECT ST_Contains(a, b), ST_Covers(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'LINESTRING (80 190,110 190,140 150,150 80)'::geometry AS b
) AS example;
```

Output

```
f | t
```

Figure



This example tests a circle within another circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_Contains(smallc, bigc) AS smallcontainsbig,
  ST_Contains(bigc, smallc) AS bigcontainssmall,
  ST_Contains(bigc, ST_Union(smallc, bigc)) AS bigcontainsunion,
  ST_Equals(bigc, ST_Union(smallc, bigc)) AS bigisunion,
  ST_Covers(bigc, ST_ExteriorRing(bigc)) AS bigcoversexterior,
  ST_Contains(bigc, ST_ExteriorRing(bigc)) AS bigcontainsexterior
FROM circles;
```

Output

```
-[ RECORD 1 ]-----
smallcontainsbig   | f
bigcontainssmall   | t
bigcontainsunion   | t
bigisunion         | t
bigcoversexterior  | t
bigcontainsexterior | f
```

Figure

This example demonstrates the difference between contains and contains-properly predicates.

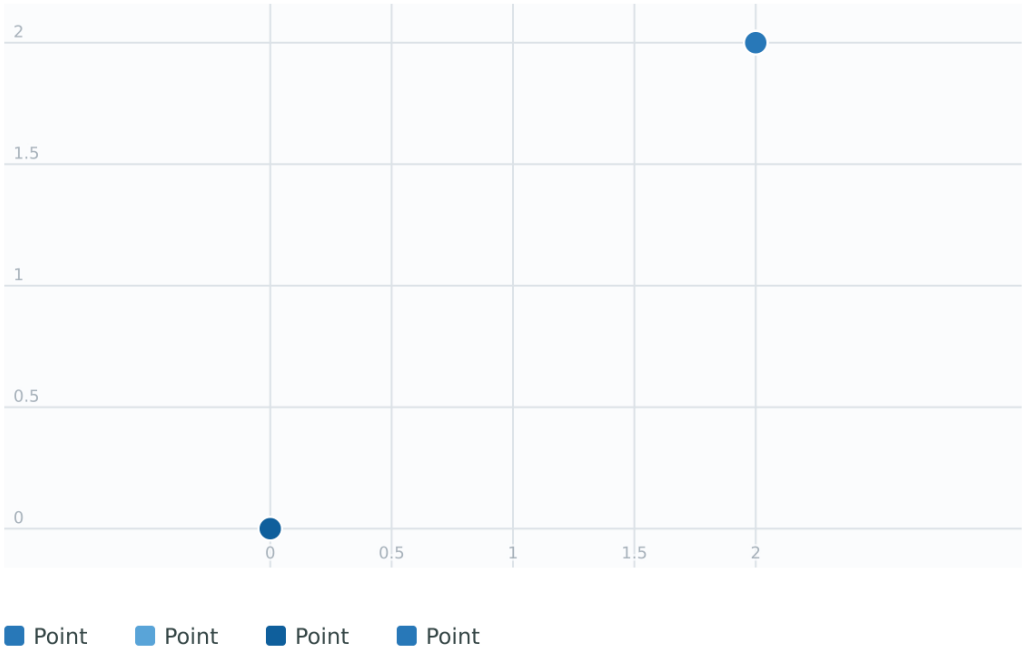
Code

```
SELECT
  ST_GeometryType(geom_a) AS geomtype,
  ST_Contains(geom_a, geom_a) AS acontainsa,
  ST_ContainsProperly(geom_a, geom_a) AS acontainspropa,
  ST_Contains(geom_a, ST_Boundary(geom_a)) AS acontainsba,
  ST_ContainsProperly(geom_a, ST_Boundary(geom_a)) AS acontainspropba
FROM (VALUES
    (ST_Buffer(ST_Point(1, 1), 5, 1)),
    (ST_MakeLine(ST_Point(1, 1), ST_Point(-1, -1))),
    (ST_Point(1, 1))
  ) AS foo(geom_a);
```

Output

```
geomtype    | acontainsa | acontainspropa | acontainsba | acontainspropba
------------+------------+----------------+-------------+-----------------
ST_Polygon    | t          | f              | f           | f
ST_LineString | t          | f              | f           | f
ST_Point      | t          | t              | f           | f
```

Figure



Se även

ST_Boundary, ST_ContainsProperly, ST_Covers, ST_CoveredBy, ST_Equals, ST_Within

7.11.1.3 ST_ContainsProperly

ST_ContainsProperly — Testar om varje punkt i B ligger i det inre av A

Synopsis

boolean **ST_ContainsProperly**(geometry geomA, geometry geomB);

Beskrivning

Returnerar `sant` om varje punkt i B ligger i A:s inre (eller motsvarande, ingen punkt i B ligger i A:s gräns eller yttre).

I matematiska termer: *ST_ContainsProperly(A, B) ⇔ Int(A) ∩ B = B*

A innehåller B på rätt sätt om DE-9IM-överskärningsmatrisen för de två geometrierna stämmer överens med [T**FF*FF*]

A innehåller inte riktigt sig själv, men innehåller sig själv.

Ett användningsområde för detta predikat är att beräkna skärningspunkterna mellan en uppsättning geometrier och en stor polygonal geometri. Eftersom intersektion är en ganska långsam operation kan det vara effektivare att använda containsProperly för att filtrera bort testgeometrier som ligger helt inom området. I dessa fall vet man på förhand att skärningspunkten är exakt den ursprungliga testgeometrin.

Notera
&index_aware; För att undvika indexanvändning, använd funktionen `_ST_ContainsProperly`.

Notera
Fördelen med detta predikat jämfört med ST_Contains och ST_Intersects är att det kan beräknas mer effektivt, utan att behöva beräkna topologi på enskilda punkter.

Utförs av GEOS-modulen.

Tillgänglighet: 1.4.0

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Viktigt
Använd inte denna funktion med ogiltiga geometrier. Du kommer att få oväntade resultat.

Exempel

This example tests a circle within another circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_ContainsProperly(smallc, bigc) AS smallcontainspropbig,
  ST_ContainsProperly(bigc, smallc) AS bigcontainspropsmall,
  ST_ContainsProperly(bigc, ST_Union(smallc, bigc)) AS bigcontainspropunion,
  ST_Equals(bigc, ST_Union(smallc, bigc)) AS bigisunion,
  ST_Covers(bigc, ST_ExteriorRing(bigc)) AS bigcoversexterior,
  ST_ContainsProperly(bigc, ST_ExteriorRing(bigc)) AS bigcontainsexterior
FROM circles;
```

Output

```
-[ RECORD 1 ]-----
smallcontainspropbig   | f
bigcontainspropsmall   | t
bigcontainspropunion   | f
bigisunion             | t
bigcoversexterior      | t
bigcontainsexterior    | f
```

Figure

This example demonstrates the difference between contains and contains-properly predicates.

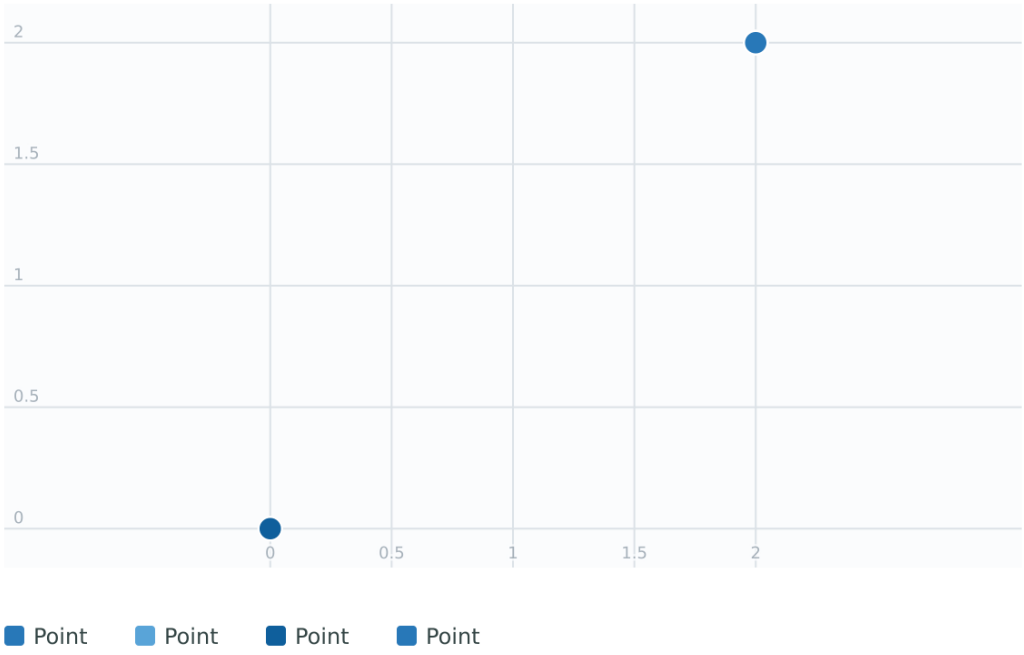
Code

```
WITH input(geom) AS (VALUES
  (ST_Buffer(ST_Point(1, 1), 5, 1)),
  (ST_MakeLine(ST_Point(1, 1), ST_Point(-1, -1))),
  (ST_Point(1, 1))
)
SELECT ST_GeometryType(geom) AS geomtype,
       ST_Contains(geom, geom) AS acontainsa,
       ST_ContainsProperly(geom, geom) AS acontainspropa,
       ST_Contains(geom, ST_Boundary(geom)) AS acontainsba,
       ST_ContainsProperly(geom, ST_Boundary(geom)) AS acontainspropba
FROM input;
```

Output

```
geomtype    | acontainsa | acontainspropa | acontainsba | acontainspropba
------------+------------+----------------+-------------+-----------------
ST_Polygon    | t          | f              | f           | f
ST_LineString | t          | f              | f           | f
ST_Point      | t          | t              | f           | f
```

Figure



Se även

ST_GeometryType, ST_Boundary, ST_Contains, ST_Covers, ST_CoveredBy, ST_Equals, ST_Relate, ST_Within

7.11.1.4 ST_CoveredBy

ST_CoveredBy — Testar om varje punkt i A ligger i B

Synopsis

boolean **ST_CoveredBy**(geometry geomA, geometry geomB);
boolean **ST_CoveredBy**(geography geogA, geography geogB);

Beskrivning

Returnerar `true` om varje punkt i Geometry/Geography A ligger inuti (dvs. skär insidan eller gränsen av) Geometry/Geography B. Testar på motsvarande sätt att ingen punkt i A ligger utanför (i utsidan av) B.

I matematiska termer: *ST_CoveredBy(A, B)* ⇔ *A* ▯ *B* = *A*

ST_CoveredBy är det omvända av ST_Covers. Så `ST_CoveredBy(A,B) = ST_Covers(B,A)`..

Generellt bör denna funktion användas istället för ST_Within, eftersom den har en enklare definition som inte har den egenheten att gränser inte ligger inom sin geometri".

Notera
&index_aware; För att undvika indexanvändning, använd funktionen _ST_CoveredBy.

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Viktigt
Använd inte denna funktion med ogiltiga geometrier. Du kommer att få oväntade resultat.

Utförs av GEOS-modulen

Tillgänglighet: 1.2.2

OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Inte en OGC-standard, men Oracle har den också.

Exempel

This example tests a circle covered by another circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_CoveredBy(smallc, smallc) AS smallinsmall,
  ST_CoveredBy(smallc, bigc) AS smallcoveredbybig,
  ST_CoveredBy(ST_ExteriorRing(bigc), bigc) AS exteriorcoveredbybig,
  ST_Within(ST_ExteriorRing(bigc), bigc) AS exteriorwithinbig
FROM circles;
```

Output

```
smallinsmall | smallcoveredbybig | exteriorcoveredbybig | exteriorwithinbig
-------------+-------------------+----------------------+------------------
 t           | t                 | t                    | f
(1 row)
```

Figure

Se även

ST_Contains, ST_Covers, ST_ExteriorRing, ST_Within

7.11.1.5 ST_Covers

ST_Covers — Testar om varje punkt i B ligger i A

Synopsis

boolean **ST_Covers**(geometry geomA, geometry geomB);
boolean **ST_Covers**(geography geogpolyA, geography geogpointB);

Beskrivning

Returnerar `true` om varje punkt i Geometry/Geography B ligger inuti (dvs. skär insidan eller gränsen av) Geometry/Geography A. Testar på motsvarande sätt att ingen punkt i B ligger utanför (i utsidan av) A.

I matematiska termer: *ST_Covers(A, B)* ⇔ *A* ∩ *B* = *B*

ST_Covers är det omvända av ST_CoveredBy. Alltså, ST_Covers `(A,B) = ST_CoveredBy(B,A)`..

Generellt bör denna funktion användas istället för ST_Contains, eftersom den har en enklare definition som inte har den egenheten att geometrier inte innehåller sin gräns".

Notera
&index_aware; För att undvika indexanvändning, använd funktionen `_ST_Covers`.

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Viktigt
Använd inte denna funktion med ogiltiga geometrier. Du kommer att få oväntade resultat.

Utförs av GEOS-modulen

Förbättrad: 2.4.0 Stöd för polygon i polygon och linje i polygon har lagts till för geograftyp

Förbättrad: 2.3.0 Förbättring av PIP-kortslutning för geometri utökad för att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon.

Tillgänglighet: 1.5 - stöd för geografi infördes.

Tillgänglighet: 1.2.2

OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Inte en OGC-standard, men Oracle har den också.

Exempel

Exempel på geometri

This example tests a circle covering another circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_Covers(smallc, smallc) AS smallinsmall,
  ST_Covers(smallc, bigc) AS smallcoversbig,
  ST_Covers(bigc, ST_ExteriorRing(bigc)) AS bigcoversexterior,
  ST_Contains(bigc, ST_ExteriorRing(bigc)) AS bigcontainsexterior
FROM circles;
```

Output

```
smallinsmall | smallcoversbig | bigcoversexterior | bigcontainsexterior
-------------+----------------+-------------------+--------------------
 t           | f              | t                 | f
(1 row)
```

Figure

Geography example

This example compares a point with a 300-meter buffer against a point and against a 10-meter buffer around that point.

Code

```
SELECT ST_Covers(geog_poly, geog_pt) As poly_covers_pt,
  ST_Covers(ST_Buffer(geog_pt, 10), geog_pt) As buff_10m_covers_cent
  FROM (SELECT ST_Buffer('SRID=4326;POINT(-99.327 31.4821)'::geography, 300) As geog_poly,
        'SRID=4326;POINT(-99.33 31.483)'::geography As geog_pt ) As foo;
```

Output

```
poly_covers_pt | buff_10m_covers_cent
---------------+---------------------
 f             | t
```

Se även

ST_Contains, ST_CoveredBy, ST_Within

7.11.1.6 ST_Crosses

ST_Crosses — Testar om två geometrier har vissa, men inte alla, inre punkter gemensamt

Synopsis

boolean **ST_Crosses**(geometry g1, geometry g2);

Beskrivning

Jämför två geometriobjekt och returnerar `true` om deras skärningspunkt ”korsar varandra spatialtt”, dvs. geometrierna har vissa men inte alla inre punkter gemensamma. Skärningspunkten mellan geometriernas inre punkter måste vara icke-tom och ha en dimension som är mindre än den maximala dimensionen för de två geometrierna i indata, och skärningspunkten mellan de två geometrierna får inte vara lika med någon av geometrierna. I annat fall returneras `false`. Crosses-relationen är symmetrisk och irreflexiv.

I matematiska termer: *ST_Crosses(A, B) ⇔ (dim(Int(A) ∩ Int(B)) < max(dim(Int(A)), dim(Int(B)))) ∧ (A ∩ B ≠ A) ∧ (A ∩ B ≠ B)*

Geometrier korsas om deras DE-9IM Intersection Matrix matchar:

- `T*T******` för situationerna Point/Line, Point/Area och Line/Area
- `T*****T**` för Line/Point-, Area/Point- och Area/Line-situationer
- `0********` för Line/Line-situationer
- resultatet är `falskt` för Point/Point- och Area/Area-situationer

Notera
OpenGIS Simple Features Specification definierar detta predikat endast för situationerna Point/Line, Point/Area, Line/Line och Line/Area. JTS / GEOS utökar definitionen till att även gälla Line/Point, Area/Point och Area/Line situationer. Detta gör relationen symmetrisk.

Notera
Denna funktion inkluderar automatiskt en jämförelse av begränsningsrutor som använder alla spatiala index som finns tillgängliga för geometrierna.

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.13.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.29

Exempel

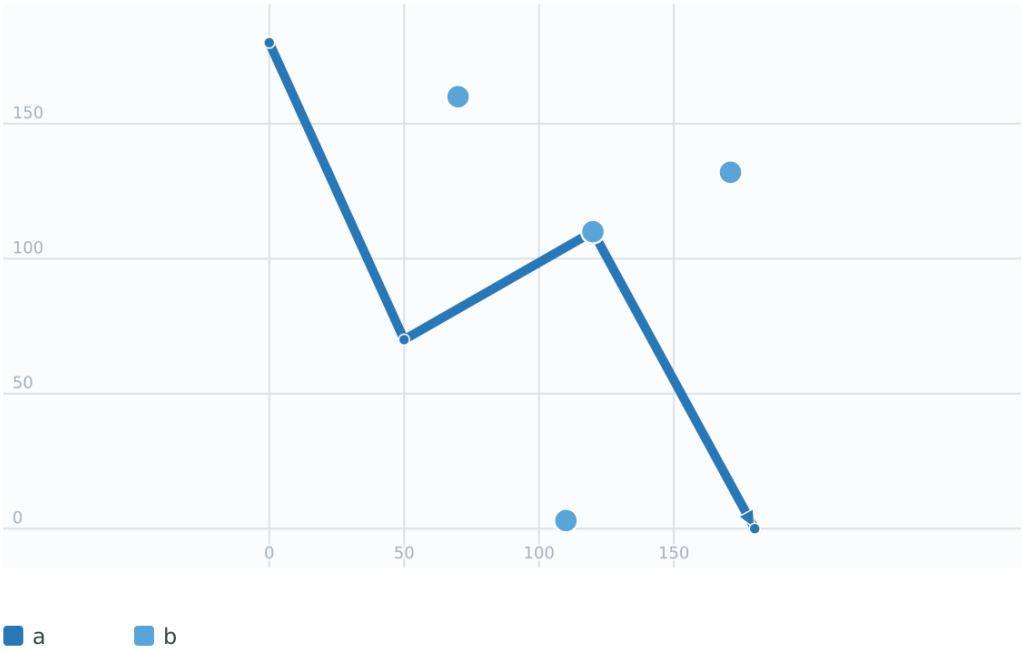
Följande situationer returnerar alla `true`.

A LineString crosses a MultiPoint when some, but not all, points lie on the line.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'MULTIPOINT (80 170,120 13,130 120,181 142)'::geometry AS b
) AS example;
```

Figure

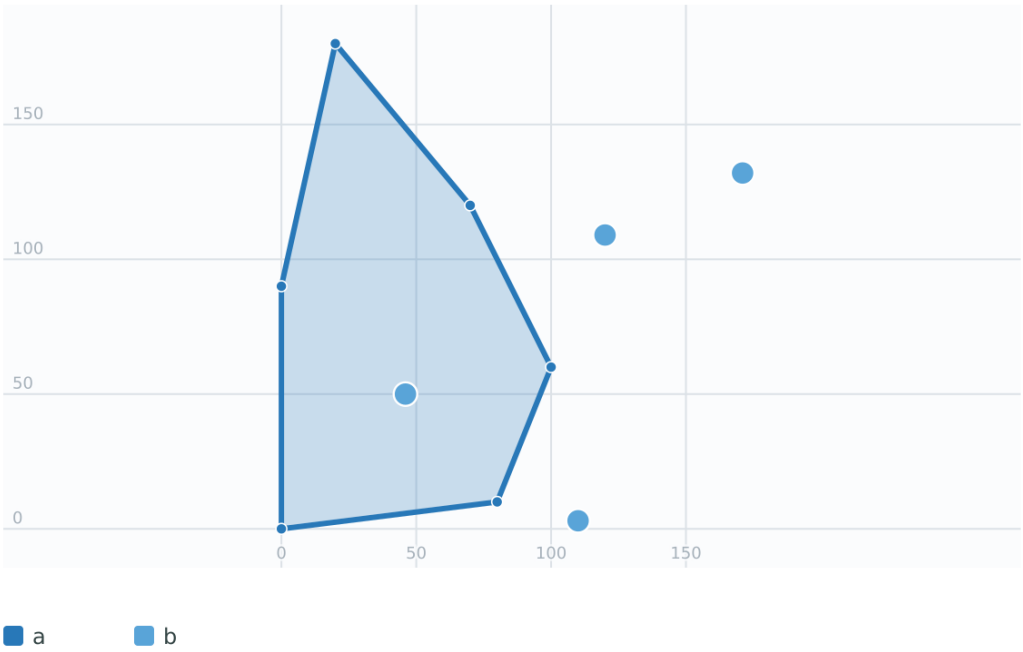


A Polygon crosses a MultiPoint when some, but not all, points lie in its interior.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'POLYGON ((30 190,10 100,10 10,90 20,110 70,80 130,30 190))'::geometry AS a,
  'MULTIPOINT (56 60,120 13,130 119,181 142)'::geometry AS b
) AS example;
```

Figure



A LineString crosses a Polygon when part of the line lies inside and part outside.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'POLYGON ((30 190,10 100,10 10,90 20,110 70,80 130,30 190))'::geometry AS a,
  'LINESTRING (30 40,70 50,120 150,190 190)'::geometry AS b
) AS example;
```

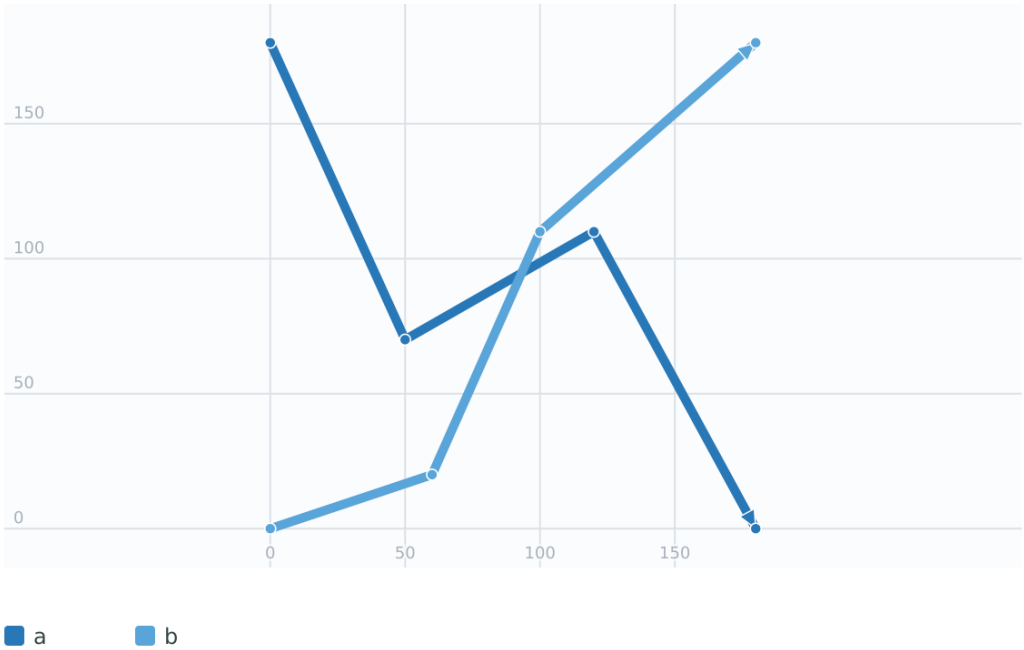
Figure

Two LineStrings cross when they share an interior point.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'LINESTRING (10 10,70 30,110 120,190 190)'::geometry AS b
) AS example;
```

Figure



Tänk dig en situation där en användare har två tabeller: en tabell med vägar och en tabell med motorvägar.

Code

```
CREATE TABLE roads (
  id serial NOT NULL,
  geom geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↩
    road_id)
);
```

Code

```
CREATE TABLE highways (
  id serial NOT NULL,
  the_gem geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↩
    road_id)
);
```

För att få fram en lista över vägar som korsar en motorväg, använd en fråga som liknar:

Code

```
SELECT roads.id
FROM roads, highways
WHERE ST_Crosses(roads.geom, highways.geom);
```

Se även

ST_Contains, ST_Overlaps

7.11.1.7 ST_Disjoint

ST_Disjoint — Testar om två geometrier inte har några gemensamma punkter

Synopsis

boolean **ST_Disjoint**(geometry A , geometry B);

Beskrivning

Returnerar `sant` om två geometrier är disjunkta. Geometrier är disjunkta om de inte har någon gemensam punkt.

Om något annat spatialt förhållande är sant för ett par geometrier är de inte disjunkta. Disjunkt innebär att ST_Intersects är falskt.

I matematiska termer: *ST_Disjoint(A, B)* ⇔ *A* ∩ *B* = ∅

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Utförs av GEOS-modulen

Notera
Detta funktionsanrop använder inte index. Ett negerat ST_Intersects -predikat kan användas som ett mer performant alternativ som använder index: `ST_Disjoint(A,B) = NOT ST_Intersects(A,B)`

Notera
OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2 //s2.1.13.3 - a.Relate(b, ’FF*FF****’)

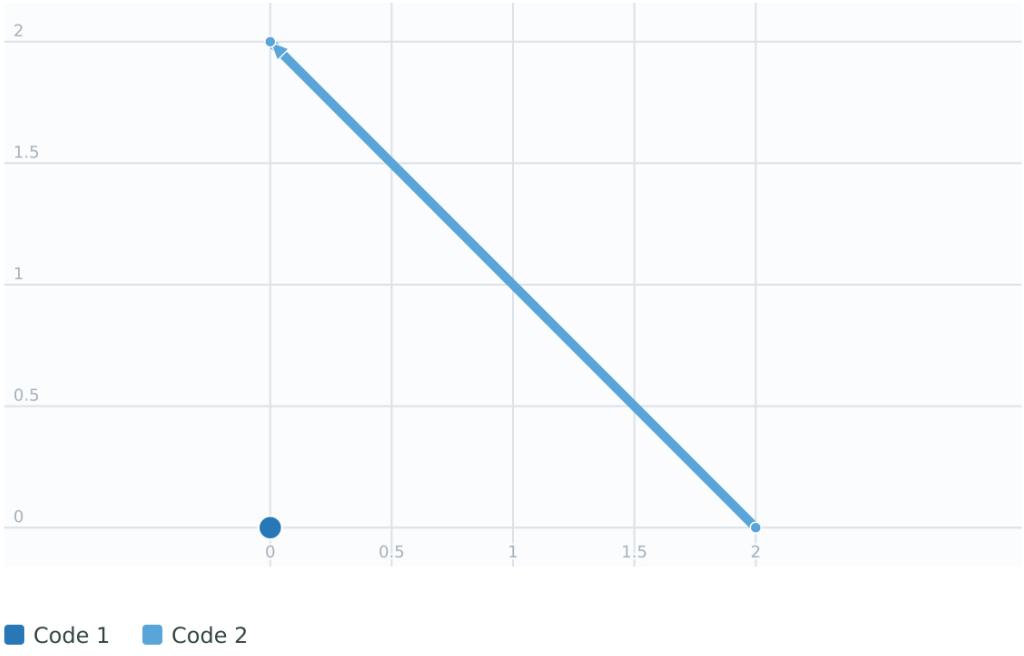
Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.26

Exempel

Code

```
SELECT ST_Disjoint('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

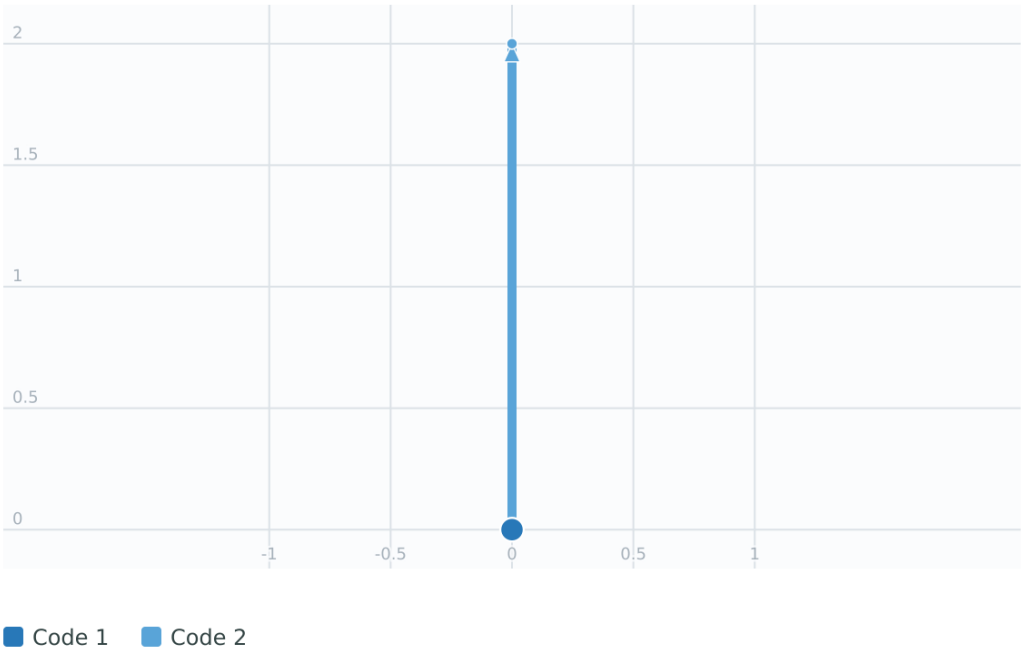
Figure



Code

```
SELECT ST_Disjoint('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Figure



Se även

ST_Intersects

7.11.1.8 ST_Equals

ST_Equals — Testar om två geometrier innehåller samma uppsättning punkter

Synopsis

boolean **ST_Equals**(geometry A, geometry B);

Beskrivning

Returnerar `sant` om de givna geometrierna är topologiskt lika". Använd detta för ett bättresvar än -". Topologisk likhet innebär att geometrierna har samma dimension och att deras punktuppsättningar upptar samma utrymme. Detta innebär att ordningen på hörnen kan vara olika i topologiskt lika geometrier. För att verifiera att punktordningen är konsekvent använd ST_OrderingEquals (det måste noteras att ST_OrderingEquals är lite strängare än att bara verifiera att punktordningen är densamma).

I matematiska termer: *ST_Equals(A, B)* ⇔ *A = B*

Följande relation gäller: *ST_Equals(A, B)* ⇔ *ST_Within(A,B)* ∧ *ST_Within(B,A)*

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.24

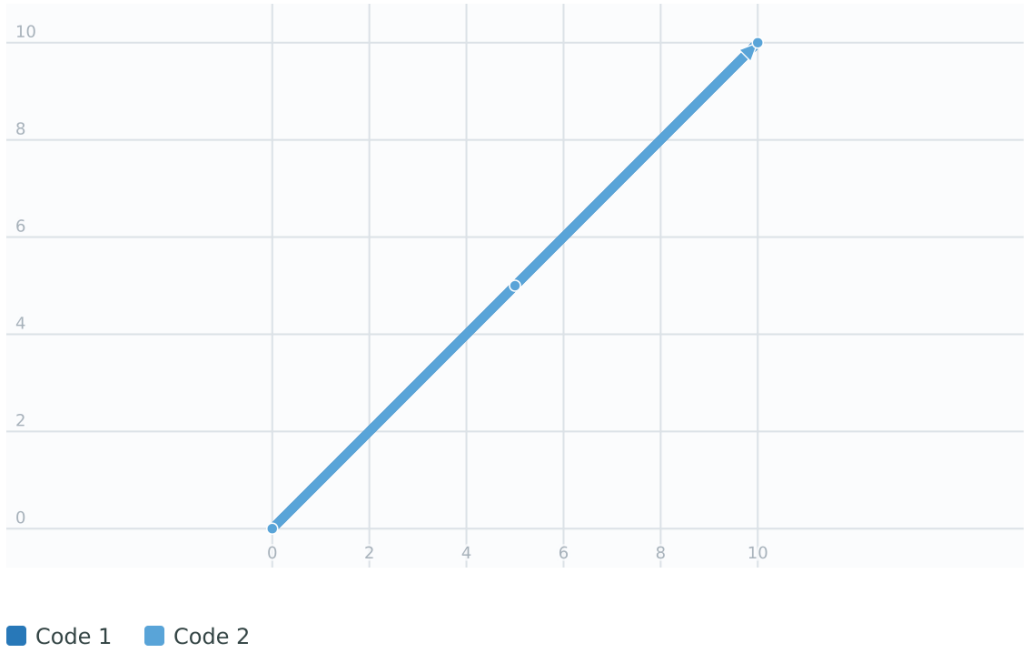
Ändrad: 2.2.0 Returnerar sant även för ogiltiga geometrier om de är binärt lika

Exempel

Code

```
SELECT ST_Equals('LINESTRING(0 0,10 10)',
'LINESTRING(0 0,5 5,10 10)');
```

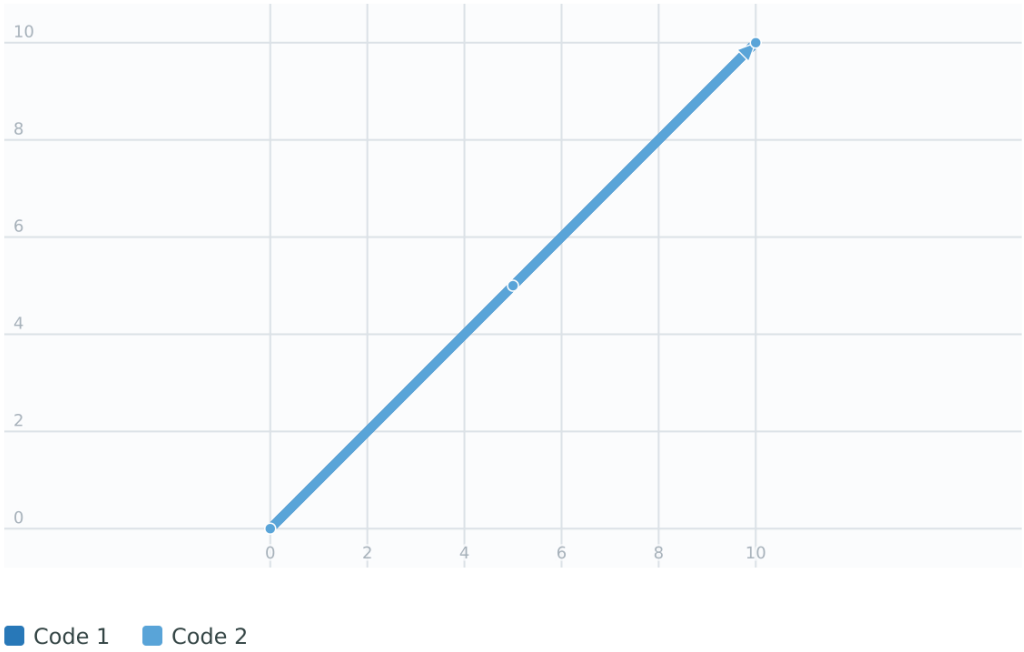
Figure



Code

```
SELECT ST_Equals(ST_Reverse('LINESTRING(0 0,10 10)'),
'LINESTRING(0 0,5 5,10 10)');
```

Figure



Se även

ST_IsValid, ST_OrderingEquals, ST_Reverse, ST_Within

7.11.1.9 ST_Intersects

ST_Intersects — Testar om två geometrier skär varandra (de har minst en gemensam punkt)

Synopsis

boolean **ST_Intersects**(geometry geomA , geometry geomB);
boolean **ST_Intersects**(geography geogA , geography geogB);

Beskrivning

Returnerar `true` om två geometrier korsar varandra. Geometrier korsar varandra om de har någon gemensam punkt.

För geografi används en avståndstolerans på 0,00001 meter (så att punkter som ligger mycket nära varandra anses korsa varandra).

I matematiska termer: *ST_Intersects(A, B)* ⇔ *A* ∩ *B* ≠ ∅

Geometrier skär varandra om deras DE-9IM Intersection Matrix matchar en av följande:

- T********
- *T*******
- ***T*****

- ****T****

Spatial korsning är underförstådd i alla andra tester av spatiala relationer, utom ST_Disjoint, som testar att geometrier INTE korsar varandra.

Notera
Denna funktion inkluderar automatiskt en jämförelse av begränsningsrutor som använder alla spatiala index som finns tillgängliga för geometrierna.

Ändrad: 3.0.0 SFCGAL-versionen har tagits bort och inbyggt stöd för 2D TINS har lagts till.

Förbättrad: 2.5.0 Stöder GEOMETRYCOLLECTION.

Förbättrad: 2.3.0 Förbättring av PIP-kortslutning utökad till att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon.

Utförs av GEOS-modulen (för geometri), geografi är inhemsk

Tillgänglighet: 1.5 stöd för geografi infördes.

Notera
För geografi har denna funktion en avståndstolerans på cirka 0,00001 meter och använder sfären i stället för sfäroidberäkning.

Notera
OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2 //s2.1.13.3 - ST_Intersects(g1, g2) --> Not (ST_Disjoint(g1, g2))

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.27

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel på geometri

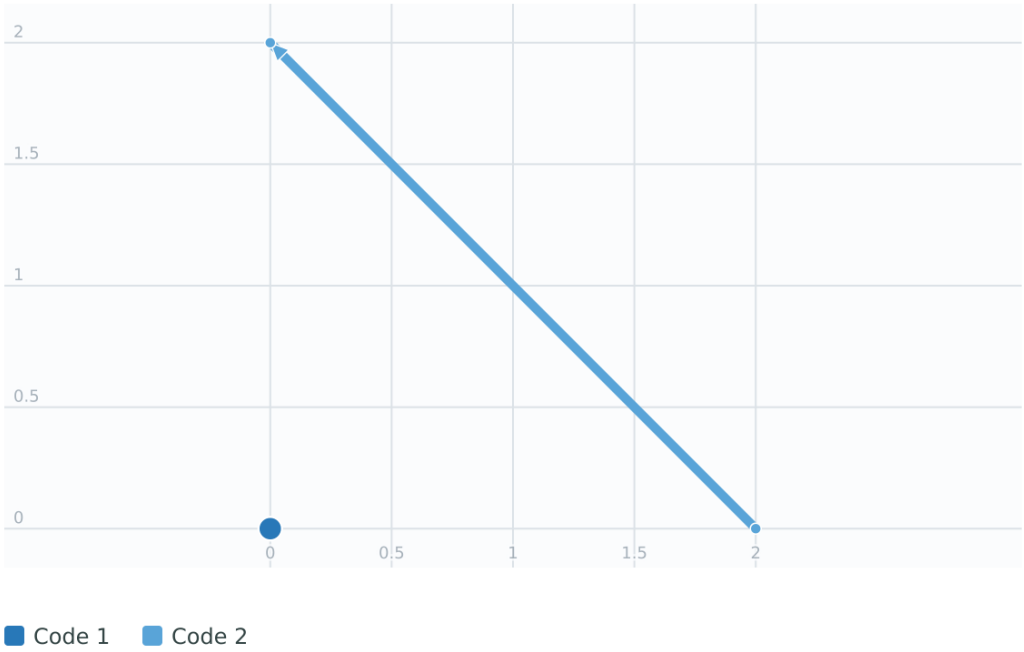
Code

```
SELECT ST_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
f
```

Figure



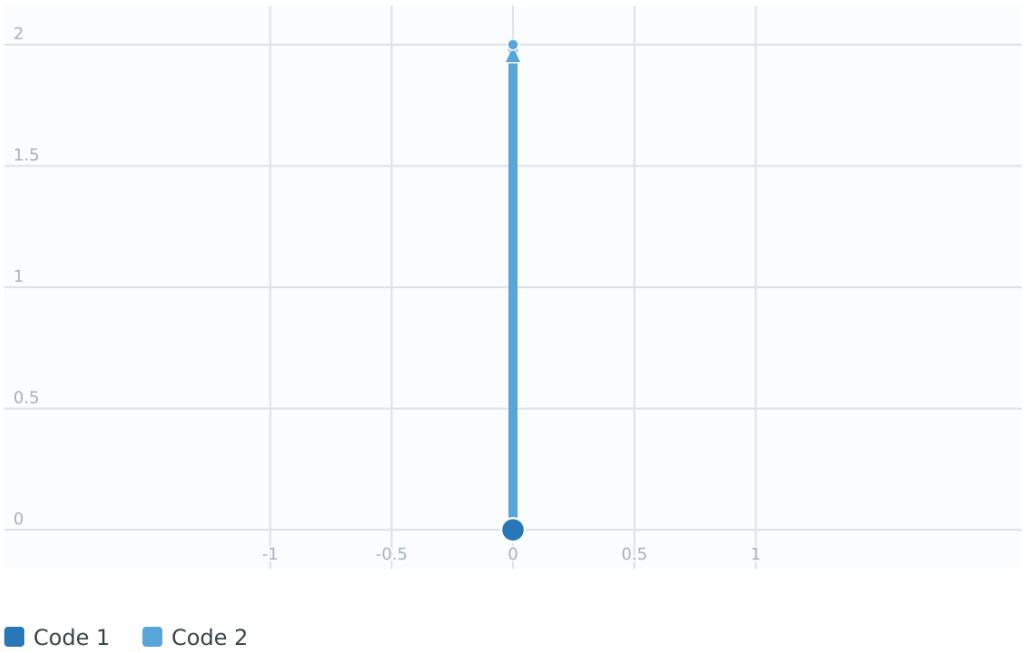
Code

```
SELECT ST_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Output

```
t
```

Figure



This example looks up rows in a table; for faster lookup, the geometry column should have a GiST index.

Code

```
SELECT id, name FROM cities WHERE ST_Intersects(geom, 'SRID=4326;POLYGON((28 53,27.707 ↩
    52.293,27 52,26.293 52.293,26 53,26.293 53.707,27 54,27.707 53.707,28 53))');
```

Output

```
id | name
----+-------
  2 | Minsk
(1 row)
```

Geografiska exempel

Code

```
SELECT ST_Intersects('SRID=4326;LINESTRING(-43.23456 72.4567,-43.23456 72.4568)'::geography ↩
    ,
'SRID=4326;POINT(-43.23456 72.4567772)'::geography
);
```

Output

```
t
```

Se även

&&, ST_3DIntersects, ST_Disjoint

7.11.1.10 ST_LineCrossingDirection

ST_LineCrossingDirection — Returnerar ett tal som anger korsningsbeteendet för två LineStrings

Synopsis

integer **ST_LineCrossingDirection**(geometry linestringA, geometry linestringB);

Beskrivning

Givet två linestringsar returnerar ett heltal mellan -3 och 3 som indikerar vilken typ av korsningsbeteende som finns mellan dem. 0 indikerar ingen korsning. Detta stöds endast för `LINESTRINGs`.

Korsningsnumret har följande innebörd:

- 0: LINJE UTAN KORS
- -1: LINJE KORS VÄNSTER
- 1: LINJE KORS HÖGER
- -2: LINJE MULTIKORS SLUT VÄNSTER
- 2: LINJE MULTIKORS SLUT HÖGER
- -3: LINJE FLERKORS SLUT SAMMA FÖRSTA VÄNSTER
- 3: LINJE MULTIKORS SLUT SAMMA FÖRSTA HÖGER

Tillgänglighet: 1.4

Exempel

Exempel: LINE CROSS LEFT och LINE CROSS RIGHT

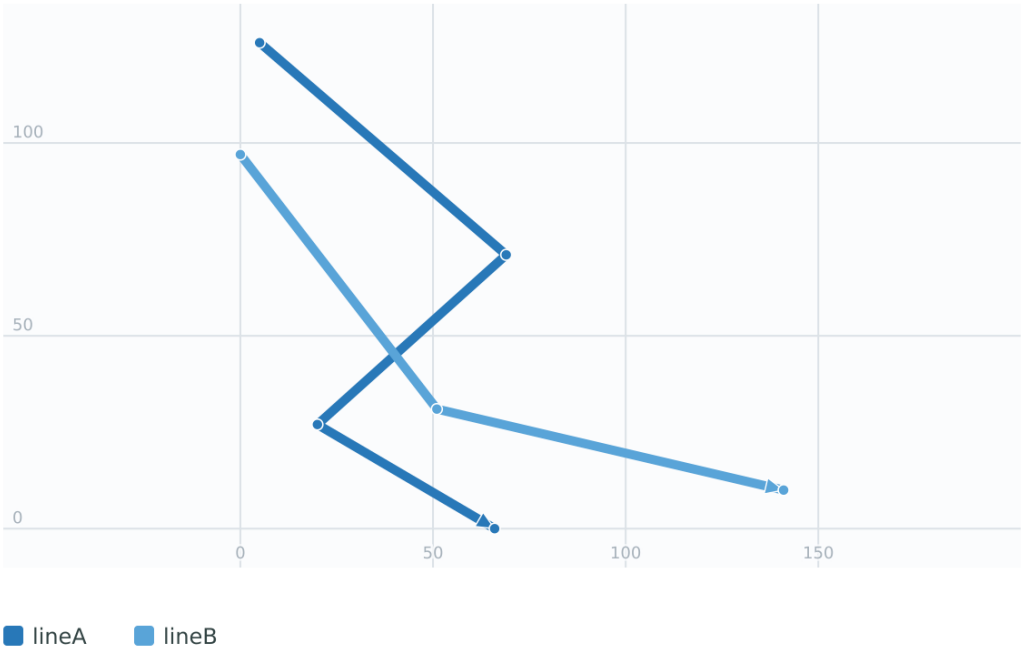
Code

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
  'LINESTRING(25 169,89 114,40 70,86 43)'::geometry As lineA,
  'LINESTRING (20 140,71 74,161 53)'::geometry As lineB
  ) As foo;
```

Output

```
A_cross_B | B_cross_A
----------+----------
       -1 |         1
```

Figure



Exempel: LINJE MULTICROSS SLUT SAME FÖRSTA VÄNSTER och LINJE MULTICROSS SLUT SAME FÖRSTA HÖGER

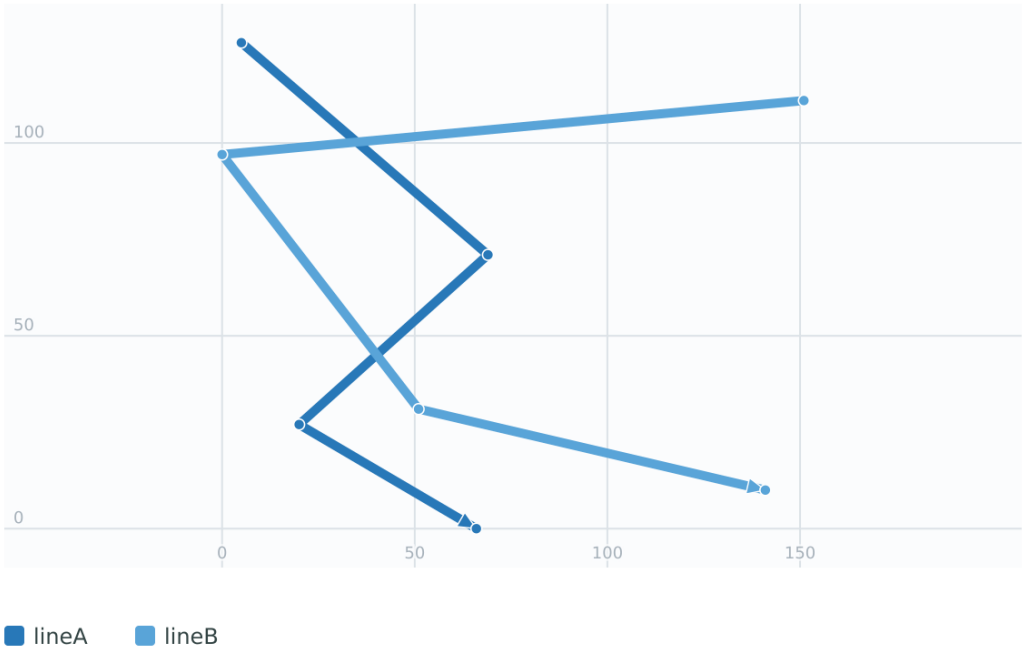
Code

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
 'LINESTRING(25 169,89 114,40 70,86 43)'::geometry As lineA,
 'LINESTRING(171 154,20 140,71 74,161 53)'::geometry As lineB
  ) As foo;
```

Output

```
A_cross_B | B_cross_A
----------+----------
        3 |        -3
```

Figure



Exempel: LINE MULTICROSS END LEFT och LINE MULTICROSS END RIGHT

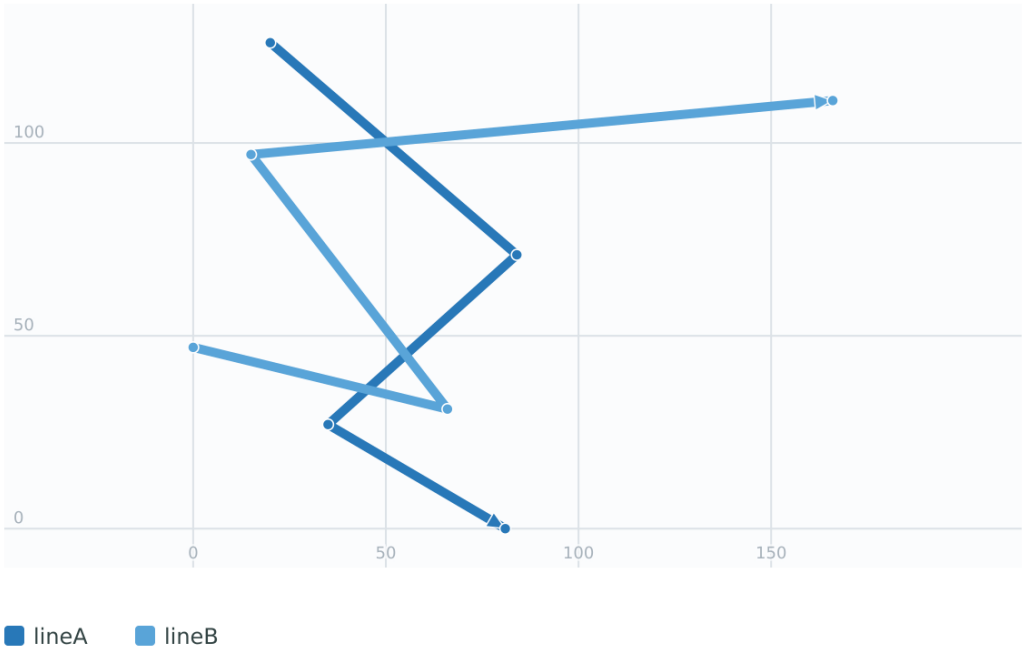
Code

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
  'LINESTRING(25 169,89 114,40 70,86 43)'::geometry As lineA,
  'LINESTRING(5 90,71 74,20 140,171 154)'::geometry As lineB
  ) As foo;
```

Output

```
A_cross_B | B_cross_A
----------+----------
       -2 |        2
```

Figure



Exempel: Hittar alla gator som korsar

Code

```
SELECT s1.gid, s2.gid, ST_LineCrossingDirection(s1.geom, s2.geom)
  FROM streets s1 CROSS JOIN streets s2
        ON (s1.gid != s2.gid AND s1.geom && s2.geom )
WHERE ST_LineCrossingDirection(s1.geom, s2.geom)
> 0;
```

Se även

ST_Crosses

7.11.1.11 ST_OrderingEquals

ST_OrderingEquals — Testar om två geometrier representerar samma geometri och har punkter i samma riktningsordning

Synopsis

boolean **ST_OrderingEquals**(geometry A, geometry B);

Beskrivning

ST_OrderingEquals jämför två geometrier och returnerar t (TRUE) om geometrierna är lika och koordinaterna är i samma ordning, annars returnerar den f (FALSE).

Notera
Denna funktion är implementerad enligt ArcSDE SQL-specifikationen snarare än SQL-MM. http://edndoc.esri.com/arcsde/9.1/sql_api/sqlapi3.htm#ST_OrderingEquals

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.43

Exempel

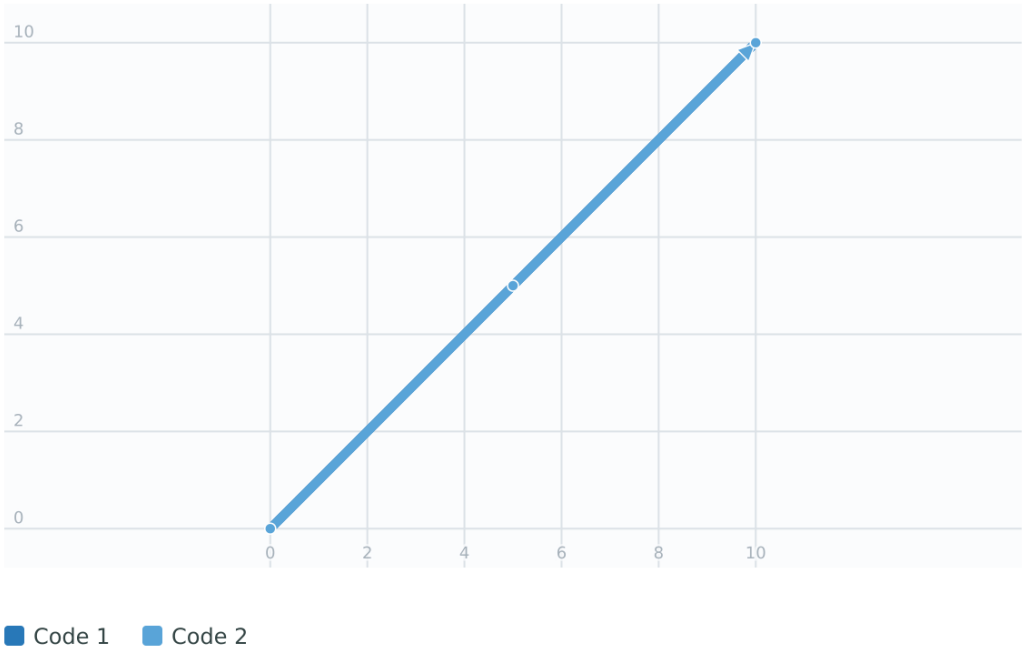
Code

```
SELECT ST_OrderingEquals('LINESTRING(0 0,10 10)',
'LINESTRING(0 0,5 5,10 10)');
```

Output

```
f
```

Figure



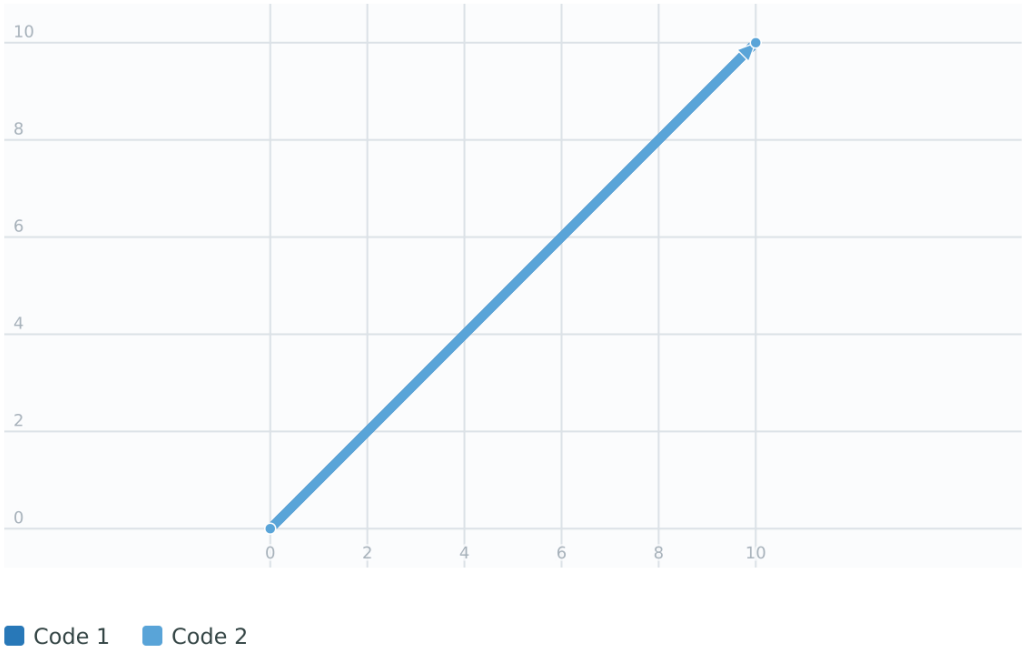
Code

```
SELECT ST_OrderingEquals('LINESTRING(0 0,10 10)',
'LINESTRING(0 0,10 10)');
```

Output

```
t
```

Figure



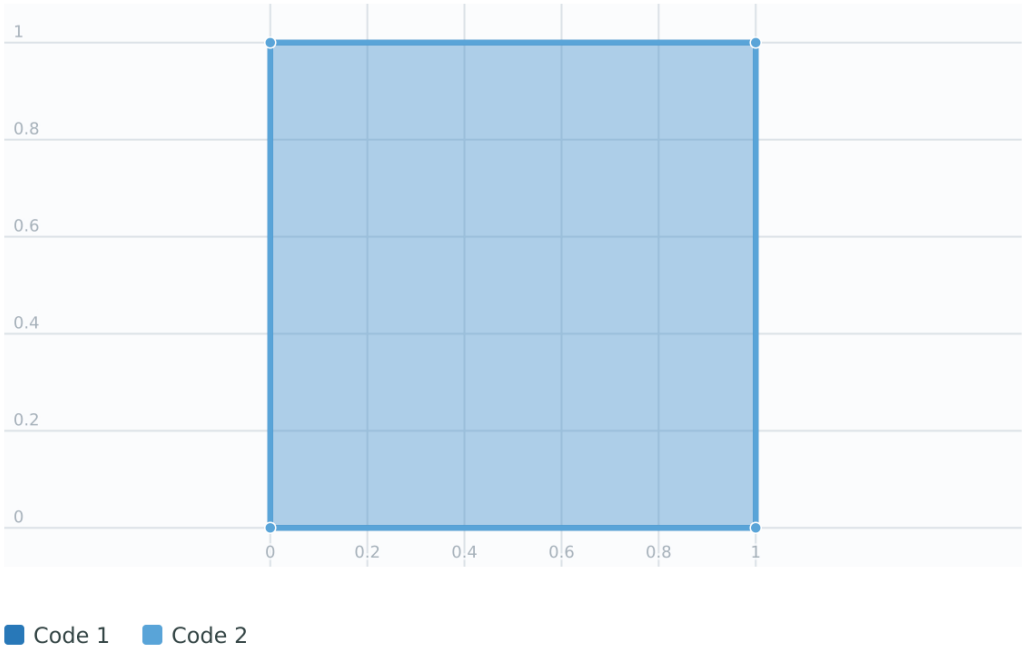
Code

```
SELECT ST_OrderingEquals('POLYGON((0 0,0 1,1 1,1 0,0 0))',
'POLYGON((0 0,1 0,1 1,0 1,0 0))');
```

Output

```
f
```

Figure



Se även

&&, ST_Equals, ST_Reverse

7.11.1.12 ST_Overlaps

ST_Overlaps — Testar om två geometrier har samma dimension och skär varandra, men var och en har minst en punkt som inte finns i den andra

Synopsis

boolean **ST_Overlaps**(geometry A, geometry B);

Beskrivning

Returnerar TRUE om geometri A och B ”överlappar varandra spatialt”. Två geometrier överlappar varandra om de har samma dimension, deras interiörer skär varandra i den dimensionen och var och en har minst en punkt inuti den andra (eller motsvarande, ingen av dem täcker den andra). Överlappningsrelationen är symmetrisk och irreflexiv.

I matematiska termer: *ST_Overlaps(A, B) ⇔ (dim(A) = dim(B) = dim(Int(A) ▯ Int(B))) ∧ (A ▯ B ≠ A) ∧ (A ▯ B ≠ B)*

Notera
&index_aware; För att undvika indexanvändning, använd funktionen _ST_Overlaps.

Utförs av GEOS-modulen

Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION

OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2 // s2.1.13.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.32

Exempel

ST_Overlaps returnerar TRUE i följande situationer:

Two MultiPoints overlap when they share some, but not all, points.

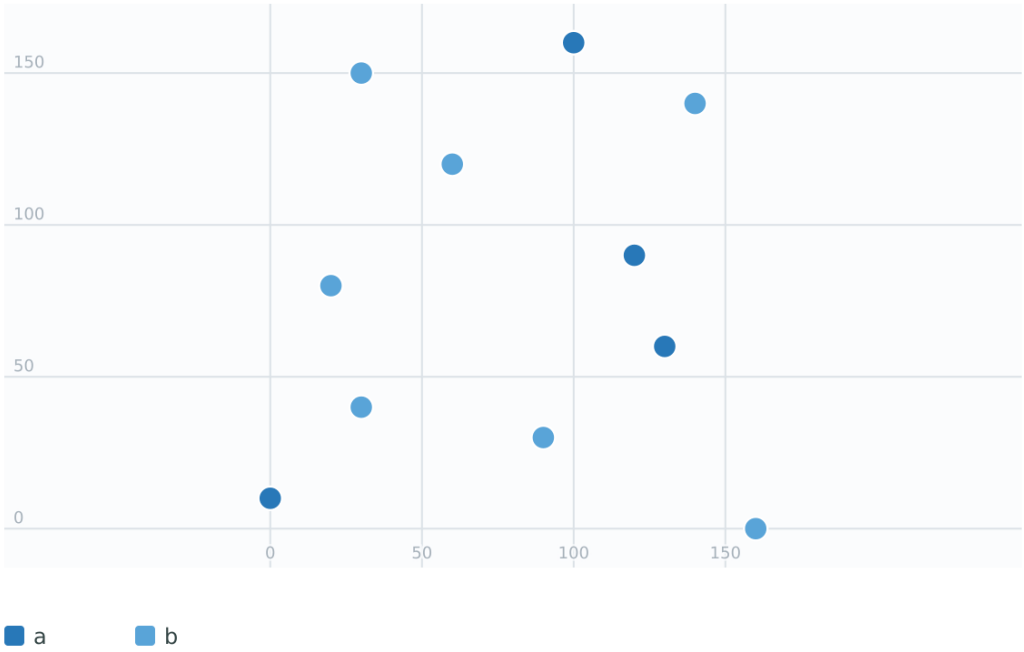
Code

```
SELECT ST_Overlaps(a, b)
FROM (SELECT
  'MULTIPOINT ((20 20),(40 90),(80 130),(140 100),(110 40),(120 170),(150 70))'::geometry ↩
    AS a,
  'MULTIPOINT ((180 10),(40 90),(80 130),(160 150),(110 40),(50 50),(50 160))'::geometry AS ↩
     b
) AS example;
```

Output

```
t
```

Figure



Two LineStrings overlap along part of their paths.

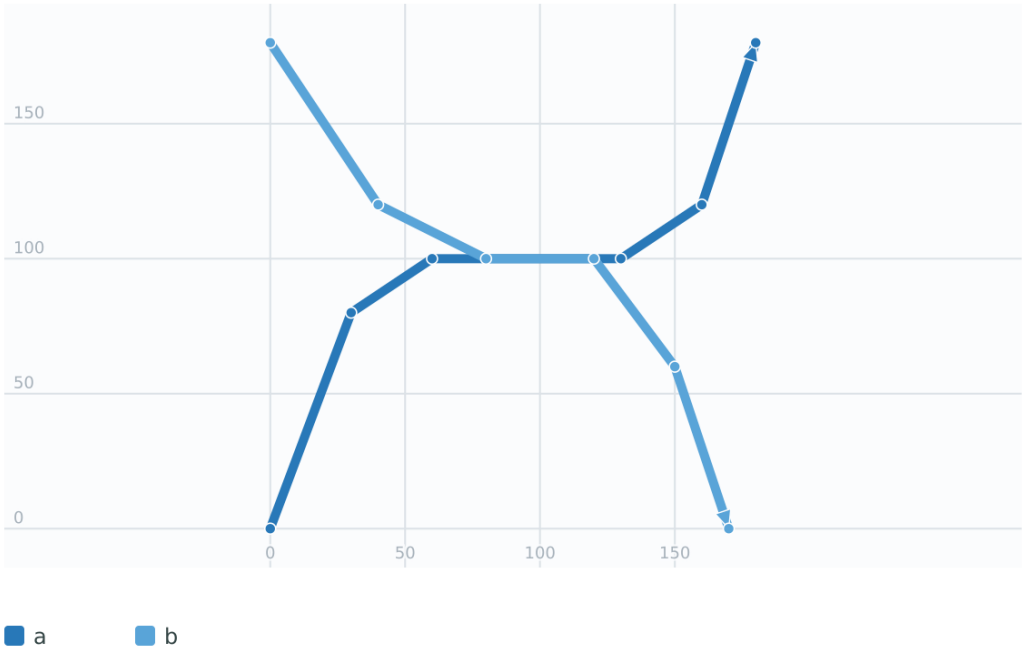
Code

```
SELECT ST_Overlaps(a, b)
FROM (SELECT
  'LINESTRING (10 10,40 90,70 110,140 110,170 130,190 190)'::geometry AS a,
  'LINESTRING (10 190,50 130,90 110,130 110,160 70,180 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two Polygons overlap where their interiors intersect and neither contains the other.

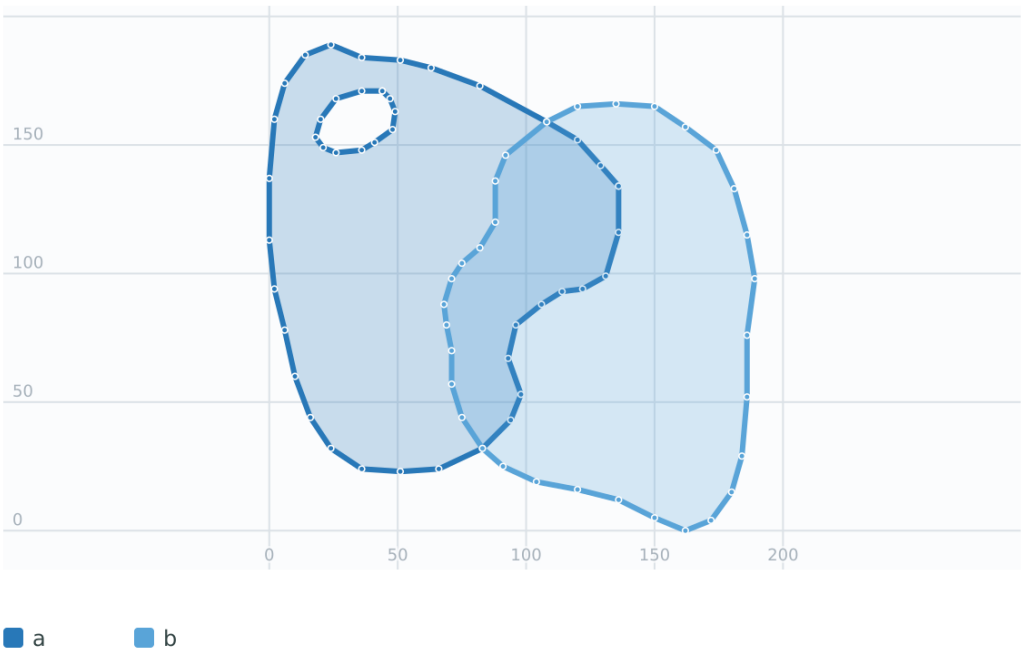
Code

```
SELECT ST_Overlaps(a, b)
FROM (SELECT
  'POLYGON ((140 140,140 122,135 105,126 100,118 99,110 94,100 86,97 73,102 59,98 49,87 ↩
    38,70 30,55 29,40 30,28 38,20 50,14 66,10 84,6 100,4 119,4 143,6 166,10 180,18 191,28 ↩
    195,40 190,55 189,67 186,86 179,112 165,124 158,133 148,140 140),(48 177,40 177,30 ↩
    174,24 166,22 159,25 155,30 153,40 154,45 157,52 162,53 169,51 174,48 177))'::geometry ↩
    AS a,
  'POLYGON ((75 63,79 50,87 38,95 31,108 25,124 22,140 18,154 11,166 6,176 10,184 21,188 ↩
    35,190 58,190 82,193 104,190 121,185 139,178 154,166 163,154 171,139 172,124 171,112 ↩
    165,96 152,92 142,92 126,86 116,79 110,75 104,72 94,73 86,75 76,75 63))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



En punkt på en LineString är innesluten, men eftersom den har en lägre dimension överlappar eller korsar den inte varandra.

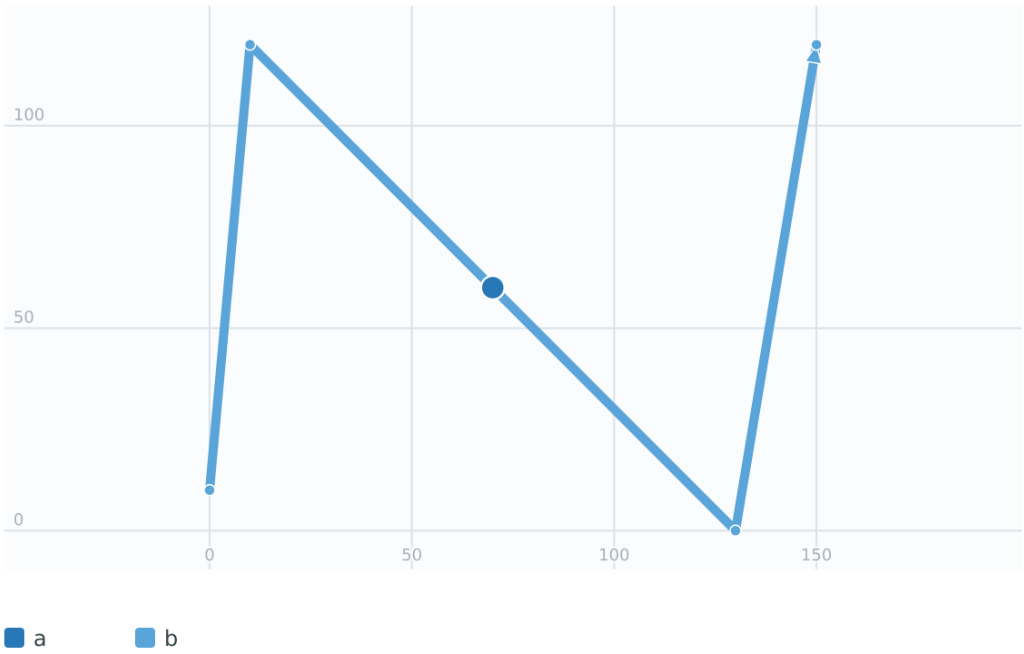
Code

```
SELECT ST_Overlaps(a, b) AS overlaps, ST_Crosses(a, b) AS crosses,
       ST_Intersects(a, b) AS intersects, ST_Contains(b, a) AS b_contains_a
FROM (SELECT 'POINT (100 100)'::geometry As a,
             'LINESTRING (30 50,40 160,160 40,180 160)'::geometry  AS b) AS t
```

Output

```
overlaps | crosses | intersects | b_contains_a
---------+---------+------------+--------------
f        | f       | t          | t
```

Figure



En LineString som delvis täcker en Polygon skär och korsar, men överlappar inte eftersom den har olika dimensioner.

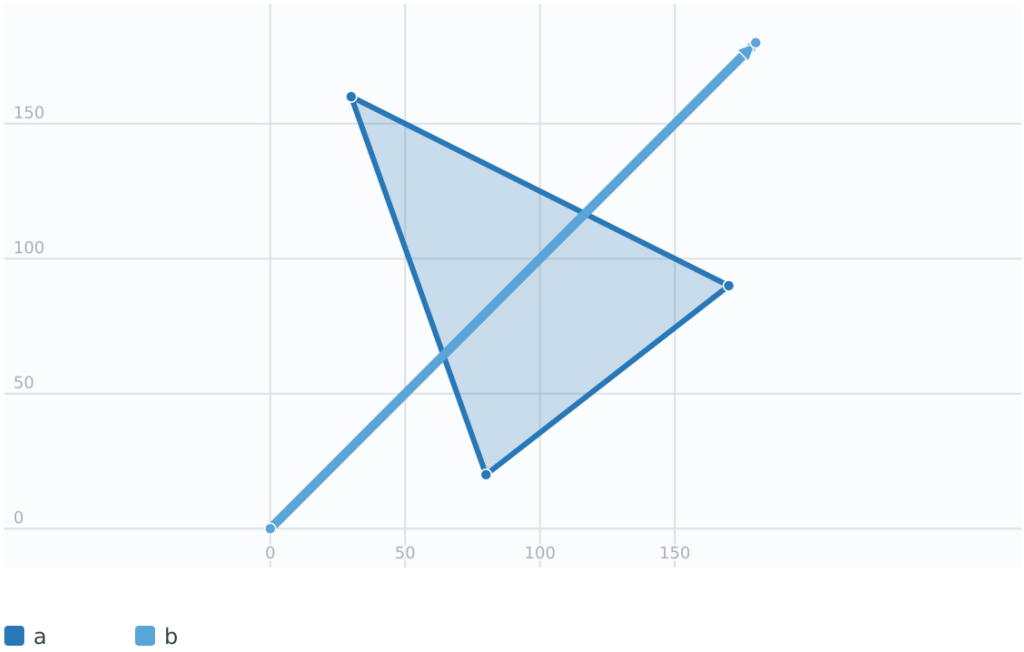
Code

```
SELECT ST_Overlaps(a, b) AS overlaps, ST_Crosses(a, b) AS crosses,
       ST_Intersects(a, b) AS intersects, ST_Contains(a, b) AS contains
FROM (SELECT 'POLYGON ((40 170,90 30,180 100,40 170))'::geometry AS a,
             'LINESTRING(10 10,190 190)'::geometry AS b) AS t;
```

Output

```
overlap | crosses | intersects | contains
---------+---------+------------+--------------
 f       | t       | t          | f
```

Figure



Två polygoner som skär varandra men där ingen av dem är innesluten i den andra överlappar varandra, men korsar inte varandra eftersom deras skärningspunkt har samma dimension.

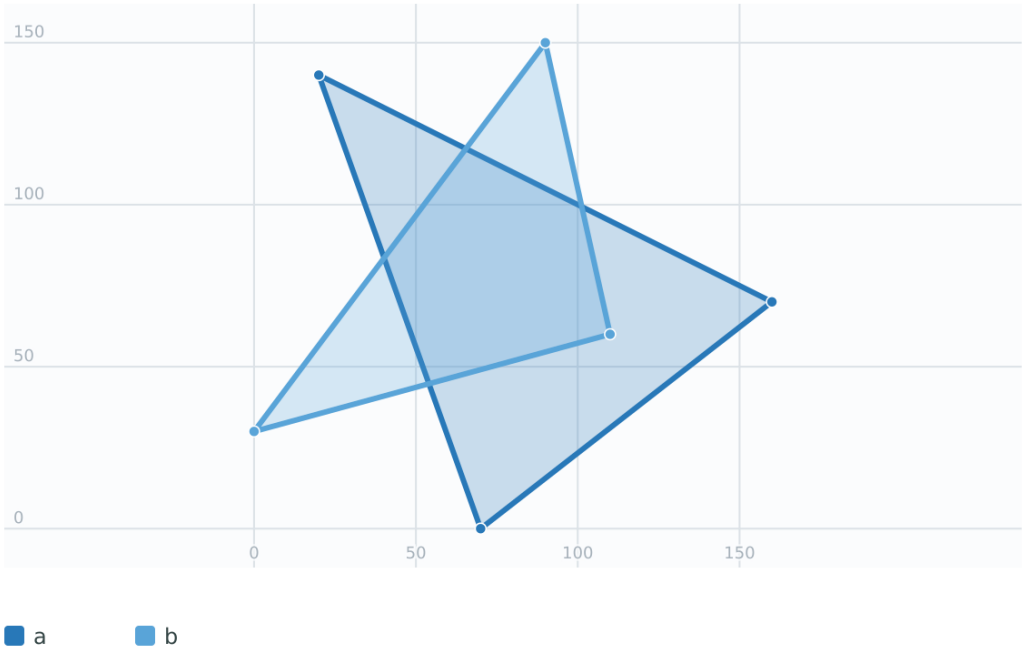
Code

```
SELECT ST_Overlaps(a, b) AS overlaps, ST_Crosses(a, b) AS crosses,
       ST_Intersects(a, b) AS intersects, ST_Contains(b, a) AS b_contains_a,
       ST_Dimension(a) AS dim_a, ST_Dimension(b) AS dim_b,
       ST_Dimension(ST_Intersection(a, b)) AS dim_int
FROM (SELECT 'POLYGON ((40 170,90 30,180 100,40 170))'::geometry AS a,
             'POLYGON ((110 180,20 60,130 90,110 180))'::geometry AS b) As t;
```

Output

```
overlaps | crosses | intersects | b_contains_a | dim_a | dim_b | dim_int
----------+---------+------------+--------------+-------+-------+-----------
 t        | f       | t          | f            |     2 |     2 |       2
```

Figure



Se även

ST_Contains, ST_Crosses, ST_Dimension, ST_Intersects

7.11.1.13 ST_Relate

ST_Relate — Testar om två geometrier har en topologisk relation som matchar ett Intersection Matrix-mönster, eller beräknar deras Intersection Matrix

Synopsis

boolean **ST_Relate**(geometry geomA, geometry geomB, text intersectionMatrixPattern);
text **ST_Relate**(geometry geomA, geometry geomB);
text **ST_Relate**(geometry geomA, geometry geomB, integer boundaryNodeRule);

Beskrivning

Dessa funktioner gör det möjligt att testa och utvärdera det spatiala (topologiska) förhållandet mellan två geometrier, enligt definitionen i Dimensionally Extended 9-Intersection Model (DE-9IM).

DE-9IM specificeras som en matris med 9 element som anger dimensionen på skärningspunkterna mellan inre, yttre och yttre delar av två geometrier. Den representeras av en textsträng med 9 tecken som använder symbolerna F", "0", "1", "2"(t.ex. `FF1FF0102`").

En viss typ av spatial relation kan testas genom att matcha intersektionsmatrisen med ett *intersektionsmatrismönster*. Mönster kan innehålla tilläggssymbolerna T"(som betyder skärningspunkten är inte tom") och "*"(som betyder "vilket värde som helst"). Vanliga spatiala relationer tillhandahålls av de namngivna funktionerna ST_Contains, ST_ContainsProperly, ST_Covers, ST_CoveredBy, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps, ST_Touches, och ST_Within. Genom att använda ett explicit mönster kan man testa flera villkor för skärningar, korsningar osv. i ett steg.

Det gör det också möjligt att testa spatiala relationer som inte har en namngiven spatial relationsfunktion. Till exempel har relationen ”Interior-IntersectsDE-9IM-mönstret T********, som inte utvärderas av något namngivet predikat.

Mer information finns på Avsnitt 5.1.

Variant 1: Testar om två geometrier är spatialt relaterade enligt det angivna `intersectionMatrixPattern`.

Notera
Till skillnad från de flesta namngivna spatiala relationspredikaten innehåller detta INTE automatiskt ett indexanrop. Anledningen är att vissa relationer är sanna för geometrier som INTE korsar varandra (t.ex. Disjoint). Om du använder ett relationsmönster som kräver intersektion, inkludera då indexanropet &&.

Notera
Det är bättre att använda en namngiven relationsfunktion om en sådan finns tillgänglig, eftersom de automatiskt använder ett spatialt index om ett sådant finns. De kan också implementera prestandaoptimeringar som inte är tillgängliga med fullständig relationsutvärdering.

Variant 2: Returnerar DE-9IM-matrissträngen för det spatiala förhållandet mellan de två indatageometrierna. Matrissträngen kan testas för att matcha ett DE-9IM-mönster med hjälp av ST_RelateMatch.

Variant 3: Som variant 2, men tillåter specificering av en **Boundary Node Rule**. En Boundary Node Rule ger möjlighet till finare kontroll över om MultiLineStrings ändpunkter anses ligga i DE-9IM Interior eller Boundary. Värdena för `boundaryNodeRule` är:

- 1: **OGC-Mod2** - linjens ändpunkter är i Boundary om de förekommer ett udda antal gånger. Detta är den regel som definieras av OGC SFS-standarden och är standard för `ST_Relate`.
- 2: **Slutpunkt** - alla slutpunkter ligger i Boundary.
- 3: **MultivalentEndpoint** - ändpunkterna är i Boundary om de förekommer mer än en gång. Med andra ord är gränsen alla de änslutna”eller ”inre”ändpunkterna (men inte de ”icke anslutna/yttre”).
- 4: **MonovalentEndpoint** - ändpunkterna ligger i Boundary om de bara förekommer en gång. Med andra ord är gränsen alla de öbundna”eller ”yttre”ändpunkterna.

Denna funktion finns inte i OGC-specifikationen, men är underförstådd. se s2.1.13.2

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2 // s2.1.13.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.25

Utförs av GEOS-modulen

Förbättrad: 2.0.0 - stöd för att ange en regel för gränsnoder har lagts till.

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Exempel

Använda den booleska värderingsfunktionen för att testa spatiala relationer.

Code

```
SELECT ST_Relate('POINT(1 2)', ST_Buffer('POINT(1 2)', 2), '0FFFFF212');
```

Output

```
t
```

Code

```
SELECT ST_Relate('POINT(1 2)', ST_Buffer('POINT(1 2)', 2), '*FF*FF212');
```

Output

```
t
```

Testa ett anpassat spatialt relationsmönster som ett frågevillkor, med && för att aktivera användning av ett spatialt index.

Find compounds that properly intersect (not just touch) a poly (Interior Intersects)

Code

```
SELECT c.* , p.name As poly_name
    FROM polys AS p
    INNER JOIN compounds As c
          ON c.geom && p.geom
             AND ST_Relate(p.geom, c.geom, 'T********');
```

Beräkning av intersektionsmatrisen för spatiala relationer.

Code

```
SELECT ST_Relate('POINT(1 2)',
ST_Buffer('POINT(1 2)', 2));
```

Output

```
0FFFFF212
```

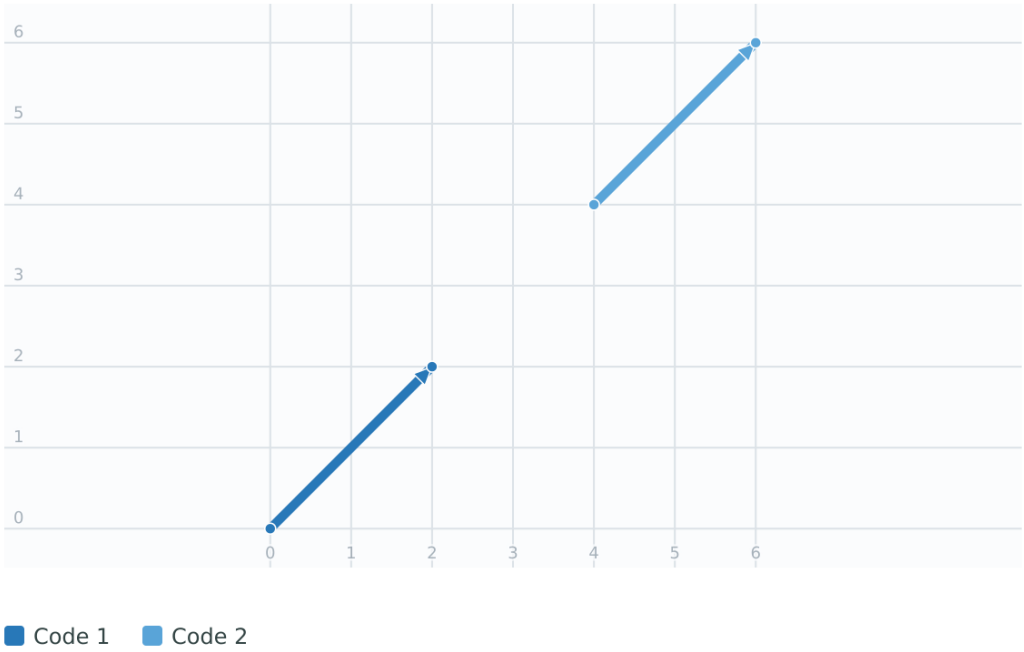
Code

```
SELECT ST_Relate('LINESTRING(1 2,3 4)',
'LINESTRING(5 6,7 8)' );
```

Output

```
FF1FF0102
```

Figure



Använda olika Boundary Node Rules för att beräkna det spatiala förhållandet mellan en LineString och en MultiLineString med en duplicerad ändpunkt `(3 3)`::

- Med hjälp av **OGC-Mod2-regeln** (1) är den duplicerade ändpunkten i det **inre** av MultiLineString, så DE-9IM-matrisposten [aB:bI] är `0` och [aB:bB] är `F`..
- Med hjälp av **ändpunktsregeln** (2) ligger den duplicerade ändpunkten i **gränsen** för MultiLineString, så DE-9IM-matrisens post [aB:bI] är `F` och [aB:bB] är `0`..

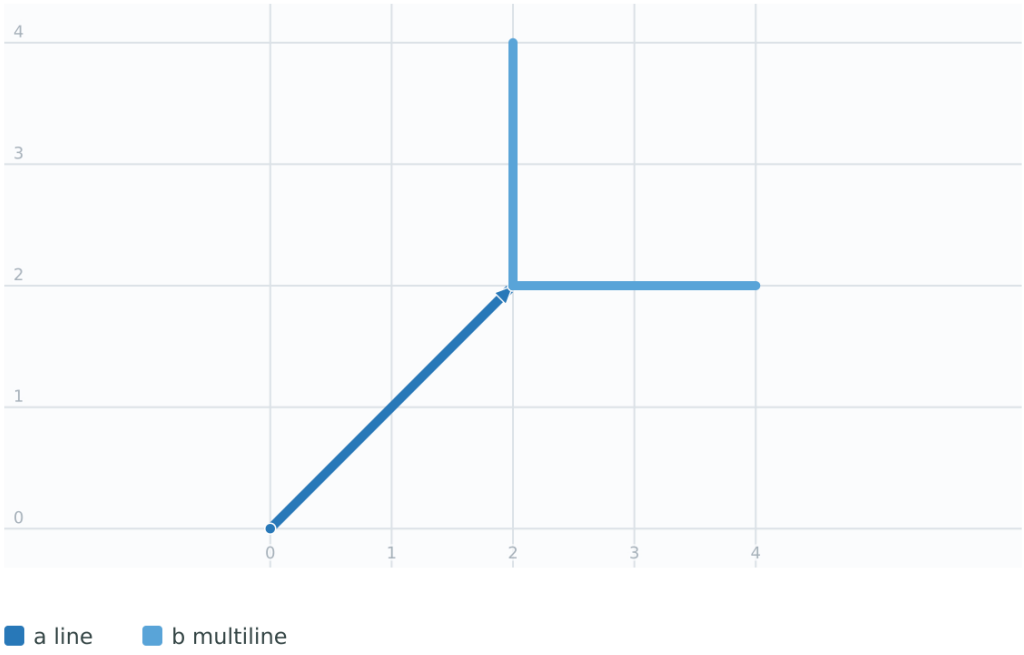
Code

```
WITH data AS (SELECT
  'LINESTRING(1 1,3 3)'::geometry AS a_line,
  'MULTILINESTRING((3 3,3 5),(3 3,5 3))'::geometry AS b_multiline
)
SELECT ST_Relate(a_line, b_multiline, 1) AS bnr_mod2,
       ST_Relate(a_line, b_multiline, 2) AS bnr_endpoint
    FROM data;
```

Output

```
bnr_mod2  | bnr_endpoint
----------+-------------
 FF10F0102 | FF1F00102
```

Figure



Se även

Avsnitt 5.1, ST_RelateMatch, ST_Contains, ST_ContainsProperly, ST_Covers, ST_CoveredBy, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within

7.11.1.14 ST_RelateMatch

ST_RelateMatch — Testar om en DE-9IM Intersection Matrix matchar ett Intersection Matrix-mönster

Synopsis

boolean **ST_RelateMatch**(text intersectionMatrix, text intersectionMatrixPattern);

Beskrivning

Testar om ett `intersectionMatrix-värde` enligt Dimensionally Extended 9-Intersection Model (DE-9IM) uppfyller ett `intersectionMatrixPattern`. Värden för intersektionsmatriser kan beräknas med ST_Relate.

Mer information finns på Avsnitt 5.1.

Utförs av GEOS-modulen

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT ST_RelateMatch('101202FFF', 'TTTTTTFFF') ;
```

Output

```
t
```

Mönster för vanliga spatiala relationer matchade mot intersektionsmatrisvärden, för en linje i olika positioner i förhållande till en polygon

Code

```
SELECT pat.name AS relationship, pat.val AS pattern,
       mat.name AS position, mat.val AS matrix,
       ST_RelateMatch(mat.val, pat.val) AS match
    FROM (VALUES ( 'Equality', 'T1FF1FFF1' ),
                 ( 'Overlaps', 'T*T***T**' ),
                 ( 'Within', 'T*F**F***' ),
                 ( 'Disjoint', 'FF*FF****' )) AS pat(name, val)
    CROSS JOIN
        (VALUES  ('non-intersecting', 'FF1FF0212'),
                 ('overlapping', '1010F0212'),
                 ('inside', '1FF0FF212')) AS mat(name, val);
```

Output

```
relationship |  pattern  |     position     |  matrix   | match
-------------+-----------+------------------+-----------+-------
 Equality    | T1FF1FFF1 | non-intersecting | FF1FF0212 | f
 Equality    | T1FF1FFF1 | overlapping      | 1010F0212 | f
 Equality    | T1FF1FFF1 | inside           | 1FF0FF212 | f
 Overlaps    | T*T***T** | non-intersecting | FF1FF0212 | f
 Overlaps    | T*T***T** | overlapping      | 1010F0212 | t
 Overlaps    | T*T***T** | inside           | 1FF0FF212 | f
 Within      | T*F**F*** | non-intersecting | FF1FF0212 | f
 Within      | T*F**F*** | overlapping      | 1010F0212 | f
 Within      | T*F**F*** | inside           | 1FF0FF212 | t
 Disjoint    | FF*FF**** | non-intersecting | FF1FF0212 | t
 Disjoint    | FF*FF**** | overlapping      | 1010F0212 | f
 Disjoint    | FF*FF**** | inside           | 1FF0FF212 | f
```

Se även

Avsnitt 5.1, ST_Relate

7.11.1.15 ST_Touches

ST_Touches — Testar om två geometrier har minst en gemensam punkt, men deras inre delar inte skär varandra

Synopsis

boolean **ST_Touches**(geometry A, geometry B);

Beskrivning

Returnerar TRUE om A och B skär varandra, men deras interiörer inte skär varandra. På motsvarande sätt har A och B minst en gemensam punkt och de gemensamma punkterna ligger inom minst en gräns. För Point/Point-indata är relationen alltid FALSK, eftersom punkter inte har någon gräns.

I matematiska termer: *ST_Touches(A, B) ⇔ (Int(A) ∩ Int(B) = ∅) ∧ (A ∩ B ≠ ∅)*

Detta förhållande gäller om DE-9IM Intersection Matrix för de två geometrierna matchar en av:

- FT*******
- F**T*****
- F***T****

Notera
&index_aware; För att undvika att använda ett index, använd _ST_Touches istället.

Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2 // s2.1.13.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.28

Exempel

Predikatet ST_Touches returnerar TRUE i följande exempel.

Two Polygons touching along a boundary segment.

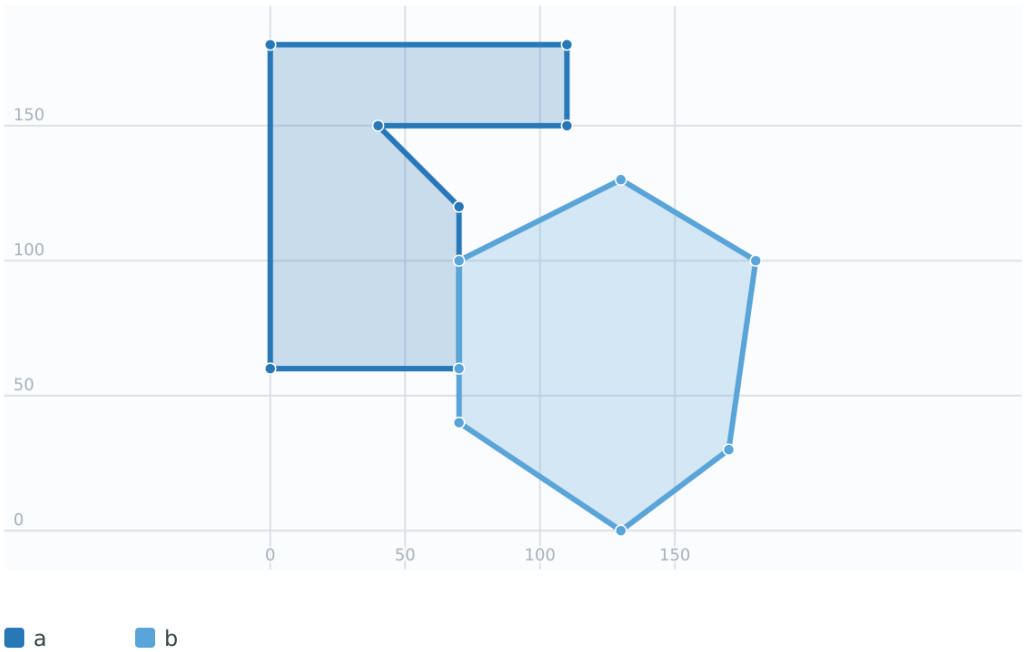
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190))'::geometry AS a,
  'POLYGON ((80 110,140 140,190 110,180 40,140 10,80 50,80 70,80 110))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two Polygons touching at a boundary point.

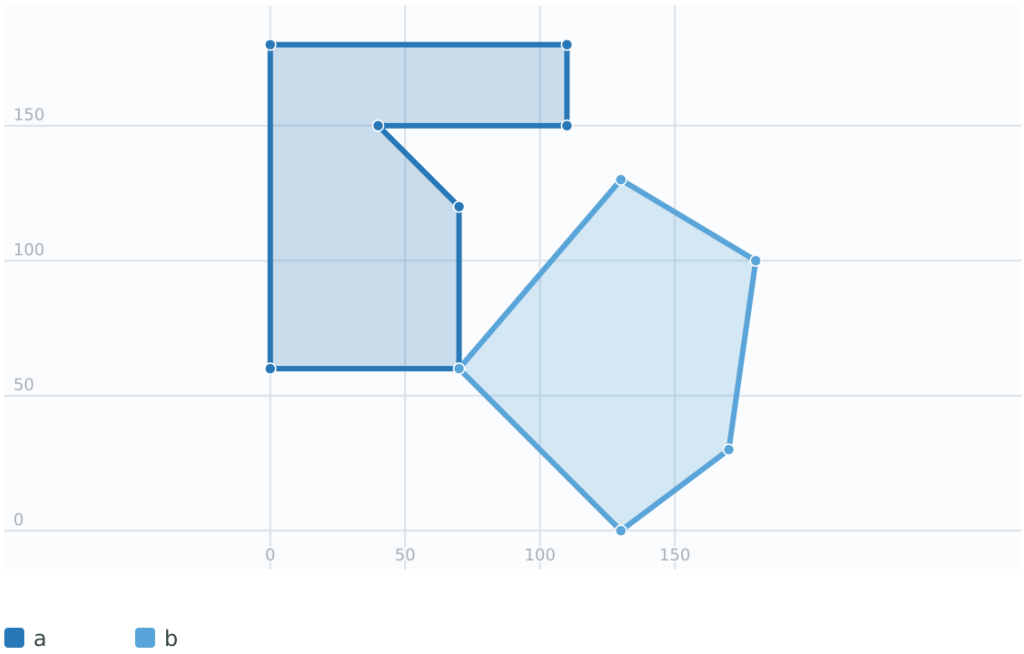
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190))'::geometry AS a,
  'POLYGON ((140 140,190 110,180 40,140 10,80 70,140 140))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon and a LineString touching at the line endpoint.

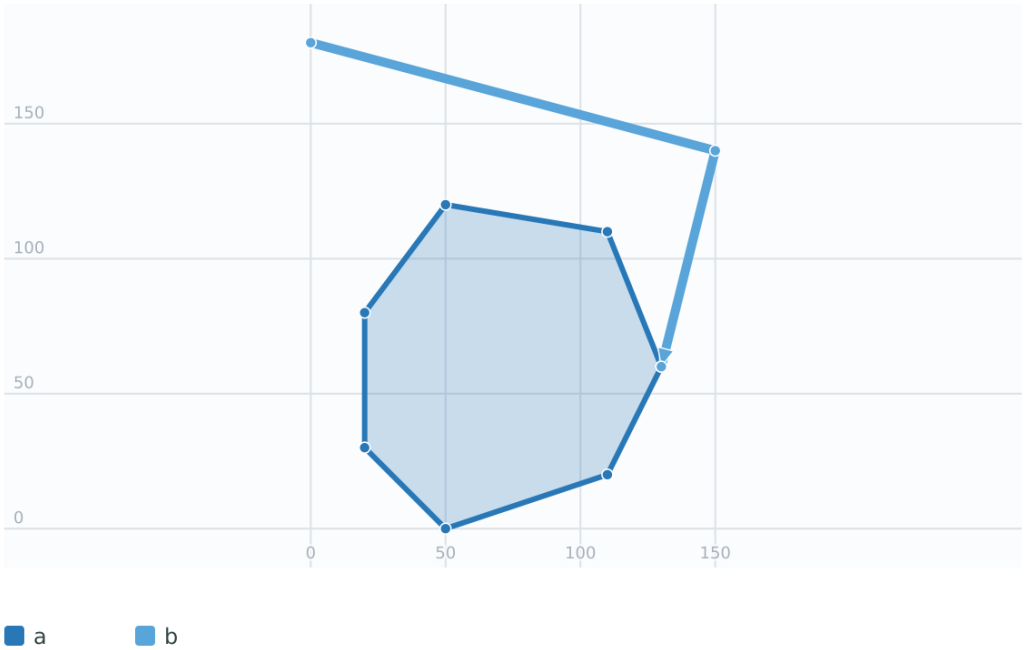
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((80 10,50 40,50 90,80 130,140 120,160 70,140 30,80 10))'::geometry AS a,
  'LINESTRING (30 190,180 150,160 70)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two LineStrings touching at their endpoints.

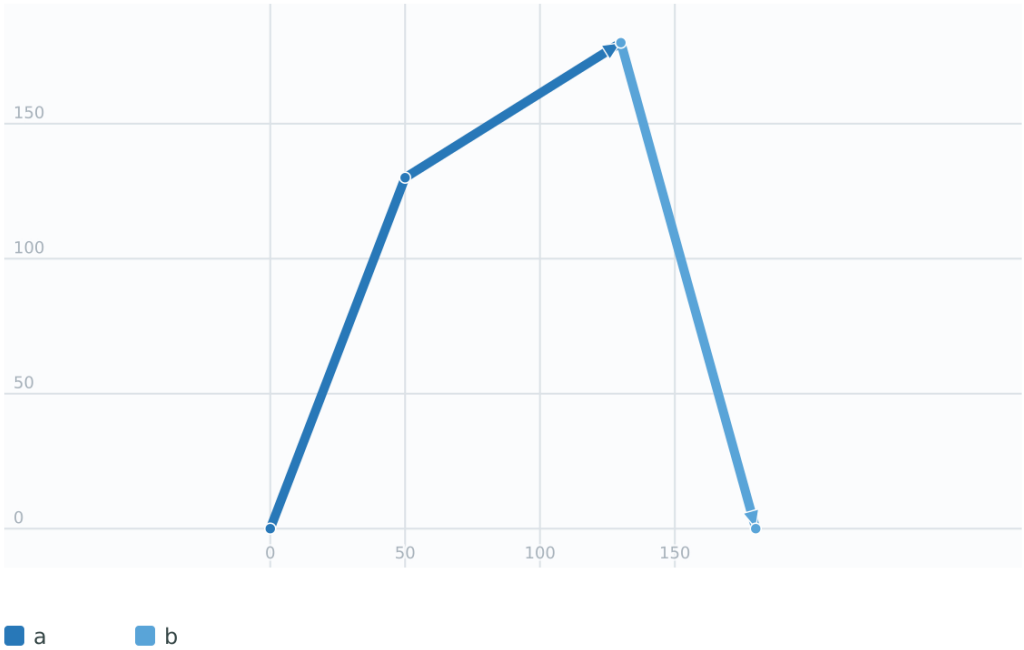
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'LINESTRING (10 10,60 140,140 190)'::geometry AS a,
  'LINESTRING (140 190,190 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two LineStrings touching where an endpoint meets the other line.

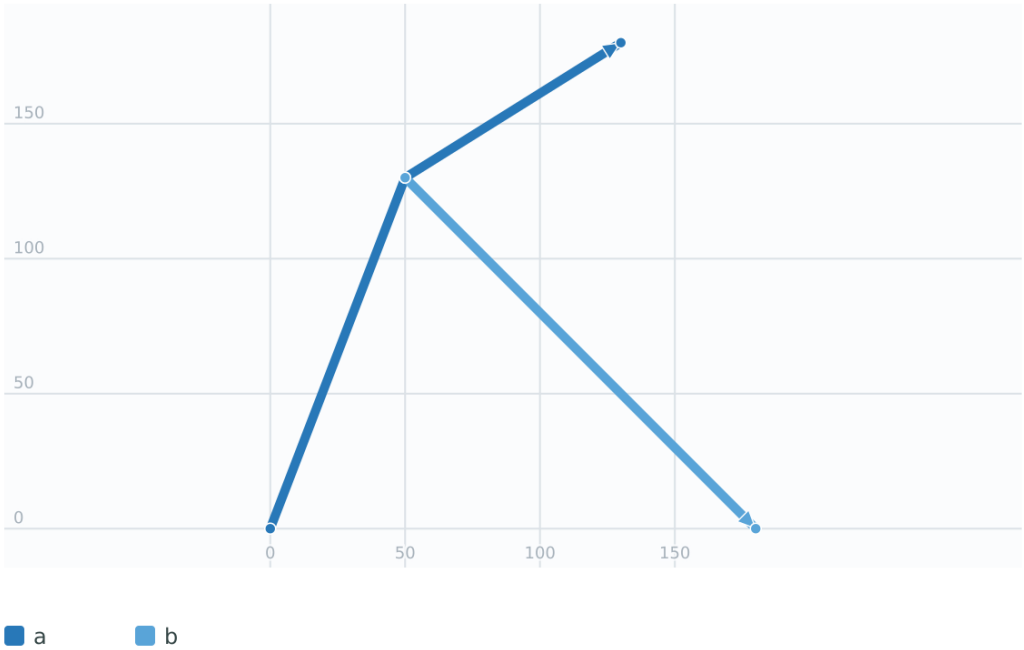
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'LINESTRING (10 10,60 140,140 190)'::geometry AS a,
  'LINESTRING (60 140,190 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Point touching a Polygon boundary.

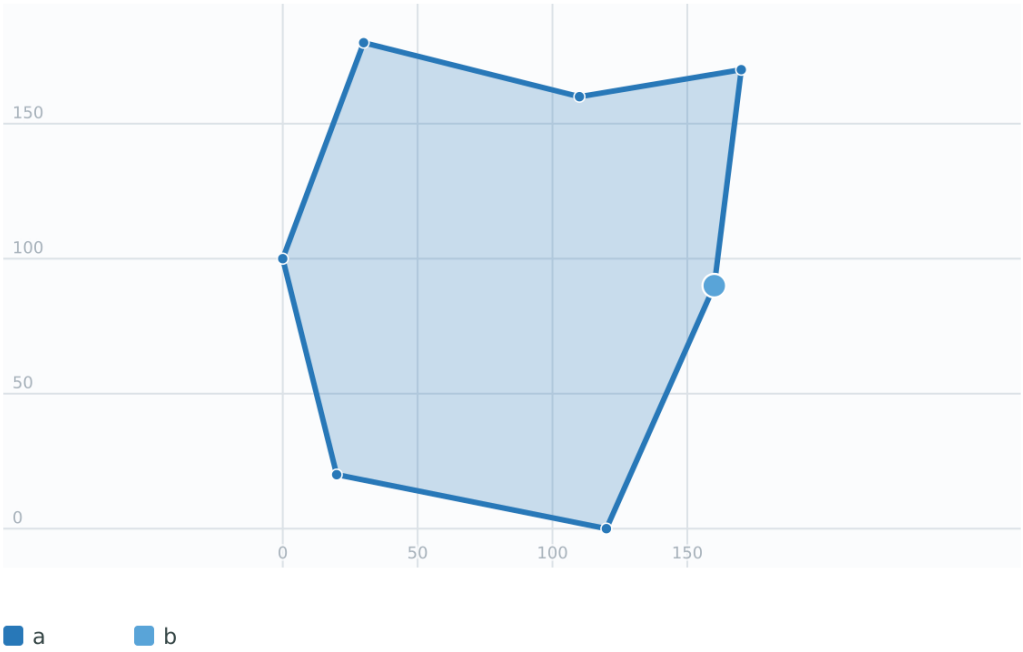
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((30 30,10 110,40 190,120 170,180 180,170 100,130 10,30 30))'::geometry AS a,
  'POINT (170 100)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



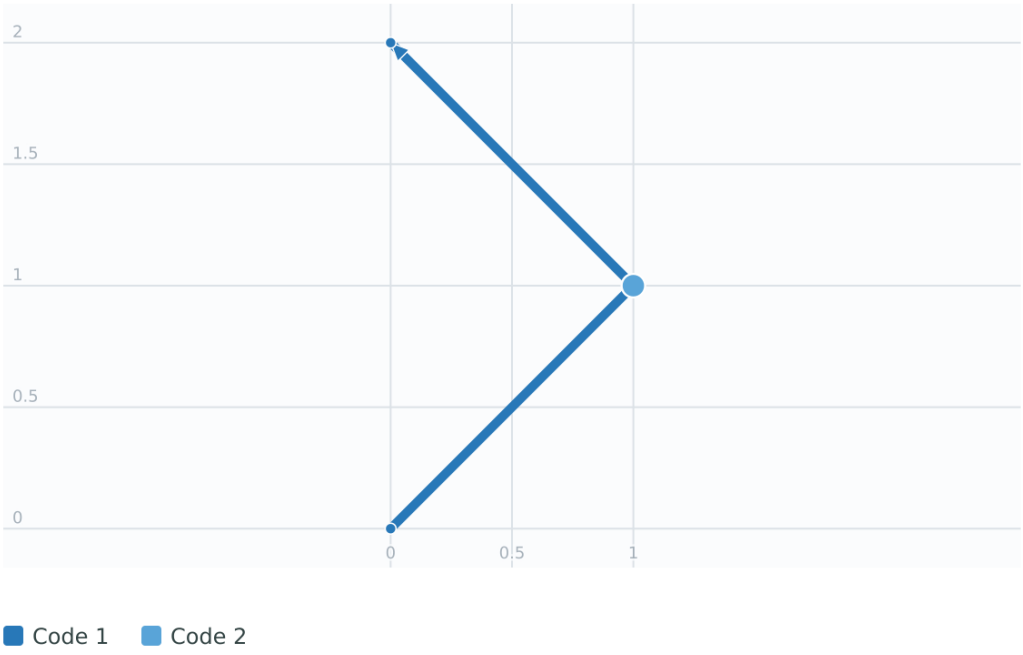
Code

```
SELECT ST_Touches('LINESTRING(0 0,1 1,0 2)'::geometry, 'POINT(1 1)'::geometry);
```

Output

```
f
```

Figure



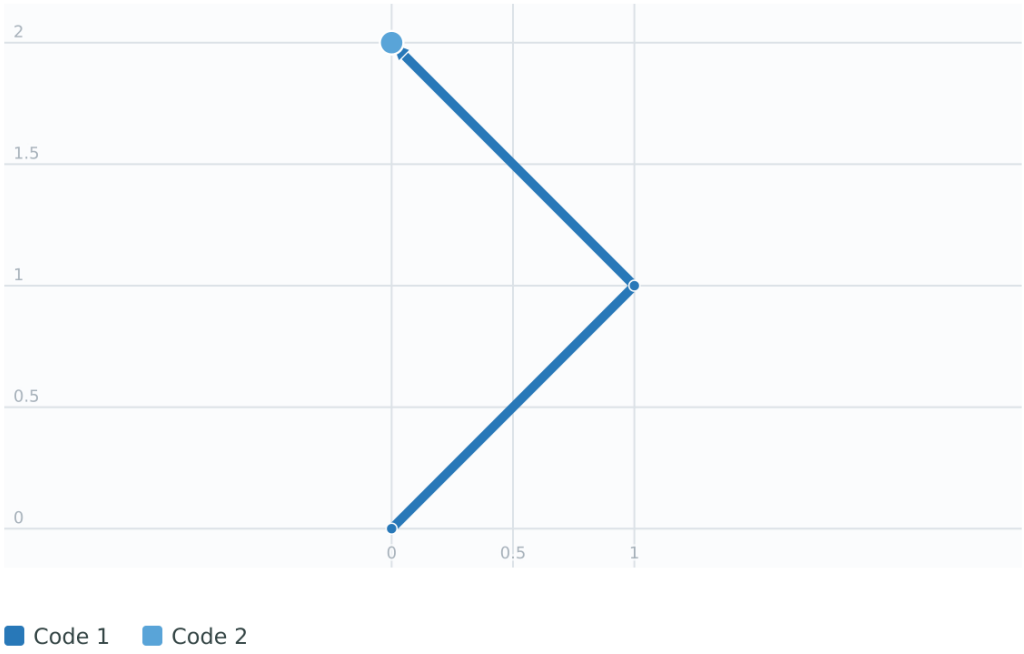
Code

```
SELECT ST_Touches('LINESTRING(0 0,1 1,0 2)'::geometry, 'POINT(0 2)'::geometry);
```

Output

```
t
```

Figure



7.11.1.16 ST_Within

ST_Within — Testar om varje punkt i A ligger i B, och deras interiörer har en gemensam punkt

Synopsis

boolean **ST_Within**(geometry A, geometry B);

Beskrivning

Returnerar TRUE om geometri A ligger inom geometri B. A ligger inom B om och endast om alla punkter i A ligger inom (dvs. i det inre eller på gränsen till) B (eller motsvarande, inga punkter i A ligger i det yttre av B), och det inre av A och B har minst en punkt gemensamt.

För att denna funktion ska vara meningsfull måste båda källgeometrierna ha samma koordinatprojektion och samma SRID.

I matematiska termer: *ST_Within(A, B) ⇔ (A ∩ B = A) ∧ (Int(A) ∩ Int(B) ≠ ∅)*

Within-relationen är reflexiv: varje geometri är inom sig själv. Relationen är antisymmetrisk: om `ST_Within(A,B) = true` och `ST_Within(B,A) = true`, så måste de två geometrierna vara topologiskt lika(`ST_Equals(A,B) = true`).

ST_Within är det omvända av ST_Contains. Så `ST_Within(A,B) = ST_Contains(B,A)`..

Notera
Eftersom interiörerna måste ha en gemensam punkt är en finess i definitionen att linjer och punkter som ligger helt inom polygoners eller linjers gränser *inte* är inom geometrin. För ytterligare detaljer se Subtiliteter i OGC Covers, Contains, Within. Predikatet ST_CoveredBy ger en mer inkluderande relation.

Notera
&index_aware; För att undvika indexanvändning, använd funktionen `_ST_Within`.

Utförs av GEOS-modulen

Förbättrad: 2.3.0 Förbättring av PIP-kortslutning för geometri utökad för att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon.

Förbättrad: 3.0.0 aktiverade stöd för `GEOMETRYCOLLECTION`

Viktigt
Använd inte denna funktion med ogiltiga geometrier. Du kommer att få oväntade resultat.

OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.2 // s2.1.13.3 - a.Relate(b, ’T*F**F***’)

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.30

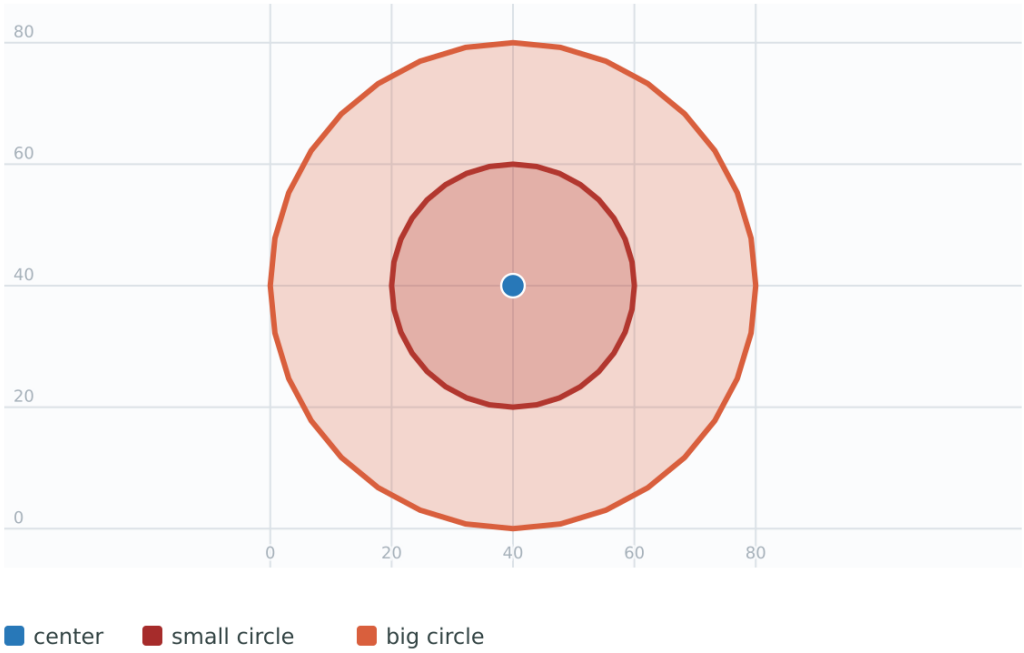
Exempel

This example defines two concentric circles once in a CTE so the same inputs can be rendered and then reused in related predicate checks.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 20) AS smallc,
         ST_Buffer(center, 40) AS bigc
  FROM (SELECT 'POINT(50 50)'::geometry AS center) AS p
)
SELECT smallc AS small_circle,
       bigc AS big_circle
FROM circles;
```

Figure



Using the same circles, `ST_Within` is reflexive on each circle, false from the larger circle into the smaller one, and true for `ST_Union` only when the union does not extend beyond the larger circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 20) AS smallc,
         ST_Buffer(center, 40) AS bigc
  FROM (SELECT 'POINT(50 50)'::geometry AS center) AS p
)
SELECT
  ST_Within(smallc, smallc) AS smallinsmall,
  ST_Within(smallc, bigc) AS smallinbig,
  ST_Within(bigc, smallc) AS biginsmall,
  ST_Within(ST_Union(smallc, bigc), bigc) AS unioninbig,
  ST_Within(bigc, ST_Union(smallc, bigc)) AS biginunion,
  ST_Equals(bigc, ST_Union(smallc, bigc)) AS bigisunion
FROM circles;
```

Output

```
t | t | f | t | t | t
```

Figure

Se även

ST_Contains, ST_CoveredBy, ST_Equals, ST_IsValid

7.11.2 Relationer på distans

7.11.2.1 ST_3DDWithin

ST_3DDWithin — Testar om två 3D-geometrier befinner sig inom ett givet 3D-avstånd

Synopsis

boolean **ST_3DDWithin**(geometry g1, geometry g2, double precision distance_of_srid);

Beskrivning

Returnerar true om 3D-avståndet mellan två geometrivärden inte är större än distance `distance_of_srid`. Avståndet anges i enheter som definieras av geometriernas spatiala referenssystem. För att denna funktion ska vara meningsfull måste källgeometrierna befinna sig i samma koordinatsystem (ha samma SRID).

Notera
Denna funktion inkluderar automatiskt en jämförelse av begränsningsrutor som använder alla spatiala index som finns tillgängliga för geometrierna.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM ?

Tillgänglighet: 2.0.0

Exempel

Compare the same transformed point and line in both 3D and 2D. The 3D test fails because the Z difference matters even though the 2D footprints are close enough.

Code

```
WITH sample AS (
  SELECT ST_Transform('SRID=4326;POINT(-72.1235 42.3521 4)'::geometry, 2163) AS input_point ↩
    ,
         ST_Transform(
           'SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)'::geometry,
           2163
         ) AS input_line
)
SELECT ST_3DDWithin(input_point, input_line, 126.8) AS within_dist_3d,
       ST_DWithin(input_point, input_line, 126.8) AS within_dist_2d,
       input_point AS input_point,
       input_line AS input_line
FROM sample;
```

Output

```
-[ RECORD 1 ↩
    ]--+-----------------------------------------------------------------------------------------------------------------------------------------------------------------

within_dist_3d | f
within_dist_2d | t
input_point    | POINT Z (-11774128.9 5237160.2 4)
input_line     | LINESTRING Z (-11774339.5 5243556.8 15,-11773706.7 5215256.3 20)
```

Se även

ST_3DDFullyWithin, ST_DWithin, ST_DFullyWithin, ST_3DDistance, ST_Distance, ST_3DMaxDistance, ST_Transform

7.11.2.2 ST_3DDFullyWithin

ST_3DDFullyWithin — Testar om två 3D-geometrier är helt inom ett givet 3D-avstånd

Synopsis

boolean **ST_3DDFullyWithin**(geometry g1, geometry g2, double precision distance);

Beskrivning

Returnerar true om 3D-geometrierna är helt inom det angivna avståndet från varandra. Avståndet anges i enheter som definieras av geometriernas spatiala referenssystem. För att denna funktion ska vara meningsfull måste båda källgeometrierna ha samma koordinatprojektion och samma SRID.

Notera
Denna funktion inkluderar automatiskt en jämförelse av begränsningsrutor som använder alla spatiala index som finns tillgängliga för geometrierna.

Tillgänglighet: 2.0.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Exempel

This example compares distance-within predicates in 2D and 3D. The point is distance-within the line in 3D, but the full 3D line is not within the same distance of the point.

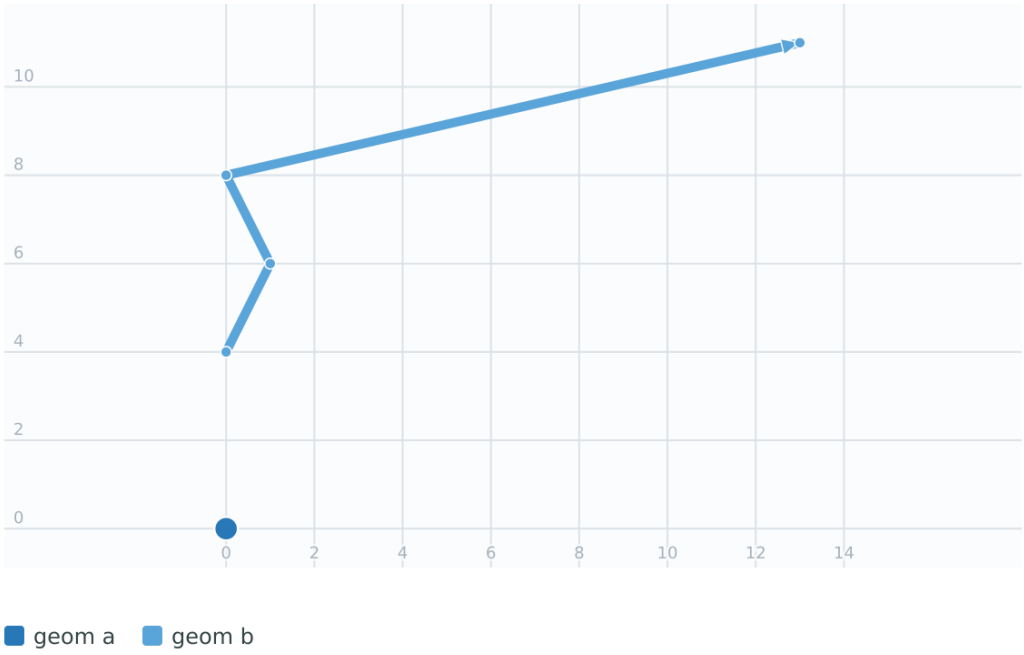
Code

```
WITH data AS (
  SELECT
    'POINT(1 1 2)'::geometry AS geom_a,
    'LINESTRING(1 5 2,2 7 20,1 9 100,14 12 3)'::geometry AS geom_b
)
SELECT
  ST_3DDFullyWithin(geom_a, geom_b, 10) AS d3dfullywithin10,
  ST_3DDWithin(geom_a, geom_b, 10) AS d3dwithin10,
  ST_DFullyWithin(geom_a, geom_b, 20) AS d2dfullywithin20,
  ST_3DDFullyWithin(geom_a, geom_b, 20) AS d3dfullywithin20
FROM data;
```

Output

```
d3dfullywithin10 | d3dwithin10 | d2dfullywithin20 | d3dfullywithin20
-----------------+-------------+------------------+-----------------
 f               | t           | t                | f
```

Figure



Se även

ST_3DDWithin, ST_DWithin, ST_DFullyWithin, ST_3DMaxDistance

7.11.2.3 ST_DFullyWithin

ST_DFullyWithin — Testar om en geometri är helt inom ett avstånd från en annan

Synopsis

boolean **ST_DFullyWithin**(geometry g1, geometry g2, double precision distance);

Beskrivning

Returnerar true om `g2` är helt inom `avståndet` från `g1`. Visuellt är villkoret sant om `g2` ligger inom en `avståndsbuffert` för `g1`. Avståndet anges i enheter som definieras av geometriernas spatiala referenssystem.

Notera
Denna funktion inkluderar automatiskt en jämförelse av begränsningsrutor som använder alla spatiala index som finns tillgängliga för geometrierna.

Tillgänglighet: 1.5.0

Ändrad: 3.5.0 : logiken bakom funktionen använder nu ett test av inneslutning inom en buffert, snarare än ST_MaxDistance-algoritmen. Resultaten kommer att skilja sig från tidigare versioner, men bör ligga närmare användarnas förväntningar.

Exempel

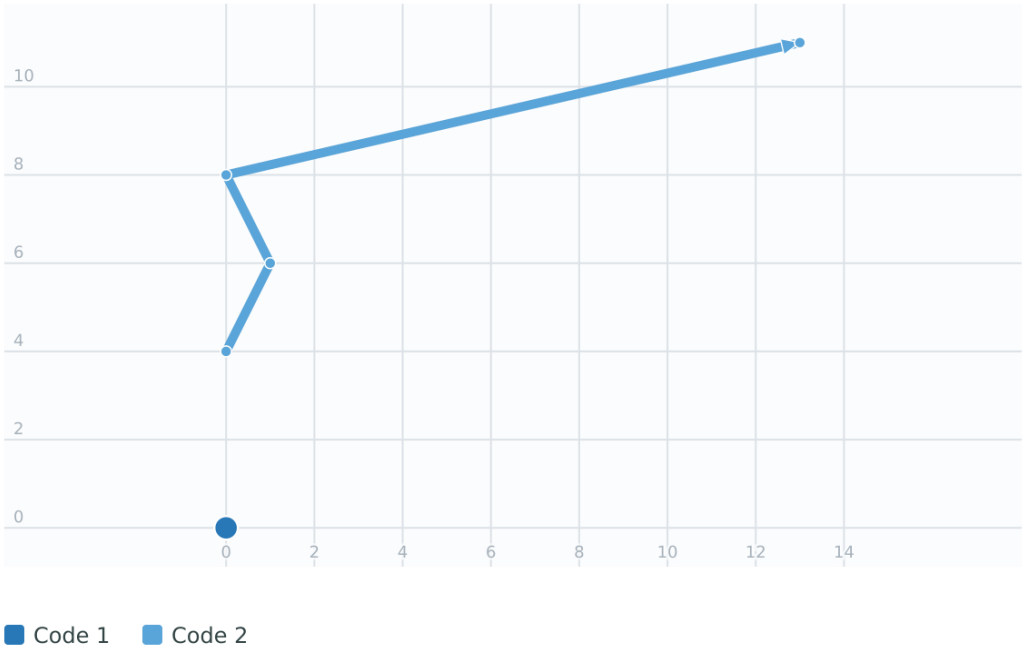
Code

```
SELECT
    ST_DFullyWithin(geom_a, geom_b, 10) AS DFullyWithin10,
    ST_DWithin(geom_a, geom_b, 10) AS DWithin10,
    ST_DFullyWithin(geom_a, geom_b, 20) AS DFullyWithin20
FROM (VALUES
    ('POINT(1 1)'::geometry, 'LINESTRING(1 5,2 7,1 9,14 12)'::geometry)
    ) AS v(geom_a, geom_b)
```

Output

```
dfullywithin10 | dwithin10 | dfullywithin20
---------------+-----------+----------------
 f             | t         | t
```

Figure



Se även

ST_MaxDistance, ST_DWithin, ST_3DDWithin, ST_3DDFullyWithin

7.11.2.4 ST_DWithin

ST_DWithin — Testar om två geometrier ligger inom ett givet avstånd

Synopsis

boolean **ST_DWithin**(geometry g1, geometry g2, double precision distance_of_srid);
boolean **ST_DWithin**(geography gg1, geography gg2, double precision distance_meters, boolean use_spher
= true);

Beskrivning

Returnerar true om geometrierna ligger inom ett givet avstånd

För geometri: Avståndet anges i enheter som definieras av geometriernas spatiala referenssystem. För att denna funktion ska vara meningsfull måste källgeometrierna vara i samma koordinatsystem (ha samma SRID).

För geografi: enheterna är i meter och avståndsmätning sker som standard med `use_spheroid = true`. För snabbare utvärdering, använd `use_spheroid = false` för att mäta på sfären.

Notera
Använd ST_3DDWithin för 3D-geometrier.

Notera
Detta funktionsanrop innehåller en jämförelse av begränsningsrutor som använder alla index som finns tillgängliga för geometrierna.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Tillgänglighet: 1.5.0 stöd för geografi infördes

Förbättrad: 2.1.0 förbättrad hastighet för geografi. Se Göra geografi snabbare för mer information.

Förbättrad: 2.1.0 stöd för krökta geometrier infördes.

Före 1.3 användes ST_Expand ofta i kombination med && och ST_Distance för att testa avstånd, och före 1.3.4 använde den här funktionen den logiken. Från och med 1.3.4 använder ST_DWithin en snabbare funktion för kortslutningsavstånd.

Exempel

This self-contained geometry example keeps the compared shapes visible while also returning the boolean results.

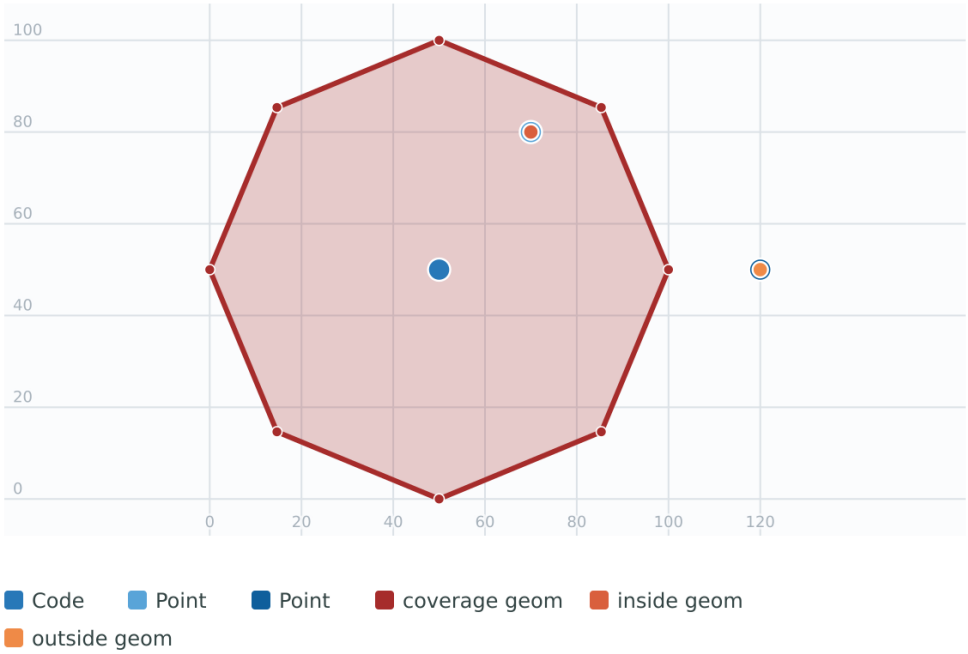
Code

```
WITH sample AS (
  SELECT ST_Buffer('POINT(100 90)'::geometry, 50, 'quad_segs=2') AS coverage,
         ST_Point(120, 120) AS inside_pt,
         ST_Point(170, 90) AS outside_pt
)
SELECT ST_DWithin(coverage, inside_pt, 0) AS inside_now,
       ST_DWithin(coverage, outside_pt, 10) AS outside_within_10,
       coverage AS coverage_geom,
       inside_pt AS inside_geom,
       outside_pt AS outside_geom
FROM sample;
```

Output

```
-[ RECORD 1 ]-----
inside_now        | t
outside_within_10 | f
coverage_geom     | POLYGON((150 90,135 55,100 40,65 55,50 90,65 125,100 140,135 125,150 90))
inside_geom       | POINT(120 120)
outside_geom      | POINT(170 90)
```

Figure



Find the nearest hospital to each school. That is within 3000 units of the school. We do an ST_DWithin search to utilize indexes to limit our search list. That the non-indexable ST_Distance needs to process. If the units of the spatial reference is meters then units would be meters.

Code

```
SELECT DISTINCT ON (s.gid) s.gid, s.school_name, s.geom, h.hospital_name
  FROM schools s
    LEFT JOIN hospitals h ON ST_DWithin(s.geom, h.geom, 3000)
  ORDER BY s.gid, ST_Distance(s.geom, h.geom);
```

The schools with no close hospitals. Find all schools with no hospital within 3000 units away from the school. Units are in units of spatial reference (for example meters, feet, or degrees).

Code

```
SELECT s.gid, s.school_name
  FROM schools s
    LEFT JOIN hospitals h ON ST_DWithin(s.geom, h.geom, 3000)
  WHERE h.gid IS NULL;
```

Find locations more than 3000 units away from their nearest hospital. Use `NOT EXISTS` so that a location is returned only when no nearby hospital exists.

Code

```
SELECT l.location_name
  FROM locations l
  WHERE NOT EXISTS (
    SELECT 1
    FROM hospitals h
    WHERE ST_DWithin(l.geom, h.geom, 3000)
  );
```

Find broadcasting towers that a receiver with limited range can receive. Data is geometry in Spherical Mercator (SRID=3857), ranges are approximate.

Code

```
CREATE INDEX ON broadcasting_towers USING gist (geom);
CREATE INDEX ON broadcasting_towers USING gist (ST_Expand(geom, sending_range));
```

Query towers that a 4-kilometer receiver in Minsk Hackerspace can get. Note: two conditions are required because the shorter `LEAST(b.sending_range, 4000)` will not use an index.

Code

```
SELECT b.tower_id, b.geom
  FROM broadcasting_towers b
  WHERE ST_DWithin(b.geom, 'SRID=3857;POINT(3072163.4 7159374.1)', 4000)
    AND ST_DWithin(b.geom, 'SRID=3857;POINT(3072163.4 7159374.1)', b.sending_range);
```

Se även

ST_Distance, ST_3DDWithin

7.11.2.5 ST_PointInsideCircle

ST_PointInsideCircle — Testar om en punktgeometri ligger inom en cirkel definierad av centrum och radie

Synopsis

boolean **ST_PointInsideCircle**(geometry a_point, float center_x, float center_y, float radius);

Beskrivning

Returnerar true om geometrin är en punkt och ligger inom cirkeln med centrum `center_x,center_y` och radie `radius`.

Varning
Använder inte spatiala index. Använd ST_DWithin istället.

Tillgänglighet: 1.2

Ändrad: 2.2.0 I tidigare versioner kallades detta ST_Point_Inside_Circle

Exempel

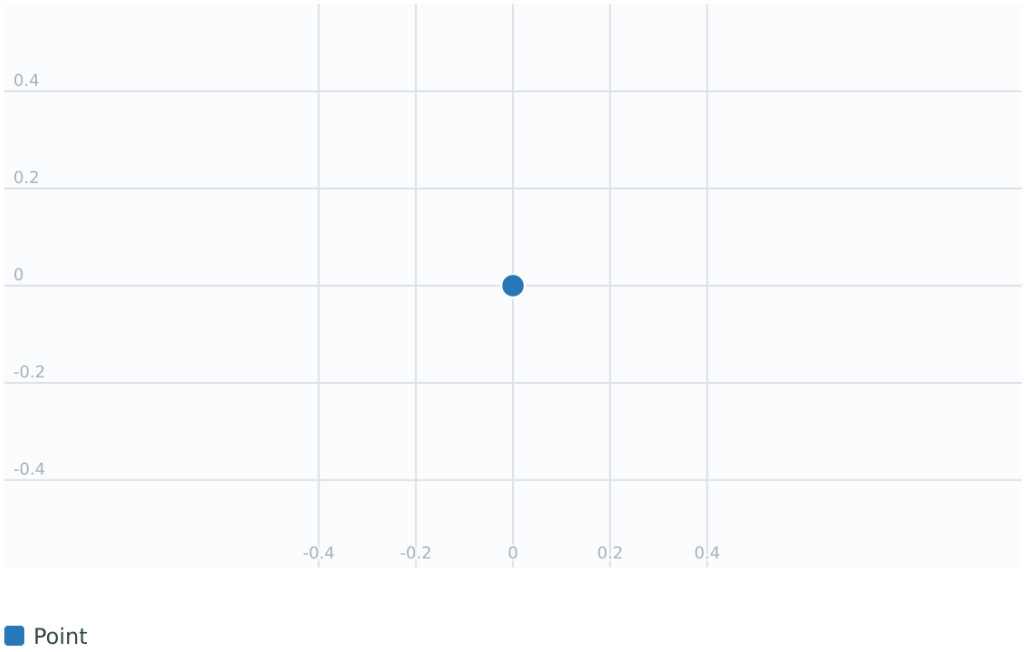
Code

```
SELECT ST_PointInsideCircle(ST_Point(1, 2), 0.5, 2, 3);
```

Output

```
t
```

Figure



Se även

ST_DWithin

7.12 Mätningsfunktioner

7.12.1 ST_Area

ST_Area — Returnerar arean för en polygonal geometri.

Synopsis

float **ST_Area**(geometry g1);
float **ST_Area**(geography geog, boolean use_spheroid = true);

Beskrivning

Returnerar arean för en polygonal geometri. För geometrityper beräknas en 2D kartesisk (plan) area, med enheter som anges av SRID. För geograftyper bestäms arean som standard på en sfäroid med

enheter i kvadratmeter. Om du vill beräkna arean med den snabbare men mindre exakta sfäriska modellen använder du `ST_Area(geog,false)`..

Förbättrad: 2.0.0 - stöd för 2D polyhedrala ytor infördes.

Förbättrad: 2.2.0 - mätning på sfäroid utförd med GeographicLib för förbättrad noggrannhet och robusthet. Kräver PROJ >= 4.9.0 för att dra nytta av den nya funktionen.

Ändrad: 3.0.0 - är inte längre beroende av SFCGAL.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.1.2, 9.5.3

Denna funktion stöder polyedriska ytor.

Notera
För polyedriska ytor stöds endast 2D-polyedriska ytor (inte 2,5D). För 2,5D kan svaret bli något annat än noll, men bara för de ytor som ligger helt i XY-planet.

Exempel

Returnera yta i kvadratfot för en tomt i Massachusetts och multiplicera med omvandling för att få kvadratmeter. Observera att detta är i kvadratfot eftersom EPSG:2249 är Massachusetts State Plane Feet

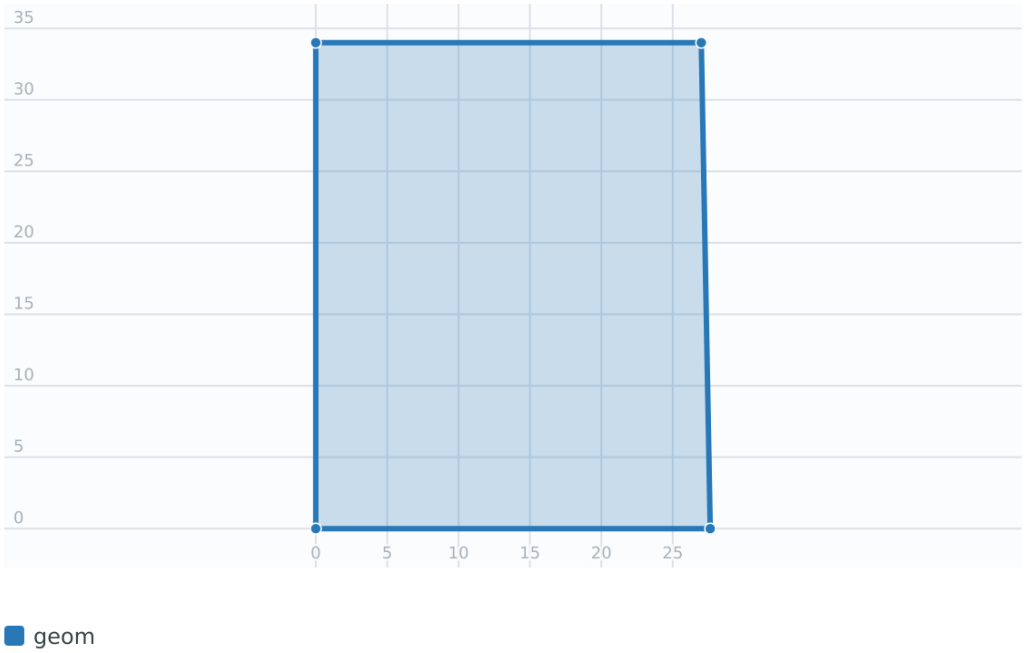
Code

```
SELECT ST_Area(geom) AS sqft,
       ST_Area(geom) * 0.3048 ^ 2 AS sqm
FROM (
  SELECT 'SRID=2249;POLYGON((743238 2967416,743238 2967450,
          743265 2967450,743265.625 2967416,743238 2967416))'::geometry AS geom
) AS subquery;
```

Output

```
sqft   |     sqm
--------+-------------
928.625 | 86.27208552
```

Figure



Return area square feet and transform to Massachusetts state plane meters (EPSG:26986) to get square meters. Note this is in square feet because 2249 is Massachusetts State Plane Feet and transformed area is in square meters since EPSG:26986 is state plane Massachusetts meters.

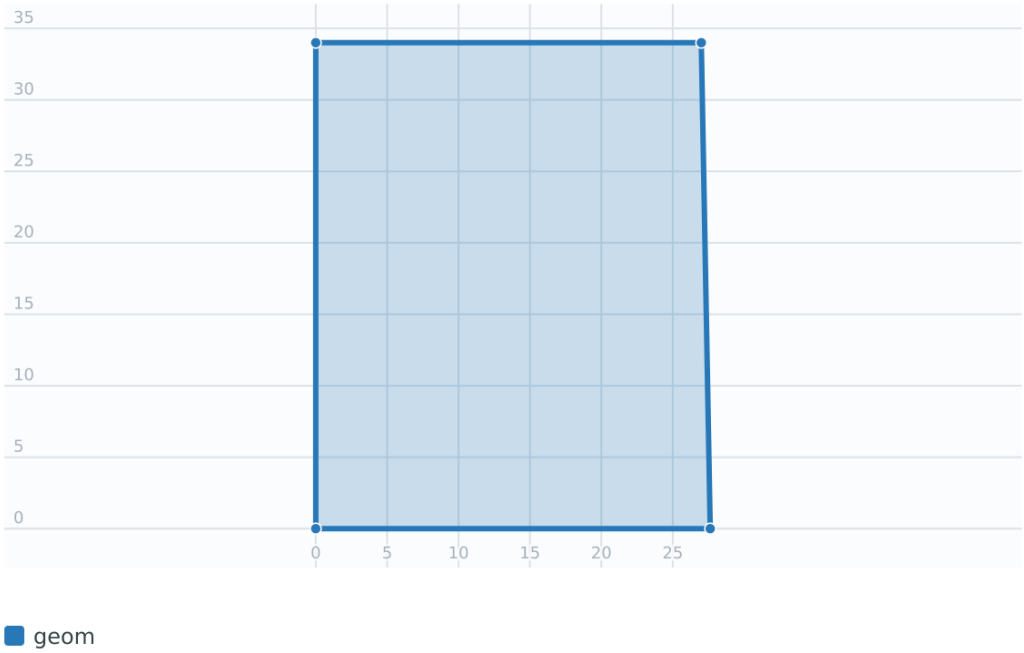
Code

```
SELECT ST_Area(geom) AS sqft,
       round(ST_Area(ST_Transform(geom, 26986))::numeric, 6) AS sqm
FROM (
  SELECT 'SRID=2249;POLYGON((743238 2967416,743238 2967450,
          743265 2967450,743265.625 2967416,743238 2967416))'::geometry AS geom
) AS subquery;
```

Output

```
sqft   |    sqm
--------+-----------
928.625 | 86.272431
```

Figure



Returnera area kvadratfot och kvadratmeter med hjälp av datatypen geography. Observera att vi omvandlar vår geometri till geografi (innan du kan göra det måste du se till att din geometri är i WGS 84 long lat 4326). Geografi mäter alltid i meter. Detta är bara för demonstration för att jämföra. Normalt kommer din tabell redan att lagras i datatypen Geography.

Code

```
SELECT round((ST_Area(geog) / 0.3048 ^ 2)::numeric, 2) AS sqft_spheroid,
       round((ST_Area(geog, false) / 0.3048 ^ 2)::numeric, 2) AS sqft_sphere,
       round(ST_Area(geog)::numeric, 2) AS sqm_spheroid
FROM (
  SELECT ST_Transform(
           'SRID=2249;POLYGON((743238 2967416,743238 2967450,
           743265 2967450,743265.625 2967416,743238 2967416))'::geometry,
           4326
         )::geography AS geog
) AS subquery;
```

Output

```
sqft_spheroid | sqft_sphere | sqm_spheroid
---------------+-------------+--------------
        928.68 |      926.61 |        86.28
```

Om dina data redan finns i geografin:

Code

```
SELECT ST_Area(geog) / 0.3048 ^ 2 AS sqft,
               ST_Area(the_geog) AS sqm
        FROM somegeogtable;
```

Se även

CG_3DArea, ST_GeomFromText, ST_GeographyFromText, ST_SetSRID, ST_Transform

7.12.2 ST_Azimuth

ST_Azimuth — Returnerar den nordbaserade azimuten för en linje mellan två punkter.

Synopsis

float **ST_Azimuth**(geometry origin, geometry target);
float **ST_Azimuth**(geography origin, geography target);

Beskrivning

Returnerar målpunktens azimut i radianer från ursprungspunkten, eller NULL om de två punkterna sammanfaller. Azimutvinkeln är en positiv medurs vinkel som refereras från den positiva Y-axeln (geometri) eller den norra meridianen (geografi): North = 0; Northeast = π/4; East = π/2; Southeast = 3π/4; South = π; Southwest 5π/4; West = 3π/2; Northwest = 7π/4.

För den geografiska typen är azimutlösningen känd som det inversa geodetiska problemet.

Azimuten är ett matematiskt koncept definierat som vinkeln mellan en referensvektor och en punkt, med vinkelenheter i radianer. Resultatvärdet i radianer kan konverteras till grader med hjälp av PostgreSQL-funktionen `grader ()`..

For offsetting lines relative to their direction, use ST_OffsetCurve. For projecting a point by distance and bearing, use ST_Project.

Tillgänglighet: 1.1.0

Förbättrad: 2.0.0 stöd för geografi infördes.

Förbättrad: 2.2.0 mätning på sfäroid utförd med GeographicLib för förbättrad noggrannhet och robusthet. Kräver PROJ >= 4.9.0 för att dra nytta av den nya funktionen.

Exempel

Geometri Azimut i grader

Code

```
SELECT degrees(ST_Azimuth(ST_Point(25, 45), ST_Point(75, 100))) AS degA_B,
       degrees(ST_Azimuth(ST_Point(75, 100), ST_Point(25, 45) )) AS degB_A;
```

Output

```
dega_b       |     degb_a
-----------------+-----------------
42.27368900609371 | 222.27368900609372
```

Compass direction labels can be computed from the azimuth.

Code

```
WITH directions(deg) AS (VALUES
  (0), (45), (90), (135), (180), (225), (270), (315)
),
bearings AS (
  SELECT deg,
         degrees(ST_Azimuth(ST_Point(0, 0),
                            ST_MakePoint(sin(radians(deg)),
                                         cos(radians(deg))))) AS azimuth
  FROM directions
```

```
)
SELECT deg,
       round(azimuth::numeric, 1) AS azimuth,
       (ARRAY['N', 'NE', 'E', 'SE', 'S', 'SW', 'W', 'NW', 'N'])
         [floor((azimuth + 22.5) / 45)::int + 1] AS direction
FROM bearings
ORDER BY deg;
```

Output

```
deg | azimuth | direction
-----+---------+-----------
   0 |     0.0 | N
  45 |    45.0 | NE
  90 |    90.0 | E
 135 |   135.0 | SE
 180 |   180.0 | S
 225 |   225.0 | SW
 270 |   270.0 | W
 315 |   315.0 | NW
```

Azimuth from Point(25, 45) to Point(75, 100), shown against the positive Y axis (north).

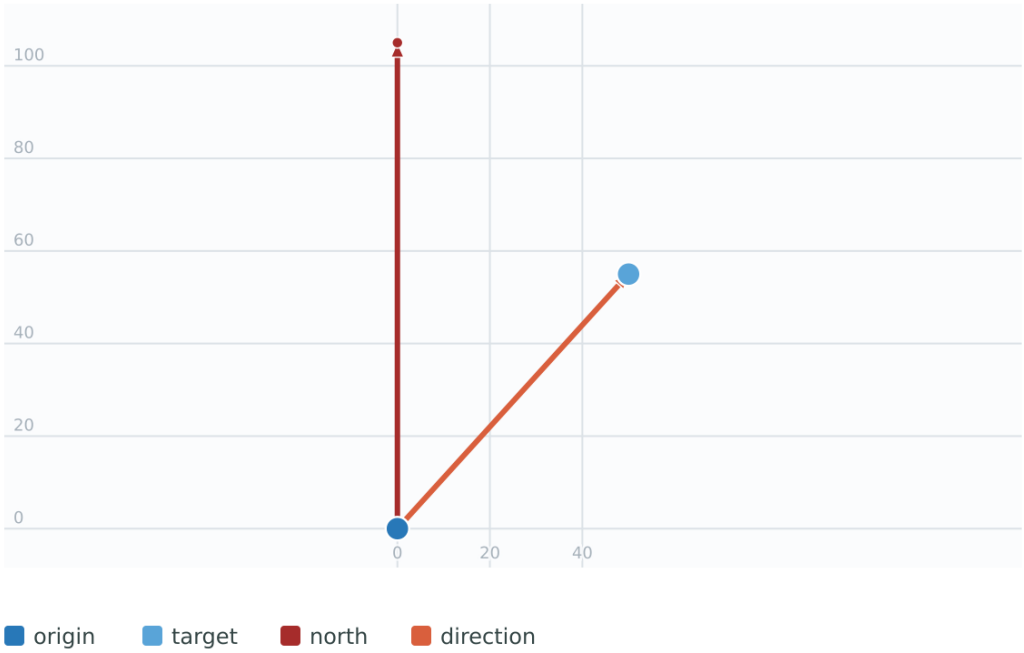
Code

```
WITH points AS (
  SELECT 'POINT(25 45)'::geometry AS origin,
         'POINT(75 100)'::geometry AS target
)
SELECT round(degrees(ST_Azimuth(origin, target))::numeric, 6) AS degrees,
       ST_MakeLine(origin, ST_Translate(origin, 0, 105)) AS north,
       ST_MakeLine(origin, target) AS direction
FROM points;
```

Output

```
42.273689 | LINESTRING(25 45,25 150) | LINESTRING(25 45,75 100)
```

Figure



Reverse azimuth from Point(75, 100) to Point(25, 45).

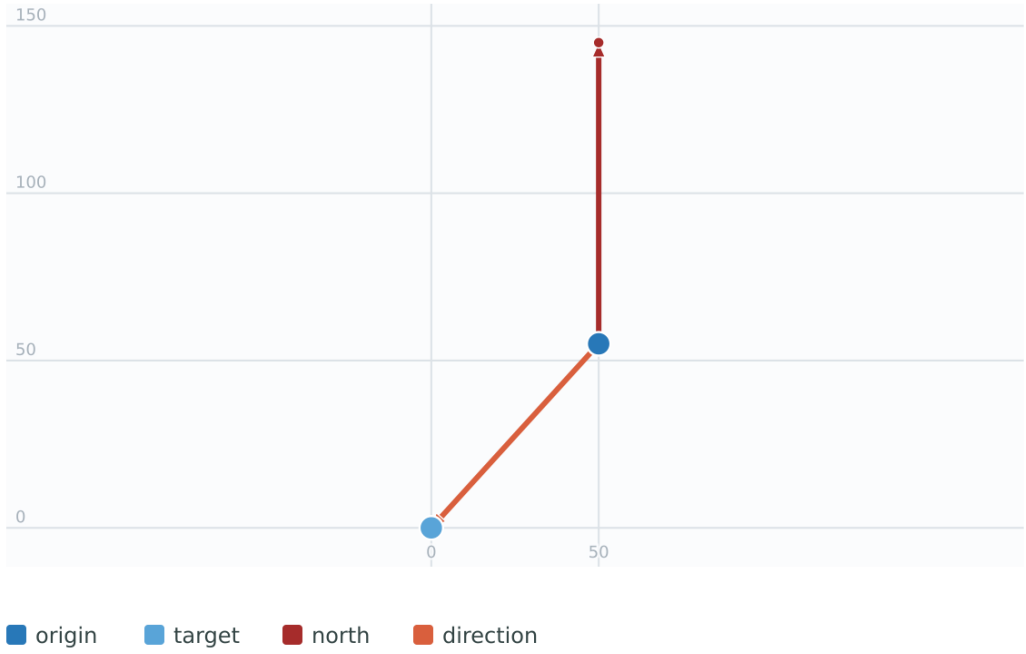
Code

```
WITH points AS (
  SELECT 'POINT(75 100)'::geometry AS origin,
         'POINT(25 45)'::geometry AS target
)
SELECT round(degrees(ST_Azimuth(origin, target))::numeric, 6) AS degrees,
       ST_MakeLine(origin, ST_Translate(origin, 0, 90)) AS north,
       ST_MakeLine(origin, target) AS direction
FROM points;
```

Output

```
222.273689 | LINESTRING(75 100,75 190) | LINESTRING(75 100,25 45)
```

Figure



Se även

ST_Angle, ST_Point, ST_Translate, ST_Project, PostgreSQL Matematiska funktioner

7.12.3 ST_Angle

ST_Angle — Returnerar vinkeln mellan två vektorer som definieras av 3 eller 4 punkter, eller 2 linjer.

Synopsis

float **ST_Angle**(geometry point1, geometry point2, geometry point3, geometry point4);
float **ST_Angle**(geometry line1, geometry line2);

Beskrivning

Beräknar den medurs vridna vinkeln mellan två vektorer.

Variant 1: beräknar den vinkel som omsluts av punkterna P1-P2-P3. Om en 4:e punkt anges beräknas vinkeln mellan punkterna P1-P2 och P3-P4

Variant 2: beräknar vinkeln mellan två vektorer S1-E1 och S2-E2, definierade av start- och slutpunkterna för indatalinjerna

Resultatet är en positiv vinkel mellan 0 och 2π radianer. Radiansresultatet kan konverteras till grader med hjälp av PostgreSQL-funktionen `degrees()`.

Observera att `ST_Angle(P1,P2,P3) = ST_Angle(P2,P1,P2,P3)`.

Tillgänglighet: 2.5.0

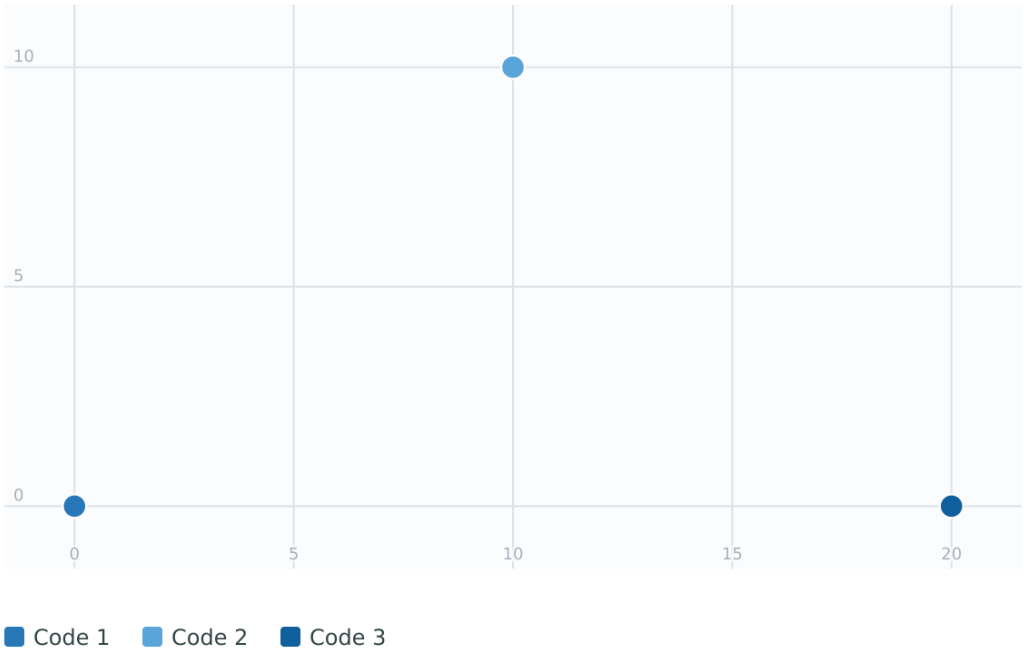
Exempel

Vinkel mellan tre punkter

Code

```
SELECT degrees(ST_Angle('POINT(0 0)', 'POINT(10 10)', 'POINT(20 0)'));
```

Figure

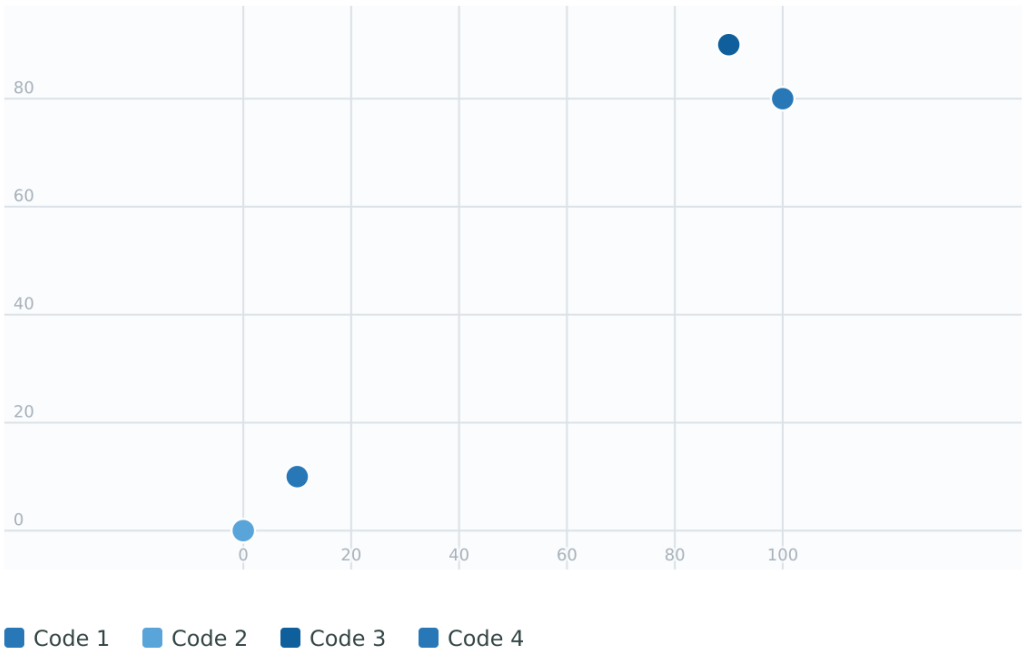


Vinkel mellan vektorer definierade av fyra punkter

Code

```
SELECT degrees(ST_Angle('POINT (10 10)', 'POINT (0 0)', 'POINT(90 90)', 'POINT (100 80)'));
```

Figure

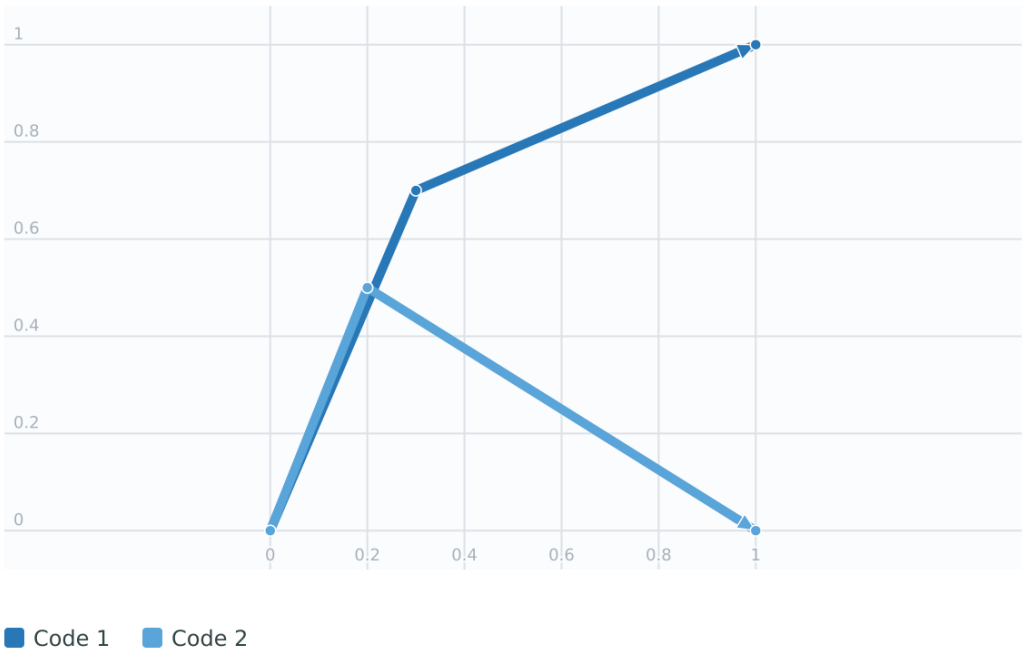


Vinkel mellan vektorer som definieras av linjernas start- och slutpunkter

Code

```
SELECT degrees(ST_Angle('LINESTRING(0 0,0.3 0.7,1 1)', 'LINESTRING(0 0,0.2 0.5,1 0)'));
```

Figure



Se även

ST_Azimuth

7.12.4 ST_ClosestPoint

ST_ClosestPoint — Returnerar den 2D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste linjen från den ena geometrin till den andra.

Synopsis

geometry **ST_ClosestPoint**(geometry geom1, geometry geom2);
geography **ST_ClosestPoint**(geography geom1, geography geom2, boolean use_spheroid = true);

Beskrivning

Returnerar den 2-dimensionella punkt på `geom1` som ligger närmast `geom2`. Detta är den första punkten på den kortaste linjen mellan geometrierna (enligt beräkning av ST_ShortestLine).

Notera
Om du har en 3D-geometri kanske du föredrar att använda ST_3DClosestPoint.

Förbättrad: 3.4.0 - Stöd för geografi.

Tillgänglighet: 1.5.0

Exempel

Närmaste punkt för en Point och en LineString är själva punkten. Närmaste punkt för en LineString och en Point är en punkt på linjen.

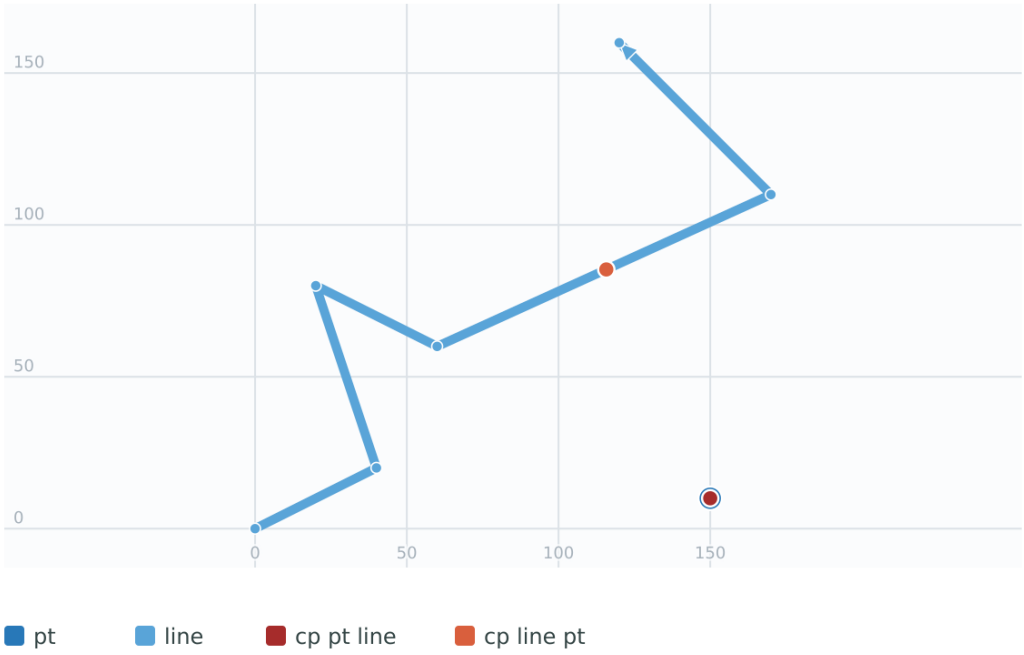
Code

```
SELECT ST_ClosestPoint(pt, line) AS cp_pt_line, ST_ClosestPoint(line, pt) AS cp_line_pt
    FROM (SELECT 'POINT (160 40)'::geometry AS pt,
                 'LINESTRING (10 30,50 50,30 110,70 90,180 140,130 190)'::geometry AS line
    ) AS t;
```

Output

```
cp_pt_line   |               cp_line_pt
---------------+----------------------------------------
 POINT(160 40)  | POINT(125.75342465753425 115.34246575342466)
```

Figure

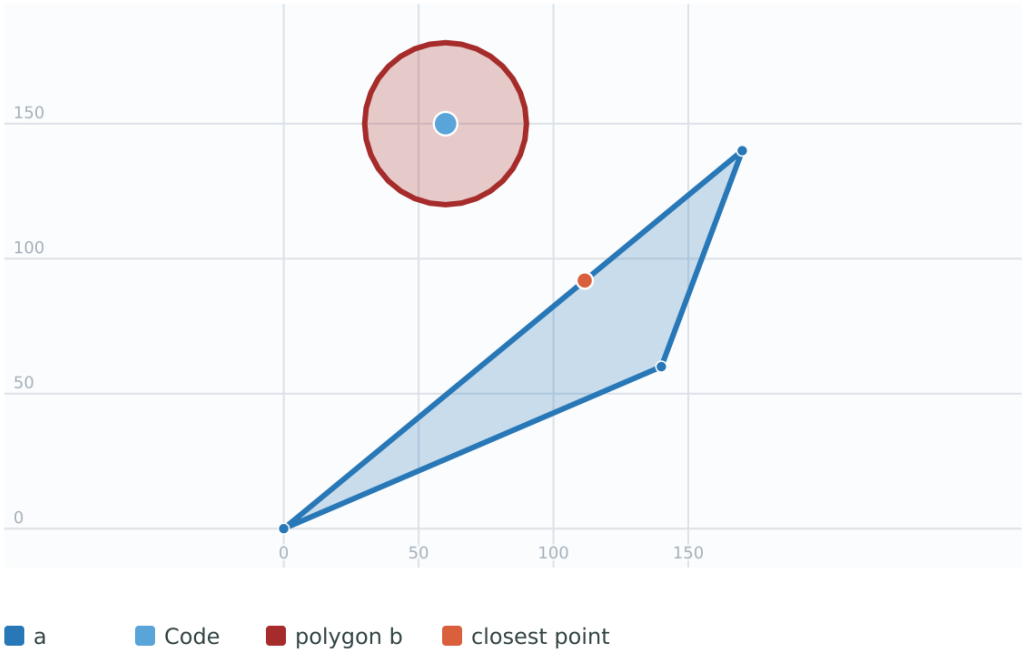


Find the closest point on polygon A to polygon B.

Code

```
WITH source AS (
  SELECT 'POLYGON ((190 150,20 10,160 70,190 150))'::geometry AS a,
         ST_Buffer('POINT(80 160)'::geometry, 30) AS b
)
SELECT b AS polygon_b,
       ST_ClosestPoint(a, b) AS closest_point
FROM source;
```

Figure



Se även

ST_3DClosestPoint, ST_Distance, ST_LongestLine, ST_ShortestLine, ST_MaxDistance

7.12.5 ST_3DClosestPoint

ST_3DClosestPoint — Returnerar den 3D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste 3D-linjen.

Synopsis

geometry **ST_3DClosestPoint**(geometry g1, geometry g2);

Beskrivning

Returnerar den 3-dimensionella punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste 3D-linjen. 3D-längden på den kortaste 3D-linjen är 3D-avståndet.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Tillgänglighet: 2.0.0

Ändrad: 2.2.0 - om 2 2D-geometrier matas in returneras en 2D-punkt (istället för det gamla beteendet som antog 0 för saknad Z). Vid 2D och 3D antas Z inte längre vara 0 för saknad Z.

Exempel

Compare the 3D closest point to the 2D closest point for a LineString and a Point.

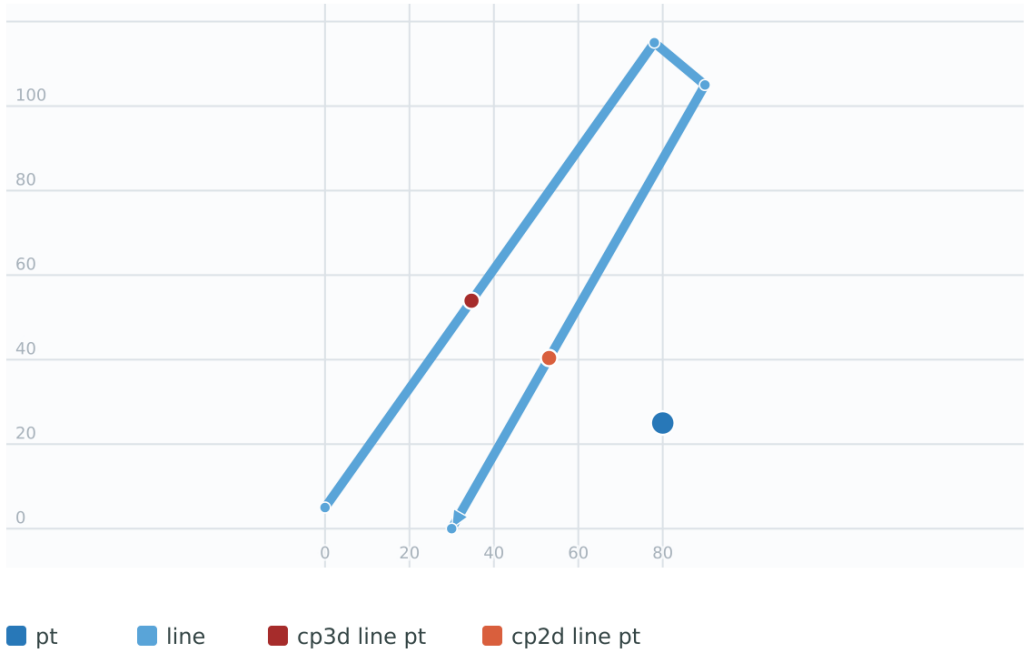
Code

```
WITH input AS (
  SELECT 'POINT(100 100 30)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 1000)'::geometry AS line
)
SELECT ST_3DClosestPoint(line, pt) AS cp3d_line_pt,
       ST_ClosestPoint(line, pt) AS cp2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------
cp3d_line_pt   | POINT(54.69937988676193 128.93502291722837 11.547586950660556)
cp2d_line_pt   | POINT(73.07692307692307 115.38461538461539)
```

Figure



Compare the 3D closest point to the 2D closest point for a LineString and a MultiPoint.

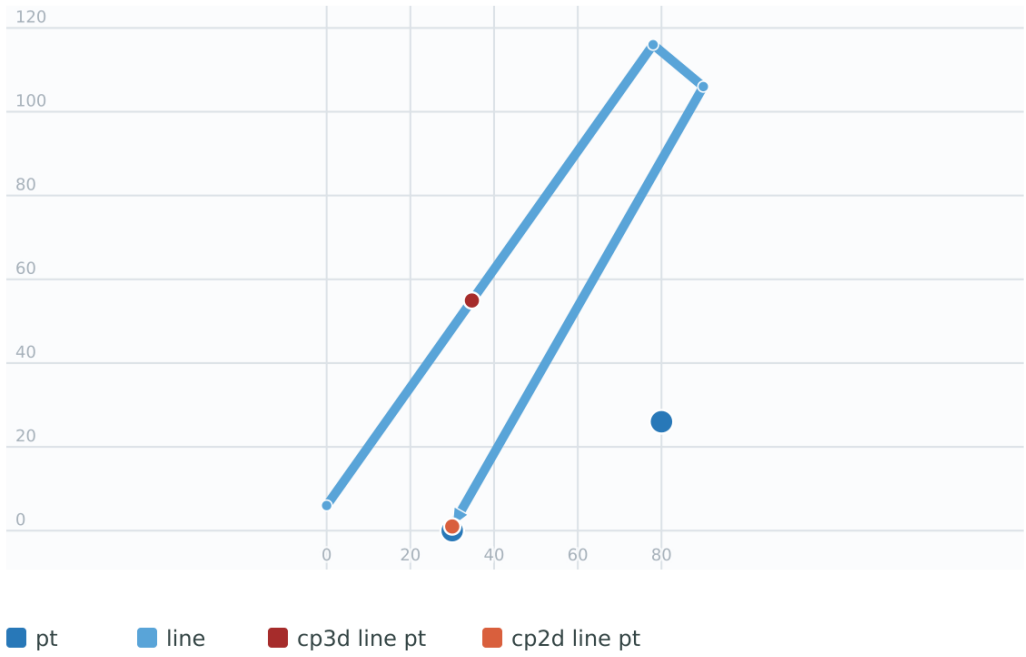
Code

```
WITH input AS (
  SELECT 'MULTIPOINT(100 100 30,50 74 1000)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 900)'::geometry AS line
)
SELECT ST_3DClosestPoint(line, pt) AS cp3d_line_pt,
       ST_ClosestPoint(line, pt) AS cp2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------
cp3d_line_pt   | POINT(54.69937988676193 128.93502291722837 11.547586950660556)
cp2d_line_pt   | POINT(50 75)
```

Figure



Compare the 3D closest point to the 2D closest point for a Polygon and a MultiLineString.

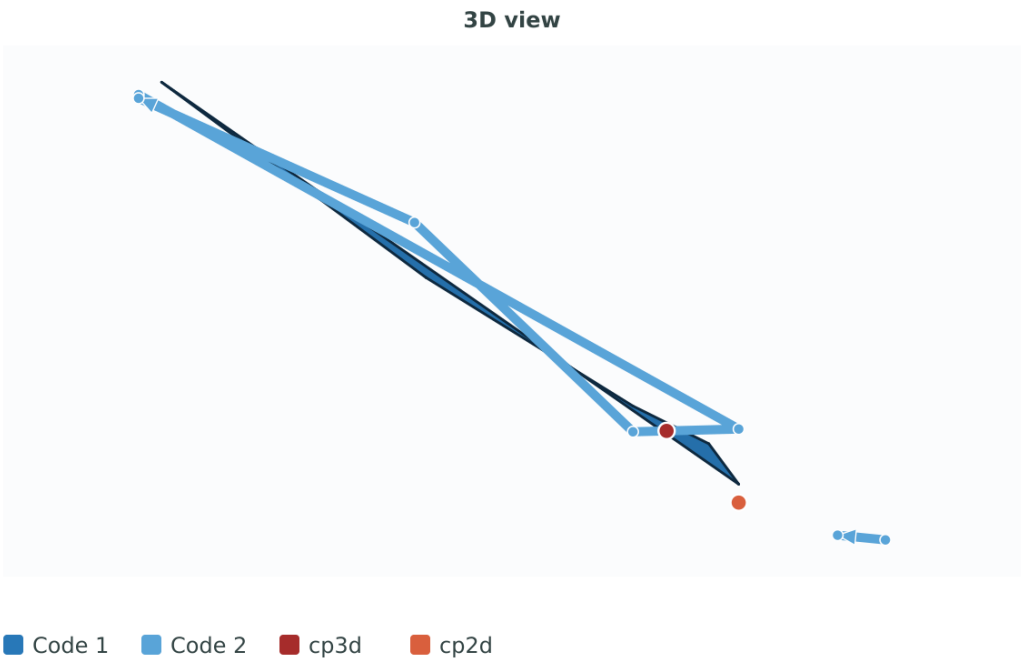
Code

```
WITH input AS (
  SELECT ST_GeomFromEWKT(
           'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))'
         ) AS poly,
         ST_GeomFromEWKT(
           'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),
                            (1 10 2,5 20 1))'
         ) AS mline
)
SELECT ST_3DClosestPoint(poly, mline) AS cp3d,
       ST_ClosestPoint(poly, mline) AS cp2d
FROM input;
```

Output

```
-[ RECORD 1 ]--+-------------------------------------
cp3d           | POINT(40.45454545454545 53.63636363636364 5)
cp2d           | POINT(20 40)
```

Figure



Se även

ST_AsEWKT, ST_ClosestPoint, ST_3DDistance, ST_3DShortestLine

7.12.6 ST_Distance

ST_Distance — Returnerar avståndet mellan två geometri- eller geografivärden.

Synopsis

float **ST_Distance**(geometry g1, geometry g2);
float **ST_Distance**(geography geog1, geography geog2, boolean use_spheroid = true);

Beskrivning

För geometry -typer returneras det minsta kartesiska (plana) 2D-avståndet mellan två geometrier, i projicerade enheter (spatiala ref-enheter).

För geography -typer är standardvärdet att returnera det minsta geodetiska avståndet mellan två geografiska platser i meter, beräkna på den sfäroid som bestäms av SRID. Om `use_spheroid` är false används en snabbare sfärisk beräkning.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.23

Denna metod stöder cirkulära strängar och kurvor.

Tillgänglighet: 1.5.0 Stöd för geografi infördes i 1.5. Hastighetsförbättringar för planar för att bättre hantera stora eller många vertexgeometrier

Förbättrad: 2.1.0 förbättrad hastighet för geografi. Se Göra geografi snabbare för mer information.

Förbättrad: 2.1.0 - stöd för krökta geometrier infördes.

Förbättrad: 2.2.0 - mätning på sfäroid utförd med GeographicLib för förbättrad noggrannhet och robusthet. Kräver PROJ >= 4.9.0 för att dra nytta av den nya funktionen.

Ändrad: 3.0.0 - är inte längre beroende av SFCGAL.

Exempel på geometri

Geometriexempel - enheter i plana grader 4326 är WGS 84 long lat, enheter är grader.

Code

```
SELECT ST_Distance(
'SRID=4326;POINT(-72.1235 42.3521)'::geometry,
'SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry );
```

Geometriexempel - enheter i meter (SRID: 3857, proportionell mot pixlar på populära webbkartor). Även om värdet är felaktigt kan närliggande värden jämföras korrekt, vilket gör det till ett bra val för algoritmer som KNN eller KMEANS.

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 3857),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 3857) );
```

Geometriexempel - enheter i meter (SRID: 3857 som ovan, men korrigerad med cos(lat) för att ta hänsyn till distorsion)

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 3857),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 3857)
        ) * cosd(42.3521);
```

Geometriexempel - enheter i meter (SRID: 26986 Massachusetts state plane meter) (mest exakt för Massachusetts)

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 26986),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 26986) );
```

Geometriexempel - enheter i meter (SRID: 2163 US National Atlas Equal area) (minst exakt)

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 2163),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 2163) );
```

Geografiska exempel

Samma som geometriexemplet men notera enheter i meter - använd sfär för något snabbare och mindre exakta beräkningar.

Code

```
SELECT ST_Distance(gg1, gg2) As spheroid_dist, ST_Distance(gg1, gg2, false) As sphere_dist
FROM (SELECT
    'SRID=4326;POINT(-72.1235 42.3521)'::geography as gg1,
    'SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geography as gg2
    ) As foo  ;
```

Output

```
spheroid_dist   |   sphere_dist
-----------------+-----------------
 123.802076746848 | 123.475736916397
```

Se även

ST_3DDistance, ST_DWithin, ST_DistanceSphere, ST_DistanceSpheroid, ST_MaxDistance, ST_HausdorffDis
ST_FrechetDistance, ST_Transform

7.12.7 ST_3DDistance

ST_3DDistance — Returnerar det kartesiska 3D-minsta avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.

Synopsis

float **ST_3DDistance**(geometry g1, geometry g2);

Beskrivning

Returnerar det 3-dimensionella minsta kartesiska avståndet mellan två geometrier i projicerade enheter (spatiala ref-enheter).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM ISO/IEC 13249-3

Tillgänglighet: 2.0.0

Ändrad: 2.2.0 - När det gäller 2D och 3D antas Z inte längre vara 0 om Z saknas.

Ändrad: 3.0.0 - SFCGAL-versionen borttagen

Exempel

Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point and line compared 2D point and line) Note that currently no vertical datum support so Z is not transformed and assumed to be same units as final.

Code

```
SELECT ST_3DDistance(
            ST_Transform('SRID=4326;POINT(-72.1235 42.3521 4)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)':: ↩
    geometry, 2163)
            ) As dist_3d,
            ST_Distance(
                ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)':: ↩
    geometry, 2163)
            ) As dist_2d;
```

Output

```
dist_3d      |     dist_2d
------------------+-----------------
 127.295059324629 | 126.66425605671
```

Multilinestring and polygon both 3D and 2D distance. Same example as 3D closest point example.

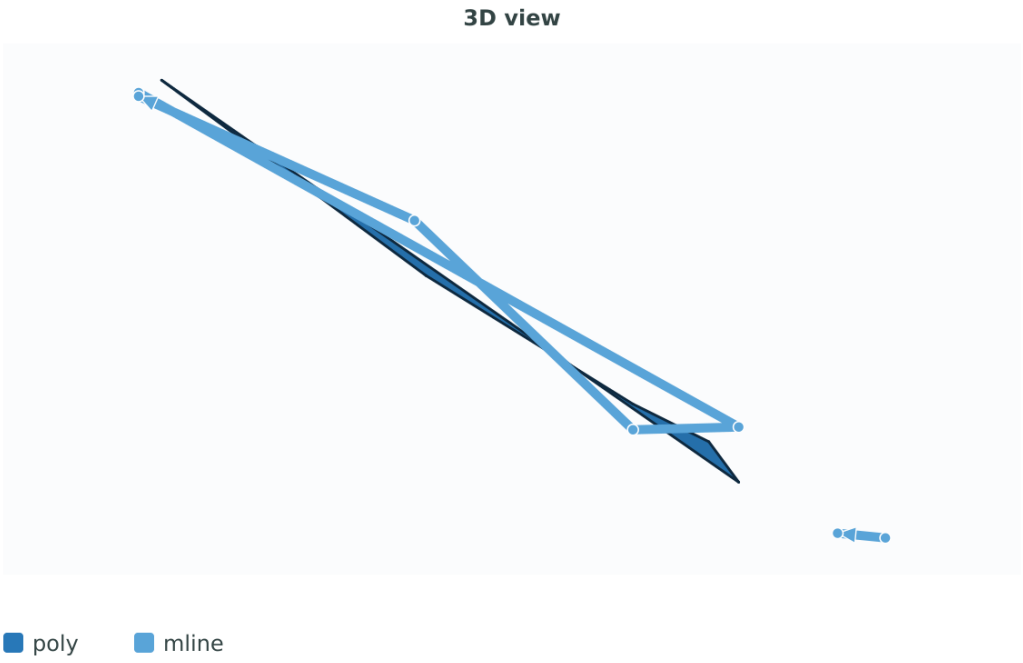
Code

```
SELECT ST_3DDistance(poly, mline) As dist3d,
    ST_Distance(poly, mline) As dist2d
        FROM (SELECT  'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))':: ↩
    geometry as poly,
                'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),(1 10 2,5 ↩
     20 1))'::geometry as mline) as foo;
```

Output

```
dist3d       | dist2d
------------------+--------
                0 |      0
```

Figure



Se även

ST_Distance, ST_3DClosestPoint, ST_3DDWithin, ST_3DMaxDistance, ST_3DShortestLine, ST_Transform

7.12.8 ST_DistanceSphere

ST_DistanceSphere — Returnerar minsta avstånd i meter mellan två lon/lat-geometrier med hjälp av en sfärisk jordmodell.

Synopsis

float **ST_DistanceSphere**(geometry geomlonlatA, geometry geomlonlatB, float8 radius=6371008);

Beskrivning

Returnerar minsta avstånd i meter mellan två lon/lat-punkter. Använder en sfärisk jord och en radie som härrör från den sfäroid som definieras av SRID. Snabbare än ST_DistanceSpheroid, men mindre exakt. PostGIS-versioner före 1.5 implementeras endast för punkter.

Tillgänglighet: 1.5 - stöd för andra geometrityper än punkter infördes. Tidigare versioner fungerar bara med punkter.

Ändrad: 2.2.0 I tidigare versioner brukade detta kallas ST_Distance_Sphere

Exempel

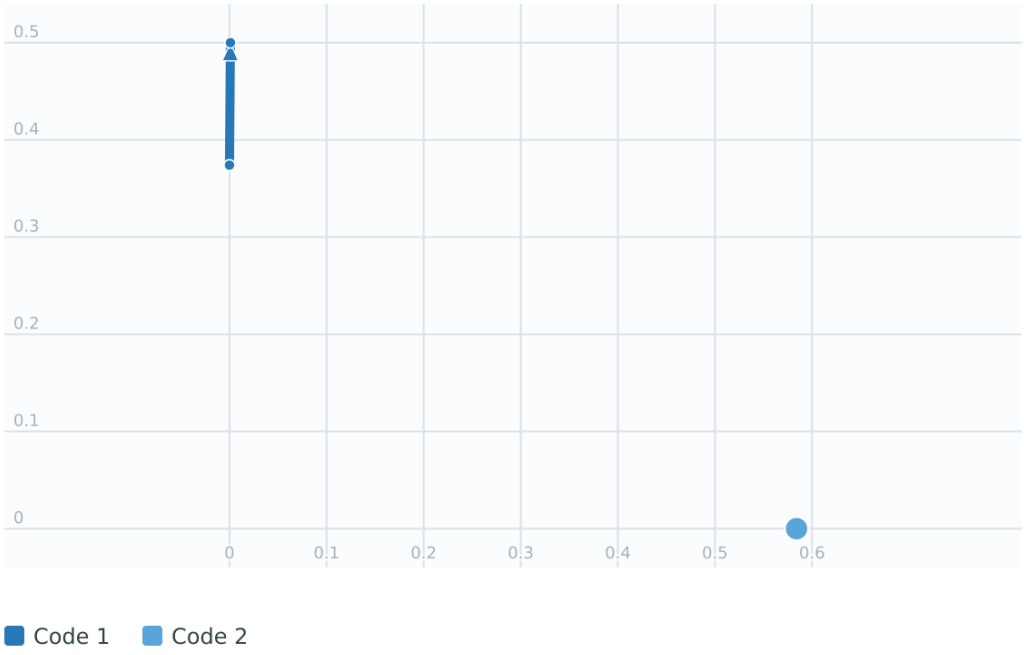
Code

```
WITH sample AS (
    SELECT
        ST_GeomFromText('LINESTRING(-118.584 38.374,-118.583 38.5)', 4326) AS geom,
        ST_GeomFromText('POINT(-118 38)', 4326) AS pt
)
SELECT
    round(ST_DistanceSphere(ST_Centroid(geom), pt)::numeric, 2) AS dist_meters,
    round(ST_Distance(
        ST_Transform(ST_Centroid(geom), 32611),
        ST_Transform(pt, 32611)
    )::numeric, 2) AS dist_utm11_meters,
    round(ST_Distance(ST_Centroid(geom), pt)::numeric, 5) AS dist_degrees,
    round(ST_Distance(
        ST_Transform(geom, 32611),
        ST_Transform(pt, 32611)
    )::numeric, 2) AS min_dist_line_point_meters
FROM sample;
```

Output

```
dist_meters | dist_utm11_meters | dist_degrees | min_dist_line_point_meters
------------+-------------------+--------------+---------------------------
    70424.71 |          70438.00 |      0.72900 |                   65871.18
```

Figure



Se även

ST_Distance, ST_DistanceSpheroid

7.12.9 ST_DistanceSpheroid

ST_DistanceSpheroid — Returnerar det minsta avståndet mellan två lon/lat-geometrier med hjälp av en sfäroid jordmodell.

Synopsis

float **ST_DistanceSpheroid**(geometry geomlonlatA, geometry geomlonlatB, spheroid measurement_spheroc

Beskrivning

Returnerar minsta avstånd i meter mellan två lon/lat-geometrier givet en viss sfäroid. Se förklaringen av sfäroider som ges för ST_LengthSpheroid.

Notera
Denna funktion tittar inte på geometrins SRID. Den förutsätter att geometrins koordinater är baserade på den angivna sfäroiden.

Tillgänglighet: 1.5 - stöd för andra geometrityper än punkter infördes. Tidigare versioner fungerar bara med punkter.

Ändrad: 2.2.0 I tidigare versioner kallades detta ST_Distance_Spheroid

Exempel

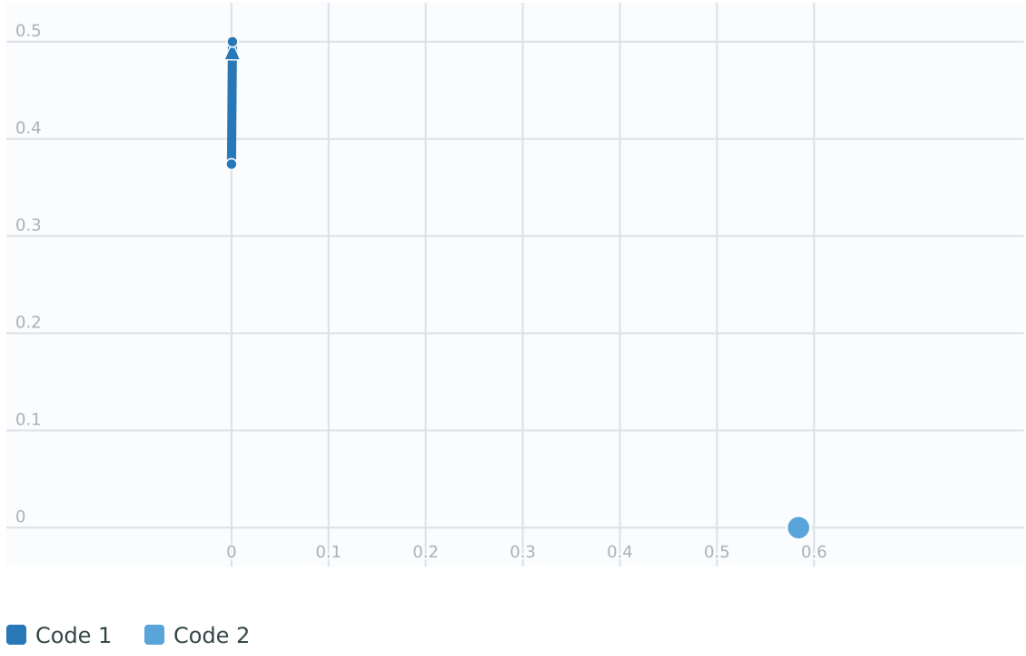
Code

```
WITH input AS (
  SELECT ST_Centroid(
           ST_GeomFromText(
             'LINESTRING(-118.584 38.374,-118.583 38.5)', 4326
           )
         ) AS source,
         ST_GeomFromText('POINT(-118 38)', 4326) AS target
)
SELECT round(
         ST_DistanceSpheroid(
           source,
           target,
           'SPHEROID["WGS 84",6378137,298.257223563]'
         )::numeric,
         2
       ) AS dist_meters_spheroid,
       round(ST_DistanceSphere(source, target)::numeric, 2)
         AS dist_meters_sphere,
       round(
         ST_Distance(
           ST_Transform(source, 32611),
           ST_Transform(target, 32611)
         )::numeric,
         2
       ) AS dist_utm11_meters
FROM input;
```

Output

```
dist_meters_spheroid | dist_meters_sphere | dist_utm11_meters
---------------------+--------------------+------------------
            70454.92 |           70424.71 |          70438.00
```

Figure



Se även

ST_Distance, ST_DistanceSphere

7.12.10 ST_FrechetDistance

ST_FrechetDistance — Returnerar Fréchet-avståndet mellan två geometrier.

Synopsis

float **ST_FrechetDistance**(geometry g1, geometry g2, float densifyFrac = -1);

Beskrivning

Implementerar algoritm för att beräkna Fréchet-avståndet begränsat till diskreta punkter för båda geometrierna, baserat på Computing Discrete Fréchet Distance. Fréchet-avståndet är ett mått på likhet mellan kurvor som tar hänsyn till placeringen och ordningsföljden av punkterna längs kurvorna. Därför är det ofta bättre än Hausdorff-avståndet.

När det valfria alternativet densifyFrac anges utför denna funktion en segmentförtätning innan det diskreta Fréchet-avståndet beräknas. Parametern densifyFrac anger den fraktion som varje segment ska förtätas med. Varje segment kommer att delas upp i ett antal lika långa undersegment, vars fraktion av den totala längden är närmast den angivna fraktionen.

Enheterna är i enheterna i geometriernas spatiala referenssystem.

Notera
Den nuvarande implementationen stöder endast toppar som diskreta platser. Detta kan utökas så att en godtycklig täthet av punkter kan användas.

Notera
Ju mindre densifyFrac vi anger, desto mer exakt Fréchet-avstånd får vi. Men beräkningstiden och minnesanvändningen ökar med kvadraten på antalet delsegment.

Utförs av GEOS-modulen.

Tillgänglighet: 2.4.0 - kräver GEOS >= 3.7.0

Exempel

Both lines are traversed from left to right. The first line is straight, while the second bows upward through its middle vertex. The point ordering matters, so keeping both inputs in the same direction makes the comparison clearer.

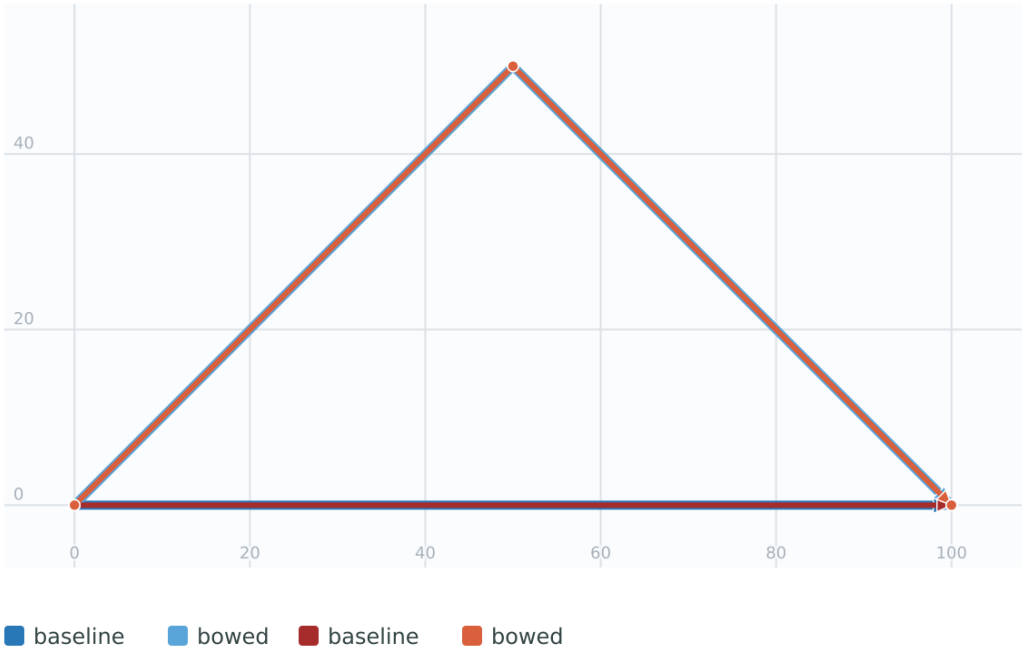
Code

```
WITH lines AS (
  SELECT 'LINESTRING (0 0,100 0)'::geometry AS baseline,
         'LINESTRING (0 0,50 50,100 0)'::geometry AS bowed
)
SELECT baseline AS baseline,
       bowed AS bowed,
       ST_FrechetDistance(baseline, bowed) AS discrete_distance,
       ST_FrechetDistance(baseline, bowed, 0.5) AS densified_distance
FROM lines;
```

Output

```
-[ RECORD 1 ]-----+---------------------------
baseline           | LINESTRING(0 0,100 0)
bowed              | LINESTRING(0 0,50 50,100 0)
discrete_distance  | 70.7106781186548
densified_distance | 50
```

Figure



Se även

ST_HausdorffDistance

7.12.11 ST_HausdorffDistance

ST_HausdorffDistance — Returnerar Hausdorff-avståndet mellan två geometrier.

Synopsis

float **ST_HausdorffDistance**(geometry g1, geometry g2);
float **ST_HausdorffDistance**(geometry g1, geometry g2, float densifyFrac);

Beskrivning

Returnerar Hausdorff-avståndet mellan två geometrier. Hausdorff-avståndet är ett mått på hur lika eller olika två geometrier är.

Funktionen beräknar faktiskt det diskreta Hausdorff-avståndet". Detta är Hausdorff-avståndet som beräknas vid diskreta punkter på geometrierna. Parametern *densifyFrac* kan specificeras för att ge ett mer exakt svar genom att segmenten förtätas innan det diskreta Hausdorff-avståndet beräknas. Varje segment delas upp i ett antal lika långa undersegment vars fraktion av segmentlängden ligger närmast den givna fraktionen.

Enheterna är i enheterna i geometriernas spatiala referenssystem.

Notera
Den här algoritmen är INTE likvärdig med standardavståndet Hausdorff. Den beräknar dock en approximation som är korrekt för en stor delmängd av användbara fall. Ett viktigt fall är linjesträngar som är ungefär parallella med varandra och ungefär lika långa. Detta är ett användbart mått för linjematchning.

Tillgänglighet: 1.5.0

Exempel

Hausdorff distance and minimum distance between two lines. The Hausdorff witness segment from the original illustration is retained and its length is checked against the computed distance.

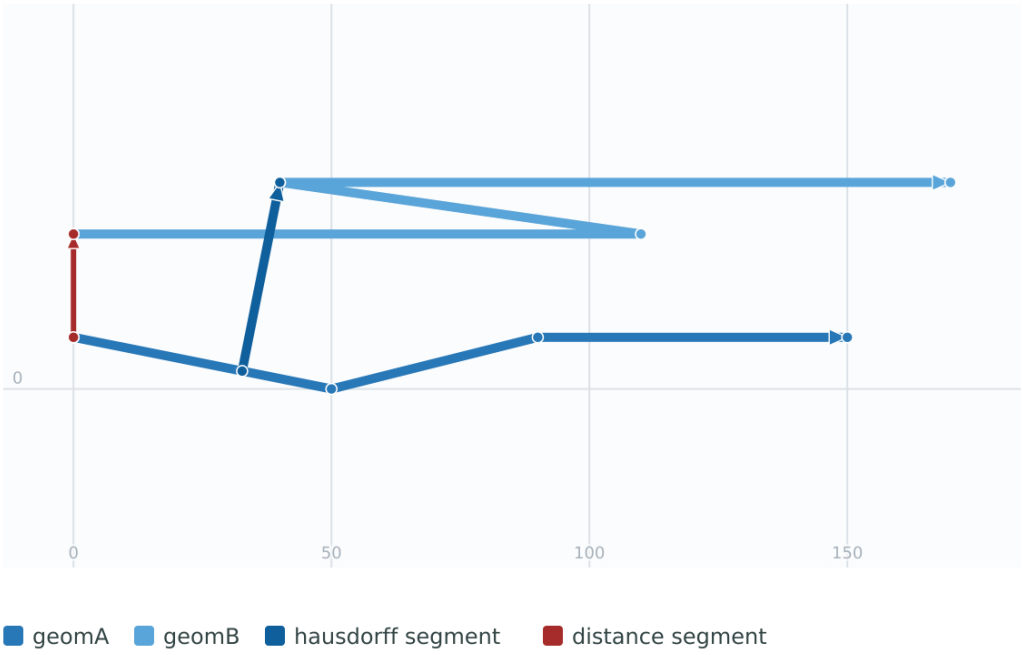
Code

```
WITH lines AS (
  SELECT 'LINESTRING (20 70,70 60,110 70,170 70)'::geometry AS geomA,
         'LINESTRING (20 90,130 90,60 100,190 100)'::geometry AS geomB,
         'LINESTRING (52.69230769230769 63.46153846153846,60 100)'::geometry AS  ↩
    hausdorff_segment
)
SELECT round(ST_HausdorffDistance(geomA, geomB)::numeric, 6) AS hausdorff_distance,
       round(ST_Distance(geomA, geomB)::numeric, 6) AS distance,
       round(ST_Length(hausdorff_segment)::numeric, 6) AS witness_length,
       ST_ShortestLine(geomA, geomB) AS distance_segment
FROM lines;
```

Output

```
37.262066 | 20.000000 | 37.262066 | LINESTRING(20 70,20 90)
```

Figure

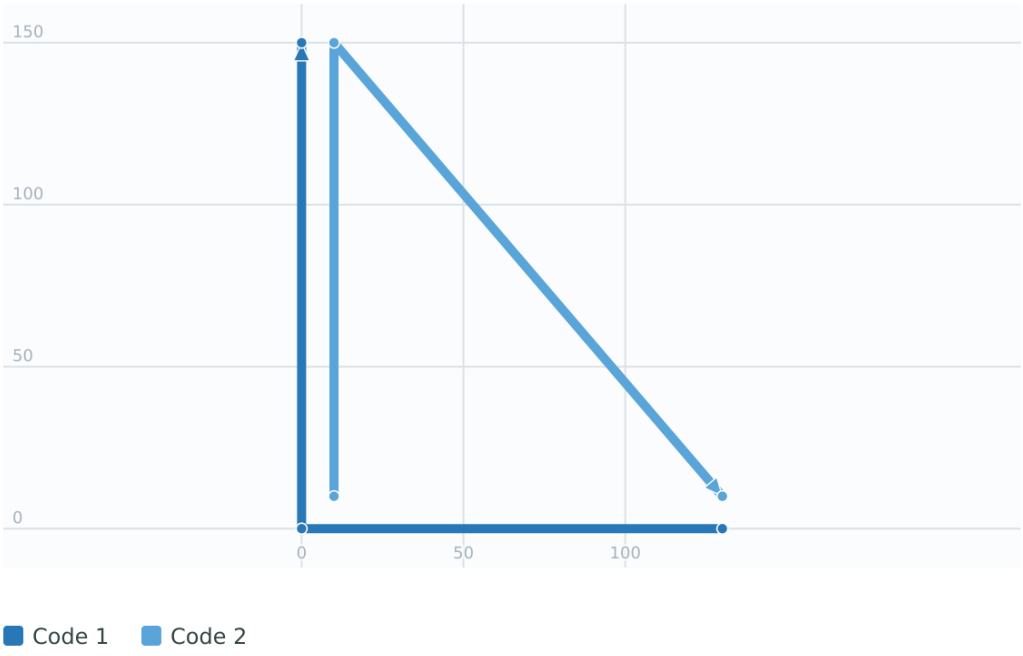


Exempel: Hausdorff-avstånd med förtätning.

Code

```
SELECT ST_HausdorffDistance(
            'LINESTRING (130 0,0 0,0 150)'::geometry,
            'LINESTRING (10 10,10 150,130 10)'::geometry,
            0.5);
```

Figure



Exempel: För varje byggnad, hitta den tomt som bäst representerar den. Först krävs att tomten korsar

byggnadens geometri. `DISTINCT ON` garanterar att varje byggnad bara listas en gång. `ORDER BY .. ST_HausdorffDistance` väljer det skifte som är mest likt byggnaden.

Code

```
SELECT DISTINCT ON (buildings.gid) buildings.gid, parcels.parcel_id
   FROM buildings
       INNER JOIN parcels
       ON ST_Intersects(buildings.geom, parcels.geom)
   ORDER BY buildings.gid, ST_HausdorffDistance(buildings.geom, parcels.geom);
```

Se även

ST_FrechetDistance

7.12.12 ST_Length

ST_Length — Returnerar 2D-längden för en linjär geometri.

Synopsis

float **ST_Length**(geometry a_2dlinestring);
float **ST_Length**(geography geog, boolean use_spheroid = true);

Beskrivning

För geometrityper: returnerar geometrins kartesiska 2D-längd om det är en LineString, MultiLineString, ST_Curve, ST_MultiCurve. För arealgeometrier returneras 0; använd ST_Perimeter i stället. Längdenheterna bestäms av geometrins spatiala referenssystem.

For geography types: computation is performed using the inverse geodesic calculation. Units of length are in meters. The spheroid is specified by the SRID; when no SRID is provided, WGS84 is used. If `use_spheroid = false`, then the calculation is based on a sphere instead of a spheroid.

För geometri är detta för närvarande ett alias för ST_Length2D, men detta kan ändras för att stödja högre dimensioner.

Ändrad: 2.0.0 Genomgripande ändring - i tidigare versioner gav en tillämpning av detta på en MULTI/-POLYGON av typen geografi omkretsen av POLYGONEN/MULTIPOLYGONEN. I 2.0.0 ändrades detta till att returnera 0 för att vara i linje med geometrins beteende. Använd ST_Perimeter om du vill ha omkretsen av en polygon

Notera
För geografi använder beräkningen som standard en sfäroid modell. Om du vill använda den snabbare men mindre exakta sfäriska beräkningen använder du ST_Length(gg,false);

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.5.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 7.1.2, 9.3.4

Tillgänglighet: 1.5.0 Stöd för geografi infördes i 1.5.

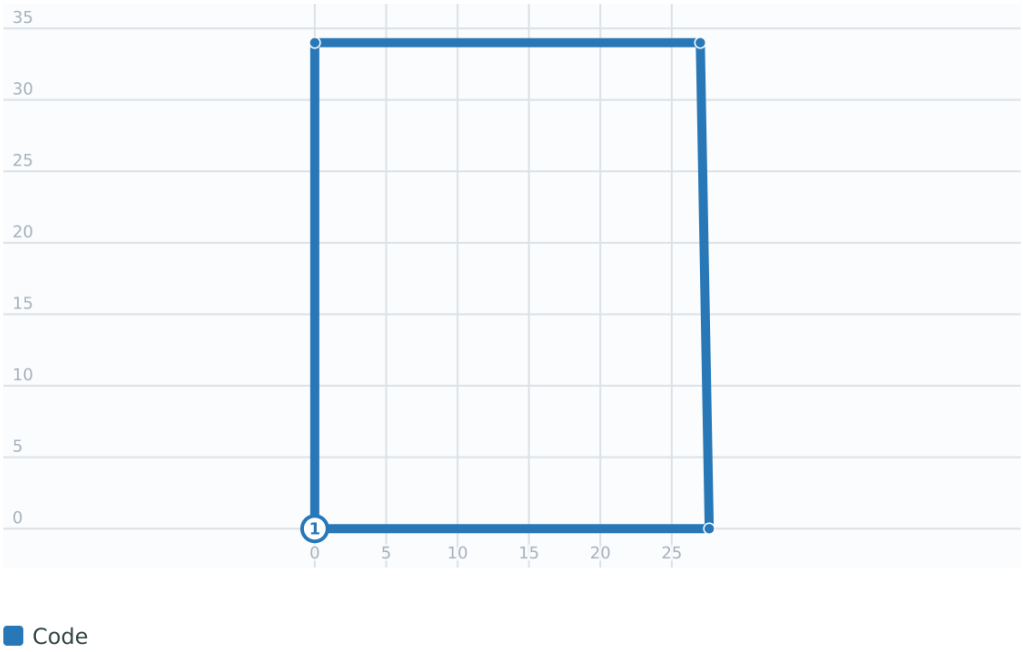
Exempel på geometri

Returnerar längd i fot för radsträng. Observera att detta är i fot eftersom EPSG:2249 är Massachusetts State Plane Feet

Code

```
SELECT ST_Length(ST_GeomFromText(
'LINESTRING(743238 2967416,743238 2967450,743265 2967450,
743265.625 2967416,743238 2967416)',
2249));
```

Figure

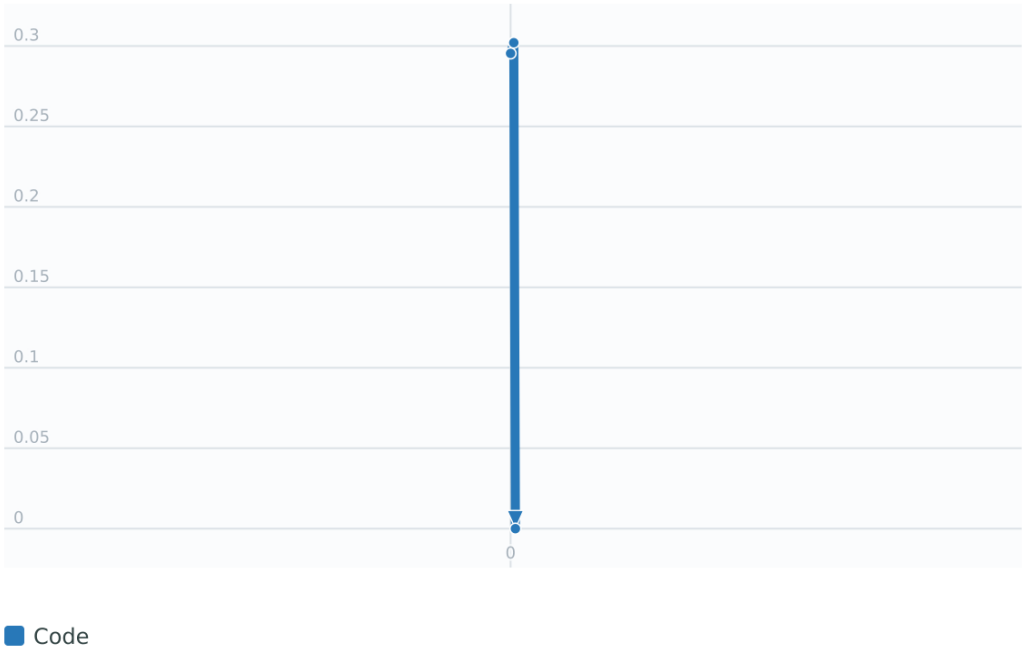


This example transforms a WGS 84 LineString to Massachusetts State Plane meters.

Code

```
SELECT ST_Length(ST_Transform(
        'SRID=4326;LINESTRING(-72.1260 42.45,-72.1240 42.45666,-72.123 42.1546)'::geometry,
            26986
    )
);
```

Figure



Geografiska exempel

Returnera längden på WGS 84-geografilinjen

This example uses the default spheroid calculation.

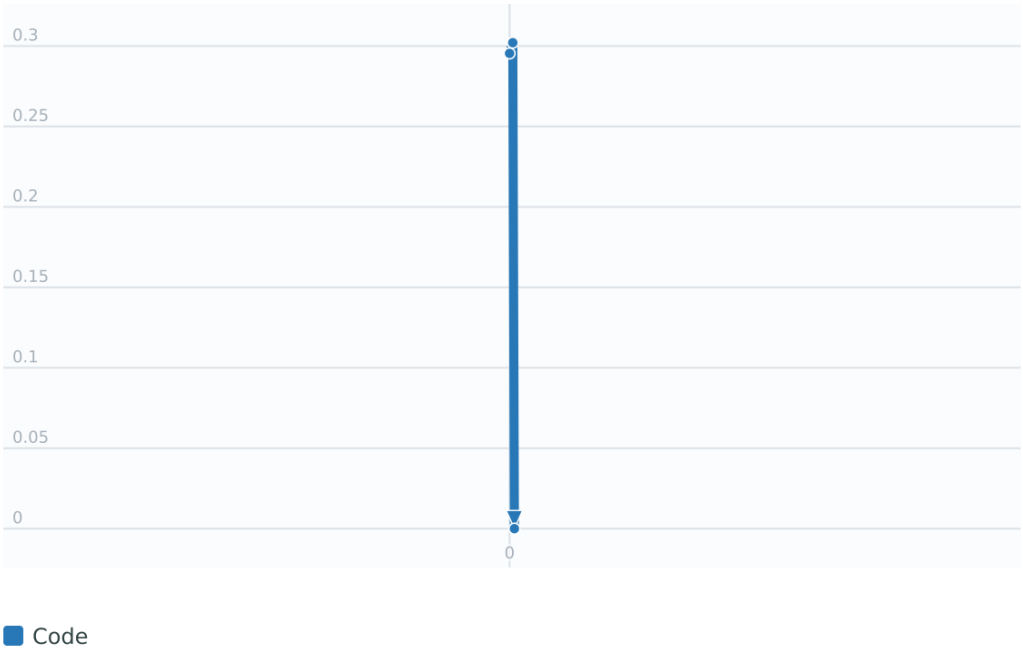
Code

```
SELECT ST_Length(the_geog) As length_spheroid, ST_Length(the_geog, false) As length_sphere
FROM (SELECT ST_GeographyFromText('SRID=4326;LINESTRING(-72.1260 42.45,-72.1240 ↩
    42.45666,-72.123 42.1546)') As the_geog)
 As foo;
```

Output

```
length_spheroid  |  length_sphere
-----------------+-----------------
34310.5703627284 | 34346.20609607416
```

Figure



Se även

ST_GeographyFromText, ST_GeomFromEWKT, ST_LengthSpheroid, ST_Perimeter, ST_Transform

7.12.13 ST_Length2D

ST_Length2D — Returnerar 2D-längden för en linjär geometri. Alias för `ST_Length`

Synopsis

float **ST_Length2D**(geometry a_2dlinestring);

Beskrivning

Returnerar geometrins 2D-längd om det är en linestrings eller multilinestrings. Detta är ett alias för `ST_Length`

Se även

ST_Length, ST_3DLength

7.12.14 ST_3DLength

ST_3DLength — Returnerar 3D-längden för en linjär geometri.

Synopsis

float **ST_3DLength**(geometry a_3dlinestring);

Beskrivning

Returnerar den 3-dimensionella eller 2-dimensionella längden på geometrin om det är en LineString eller MultiLineString. För 2-d linjer returneras bara 2-d längden (samma som ST_Length och ST_Length2D)

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 7.1, 10.3

Ändrad: 2.0.0 I tidigare versioner brukade detta kallas ST_Length3D

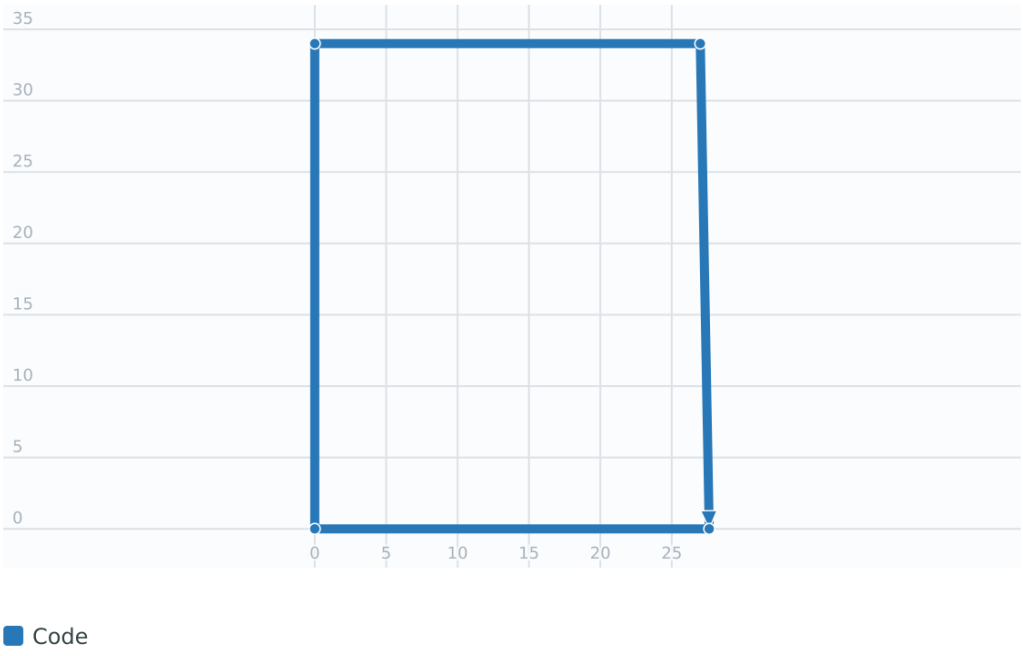
Exempel

Returlängd i fot för en 3D-kabel. Observera att detta är i fot eftersom EPSG:2249 är Massachusetts State Plane Feet

Code

```
SELECT ST_3DLength(ST_GeomFromText(
'LINESTRING(743238 2967416 1,743238 2967450 1,743265 2967450 3,
743265.625 2967416 3,743238 2967416 3)',
2249));
```

Figure



Se även

ST_Length, ST_Length2D

7.12.15 ST_LengthSpheroid

ST_LengthSpheroid — Returnerar 2D- eller 3D-längd/perimeter för en lon/lat-geometri på en sfäroid.

Synopsis

float **ST_LengthSpheroid**(geometry a_geometry, spheroid a_spheroid);

Beskrivning

Calculates the length or perimeter of a geometry on a spheroid. This is useful if the coordinates of the geometry are in longitude/latitude and a length is desired without reprojection. The spheroid is specified by a text value as follows:

```
SPHEROID[<NAME>,<SEMI-MAJOR AXIS>,<INVERSE FLATTENING>]
```

Till exempel:

```
SPHEROID["GRS_1980",6378137,298.257222101]
```

Tillgänglighet: 1.2.2

Ändrad: 2.2.0 I tidigare versioner kallades detta ST_Length_Spheroid och hade aliaset ST_3DLength_Sphero

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Measure a table geometry directly on a spheroid when the coordinates are stored in longitude and latitude.

Code

```
SELECT ST_LengthSpheroid(
    geometry_column,
    'SPHEROID["GRS_1980",6378137,298.257222101]'
    )
FROM geometry_table;
```

This example measures a 2D MultiLineString and each component separately on the same spheroid.

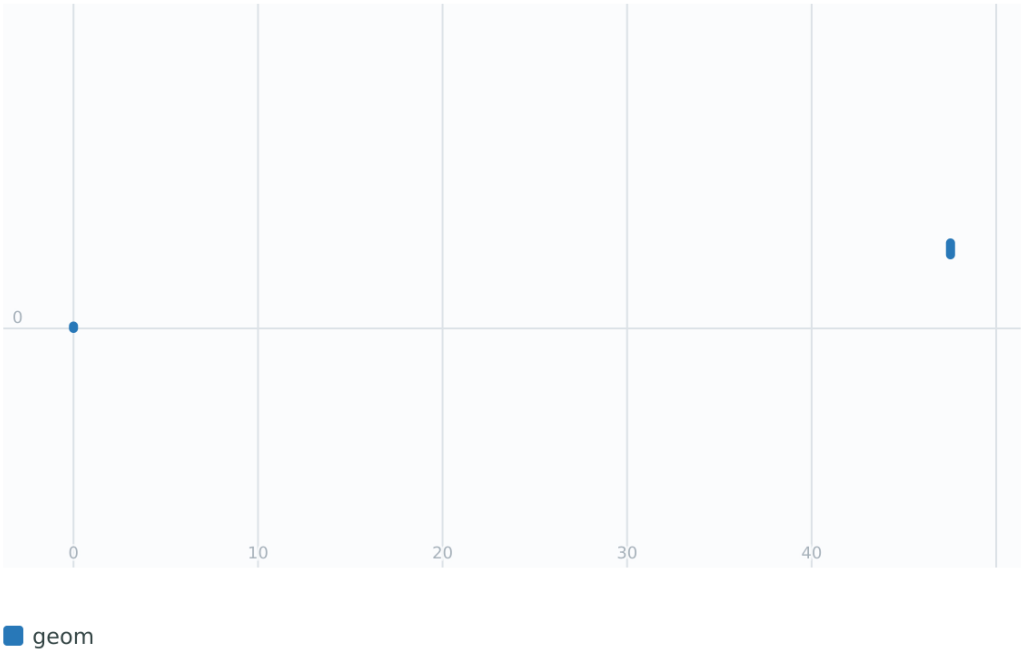
Code

```
WITH sample AS (
  SELECT 'MULTILINESTRING((-118.584 38.374,-118.583 38.5),
          (-71.05957 42.3589,-71.061 43))'::geometry AS geom,
         CAST('SPHEROID["GRS_1980",6378137,298.257222101]' AS spheroid) AS sph_m
)
SELECT ST_LengthSpheroid(geom, sph_m) AS tot_len,
       ST_LengthSpheroid(ST_GeometryN(geom, 1), sph_m) AS len_line1,
       ST_LengthSpheroid(ST_GeometryN(geom, 2), sph_m) AS len_line2
FROM sample;
```

Output

```
tot_len      |    len_line1     |    len_line2
-----------------+-----------------+-----------------
 85204.52077117894 | 13986.87252824332 | 71217.64824293563
```

Figure



This example uses 3D input; Z contributes to the measured result.

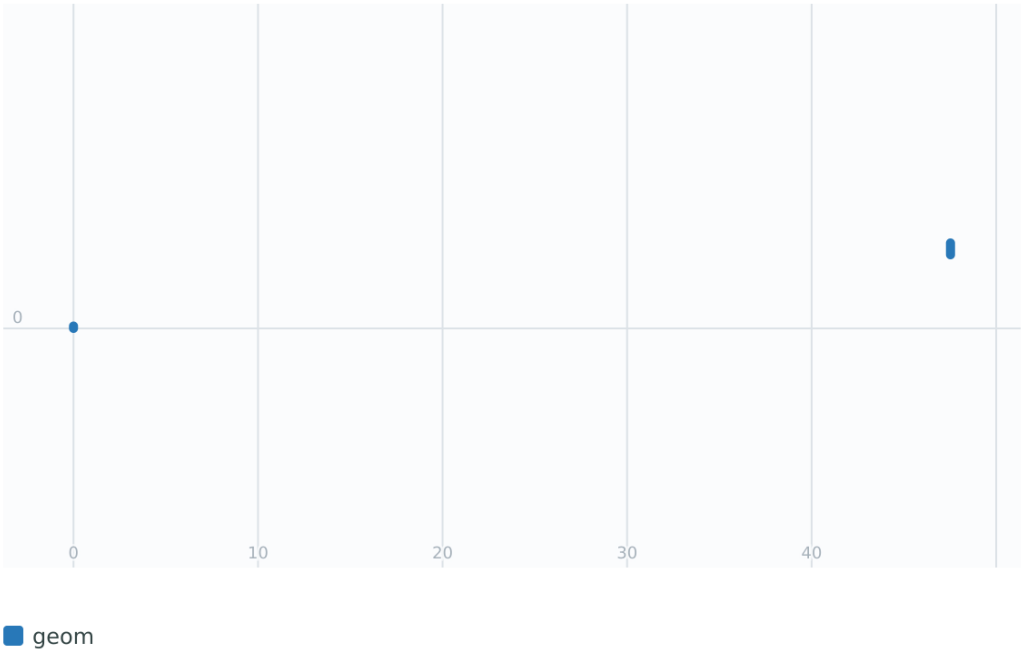
Code

```
WITH sample AS (
  SELECT 'MULTILINESTRING((-118.584 38.374 20,-118.583 38.5 30),
          (-71.05957 42.3589 75,-71.061 43 90))'::geometry AS geom,
         CAST('SPHEROID["GRS_1980",6378137,298.257222101]' AS spheroid) AS sph_m
)
SELECT ST_LengthSpheroid(geom, sph_m) AS tot_len,
       ST_LengthSpheroid(ST_GeometryN(geom, 1), sph_m) AS len_line1,
       ST_LengthSpheroid(ST_GeometryN(geom, 2), sph_m) AS len_line2
FROM sample;
```

Output

```
tot_len      |    len_line1    |    len_line2
-----------------+-----------------+-----------------
 85204.52592562366 | 13986.876103023424 | 71217.64982260024
```

Figure



Se även

ST_GeometryN, ST_Length

7.12.16 ST_LongestLine

ST_LongestLine — Returnerar den längsta 2D-linjen mellan två geometrier.

Synopsis

geometry **ST_LongestLine**(geometry g);
geometry **ST_LongestLine**(geometry g1, geometry g2);

Beskrivning

Returnerar den 2-dimensionella längsta linjen mellan punkterna i två geometrier. Linjen som returneras börjar på `g1` och slutar på `g2`.

Den längsta linjen finns alltid mellan två hörn. Funktionen returnerar den första längsta linjen om mer än en hittas. Linjens längd är lika med det avstånd som returneras av ST_MaxDistance.

If only one geometry is provided, or g1 and g2 are the same geometry, returns the line between the two vertices farthest apart in the geometry. The endpoints of the line lie on the circle computed by ST_MinimumBoundingCircle.

Tillgänglighet: 1.5.0

Enhanced: 3.7.0 - support for a single geometry input.

Exempel

Longest line between a Point and a LineString.

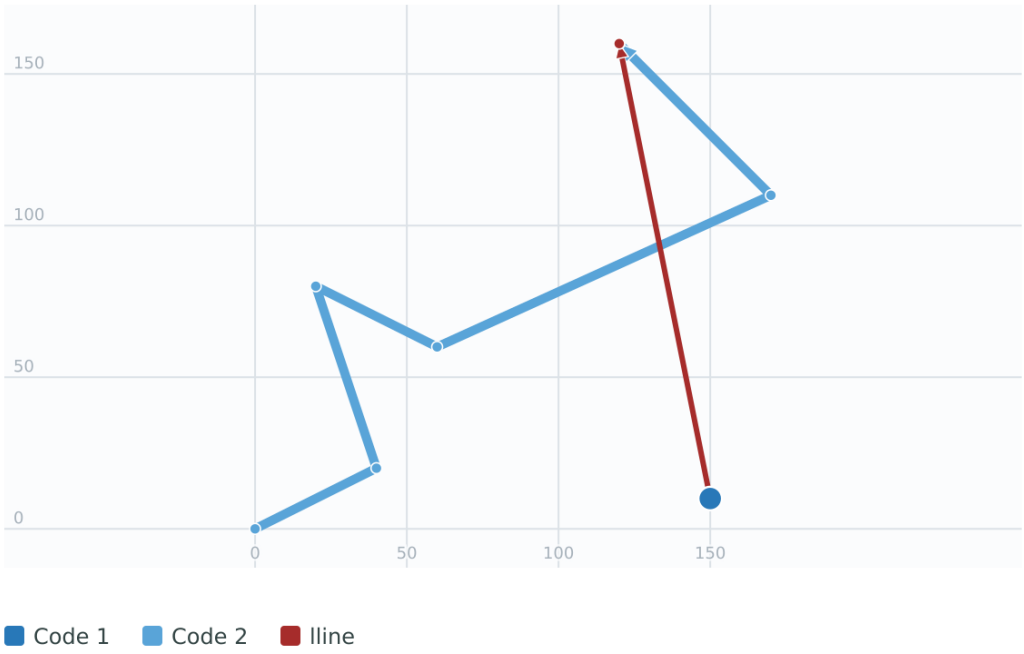
Code

```
SELECT ST_LongestLine(
        'POINT (160 40)',
        'LINESTRING (10 30,50 50,30 110,70 90,180 140,130 190)' ) AS lline;
```

Output

```
LINESTRING(160 40,130 190)
```

Figure



Longest line between two polygons.

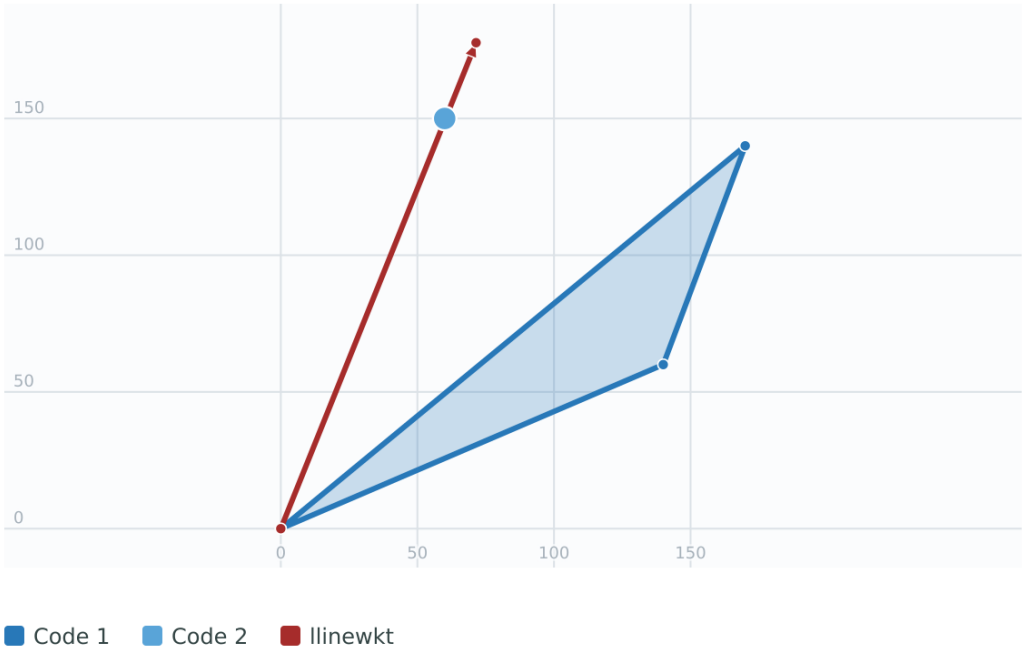
Code

```
SELECT ST_LongestLine(
        'POLYGON ((190 150,20 10,160 70,190 150))',
        ST_Buffer('POINT(80 160)', 30)
            ) AS llinewkt;
```

Output

```
LINESTRING(20 10,91.4805029709527 187.7163859753386)
```

Figure



Längsta linjen över en enda geometri. Linjens längd är lika med det maximala avståndet. Linjens ändpunkter ligger på den minsta begränsningscirkeln.

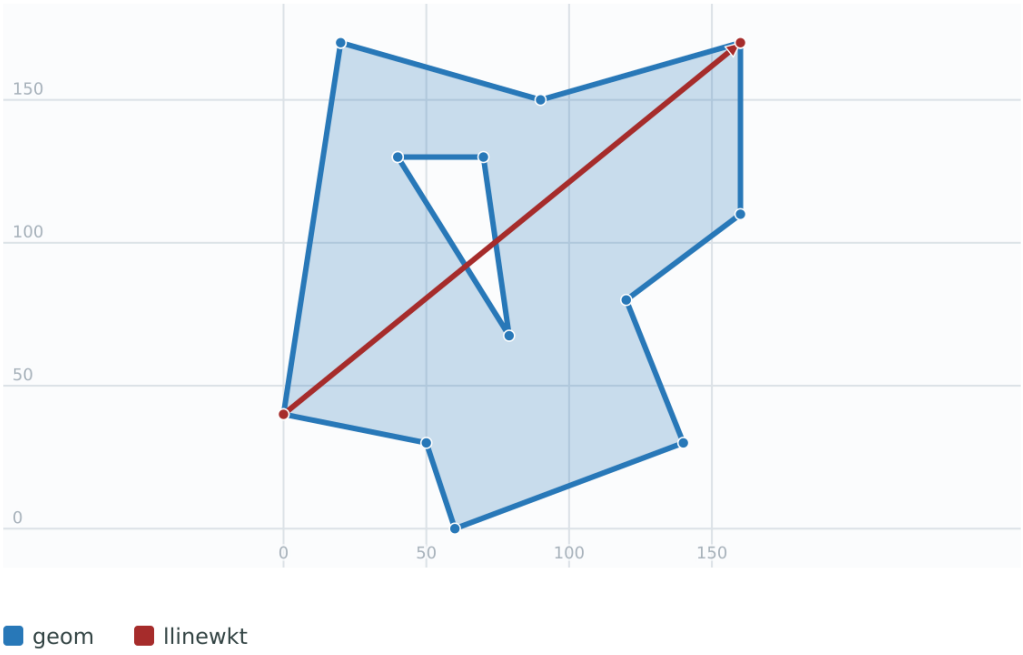
Code

```
SELECT ST_LongestLine(geom) AS llinewkt,
                  ST_MaxDistance(geom) AS max_dist,
                  ST_Length(ST_LongestLine(geom)) AS lenll
FROM (SELECT 'POLYGON ((40 180,110 160,180 180,180 120,140 90,160 40,80 10,70 40,20 50,40 ↩
    180),
              (60 140,99 77.5,90 140,60 140))'::geometry AS geom) AS t;
```

Output

```
llinewkt          |      max_dist      |       lenll
--------------------------+--------------------+--------------------
 LINESTRING(20 50,180 180) | 206.15528128088303 | 206.15528128088303
```

Figure



Se även

ST_MaxDistance, ST_ShortestLine, ST_3DLongestLine, ST_MinimumBoundingCircle

7.12.17 ST_3DLongestLine

ST_3DLongestLine — Returnerar den längsta 3D-linjen mellan två geometrier

Synopsis

geometry **ST_3DLongestLine**(geometry g1, geometry g2);

Beskrivning

Returnerar den 3-dimensionella längsta linjen mellan två geometrier. Funktionen returnerar den första längsta linjen om det finns fler än en. Linjen som returneras börjar i g1 och slutar i g2. Linjens 3D-längd är lika med det avstånd som returneras av ST_3DMaxDistance.

Tillgänglighet: 2.0.0

Ändrad: 2.2.0 - om 2 2D-geometrier matas in returneras en 2D-punkt (istället för det gamla beteendet som antog 0 för saknad Z). Vid 2D och 3D antas Z inte längre vara 0 för saknad Z.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Exempel

Longest line between a LineString and a Point, comparing the 3D and 2D results.

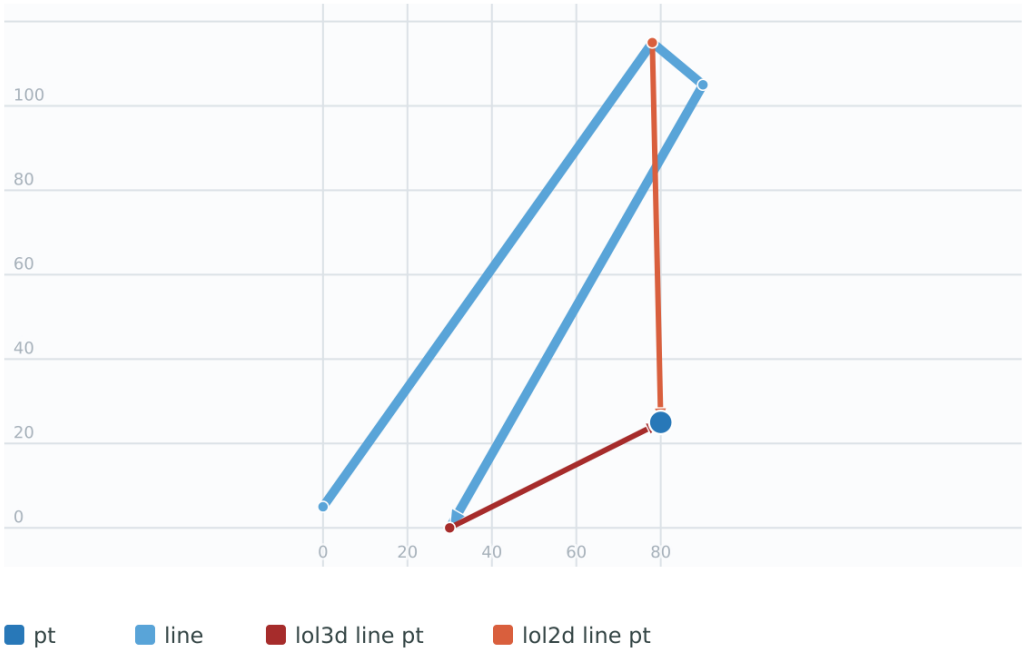
Code

```
SELECT ST_3DLongestLine(line, pt) AS lol3d_line_pt, ST_LongestLine(line, pt) AS  ↩
    lol2d_line_pt
FROM (
    SELECT 'POINT(100 100 30)'::geometry AS pt,
           'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 1000)'::geometry AS line
) AS foo;
```

Output

```
lol3d_line_pt           |       lol2d_line_pt
--------------------------------+---------------------------
 LINESTRING(50 75 1000,100 100 30) | LINESTRING(98 190,100 100)
```

Figure



Longest line between a LineString and a MultiPoint, comparing the 3D and 2D results.

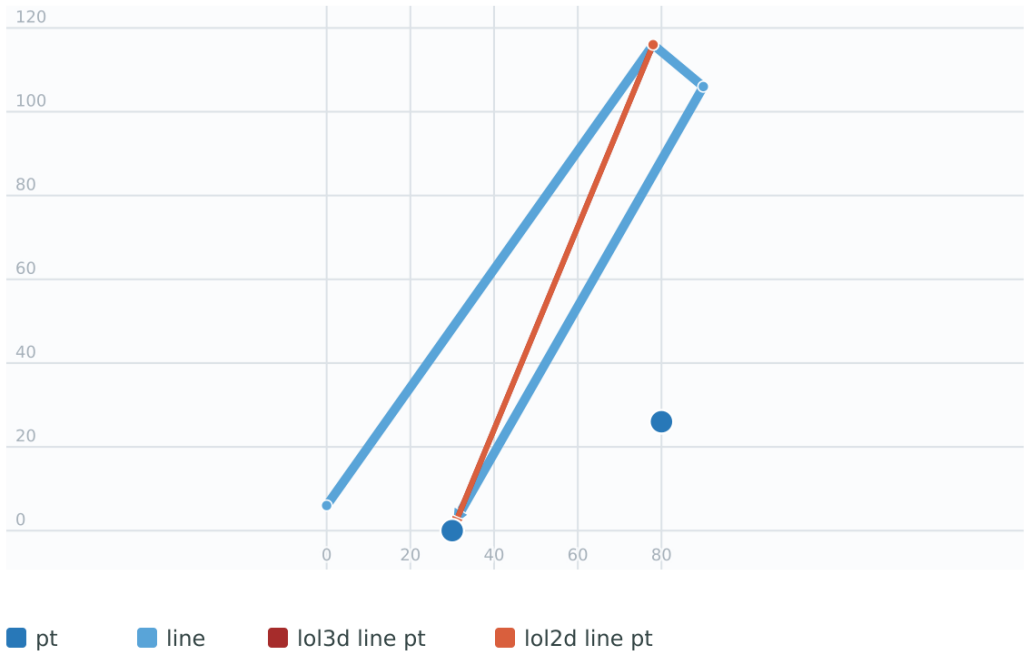
Code

```
SELECT ST_3DLongestLine(line, pt) AS lol3d_line_pt, ST_LongestLine(line, pt) AS  ↩
    lol2d_line_pt
FROM (
    SELECT 'MULTIPOINT(100 100 30,50 74 1000)'::geometry AS pt,
           'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 900)'::geometry AS line
) AS foo;
```

Output

```
lol3d_line_pt          |      lol2d_line_pt
--------------------------------+-------------------------
 LINESTRING(98 190 1,50 74 1000) | LINESTRING(98 190,50 74)
```

Figure



Longest line between a Polygon and a MultiLineString, comparing the 3D and 2D results.

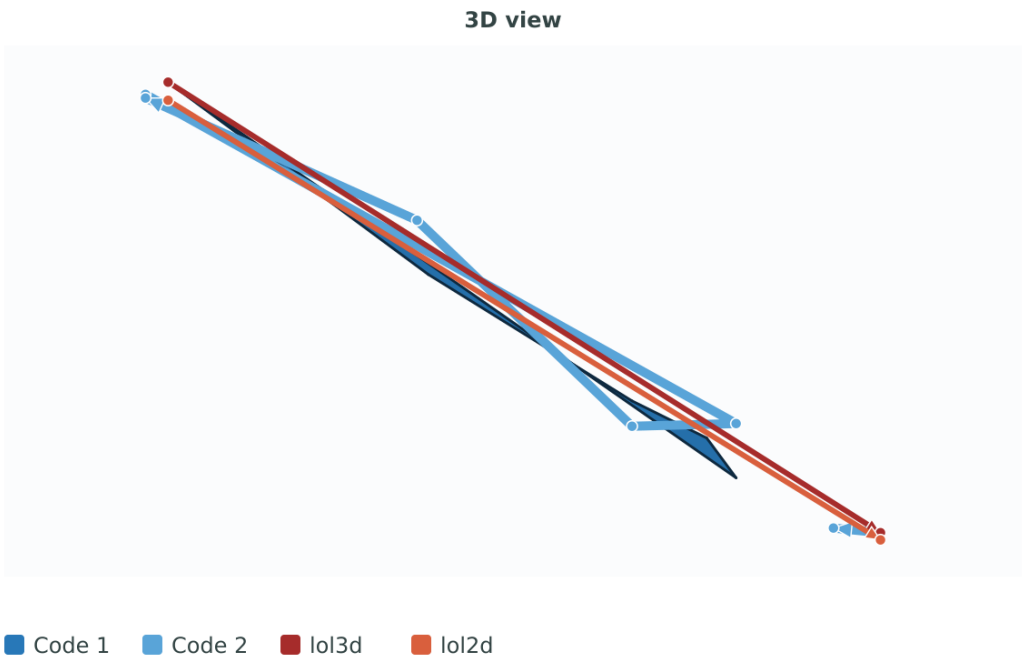
Code

```
SELECT ST_3DLongestLine(poly, mline) AS lol3d, ST_LongestLine(poly, mline) AS lol2d
FROM (
    SELECT ST_GeomFromEWKT(
                'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))'
            ) AS poly,
            ST_GeomFromEWKT(
                'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),
                (1 10 2,5 20 1))'
            ) AS mline
) AS foo;
```

Output

```
lol3d             |          lol2d
------------------------------+-------------------------
 LINESTRING(175 150 5,1 10 2) | LINESTRING(175 150,1 10)
```

Figure



Se även

ST_3DClosestPoint, ST_3DDistance, ST_LongestLine, ST_3DShortestLine, ST_3DMaxDistance

7.12.18 ST_MaxDistance

ST_MaxDistance — Returnerar det största 2D-avståndet mellan två geometrier i projicerade enheter.

Synopsis

float **ST_MaxDistance**(geometry g);
float **ST_MaxDistance**(geometry g1, geometry g2);

Beskrivning

Returnerar det 2-dimensionella maximala avståndet mellan två geometrier, i projicerade enheter. Det maximala avståndet inträffar alltid mellan två hörn. Detta är längden på den linje som returneras av ST_LongestLine.

If only one geometry is provided, or g1 and g2 are the same geometry, returns the distance between the two vertices farthest apart in that geometry.

Tillgänglighet: 1.5.0

Enhanced: 3.7.0 - support for a single geometry input.

Exempel

Maximalt avstånd mellan en punkt och linjer.

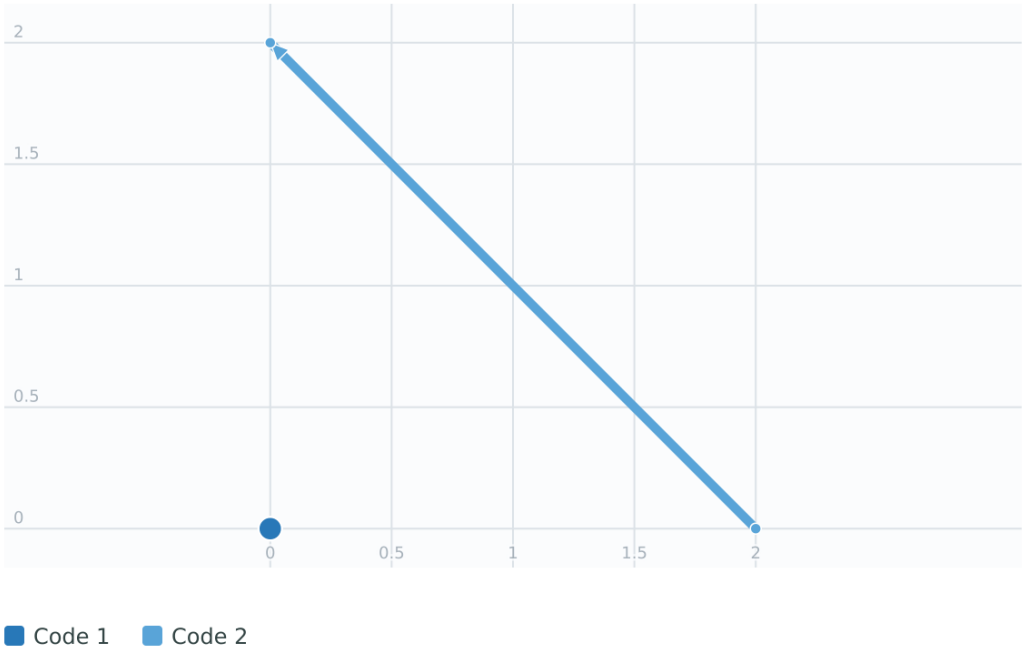
Code

```
SELECT ST_MaxDistance('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
2
```

Figure



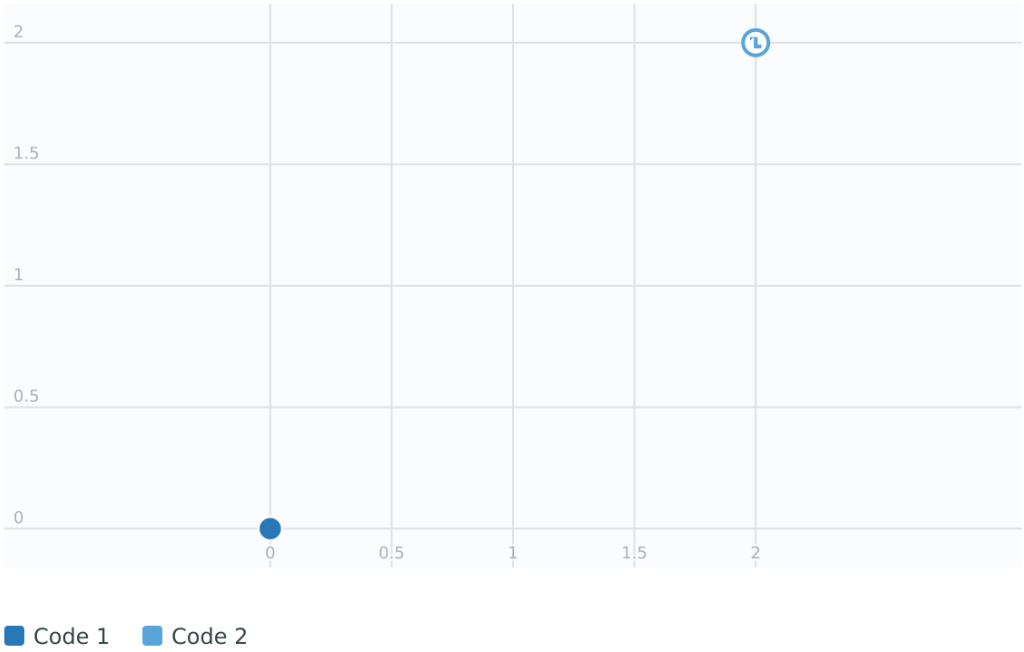
Code

```
SELECT ST_MaxDistance('POINT(0 0)'::geometry, 'LINESTRING ( 2 2,2 2 )'::geometry);
```

Output

```
2.82842712474619
```

Figure



Maximalt avstånd mellan hörnpunkter i en och samma geometri.

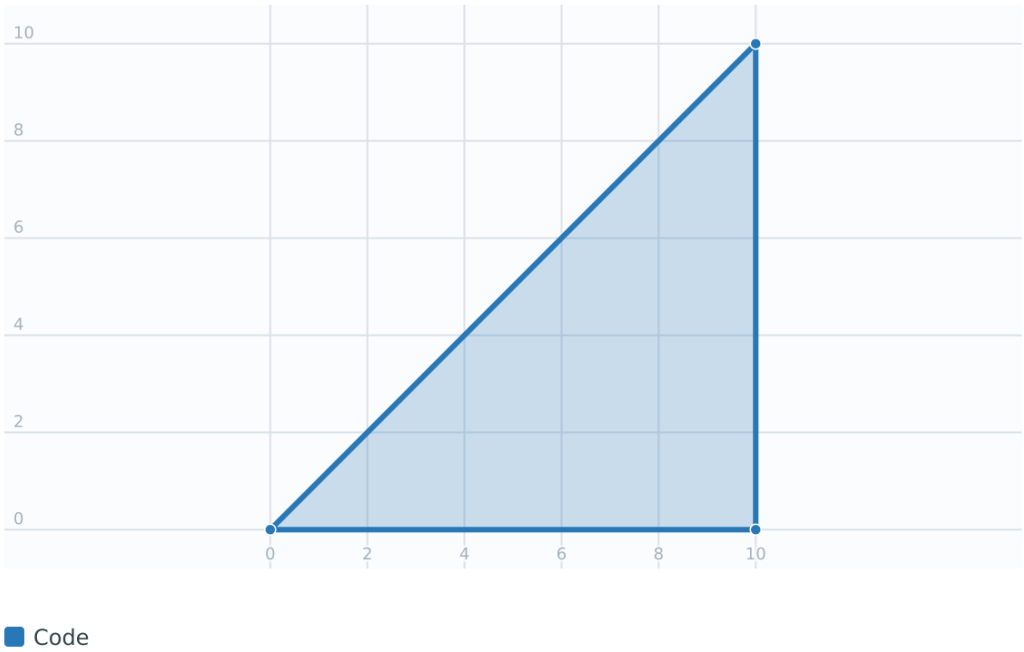
Code

```
SELECT ST_MaxDistance('POLYGON ((10 10,10 0,0 0,10 10))'::geometry);
```

Output

```
14.142135623730951
```

Figure



Se även

ST_Distance, ST_LongestLine, ST_DFullyWithin

7.12.19 ST_3DMaxDistance

ST_3DMaxDistance — Returnerar det maximala kartesiska 3D-avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.

Synopsis

float **ST_3DMaxDistance**(geometry g1, geometry g2);

Beskrivning

Returnerar det 3-dimensionella maximala cartesiska avståndet mellan två geometrier i projicerade enheter (spatiala ref-enheter).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Tillgänglighet: 2.0.0

Ändrad: 2.2.0 - När det gäller 2D och 3D antas Z inte längre vara 0 om Z saknas.

Exempel

Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point and line compared 2D point and line) Note that currently no vertical datum support so Z is not transformed and assumed to be same units as final.

Code

```
SELECT ST_3DMaxDistance(
            ST_Transform('SRID=4326;POINT(-72.1235 42.3521 10000)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)':: ↩
    geometry, 2163)
            ) As dist_3d,
            ST_MaxDistance(
                ST_Transform('SRID=4326;POINT(-72.1235 42.3521 10000)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)':: ↩
    geometry, 2163)
            ) As dist_2d;
```

Output

```
dist_3d      |     dist_2d
------------------+------------------
 24383.7467488441 | 22247.8472107251
```

Se även

ST_Distance, ST_3DDWithin, ST_3DMaxDistance, ST_Transform

7.12.20 ST_MinimumClearance

ST_MinimumClearance — Returnerar den minsta frigången för en geometri, ett mått på geometrins robusthet.

Synopsis

float **ST_MinimumClearance**(geometry g);

Beskrivning

Det är möjligt att en geometri uppfyller kriterierna för giltighet enligt ST_IsValid (polygoner) eller ST_IsSimple (linjer), men att den blir ogiltig om en av dess hörnpunkter flyttas ett litet avstånd. Detta kan ske på grund av förlust av precision vid konvertering till textformat (t.ex. WKT, KML, GML, GeoJSON) eller binära format som inte använder koordinater med dubbel precision och flyttal (t.ex. MapInfo TAB).

Det minsta avståndet är ett kvantitativt mått på en geometris robusthet mot förändringar i koordinatprecisionen. Det är det största avstånd med vilket geometrins hörn kan flyttas utan att skapa en ogiltig geometri. Större värden på minsta avstånd indikerar större robusthet.

Om en geometri har en minsta frihöjd på `e`, då:

- Inga två distinkta hörn i geometrin ligger närmare varandra än avståndet `e`.
- Ingen toppunkt är närmare än `e` ett linjesegment där den inte är en ändpunkt.

Om det inte finns något minsta spelrum för en geometri (t.ex. en enskild punkt eller en MultiPoint vars punkter är identiska), är returvärdet `Infinity`.

För att undvika validitetsproblem som orsakas av precisionsförlust kan ST_ReducePrecision minska koordinatprecisionen samtidigt som polygongeometrin förblir giltig.

Tillgänglighet: 2.3.0

Exempel

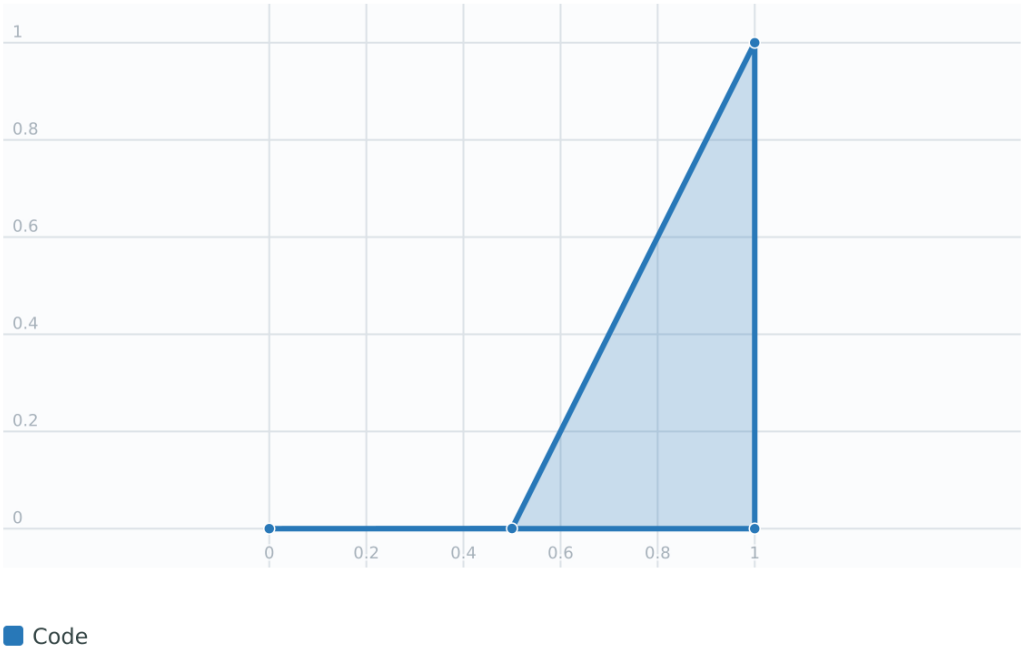
Code

```
SELECT ST_MinimumClearance('POLYGON ((0 0,1 0,1 1,0.5 3.2e-4,0 0))');
```

Output

```
0.00032
```

Figure



Se även

ST_MinimumClearanceLine, ST_IsSimple, ST_IsValid, ST_ReducePrecision

7.12.21 ST_MinimumClearanceLine

ST_MinimumClearanceLine — Returnerar LineString med två punkter som sträcker sig över en geometris minsta spelrum.

Synopsis

Geometry **ST_MinimumClearanceLine**(geometry g);

Beskrivning

Returnerar den tvåpunkts LineString som sträcker sig över en geometris minsta spelrum. Om geometrin inte har något minsta spelrum returneras `LINESTRING EMPTY`.

Utförs av GEOS-modulen.

Tillgänglighet: 2.3.0 - kräver GEOS >= 3.6.0

Exempel

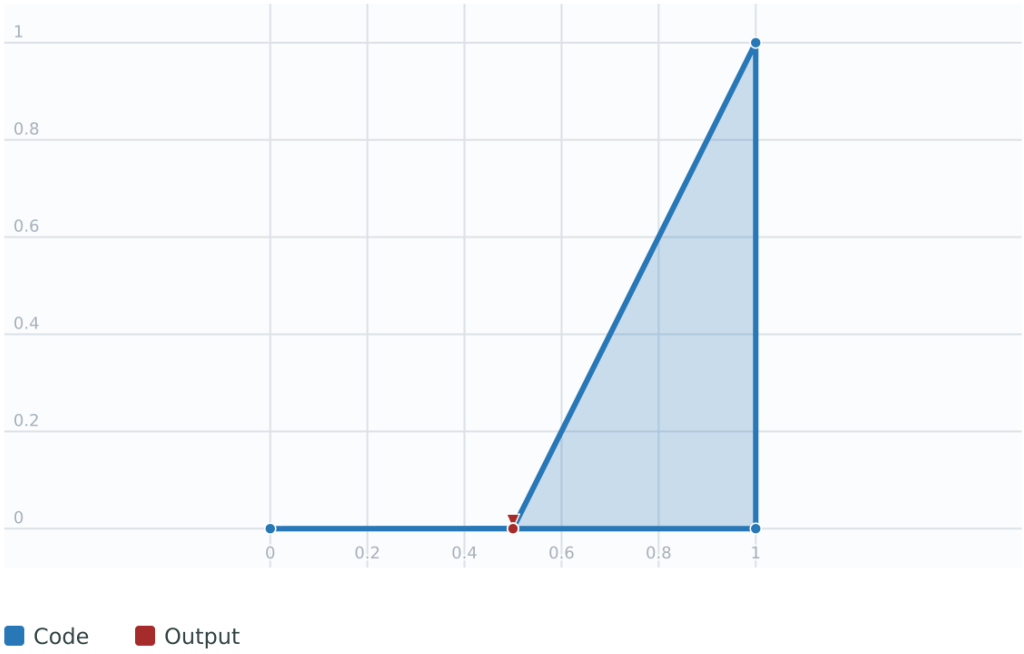
Code

```
SELECT ST_MinimumClearanceLine('POLYGON ((0 0,1 0,1 1,0.5 3.2e-4,0 0))');
```

Output

```
LINESTRING(0.5 0.00032,0.5 0)
```

Figure



Se även

ST_MinimumClearance

7.12.22 ST_Perimeter

ST_Perimeter — Returnerar längden på gränsen för en polygonal geometri eller geografi.

Synopsis

float **ST_Perimeter**(geometry g1);
float **ST_Perimeter**(geography geog, boolean use_spheroid = true);

Beskrivning

Returnerar 2D-perimetern för geometrin/geografin om den är en ST_Surface, ST_MultiSurface (Polygon, MultiPolygon). 0 returneras för icke-verkliga geometrier. För linjära geometrier används ST_Length. För geometrityper specificeras enheter för perimetermått av geometrins spatiala referenssystem.

For geography types, the calculations are performed using the inverse geodesic problem, where perimeter units are in meters. The spheroid is specified by the SRID; when no SRID is provided, WGS84 is used. If `use_spheroid = false`, then calculations will approximate a sphere instead of a spheroid.

För närvarande är detta ett alias för ST_Perimeter2D, men detta kan ändras för att stödja högre dimensioner.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.5.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.1.3, 9.5.4

Tillgänglighet 2.0.0: Stöd för geografi infördes

Exempel

Geometry.

Returnerar omkrets i fot för Polygon och MultiPolygon. Observera att detta är i fot eftersom EPSG:2249 är Massachusetts State Plane Feet

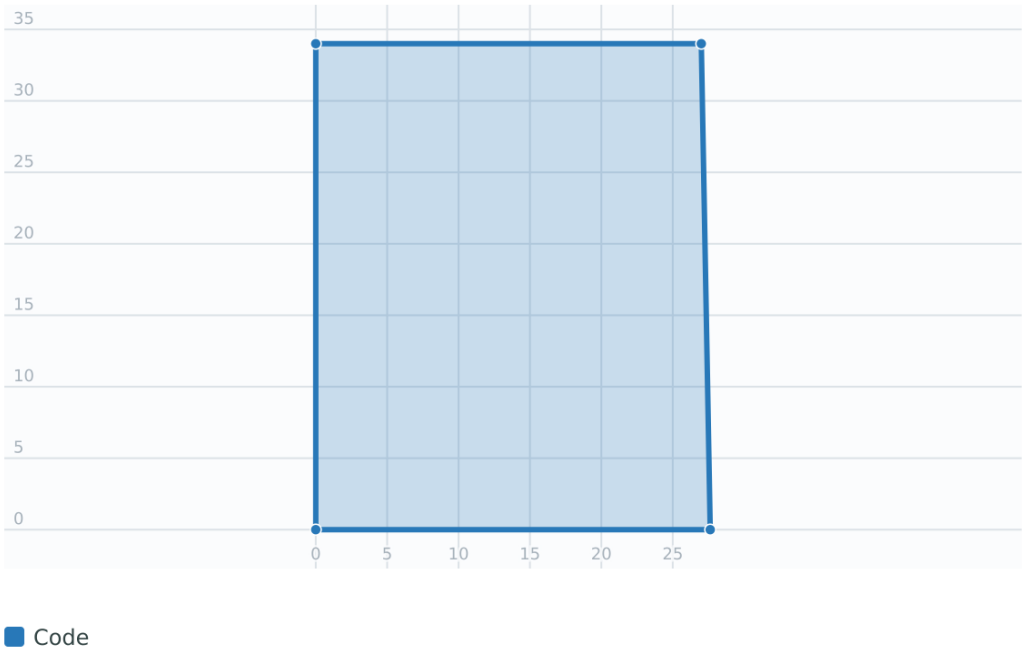
Code

```
SELECT ST_Perimeter(ST_GeomFromText(
'POLYGON((743238 2967416,743238 2967450,743265 2967450,
743265.625 2967416,743238 2967416))',
2249));
```

Output

```
122.63074400009504
```

Figure



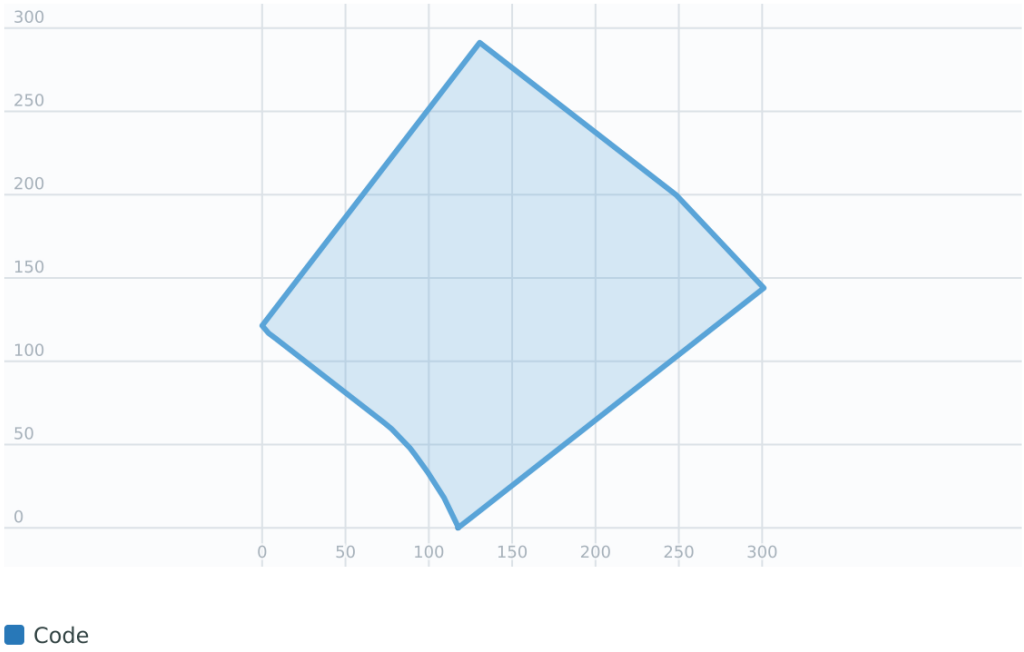
Code

```
SELECT ST_Perimeter(ST_GeomFromText(
'MULTIPOLYGON(((763104.471273676 2949418.44119003,
763104.477769673 2949418.42538203,
763104.189609677 2949418.22343004,763104.471273676 2949418.44119003)),
((763104.471273676 2949418.44119003,763095.804579742 2949436.33850239,
763086.132105649 2949451.46730207,763078.452329651 2949462.11549407,
763075.354136904 2949466.17407812,763064.362142565 2949477.64291974,
763059.953961626 2949481.28983009,762994.637609571 2949532.04103014,
762990.568508415 2949535.06640477,762986.710889563 2949539.61421415,
763117.237897679 2949709.50493431,763235.236617789 2949617.95619822,
763287.718121842 2949562.20592617,763111.553321674 2949423.91664605,
763104.471273676 2949418.44119003)))',
2249));
```

Output

```
845.227713366825
```

Figure



Returnerar omkrets i meter och fot för Polygon och MultiPolygon. Observera att detta är geografi (WGS 84 long lat)

Code

```
SELECT round(ST_Perimeter(geog)::numeric, 8) As per_meters,
       round((ST_Perimeter(geog)/0.3048)::numeric, 8) As per_ft
FROM ST_GeogFromText('POLYGON((-71.1776848522251 42.3902896512902,-71.1776843766326 ↩
    42.3903829478009,
-71.1775844305465 42.3903826677917,-71.1775825927231 42.3902893647987,-71.1776848522251 ↩
    42.3902896512902))') As geog;
```

Output

```
per_meters  |    per_ft
------------+-------------
 37.37904625 | 122.63466619
```

Figure

This example uses a MultiPolygon.

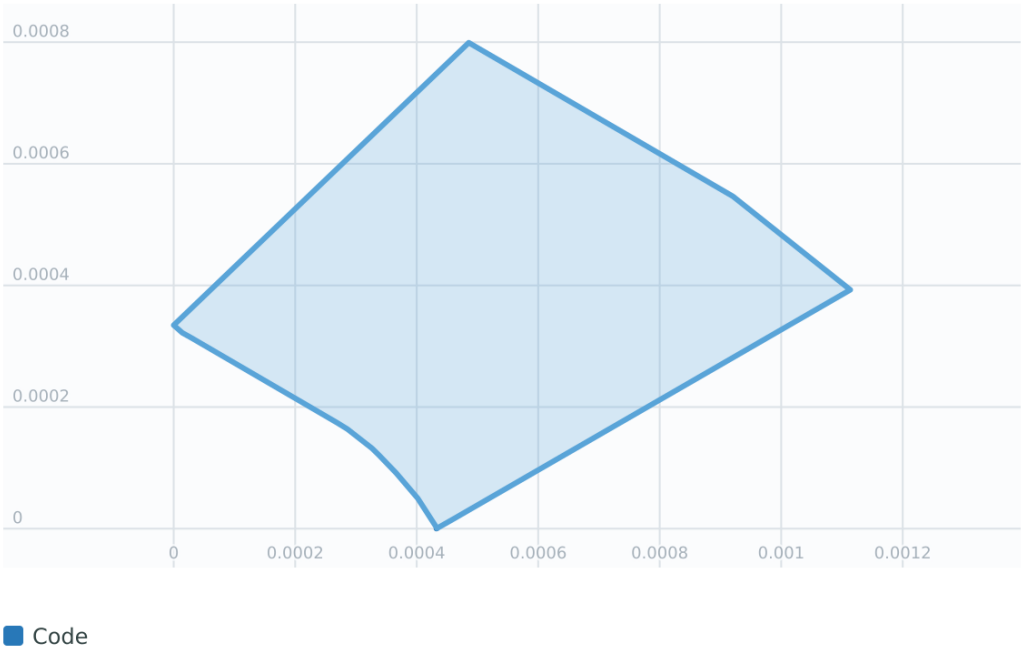
Code

```
SELECT round(ST_Perimeter(geog)::numeric, 8) As per_meters,
       round(ST_Perimeter(geog, false)::numeric, 8) As per_sphere_meters,
       round((ST_Perimeter(geog)/0.3048)::numeric, 8) As per_ft
FROM ST_GeogFromText('MULTIPOLYGON(((-71.1044543107478 42.340674480411,-71.1044542869917 ↩
    42.3406744369506,
-71.1044553562977 42.340673886454,-71.1044543107478 42.340674480411)),
((-71.1044543107478 42.340674480411,-71.1044860600303 42.3407237015564,-71.1045215770124 ↩
    42.3407653385914,
-71.1045498002983 42.3407946553165,-71.1045611902745 42.3408058316308,-71.1046016507427 ↩
    42.340837442371,
-71.104617893173 42.3408475056957,-71.1048586153981 42.3409875993595,-71.1048736143677 ↩
    42.3409959528211,
-71.1048878050242 42.3410084812078,-71.1044020965803 42.3414730072048,
-71.1039672113619 42.3412202916693,-71.1037740497748 42.3410666421308,
-71.1044280218456 42.3406894151355,-71.1044543107478 42.340674480411)))') As geog;
```

Output

```
per_meters  | per_sphere_meters |    per_ft
------------+-------------------+-------------
 257.63428368 |      257.41231145 | 845.25683623
```

Figure



Se även

ST_GeogFromText, ST_GeomFromText, ST_Length

7.12.23 ST_Perimeter2D

ST_Perimeter2D — Returnerar 2D-perimetern för en polygonal geometri. Alias för `ST_Perimeter`.

Synopsis

float **ST_Perimeter2D**(geometry geomA);

Beskrivning

Returnerar den 2-dimensionella omkretsen för en polygonal geometri.

Notera
Detta är för närvarande ett alias för ST_Perimeter. I framtida versioner kan ST_Perimeter returnera den högsta dimensionens omkrets för en geometri. Detta är fortfarande under övervägande

Se även

ST_Perimeter

7.12.24 ST_3DPerimeter

ST_3DPerimeter — Returnerar 3D-perimetern för en polygonal geometri.

Synopsis

float **ST_3DPerimeter**(geometry geomA);

Beskrivning

Returnerar geometrins 3-dimensionella omkrets, om den är en polygon eller multipolygon. Om geometrin är 2-dimensionell returneras den 2-dimensionella omkretsen.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM ISO/IEC 13249-3: 8.1, 10.5

Ändrad: 2.0.0 I tidigare versioner brukade detta kallas ST_Perimeter3D

Exempel

Omkretsen av en något upphöjd polygon i luften i Massachusetts state plan feet

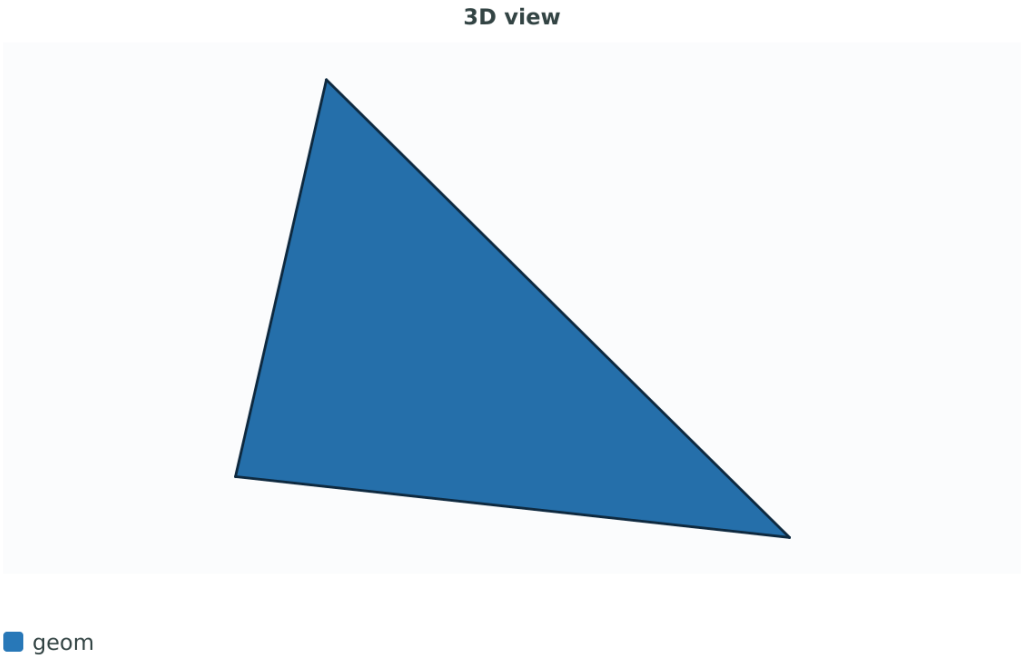
Code

```
SELECT ST_3DPerimeter(geom), ST_Perimeter2D(geom), ST_Perimeter(geom) FROM
            (SELECT 'SRID=2249;POLYGON((743238 2967416 2,743238 2967450 1,
743265.625 2967416 1,743238 2967416 2))'::geometry As geom) As foo;
```

Output

```
ST_3DPerimeter  |  st_perimeter2d  |   st_perimeter
-----------------+-----------------+-----------------
105.46579359767418 | 105.43299727218763 | 105.43299727218763
```

Figure



Se även

ST_GeomFromEWKT, ST_Perimeter, ST_Perimeter2D

7.12.25 ST_ShortestLine

ST_ShortestLine — Returnerar den kortaste 2D-linjen mellan två geometrier

Synopsis

geometry **ST_ShortestLine**(geometry geom1, geometry geom2);
geography **ST_ShortestLine**(geography geom1, geography geom2, boolean use_spheroid = true);

Beskrivning

Returnerar den 2-dimensionella kortaste linjen mellan två geometrier. Den returnerade linjen börjar i `geom1` och slutar i `geom2`. Om `geom1` och `geom2` skär varandra blir resultatet en linje med start och slut i en skärningspunkt. Linjens längd är densamma som ST_Distance returnerar för g1 och g2.

Förbättrad: 3.4.0 - stöd för geografi.

Tillgänglighet: 1.5.0

Exempel

Shortest line between a Point and a LineString.

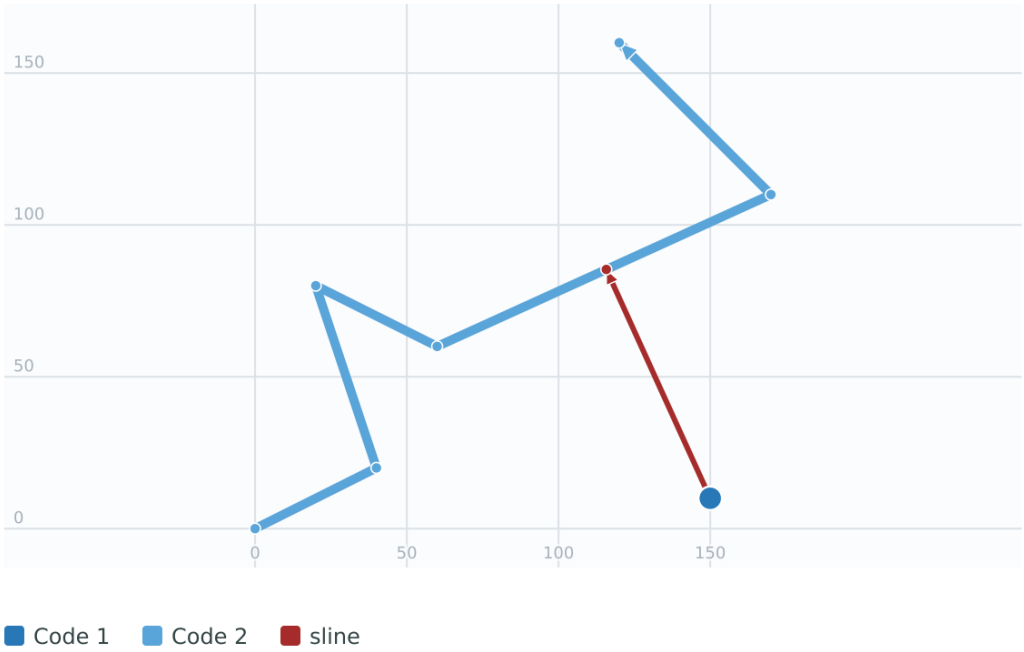
Code

```
SELECT ST_ShortestLine(
        'POINT (160 40)',
        'LINESTRING (10 30,50 50,30 110,70 90,180 140,130 190)') As sline;
```

Output

```
LINESTRING(160 40,125.75342465753425 115.34246575342466)
```

Figure



Shortest line between polygons.

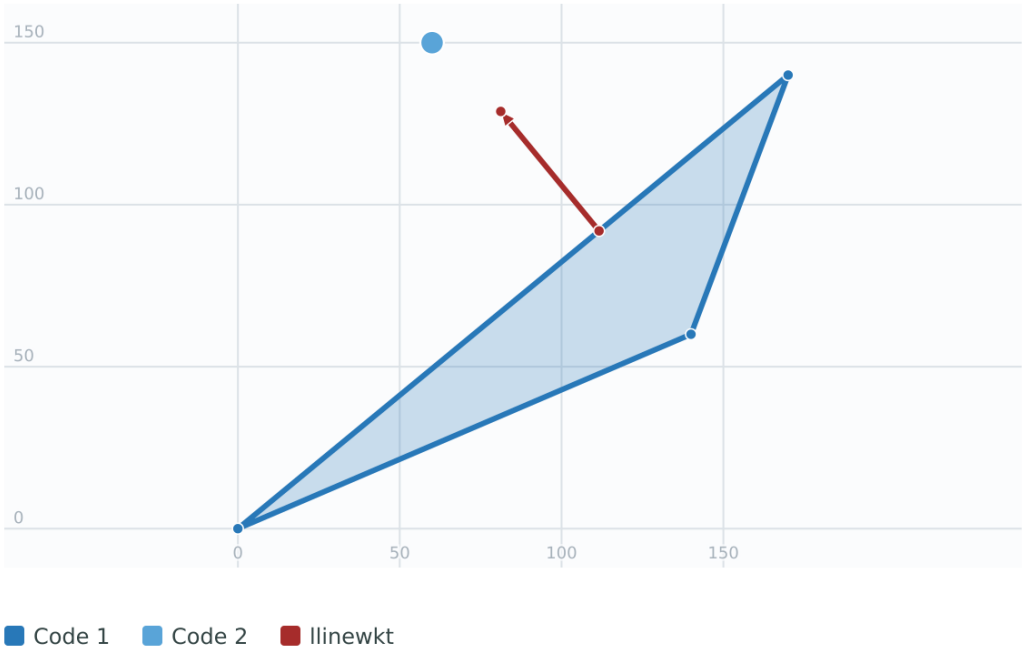
Code

```
SELECT ST_ShortestLine(
         'POLYGON ((190 150,20 10,160 70,190 150))',
         ST_Buffer('POINT(80 160)', 30)
                 ) AS llinewkt;
```

Output

```
LINESTRING(131.59149149528952 101.89887534906195,101.21320343559643 138.78679656440357)
```

Figure



Se även

ST_ClosestPoint, ST_Distance, ST_LongestLine, ST_MaxDistance

7.12.26 ST_3DShortestLine

ST_3DShortestLine — Returnerar den kortaste 3D-linjen mellan två geometrier

Synopsis

geometry **ST_3DShortestLine**(geometry g1, geometry g2);

Beskrivning

Returnerar den 3-dimensionella kortaste linjen mellan två geometrier. Funktionen returnerar endast den första kortaste linjen om det finns fler än en som funktionen hittar. Om g1 och g2 skär varandra i endast en punkt returnerar funktionen en linje med både start och slut i den skärningspunkten. Om g1 och g2 skär varandra i mer än en punkt kommer funktionen att returnera en linje med start och slut i samma punkt, men det kan vara vilken som helst av de skärande punkterna. Linjen som returneras börjar alltid i g1 och slutar i g2. 3D-längden på linjen som denna funktion returnerar kommer alltid att vara densamma som ST_3DDistance returnerar för g1 och g2.

Tillgänglighet: 2.0.0

Ändrad: 2.2.0 - om 2 2D-geometrier matas in returneras en 2D-punkt (istället för det gamla beteendet som antog 0 för saknad Z). Vid 2D och 3D antas Z inte längre vara 0 för saknad Z.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Exempel

Compare the 3D shortest line to the 2D shortest line for a LineString and a Point.

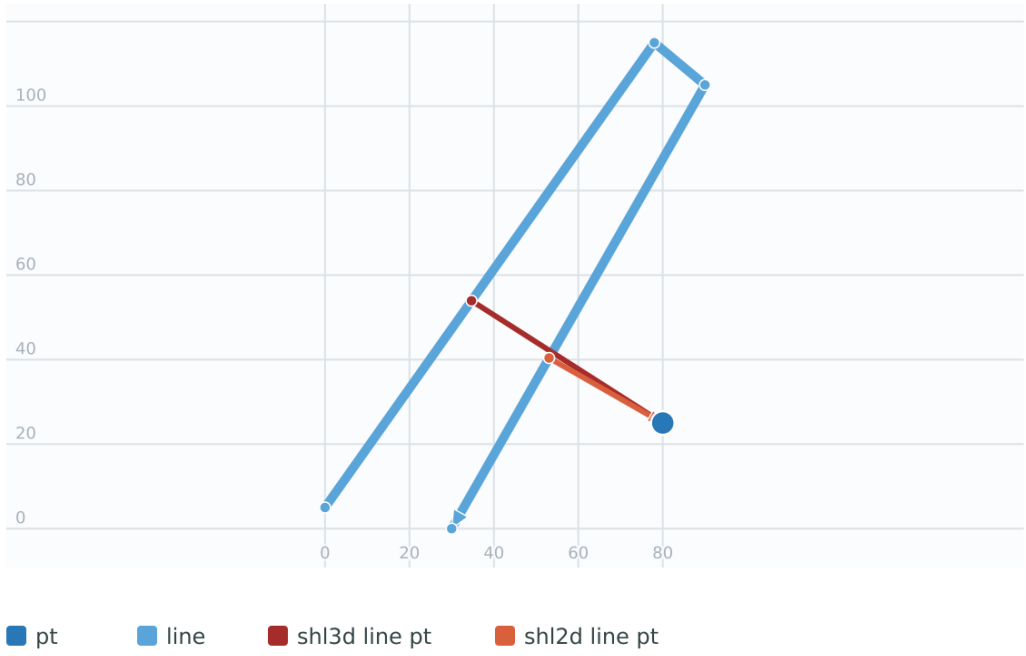
Code

```
WITH input AS (
  SELECT 'POINT(100 100 30)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 1000)'::geometry AS line
)
SELECT ST_3DShortestLine(line, pt) AS shl3d_line_pt,
       ST_ShortestLine(line, pt) AS shl2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+-------------------------------------------------------------------------------
shl3d_line_pt  | LINESTRING(54.69937988676193 128.93502291722837 11.547586950660556,100 100 30)
shl2d_line_pt  | LINESTRING(73.07692307692307 115.38461538461539,100 100)
```

Figure



Compare the 3D shortest line to the 2D shortest line for a LineString and a MultiPoint.

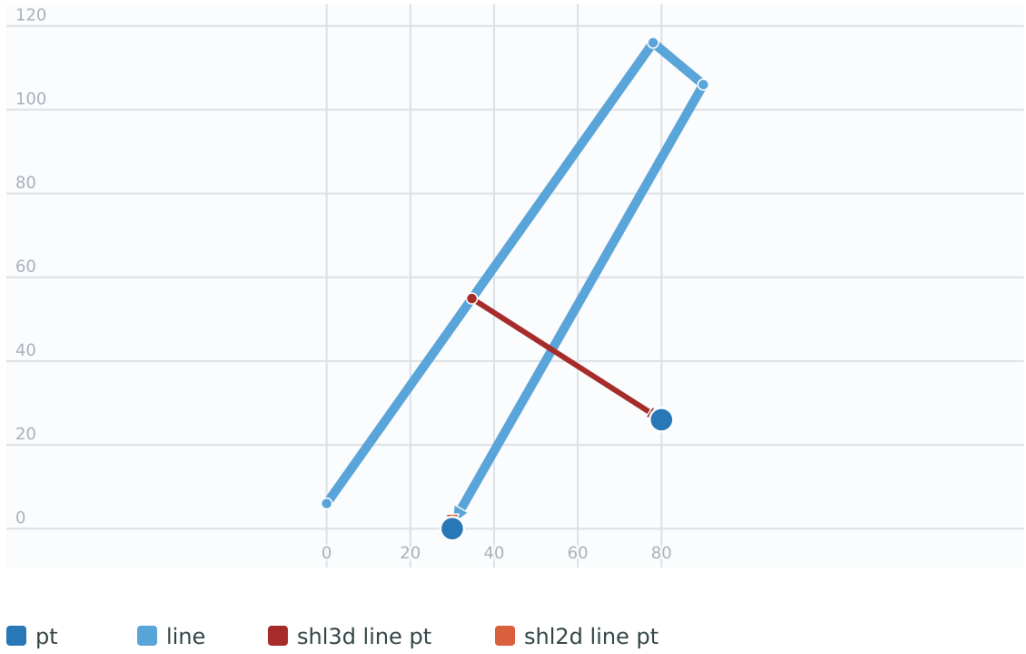
Code

```
WITH input AS (
  SELECT 'MULTIPOINT(100 100 30,50 74 1000)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 900)'::geometry AS line
)
SELECT ST_3DShortestLine(line, pt) AS shl3d_line_pt,
       ST_ShortestLine(line, pt) AS shl2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+-------------------------------------------------------------------------------
shl3d_line_pt  | LINESTRING(54.69937988676193 128.93502291722837 11.547586950660556,100 100 30)
shl2d_line_pt  | LINESTRING(50 75,50 74)
```

Figure



Compare the 3D shortest line to the 2D shortest line for a Polygon and a MultiLineString.

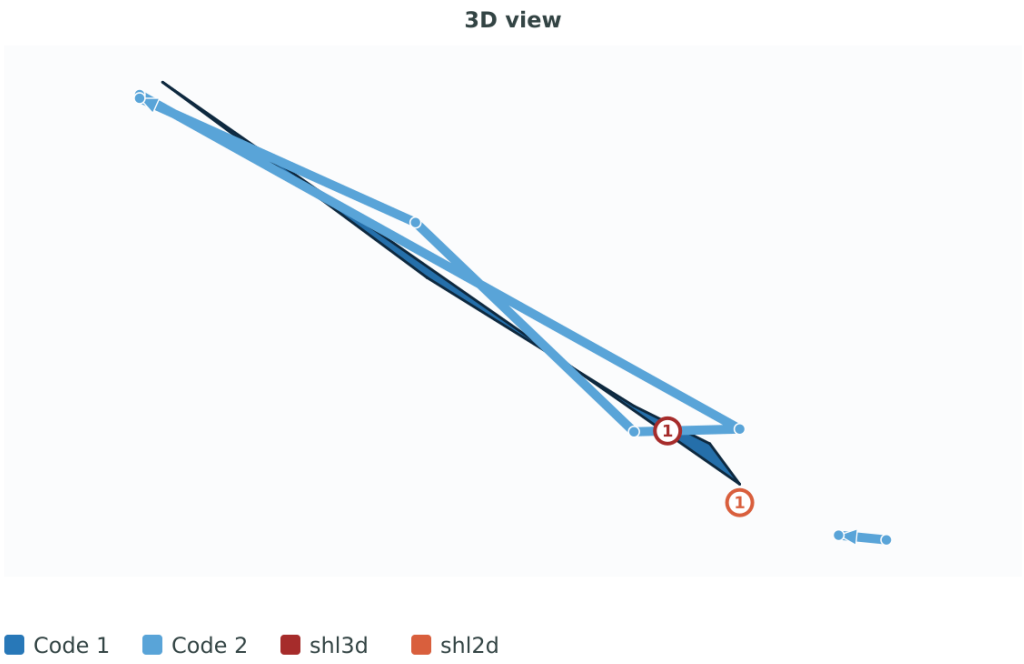
Code

```
WITH input AS (
  SELECT ST_GeomFromEWKT(
           'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))'
         ) AS poly,
         ST_GeomFromEWKT(
           'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),
                            (1 10 2,5 20 1))'
         ) AS mline
)
SELECT ST_3DShortestLine(poly, mline) AS shl3d,
       ST_ShortestLine(poly, mline) AS shl2d
FROM input;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------------------------------------
shl3d          | LINESTRING(40.45454545454545 53.63636363636364 5,40.45454545454545 53.63636363636364 5)
shl2d          | LINESTRING(20 40,20 40)
```

Figure



Se även

ST_3DClosestPoint, ST_3DDistance, ST_LongestLine, ST_ShortestLine, ST_3DMaxDistance

7.13 Funktioner för överlagring

7.13.1 ST_ClipByBox2D

ST_ClipByBox2D — Beräknar den del av en geometri som faller inom en rektangel.

Synopsis

geometry **ST_ClipByBox2D**(geometry geom, box2d box);

Beskrivning

Klipper en geometri med en 2D-box på ett snabbt och tolerant men eventuellt ogiltigt sätt. Topologiskt ogiltiga indatageometrier resulterar inte i att undantag kastas. Det är inte garanterat att utdatageometrin är giltig (i synnerhet kan självskärningar för en polygon introduceras).

Utförs av GEOS-modulen.

Tillgänglighet: 2.2.0

Exempel

This example relies on the implicit cast from geometry to box2d for the second parameter.

Code

```
SELECT ST_ClipByBox2D(geom, ST_MakeEnvelope(0, 0, 10, 10)) FROM mytab;
```

Se även

ST_Intersection, ST_MakeBox2D, ST_MakeEnvelope

7.13.2 ST_Difference

ST_Difference — Beräknar en geometri som representerar den del av geometri A som inte skär geometri B.

Synopsis

geometry **ST_Difference**(geometry geomA, geometry geomB, float8 gridSize = -1);

Beskrivning

Returnerar en geometri som representerar den del av geometri A som inte skär geometri B. Detta är ekvivalent med `A - ST_Intersection(A,B).` Om A är helt innesluten i B returneras en tom atomgeometri av lämplig typ.

Notera
Detta är den enda överlagringsfunktionen där inmatningsordningen har betydelse. ST_Difference(A, B) returnerar alltid en del av A.

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

Utförs av GEOS-modulen

Förbättrad: 3.1.0 accepterar en gridSize-parameter.

Kräver GEOS >= 3.9.0 för att använda parametern gridSize.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.20

Denna funktion stöder 3d och kommer inte att tappa z-index. Resultatet beräknas dock endast med hjälp av XY. De resulterande Z-värdena kopieras, medelvärdesbildas eller interpoleras.

Exempel

Skillnaden mellan 2D-linestrings.

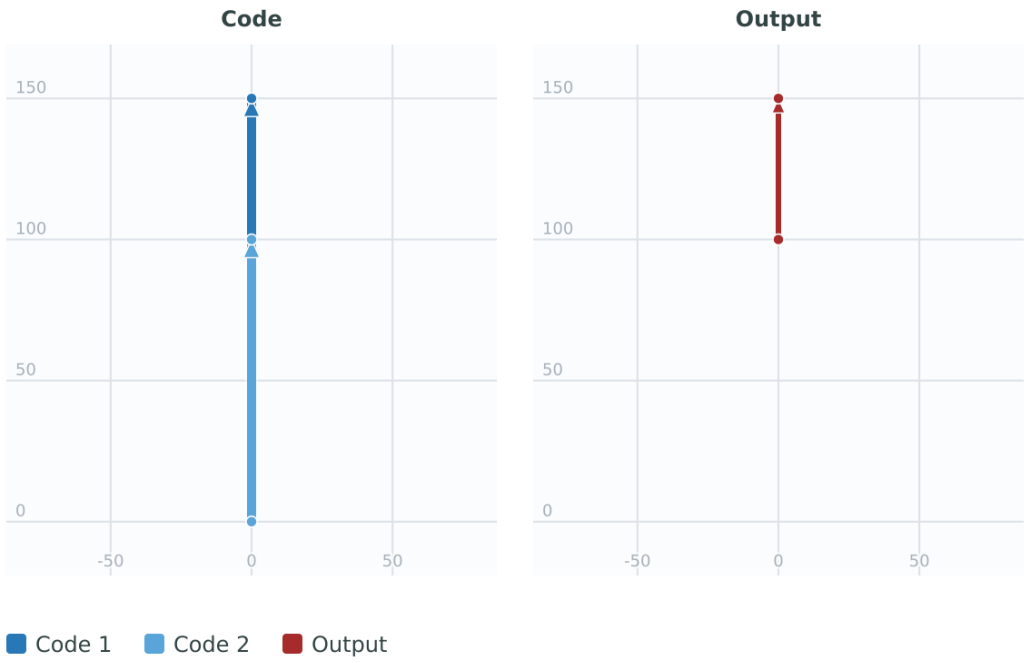
Code

```
SELECT ST_Difference('LINESTRING(50 100,50 200)'::geometry,
        'LINESTRING(50 50,50 150)'::geometry
    );
```

Output

```
LINESTRING(50 150,50 200)
```

Figure



Skillnaden mellan 3D-punkter.

Code

```
SELECT ST_Difference('MULTIPOINT(-118.58 38.38 5,-118.60 38.329 6,-118.614 38.281 7)' ::  ↩
    geometry,
 'POINT(-118.614 38.281 5)' :: geometry
);
```

Output

```
MULTIPOINT(-118.6 38.329 6,-118.58 38.38 5)
```

Figure

Se även

ST_SymDifference, ST_Intersection, ST_Union, ST_ReducePrecision

7.13.3 ST_Intersection

ST_Intersection — Beräknar en geometri som representerar den delade delen av geometrierna A och B.

Synopsis

geometry **ST_Intersection**(geometry geomA , geometry geomB , float8 gridSize = -1);
geography **ST_Intersection**(geography geogA , geography geogB);

Beskrivning

Returnerar en geometri som representerar punktuppsättningens skärningspunkt mellan två geometrier. Med andra ord, den del av geometri A och geometri B som delas mellan de två geometrierna.

Om geometrierna inte har några gemensamma punkter (dvs. är disjunkta) returneras en tom atomgeometri av lämplig typ.

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

ST_Intersection i kombination med ST_Intersects är användbart för att klippa geometrier, t.ex. i frågor om bounding box, buffert eller region där du bara behöver den del av en geometri som ligger inom ett land eller en region av intresse.

Notera
Spatial predicates such as ST_Intersects and overlay functions such as `ST_Intersection` are computed by different algorithms. For very close or nearly coincident coordinates, floating-point precision and geometric robustness may make a predicate return true while the computed intersection is empty, or place an intersection point slightly away from the input line. For repeatable overlay results, snap inputs to a suitable grid first, for example with ST_ReducePrecision, or use the `gridSize` parameter when all input vertices already lie on the grid.

Notera

För geografin är detta ett tunt hölje runt geometriimplementeringen. Den bestämmer först den bästa SRID som passar avgränsningsrutan för de 2 geografiobjekten (om geografiobjekt är inom en halv zon UTM men inte samma UTM väljer en av dem) (gynnar UTM eller Lambert Azimuthal Equal Area (LAEA) nord / sydpol och faller tillbaka på Mercator i värsta fall) och sedan intersektion i den bästa passande plana spatiala ref och retransformerar tillbaka till WGS84-geografi.

Varning
Denna funktion kommer att släppa M-koordinatvärdena om de finns.

Varning
If working with 3D geometries, you may want to use SFCGAL based CG_3DIntersection which does a proper 3D intersection for 3D geometries. Although this function works with Z-coordinate, it does an averaging of Z-Coordinate.

Utförs av GEOS-modulen

Förbättrad: 3.1.0 accepterar en gridSize-parameter

Kräver GEOS >= 3.9.0 för att använda parametern gridSize

Ändrat: 3.0.0 är inte beroende av SFCGAL.

Tillgänglighet: 1.5 stöd för datatypen Geography infördes.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.18

Denna funktion stöder 3d och kommer inte att tappa z-index. Resultatet beräknas dock endast med hjälp av XY. De resulterande Z-värdena kopieras, medelvärdesbildas eller interpoleras.

Exempel

Code

```
SELECT ST_Intersection('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
PUNKT EMPTY
```

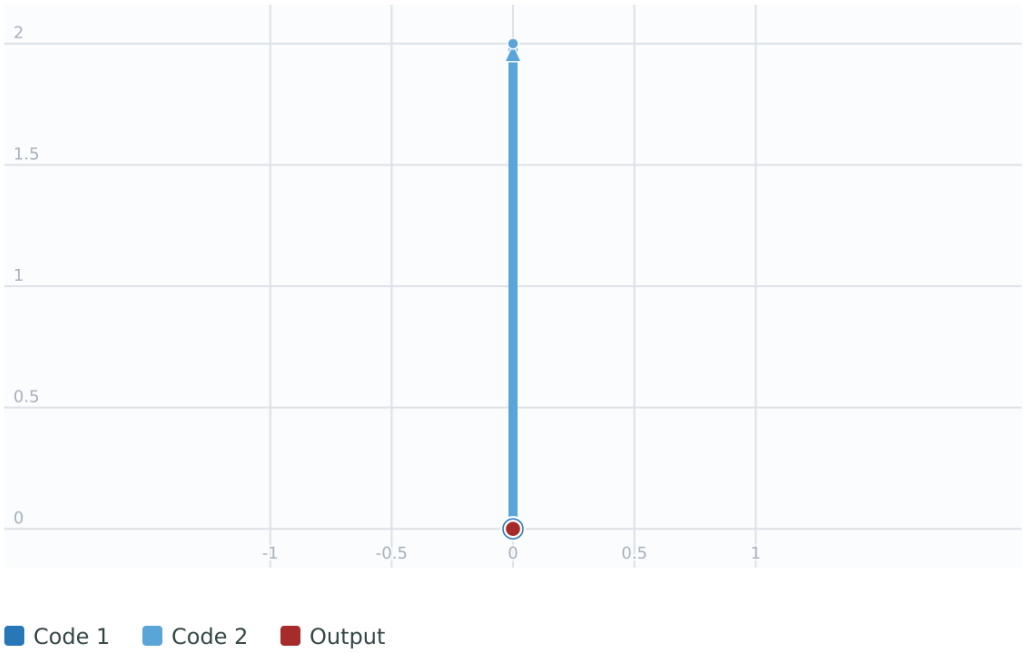
Code

```
SELECT ST_Intersection('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Output

```
PUNKT(0 0)
```

Figure



Klipp ut alla linjer (spår) efter land. Här antar vi att landets geom är POLYGON eller MULTIPOLYGON. OBS: vi behåller bara korsningar som resulterar i en LINESTRING eller MULTILINESTRING eftersom vi inte bryr oss om spår som bara delar en punkt. Dumpen behövs för att expandera en geometrisamling till enskilda MULT*-delar. Nedanstående är ganska generisk och kommer att fungera för polys, etc. genom att bara ändra var klausulen.

Code

```
select clipped.gid, clipped.f_name, clipped_geom
from (
         select trails.gid, trails.f_name,
             (ST_Dump(ST_Intersection(country.geom, trails.geom))).geom clipped_geom
         from country
              inner join trails on ST_Intersects(country.geom, trails.geom)
     ) as clipped
where ST_Dimension(clipped.clipped_geom) = 1;
```

För polys, t.ex. polygonlandmärken, kan du också använda det ibland snabbare hacket att buffring av allt med 0,0 utom en polygon resulterar i en tom geometrisamling. (Så en geometrisamling som innehåller polygoner, linjer och punkter som buffras med 0,0 skulle bara lämna kvar polygonerna och upplösa samlingens skal)

Code

```
select poly.gid,
    ST_Multi(ST_Buffer(ST_Intersection(country.geom, poly.geom),
            0.0
        )
    ) clipped_geom
from country
     inner join poly on ST_Intersects(country.geom, poly.geom)
where not ST_IsEmpty(ST_Buffer(ST_Intersection(country.geom, poly.geom), 0.0));
```

Note this is not a true 3D intersection. It uses 2.5D geometries (geometries with Z ordinates, but where calculations are performed in 2D). The same input is compared with CG_3DIntersection, which performs a true 3D intersection.

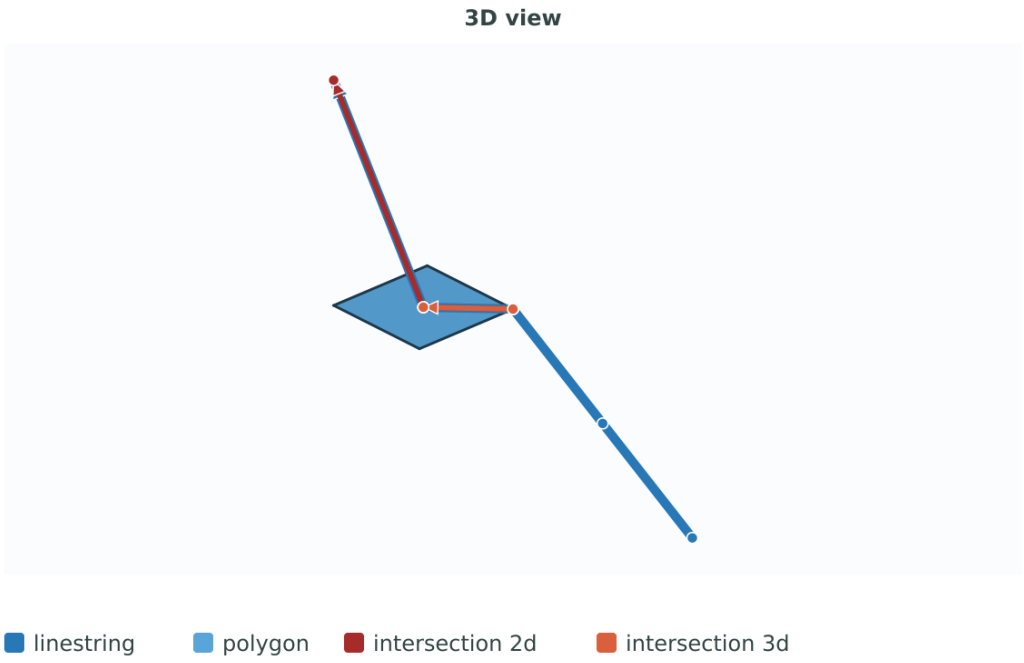
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)'::geometry AS
    input_linestring,
         'POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8))'::geometry AS input_polygon
)
SELECT input_linestring AS input_linestring,
       input_polygon AS input_polygon,
       ST_Intersection(input_linestring, input_polygon) AS intersection_2d,
       CG_3DIntersection(input_linestring, input_polygon) AS intersection_3d
FROM data;
```

Output

```
LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10) | POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8)) | LINESTRING Z (1
    1 8,0.5 0.5 8,0 0 10) | LINESTRING Z (1 1 8,0.5 0.5 8)
```

Figure



Se även

CG_3DIntersection, ST_Difference, ST_Union, ST_ClipByBox2D, ST_Dimension, ST_Dump, ST_Force2D, ST_SymDifference, ST_Intersects, ST_Multi, ST_ReducePrecision

7.13.4 ST_MemUnion

ST_MemUnion — Aggregatfunktion som kombinerar geometrier på ett minneseffektivt men långsammare sätt

Synopsis

geometry **ST_MemUnion**(geometry set geomfield);

Beskrivning

En aggregeringsfunktion som förenar indatageometrierna och sammanfogar dem för att producera en resultatgeometri utan överlappningar. Utdata kan vara en enskild geometri, en MultiGeometry eller en Geometry Collection.

Notera
Ger samma resultat som ST_Union, men använder mindre minne och mer processortid. Denna aggregatfunktion fungerar genom att geometrierna sammanfogas stegvis, i motsats till ST_Union-aggregatet som först ackumulerar en matris och sedan sammanfogar innehållet med hjälp av en snabb algoritm.

Denna funktion stöder 3d och kommer inte att tappa z-index. Resultatet beräknas dock endast med hjälp av XY. De resulterande Z-värdena kopieras, medelvärdesbildas eller interpoleras.

Exempel

Code

```
SELECT id,
       ST_MemUnion(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

Se även

ST_Union

7.13.5 ST_Node

ST_Node — Noder en samling av linjer.

Synopsis

geometry **ST_Node**(geometry geom);

Beskrivning

Returnerar en (Multi)LineString som representerar den fullständigt nodade versionen av en samling linestrings. Nodningen bevarar alla indatans noder och introducerar minsta möjliga antal nya noder. Det resulterande linjeverket är upplöst (dubbletter av linjer tas bort).

Detta är ett bra sätt att skapa helt kodade linjearbeten som kan användas som underlag för ST_Polygonize.

ST_UnaryUnion kan också användas för att nodera och lösa upp linjeverk. Det finns ett alternativ för att ange en gridSize, vilket kan ge enklare och mer robusta resultat. Se även ST_Union för en aggregerad variant.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Utförs av GEOS-modulen.

Tillgänglighet: 2.0.0

Ändrad: 2.4.0 denna funktion använder GEOSNode internt istället för GEOSUnaryUnion. Detta kan leda till att de resulterande linestrings har en annan ordning och riktning jämfört med PostGIS < 2.4.

Exempel

Noding a 3D LineString which self-intersects.

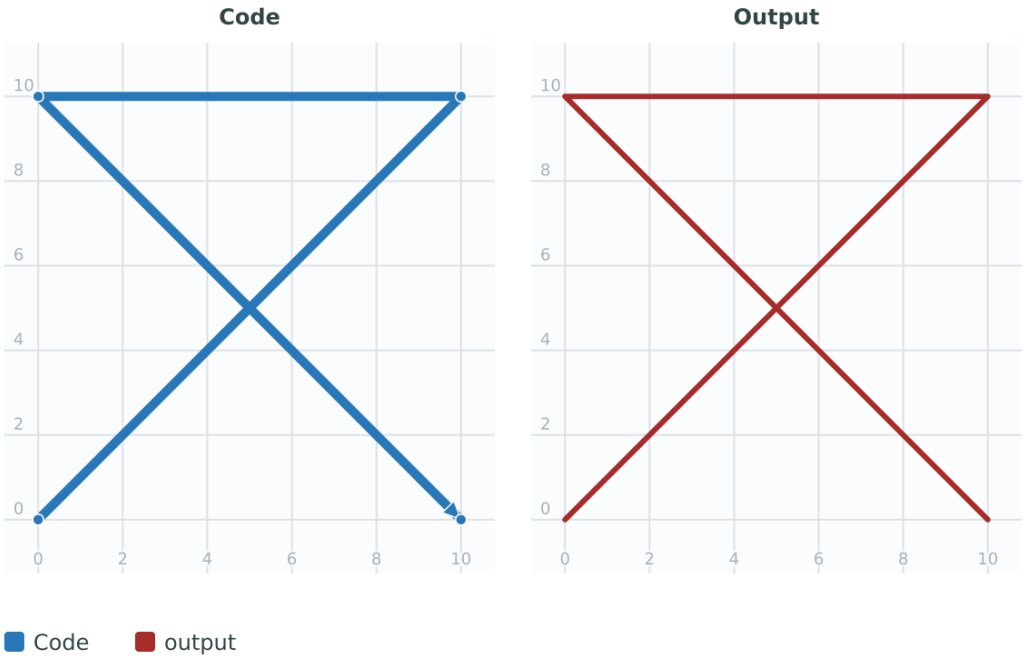
Code

```
SELECT ST_Node('LINESTRINGZ(0 0 0, 10 10 10, 0 10 5, 10 0 3)'::geometry) As  output;
```

Output

```
MULTILINESTRING Z ((0 0 0,5 5 4.5),(5 5 4.5,10 10 10,0 10 5,5 5 4.5),(5 5 4.5,10 0 3))
```

Figure



Kodning av två LineStrings som delar gemensamt linework. Observera att resultatets linjeföring är upplöst.

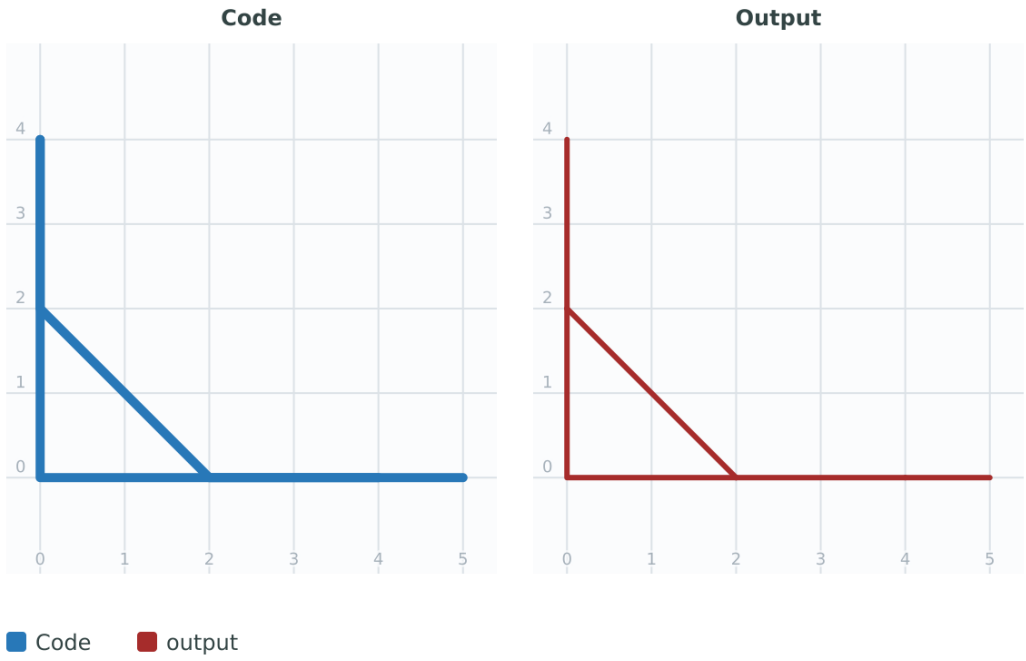
Code

```
SELECT ST_Node('MULTILINESTRING ((2 5,2 1,7 1),(6 1,4 1,2 3,2 5))'::geometry) As  output;
```

Output

```
MULTILINESTRING((2 5,2 3),(2 3,2 1,4 1),(4 1,2 3),(4 1,6 1),(6 1,7 1))
```

Figure



Se även

ST_UnaryUnion, ST_Union

7.13.6 ST_Split

ST_Split — Returnerar en samling geometrier som skapats genom att dela en geometri med en annan geometri.

Synopsis

geometry **ST_Split**(geometry input, geometry blade);

Beskrivning

Funktionen stöder delning av en LineString med en (Multi)Point-, (Multi)LineString- eller (Multi)Polygon-gräns, eller en (Multi)Polygon med en LineString. När en (Multi)Polygon används som blad, används

dess linjära komponenter (gränsen) för att dela upp indata. Den resulterande geometrin är alltid en samling.

Denna funktion är på sätt och vis motsatsen till ST_Union. Genom att tillämpa ST_Union på den returnerade samlingen bör man teoretiskt sett få fram den ursprungliga geometrin (även om detta kanske inte är exakt fallet på grund av numerisk avrundning).

Notera
Om inmatningen och bladet inte korsar varandra på grund av problem med numerisk precision kanske inmatningen inte delas som förväntat. För att undvika denna situation kan det vara nödvändigt att först fästa inmatningen på bladet med hjälp av ST_Snap med en liten tolerans.

Tillgänglighet: 2.0.0 kräver GEOS

Förbättrad: 2.2.0 stöd för att dela en linje med en multilinje, en multipunkt eller (multi)polygongräns infördes.

Förbättrad: 2.5.0 stöd för att dela en polygon med en multilinje infördes.

Exempel

Split a circular polygon by a diagonal line.

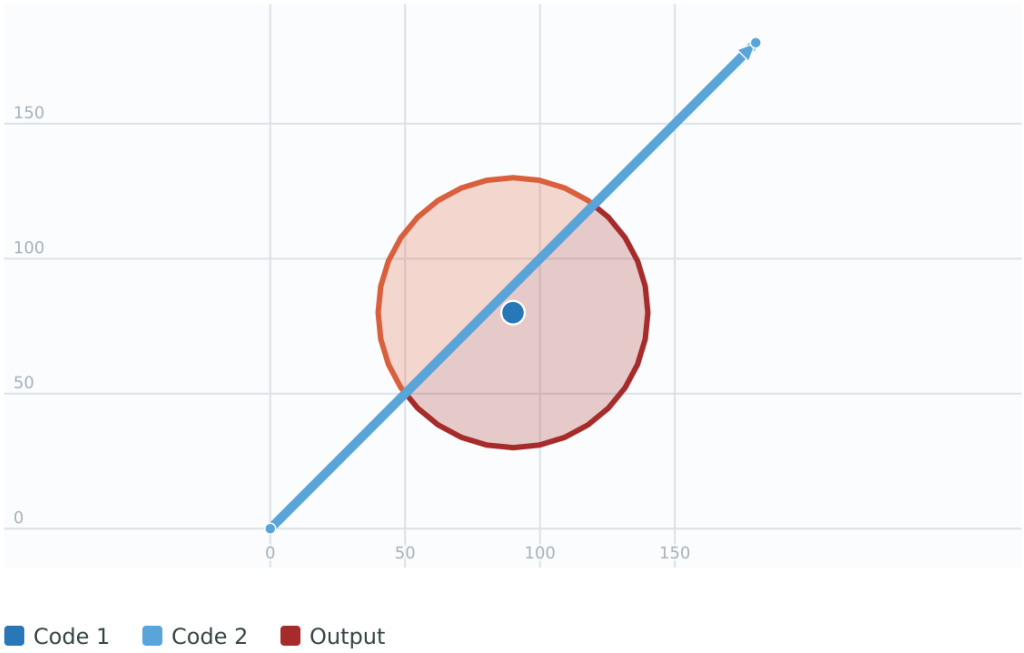
Code

```
SELECT
         ST_Split(
           ST_Buffer('POINT(100 90)'::geometry, 50),
           'LINESTRING(10 10,190 190)'::geometry
         );
```

Output

```
GEOMETRYCOLLECTION(POLYGON((150 90,149 80.2,146.2 70.9,141.6 62.2,135.4 54.6,127.8 48.4,119.1 43.8,109.8 41,100 ↩
    40,90.2 41,80.9 43.8,72.2 48.4,64.6 54.6,60.1 60.1,129.9 129.9,135.4 125.4,141.6 117.8,146.2 109.1,149 99.8,150 ↩
    90)), POLYGON((60.1 60.1,58.4 62.2,53.8 70.9,51 80.2,50 90,51 99.8,53.8 109.1,58.4 117.8,64.6 125.4,72.2 ↩
    131.6,80.9 136.2,90.2 139,100 140,109.8 139,119.1 136.2,127.8 131.6,129.9 129.9,60.1 60.1)))
```

Figure



Dela en MultiLineString genom en punkt, där punkten ligger exakt på båda LineStrings element.

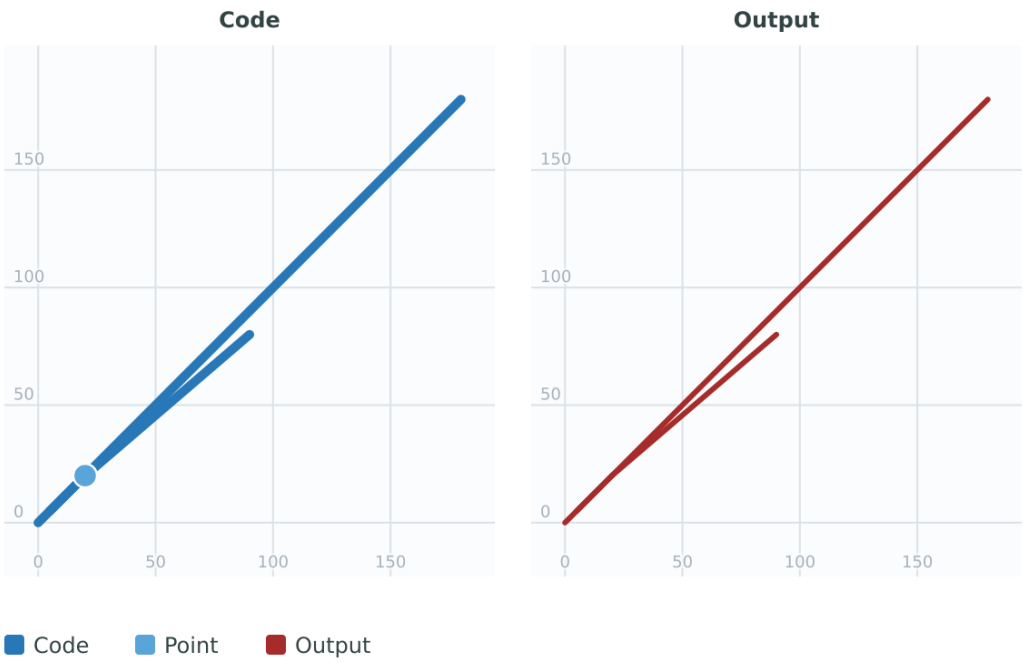
Code

```
SELECT ST_Split('MULTILINESTRING((10 10,190 190),(15 15,30 30,100 90))',
    ST_Point(30, 30));
```

Output

```
GEOMETRYCOLLECTION(
    LINESTRING(10 10,30 30),
    LINESTRING(30 30,190 190),
    LINESTRING(15 15,30 30),
    LINESTRING(30 30,100 90)
)
```

Figure



Dela en LineString med en punkt, där punkten inte ligger exakt på linjen. Visar hur man använder ST_Snap för att snäppa linjen till punkten så att den kan delas.

Code

```
WITH data AS (SELECT
  'LINESTRING(0 0,100 100)'::geometry AS line,
  'POINT(51 50)':: geometry AS point
)
SELECT ST_Split(ST_Snap(line, point, 1), point) AS snapped_split,
       ST_Split(line, point) AS not_snapped_not_split
       FROM data;
```

Figure

Se även

ST_Snap, ST_Union

7.13.7 ST_Subdivide

ST_Subdivide — Beräknar en rätlinjig subdivision av en geometri.

Synopsis

setof geometry **ST_Subdivide**(geometry geom, integer max_vertices=256, float8 gridSize = -1);

Beskrivning

Returnerar en uppsättning geometrier som är resultatet av att dela in `geom` i delar med hjälp av rätlinjiga linjer, där varje del inte innehåller mer än `max_vertices`.

`max_vertices` måste vara 5 eller mer, eftersom det behövs 5 punkter för att representera en sluten låda.

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

Punkt-i-polygon och andra spatiala operationer är normalt snabbare för indexerade, indelade datamängder. Eftersom avgränsningsboxarna för delarna vanligtvis täcker ett mindre område än den ursprungliga geometrins bbox, ger indexfrågor färre ”hit-fall. Träff-fallen går snabbare eftersom de spatiala operationer som utförs av indexets omkontroll bearbetar färre punkter.

Notera
When casting a subdivided geometry to geography, the resulting geography may differ from the original. Subdivision adds vertices in planar (geometry) space. If vertices are inserted along the boundary, they will alter the geographical representation, where edges are interpreted as geodesic segments. To minimize distortion, first densify the geography using ST_Segmentize to add geodesic vertices, then cast to geometry before subdivision.

Notera
Detta är en set-returning function (SRF) som returnerar en uppsättning rader som innehåller enskilda geometrivärden. Den kan användas i en SELECT-lista eller en FROM-sats för att producera en resultatuppsättning med en post för varje resultatgeometri.

Utförs av GEOS-modulen.

Tillgänglighet: 2.2.0

Förbättrad: 2.5.0 återanvänder befintliga punkter vid polygondelning, antalet vertex sänks från 8 till 5.

Förbättrad: 3.1.0 accepterar en gridSize-parameter.

Kräver GEOS >= 3.9.0 för att använda parametern gridSize

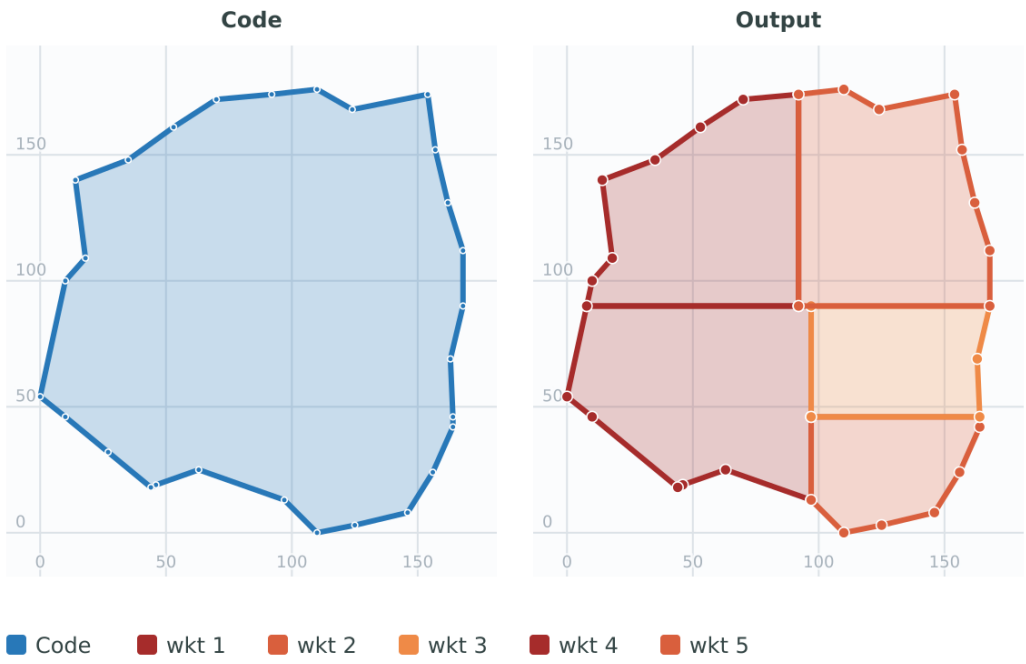
Exempel

Exempel: Dela upp en polygon i delar med högst 10 hörn och tilldela varje del ett unikt id.

Code

```
SELECT row_number() OVER() As rn,geom As wkt
    FROM (SELECT ST_Subdivide('POLYGON((132 10,119 23,85 35,68 29,66 28,49 42,32 56,22 ↩
    64,32 110,40 119,36 150,
        57 158,75 171,92 182,114 184,132 186,146 178,176 184,179 162,184 141,190 122,
        190 100,185 79,186 56,186 52,178 34,168 18,147 13,132 10))'::geometry, 10))  AS f( ↩
    geom);
```

Figure



Example: Densify a long geography line using ST_Segmentize(geography, distance), and use ST_Subdivide to split the resulting line into sublines of 8 vertices. Densification minimizes the impact of changes to the geography representation of a geometry when subdividing.

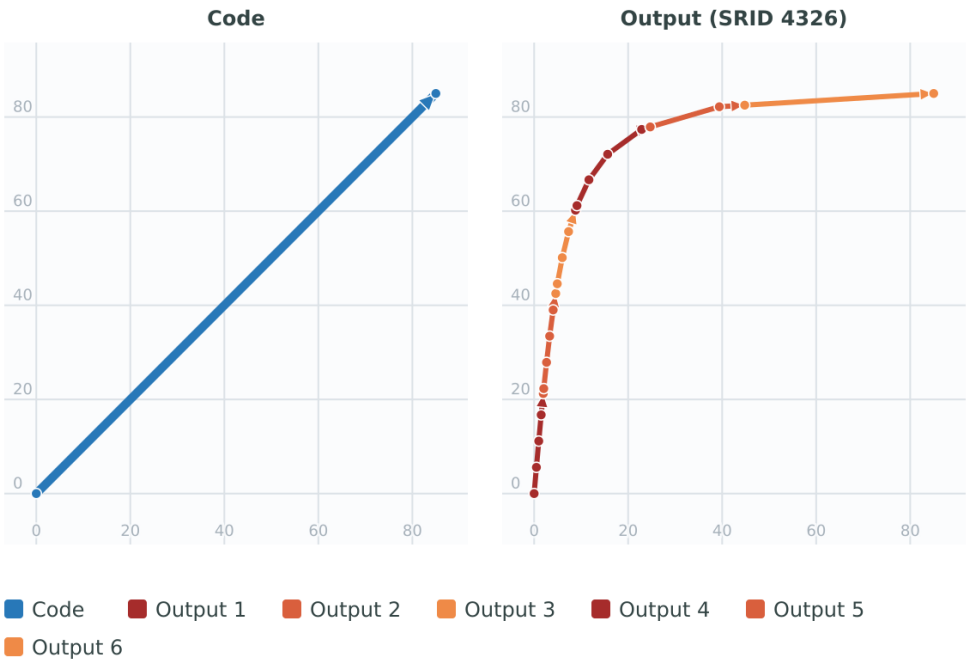
Code

```
SELECT ST_Subdivide(ST_Segmentize('LINESTRING(0 0,85 85)'::geography,
                            1200000)::geometry, 8);
```

Output

```
LINESTRING(0 0,0.487578359029357 5.57659056746196,0.984542144675897 11.1527721155093,1.50101059639722  ↩
    16.7281035483571,1.94532113630331 21.25)
LINESTRING(1.94532113630331 21.25,2.04869538062779 22.3020741387339,2.64204641967673 27.8740533545155,3.29994062412787 ↩
     33.443216802941,4.04836719489742 39.0084282520239,4.59890468420694 42.5)
LINESTRING(4.59890468420694 42.5,4.92498503922732 44.5680389206321,5.98737409390639 50.1195229244701,7.3290919767674  ↩
    55.6587646879025,8.79638749938413 60.1969505994924)
LINESTRING(8.79638749938413 60.1969505994924,9.11375579533779 61.1785363177625,11.6558166691368  ↩
    66.6648504160202,15.642041247655 72.0867690601745,22.8716627200212 77.3609628116894,24.6991785131552  ↩
    77.8939011989848)
LINESTRING(24.6991785131552 77.8939011989848,39.4046096622744 82.1822848017636,44.7994523421035 82.5156766227011)
LINESTRING(44.7994523421035 82.5156766227011,85 85)
```

Figure



Ett exempel: Dela upp de komplexa geometrierna i en tabell på plats. De ursprungliga geometriposterna tas bort från källtabellen och nya poster för varje underindelad resultatgeometri infogas.

Code

```
WITH complex_areas_to_subdivide AS (
    DELETE from polygons_table
    WHERE ST_NPoints(geom)
> 255
    RETURNING id, column1, column2, column3, geom
)
INSERT INTO polygons_table (fid, column1, column2, column3, geom)
    SELECT fid, column1, column2, column3,
           ST_Subdivide(geom, 255) as geom
    FROM complex_areas_to_subdivide;
```

Exempel: Skapa en ny tabell som innehåller indelade geometrier och behåll nyckeln för den ursprungliga geometrin så att den nya tabellen kan kopplas till källtabellen. Eftersom ST_Subdivide är en set-returning (tabell)-funktion som returnerar en uppsättning rader med ett enda värde, producerar denna syntax automatiskt en tabell med en rad för varje resultatdel.

Code

```
CREATE TABLE subdivided_geoms AS
    SELECT pkey, ST_Subdivide(geom) AS geom
    FROM original_geoms;
```

Se även

ST_ClipByBox2D, ST_Segmentize, ST_Split, ST_NPoints, ST_ReducePrecision

7.13.8 ST_SymDifference

ST_SymDifference — Beräknar en geometri som representerar de delar av geometrierna A och B som inte korsar varandra.

Synopsis

geometry **ST_SymDifference**(geometry geomA, geometry geomB, float8 gridSize = -1);

Beskrivning

Returnerar en geometri som representerar de delar av geonetrierna A och B som inte skär varandra. Detta är likvärdigt med `ST_Union(A,B) - ST_Intersection(A,B)`. Det kallas en symmetrisk differens eftersom `ST_SymDifference(A,B) = ST_SymDifference(B,A)`..

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

Utförs av GEOS-modulen

Förbättrad: 3.1.0 accepterar en gridSize-parameter.

Kräver GEOS >= 3.9.0 för att använda parametern gridSize

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.21

Denna funktion stöder 3d och kommer inte att tappa z-index. Resultatet beräknas dock endast med hjälp av XY. De resulterande Z-värdena kopieras, medelvärdesbildas eller interpoleras.

Exempel

This example shows a 2D-safe symmetric difference of two LineStrings.

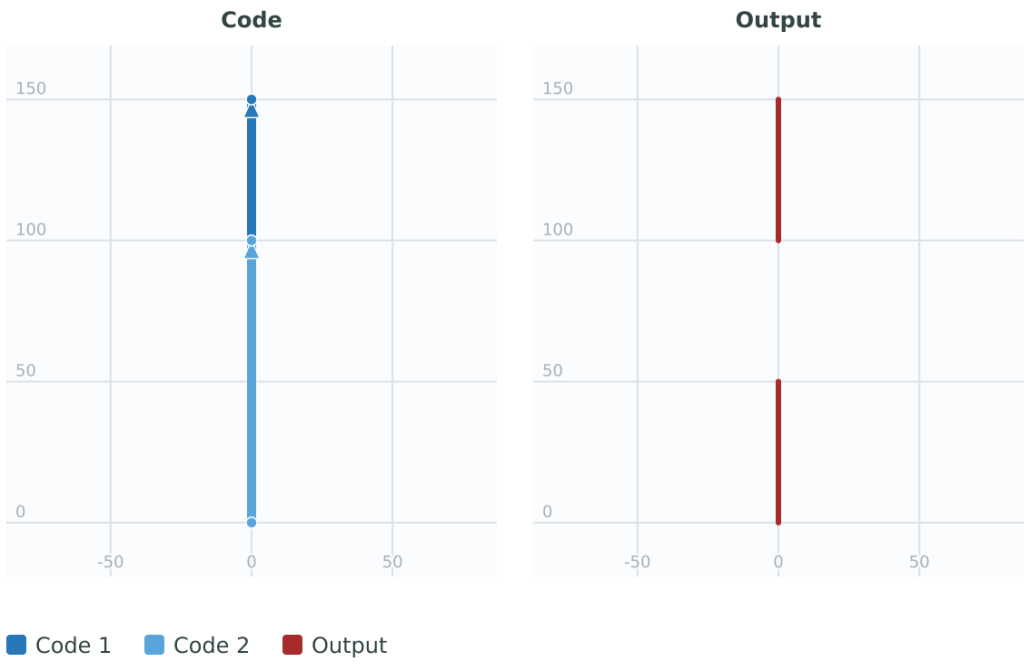
Code

```
SELECT ST_SymDifference('LINESTRING(50 100,50 200)',
        'LINESTRING(50 50,50 150)'
    );
```

Output

```
MULTILINESTRING((50 150,50 200),(50 50,50 100))
```

Figure



In 3D, this example does not produce the expected result.

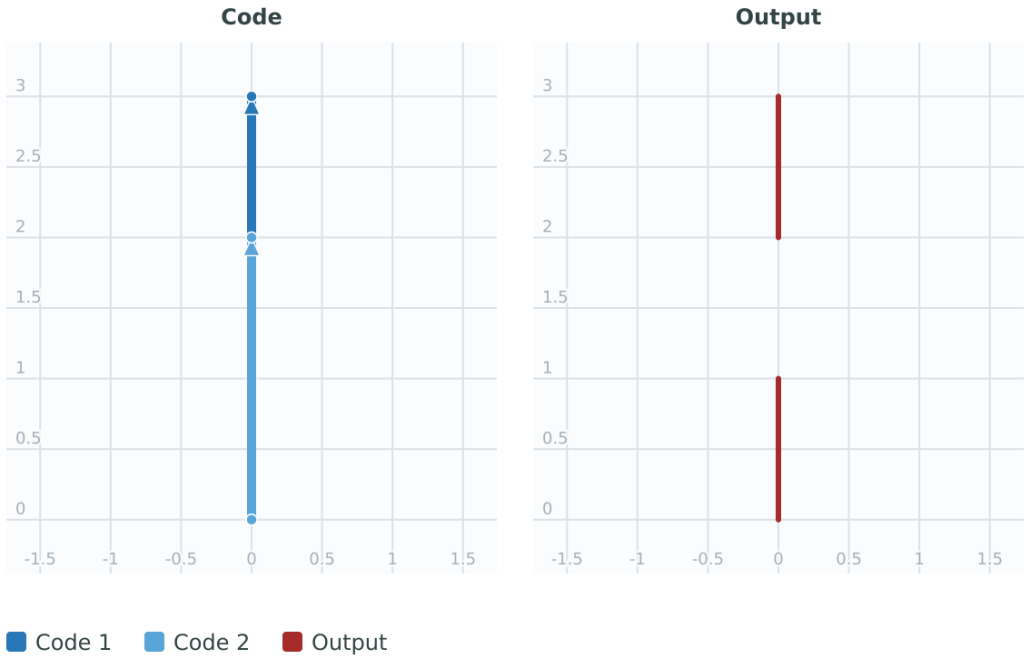
Code

```
SELECT ST_SymDifference('LINESTRING(1 2 1,1 4 2)',
    'LINESTRING(1 1 3,1 3 4)');
```

Output

```
MULTILINESTRING((1 3 4,1 4 2),(1 1 3,1 2 1))
```

Figure



Se även

ST_Difference, ST_Intersection, ST_Union, ST_ReducePrecision

7.13.9 ST_UnaryUnion

ST_UnaryUnion — Beräknar sammanslagningen av komponenterna i en enda geometri.

Synopsis

geometry **ST_UnaryUnion**(geometry geom, float8 gridSize = -1);

Beskrivning

En variant av ST_Union med en enda inmatning. Inmatningen kan vara en enskild geometri, en MultiGeometry eller en GeometryCollection. Unionen tillämpas på de enskilda elementen i indata.

Denna funktion kan användas för att fixa MultiPolygoner som är ogiltiga på grund av överlappande komponenter. Alla indatakomponenter måste dock vara giltiga. En ogiltig indatakomponent, t.ex. en polygon med fluga, kan orsaka ett fel. Av denna anledning kan det vara bättre att använda ST_MakeValid.

En annan användning av denna funktion är att nodera och lösa upp en samling linestrings som korsar eller överlappar varandra för att göra dem enkla.(ST_Node gör också detta, men det ger inte `gridSize-alternativet`.)

Det är möjligt att kombinera ST_UnaryUnion med ST_Collect för att finjustera hur många geometrier som ska förenas samtidigt. Detta möjliggör en avvägning mellan minnesanvändning och beräkningstid, vilket ger en balans mellan ST_Union och ST_MemUnion.

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

Denna funktion stöder 3d och kommer inte att tappa z-index. Resultatet beräknas dock endast med hjälp av XY. De resulterande Z-värdena kopieras, medelvärdesbildas eller interpoleras.

Förbättrad: 3.1.0 accepterar en gridSize-parameter.

Kräver GEOS >= 3.9.0 för att använda parametern gridSize

Tillgänglighet: 2.0.0

Se även

ST_Union, ST_MemUnion, ST_MakeValid, ST_Collect, ST_Node, ST_ReducePrecision

7.13.10 ST_Union

ST_Union — Beräknar en geometri som representerar punktuppsättningssammanslagningen av indatageometrierna.

Synopsis

geometry **ST_Union**(geometry g1, geometry g2);
geometry **ST_Union**(geometry g1, geometry g2, float8 gridSize);
geometry **ST_Union**(geometry[] g1_array);
geometry **ST_Union**(geometry set g1field);
geometry **ST_Union**(geometry set g1field, float8 gridSize);

Beskrivning

Förenar indatageometrierna och sammanfogar geometrierna för att producera en resultatgeometri utan överlappningar. Utdata kan vara en atomär geometri, en MultiGeometry eller en Geometry Collection. Finns i flera varianter:

Variant med två inmatningar: returnerar en geometri som är en förening av två inmatningsgeometrier. Om någon av indata är NULL, returneras NULL.

Array variant: returnerar en geometri som är en sammanslagning av en array av geometrier.

Aggregerad variant: returnerar en geometri som är föreningen av en raduppsättning geometrier. ST_Union () -funktionen är en äggregeradfunktion i terminologin för PostgreSQL. Det betyder att den fungerar på rader med data, på samma sätt som SUM () och AVG () -funktionerna gör och som de flesta aggregat ignorerar den också NULL-geometrier.

Se ST_UnaryUnion för en variant utan aggregat och med en enda inmatning.

ST_Union-arrayen och set-varianterna använder den snabba Cascaded Union-algoritmen som beskrivs i http://blog.cleverelephant.ca/2009/01/must-faster-unions-in-postgis-14.html

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

Notera
ST_Collect kan ibland användas i stället för ST_Union, om resultatet inte behöver vara icke-överlappande. ST_Collect är vanligtvis snabbare än ST_Union eftersom den inte utför någon bearbetning av de insamlade geometrierna.

Utförs av GEOS-modulen.

ST_Union skapar MultiLineString och syr inte ihop LineStrings till en enda LineString. Använd ST_LineMerg för att sy LineStrings.

OBS: den här funktionen hette tidigare GeomUnion(), som döptes om från ”Union”eftersom UNION är ett reserverat SQL-ord.

Förbättrad: 3.1.0 accepterar en gridSize-parameter.

Kräver GEOS >= 3.9.0 för att använda parametern gridSize

Ändrat: 3.0.0 är inte beroende av SFCGAL.

Tillgänglighet: 1.4.0 - ST_Union förbättrades. ST_Union (geomarray) introducerades och även snabbare aggregerad samling i PostgreSQL.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.3

Notera
Aggregerad version definieras inte uttryckligen i OGC SPEC.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 5.1.19 z-index (höjd) när polygoner är inblandade.

Denna funktion stöder 3d och kommer inte att tappa z-index. Resultatet beräknas dock endast med hjälp av XY. De resulterande Z-värdena kopieras, medelvärdesbildas eller interpoleras.

Exempel

Aggregerat exempel

Code

```
SELECT id,
       ST_Union(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

Icke-aggregerat exempel

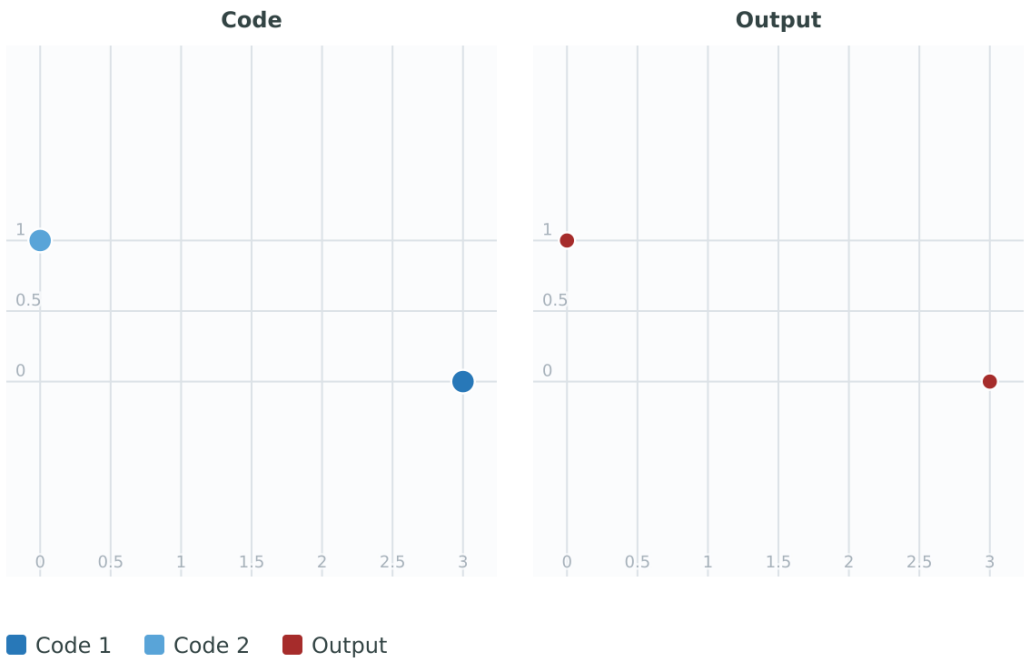
Code

```
SELECT ST_Union('POINT(1 2)'::geometry, 'POINT(-2 3)'::geometry);
```

Output

```
MULTIPOINT((1 2),(-2 3))
```

Figure



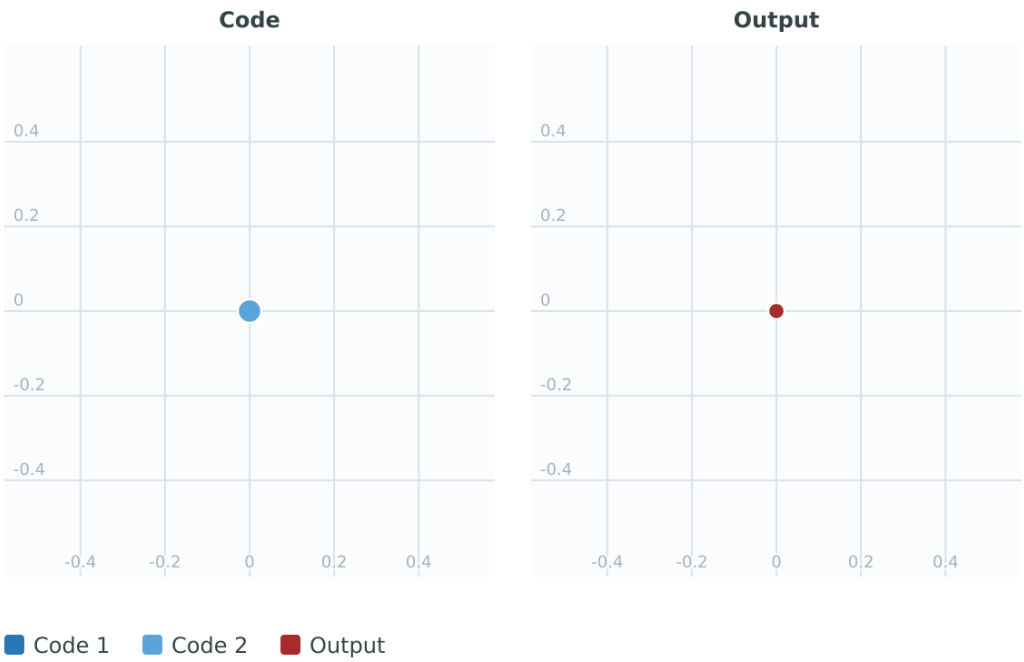
Code

```
SELECT ST_Union('POINT(1 2)'::geometry, 'POINT(1 2)'::geometry);
```

Output

```
POINT(1 2)
```

Figure



3D example - sort of supports 3D (and with mixed geometry types!)

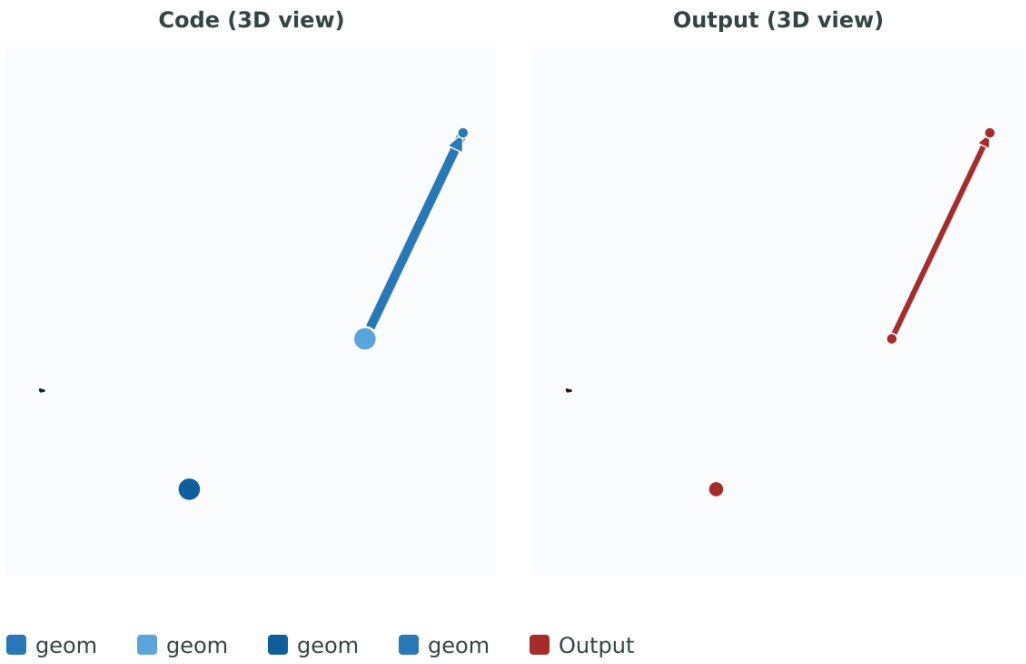
Code

```
SELECT ST_Union(geom)
FROM (
    SELECT 'POLYGON Z((-7 4.2 5,-7.1 4.2 5,-7.1 4.3 5,-7 4.2 5))'::geometry geom
    UNION ALL
    SELECT 'POINT Z(5 5 5)'::geometry geom
    UNION ALL
    SELECT 'POINT Z(-2 3 1)'::geometry geom
    UNION ALL
    SELECT 'LINESTRING Z(5 5 5,10 10 10)'::geometry geom
) AS foo;
```

Output

```
GEOMETRYCOLLECTION(POINT(-2 3 1), LINESTRING(5 5 5,10 10 10), POLYGON((-7 4.2 5,-7.1 4.2 5,-7.1 4.3 5,-7 4.2 5)))
```

Figure



This 3D example does not mix dimensions.

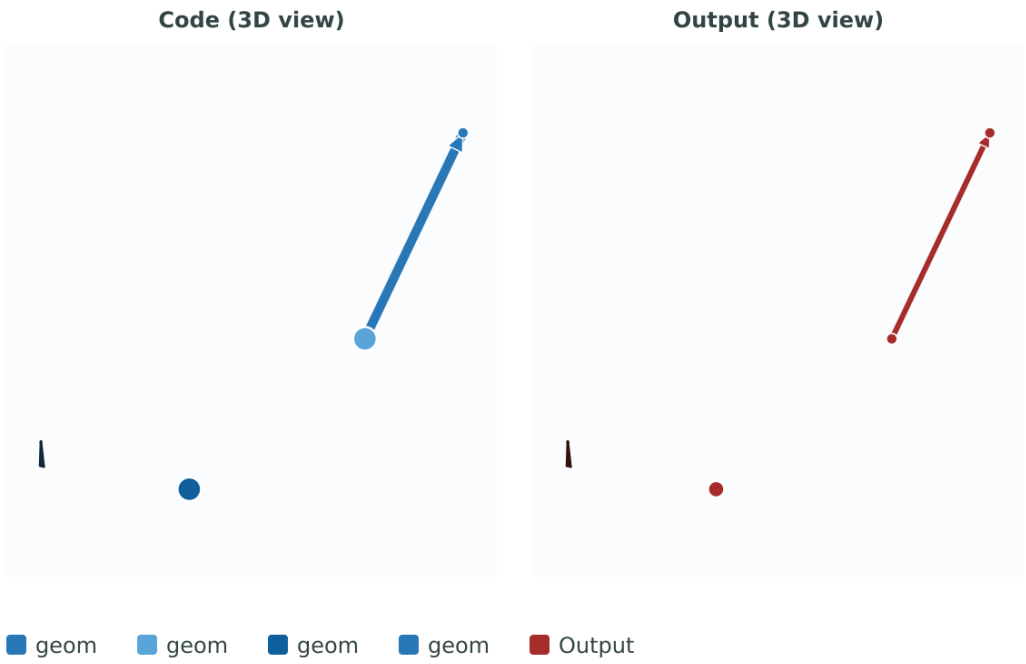
Code

```
SELECT ST_Union(geom)
FROM (
    SELECT 'POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2,-7 4.2 2))'::geometry geom
    UNION ALL
    SELECT 'POINT(5 5 5)'::geometry geom
    UNION ALL
    SELECT 'POINT(-2 3 1)'::geometry geom
    UNION ALL
    SELECT 'LINESTRING(5 5 5,10 10 10)'::geometry geom
) AS foo;
```

Output

```
GEOMETRYCOLLECTION(POINT(-2 3 1), LINESTRING(5 5 5,10 10 10), POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2,-7 4.2 2)))
```

Figure



This example unions an array built from a query.

Code

```
SELECT ST_Union(ARRAY(SELECT geom FROM sometable));
```

This example unions an array of linestrings.

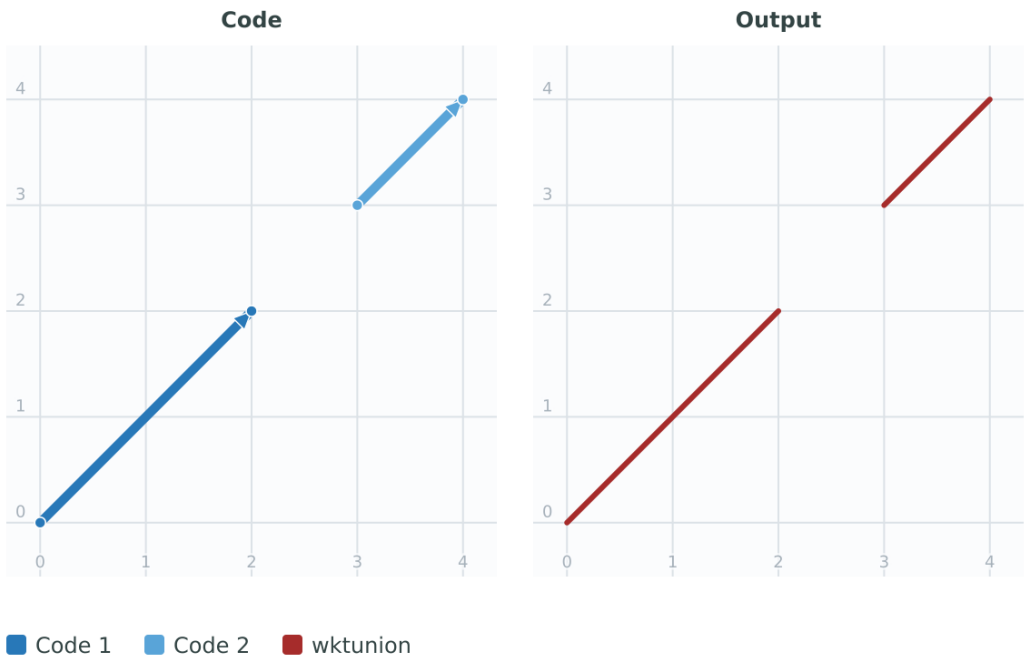
Code

```
SELECT ST_Union(ARRAY[
    'LINESTRING(1 2,3 4)'::geometry,
    'LINESTRING(4 5,5 6)'::geometry
]) As wktunion;
```

Output

```
MULTILINESTRING((1 2,3 4),(4 5,5 6))
```

Figure



Se även

ST_CoverageUnion, ST_UnaryUnion, ST_MemUnion, ST_Collect, ST_Intersection, ST_Difference, ST_SymDi
ST_LineMerge

7.14 Geometribearbetning

7.14.1 ST_Buffer

ST_Buffer — Beräknar en geometri som täcker alla punkter inom ett givet avstånd från en geometri.

Synopsis

geometry **ST_Buffer**(geometry g1, float radius_of_buffer, text buffer_style_parameters = ”);
geometry **ST_Buffer**(geometry g1, float radius_of_buffer, integer num_seg_quarter_circle);
geography **ST_Buffer**(geography g1, float radius_of_buffer, text buffer_style_parameters);
geography **ST_Buffer**(geography g1, float radius_of_buffer, integer num_seg_quarter_circle);

Beskrivning

Beräknar en POLYGON eller MULTIPOLYGON som representerar alla punkter vars avstånd från en geometri/geografi är mindre än eller lika med ett givet avstånd. Ett negativt avstånd krymper geometrin i stället för att expandera den. Ett negativt avstånd kan krympa en polygon helt och hållet, i vilket fall POLYGON EMPTY returneras. För punkter och linjer returnerar negativa avstånd alltid tomma resultat.

För geometri anges avståndet i enheterna för geometrins spatiala referenssystem. För geografi anges avståndet i meter.

Den valfria tredje parametern styr buffertens noggrannhet och stil. Noggrannheten för cirkelbågar i bufferten anges som antalet linjesegment som används för att approximera en kvartscirkel (standard är 8). Buffertstilen kan specificeras genom att tillhandahålla en lista med blankseparerade nyckel=värde-par enligt följande:

- 'quad_segs=#' : antal linjesegment som används för att approximera en kvartscirkel (standard är 8).
- 'endcap=round|flat|square' : endcap-stil (standard är round"). buttäccepteras som synonym för fflat".
- 'join=round|mitre|bevel' : fogningsstil (standard är round"). Miteräccepteras som synonym till mmitre".
- 'mitre_limit=#.#' : gräns för mitraförhållande (påverkar endast miterad fogning). 'miter_limit' accepteras som en synonym till 'mitre_limit'.
- 'side=both|left|right' : defaults to 'both'. 'left' or 'right' performs a single-sided buffer on the geometry, with the buffered side relative to the direction of the line. This is only applicable to LINESTRING geometry and does not affect POINT or POLYGON geometries. By default end caps are square when 'left' or 'right' are specified.

Notera

För geografin är detta ett tunt hölje runt geometriimplementeringen. Den bestämmer ett planar spatialt referenssystem som bäst passar in i det geografiska objektets avgränsningsbox (försöker UTM, Lambert Azimuthal Equal Area (LAEA) North/South pole, och slutligen Mercator). Bufferten beräknas i det plana utrymmet och transformeras sedan tillbaka till WGS84. Detta kanske inte ger önskat resultat om inmatningsobjektet är mycket större än en UTM-zon eller korsar datalinjen

---

Notera

Buffer kan hantera ogiltiga indata och utdata är alltid en giltig polygonal geometri. Buffring med avstånd 0 används ibland som ett sätt att reparera ogiltiga polygoner. ST_MakeValid är mer lämplig för denna process eftersom den kan hantera multipolygoner.

Notera

Buffring används ibland för att utföra en sökning inom avstånd. För detta användningsfall är det mer effektivt att använda ST_DWithin.

Notera

Denna funktion ignorerar Z-dimensionen. Den ger alltid ett 2D-resultat även när den används på en 3D-geometri.

Förbättrad: 2.5.0 - ST_Buffer geometri stöd förbättrades för att möjliggöra sidobuffring specifikation `side=both|left|right`.

Tillgänglighet: 1.5 - ST_Buffer har förbättrats för att stödja olika ändkapslar och join-typer. Dessa är användbara för att t.ex. konvertera väglinjer till polygonvägar med platta eller fyrkantiga kanter istället för rundade kanter. Tunt omslag för geografi har lagts till.

Utförs av GEOS-modulen.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1.30

Exempel

The textual results are rounded to whole coordinate units for readability. Each result remains available as text, while the manual initially favors the shared input/output figure.

quad_segs=8 (standard)

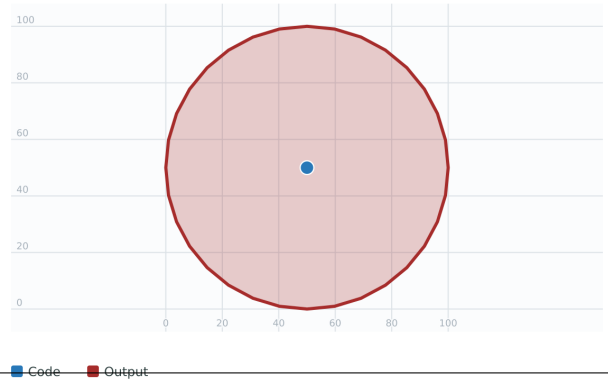
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('POINT(100 90)'),
  50, 'quad_segs=8');
```

Output

```
POLYGON((150 90,149 80,146 71,142 62,135 55,128 48,119
44,110 41,100 40,90 41,81 44,72 48,65 55,58 62,54 71,51
80,50 90,51 100,54 109,58 118,65 125,72 132,81 136,90
139,100 140,110 139,119 136,128 132,135 125,142 118,146
109,149 100,150 90))
```

Figure



quad_segs=2

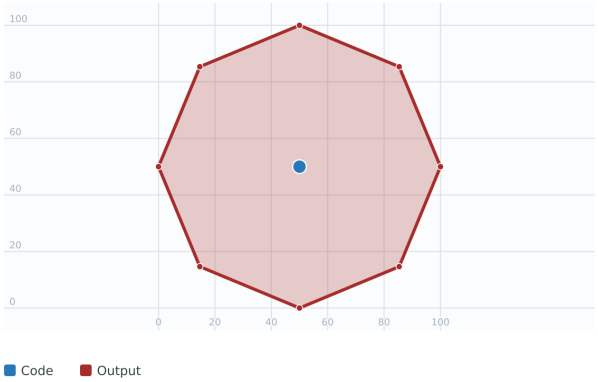
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('POINT(100 90)'),
  50, 'quad_segs=2');
```

Output

```
POLYGON((150 90,135 55,100 40,65 55,50 90,65 125,100
140,135 125,150 90))
```

Figure



endcap=rund join=rund (standard)

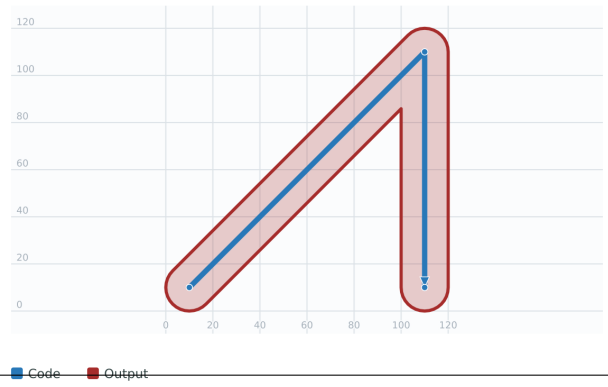
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'endcap=round join=round');
```

Output

```
POLYGON((143 157,144 158,146 159,148 160,150 160,152
160,154 159,156 158,157 157,158 156,159 154,160 152,160
150,160 50,160 48,159 46,158 44,157 43,156 42,154 41,152
40,150 40,148 40,146 41,144 42,143 43,142 44,141 46,140
48,140 50,140 126,57 43,56 42,54 41,52 40,50 40,48 40,46
41,44 42,43 43,42 44,41 46,40 48,40 50,40 52,41 54,42
56,43 57,143 157))
```

Figure



ändkapsel=fyrkantsformad

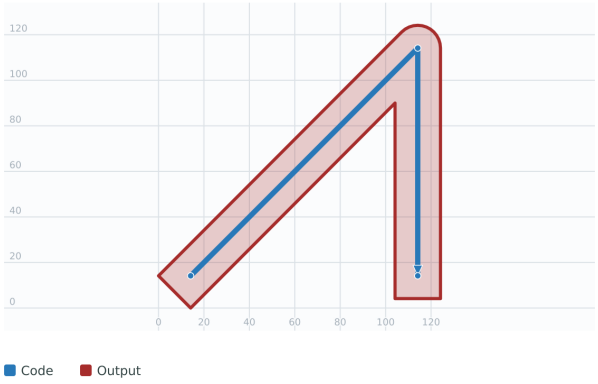
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'endcap=square join=round');
```

Output

```
POLYGON((143 157,144 158,146 159,148 160,150 160,152
160,154 159,156 158,157 157,158 156,159 154,160 152,160
150,160 50,160 40,140 40,140 126,57 43,50 36,36 50,143
157))
```

Figure



ändkapsel=platt

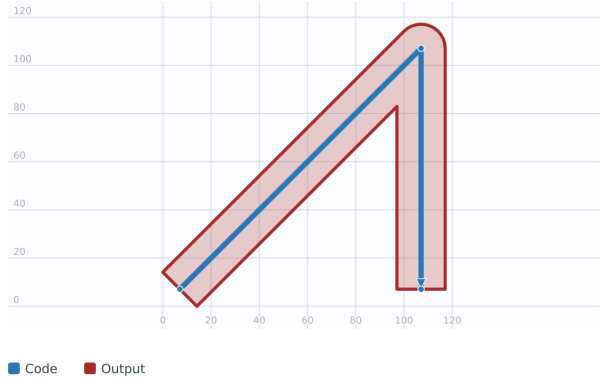
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'endcap=flat join=round');
```

Output

```
POLYGON((143 157,144 158,146 159,148 160,150 160,152
160,154 159,156 158,157 157,158 156,159 154,160 152,160
150,160 50,140 50,140 126,57 43,43 57,143 157))
```

Figure



skarv=fasad

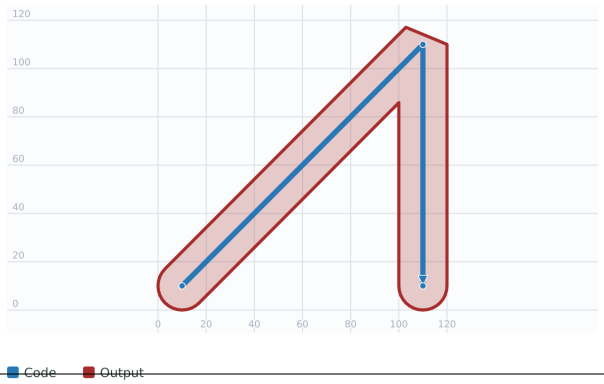
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'join=bevel');
```

Output

```
POLYGON((143 157,160 150,160 50,160 48,159 46,158 44,157
43,156 42,154 41,152 40,150 40,148 40,146 41,144 42,143
43,142 44,141 46,140 48,140 50,140 126,57 43,56 42,54
41,52 40,50 40,48 40,46 41,44 42,43 43,42 44,41 46,40
48,40 50,40 52,41 54,42 56,43 57,143 157))
```

Figure



join=mitre mitre_limit=5.0 (standard mitre-gräns)

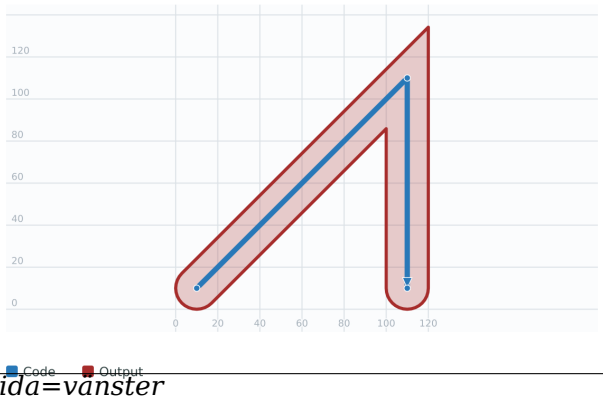
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'join=mitre mitre_limit=5.0');
```

Output

```
POLYGON((160 174,160 50,160 48,159 46,158 44,157 43,156
42,154 41,152 40,150 40,148 40,146 41,144 42,143 43,142
44,141 46,140 48,140 50,140 126,57 43,56 42,54 41,52 40,
50 40,48 40,46 41,44 42,43 43,42 44,41 46,40 48,40 50,40
52,41 54,42 56,43 57,160 174))
```

Figure



skarv=mitra mitre_limit=1

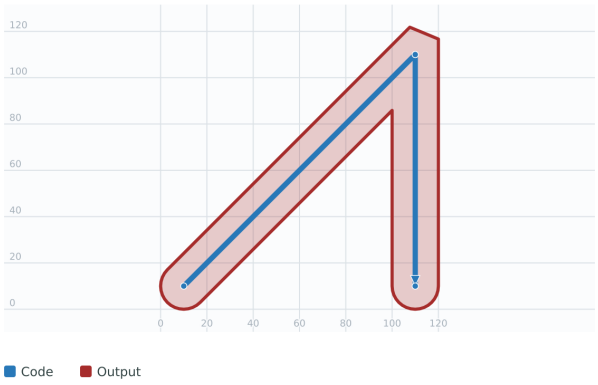
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'join=mitre mitre_limit=1.0');
```

Output

```
POLYGON((148 162,160 157,160 50,160 48,159 46,158 44,157
43,156 42,154 41,152 40,150 40,148 40,146 41,144 42,143
43,142 44,141 46,140 48,140 50,140 126,57 43,56 42,54
41,52 40,50 40,48 40,46 41,44 42,43 43,42 44,41 46,40
48,40 50,40 52,41 54,42 56,43 57,148 162))
```

Figure



sida=vänster

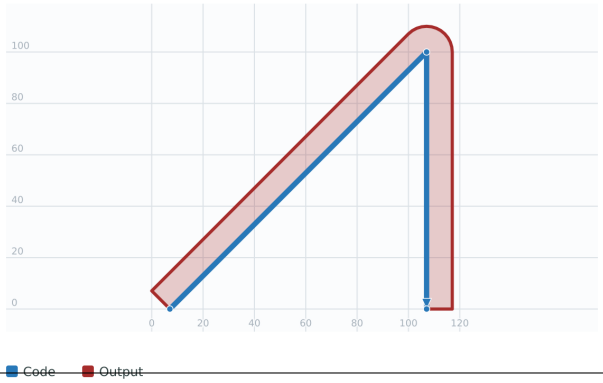
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'side=left');
```

Output

```
POLYGON((150 50,150 150,50 50,43 57,143 157,144 158,146
159,148 160,150 160,152 160,154 159,156 158,157 157,158
156,159 154,160 152,160 150,160 50,150 50))
```

Figure



sida=höger

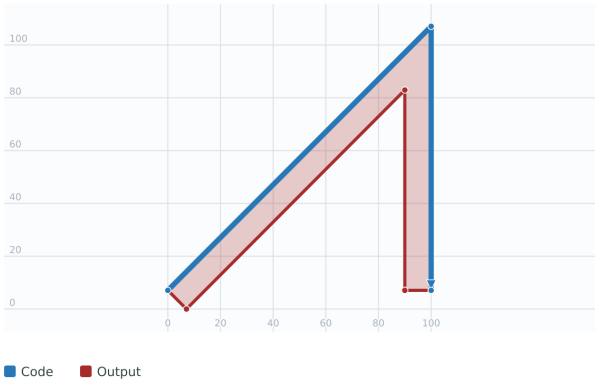
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'side=right');
```

Output

```
POLYGON((50 50,150 150,150 50,140 50,140 126,57 43,50
50))
```

Figure



sida=vänster led=mitre

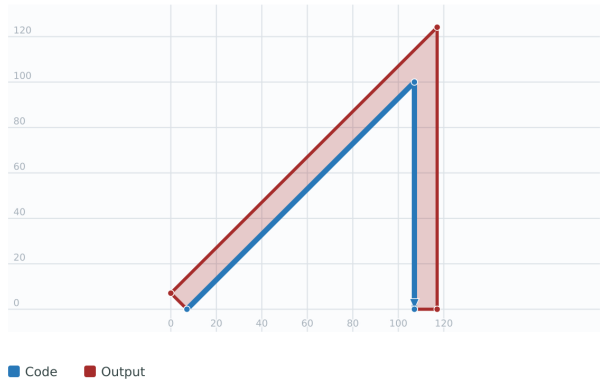
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
    150,150 50)'),
  10, 'side=left join=mitre');
```

Output

```
POLYGON((150 50,150 150,50 50,43 57,160 174,160 50,150
50))
```

Figure



right-hand winding, polygon boundary side=left

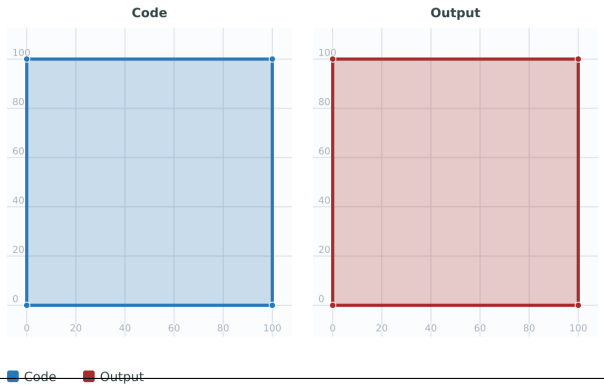
Code

```
SELECT ST_Buffer(
  ST_ForceRHR(ST_Boundary(ST_GeomFromText ↩
    (
     'POLYGON ((50 50,50 150,150 150,150 ↩
    50,50 50))'))),
  20, 'side=left');
```

Output

```
POLYGON((50 50,50 150,150 150,150 50,50 50))
```

Figure



right-hand winding, polygon boundary side=right

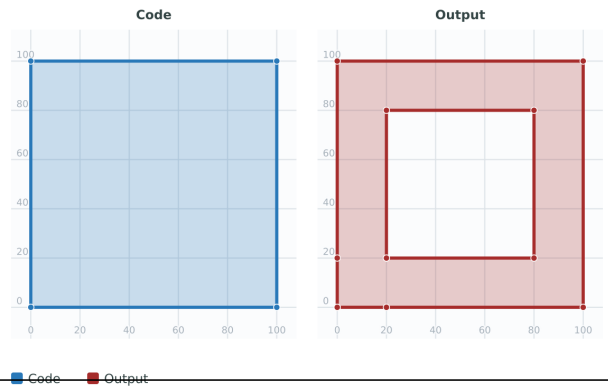
Code

```
SELECT ST_Buffer(
  ST_ForceRHR(ST_Boundary(ST_GeomFromText ↩
    (
     'POLYGON ((50 50,50 150,150 150,150 ↩
    50,50 50))'))),
  20, 'side=right');
```

Output

```
POLYGON((50 50,50 70,50 150,150 150,150 50,70 50,50 50),
(70 70,130 70,130 130,70 130,70 70))
```

Figure



A buffered point approximates a circle. A buffered point forcing approximation of (see diagram) 2 points per quarter circle is poly with 8 sides (see diagram)

Code

```
SELECT ST_NPoints(ST_Buffer('POINT(100 90)'::geometry, 50)) As promisingcircle_pcount,
ST_NPoints(ST_Buffer('POINT(100 90)'::geometry, 50, 2)) As lamecircle_pcount;
```

Output

```
promisingcircle_pcount | lamecircle_pcount
-----------------------+-------------------
                    33 |                 9
```

A lighter, less smooth circle using only 2 points per quarter circle is an octagon.

The following example creates a 100 meter octagon in NAD83 longitude/latitude by transforming to the Massachusetts state plane meter projection and then buffering. The figure shows the input and output in separate panels because the coordinates are in different SRIDs.

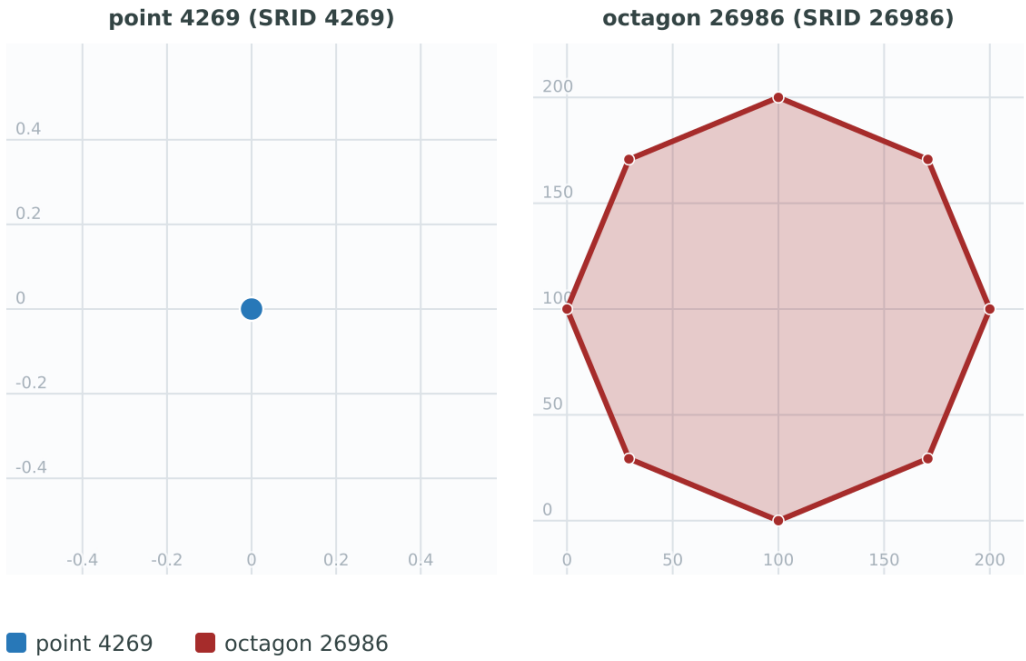
Code

```
WITH input AS (
  SELECT ST_SetSRID(ST_Point(-71.063526, 42.35785), 4269) AS point_4269
), projected AS (
  SELECT point_4269,
         ST_Transform(point_4269, 26986) AS point_26986
  FROM input
)
SELECT point_4269 AS input_point_4269,
       ST_Buffer(point_26986, 100, 2) AS octagon_26986
FROM projected;
```

Output

```
SRID=4269;POINT(-71.063526 42.35785) | SRID=26986;POLYGON((236057.5905746494 900908.7599186979,236028.30125276805 ↩
    900838.0492405792,235957.5905746494 900808.7599186979,235886.87989653074 900838.0492405792,235857.5905746494 ↩
    900908.7599186979,235886.87989653074 900979.4705968165,235957.5905746494 901008.7599186979,236028.30125276805 ↩
    900979.4705968165,236057.5905746494 900908.7599186979))
```

Figure



Buffering a polygon by a positive distance and then by the same negative distance is a morphological closing operation. It can be useful for smoothing noisy boundaries, such as simplifying a coastline by removing narrow inlets and bays. The result intentionally changes the shape and area of the input, and the distance must be chosen in the units of the input SRS. For data in a geographic SRS, transform to a projected SRS suited to the operation first; for cartographic visualization in Web Mercator, using that display SRS can be appropriate.

Smooth a coastline using a 1 km closing operation, then simplify the result. The example uses Massachusetts State Plane meter units for the operation.

Code

```
WITH projected AS (
  SELECT id, ST_Transform(geom, 26986) AS geom
  FROM coast
),
closed AS (
  SELECT id, ST_Buffer(ST_Buffer(geom, 1000), -1000) AS geom
  FROM projected
)
SELECT id, ST_Transform(ST_SimplifyPreserveTopology(geom, 50), 4326) AS geom
FROM closed;
```

The same technique is described in Paul Ramsey's Removing Complexities. The final simplification step is optional; use ST_SimplifyPreserveTopology when polygonal validity matters, and check the result against the cartographic or analytical tolerance required by the application.

A small negative buffer can also be used as a heuristic to find narrow polygon spikes. The query below compares how much area and perimeter remain after erosion. A large `erosion_index` indicates that the perimeter dropped proportionally more than the area. This is only a preselection aid; the threshold and buffer distance must be chosen for the data units and the kind of spike being checked.

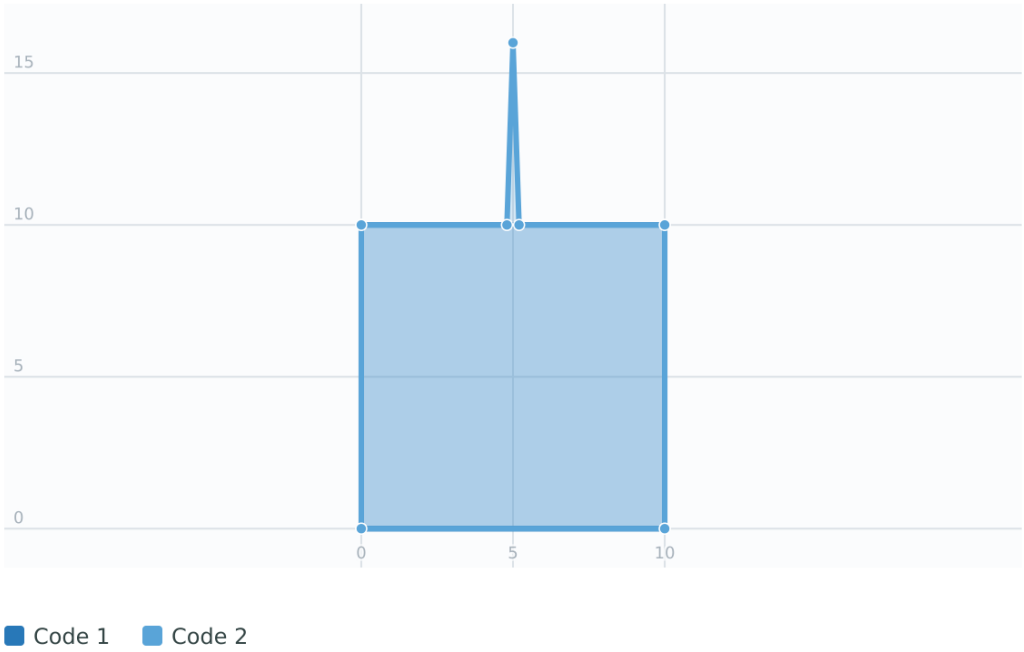
Code

```
WITH sample(id, geom) AS (
  VALUES
    ('plain', 'POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry),
    ('spike', 'POLYGON((0 0,10 0,10 10,5.2 10,5 16,4.8 10,0 10,0 0))'::geometry)
),
eroded AS (
  SELECT id, geom, ST_Buffer(geom, -0.5) AS eroded_geom
  FROM sample
),
scored AS (
  SELECT id,
         round((
           ST_Area(eroded_geom) / ST_Area(geom) /
           NULLIF(ST_Perimeter(eroded_geom) / ST_Perimeter(geom), 0)
         )::numeric, 3) AS erosion_index
  FROM eroded
  WHERE NOT ST_IsEmpty(eroded_geom)
)
SELECT id, erosion_index, erosion_index > 1.1 AS possible_spike
FROM scored
ORDER BY id;
```

Output

```
id    | erosion_index | possible_spike
-------+---------------+----------------
 plain |         0.900 | f
 spike |         1.147 | t
```

Figure



A variable-distance buffer can be approximated by buffering each vertex of a line by a distance interpolated along the line, building the convex hull for each consecutive pair of vertex buffers, and unioning the segment hulls. As with any geometry buffer, distances are in the units of the input SRS.

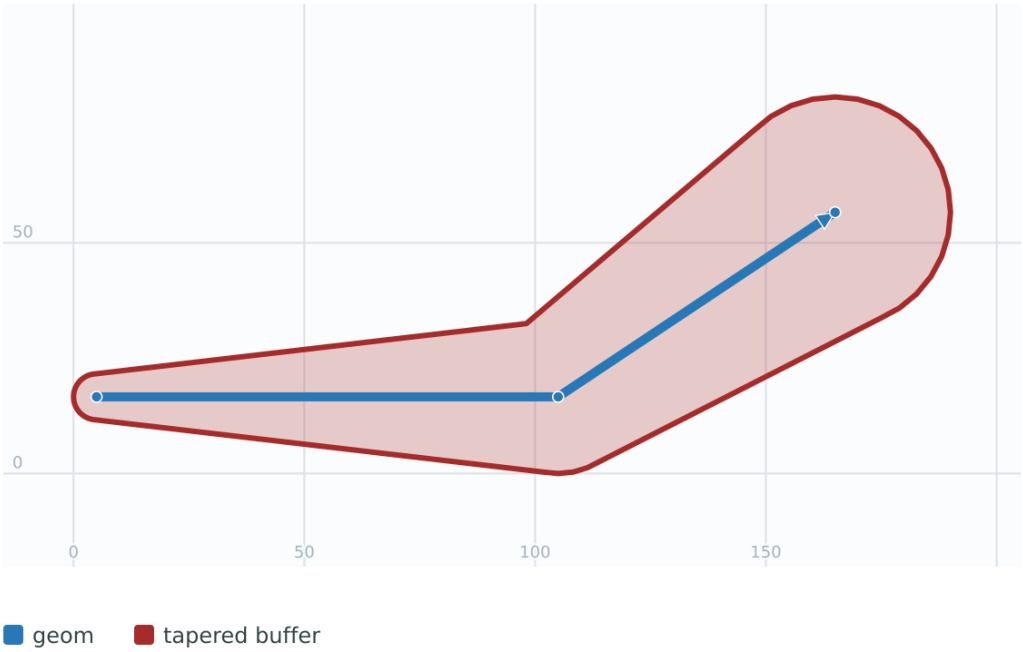
Taper a line buffer from 5 to 25 units along the line.

Code

```
WITH line AS (
  SELECT 'LINESTRING(0 0, 100 0, 160 40)'::geometry AS geom
),
vertices AS (
  SELECT (dump).path[1] AS n,
         (dump).geom AS geom,
         line.geom AS line_geom
  FROM line
  CROSS JOIN LATERAL ST_DumpPoints(line.geom) AS dump
),
radii AS (
  SELECT n,
         geom,
         5 + (25 - 5) * ST_LineLocatePoint(line_geom, geom) AS radius
  FROM vertices
),
segments AS (
  SELECT n,
         geom,
         radius,
         lead(geom) OVER (ORDER BY n) AS next_geom,
         lead(radius) OVER (ORDER BY n) AS next_radius
  FROM radii
)
SELECT ST_Union(
         ST_ConvexHull(
           ST_Collect(
             ST_Buffer(geom, radius),
             ST_Buffer(next_geom, next_radius)
```

```
          )
        )
      ) AS tapered_buffer
FROM segments
WHERE next_geom IS NOT NULL;
```

Figure



Se även

ST_Collect, ST_ConvexHull, ST_DWithin, ST_DumpPoints, ST_LineLocatePoint, ST_SetSRID, ST_SimplifyPre
ST_Transform, ST_Union, ST_MakeValid

7.14.2 ST_BuildArea

ST_BuildArea — Skapar en polygonal geometri som bildas av linjerna i en geometri.

Synopsis

geometry **ST_BuildArea**(geometry geom);

Beskrivning

Skapar en ytgeometri som består av de ingående linjerna i indatageometrin. Indata kan vara en LineString, MultiLineString, Polygon, MultiPolygon eller en GeometryCollection. Resultatet är en polygon eller multipolygon, beroende på indata. Om indatalinjerna inte bildar polygoner returneras NULL.

Till skillnad från ST_MakePolygon accepterar denna funktion ringar som bildas av flera linjer och kan bilda valfritt antal polygoner.

Denna funktion omvandlar inre ringar till hål. Om du även vill omvandla inre ringar till polygoner använder du ST_Polygonize.

Notera
Ingångslinjen måste vara korrekt nodad för att denna funktion ska fungera korrekt. ST_Node kan användas för att noda linjer.
Om inmatningslinjerna korsar varandra kommer denna funktion att producera ogiltiga polygoner. ST_MakeValid kan användas för att säkerställa att utdata är giltiga.

Tillgänglighet: 1.1.0

Exempel

Building an area from five input lines.

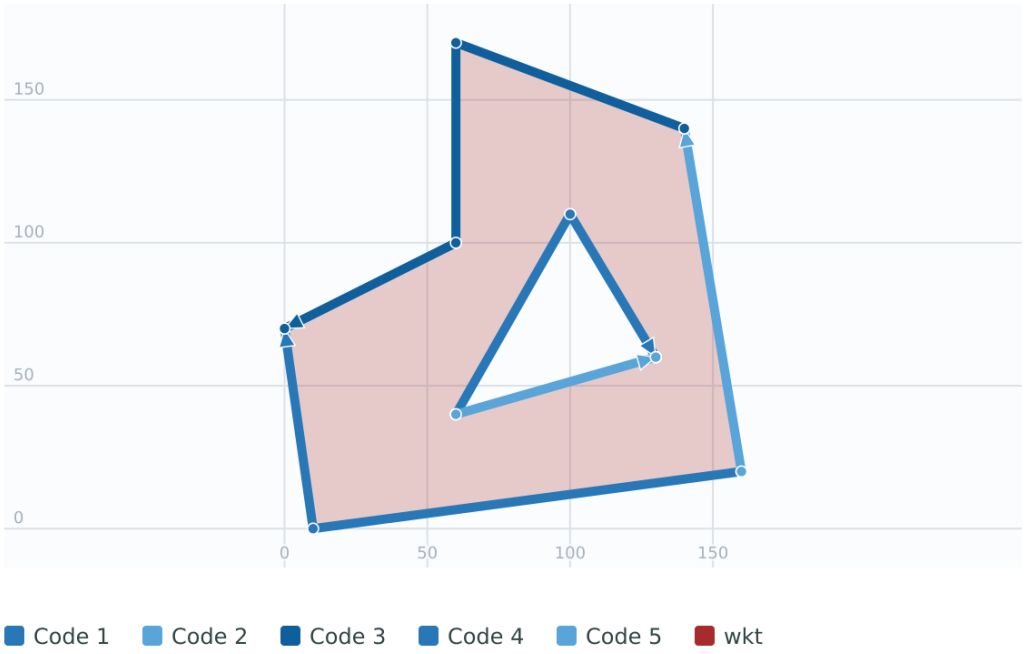
Code

```
WITH data(geom) AS (VALUES
   ('LINESTRING (180 40,30 20,20 90)'::geometry),
  ('LINESTRING (180 40,160 160)'::geometry),
  ('LINESTRING (160 160,80 190,80 120,20 90)'::geometry),
  ('LINESTRING (80 60,120 130,150 80)'::geometry),
  ('LINESTRING (80 60,150 80)'::geometry)
)
SELECT ST_BuildArea(ST_Collect(geom )) AS wkt
    FROM data;
```

Output

```
POLYGON((180 40,30 20,20 90,80 120,80 190,160 160,180 40),(150 80,120 130,80 60,150 80))
```

Figure

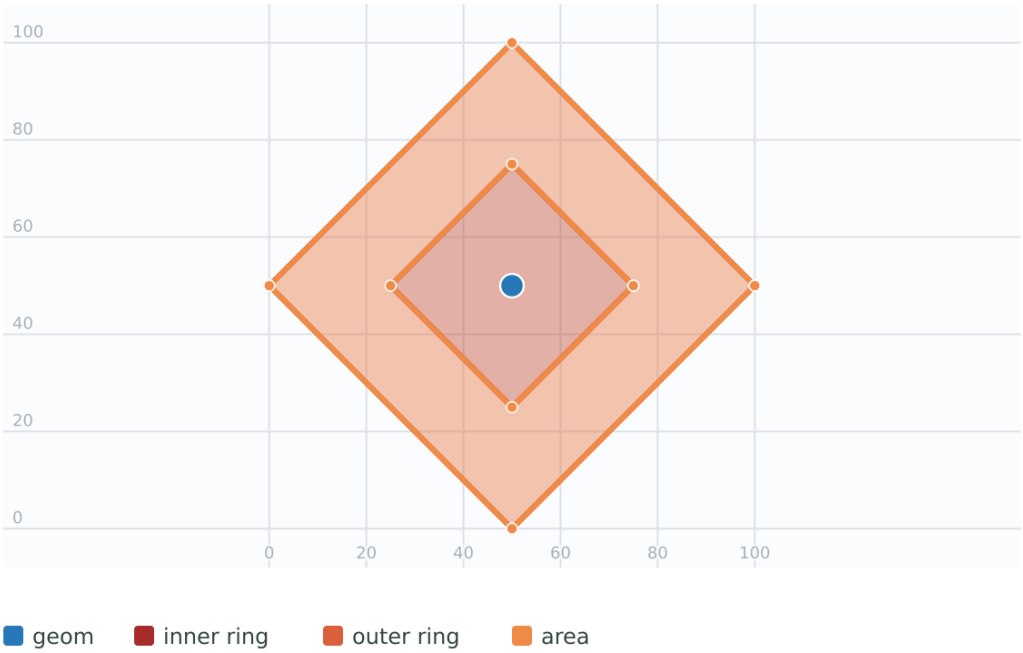


Create a donut from two circular polygons.

Code

```
WITH input AS (
  SELECT 'POINT(100 90)'::geometry AS geom
),
rings AS (
  SELECT ST_Buffer(geom, 25, 'quad_segs=1') AS inring,
         ST_Buffer(geom, 50, 'quad_segs=1') AS outring
  FROM input
)
SELECT inring AS inner_ring,
       outring AS outer_ring,
       ST_BuildArea(ST_Collect(inring, outring)) AS area
FROM rings;
```

Figure



Input linework that crosses at non-endpoint intersections must be noded before building areas:

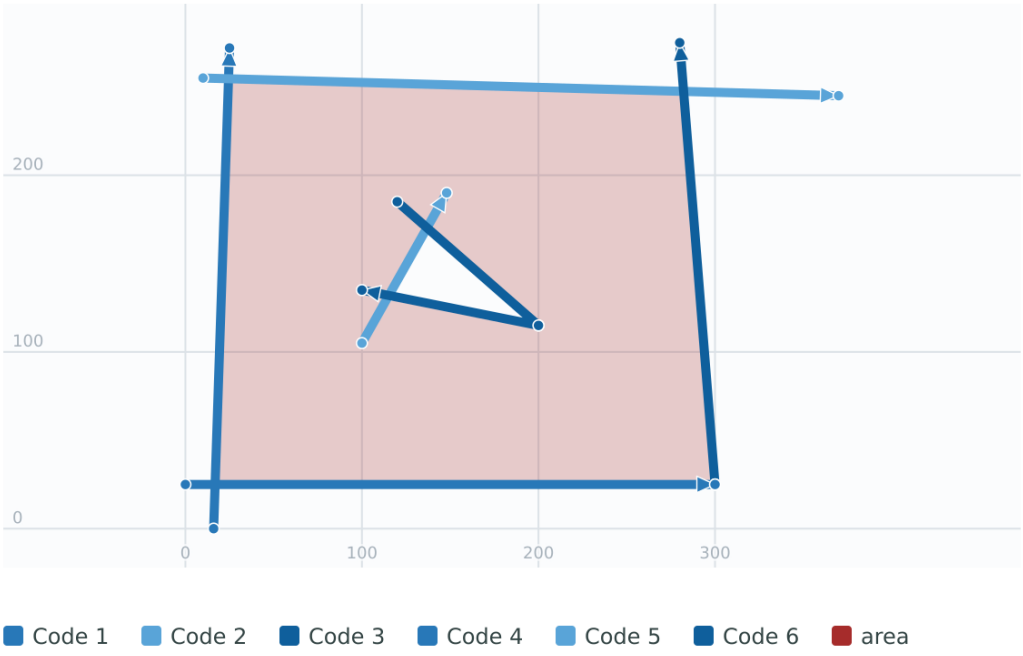
Code

```
WITH lines(geom) AS (VALUES
  ('LINESTRING(36 35,45 307)'::geometry),
  ('LINESTRING(30 290,390 280)'::geometry),
  ('LINESTRING(320 60,300 310)'::geometry),
  ('LINESTRING(20 60,320 60)'::geometry),
  ('LINESTRING(120 140,168 225)'::geometry),
  ('LINESTRING(140 220,220 150,120 170)'::geometry)
),
noded AS (
  SELECT ST_Node(ST_UnaryUnion(ST_Collect(geom))) AS geom FROM lines
)
SELECT ST_BuildArea(geom) AS area
FROM noded;
```

Output

```
POLYGON((320 60,36.8 60,44.4 289.6,302.2 282.4,320 60),(156.9 205.3,135.2 167,220 150,156.9 205.3))
```

Figure



Se även

ST_Collect, ST_MakePolygon, ST_MakeValid, ST_Node, ST_Polygonize, ST_BdPolyFromText, ST_BdMPolyFr
(wrappers till denna funktion med standard OGC-gränssnitt)

7.14.3 ST_Centroid

ST_Centroid — Returnerar den geometriska mittpunkten för en geometri.

Synopsis

geometry **ST_Centroid**(geometry g1);
geography **ST_Centroid**(geography g1, boolean use_spheroid = true);

Beskrivning

Beräknar en punkt som är den geometriska tyngdpunkten för en geometri. För`[MULTI]POINT`sär centroiden det aritmetiska medelvärdet av de inmatade koordinaterna. För `[MULTI]LINESTRING`sberäknas centroiden med hjälp av den viktade längden för varje linjesegment. För `[MULTI]POLYGONER`beräknas centroiden i termer av area. Om en tom geometri anges, returneras en tom `GEOMETRYCOLLECTION`. Om `NULL` anges, returneras `NULL`. Om `CIRCULARSTRING` eller `COMPOUNDCURVE` anges konverteras de till linestring med CurveToLine först, sedan på samma sätt som för `LINESTRING`

För indata med blandade dimensioner är resultatet lika med centroiden för komponentgeometrierna med den högsta dimensionen (eftersom geometrierna med lägre dimension bidrar med noll ”vikttill centroiden).

Observera att för polygonala geometrier behöver centroiden inte nödvändigtvis ligga i polygonens inre. Se t.ex. diagrammet nedan över centroiden i en C-formad polygon. För att konstruera en punkt som garanterat ligger i det inre av en polygon används ST_PointOnSurface.

Nytt i 2.3.0 : stödjer `CIRCULARSTRING` och `COMPOUNDCURVE` (med CurveToLine)

Tillgänglighet: 2.4.0 stöd för geografi infördes.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.1.4, 9.5.5

Exempel

Centroid of a MultiPoint.

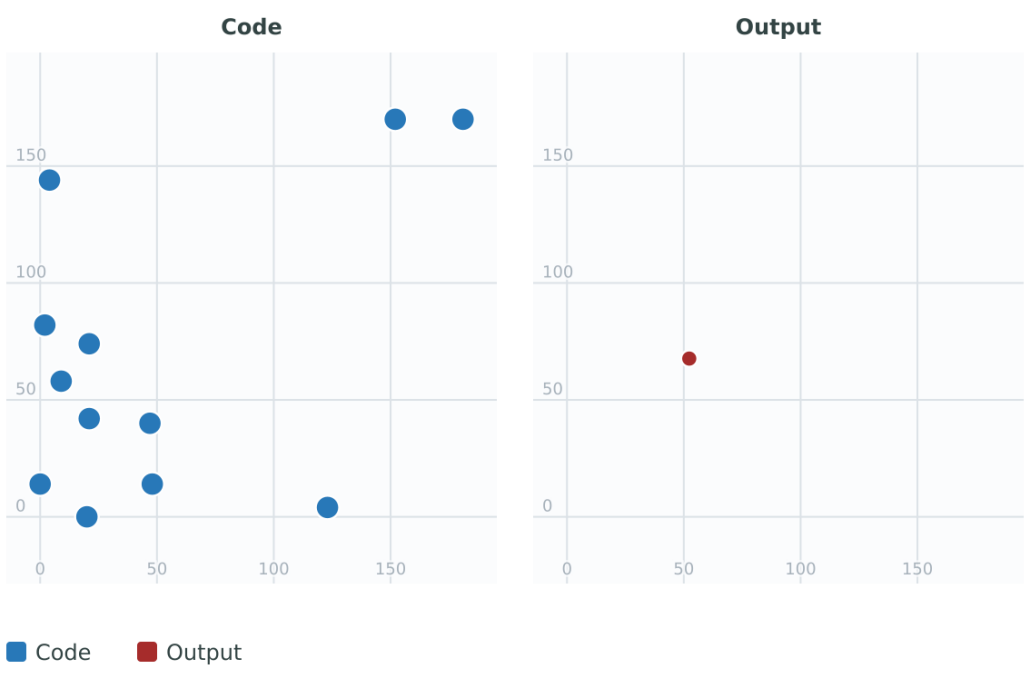
Code

```
SELECT ST_Centroid(
  'MULTIPOINT (8 24,10 92,12 154,17 68,28 10,29 52,29 84,55 50,56 24,131 14,160 180,189 ↩
   180)');
```

Output

```
POINT(60.333 77.667)
```

Figure



Centroid of a LineString.

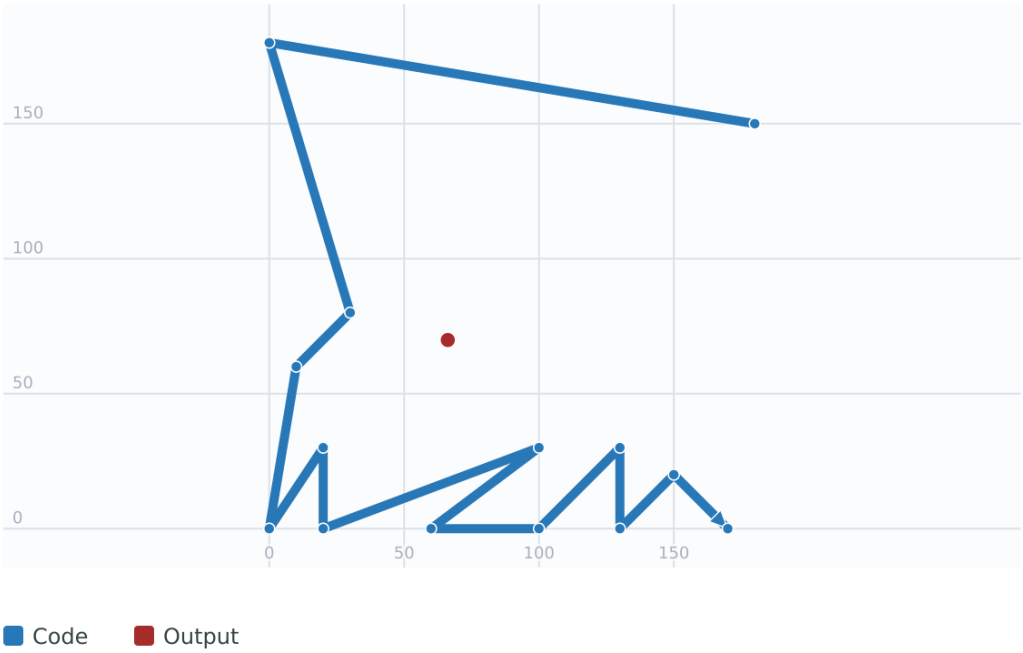
Code

```
SELECT ST_Centroid(
  'LINESTRING (190 160,10 190,40 90,20 70,10 10,30 40,30 10,110 40,70 10,110 10,140 40,140 ↩
   10,160 30,180 10)');
```

Output

```
POINT(76.191 79.876)
```

Figure



Centroid of a Polygon.

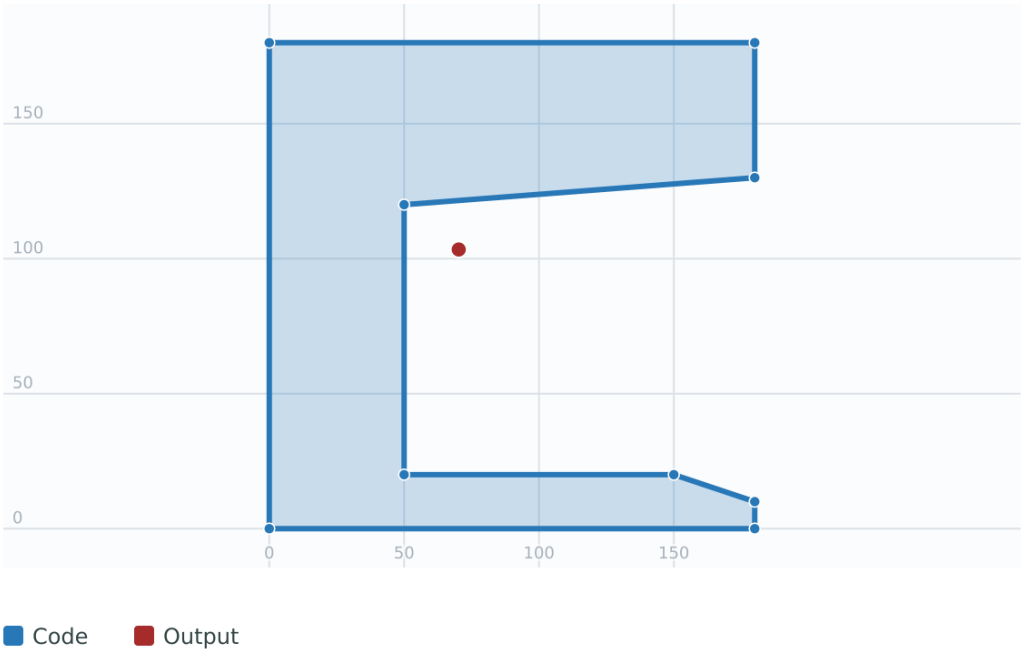
Code

```
SELECT ST_Centroid(
  'POLYGON ((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))');
```

Output

```
POINT(80.251 113.405)
```

Figure



Centroid of the highest-dimension components of a GeometryCollection.

Code

```
SELECT ST_Centroid(
  'GEOMETRYCOLLECTION (POLYGON ((190 170,180 100,80 140,80 160,130 160,110 180,110 190,180 ↩
   180,190 170)),
    LINESTRING (80 120,120 20,140 70,150 30,180 50,190 10),
    MULTIPOINT (19 150,22 49,30 13,32 101,45 35,67 88,75 16))');
```

Output

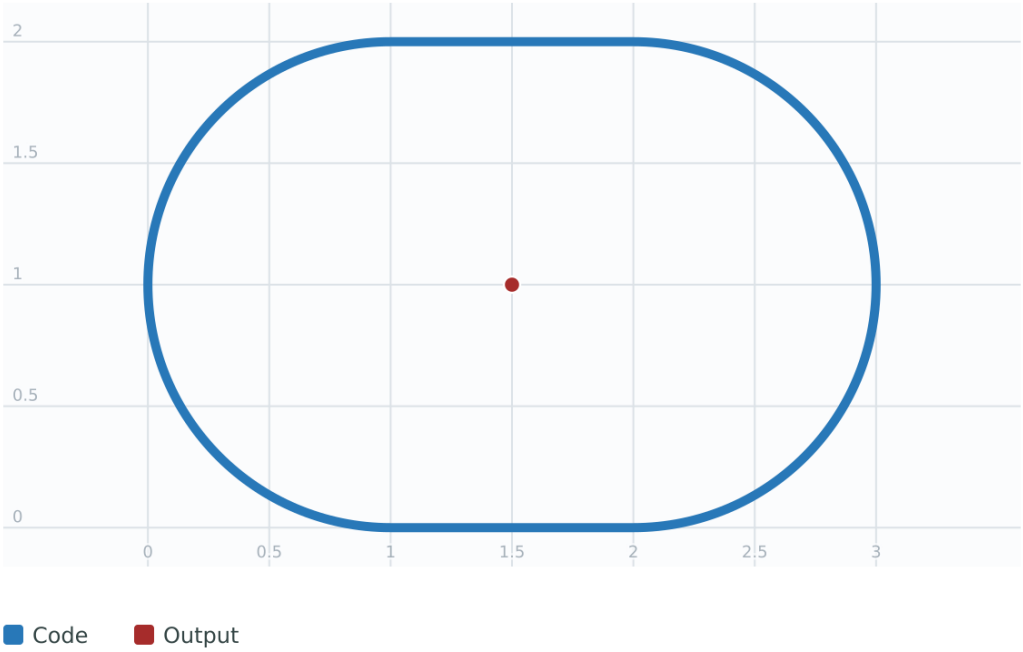
```
POINT(143.361 148.263)
```

Figure

Code

```
SELECT ST_Centroid(g)
FROM  ST_GeomFromText('CIRCULARSTRING(0 2,-1 1,0 0,0.5 0,1 0,2 1,1 2,0.5 2,0 2)')  AS g ;
```

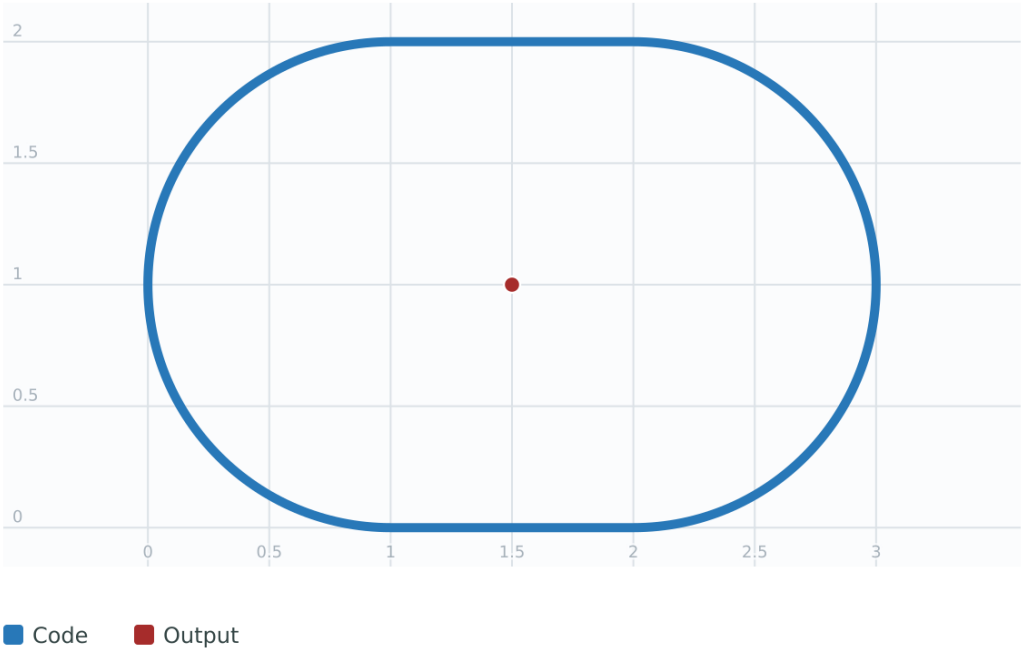
Figure



Code

```
SELECT ST_Centroid(g)
FROM  ST_GeomFromText('COMPOUNDCURVE(CIRCULARSTRING(0 2,-1 1,0 0),(0 0,0.5 0,1 0),  ↩
    CIRCULARSTRING( 1 0,2 1,1 2),(1 2,0.5 2,0 2))' ) AS g;
```

Figure



Centroids of the parts of a MultiPolygon.

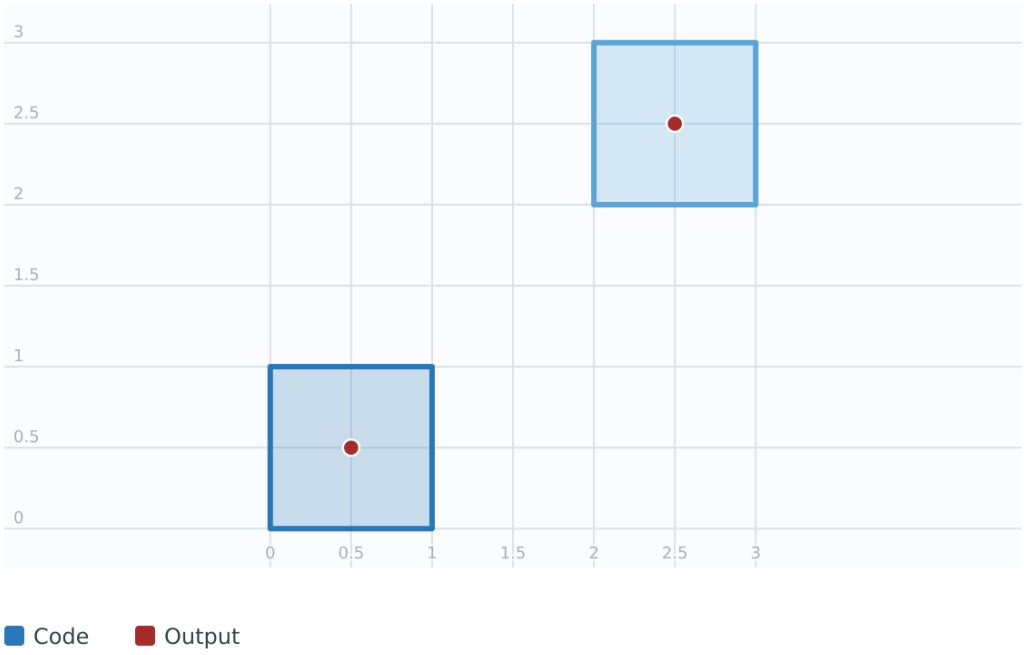
Code

```
SELECT ST_Collect(ST_Centroid(geom) ORDER BY path)
FROM ST_Dump('MULTIPOLYGON (
    ((0 0,0 1,1 1,1 0,0 0)),
    ((2 2,2 3,3 3,3 2,2 2))
  )'::geometry);
```

Output

```
MULTIPOINT((0.5 0.5),(2.5 2.5))
```

Figure



Se även

ST_Collect, ST_Dump, ST_PointOnSurface, ST_GeometricMedian

7.14.4 ST_ChaikinSmoothing

ST_ChaikinSmoothing — Returnerar en utjämnad version av en geometri med hjälp av Chaikin-algoritmen

Synopsis

geometry **ST_ChaikinSmoothing**(geometry geom, integer nIterations = 1, boolean preserveEndPoints = false);

Beskrivning

Utjämnar en linjär eller polygonal geometri med hjälp av Chaikins algoritm. Graden av utjämning styrs av parametern `nIterations`. Vid varje iteration ersätts varje inre toppunkt med två toppunkter som ligger på 1/4 av längden på linjesegmenten före och efter toppunkten. En rimlig grad av utjämning uppnås med 3 iterationer; den maximala graden är begränsad till 5.

Om `preserveEndPoints` är true utjämnas inte ändpunkterna för Polygon-ringar. Slutpunkterna för LineStrings bevaras alltid.

Notera
Antalet toppar fördubblas för varje iteration, så resultatgeometrin kan ha många fler punkter än indata. För att minska antalet punkter kan du använda en förenklingsfunktion på resultatet (se ST_Simplify, ST_SimplifyPreserveTopology och ST_SimplifyVW).

Resultatet har interpolerade värden för Z- och M-dimensionerna när sådana finns.

Denna funktion stöder 3d och kommer inte att tappa z-index.
Tillgänglighet: 2.5.0

Exempel

Smoothing a Polygon using 1, 2 and 3 iterations.

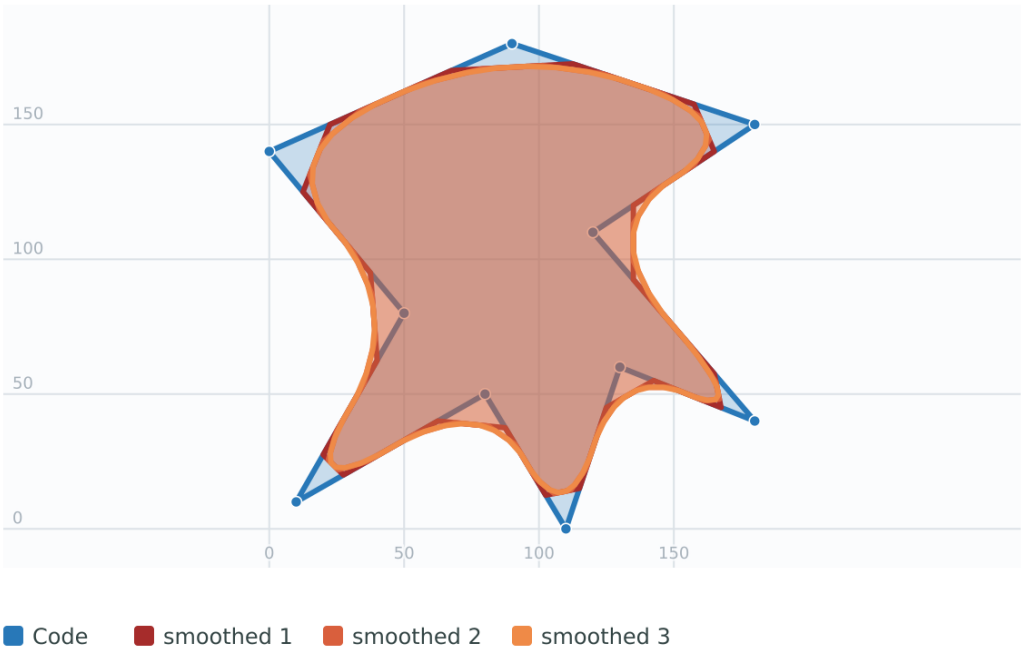
Code

```
SELECT n AS iterations, ST_ChaikinSmoothing(
    'POLYGON ((20 20,60 90,10 150,100 190,190 160,130 120,190 50,140 70,120 10,90 60,20 20)
    )', n
  ) AS smoothed
FROM generate_series(1, 3) AS n;
```

Output

```
1 | POLYGON((30 37.5,50 72.5,47.5 105,22.5 135,32.5 160,77.5 180,122.5 182.5,167.5 167.5,175 150,145 130,145 102.5,175
     67.5,177.5 55,152.5 65,135 55,125 25,112.5 22.5,97.5 47.5,72.5 50,37.5 30,30 37.5))
2 | POLYGON((35 46.25,45 63.75,49.375 80.625,48.125 96.875,41.25 112.5,28.75 127.5,25 141.25,30 153.75,43.75 165,66.25
     175,88.75 180.625,111.25 181.875,133.75 178.75,156.25 171.25,169.375 163.125,173.125 154.375,167.5 145,152.5
    135,145 123.125,145 109.375,152.5 93.75,167.5 76.25,175.625 64.375,176.875 58.125,171.25 57.5,158.75 62.5,148.125
     62.5,139.375 57.5,132.5 47.5,127.5 32.5,121.875 24.375,115.625 23.125,108.75 28.75,101.25 41.25,91.25
    48.125,78.75 49.375,63.75 45,46.25 35,35.625 31.875,31.875 35.625,35 46.25))
3 | POLYGON((37.5 50.625,42.5 59.375,46.09375 67.96875,48.28125 76.40625,49.0625 84.6875,48.4375 92.8125,46.40625
    100.78125,42.96875 108.59375,38.125 116.25,31.875 123.75,27.8125 130.9375,25.9375 137.8125,26.25 144.375,28.75
    150.625,33.4375 156.5625,40.3125 162.1875,49.375 167.5,60.625 172.5,71.875 176.40625,83.125 179.21875,94.375
    180.9375,105.625 181.5625,116.875 181.09375,128.125 179.53125,139.375 176.875,150.625 173.125,159.53125
    169.21875,166.09375 165.15625,170.3125 160.9375,172.1875 156.5625,171.71875 152.03125,168.90625 147.34375,163.75
    142.5,156.25 137.5,150.625 132.03125,146.875 126.09375,145 119.6875,145 112.8125,146.875 105.46875,150.625
    97.65625,156.25 89.375,163.75 80.625,169.53125 73.28125,173.59375 67.34375,175.9375 62.8125,176.5625
    59.6875,175.46875 57.96875,172.65625 57.65625,168.125 58.75,161.875 61.25,156.09375 62.5,150.78125 62.5,145.9375
    61.25,141.5625 58.75,137.65625 55,134.21875 50,131.25 43.75,128.75 36.25,126.09375 30.46875,123.28125
    26.40625,120.3125 24.0625,117.1875 23.4375,113.90625 24.53125,110.46875 27.34375,106.875 31.875,103.125
    38.125,98.75 42.96875,93.75 46.40625,88.125 48.4375,81.875 49.0625,75 48.28125,67.5 46.09375,59.375 42.5,50.625
    37.5,43.59375 34.21875,38.28125 32.65625,34.6875 32.8125,32.8125 34.6875,32.65625 38.28125,34.21875 43.59375,37.5
     50.625))
```

Figure



Smoothing a LineString using 1, 2 and 3 iterations.

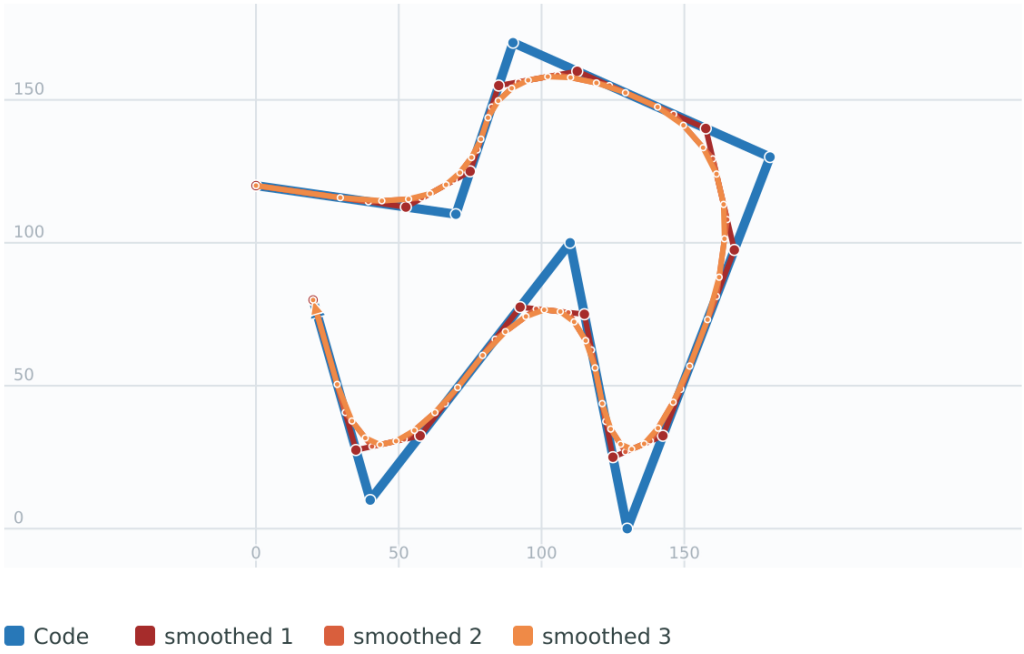
Code

```
SELECT n AS iterations, ST_ChaikinSmoothing(
    'LINESTRING (10 140,80 130,100 190,190 150,140 20,120 120,50 30,30 100)', n
  ) AS smoothed
FROM generate_series(1, 3) AS n;
```

Output

```
1 | LINESTRING(10 140,62.5 132.5,85 145,95 175,122.5 180,167.5 160,177.5 117.5,152.5 52.5,135 45,125 95,102.5 ↩
    97.5,67.5 52.5,45 47.5,30 100)
2 | LINESTRING(10 140,49.375 134.375,68.125 135.625,79.375 141.875,87.5 152.5,92.5 167.5,101.875 176.25,115.625 ↩
    178.75,133.75 175,156.25 165,170 149.375,175 128.125,171.25 101.25,158.75 68.75,148.125 50.625,139.375 ↩
    46.875,132.5 57.5,127.5 82.5,119.375 95.625,108.125 96.875,93.75 86.25,76.25 63.75,61.875 51.25,50.625 ↩
    48.75,41.25 60.625,30 100)
3 | LINESTRING(10 140,39.53125 135.78125,54.0625 134.6875,63.4375 135.3125,70.9375 137.1875,76.5625 140.3125,81.40625 ↩
    144.53125,85.46875 149.84375,88.75 156.25,91.25 163.75,94.84375 169.6875,99.53125 174.0625,105.3125 ↩
    176.875,112.1875 178.125,120.15625 177.8125,129.21875 175.9375,139.375 172.5,150.625 167.5,159.6875 ↩
    161.09375,166.5625 153.28125,171.25 144.0625,173.75 133.4375,174.0625 121.40625,172.1875 107.96875,168.125 ↩
    93.125,161.875 76.875,156.09375 64.21875,150.78125 55.15625,145.9375 49.6875,141.5625 47.8125,137.65625 ↩
    49.53125,134.21875 54.84375,131.25 63.75,128.75 76.25,125.46875 85.78125,121.40625 92.34375,116.5625 ↩
    95.9375,110.9375 96.5625,104.53125 94.21875,97.34375 88.90625,89.375 80.625,80.625 69.375,72.65625 ↩
    60.625,65.46875 54.375,59.0625 50.625,53.4375 49.375,48.28125 51.71875,43.59375 57.65625,38.4375 70.46875,30 100)
```

Figure



Smoothing a triangle shows the weakest visual change, so it is kept last.

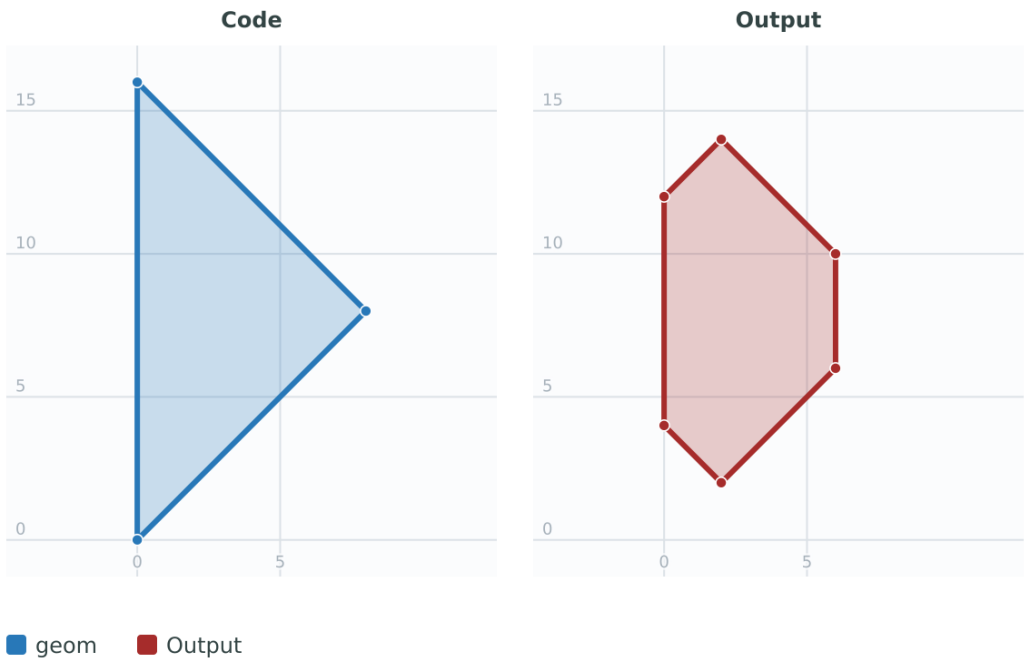
Code

```
SELECT ST_ChaikinSmoothing(geom) smoothed
FROM (SELECT  'POLYGON((0 0,8 8,0 16,0 0))'::geometry geom) AS foo;
```

Output

```
POLYGON((2 2,6 6,6 10,2 14,0 12,0 4,2 2))
```

Figure



Se även

ST_Simplify, ST_SimplifyPreserveTopology, ST_SimplifyVW

7.14.5 ST_CatmullRomSmoothing

ST_CatmullRomSmoothing — Returns a smoothed version of a geometry, using the Catmull-Rom spline algorithm

Synopsis

geometry **ST_CatmullRomSmoothing**(geometry geom, integer nSegments = 5);

Beskrivning

Smoothes a linear or polygonal geometry using the Catmull-Rom spline algorithm. Unlike ST_ChaikinSmooth this is an *interpolating* spline: the output curve passes through every original vertex. Between each pair of consecutive original vertices, `nSegments` - 1 new intermediate points are inserted.

The `nSegments` parameter controls the density of the output. With `nSegments` = 5 (the default), each original span is divided into 5 sub-segments (inserting 4 new points per span). The minimum value is 2.

At least 4 input vertices are required to apply smoothing; geometries with fewer vertices are returned unchanged. Points and multipoints are always returned unchanged.

Notera
The output vertex count grows as `1 + (N-1) * nSegments` for open lines, and `1 + N * nSegments` for closed rings, where N is the number of input vertices. For large geometries, use a simplification function on the result to reduce the number of points (see ST_Simplify, ST_SimplifyPreserveTopology and ST_SimplifyVW).

Resultatet har interpolerade värden för Z- och M-dimensionerna när sådana finns.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Tillgänglighet: 3.6.0

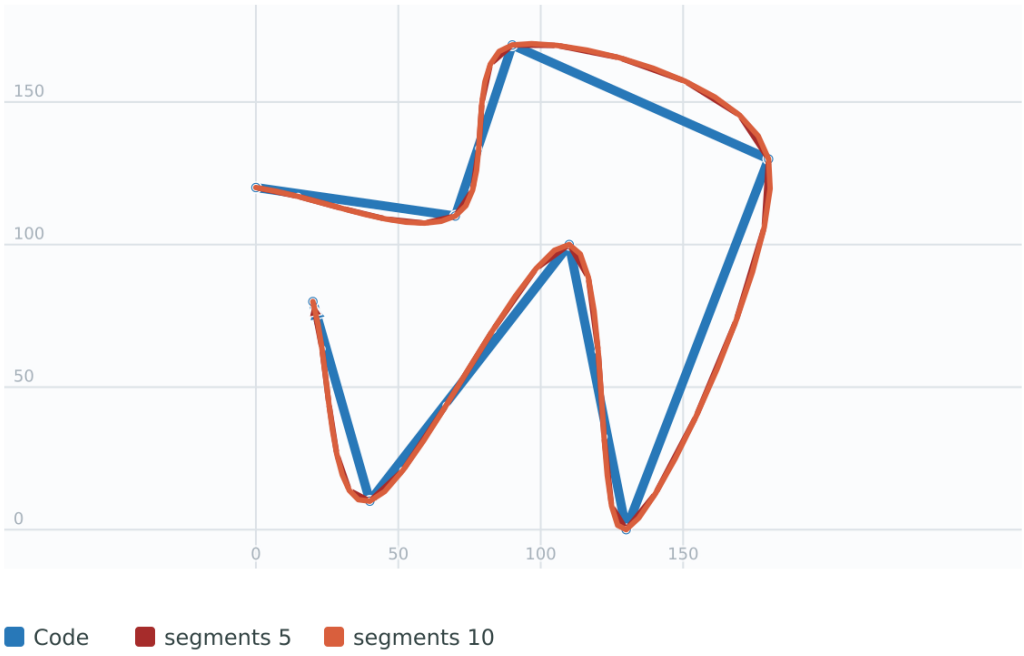
Exempel

Smoothing a LineString using nSegments = 5 and 10.

Code

```
WITH input(geom) AS (VALUES
  ('LINESTRING (10 140,80 130,100 190,190 150,140 20,120 120,50 30,30 100)'::geometry)
)
SELECT ST_CatmullRomSmoothing(geom, 5) AS segments_5,
  ST_CatmullRomSmoothing(geom, 10) AS segments_10
FROM input;
```

Figure



Smoothing a 4-point collinear line with default nSegments=5 shows the minimal change case.

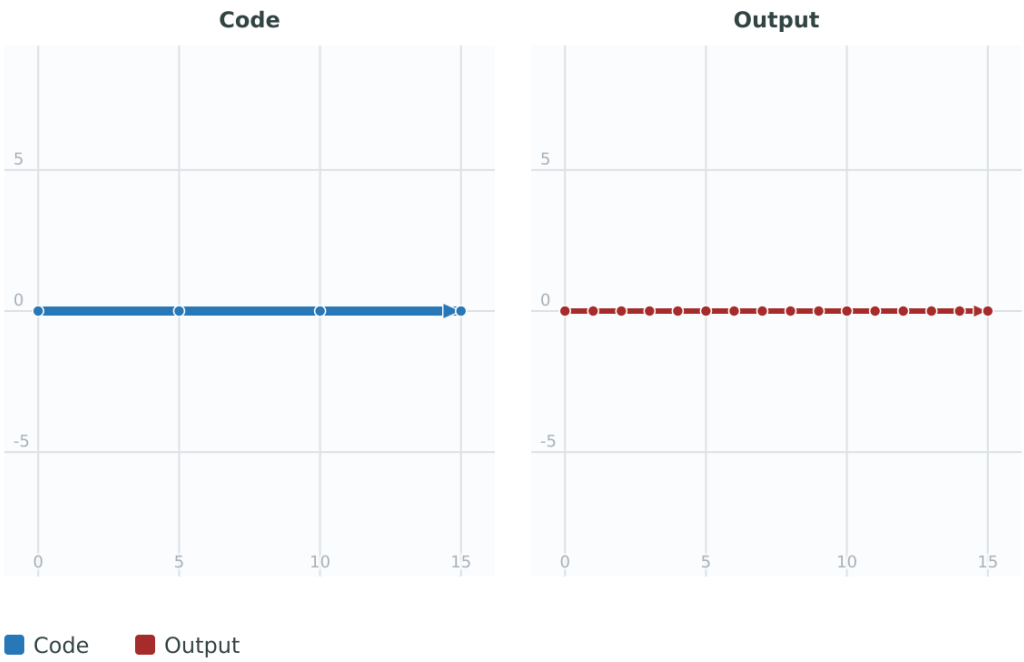
Code

```
SELECT ST_CatmullRomSmoothing('LINESTRING(0 0,5 0,10 0,15 0)');
```

Output

```
LINESTRING(0 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 0,10 0,11 0,12 0,13 0,14 0,15 0)
```

Figure

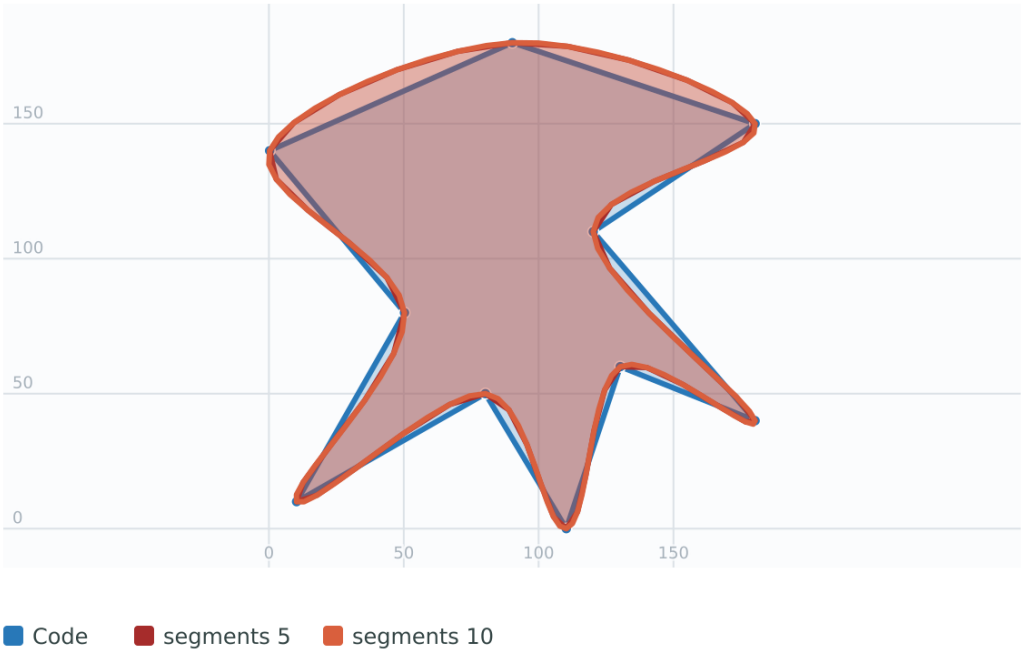


Smoothing a Polygon using nSegments = 5 and 10. This larger example is kept last.

Code

```
WITH input(geom) AS (VALUES
  ('POLYGON ((20 20,60 90,10 150,100 190,190 160,130 120,190 50,140 70,120 10,90 60,20 20))
    '::geometry)
)
SELECT ST_CatmullRomSmoothing(geom, 5) AS segments_5,
  ST_CatmullRomSmoothing(geom, 10) AS segments_10
FROM input;
```

Figure



Se även

ST_ChaikinSmoothing, ST_Simplify, ST_SimplifyPreserveTopology, ST_SimplifyVW

7.14.6 ST_ConcaveHull

ST_ConcaveHull — Beräknar en eventuellt konkav geometri som innehåller alla indatageometrins toppar

Synopsis

geometry **ST_ConcaveHull**(geometry param_geom, float param_pctconvex, boolean param_allow_holes = false);

Beskrivning

Ett konkavt skrov är en (vanligtvis) konkav geometri som innehåller indata och vars hörn är en delmängd av indatans hörn. I det allmänna fallet är det konkava skrovet en polygon. Det konkava skrovet av två eller flera kollinjära punkter är en LineString med två punkter. Det konkava skrovet av en eller flera identiska punkter är en Point. Polygonen kommer inte att innehålla hål om inte det valfria `param_allow_holes-argumentet` anges som true.

Man kan tänka på ett konkavt skrov som att ”krympa ihop”en uppsättning punkter. Detta skiljer sig från det konvexa skrovet, som är mer som att linda ett gummiband runt punkterna. Ett konkavt skrov har i allmänhet en mindre yta och representerar en mer naturlig gräns för inmatningspunkterna.

`Param_pctconvex` styr den beräknade skrovets konkavitet. Ett värde på 1 ger det konvexa skrovet. Värden mellan 1 och 0 ger skrov med ökande konkavitet. Ett värde på 0 ger ett skrov med maximal konkavitet (men fortfarande en enda polygon). Valet av ett lämpligt värde beror på hur indata ser ut, men ofta ger värden mellan 0,3 och 0,1 rimliga resultat.

Notera
Tekniskt sett fastställer `param_pctconvex` en längd som en bråkdel av skillnaden mellan de längsta och kortaste kanterna i Delaunay-trianguleringen av inmatningspunkterna. Kanter som är längre än denna längd ”eroderasfrån trianguleringen. De trianglar som återstår bildar det konkava skrovet.

För punktformiga och linjära inmatningar kommer skrovet att omsluta alla inmatningens punkter. För polygonala indata kommer skrovet att omsluta alla punkter i indata *och även* alla områden som täcks av indata. Om du vill ha ett punktvis skrov av en polygonal indata, konvertera den först till punkter med ST_Points.

This is not an aggregate function. To compute the concave hull of a set of geometries use ST_Collect (e.g. `ST_ConcaveHull(ST_Collect(geom ), 0.80)`.

Tillgänglighet: 2.0.0

Förbättrad: 3.3.0, inbyggd GEOS-implementering aktiverad för GEOS 3.11+

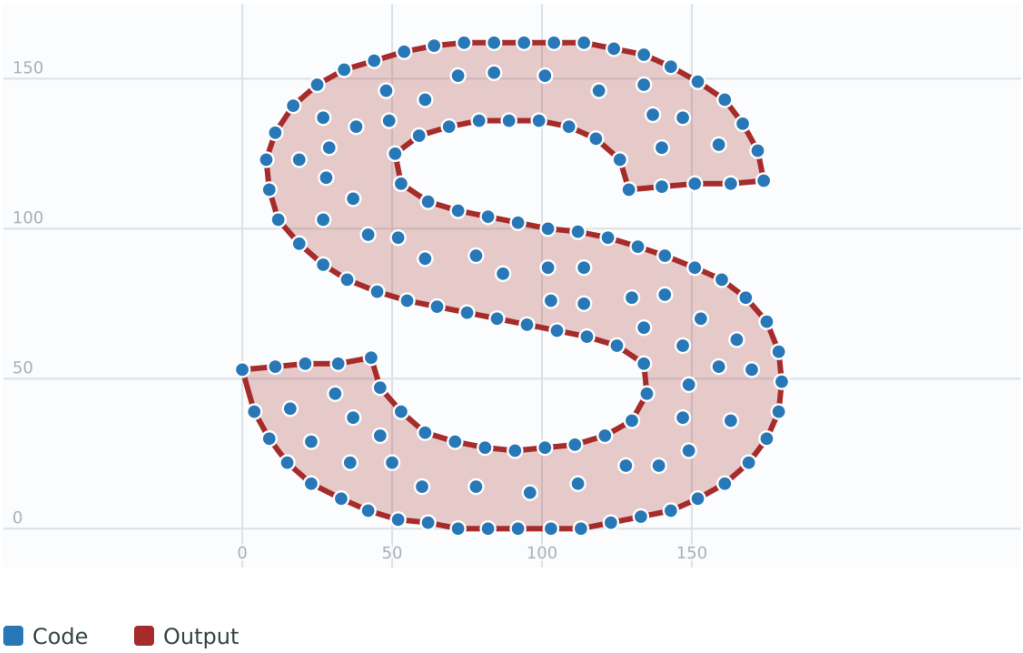
Exempel

Concave hull of a MultiPoint.

Code

```
SELECT ST_ConcaveHull('MULTIPOINT ((10 72),(53 76),(56 66),(63 58),(71 51),(81 48),(91 46) ↩
    ,(101 45),(111 46),(121 47),(131 50),(140 55),(145 64),(144 74),(135 80),(125 83),(115  ↩
    85),(105 87),(95 89),(85 91),(75 93),(65 95),(55 98),(45 102),(37 107),(29 114),(22 122) ↩
    ,(19 132),(18 142),(21 151),(27 160),(35 167),(44 172),(54 175),(64 178),(74 180),(84  ↩
    181),(94 181),(104 181),(114 181),(124 181),(134 179),(144 177),(153 173),(162 168),(171 ↩
     162),(177 154),(182 145),(184 135),(139 132),(136 142),(128 149),(119 153),(109 155) ↩
    ,(99 155),(89 155),(79 153),(69 150),(61 144),(63 134),(72 128),(82 125),(92 123),(102  ↩
    121),(112 119),(122 118),(132 116),(142 113),(151 110),(161 106),(170 102),(178 96),(185 ↩
     88),(189 78),(190 68),(189 58),(185 49),(179 41),(171 34),(162 29),(153 25),(143 23) ↩
    ,(133 21),(123 19),(113 19),(102 19),(92 19),(82 19),(72 21),(62 22),(52 25),(43 29),(33 ↩
     34),(25 41),(19 49),(14 58),(21 73),(31 74),(42 74),(173 134),(161 134),(150 133),(97  ↩
    104),(52 117),(157 156),(94 171),(112 106),(169 73),(58 165),(149 40),(70 33),(147 157) ↩
    ,(48 153),(140 96),(47 129),(173 55),(144 86),(159 67),(150 146),(38 136),(111 170),(124 ↩
     94),(26 59),(60 41),(71 162),(41 64),(88 110),(122 34),(151 97),(157 56),(39 146),(88  ↩
    33),(159 45),(47 56),(138 40),(129 165),(33 48),(106 31),(169 147),(37 122),(71 109) ↩
    ,(163 89),(37 156),(82 170),(180 72),(29 142),(46 41),(59 155),(124 106),(157 80),(175  ↩
    82),(56 50),(62 116),(113 95),(144 167))',
         0.1 );
```

Figure

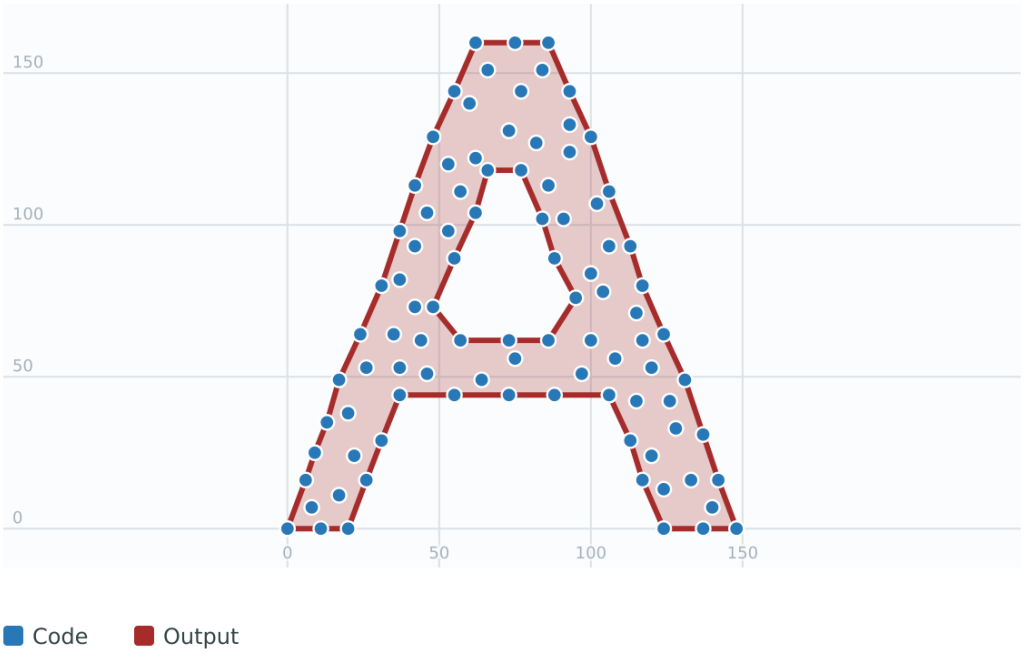


Concave hull of a MultiPoint, allowing holes.

Code

```
SELECT ST_ConcaveHull('MULTIPOINT ((132 64),(114 64),(99 64),(81 64),(63 64),(57 49),(52
    36),(46 20),(37 20),(26 20),(32 36),(39 55),(43 69),(50 84),(57 100),(63 118),(68 133)
    ,(74 149),(81 164),(88 180),(101 180),(112 180),(119 164),(126 149),(132 131),(139 113)
    ,(143 100),(150 84),(157 69),(163 51),(168 36),(174 20),(163 20),(150 20),(143 36),(139
    49),(132 64),(99 151),(92 138),(88 124),(81 109),(74 93),(70 82),(83 82),(99 82),(112
    82),(126 82),(121 96),(114 109),(110 122),(103 138),(99 151),(34 27),(43 31),(48 44),(46
     58),(52 73),(63 73),(61 84),(72 71),(90 69),(101 76),(123 71),(141 62),(166 27),(150
    33),(159 36),(146 44),(154 53),(152 62),(146 73),(134 76),(143 82),(141 91),(130 98)
    ,(126 104),(132 113),(128 127),(117 122),(112 133),(119 144),(108 147),(119 153),(110
    171),(103 164),(92 171),(86 160),(88 142),(79 140),(72 124),(83 131),(79 118),(68 113)
    ,(63 102),(68 93),(35 45))',
         0.15, true );
```

Figure



Jämförelse av ett konkavt skrov av en polygon med det konkava skrovet av de ingående punkterna. Skrovet respekterar polygonens gräns, medan det punktbaserade skrovet inte gör det.

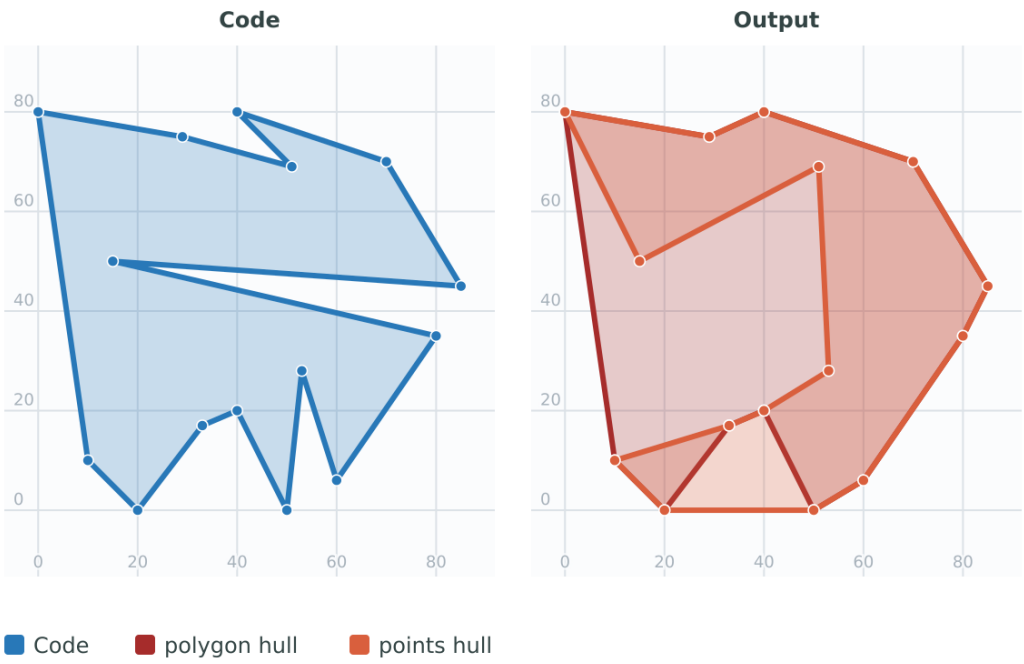
Code

```
WITH data(geom) AS (VALUES
   ('POLYGON ((10 90,39 85,61 79,50 90,80 80,95 55,25 60,90 45,70 16,63 38,60 10,50 30,43 ↩
    27,30 10,20 20,10 90))'::geometry)
)
SELECT ST_ConcaveHull(geom, 0.1) AS polygon_hull,
       ST_ConcaveHull(ST_Points(geom), 0.1) AS points_hull
    FROM data;
```

Output

```
POLYGON((60 10,50 30,43 27,30 10,20 20,10 90,39 85,50 90,80 80,95 55,90 45,70 16,60 10)) | POLYGON((25 60,10 90,39 ↩
    85,50 90,80 80,95 55,90 45,70 16,60 10,30 10,20 20,43 27,50 30,63 38,61 79,25 60))
```

Figure



Används med ST_Collect för att beräkna det konkava skrovet av en geometrisk uppsättning.

This example estimates an infected area from point observations.

Code

```
SELECT disease_type,
    ST_ConcaveHull(ST_Collect(obs_pnt), 0.3 ) AS geom
  FROM disease_obs
  GROUP BY disease_type;
```

Se även

ST_ConvexHull, ST_Collect, CG_AlphaShape, CG_OptimalAlphaShape

7.14.7 ST_ConvexHull

ST_ConvexHull — Beräknar det konvexa skrovet av en geometri.

Synopsis

geometry **ST_ConvexHull**(geometry geomA);

Beskrivning

Beräknar det konvexa skrovet av en geometri. Det konvexa skrovet är den minsta konvexa geometrin som omsluter alla geometrier i indata.

Man kan tänka sig det konvexa skrovet som den geometri som erhålls genom att linda ett gummiband runt en uppsättning geometrier. Detta skiljer sig från ett konkavt skrov som är analogt med att ”krympageometrierna. Ett konvext skrov används ofta för att bestämma ett påverkat område baserat på en uppsättning punktobservationer.

I det allmänna fallet är den konvexa skrovet en polygon. Det konvexa skrovet av två eller flera kollinjära punkter är en tvåpunkts LineString. Det konvexa skrovet av en eller flera identiska punkter är en Point.

Detta är inte en aggregeringsfunktion. För att beräkna det konvexa skrovet av en uppsättning geometrier, använd ST_Collect för att aggregera dem till en geometrisamling (t.ex. `ST_ConvexHull(ST_Collect`

Utförs av GEOS-modulen

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s2.1.1.3

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1.16

Denna funktion stöder 3d och kommer inte att tappa z-index.

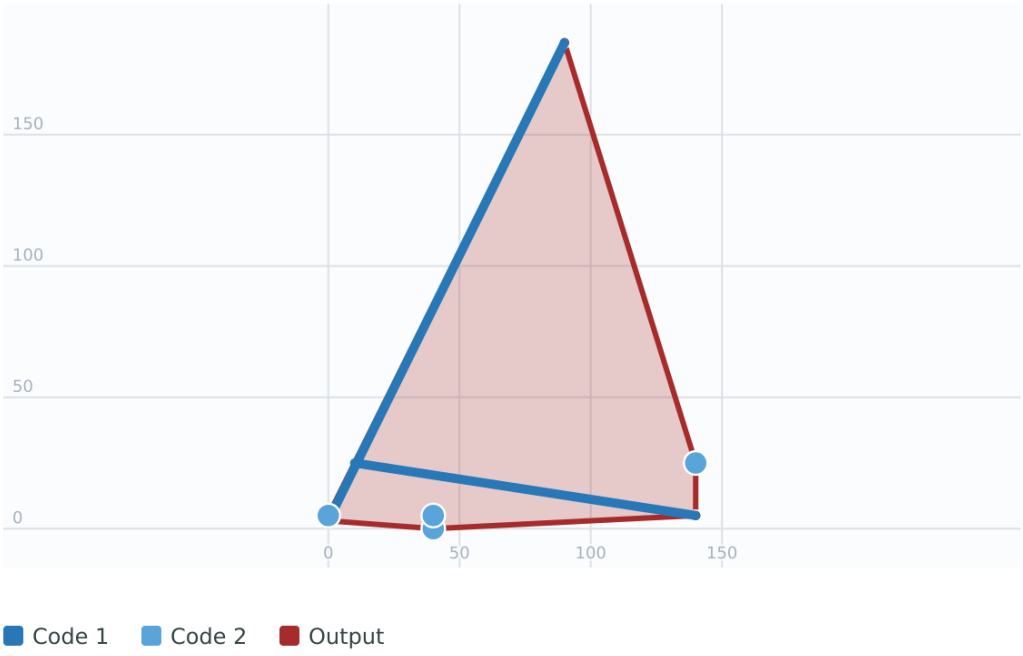
Exempel

Compute the convex hull of a MultiLineString and a MultiPoint.

Code

```
SELECT ST_ConvexHull(ST_Collect(ST_GeomFromText('MULTILINESTRING((100 190,10 8),(150 10,20 ↩
    30))'),
            ST_GeomFromText('MULTIPOINT(50 5,150 30,50 10,10 10)')
            ));
```

Figure



Används med ST_Collect för att beräkna konvexa skrov av geometriska uppsättningar.

This example returns an estimated infected area from point observations.

Code

```
SELECT d.disease_type,
    ST_ConvexHull(ST_Collect(d.geom)) As geom
    FROM disease_obs As d
    GROUP BY d.disease_type;
```

Se även

ST_Collect, ST_ConcaveHull, ST_MinimumBoundingCircle

7.14.8 ST_DelaunayTriangles

ST_DelaunayTriangles — Returnerar Delaunay-trianguleringen av hörnen i en geometri.

Synopsis

geometry **ST_DelaunayTriangles**(geometry g1, float tolerance = 0.0, int4 flags = 0);

Beskrivning

Beräknar Delaunay-trianguleringen av hörnen i indatageometrin. Den valfria `toleransen` kan användas för att fästa ihop närliggande indatavertikaler, vilket förbättrar robustheten i vissa situationer. Resultatgeometrin avgränsas av den konvexa skrovet av indatapunkterna. Representationen av resultatgeometrin bestäms av `flaggkoden`:

- `0` - en GEOMETRYCOLLECTION av triangulära POLYGONER (standard)
- `1` - en MULTILINESTRING av trianguleringens kanter
- `2` - En TIN av trianguleringen

Utförs av GEOS-modulen.

Tillgänglighet: 2.1.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

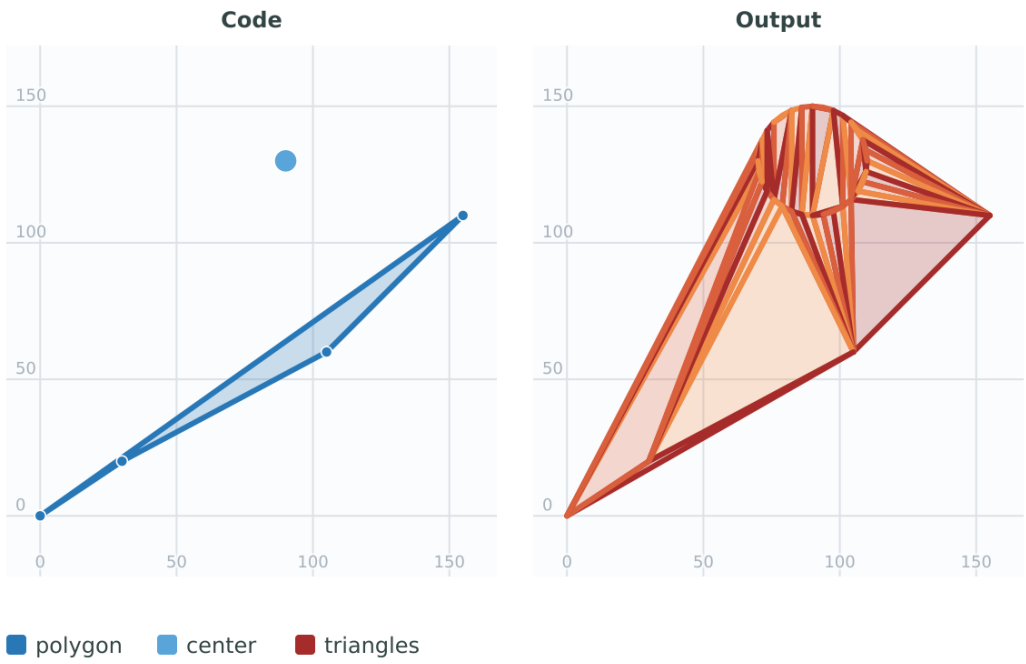
Exempel

Triangulation of the vertices of a polygon and a buffered point.

Code

```
WITH data AS (
  SELECT polygon, ST_Buffer(center, 20) AS point_buffer
  FROM (SELECT
    'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry AS polygon,
    'POINT(110 170)'::geometry AS center
  ) AS input
)
SELECT ST_DelaunayTriangles(ST_Union(polygon, point_buffer)) AS triangles
FROM data;
```

Figure

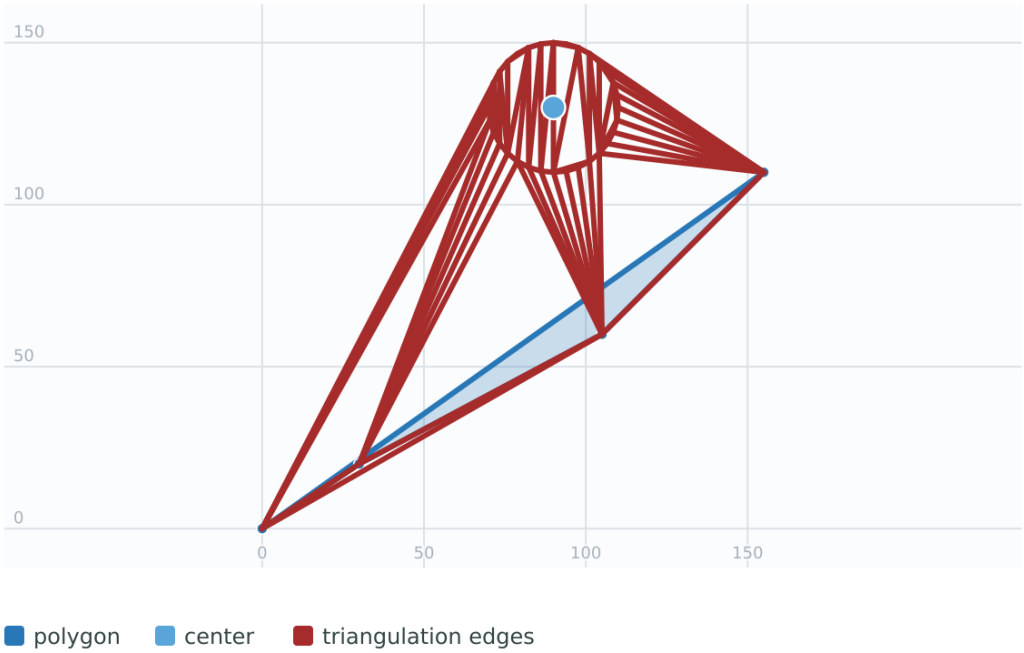


Triangulation edges returned as a `MULTILINESTRING`.

Code

```
WITH data AS (
  SELECT polygon, ST_Buffer(center, 20) AS point_buffer
  FROM (SELECT
    'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry AS polygon,
    'POINT(110 170)'::geometry AS center
  ) AS input
)
SELECT ST_DelaunayTriangles(ST_Union(polygon, point_buffer), 0.001, 1) AS ↩
    triangulation_edges
FROM data;
```

Figure

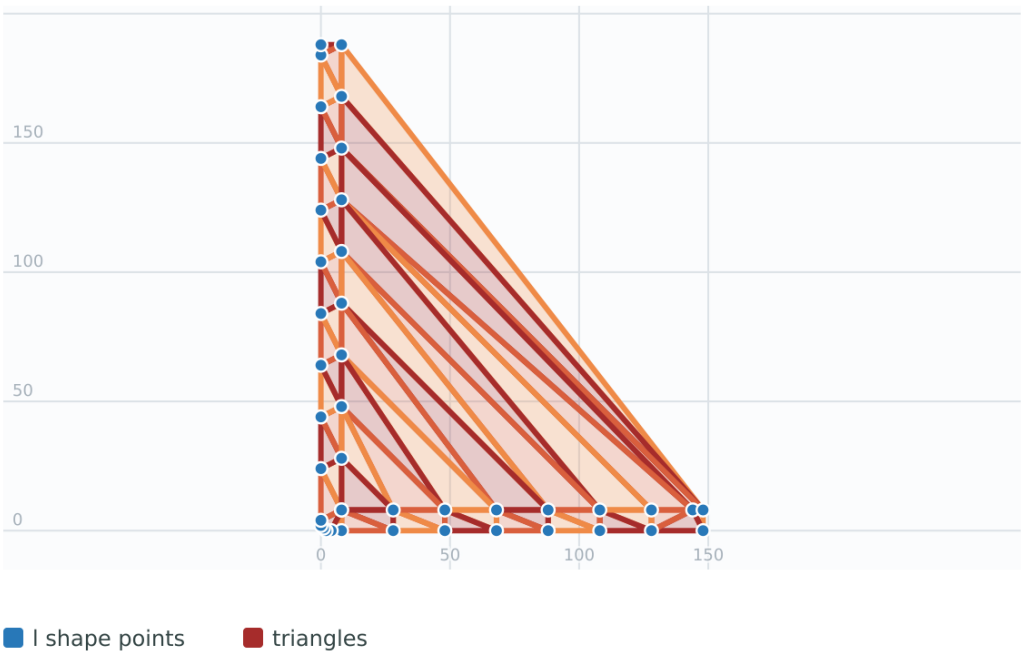


Triangulation of 42 points forming an L shape.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT(14 14,34 14,54 14,74 14,94 14,114 14,134 14,
    150 14,154 14,154 6,134 6,114 6,94 6,74 6,54 6,34 6,
    14 6,10 6,8 6,7 7,6 8,6 10,6 30,6 50,6 70,6 90,6 110,6 130,
    6 150,6 170,6 190,6 194,14 194,14 174,14 154,14 134,14 114,
    14 94,14 74,14 54,14 34,14 14)'::geometry AS l_shape_points
)
SELECT ST_DelaunayTriangles(l_shape_points) AS triangles
FROM data;
```

Figure



Exempel med hörnpunkter med Z-värden.

Code

```
SELECT ST_DelaunayTriangles(
  'MULTIPOINT Z(14 14 10,150 14 100,34 6 25,20 10 150)'::geometry);
```

Se även

ST_VoronoiPolygons, ST_TriangulatePolygon, CG_ConstrainedDelaunayTriangles, ST_VoronoiLines, ST_Con

7.14.9 ST_FilterByM

ST_FilterByM — Tar bort hörn baserat på deras M-värde

Synopsis

geometry **ST_FilterByM**(geometry geom, double precision min, double precision max = null, boolean returnM = false);

Beskrivning

Filtrerar bort vertexpunkter baserat på deras M-värde. Returnerar en geometri med endast vertexpunkter som har ett M-värde som är större än eller lika med min-värdet och mindre än eller lika med max-värdet. Om argumentet max-värde utelämnas beaktas endast min-värdet. Om det fjärde argumentet utelämnas kommer m-värdet inte att finnas med i den resulterande geometrin. Om den resulterande geometrin har för få vertexpunkter kvar för sin geometrityp kommer en tom geometri att returneras. I en geometrisamling kommer geometrier som inte har tillräckligt med punkter att utelämnas i tysthet.

Denna funktion är främst avsedd att användas tillsammans med ST_SetEffectiveArea. ST_EffectiveArea anger den effektiva arean för en vertex i dess m-värde. Med ST_FilterByM är det möjligt att få en förenklad version av geometrin utan några beräkningar, bara genom att filtrera

Notera
Det finns en skillnad i vad ST_SimplifyVW returnerar när inte tillräckligt många punkter uppfyller kriterierna jämfört med ST_FilterByM. ST_SimplifyVW returnerar geometrin med tillräckligt många punkter medan ST_FilterByM returnerar en tom geometri

Notera
Observera att den returnerade geometrin kan vara ogiltig

Notera
Denna funktion returnerar alla dimensioner, inklusive Z- och M-värden

Tillgänglighet: 2.5.0

Exempel

En linestrings är filtrerad

Code

```
SELECT ST_FilterByM(geom, 30) simplified
FROM (SELECT  ST_SetEffectiveArea('LINESTRING(5 2,3 8,6 20,7 25,10 10)'::geometry) geom) As ↩
     foo;
```

Output

```
LINESTRING(5 2,7 25,10 10)
```

Figure

Se även

ST_SetEffectiveArea, ST_SimplifyVW

7.14.10 ST_GeneratePoints

ST_GeneratePoints — Genererar en multipoint av slumpmässiga punkter som ingår i en polygon eller multipolygon.

Synopsis

geometry **ST_GeneratePoints**(geometry g, integer npoints, integer seed = 0);

Beskrivning

ST_GeneratePoints genererar en multipunkt som består av ett givet antal pseudoslumpmässiga punkter som ligger inom inmatningsområdet. Det valfria `fröet` används för att återskapa en deterministisk sekvens av punkter och måste vara större än noll.

Tillgänglighet: 2.3.0

Förbättrad: 3.0.0, tillagd parameter för utsäde

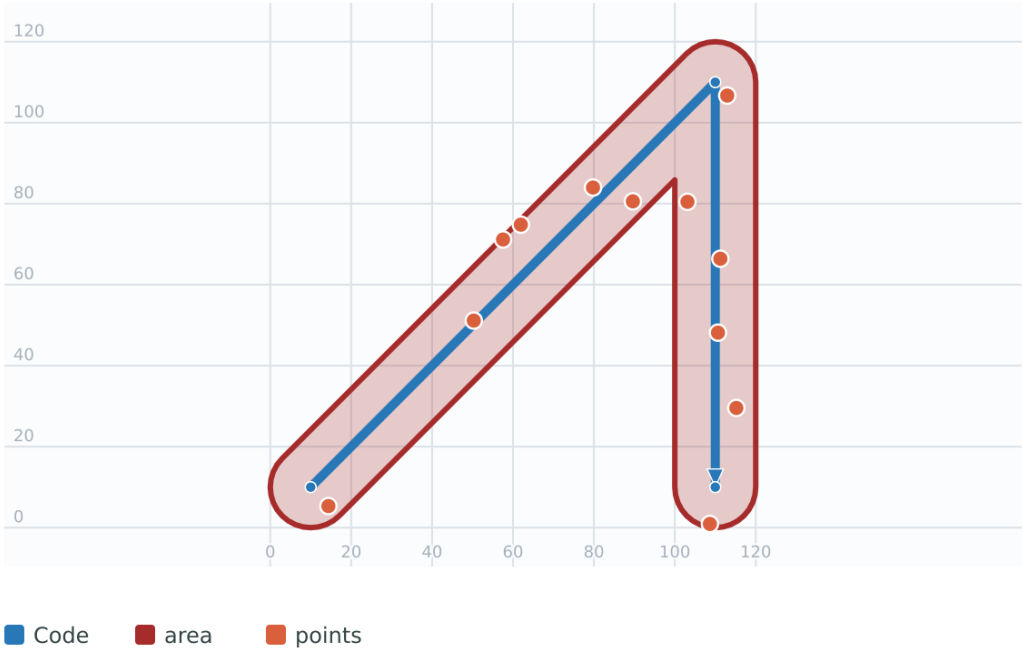
Exempel

Generate 12 points inside a buffered LineString using the deterministic seed 1996.

Code

```
WITH source AS (
  SELECT ST_Buffer(ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'),
           10, 'endcap=round join=round') AS geom
)
SELECT geom AS area,
       ST_GeneratePoints(geom, 12, 1996) AS points
FROM source;
```

Figure



Givet en tabell med polygoner s, returnera 12 individuella punkter per polygon. Resultaten kommer att vara olika varje gång du kör.

Code

```
SELECT s.id, dp.path[1] AS pt_id, dp.geom
FROM s, ST_DumpPoints(ST_GeneratePoints(s.geom, 12)) AS dp;
```

Generate dot-density points from a polygon table, using the absolute value of an attribute as the number of points and keeping the sign of the attribute for styling.

Code

```
SELECT z.id,
       CASE WHEN z.count_delta < 0 THEN -1 ELSE 1 END AS sign,
       dp.path[1] AS pt_id,
       dp.geom
FROM zones AS z
CROSS JOIN LATERAL ST_DumpPoints(
    ST_GeneratePoints(z.geom, abs(z.count_delta)::integer, z.id + 1)
) AS dp
WHERE z.count_delta <> 0;
```

Se även

ST_DumpPoints

7.14.11 ST_GeometricMedian

ST_GeometricMedian — Returnerar den geometriska medianen för en MultiPoint.

Synopsis

geometry **ST_GeometricMedian** (geometry geom, float8 tolerance = NULL, int max_iter = 10000, boolean fail_if_not_converged = false);

Beskrivning

Beräknar den ungefärliga geometriska medianen i en MultiPoint-geometri med hjälp av Weiszfeld-algoritmen. Den geometriska medianen är den punkt som minimerar summan av avstånden till indatapunkterna. Den ger ett centralitetsmått som är mindre känsligt för avvikande punkter än centroiden (masscentrum).

Algoritmen itererar tills avståndsförändringen mellan på varandra följande iterationer är mindre än den angivna `toleransparametern`. Om detta villkor inte har uppfyllts efter `max_iterations` iterationer, producerar funktionen ett fel och avslutas, såvida inte `fail_if_not_converged` är satt till `false` (standard).

Om inget `toleransargument` anges beräknas toleransvärdet baserat på omfattningen av indatageometrin.

Om den finns tolkas M-värdena för inmatningspunkterna som deras relativa vikter.

Tillgänglighet: 2.3.0

Förbättrad: 2.5.0 Lagt till stöd för M som vikt för punkter.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

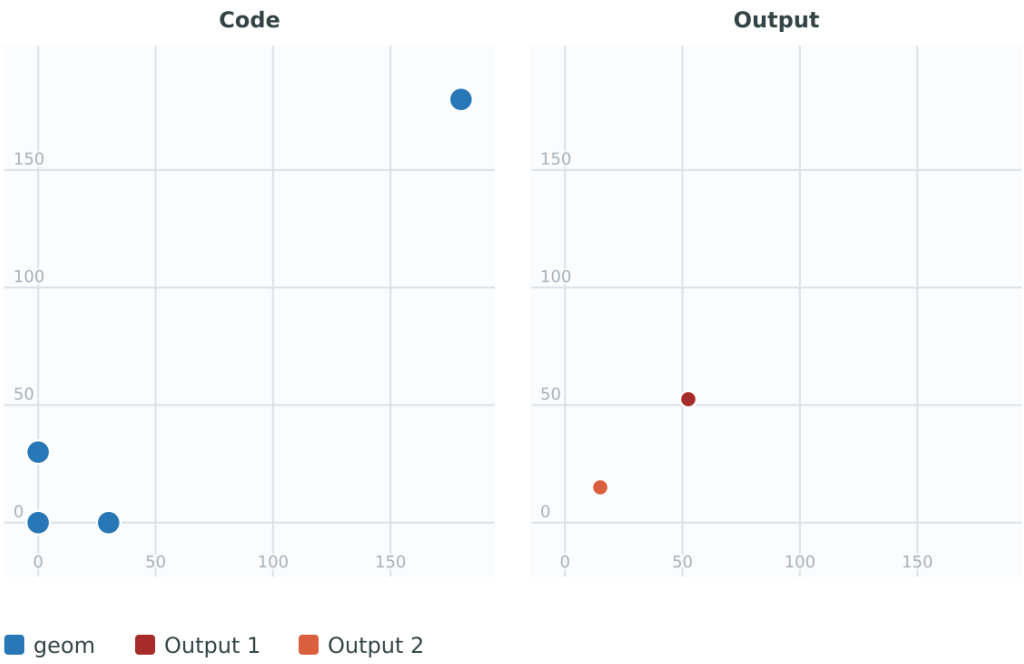
Exempel

Compare the geometric median and centroid of a MultiPoint.

Code

```
WITH test AS (
SELECT 'MULTIPOINT((10 10),(10 40),(40 10),(190 190))'::geometry geom)
SELECT
  ST_Centroid(geom) centroid,
  ST_GeometricMedian(geom) median
FROM test;
```

Figure



Se även

ST_Centroid

7.14.12 ST_LineMerge

ST_LineMerge — Returnerar de linjer som bildas genom att sy ihop en MultiLineString.

Synopsis

geometry **ST_LineMerge**(geometry amultilinestring);
geometry **ST_LineMerge**(geometry amultilinestring, boolean directed);

Beskrivning

Returnerar en LineString eller MultiLineString som bildats genom att sammanfoga linjeelementen i en MultiLineString. Linjer sammanfogas vid sina ändpunkter i 2-vägs korsningar. Linjer sammanfogas inte över korsningar med 3-vägs eller högre grad.

Om **directed** är TRUE kommer ST_LineMerge inte att ändra punktordningen inom LineStrings, så linjer med motsatta riktningar kommer inte att slås samman

Notera
Används endast med MultiLineString/LineStrings. Andra geometrityper returnerar en tom GeometryCollection

Utförs av GEOS-modulen.

Förbättrad: 3.3.0 acceptera en riktad parameter.

Kräver GEOS >= 3.11.0 för att använda den riktade parametern.

Tillgänglighet: 1.1.0

Varning
Denna funktion tar bort M-dimensionen.

Exempel

Cardinality 2

Nodes with cardinality 2 are merged away.

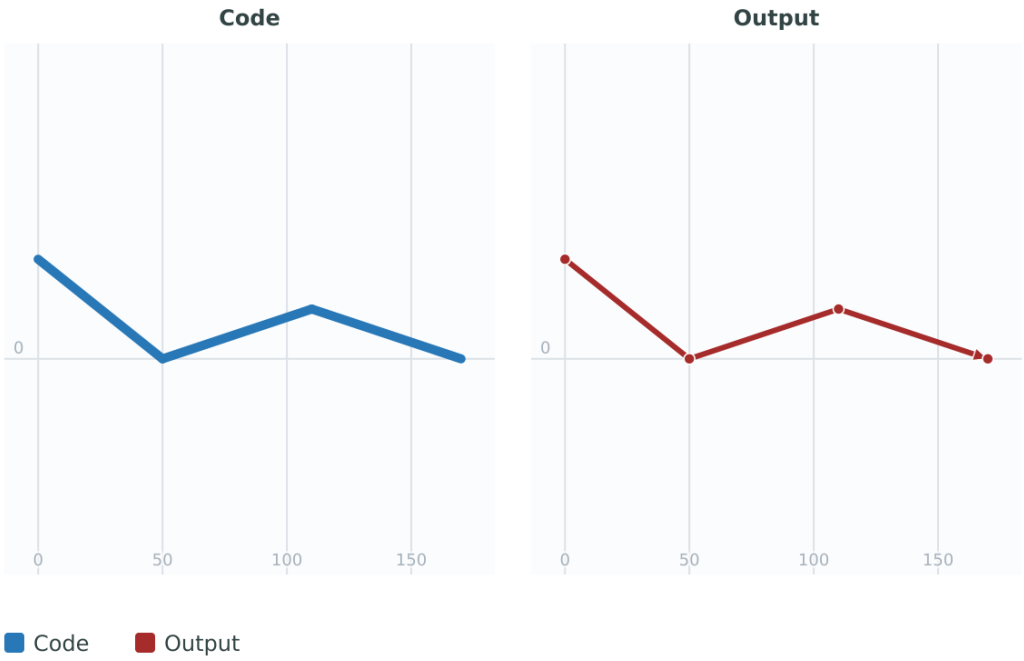
Code

```
SELECT ST_LineMerge('MULTILINESTRING((10 160,60 120),(120 140,60 120),(120 140,180 120))'
);
```

Output

```
LINESTRING(10 160,60 120,120 140,180 120)
```

Figure



Cardinality 3 or More

Linjerna är inte sammanfogade över korsningar med grad > 2.

Code

```
SELECT ST_LineMerge('MULTILINESTRING((10 160,60 120),(120 140,60 120),(120 140,180 120)
    ,(100 180,120 140))'
);
```

Output

```
MULTILINESTRING((10 160,60 120,120 140),(100 180,120 140),(120 140,180 120))
```

Figure

Non-Touching Lines

Om sammanslagning inte är möjlig på grund av linjer som inte berör varandra returneras den ursprungliga MultiLineString.

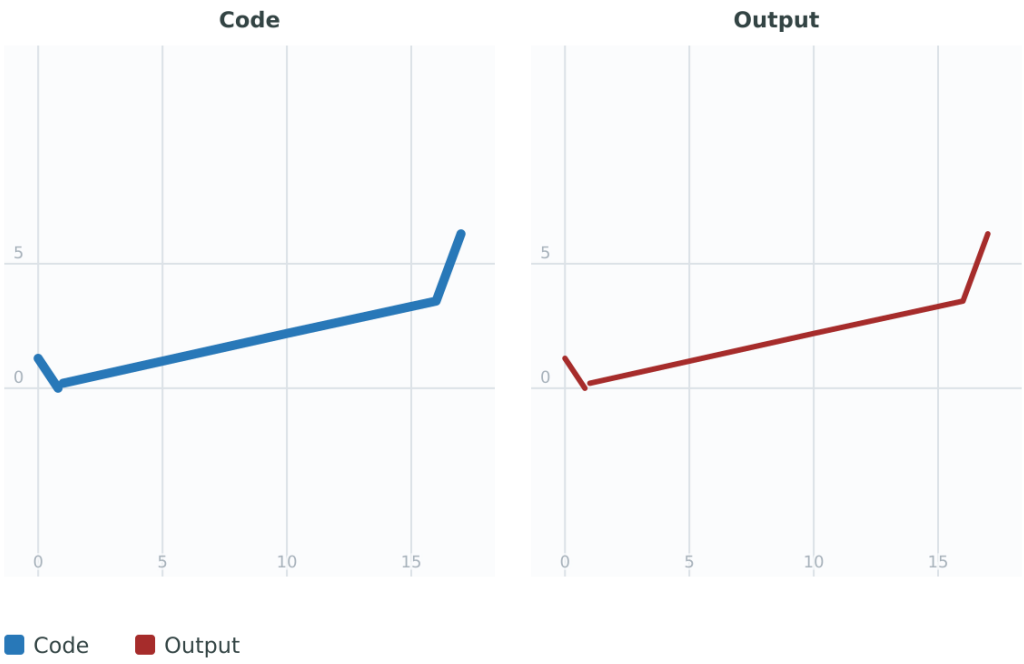
Code

```
SELECT ST_LineMerge('MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33),(-45.2 -33.2,-46
    -32))'
);
```

Output

```
MULTILINESTRING((-45.2 -33.2,-46 -32),(-29 -27,-30 -29.7,-36 -31,-45 -33))
```

Figure



Directed Parameter

Lines with opposite directions are not merged if *`directed`* is TRUE.

Code

```
SELECT ST_LineMerge('MULTILINESTRING((60 30,10 70),(120 50,60 30),(120 50,180 30))',
TRUE);
```

Output

```
MULTILINESTRING((120 50,60 30,10 70),(120 50,180 30))
```

Figure

Z-dimension Handling

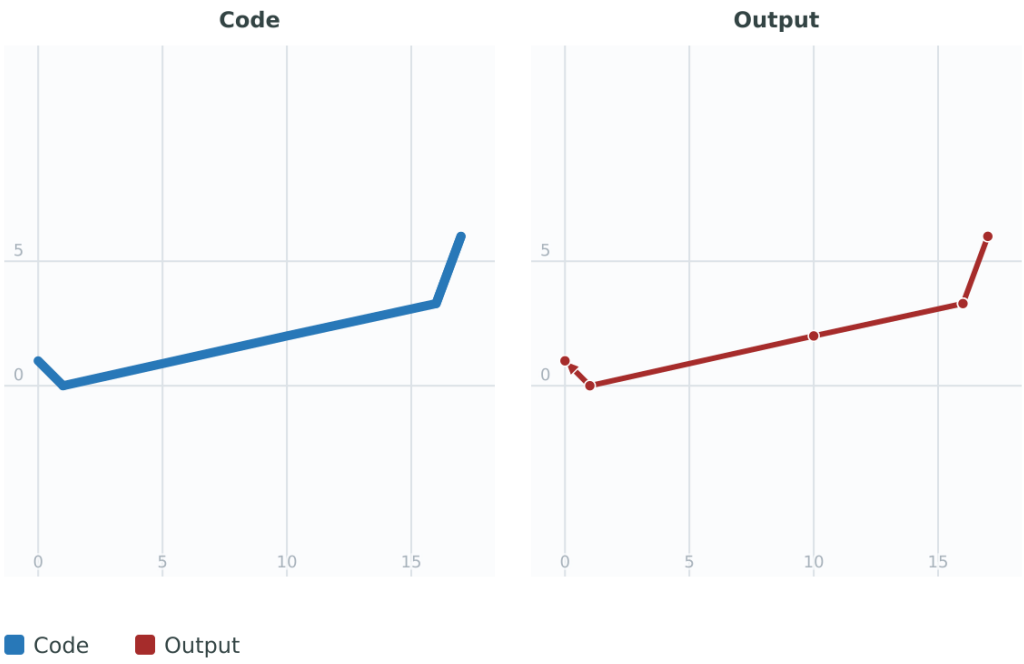
Code

```
SELECT ST_LineMerge('MULTILINESTRING((-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 6),(-29 -27 ↩
     12,-30 -29.7 5),(-45 -33 1,-46 -32 11))'
         );
```

Output

```
LINESTRING Z (-30 -29.7 5,-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 1,-46 -32 11)
```

Figure



Se även

ST_Segmentize, ST_LineSubstring

7.14.13 ST_MaximumInscribedCircle

ST_MaximumInscribedCircle — Beräknar den största cirkeln inom en geometri.

Synopsis

(geometry, geometry, double precision) **ST_MaximumInscribedCircle**(geometry geom);

Beskrivning

Hittar den största cirkeln som ingår i en (multi)polygon, eller som inte överlappar några linjer och punkter. Returnerar en post med fält:

- `center` - cirkelns mittpunkt
- `nearest` - en punkt på geometrin som ligger närmast centrum
- `radie` - cirkelns radie

För polygonala indata skrivs cirkeln in i begränsningsringarna, med de inre ringarna som gränser. För linjära och punktformade indata skrivs cirkeln in i indatats konvexa skrov, med indatans linjer och punkter som ytterligare gränser.

Tillgänglighet: 3.1.0.

Kräver GEOS >= 3.9.0.

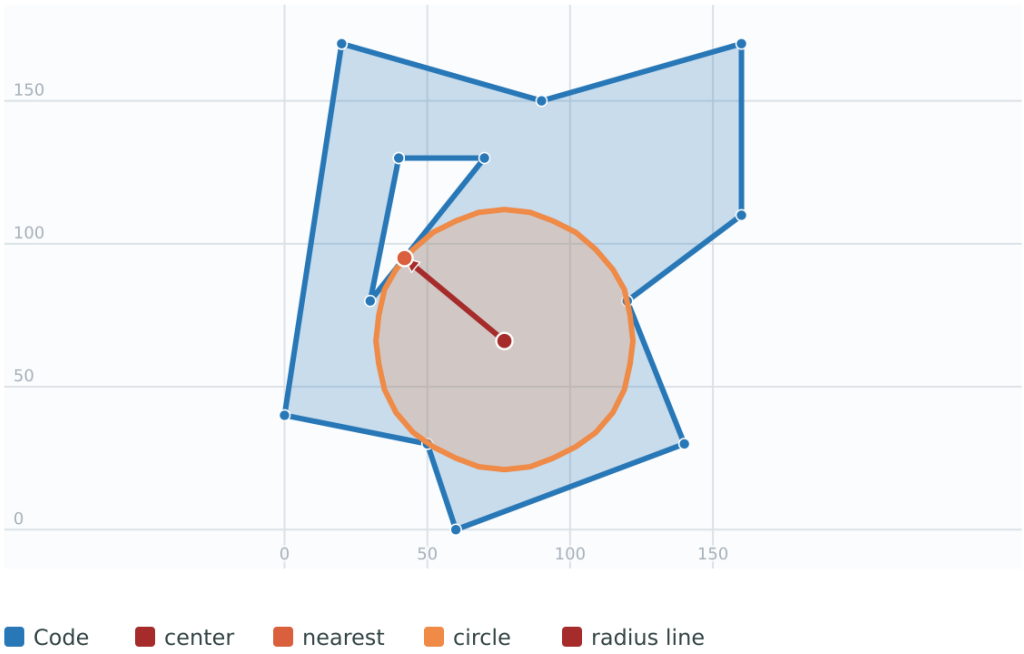
Exempel

Maximum inscribed circle of a polygon. Center, nearest point, radius, circle, and radius line are returned.

Code

```
WITH mic AS (
  SELECT * FROM ST_MaximumInscribedCircle(
    'POLYGON ((40 180,110 160,180 180,180 120,140 90,160 40,80 10,70 40,20 50,40 180),
              (60 140,50 90,90 140,60 140))')
)
SELECT round(radius::numeric, 3) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line
FROM mic;
```

Figure

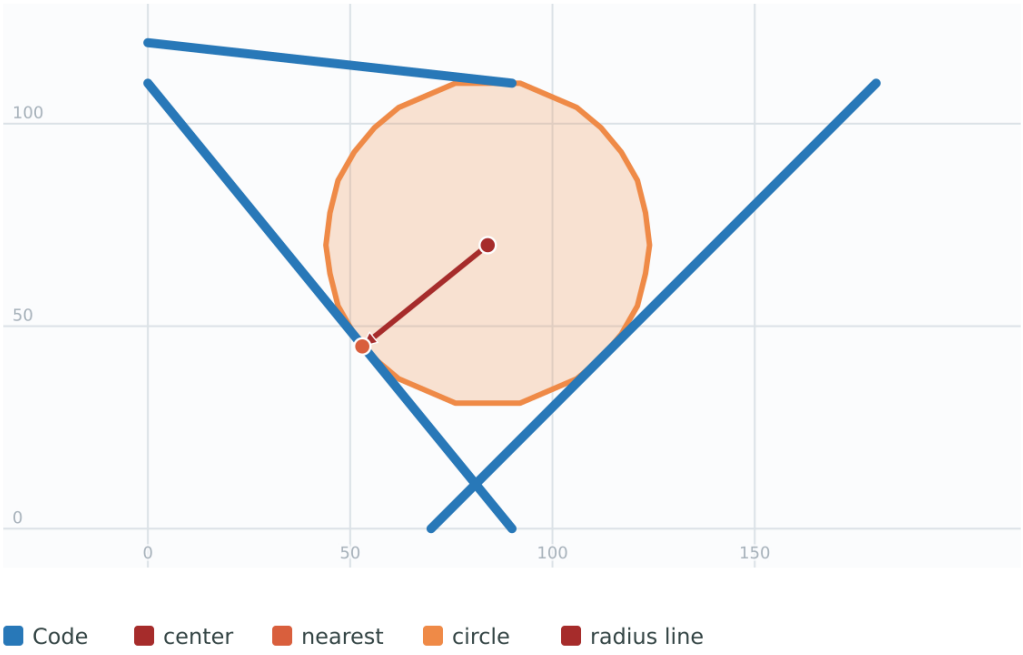


Maximum inscribed circle of a MultiLineString.

Code

```
WITH mic AS (
  SELECT * FROM ST_MaximumInscribedCircle(
    'MULTILINESTRING((100 40,10 150),(10 160,100 150),(190 150,80 40))')
)
SELECT round(radius::numeric, 3) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line
FROM mic;
```

Figure



Se även

ST_MinimumBoundingRadius, ST_LargestEmptyCircle

7.14.14 ST_LargestEmptyCircle

ST_LargestEmptyCircle — Beräknar den största cirkeln som inte överlappar en geometri.

Synopsis

(geometry, geometry, double precision) **ST_LargestEmptyCircle**(geometry geom, double precision tolerance=0.0, geometry boundary=POINT EMPTY);

Beskrivning

Finner den största cirkeln som inte överlappar en uppsättning punkt- och linjehinder. (Polygonala geometrier kan inkluderas som hinder, men endast deras begränsningslinjer används) Cirkelns centrum måste ligga innanför en polygonal gräns, vilken som standard är det konvexa skrovet av indatageometrin. Cirkelns centrum är den punkt innanför gränsen som har det längsta avståndet från hindren. Själva cirkeln består av mittpunkten och en närmaste punkt som ligger på ett hinder som bestämmer cirkelns radie.

If a boundary is supplied it must be a valid polygonal geometry with positive area.

Cirkelns centrum bestäms med en given noggrannhet som specificeras av en avståndstolerans, med hjälp av en iterativ algoritm. Om noggrannhetsavståndet inte specificeras används en rimlig standard.

Returnerar en post med fält:

- `center` - cirkelns mittpunkt

- `nearest` - en punkt på geometrin som ligger närmast centrum
- `radie` - cirkelns radie

För att hitta den största tomma cirkeln i det inre av en polygon, se ST_MaximumInscribedCircle.

Tillgänglighet: 3.4.0.

Kräver GEOS >= 3.9.0.

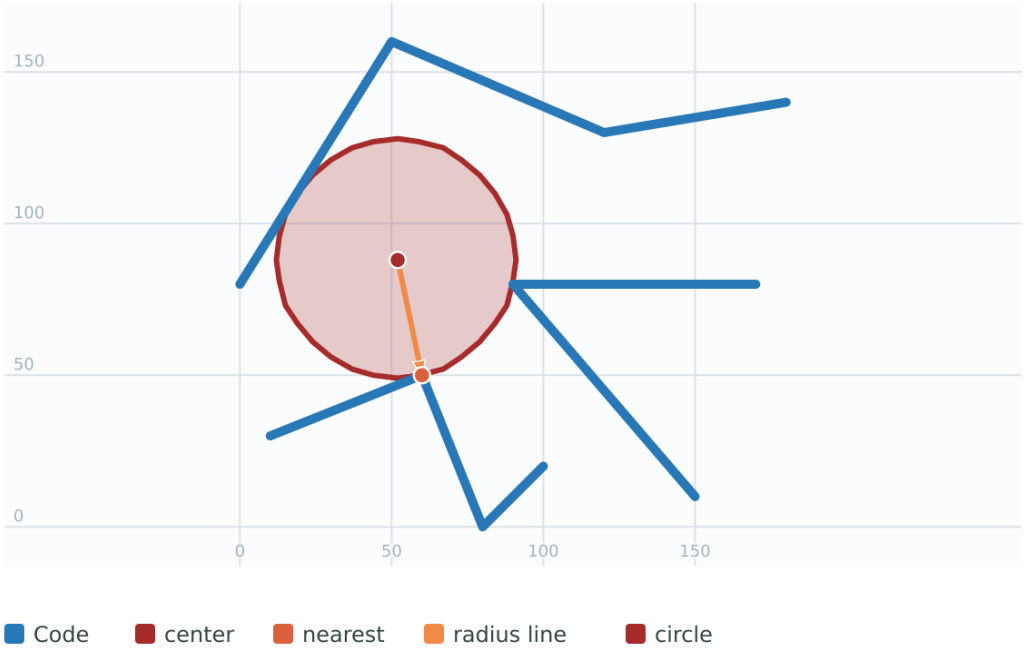
Exempel

Largest empty circle within a set of lines.

Code

```
WITH obstacles(geom) AS (VALUES ('MULTILINESTRING (
        (10 100,60 180,130 150,190 160),
        (20 50,70 70,90 20,110 40),
        (160 30,100 100,180 100))'::geometry)),
circle AS (
  SELECT result.*
  FROM obstacles
  CROSS JOIN LATERAL ST_LargestEmptyCircle(obstacles.geom) AS result
)
SELECT round(radius::numeric, 6) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle
FROM circle;
```

Figure

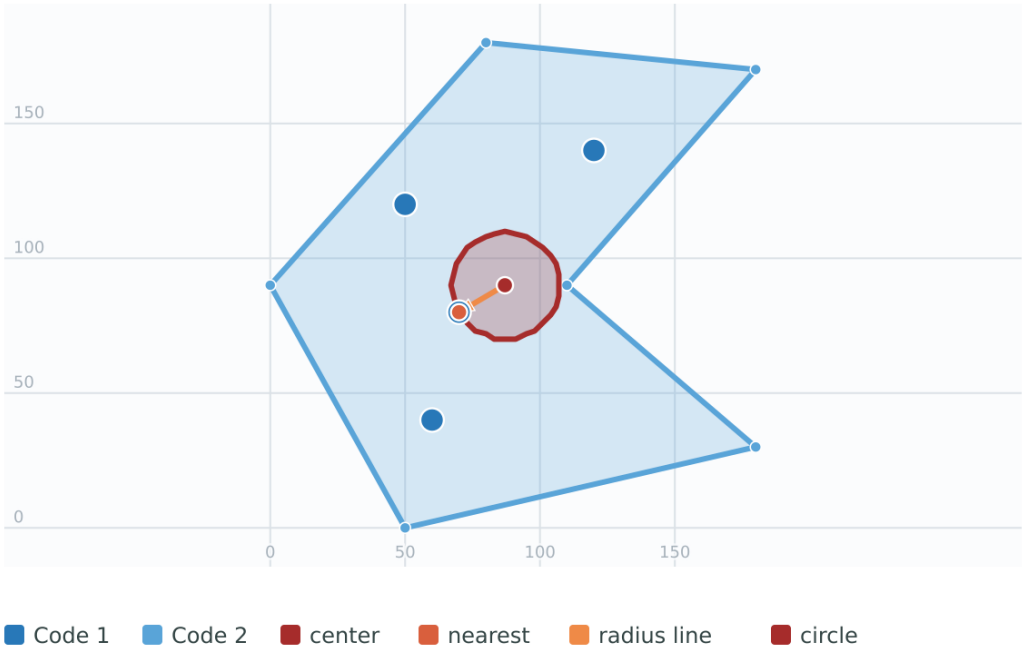


Largest empty circle within a set of points, constrained to lie in a polygon. The constraint polygon boundary is included both as an obstacle and as the constraint for the circle center.

Code

```
WITH inputs(points, boundary) AS (VALUES (
  'MULTIPOINT ((70 50),(60 130),(130 150),(80 90))'::geometry,
  'POLYGON ((90 190,10 100,60 10,190 40,120 100,190 180,90 190))'::geometry
)),
circle AS (
  SELECT result.*
  FROM inputs
  CROSS JOIN LATERAL ST_LargestEmptyCircle(
    ST_Collect(points, boundary), 0, boundary
  ) AS result
)
SELECT round(radius::numeric, 6) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle
FROM circle;
```

Figure



Se även

ST_MinimumBoundingRadius

7.14.15 ST_MinimumBoundingCircle

ST_MinimumBoundingCircle — Returnerar den minsta cirkelpolygonen som innehåller en geometri.

Synopsis

geometry **ST_MinimumBoundingCircle**(geometry geomA, integer num_segs_per_qt_circ=48);

Beskrivning

Returnerar den minsta cirkelpolygonen som innehåller en geometri.

Notera
Begränsningscirkeln approximeras av en polygon med en standard på 48 segment per kvartscirkel. Eftersom polygonen är en approximation av den minsta begränsningscirkeln kan det hända att vissa punkter i indatageometrin inte ryms inom polygonen. Approximationen kan förbättras genom att öka antalet segment. För tillämpningar där en approximation inte är lämplig kan ST_MinimumBoundingRadius användas.

Använd med ST_Collect för att få fram den minsta begränsningscirkeln för en uppsättning geometrier.

För att beräkna två punkter som ligger på den minsta cirkeln (den maximala diametern”) använder du ST_LongestLine.

Förhållandet mellan en polygons area dividerat med arean av dess minsta begränsningscirkel kallas *Reock compactness score*.

Utförs av GEOS-modulen.

Tillgänglighet: 1.4.0

Exempel

Code

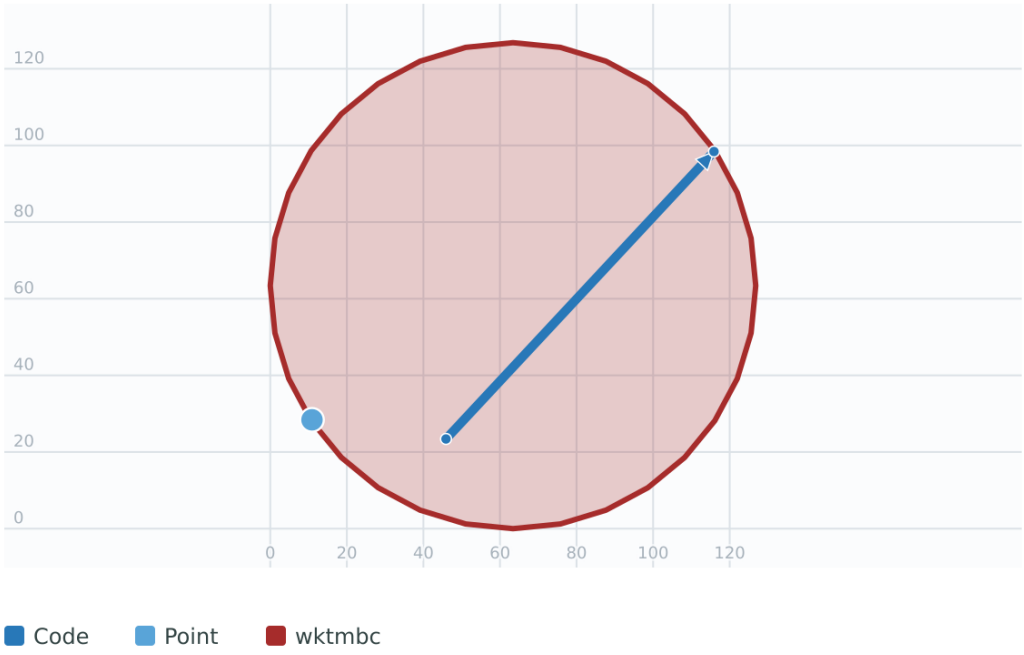
```
SELECT d.disease_type,
ST_MinimumBoundingCircle(ST_Collect(d.geom)) As geom
FROM disease_obs As d
GROUP BY d.disease_type;
```

Minimum bounding circle of a point and linestring, using 8 segments to approximate a quarter circle.

Code

```
SELECT ST_MinimumBoundingCircle(ST_Collect('LINESTRING(55 75,125 150)',
ST_Point(20, 80)), 8
) As wktmbc;
```

Figure



Se även

ST_Collect, ST_MinimumBoundingRadius, ST_LargestEmptyCircle, ST_LongestLine

7.14.16 ST_MinimumBoundingRadius

ST_MinimumBoundingRadius — Returnerar mittpunkten och radien för den minsta cirkeln som innehåller en geometri.

Synopsis

(geometry, double precision) **ST_MinimumBoundingRadius**(geometry geom);

Beskrivning

Beräknar mittpunkt och radie för den minsta cirkel som innehåller en geometri. Returnerar en post med fält:

- `center` - cirkelns mittpunkt
- `radie` - cirkelns radie

Använd med ST_Collect för att få fram den minsta begränsningscirkeln för en uppsättning geometrier.

För att beräkna två punkter som ligger på den minsta cirkeln (den maximala diametern”) använder du ST_LongestLine.

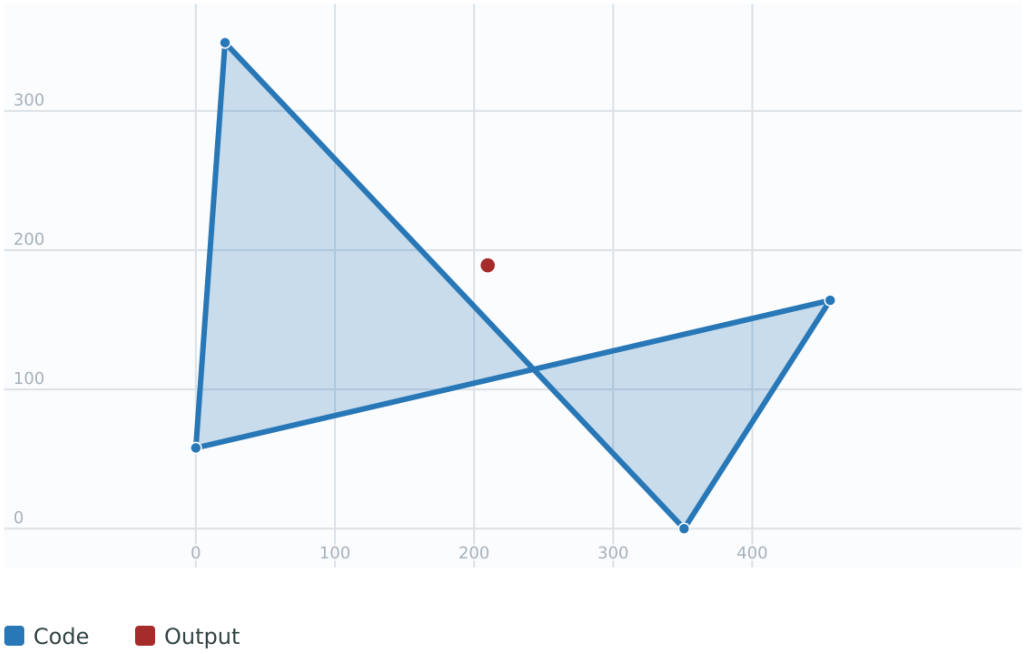
Tillgänglighet - 2.3.0

Exempel

Code

```
SELECT center, radius FROM ST_MinimumBoundingRadius('POLYGON((26426 65078,26531 65242,26075 ↩
    65136,26096 65427,26426 65078))');
```

Figure



Se även

ST_Collect, ST_MinimumBoundingCircle, ST_LongestLine

7.14.17 ST_OrientedEnvelope

ST_OrientedEnvelope — Returnerar en rektangel med minsta yta som innehåller en geometri.

Synopsis

geometry **ST_OrientedEnvelope**(geometry geom);

Beskrivning

Returnerar den roterade rektangel med minsta area som omsluter en geometri. Observera att det kan finnas mer än en sådan rektangel. Kan returnera en Point eller LineString i händelse av degenererade indata.

Tillgänglighet: 2.5.0.

Kräver GEOS >= 3.6.0.

Exempel

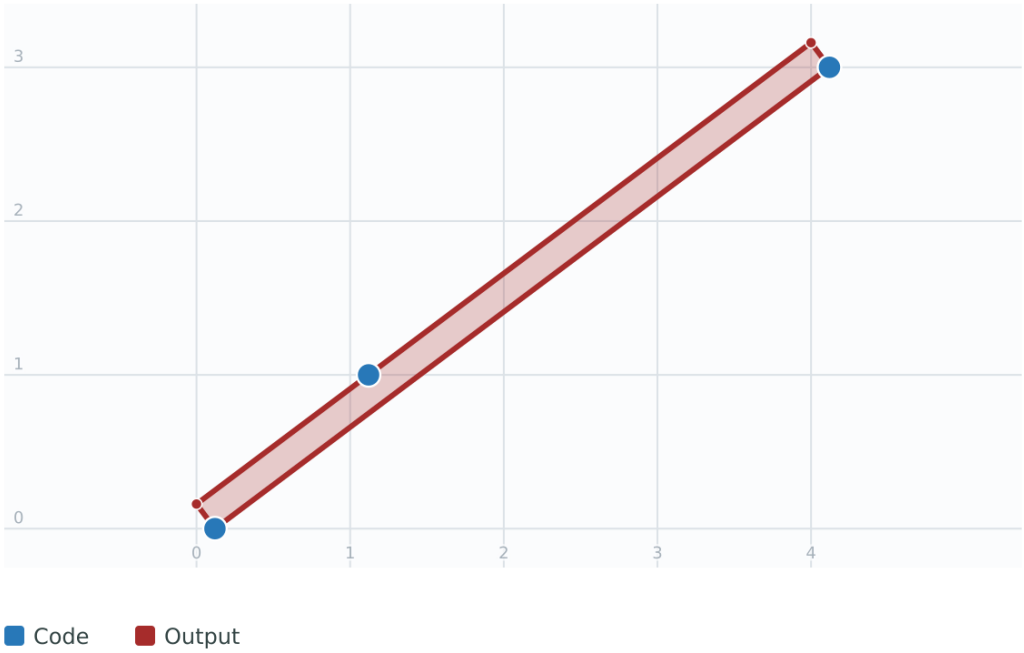
Code

```
SELECT ST_OrientedEnvelope('MULTIPOINT ((0 0),(-1 -1),(3 2))');
```

Output

```
POLYGON((3 2,-1 -1,-1.12 -0.84,2.88 2.16,3 2))
```

Figure



Oriented envelope of a Point and LineString.

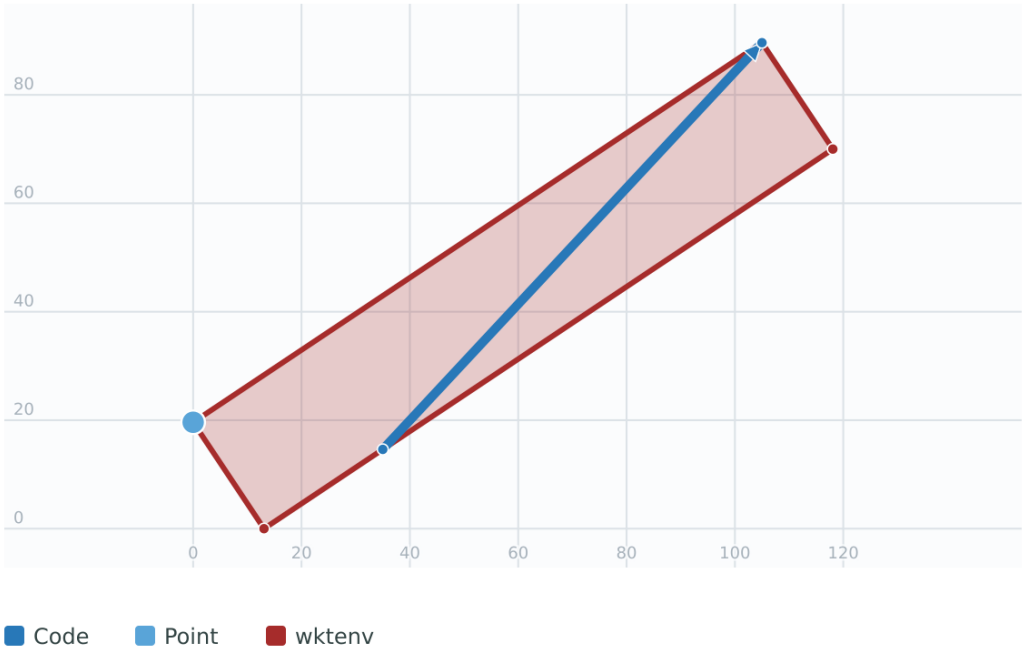
Code

```
SELECT ST_OrientedEnvelope(ST_Collect('LINESTRING(55 75,125 150)',
                ST_Point(20, 80))
                ) As wktenv;
```

Output

```
POLYGON((20 80,125 150,138.0769230769241 130.38461538461573,33.07692307692291 60.384615384615195,20 80))
```

Figure



Se även

ST_Envelope ST_MinimumBoundingCircle

7.14.18 ST_OffsetCurve

ST_OffsetCurve — Returnerar en förskjuten linje på ett givet avstånd och sida från en inmatad linje.

Synopsis

geometry **ST_OffsetCurve**(geometry line, float signed_distance, text style_parameters='');

Beskrivning

Returnerar en offsetlinje på ett givet avstånd och sida från en indatalinje. Alla punkter i de returnerade geometrierna ligger inte längre bort än det angivna avståndet från indatageometrin. Användbar för att beräkna parallella linjer kring en mittlinje.

För positivt avstånd är förskjutningen på vänster sida om inmatningsraden och behåller samma riktning. För ett negativt avstånd är den på höger sida och i motsatt riktning.

Avståndsenheter mäts i enheter i det spatiala referenssystemet.

Observera att utdata kan vara MULTILINESTRING eller EMPTY för vissa pusselformade indatageometrier.

Den valfria tredje parametern gör det möjligt att ange en lista med blankseparerade nyckel=värde-par för att justera operationer enligt följande:

- 'quad_segs=#' : antal segment som används för att approximera en kvartscirkel (standardvärde 8).

- ’join=round|mitre|bevel’ : fogningsstil (standard är round”). Miteräccepteras också som synonym för mitre”.
- ’mitre_limit=#.#’ : gräns för mitraförhållande (påverkar endast mitrad fogning). ’miter_limit’ accepteras också som en synonym till ’mitre_limit’.

Utförs av GEOS-modulen.

Beteendet ändrades i GEOS 3.11 så att offsetkurvorna nu har samma riktning som indatalinjen, för både positiva och negativa offsets.

Tillgänglighet: 2.0

Förbättrad: 2.5 - stöd för GEOMETRYCOLLECTION och MULTILINESTRING har lagts till

Notera
Denna funktion ignorerar Z-dimensionen. Den ger alltid ett 2D-resultat även när den används på en 3D-geometri.

Exempel

Compute an open buffer around roads.

Code

```
SELECT ST_Union(
 ST_OffsetCurve(f.geom, f.width/2, 'quad_segs=4 join=round'),
 ST_OffsetCurve(f.geom, -f.width/2, 'quad_segs=4 join=round')
) as track
FROM someroadstable;
```

Offset 15 units with `quad_segs=4 join=round`.

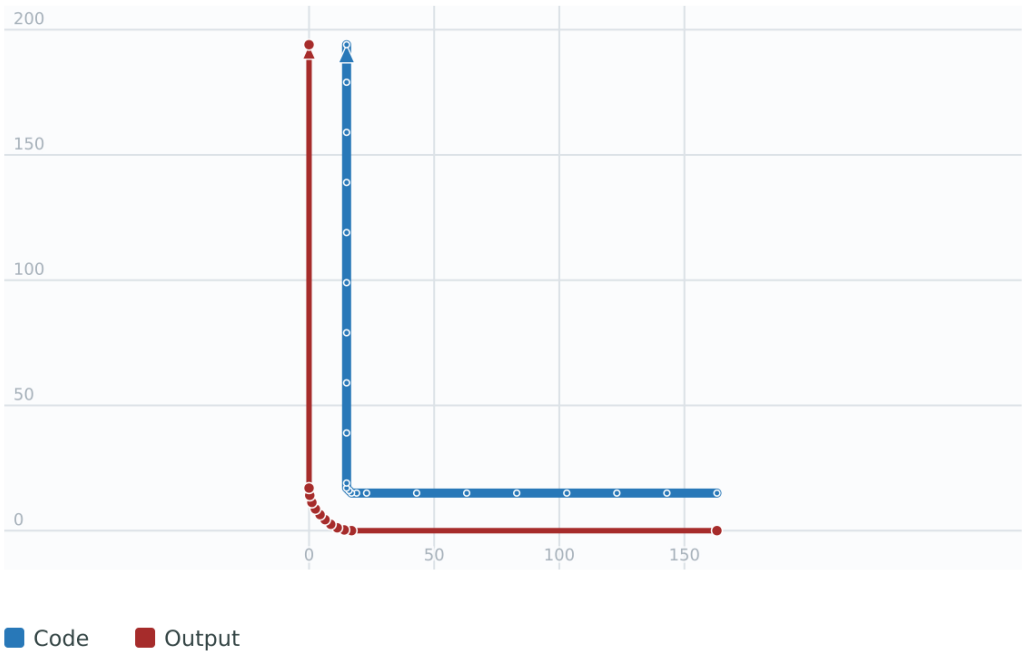
Code

```
SELECT ST_OffsetCurve(
    ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)'),
    15, 'quad_segs=4 join=round');
```

Output

```
LINESTRING(164 1,18 1,15.073645169758077 1.288220793951544,
    12.259748514523654 2.141807012330698,9.66644650470597 3.527955815461818,
    7.393398282201788 5.393398282201788,5.393398282201788 7.393398282201788,
    3.527955815461818 9.66644650470597,2.141807012330696 12.259748514523654,
    1.288220793951544 15.073645169758075,1 18,1 195)
```

Figure



Offset -15 units with `quad_segs=4 join=round`.

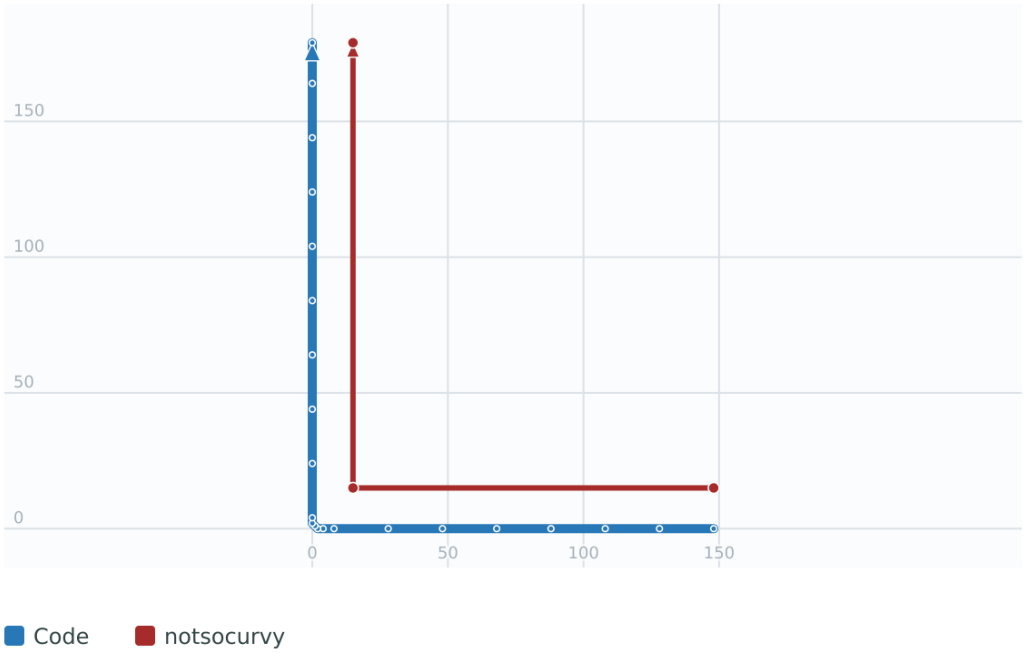
Code

```
SELECT ST_OffsetCurve(
    geom,
    -15,
    'quad_segs=4 join=round') As notsocurvy
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
LINESTRING(164 31,31 31,31 195)
```

Figure



Apply two negative offsets to obtain a smoother curve; the distances combine to -45 units.

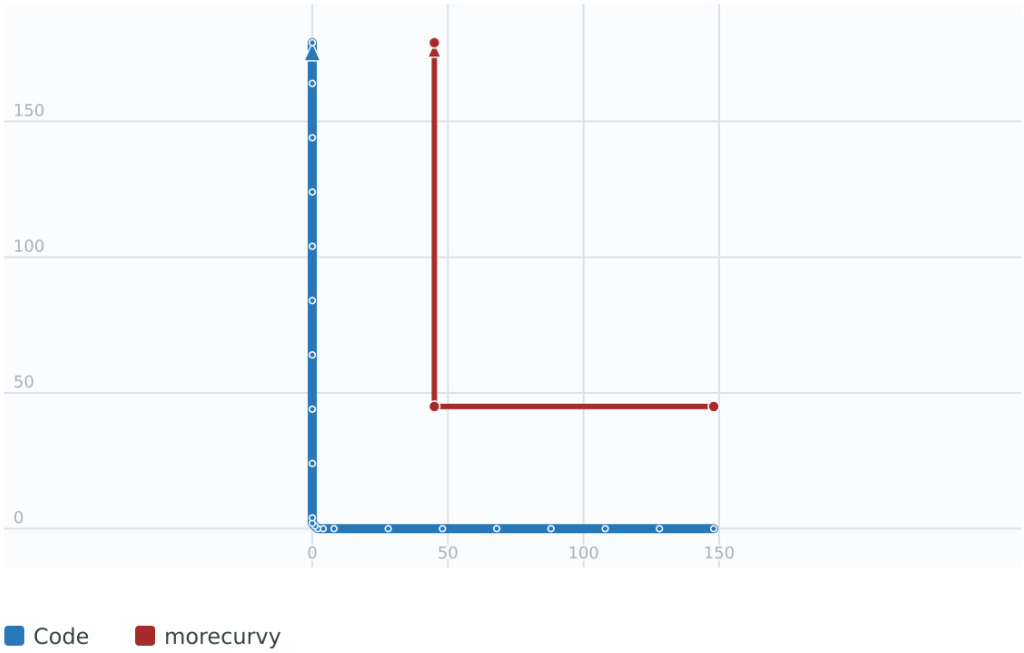
Code

```
SELECT ST_OffsetCurve(
    ST_OffsetCurve(
        geom,
        -30,
        'quad_segs=4 join=round'),
    -15,
    'quad_segs=4 join=round') As morecurvy
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
LINESTRING(164 61,61 61,61 195)
```

Figure



Collect the positive offset and the double-offset to produce parallel curves.

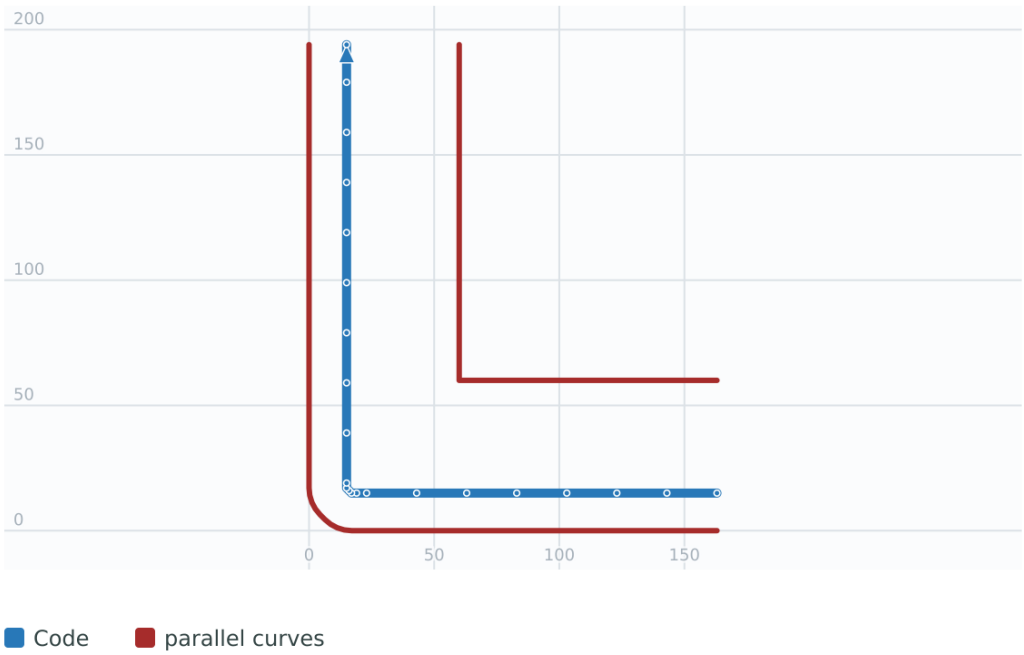
Code

```
SELECT ST_Collect(
    ST_OffsetCurve(geom, 15, 'quad_segs=4 join=round'),
    ST_OffsetCurve(
        ST_OffsetCurve(
            geom,
            -30,
            'quad_segs=4 join=round'),
        -15,
        'quad_segs=4 join=round')
) As parallel_curves
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
MULTILINESTRING((164 1,18 1,15.073645169758077 1.288220793951544,
12.259748514523654 2.141807012330698,9.66644650470597 3.527955815461818,
7.393398282201788 5.393398282201788,5.393398282201788 7.393398282201788,
3.527955815461818 9.66644650470597,2.141807012330696 12.259748514523654,
1.288220793951544 15.073645169758075,1 18,1 195),(164 61,61 61,61 195))
```

Figure



Offset 15 units with a bevel join.

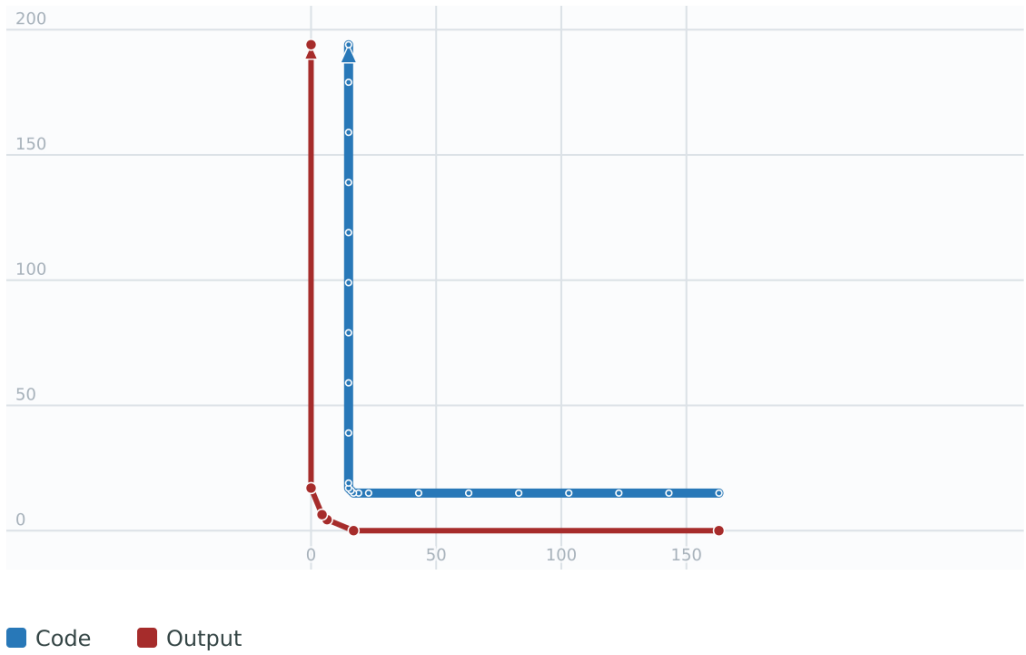
Code

```
SELECT ST_OffsetCurve(
    ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)'),
        15, 'quad_segs=4 join=bevel');
```

Output

```
LINESTRING(164 1,18 1,7.393398282201788 5.393398282201788,
    5.393398282201788 7.393398282201788,1 18,1 195)
```

Figure



Collect offsets on both sides using a mitre join and a mitre limit of 2.2.

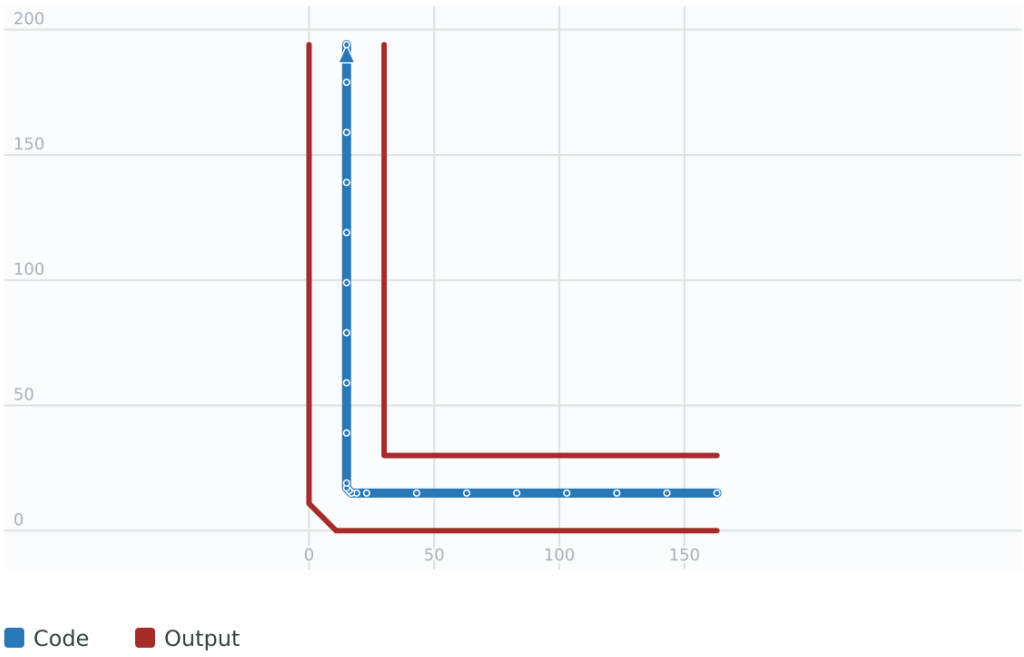
Code

```
SELECT ST_Collect(
    ST_OffsetCurve(geom, 15, 'quad_segs=4 join=mitre mitre_limit=2.2'),
    ST_OffsetCurve(geom, -15, 'quad_segs=4 join=mitre mitre_limit=2.2')
    )
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
MULTILINESTRING((164 1,11.786796564403577 1,1 11.786796564403577,1 195),
    (164 31,31 31,31 195))
```

Figure



Se även

ST_Buffer

7.14.19 ST_PointOnSurface

ST_PointOnSurface — Beräknar en punkt som garanterat ligger i en polygon eller på en geometri.

Synopsis

geometry **ST_PointOnSurface**(geometry g1);

Beskrivning

Returnerar en `POINT` som garanterat ligger i det inre av en yta (`POLYGON`, `MULTIPOLYGON` och `CURVEPOLYGON`). I PostGIS fungerar denna funktion även på linje- och punktgeometrier.

Denna metod implementerar OGC:s implementeringsspecifikation för enkla funktioner för SQL 1.1. s3.2.14.2 // s3.2.18.2

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3: 8.1.5, 9.5.6. I specifikationerna definieras ST_PointOnSurface endast för ytgeometrier. PostGIS utökar funktionen för att stödja alla vanliga geometrityper. Andra databaser (Oracle, DB2, ArcSDE) verkar endast stödja denna funktion för ytor. SQL Server 2008 stöder alla vanliga geometrityper.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Point on surface of a MultiPoint.

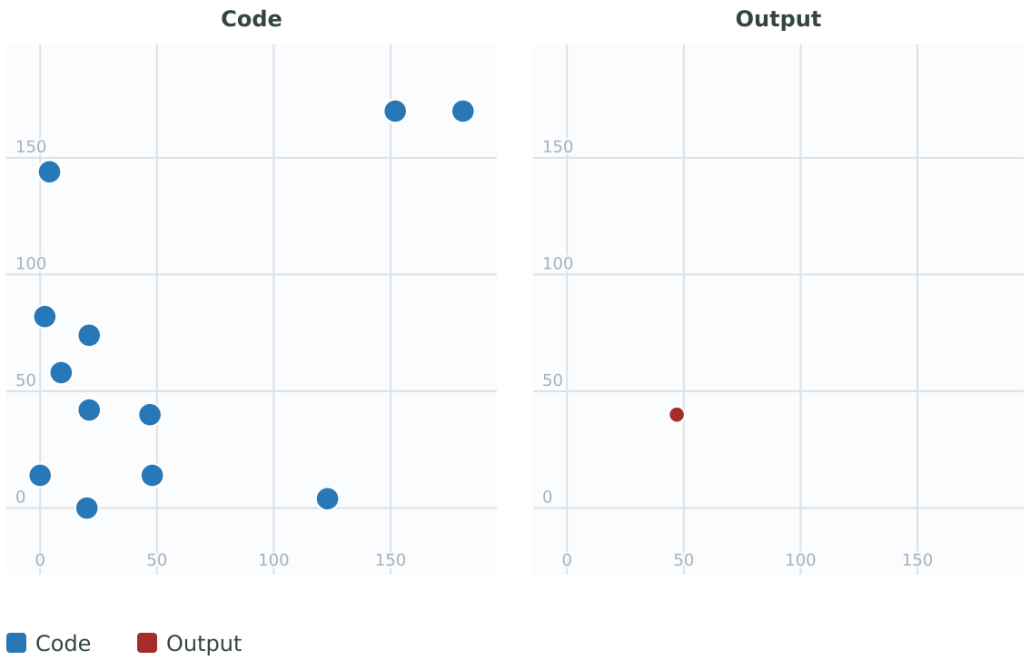
Code

```
SELECT ST_PointOnSurface(
  'MULTIPOINT (8 24,10 92,12 154,17 68,28 10,29 52,29 84,55 50,56 24,131 14,160 180,189 ↩
   180)');
```

Output

```
POINT(55 50)
```

Figure



Point on surface of a LineString.

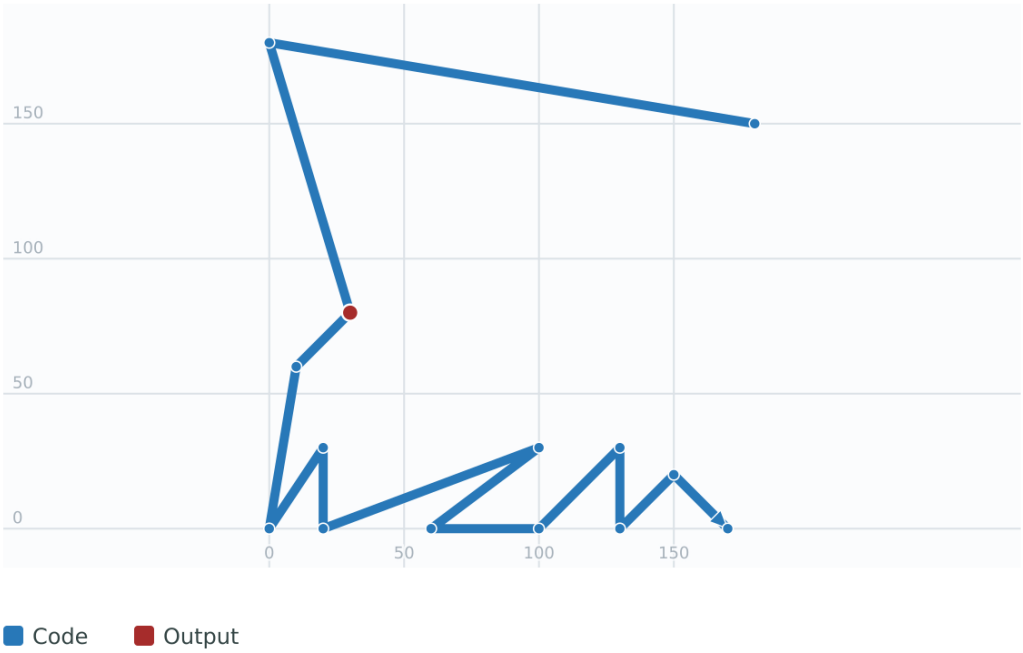
Code

```
SELECT ST_PointOnSurface(
  'LINESTRING (190 160,10 190,40 90,20 70,10 10,30 40,30 10,110 40,70 10,110 10,140 40,140 ↩
   10,160 30,180 10)');
```

Output

```
POINT(40 90)
```

Figure



Point on surface of a Polygon.

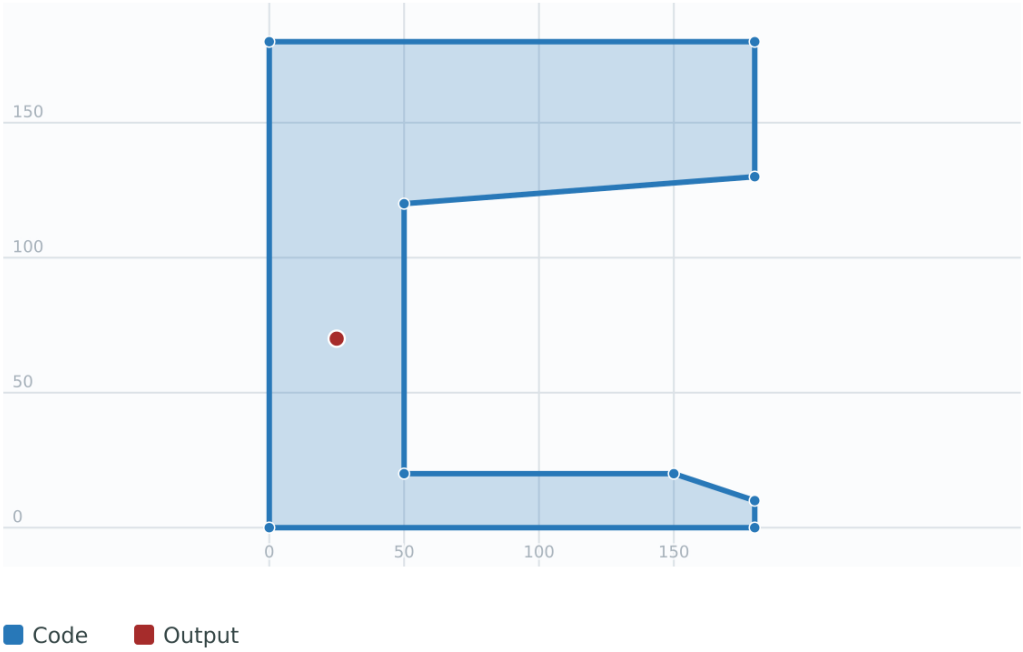
Code

```
SELECT ST_PointOnSurface(
  'POLYGON ((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))');
```

Output

```
POINT(35 80)
```

Figure



Point on surface of the highest-dimension components of a GeometryCollection.

Code

```
SELECT ST_PointOnSurface(
  'GEOMETRYCOLLECTION (POLYGON ((190 170,180 100,80 140,80 160,130 160,110 180,110 190,180 ↩
   180,190 170)),
   LINESTRING (80 120,120 20,140 70,150 30,180 50,190 10),
   MULTIPOINT (19 150,22 49,30 13,32 101,45 35,67 88,75 16))');
```

Output

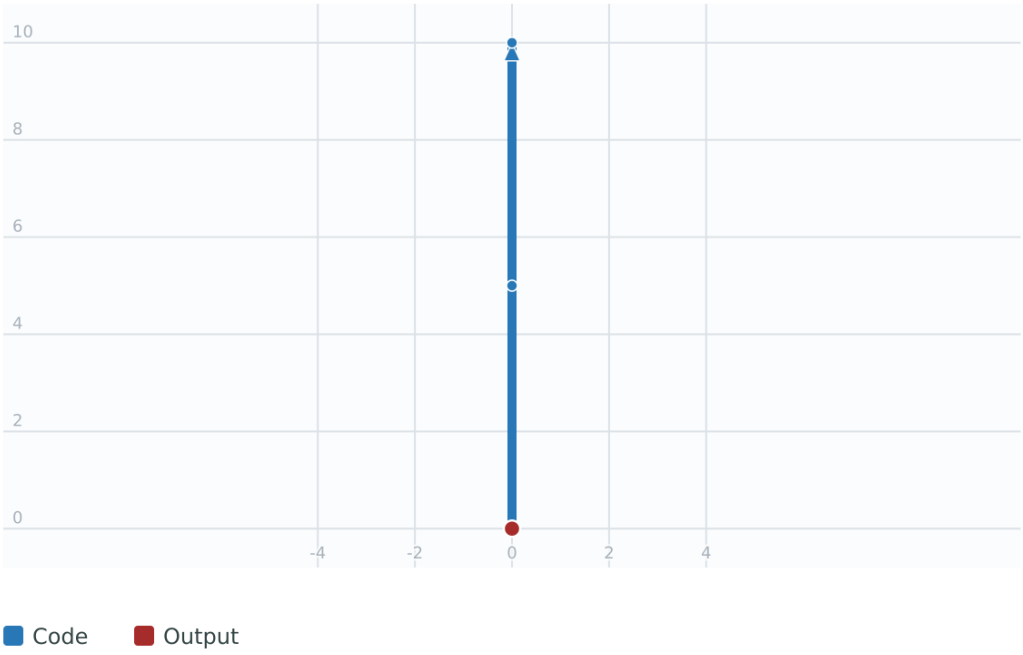
```
POINT(133.571 150)
```

Figure

Code

```
SELECT ST_PointOnSurface('LINESTRING(0 5 1,0 0 1,0 10 2)');
```

Figure

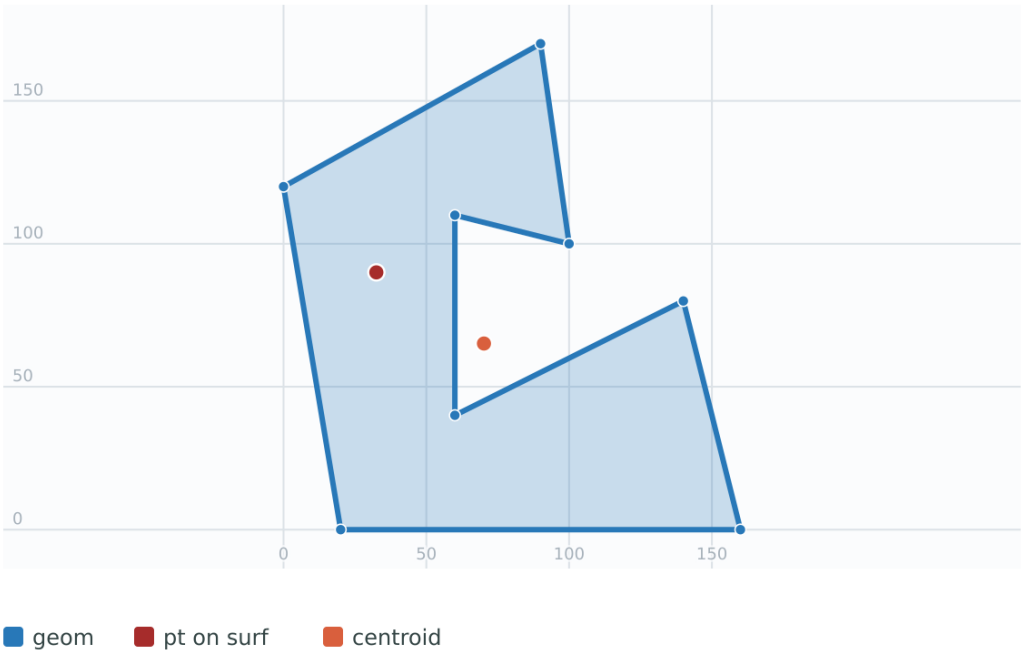


Exempel på detta: Resultatet av ST_PointOnSurface ligger garanterat inom polygoner, medan den punkt som beräknats av ST_Centroid kan ligga utanför.

Code

```
SELECT ST_PointOnSurface(geom) AS pt_on_surf, ST_Centroid(geom) AS centroid
    FROM (SELECT 'POLYGON ((130 120,120 190,30 140,50 20,190 20,
                        170 100,90 60,90 130,130 120))'::geometry AS geom) AS t;
```

Figure



Se även

ST_Centroid, ST_MaximumInscribedCircle

7.14.20 ST_Polygonize

ST_Polygonize — Beräknar en samling polygoner som bildas av linjerna i en uppsättning geometrier.

Synopsis

geometry **ST_Polygonize**(geometry set geomfield);
geometry **ST_Polygonize**(geometry[] geom_array);

Beskrivning

Skapar en GeometryCollection som innehåller de polygoner som bildas av linjeverket i en uppsättning geometrier. Om det inmatade linjeverket inte bildar några polygoner returneras en tom GeometryCollection.

Denna funktion skapar polygoner som täcker alla avgränsade områden. Om resultatet är avsett att bilda en giltig polygongeometri, använd ST_BuildArea för att förhindra att hål fylls.

Notera
Inmatningsgeometrin måste vara korrekt nodad för att denna funktion ska fungera korrekt. För att säkerställa att inmatningen är nodad använder du ST_Node på inmatningsgeometrin innan du polygoniserar.

Notera
GeometryCollections kan vara svåra att hantera med externa verktyg. Använd ST_Dump för att konvertera det polygoniserade resultatet till separata polygoner.

Utförs av GEOS-modulen.

Tillgänglighet: 1.0.0RC1

Exempel

Polygonize a set of input lines.

Code

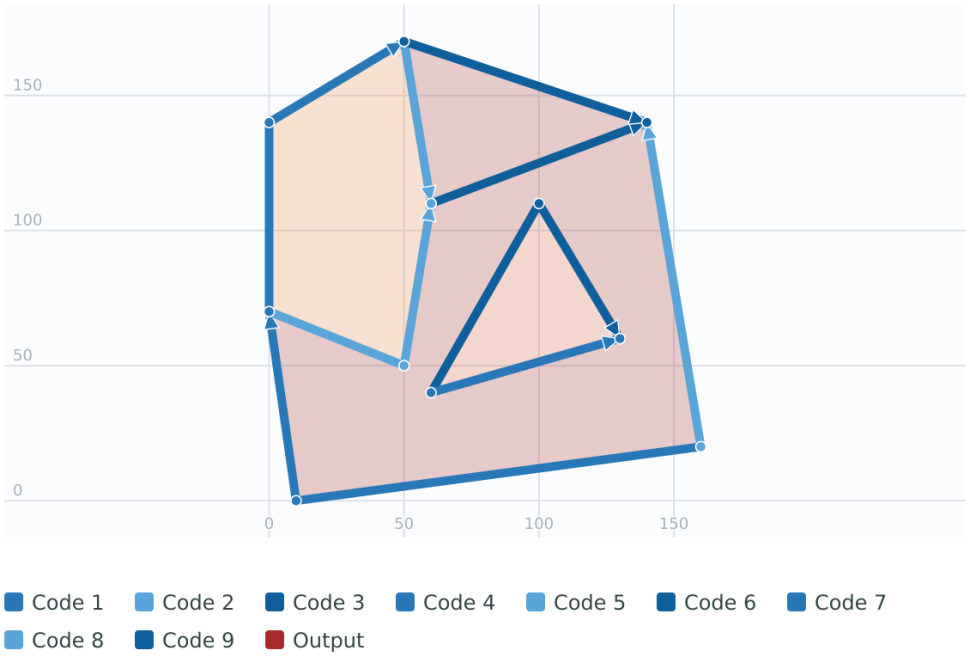
```
WITH data(geom) AS (
  VALUES
    ('LINESTRING (180 40,30 20,20 90)'::geometry),
    ('LINESTRING (180 40,160 160)'::geometry),
    ('LINESTRING (80 60,120 130,150 80)'::geometry),
    ('LINESTRING (80 60,150 80)'::geometry),
    ('LINESTRING (20 90,70 70,80 130)'::geometry),
    ('LINESTRING (80 130,160 160)'::geometry),
    ('LINESTRING (20 90,20 160,70 190)'::geometry),
    ('LINESTRING (70 190,80 130)'::geometry),
```

```
    ('LINESTRING (70 190,160 160)'::geometry)
)
SELECT ST_Polygonize(geom)
FROM data;
```

Output

```
GEOMETRYCOLLECTION(POLYGON((180 40,30 20,20 90,70 70,80 130,160 160,180 40),(150 80,120 130,80 60,150 80)), POLYGON ↩
    ((80 60,120 130,150 80,80 60)), POLYGON((80 130,70 70,20 90,20 160,70 190,80 130)), POLYGON((160 160,80 130,70  ↩
    190,160 160)))
```

Figure



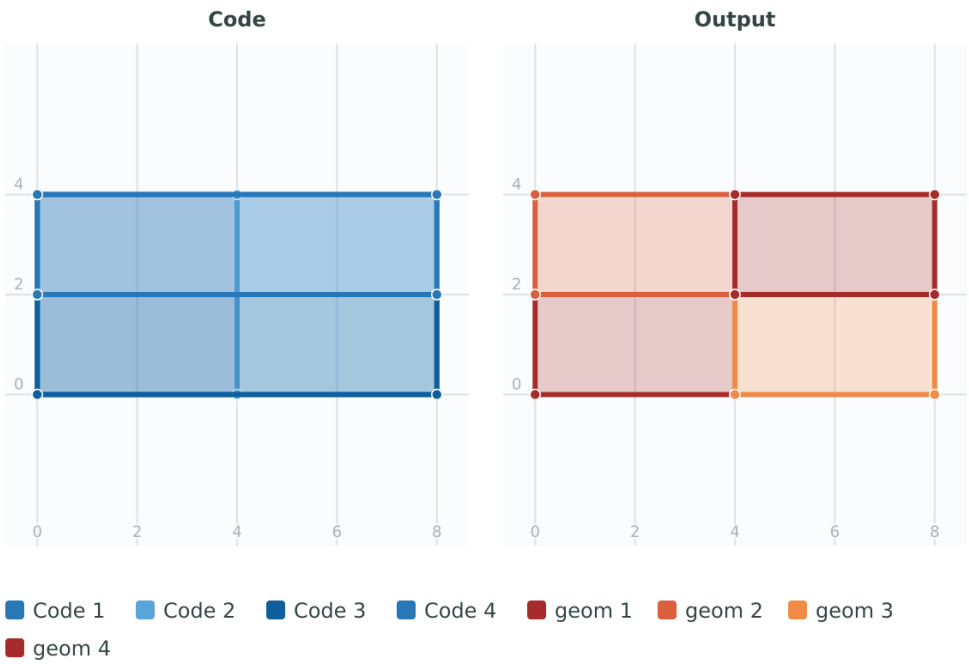
Polygonizing two polygon coverages and transferring attributes to the resultants:

Code

```
WITH poly_a(id, geom) AS (VALUES
  ('a1', 'POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0))'::geometry),
  ('a2', 'POLYGON ((4 0, 8 0, 8 4, 4 4, 4 0))'::geometry)
),
poly_b(id, geom) AS (VALUES
  ('b1', 'POLYGON ((0 0, 8 0, 8 2, 0 2, 0 0))'::geometry),
  ('b2', 'POLYGON ((0 2, 8 2, 8 4, 0 4, 0 2))'::geometry)
),
lines AS (
  SELECT ST_Boundary(geom) AS geom FROM poly_a
  UNION ALL
  SELECT ST_Boundary(geom) FROM poly_b
),
noded AS (
  SELECT ST_UnaryUnion(ST_Collect(geom)) AS geom FROM lines
),
cells AS (
  SELECT (ST_Dump(ST_Polygonize(geom))).geom AS geom FROM noded
)
SELECT a.id AS id_a,
       b.id AS id_b,
```

```
       c.geom AS geom
FROM cells AS c
LEFT JOIN poly_a AS a ON ST_Within(ST_PointOnSurface(c.geom), a.geom)
LEFT JOIN poly_b AS b ON ST_Within(ST_PointOnSurface(c.geom), b.geom)
WHERE a.id IS NOT NULL OR b.id IS NOT NULL
ORDER BY id_a, id_b;
```

Figure



Polygonisera en tabell med linestrings:

Code

```
SELECT ST_Polygonize(geom_4269) As geomtextrep
FROM (SELECT geom_4269 FROM ma.suffolk_edges) As foo;
```

Output

```
geomtextrep
-------------
 SRID=4269;GEOMETRYCOLLECTION(POLYGON((-71.040878 42.285678,-71.040943 42.2856,-71.04096  ↩
    42.285752,-71.040878 42.285678)),
 POLYGON((-71.17166 42.353675,-71.172026 42.354044,-71.17239 42.354358,-71.171794  ↩
    42.354971,-71.170511 42.354855,
 -71.17112 42.354238,-71.17166 42.353675)))
(1 row)
```

Use ST_Dump to dump out the polygonize geoms into individual polygons.

Code

```
SELECT (ST_Dump(t.polycoll)).geom AS geomtextrep
FROM (SELECT ST_Polygonize(geom_4269) AS polycoll
    FROM (SELECT geom_4269 FROM ma.suffolk_edges)
        As foo) AS t;
```

Output

```
geomtextrep
-------------
 SRID=4269;POLYGON((-71.040878 42.285678,-71.040943 42.2856,-71.04096 42.285752,
-71.040878 42.285678))
 SRID=4269;POLYGON((-71.17166 42.353675,-71.172026 42.354044,-71.17239 42.354358,
-71.171794 42.354971,-71.170511 42.354855,-71.17112 42.354238,-71.17166 42.353675))
(2 rows)
```

Se även

ST_BuildArea, ST_Boundary, ST_Collect, ST_Dump, ST_Node, ST_PointOnSurface, ST_UnaryUnion, ST_Within

7.14.21 ST_ReducePrecision

ST_ReducePrecision — Returnerar en giltig geometri med punkter som avrundats till en rutnätstolerans.

Synopsis

geometry **ST_ReducePrecision**(geometry g, float8 gridsize);

Beskrivning

Returnerar en giltig geometri där alla punkter har avrundats till den angivna rutnätstoleransen och där objekt under toleransen har tagits bort.

Till skillnad från ST_SnapToGrid kommer den returnerade geometrin att vara giltig, utan självskärande ringar eller kollapsade komponenter.

Precisionsreduktion kan användas för att:

- matcha koordinatprecisionen med datanoggrannheten
- minska antalet koordinater som behövs för att representera en geometri
- säkerställa giltig geometriutdata till format som använder lägre precision (t.ex. textformat som WKT, GeoJSON eller KML när antalet decimaler i utdata är begränsat).
- exportera giltig geometri till system som använder lägre eller begränsad precision (t.ex. SDE, Oracle toleransvärde)

Tillgänglighet: 3.1.0.

Kräver GEOS >= 3.9.0.

Exempel

Code

```
SELECT ST_ReducePrecision('POINT(1.412 19.323)', 0.1);
```

Output

```
POINT(1.4 19.3)
```

Figure

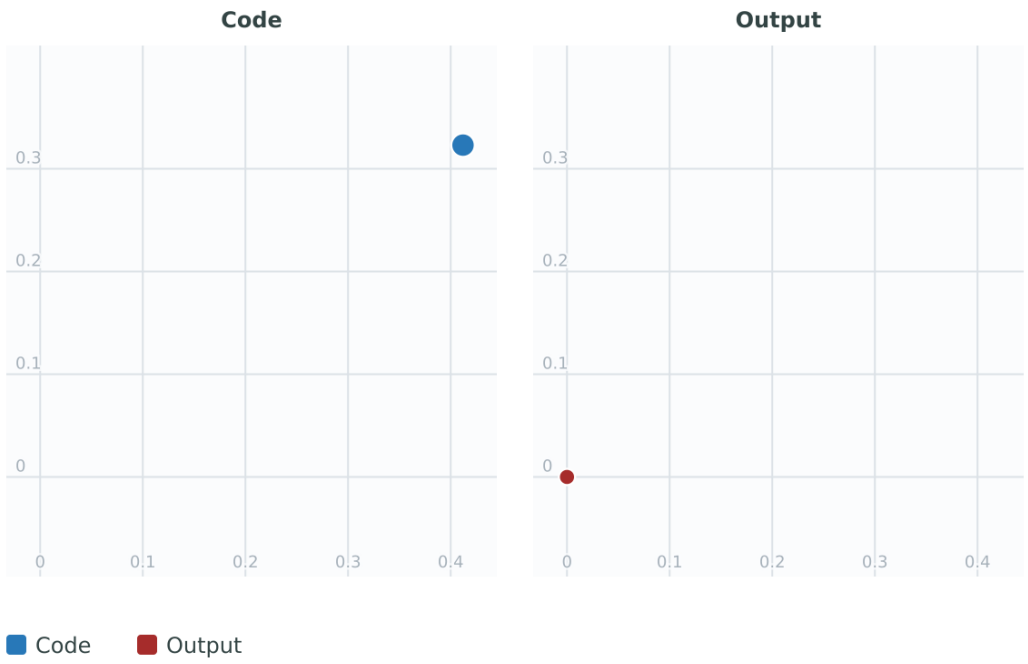
Code

```
SELECT ST_ReducePrecision('POINT(1.412 19.323)', 1.0);
```

Output

```
POINT(1 19)
```

Figure



Code

```
SELECT ST_ReducePrecision('POINT(1.412 19.323)', 10);
```

Output

```
POINT(0 20)
```

Figure

Precisionsreducering kan minska antalet toppar

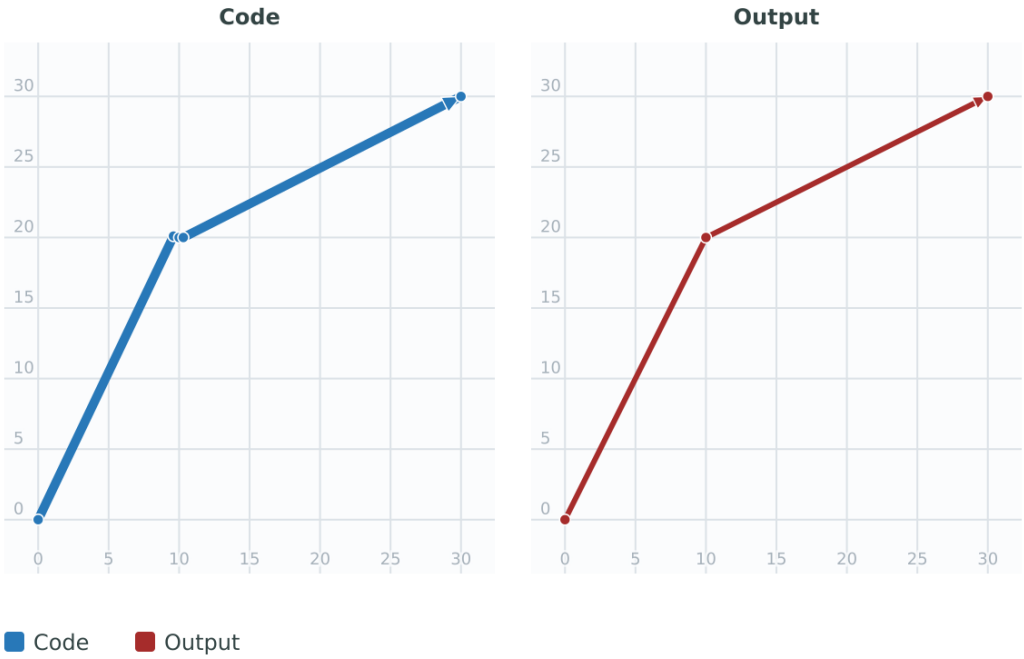
Code

```
SELECT ST_ReducePrecision('LINESTRING (10 10,19.6 30.1,20 30,20.3 30,40 40)', 1);
```

Output

```
LINESTRING (10 10, 20 30, 40 40)
```

Figure



Precisionsreducering delar upp polygoner om det behövs för att säkerställa validitet

Code

```
WITH data(geom) AS (
  VALUES ('POLYGON ((10 10,60 60.1,70 30,40 40,50 10,10 10))'::geometry)
)
SELECT geom AS input,
       ST_ReducePrecision(geom, 10) AS reduced
FROM data;
```

Output

```
-[ RECORD 1 ]-----------------------------------------------
input   | POLYGON((10 10,60 60.1,70 30,40 40,50 10,10 10))
reduced | MULTIPOLYGON(((10 10,40 40,50 10,10 10)),((40 40,60 60,70 30,40 40)))
```

Figure

Se även

ST_SnapToGrid, ST_Simplify, ST_SimplifyVW

7.14.22 ST_SharedPaths

ST_SharedPaths — Returnerar en samling som innehåller sökvägar som delas av de två inmatade linestrings/multilinestrings.

Synopsis

geometry **ST_SharedPaths**(geometry lineal1, geometry lineal2);

Beskrivning

Returnerar en samling som innehåller banor som delas av de två inmatade geometrierna. De som går i samma riktning finns i det första elementet i samlingen, de som går i motsatt riktning finns i det andra elementet. Själva banorna anges i den första geometrins riktning.

Utförs av GEOS-modulen.

Tillgänglighet: 2.0.0

Exempel: Hitta gemensamma vägar

Adjacent shared path segments may be returned as separate or merged LineStrings depending on the GEOS version.

Find the shared paths of a MultiLineString and a LineString.

Code

```
SELECT ST_SharedPaths(ST_GeomFromText('MULTILINESTRING((26 125,26 200,126 200,126 125,26 ↩
    125),
       (51 150,101 150,76 175,51 150))'),
   ST_GeomFromText('LINESTRING(151 100,126 156.25,126 125,101 150,76 175)')
   )
```

Output

```
GEOMETRYCOLLECTION(MULTILINESTRING((126 156.25,126 125),
(101 150,76 175)),MULTILINESTRING EMPTY)
```

Figure

Flip the LineString orientation to return the same paths in the opposite-direction member.

Code

```
SELECT ST_SharedPaths(ST_GeomFromText('LINESTRING(76 175,101 150,126 125,126 156.25,151 ↩
    100)'),
   ST_GeomFromText('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),
       (51 150,101 150,76 175,51 150))')
    )
```

Output

```
GEOMETRYCOLLECTION(MULTILINESTRING EMPTY,
MULTILINESTRING((76 175,101 150),(126 125,126 156.25)))
```

Figure

Se även

ST_Dump, ST_GeometryN, ST_NumGeometries

7.14.23 ST_Simplify

ST_Simplify — Returnerar en förenklad representation av en geometri med hjälp av Douglas-Peucker-algoritmen.

Synopsis

geometry **ST_Simplify**(geometry geom, float tolerance);
geometry **ST_Simplify**(geometry geom, float tolerance, boolean preserveCollapsed);

Beskrivning

Beräknar en förenklad representation av en geometri med hjälp av Douglas-Peucker-algoritmen. `Förenkling` är ett avståndsvärde, i enheterna för indata SRS. Förenklingen tar bort hörn som ligger inom toleransavståndet för det förenklade linjearbetet. Resultatet kanske inte är giltigt även om indata är det.

Funktionen kan anropas med alla typer av geometri (inklusive GeometryCollections), men endast linje- och polygonelement förenklas. Ändpunkter för linjär geometri bevaras.

Flaggan `preserveCollapsed` behåller små geometrier som annars skulle tas bort vid den angivna toleransen. Om t.ex. en 1 m lång linje förenklas med en tolerans på 10 m kommer linjen inte att försvinna när `preserveCollapsed` är true. Denna flagga är användbar för renderingsändamål för att förhindra att mycket små objekt försvinner från en karta.

Notera
Den returnerade geometrin kan förlora sin enkelhet (se ST_IsSimple), topologin kanske inte bevaras och polygonala resultat kan vara ogiltiga (se ST_IsValid). Använd ST_SimplifyPreserveTopology för att bevara topologin och säkerställa giltigheten.

Notera
Denna funktion bevarar inte gränser som delas mellan polygoner. Använd ST_CoverageSimplify om detta krävs.

Tillgänglighet: 1.2.2

Exempel

A circle simplified with a medium tolerance becomes an octagon, and with a large tolerance disappears.

Code

```
SELECT ST_NPoints(geom) AS np_before,
     ST_NPoints(ST_Simplify(geom, 0.1)) AS np01_notbadcircle,
     ST_NPoints(ST_Simplify(geom, 0.5)) AS np05_notquitecircle,
     ST_NPoints(ST_Simplify(geom, 1)) AS np1_octagon,
     ST_NPoints(ST_Simplify(geom, 10)) AS np10_gone,
     (ST_Simplify(geom, 100) is null) AS  np100_geometrygoesaway
FROM (SELECT ST_Buffer('POINT(1 3)', 10, 12) As geom) AS t;
```

Output

```
-[ RECORD 1 ]------------
np_before                | 49
np01_notbadcircle        | 33
np05_notquitecircle      | 17
np1_octagon              | 9
np10_gone                | null
np100_geometrygoesaway   | t
```

Förenkling av en uppsättning linjer. Linjerna kan korsa varandra efter förenklingen.

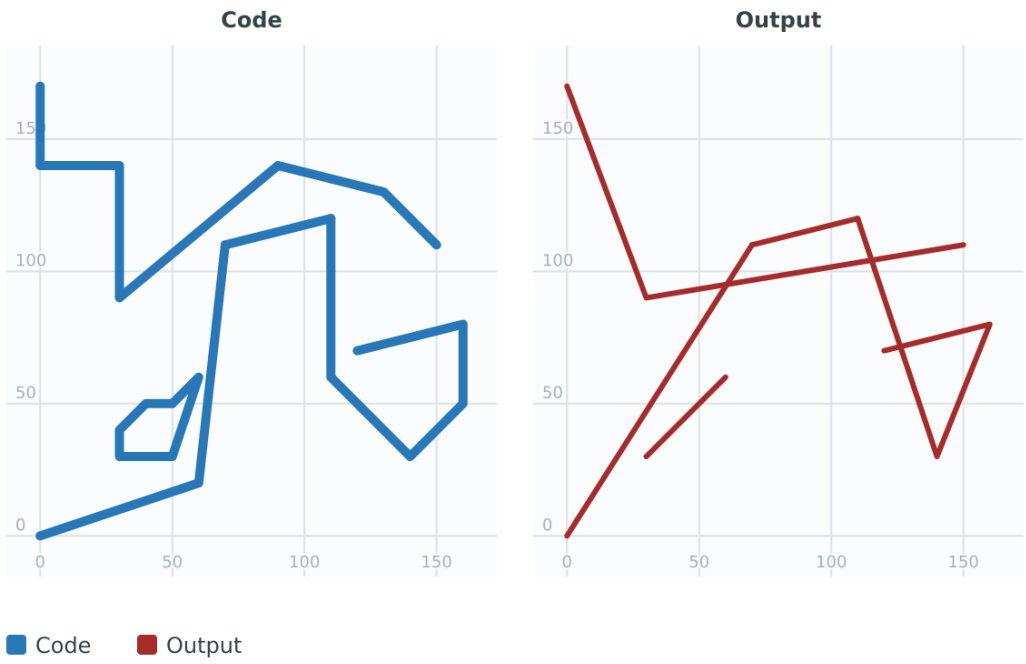
Code

```
SELECT ST_Simplify('MULTILINESTRING ((20 180,20 150,50 150,50 100,110 150,150 140,170 120) ↩
    ,(20 10,80 30,90 120),(90 120,130 130),(130 130,130 70,160 40,180 60,180 90,140 80),(50 ↩
    40,70 40,80 70,70 60,60 60,50 50,50 40))'::geometry,
40);
```

Output

```
MULTILINESTRING((20 180,50 100,170 120),(20 10,90 120),(90 120,130 130),(130 130,160 40,180 90,140 80),(50 40,80 70,50 ↩
     40))
```

Figure



Förenkla en multipolygon. Polygonala resultat kan vara ogiltiga.

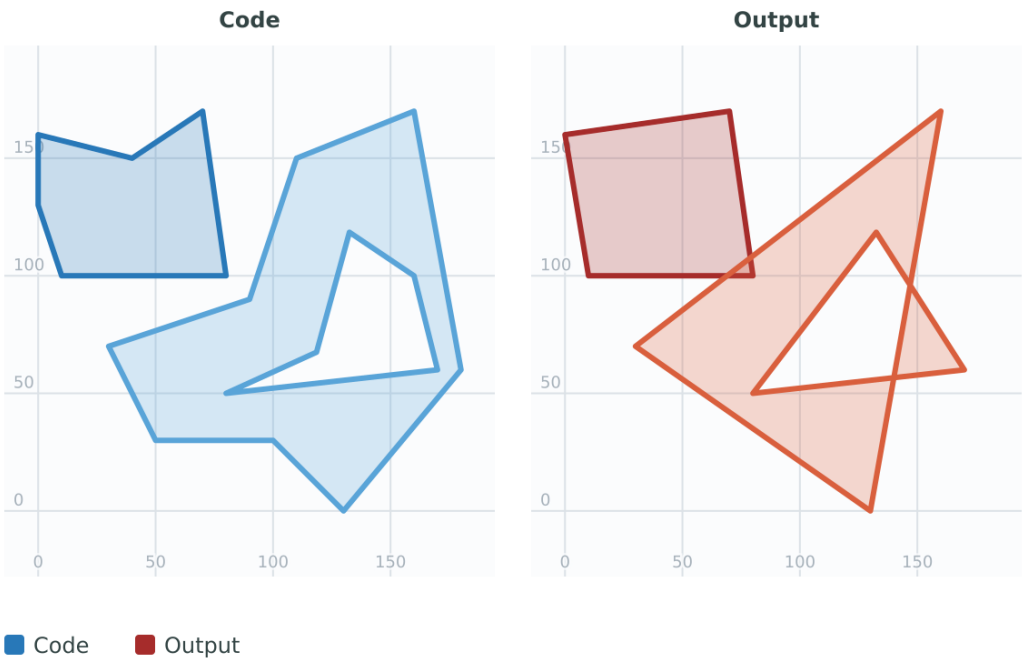
Code

```
SELECT ST_Simplify('MULTIPOLYGON (((90 110,80 180,50 160,10 170,10 140,20 110,90 110)),((40 ↩
    80,100 100,120 160,170 180,190 70,140 10,110 40,60 40,40 80),(180 70,170 110,142.5 ↩
   128.5,128.5 77.5,90 60,180 70)))'::geometry,
40);
```

Output

```
MULTIPOLYGON(((90 110,80 180,10 170,20 110,90 110)),((40 80,170 180,140 10,40 80),(180 70,142.5 128.5,90 60,180 70)))
```

Figure



Se även

ST_IsSimple, ST_SimplifyPreserveTopology, ST_SimplifyVW, ST_CoverageSimplify, Topologi ST_Simplify

7.14.24 ST_SimplifyPreserveTopology

ST_SimplifyPreserveTopology — Returnerar en förenklad och giltig representation av en geometri med hjälp av Douglas-Peucker-algoritmen.

Synopsis

geometry **ST_SimplifyPreserveTopology**(geometry geom, float tolerance);

Beskrivning

Beräknar en förenklad representation av en geometri med hjälp av en variant av Douglas-Peucker-algoritmen som begränsar förenklingen för att säkerställa att resultatet har samma topologi som indata. `Förenklingstoleransen` är ett avståndsvärde, i enheterna för indatans SRS. Förenklingen tar bort toppar som ligger inom toleransavståndet för det förenklade linjeverket, så länge som topologin bevaras. Resultatet kommer att vara giltigt och enkelt om indata är det.

Funktionen kan anropas med alla typer av geometri (inklusive GeometryCollections), men endast linje- och polygonelement förenklas. För polygonala indata kommer resultatet att ha samma antal ringar (skal och hål), och ringarna kommer inte att korsa varandra. Ringarnas ändpunkter kan förenklas. För linjära indata kommer resultatet att ha samma antal linjer, och linjerna kommer inte att korsa varandra om de inte gjorde det i den ursprungliga geometrin. Slutpunkterna för linjär geometri bevaras.

Notera
Denna funktion bevarar inte gränser som delas mellan polygoner. Använd ST_CoverageSimplify om detta krävs.

Utförs av GEOS-modulen.

Tillgänglighet: 1.3.3

Exempel

For the same example as ST_Simplify, ST_SimplifyPreserveTopology prevents the geometry from disappearing.

Code

```
SELECT  ST_NPoints(geom) AS np_before,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.1)) AS np01_notbadcircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.5)) AS np05_notquitecircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 1))   AS np1_octagon,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 10))  AS np10_minimal,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 100)) AS np100_minimal
FROM (SELECT ST_Buffer('POINT(1 3)', 10, 12) AS geom) AS t;
```

Output

```
np_before | np01_notbadcircle | np05_notquitecircle | np1_octagon | np10_minimal |  ↩
    np100_minimal
-----------+-------------------+---------------------+-------------+--------------+--------------- ←

        49 |                33 |                  17 |           9 |            4 |         ↩
         4
```

Förenkla en uppsättning linjer och bevara topologin för icke korsande linjer.

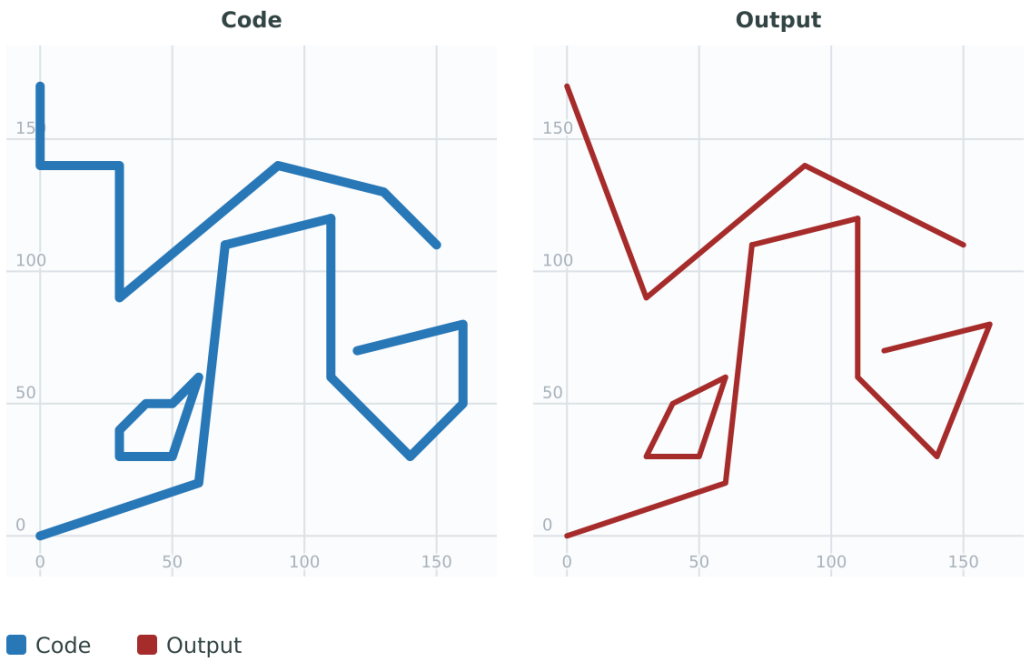
Code

```
SELECT ST_SimplifyPreserveTopology('MULTILINESTRING ((20 180,20 150,50 150,50 100,110  ↩
    150,150 140,170 120),(20 10,80 30,90 120),(90 120,130 130),(130 130,130 70,160 40,180  ↩
    60,180 90,140 80),(50 40,70 40,80 70,70 60,60 60,50 50,50 40))'::geometry,
40);
```

Output

```
MULTILINESTRING((20 180,50 100,110 150,170 120),(20 10,80 30,90 120),(90 120,130 130),(130 130,130 70,160 40,180  ↩
    90,140 80),(50 40,70 40,80 70,60 60,50 40))
```

Figure



Förenkling av en multipolygon med bibehållen topologi för skal och hål.

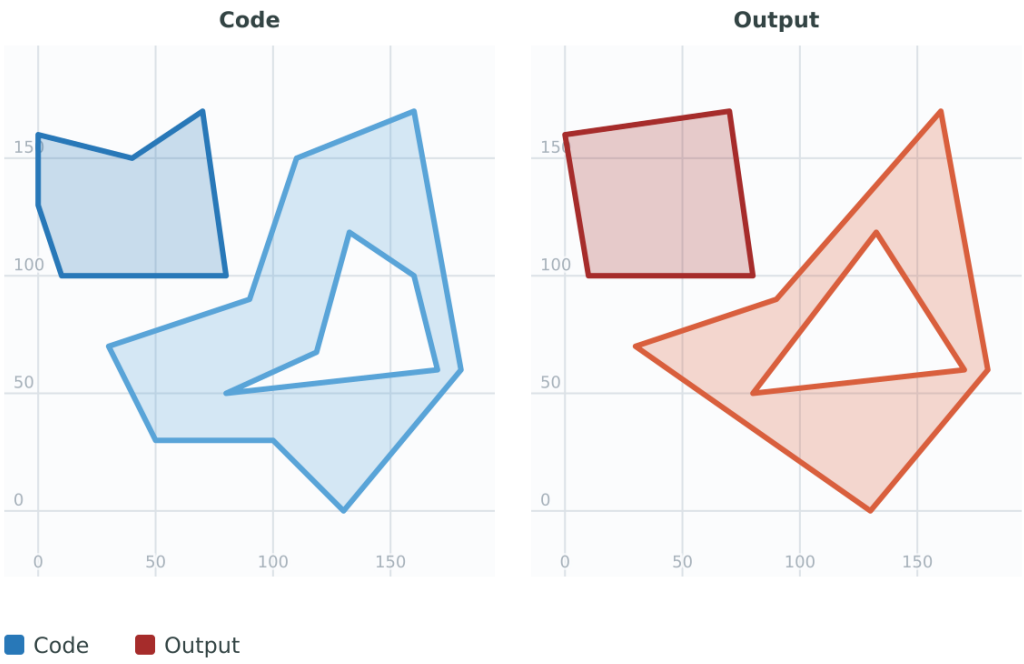
Code

```
SELECT ST_SimplifyPreserveTopology('MULTIPOLYGON (((90 110,80 180,50 160,10 170,10 140,20 ↩
    110,90 110)),((40 80,100 100,120 160,170 180,190 70,140 10,110 40,60 40,40 80),(180 ↩
    70,170 110,142.5 128.5,128.5 77.5,90 60,180 70)))'::geometry,
40);
```

Output

```
MULTIPOLYGON(((90 110,80 180,10 170,20 110,90 110)),((40 80,100 100,170 180,190 70,140 10,40 80),(180 70,142.5 ↩
    128.5,90 60,180 70)))
```

Figure



Se även

ST_Simplify, ST_SimplifyVW, ST_CoverageSimplify

7.14.25 ST_SimplifyPolygonHull

ST_SimplifyPolygonHull — Beräknar ett förenklat topologibevarande yttre eller inre skrov av en polygonal geometri.

Synopsis

geometry **ST_SimplifyPolygonHull**(geometry param_geom, float vertex_fraction, boolean is_outer = true);

Beskrivning

Beräknar ett förenklat topologibevarande yttre eller inre skrov av en polygonal geometri. Ett yttre skrov täcker helt inmatningsgeometrin. Ett inre skrov täcks helt av indatageometrin. Resultatet är en polygonal geometri som bildas av en delmängd av de ingående hörnen. MultiPolygoner och hål hanteras och ger ett resultat med samma struktur som indata.

Minskningen av antalet toppar styrs av parametern `vertex_fraction`, som är ett tal i intervallet 0 till 1. Lägre värden ger enklare resultat, med mindre antal vertex och mindre konkavitet. För både yttre och inre skrov ger en vertexfraktion på 1,0 den ursprungliga geometrin. För yttre skrov ger ett värde på 0,0 det konvexa skrovet (för en enda polygon); för inre skrov ger det en triangel.

Förenklingsprocessen fungerar genom att gradvis ta bort konkava hörn som innehåller minst yta, tills målet för antalet vertex har uppnåtts. Det förhindrar att kanter korsar varandra, så resultatet är alltid en giltig polygonal geometri.

För att få bättre resultat med geometrier som innehåller relativt långa linjesegment kan det vara nödvändigt att segmentisera”indata, enligt nedan.

Utförs av GEOS-modulen.

Tillgänglighet: 3.3.0.

Kräver GEOS >= 3.11.0.

Exempel

Outer hull of a Polygon.

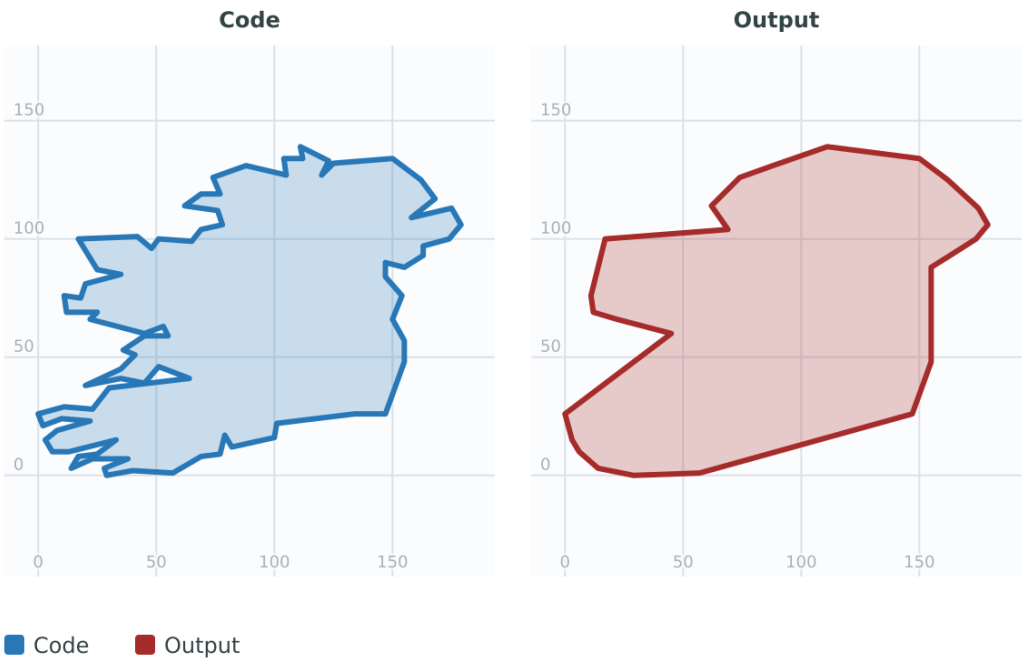
Code

```
SELECT ST_SimplifyPolygonHull(
  'POLYGON ((131 158,136 163,161 165,173 156,179 148,169 140,186 144,190 137,185 131,174 ↩
   128,174 124,166 119,158 121,158 115,165 107,161 97,166 88,166 79,158 57,145 57,112 ↩
   53,111 47,93 43,90 48,88 40,80 39,68 32,51 33,40 31,39 34,49 38,34 38,25 34,28 39,36 ↩
   40,44 46,24 41,17 41,14 46,19 50,33 54,21 55,13 52,11 57,22 60,34 59,41 68,75 72,62 ↩
   77,56 70,46 72,31 69,46 76,52 82,47 84,56 90,66 90,64 94,56 91,33 97,36 100,23 100,22 ↩
   107,29 106,31 112,46 116,36 118,28 131,53 132,59 127,62 131,76 130,80 135,89 137,87 ↩
   143,73 145,80 150,88 150,85 157,99 162,116 158,115 165,123 165,122 170,134 164,131 158)) ↩
   '::geometry,
  0.3
);
```

Output

```
POLYGON((161 165,173 156,186 144,190 137,185 131,174 124,166 119,166 79,158 57,68 32,40 31,25 34,17 41,14 46,11 57,56 ↩
    91,33 97,23 100,22 107,28 131,80 135,73 145,85 157,99 162,122 170,161 165))
```

Figure



Inner hull of a Polygon.

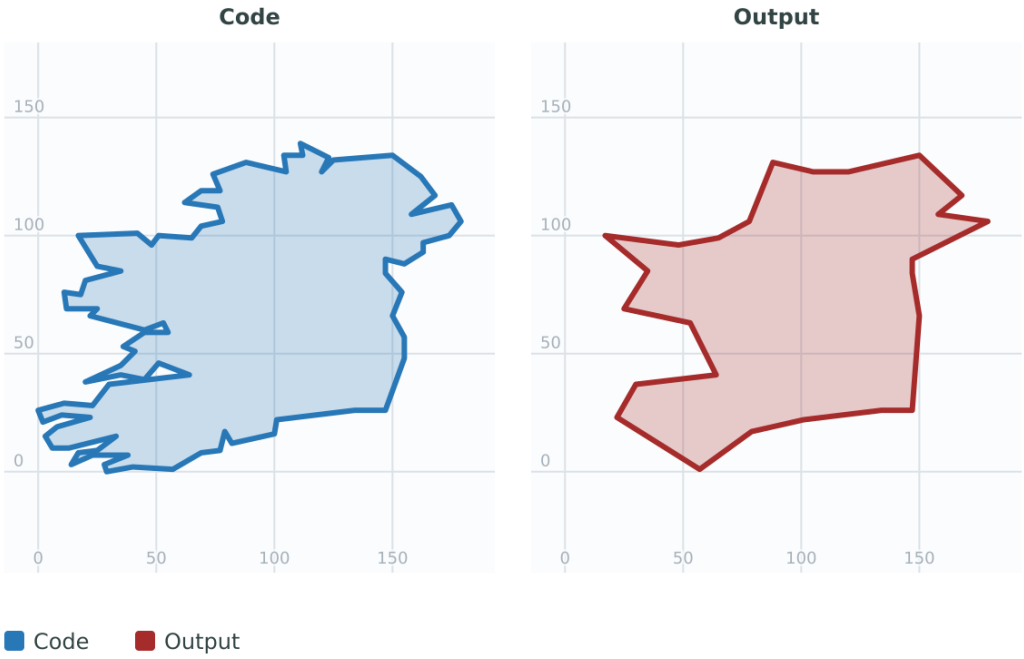
Code

```
SELECT ST_SimplifyPolygonHull(
  'POLYGON ((131 158,136 163,161 165,173 156,179 148,169 140,186 144,190 137,185 131,174 128,174 124,166 119,158 121,158 115,165 107,161 97,166 88,166 79,158 57,145 57,112 53,111 47,93 43,90 48,88 40,80 39,68 32,51 33,40 31,39 34,49 38,34 38,25 34,28 39,36 40,44 46,24 41,17 41,14 46,19 50,33 54,21 55,13 52,11 57,22 60,34 59,41 68,75 72,62 77,56 70,46 72,31 69,46 76,52 82,47 84,56 90,66 90,64 94,56 91,33 97,36 100,23 100,22 107,29 106,31 112,46 116,36 118,28 131,53 132,59 127,62 131,76 130,80 135,89 137,87 143,73 145,80 150,88 150,85 157,99 162,116 158,115 165,123 165,122 170,134 164,131 158))
  '::geometry,
  0.3, false
);
```

Output

```
POLYGON((131 158,116 158,99 162,89 137,76 130,59 127,28 131,46 116,36 100,64 94,75 72,41 68,33 54,68 32,90 48,112 53,145 57,158 57,161 97,158 115,158 121,190 137,169 140,179 148,161 165,131 158))
```

Figure

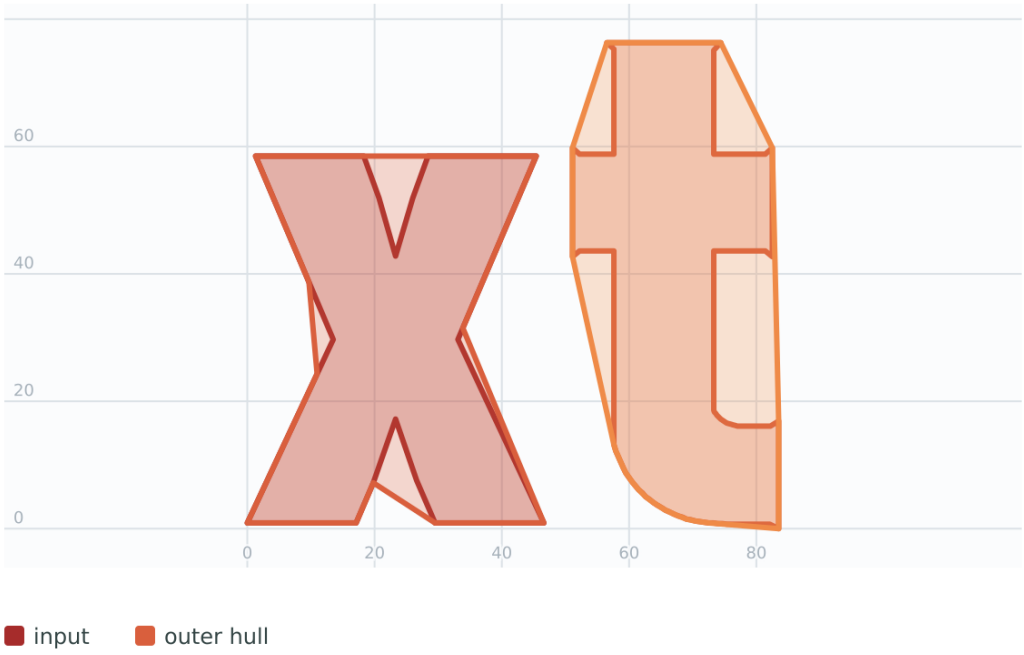


Outer hull simplification of a MultiPolygon, with segmentization.

Code

```
WITH letters AS (
  SELECT ST_Letters('xt') AS geom
)
SELECT geom AS input,
  ST_SimplifyPolygonHull(ST_Segmentize(geom, 2.0), 0.1) AS outer_hull
FROM letters;
```

Figure



Se även

ST_ConvexHull, ST_SimplifyVW, ST_ConcaveHull, ST_Segmentize

7.14.26 ST_SimplifyVW

ST_SimplifyVW — Returnerar en förenklad representation av en geometri med hjälp av Visvalingam-Whyatt-algoritmen

Synopsis

geometry **ST_SimplifyVW**(geometry geom, float tolerance);

Beskrivning

Returns a simplified representation of a geometry using the Visvalingam-Whyatt algorithm. The simplification `tolerance` is an area value, in squared units of the input SRS. Simplification removes vertices which form ”cornersåith area less than the tolerance. The result may not be valid even if the input is.

Funktionen kan anropas med alla typer av geometri (inklusive GeometryCollections), men endast linje- och polygonelement förenklas. Ändpunkter för linjär geometri bevaras.

Notera
Den returnerade geometrin kan förlora sin enkelhet (se ST_IsSimple), topologin kanske inte bevaras och polygonala resultat kan vara ogiltiga (se ST_IsValid). Använd ST_SimplifyPreserveTopology för att bevara topologin och säkerställa giltigheten. ST_CoverageSimplify bevarar också topologin och giltigheten.

This is a general-purpose simplification function. It may remove narrow polygon spikes when their effective area is below the tolerance, but it is not a dedicated spike-removal or polygon-repair operation. Use ST_IsValid to check the result and ST_MakeValid to repair invalid polygonal output if needed.

Denna funktion bevarar inte gränser som delas mellan polygoner. Använd ST_CoverageSimplify om detta krävs.

Denna funktion hanterar 3D och den tredje dimensionen kommer att påverka resultatet.

Tillgänglighet: 2.2.0

Exempel

This polygon has a narrow spike whose effective area is small enough to be removed by a tolerance of 20 square units.

Code

```
SELECT geom AS original, ST_SimplifyVW(geom, 20) AS simplified
FROM (SELECT 'POLYGON((0 0,10 0,10 10,5.2 10,5 16,4.8 10,0 10,0 0))'::geometry AS geom) AS ↩
    t;
```

Output

```
POLYGON((0 0,10 0,10 10,5.2 10,5 16,4.8 10,0 10,0 0)) | POLYGON((0 0,10 0,10 10,0 10,0 0))
```

Figure

A LineString is simplified with a minimum-area tolerance of 30. For a linear simplification using ST_Simplify, a comparable tolerance would be a length in input-SRS units, so the value here is roughly that linear tolerance squared.

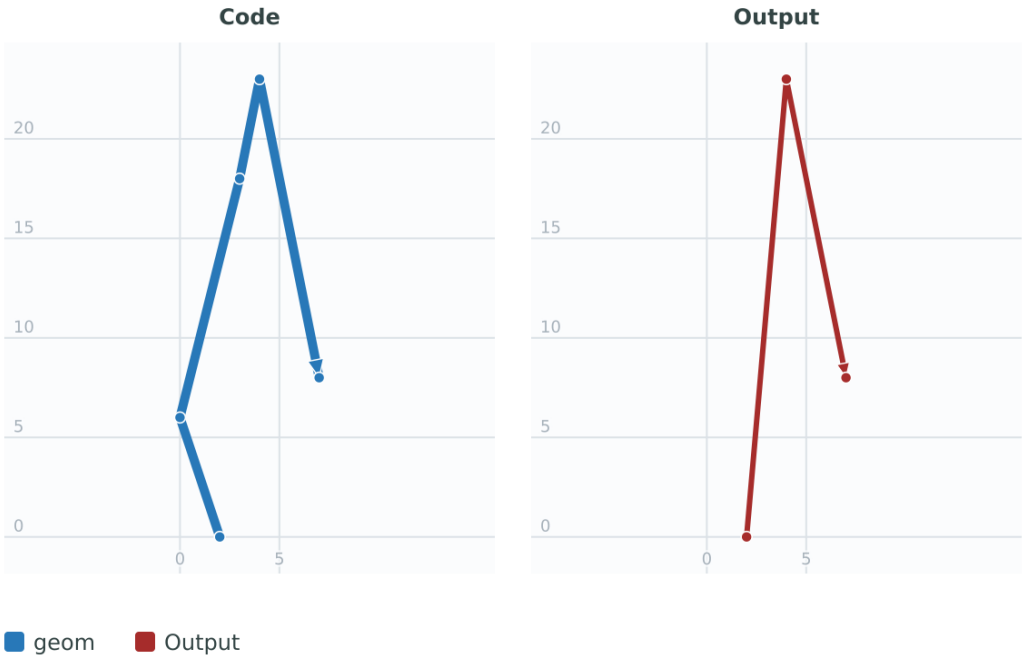
Code

```
SELECT ST_SimplifyVW(geom, 30) simplified
  FROM (SELECT 'LINESTRING(5 2,3 8,6 20,7 25,10 10)'::geometry AS geom) AS t;
```

Output

```
LINESTRING(5 2,7 25,10 10)
```

Figure



Förenkling av en linje.

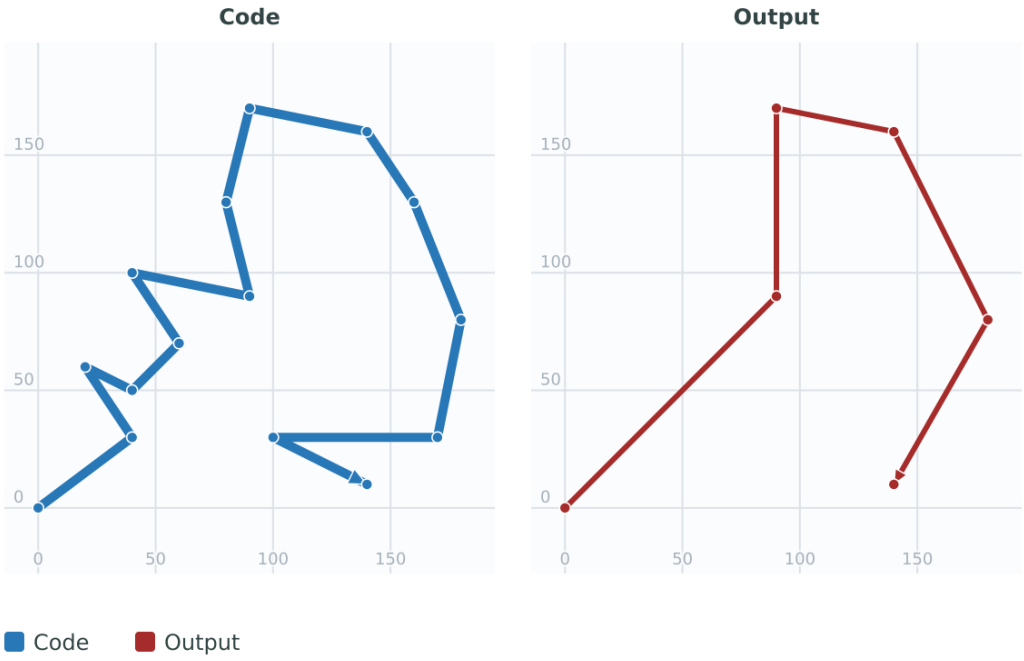
Code

```
SELECT ST_SimplifyVW('LINESTRING (10 10,50 40,30 70,50 60,70 80,50 110,100 100,90 140,100 ↩
    180,150 170,170 140,190 90,180 40,110 40,150 20)'::geometry,
1600);
```

Output

```
LINESTRING(10 10,100 100,100 180,150 170,190 90,150 20)
```

Figure



Förenkling av en polygon.

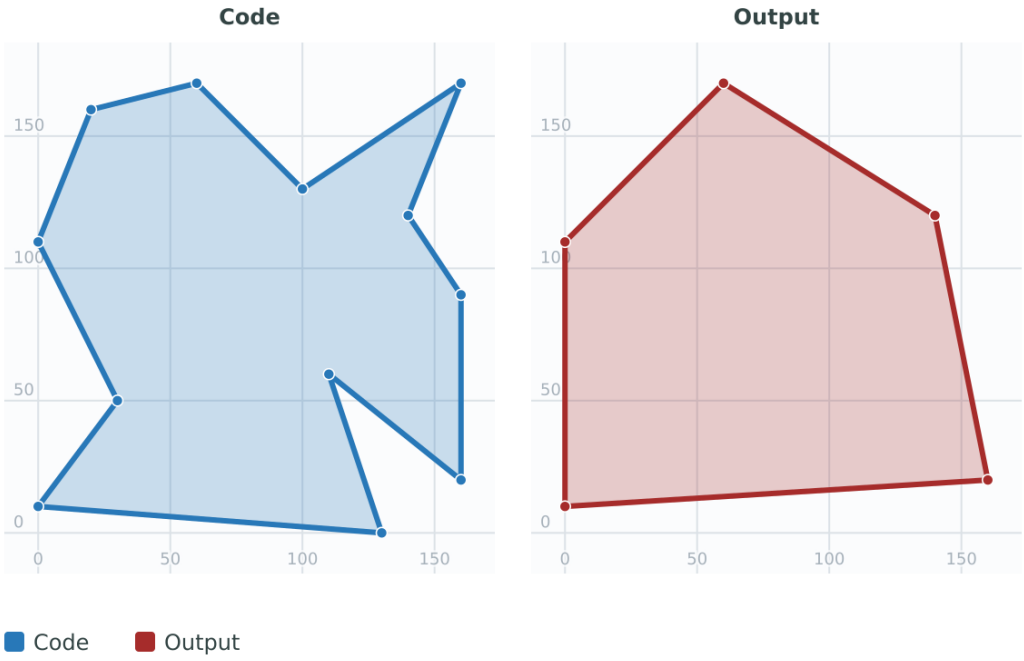
Code

```
SELECT ST_SimplifyVW('POLYGON ((20 20,50 60,20 120,40 170,80 180,120 140,180 180,160 ↩
    130,180 100,180 30,130 70,150 10,20 20))'::geometry,
1600);
```

Output

```
POLYGON((20 20,20 120,80 180,160 130,180 30,20 20))
```

Figure



Se även

ST_SetEffectiveArea, ST_Simplify, ST_SimplifyPreserveTopology, ST_CoverageSimplify, Topologi ST_Simpli

7.14.27 ST_SetEffectiveArea

ST_SetEffectiveArea — Ställer in den effektiva ytan för varje toppunkt med hjälp av Visvalingam-Whyatt-algoritmen.

Synopsis

geometry **ST_SetEffectiveArea**(geometry geom, float threshold = 0, integer set_area = 1);

Beskrivning

Ställer in den effektiva ytan för varje vertex med hjälp av Visvalingam-Whyatt-algoritmen. Den effektiva arean lagras som M-värdet för vertexen. Om den valfria parametern thresholdänvänds, returneras en förenklad geometri som endast innehåller toppar med en effektiv area som är större än eller lika med tröskelvärdet.

Denna funktion kan användas för förenkling på serversidan när ett tröskelvärde anges. Ett annat alternativ är att använda ett tröskelvärde på noll. I detta fall returneras den fullständiga geometrin med effektiva ytor som M-värden, som kan användas av klienten för att förenkla mycket snabbt.

Kommer faktiskt bara att göra något med (multi)linjer och (multi)polygoner, men du kan säkert anropa den med alla typer av geometri. Eftersom förenklingen sker objekt för objekt kan du också mata in en GeometryCollection till den här funktionen.

Notera
Observera att den returnerade geometrin kan förlora sin enkelhet (se ST_IsSimple)

Notera
Observera att topologin kanske inte bevaras och kan resultera i ogiltiga geometrier. Använd (se ST_SimplifyPreserveTopology) för att bevara topologin.

Notera
Utdatageometrin kommer att förlora all tidigare information om M-värdena

Notera
Denna funktion hanterar 3D och den tredje dimensionen kommer att påverka den effektiva ytan

Tillgänglighet: 2.2.0

Exempel

Beräkning av den effektiva ytan för en LineString. Eftersom vi använder ett tröskelvärde på noll returneras alla hörn i indatageometrin.

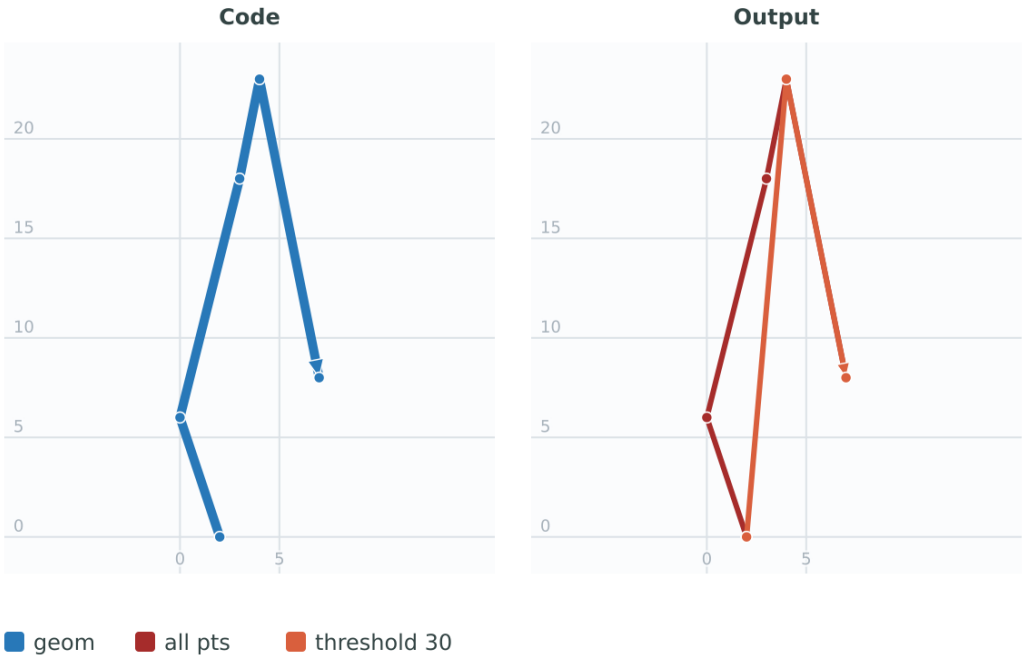
Code

```
SELECT ST_SetEffectiveArea(geom) AS all_pts,
       ST_SetEffectiveArea(geom, 30) AS threshold_30
FROM (SELECT  'LINESTRING(5 2,3 8,6 20,7 25,10 10)'::geometry geom) As foo;
```

Output

```
-[ RECORD 1 ]------------------
all_pts      | LINESTRING M (5 2 3.4e+38,3 8 29,6 20 1.5,7 25 49.5,10 10 3.4e+38)
threshold_30 | LINESTRING M (5 2 3.4e+38,7 25 49.5,10 10 3.4e+38)
```

Figure



Se även

ST_SimplifyVW

7.14.28 ST_TriangulatePolygon

ST_TriangulatePolygon — Beräknar den begränsade Delaunay-trianguleringen av polygoner

Synopsis

geometry **ST_TriangulatePolygon**(geometry geom);

Beskrivning

Beräknar den begränsade Delaunay-trianguleringen av polygoner. Stöd finns för hål och multipolygoner.

Den begränsade Delaunay-trianguleringenäv en polygon är en uppsättning trianglar som bildas från polygonens hörn och täcker den exakt, med den maximala totala inre vinkeln över alla möjliga trianguleringar. Den ger den bästa kvalitetenpå trianguleringen av polygonen.

Tillgänglighet: 3.3.0.

Kräver GEOS >= 3.11.0.

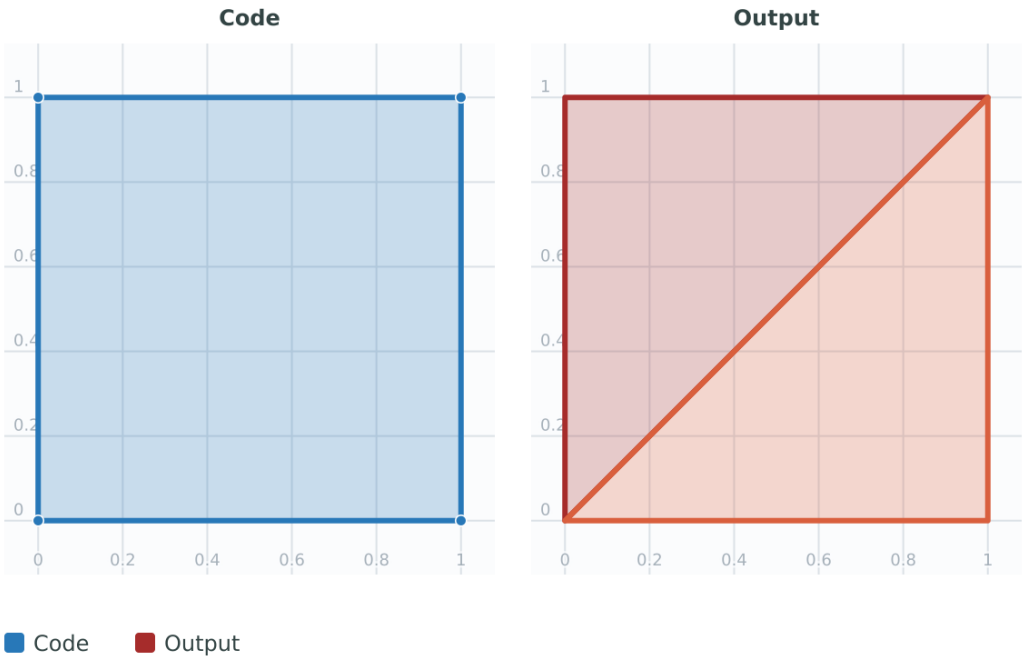
Exempel

Triangulering av en kvadrat.

Code

```
SELECT ST_TriangulatePolygon('POLYGON((0 0,0 1,1 1,1 0,0 0))');
```

Figure

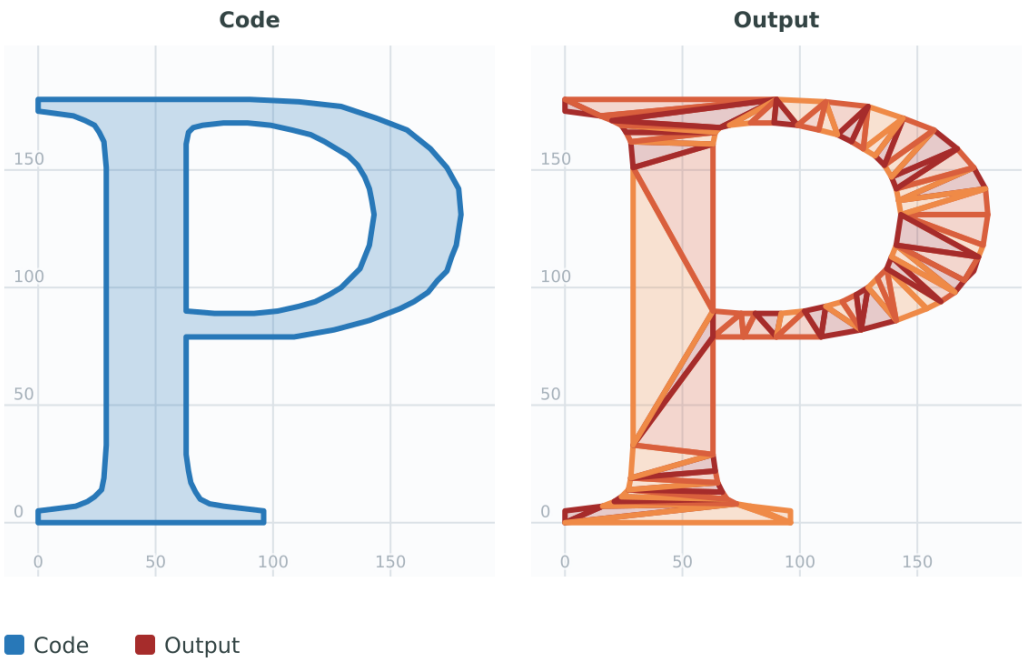


Triangulering av bokstaven P.

Code

```
SELECT ST_TriangulatePolygon('POLYGON ((26 17,31 19,34 21,37 24,38 29,39 43,39 161,38 ↩
    172,36 176,34 179,30 181,25 183,10 185,10 190,100 190,121 189,139 187,154 182,167 ↩
    177,177 169,184 161,189 152,190 141,188 128,186 123,184 117,180 113,176 108,170 104,164 ↩
    101,151 96,136 92,119 89,100 89,86 89,73 89,73 39,74 32,75 27,77 23,79 20,83 18,89 ↩
    17,106 15,106 10,10 10,10 15,26 17),(152 147,151 152,149 157,146 162,142 166,137 169,132 ↩
     172,126 175,118 177,109 179,99 180,89 180,80 179,76 178,74 176,73 171,73 100,85 99,91 ↩
    99,102 99,112 100,121 102,128 104,134 107,139 110,143 114,147 118,149 123,151 128,153 ↩
    141,152 147))'
);
```

Figure

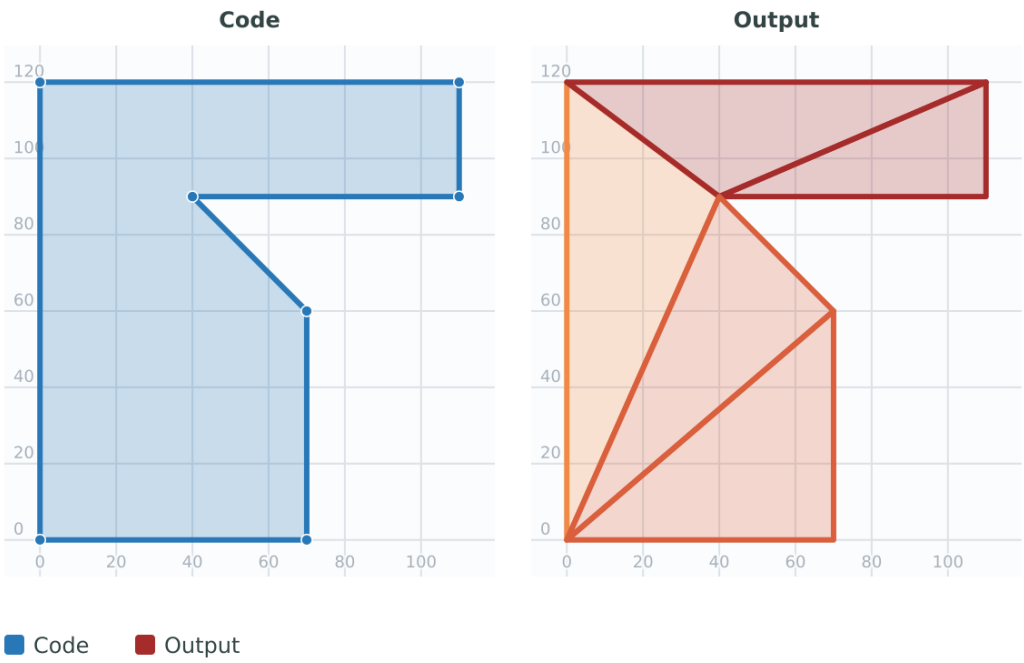


Triangulation of the polygon used in the CG_Tesselate example.

Code

```
SELECT ST_TriangulatePolygon('POLYGON (( 10 190,10 70,80 70,80 130,50 160,120 160,120 ↩
    190,10 190 ))'::geometry
);
```

Figure



Se även

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, CG_Tesselate

7.14.29 ST_VoronoiLines

ST_VoronoiLines — Returnerar gränserna för Voronoi-diagrammet för geometrins hörnpunkter.

Synopsis

geometry **ST_VoronoiLines**(geometry geom , float8 tolerance = 0.0 , geometry extend_to = NULL);

Beskrivning

Beräknar ett tvådimensionellt Voronoi-diagram från hörnen i den angivna geometrin och returnerar gränserna mellan cellerna i diagrammet som en MultiLineString. Returnerar null om indatageometrin är null. Returnerar en tom geometrisamling om indatageometrin endast innehåller ett vertex. Returnerar en tom geometrisamling om `extend_to-kuvertet` har noll area.

Valfria parametrar:

- `tolerans`: Det avstånd inom vilket hörn anses vara likvärdiga. Algoritmens robusthet kan förbättras genom att ange ett toleransavstånd som inte är noll. (standard = 0,0)
- `extend_to`: Om detta anges utökas diagrammet så att det täcker den medföljande geometrins omslutningsyta, om den inte är mindre än standardomslutningsytan (standard = NULL, standardomslutningsytan är inmatningsytans begränsningsbox utökad med ca 50%).

Utförs av GEOS-modulen.

Tillgänglighet: 2.3.0

Exempel

Voronoi diagram lines with a tolerance of 30 units.

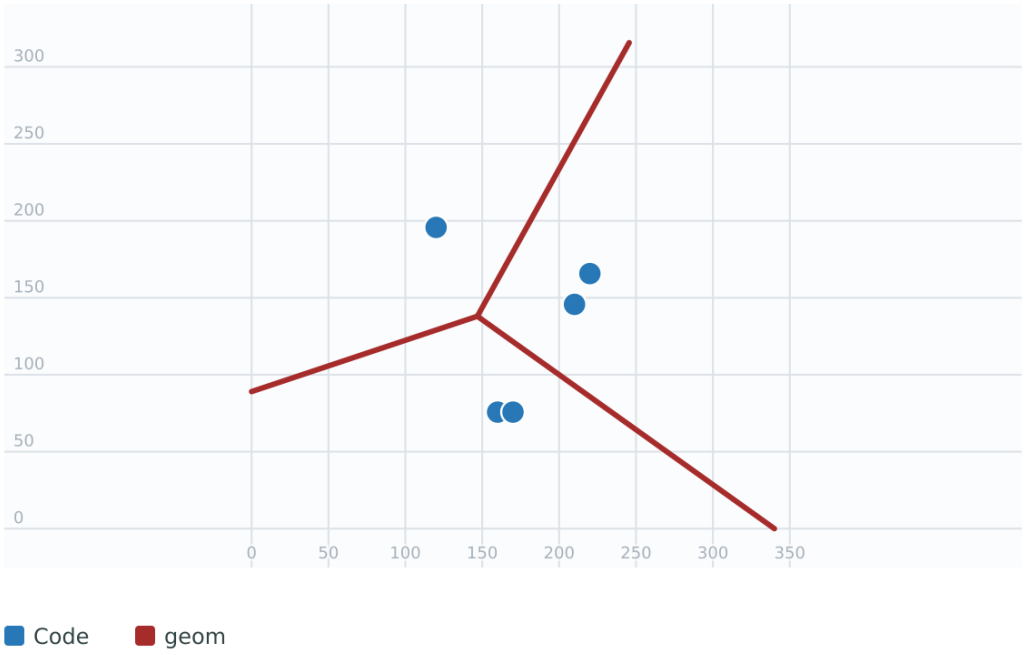
Code

```
SELECT ST_VoronoiLines('MULTIPOINT (50 30,60 30,100 100,10 150,110 120)'::geometry,
            30) AS geom;
```

Output

```
MULTILINESTRING((135.555555555556 270,36.8181818181818 92.2727272727273),(36.8181818181818 92.2727272727273,-110 ↩
    43.3333333333333),(230 -45.7142857142858,36.8181818181818 92.2727272727273))
```

Figure



Se även

ST_DelaunayTriangles, ST_VoronoiPolygons

7.14.30 ST_VoronoiPolygons

ST_VoronoiPolygons — Returnerar cellerna i Voronoi-diagrammet för hörnen i en geometri.

Synopsis

geometry **ST_VoronoiPolygons**(geometry geom , float8 tolerance = 0.0 , geometry extend_to = NULL);

Beskrivning

Beräknar ett tvådimensionellt Voronoi-diagram från hörnen i den angivna geometrin. Resultatet är en GEOMETRYCOLLECTION av POLYGONER som täcker ett kuvert som är större än omfattningen av de ingående hörnen. Returnerar null om indatageometrin är null. Returnerar en tom geometrisamling om indatageometrin endast innehåller en vertex. Returnerar en tom geometrisamling om `extend_to-kuvertet` har noll area.

Valfria parametrar:

- `tolerans`: Det avstånd inom vilket hörn anses vara likvärdiga. Algoritmens robusthet kan förbättras genom att ange ett toleransavstånd som inte är noll. (standard = 0,0)
- `extend_to`: Om detta anges utökas diagrammet så att det täcker den medföljande geometrins omslutningsyta, om den inte är mindre än standardomslutningsytan (standard = NULL, standardomslutningsytan är inmatningsytans begränsningsbox utökad med ca 50%).

Utförs av GEOS-modulen.

Tillgänglighet: 2.3.0

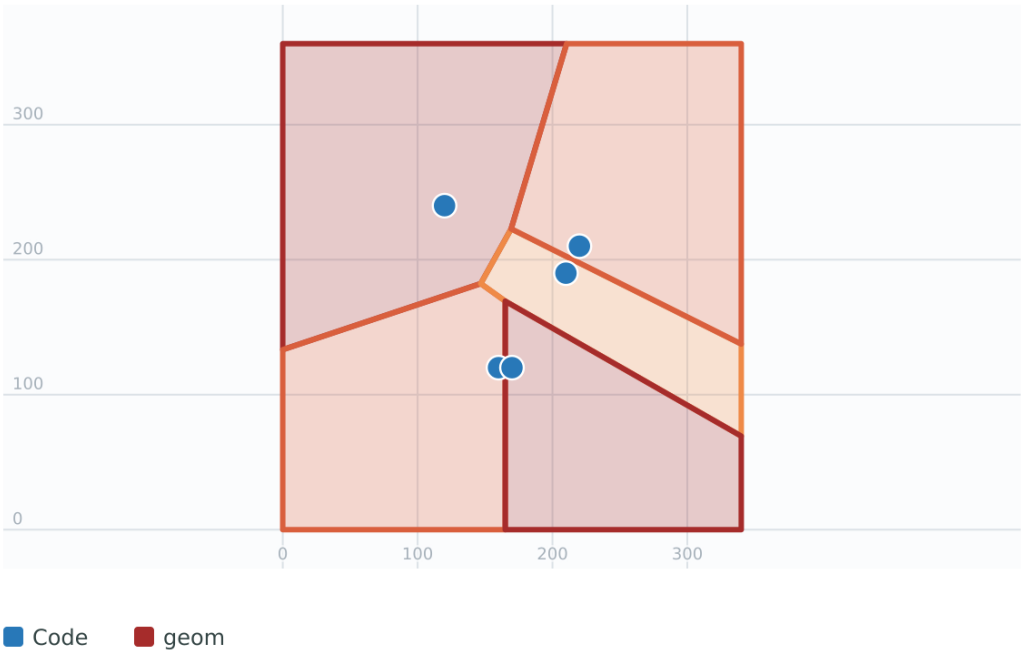
Exempel

Points overlaid on top of the Voronoi diagram.

Code

```
SELECT ST_VoronoiPolygons('MULTIPOINT (50 30,60 30,100 100,10 150,110 120)'::geometry
            ) AS geom;
```

Figure

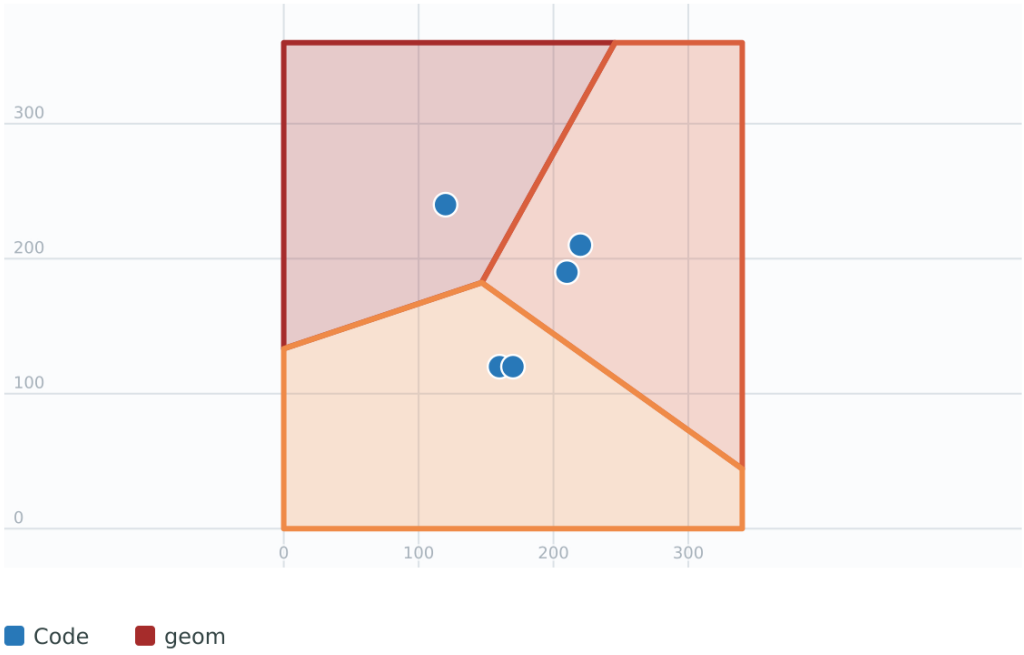


Voronoi diagram with a tolerance of 30 units.

Code

```
SELECT ST_VoronoiPolygons('MULTIPOINT (50 30,60 30,100 100,10 150,110 120)'::geometry,
            30) AS geom;
```

Figure



Se även

ST_DelaunayTriangles, ST_VoronoiLines

7.15 Täckning

7.15.1 ST_CoverageInvalidEdges

ST_CoverageInvalidEdges — Fönsterfunktion som hittar platser där polygonerna inte bildar en giltig täckning.

Synopsis

geometry **ST_CoverageInvalidEdges**(geometry winset geom, float8 tolerance = 0);

Beskrivning

En fönsterfunktion som kontrollerar om polygonerna i fönsterpartitionen bildar en giltig polygontäckning. Den returnerar linjära indikatorer som visar var ogiltiga kanter (om sådana finns) finns i varje polygon.

En uppsättning giltiga polygoner är en giltig täckning om följande villkor uppfylls:

- **Icke-överlappande** - polygoner överlappar inte varandra (deras inre delar skär inte varandra)
- **Edge-Matched** - toppar längs delade kanter är identiska

Som en fönsterfunktion returneras ett värde för varje polygon som matas in. För polygoner som bryter mot ett eller flera av giltighetsvillkoren är returvärdet en MULTILINESTRING som innehåller de problematiska kanterna. Täckningsgilla polygoner returnerar värdet NULL. Icke-polygonala eller tomma geometrier producerar också NULL-värden.

Villkoren tillåter att en giltig täckning innehåller hål (mellanrum mellan polygoner), så länge som de omgivande polygonerna är kantmatchade. Mycket smala luckor är dock ofta inte önskvärda. Om *toleransparametern* anges med ett avstånd som inte är noll, kommer kanter som bildar smalare luckor också att returneras som ogiltiga.

De polygoner som kontrolleras för täckningsgiltighet måste också vara giltiga geometrier. Detta kan kontrolleras med ST_IsValid.

Tillgänglighet: 3.4.0

Kräver GEOS >= 3.12.0

Exempel

Invalid edges caused by overlap and non-matching vertices.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((10 190,30 160,40 110,100 70,120 10,10 10,10 190))'::geometry),
  (2, 'POLYGON ((100 190,10 190,30 160,40 110,50 80,74 110.5,100 130,140 120,140 160,100 ↩
    190))'::geometry),
  (3, 'POLYGON ((140 190,190 190,190 80,140 80,140 190))'::geometry),
  (4, 'POLYGON ((180 40,120 10,100 70,140 80,190 80,180 40))'::geometry)
)
SELECT id, ST_CoverageInvalidEdges(geom) OVER ()
  FROM coverage;
```

Figure

This example tests the entire table for coverage validity.

Code

```
SELECT true = ALL (
    SELECT ST_CoverageInvalidEdges(geom) OVER () IS NULL
    FROM coverage
    );
```

Se även

ST_IsValid, ST_CoverageUnion, ST_CoverageClean, ST_CoverageSimplify

7.15.2 ST_CoverageClean

ST_CoverageClean — Beräknar en ren (kantmatchad, icke-överlappande, gap-cleared) polygontäckning, givet en icke ren indata.

Synopsis

geometry **ST_CoverageClean**(geometry winset geom, float8 gapMaximumWidth = 0, float8 snappingDistance = -1, text overlapMergeStrategy = 'MERGE_LONGEST_BORDER');

Beskrivning

A window function which adjusts the edges of a set of valid polygonal geometries to produce a clean coverage. Cleaning involves:

- snapping vertices and edges to remove small discrepancies and ensure common edges are identically noded
- merging overlaps into a parent polygon
- merging narrow gaps into adjacent polygons

gapMaximumWidth controls which gaps between polygons are merged. Gaps with width <= this distance are merged into an adjacent polygon.

snappingDistance controls snapping of vertices and edges. The default (-1) automatically determines a snapping distance based on the input extent. Set to 0.0 to turn off snapping.

overlapMergeStrategy specifies how overlaps are merged into a parent polygon:

- MERGE_LONGEST_BORDER - merges into polygon with longest common border
- MERGE_MAX_AREA - merges into polygon with maximum area
- MERGE_MIN_AREA - merges into polygon with minimum area
- MERGE_MIN_INDEX - merges into polygon with smallest input index (defined by order of input polygons)

The result is a clean polygonal coverage that will pass validation by ST_CoverageInvalidEdges and can be input to coverage processing functions.

Notera

To aid in determining a maximum gap width, gaps can be computed by cleaning with `gapMaximumWidth = > 0` and using ST_CoverageUnion to union the result coverage. Holes in the union correspond to gaps in the original dataset. Gap widths can be measured by extracting the holes as polygons and running ST_MaximumInscribedCircle on them; the gap width is twice the computed radius.

Tillgänglighet: 3.6.0 - kräver GEOS >= 3.14.0

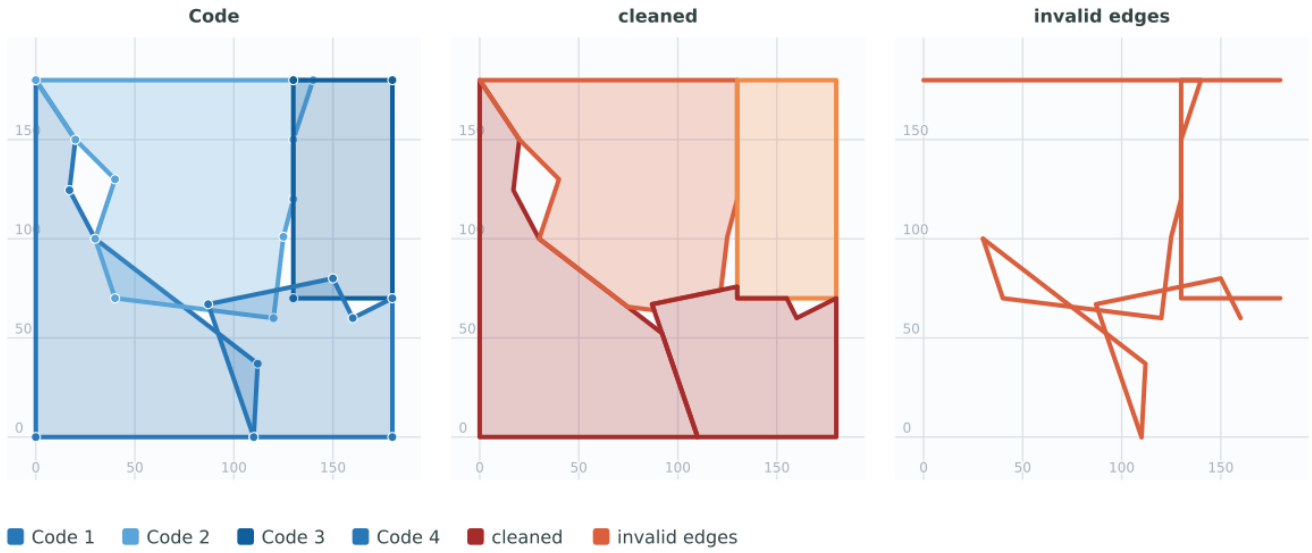
Exempel

Clean a coverage containing overlaps and narrow gaps, and compare the cleaned polygons with the original invalid edges.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((10 190,30 160,27 134.5,40 110,122 47,120 10,10 10,10 190))'::geometry),
  (2, 'POLYGON ((150 190,10 190,30 160,50 140,40 110,50 80,130 70,135 111,140 130,140 ↩
    160,150 190))'::geometry),
  (3, 'POLYGON ((140 190,190 190,190 80,140 80,140 190))'::geometry),
  (4, 'POLYGON ((190 10,120 10,97 77,160 90,170 70,190 80,190 10))'::geometry)
), processed AS (
  SELECT id,
         ST_CoverageClean(geom, 1) OVER () AS cleaned,
         ST_CoverageInvalidEdges(geom) OVER () AS invalid_edges
  FROM coverage
)
SELECT ST_Collect(cleaned ORDER BY id) AS cleaned,
       ST_Collect(invalid_edges ORDER BY id) AS invalid_edges
FROM processed;
```

Figure



Se även

ST_CoverageInvalidEdges, ST_CoverageUnion, ST_CoverageSimplify, ST_MaximumInscribedCircle

7.15.3 ST_CoverageSimplify

ST_CoverageSimplify — Fönsterfunktion som förenklar kanterna på en polygonal täckning.

Synopsis

geometry **ST_CoverageSimplify**(geometry winset geom, float8 tolerance, boolean simplifyBoundary = true);

Beskrivning

En fönsterfunktion som förenklar kanterna på polygoner i en polygonal täckning. Förenklingen bevarar täckningens topologi. Detta innebär att de förenklade polygonerna är konsekventa längs delade kanter och fortfarande utgör en giltig täckning.

Vid förenklingen används en variant av Visvalingam-Whyatt-algoritmen. *`Toleransparametern`* har avståndsenheter och är ungefär lika med kvadratroten av de triangulära områden som ska förenklas.

Om du bara vill förenkla täckningens ”inre”kanter (de som delas av två polygoner) ska du ange parametern *`simplifyBoundary`* till false.

Use this function to simplify adjacent polygon layers when shared boundaries must remain edge-matched. For a single geometry where only self-intersection and ring validity matter, see ST_SimplifyPreserv

Notera
Om indata inte är en giltig täckning kan det förekomma oväntade artefakter i utdata (t.ex. gränssnitt eller separerade gränser som verkade vara delade). Använd ST_CoverageInvalidEdges för att avgöra om en täckning är giltig.

Tillgänglighet: 3.4.0

Kräver GEOS >= 3.12.0

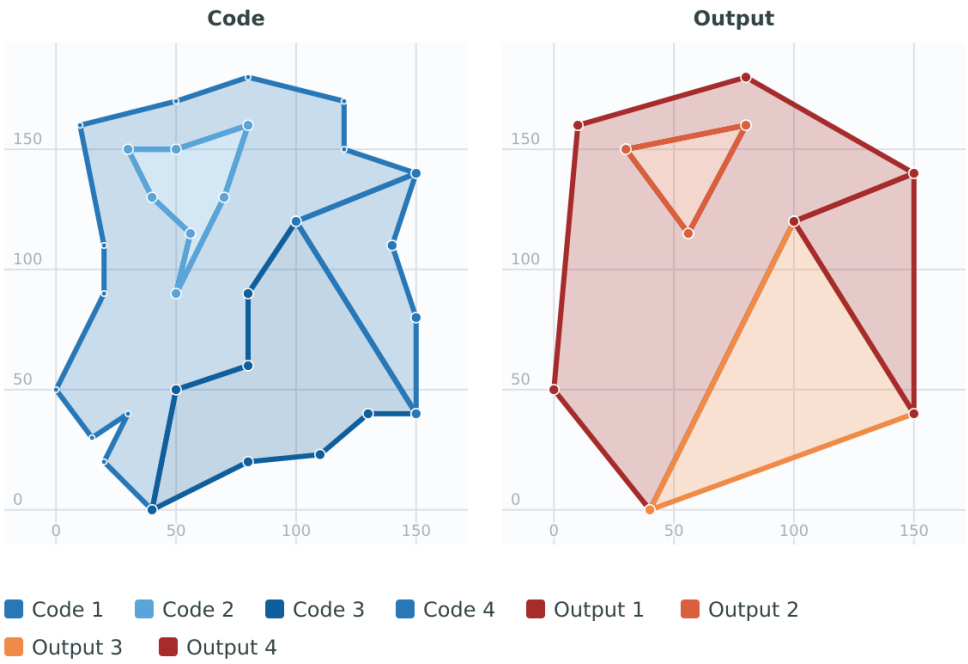
Exempel

Simplify a polygonal coverage while preserving shared boundaries.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((160 150,110 130,90 100,90 70,60 60,50 10,30 30,40 50,25 40,10 60,30 100,30 ↩
     120,20 170,60 180,90 190,130 180,130 160,160 150),(40 160,50 140,66 125,60 100,80  ↩
    140,90 170,60 160,40 160))'::geometry),
  (2, 'POLYGON ((40 160,60 160,90 170,80 140,60 100,66 125,50 140,40 160))'::geometry),
  (3, 'POLYGON ((110 130,160 50,140 50,120 33,90 30,50 10,60 60,90 70,90 100,110 130))':: ↩
    geometry),
  (4, 'POLYGON ((160 150,150 120,160 90,160 50,110 130,160 150))'::geometry)
)
SELECT id, ST_CoverageSimplify(geom, 30) OVER ()
  FROM coverage;
```

Figure



Se även

ST_CoverageInvalidEdges, ST_CoverageUnion, ST_CoverageClean

7.15.4 ST_CoverageEdges

ST_CoverageEdges — Computes the unique edges of a polygonal coverage.

Synopsis

geometry **ST_CoverageEdges**(geometry geom, integer edgetype = 0);
geometry **ST_CoverageEdges**(geometry[] geoms, integer edgetype = 0);

Beskrivning

Returns a MultiLineString representing the unique edges of a polygonal coverage. A polygonal coverage is a set of non-overlapping polygons where adjacent polygons have matching vertices along shared edges.

The *edgetype* parameter can be used to select which edges are returned:

- **0 (ALL)** - all unique edges (default)
- **1 (EXTERIOR)** - only exterior edges (non-shared)
- **2 (INTERIOR)** - only interior edges (shared)

Availability: 3.7.0

Requires GEOS >= 3.15.0

Exempel

This example returns all unique edges of a coverage.

Code

```
WITH coverage(geom) AS (
  VALUES (
    'GEOMETRYCOLLECTION(
      POLYGON((0 0,10 0,10 10,0 10,0 0)),
      POLYGON((10 0,20 0,20 10,10 10,10 0))
    )'::geometry
  )
)
SELECT geom AS input_coverage,
       ST_CoverageEdges(geom) AS all_edges
FROM coverage;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------------------------------------------------
input_coverage | GEOMETRYCOLLECTION(POLYGON((0 0,10 0,10 10,0 10,0 0)),POLYGON((10 0,20 0,20 10,10 10,10 0)))
all_edges      | MULTILINESTRING((10 0,10 10),(10 10,0 10,0 0,10 0),(10 0,20 0,20 10,10 10))
```

This example returns only the interior shared edges of a coverage.

Code

```
WITH coverage(geom) AS (
  VALUES (
    'GEOMETRYCOLLECTION(
      POLYGON((0 0,10 0,10 10,0 10,0 0)),
      POLYGON((10 0,20 0,20 10,10 10,10 0))
    )'::geometry
  )
)
SELECT geom AS input_coverage,
       ST_CoverageEdges(geom, 2) AS shared_edges
FROM coverage;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------------------------------------------------
input_coverage | GEOMETRYCOLLECTION(POLYGON((0 0,10 0,10 10,0 10,0 0)),POLYGON((10 0,20 0,20 10,10 10,10 0)))
shared_edges   | MULTILINESTRING((10 0,10 10))
```

Se även

ST_CoverageInvalidEdges, ST_CoverageSimplify, ST_CoverageUnion

7.15.5 ST_CoverageUnion

ST_CoverageUnion — Beräknar unionen av en uppsättning polygoner som bildar en täckning genom att ta bort gemensamma kanter.

Synopsis

geometry **ST_CoverageUnion**(geometry set geom);

Beskrivning

En aggregatfunktion som förenar en uppsättning polygoner som bildar en polygontäckning. Resultatet är en polygonal geometri som täcker samma område som täckningen. Denna funktion ger samma resultat som ST_Union, men använder täckningsstrukturen för att beräkna sammanslagningen mycket snabbare.

Notera
Om indata inte är en giltig täckning kan det uppstå oväntade artefakter i utdata (t.ex. polygoner som inte är sammansmälta eller överlappar varandra). Använd ST_CoverageInvalidEdges för att avgöra om en täckning är giltig.

Tillgänglighet: 3.4.0 - kräver GEOS >= 3.8.0

Exempel

Union a polygonal coverage.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((10 10,10 150,80 190,110 150,90 110,40 110,50 60,10 10))'::geometry),
  (2, 'POLYGON ((120 10,10 10,50 60,100 70,120 10))'::geometry),
  (3, 'POLYGON ((140 80,120 10,100 70,40 110,90 110,110 150,140 80))'::geometry),
  (4, 'POLYGON ((140 190,120 170,140 130,160 150,140 190))'::geometry),
  (5, 'POLYGON ((180 160,170 140,140 130,160 150,140 190,180 160))'::geometry)
)
SELECT ST_CoverageUnion(geom) AS wkt
  FROM coverage;
```

Figure

Se även

ST_CoverageInvalidEdges, ST_CoverageSimplify, ST_CoverageClean, ST_Union

7.16 Affina transformationer

7.16.1 ST_Affine

ST_Affine — Tillämpa en 3D-affin transformation på en geometri.

Synopsis

geometry **ST_Affine**(geometry geomA, float a, float b, float c, float d, float e, float f, float g, float h, float i, float xoff, float yoff, float zoff);
geometry **ST_Affine**(geometry geomA, float a, float b, float d, float e, float xoff, float yoff);

Beskrivning

Tillämpar en 3D-affin transformation på geometrin för att göra saker som att translatera, rotera och skala i ett steg.

Version 1: Anropet

Code

```
ST_Affine(geom, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff
```

) representerar transformationsmatrisen

Code

```
/ a b c xoff \
| d e f yoff | | g h i zoff
| g h i zoff |
\ 0 0 0 1 /
```

och topparna transformeras på följande sätt:

Code

```
x' = a*x + b*y + c*z + xoff
y' = d*x + e*y + f*z + yoff
z' = g*x + h*y + i*z + zoff
```

Alla översättnings- / skalningsfunktioner nedan uttrycks via en sådan affin transformation.

Version 2: Tillämpar en 2D-affin transformation på geometrin. Anropet

Code

```
ST_Affine(geom, a, b, d, e, xoff, yoff)
```

representerar transformationsmatrisen

Code

```
/ a b 0 xoff \ / a b xoff \
| d e 0 yoff | rsp.  | d e yoff | | d e yoff
| 0 0 1 0 | \ 0 0 1 /
\ 0 0 0 1 /
```

och topparna transformeras på följande sätt:

Code

```
x' = a*x + b*y + xoff
y' = d*x + e*y + yoff
z' = z
```

Denna metod är ett delfall av 3D-metoden ovan.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Tillgänglighet: 1.1.2. Namn ändrat från Affine till ST_Affine i 1.2.2

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Rotate a 3D line 180 degrees about the z axis. Note this is long-hand for doing ST_Rotate();

Code

```
SELECT ST_Affine(geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), 0, 0, 0, 1, 0, 0, 0) ↩
     As using_affine,
    ST_Rotate(geom, pi()) As using_rotate
   FROM (SELECT 'LINESTRING(1 2 3,1 4 3)'::geometry As geom) As foo;
```

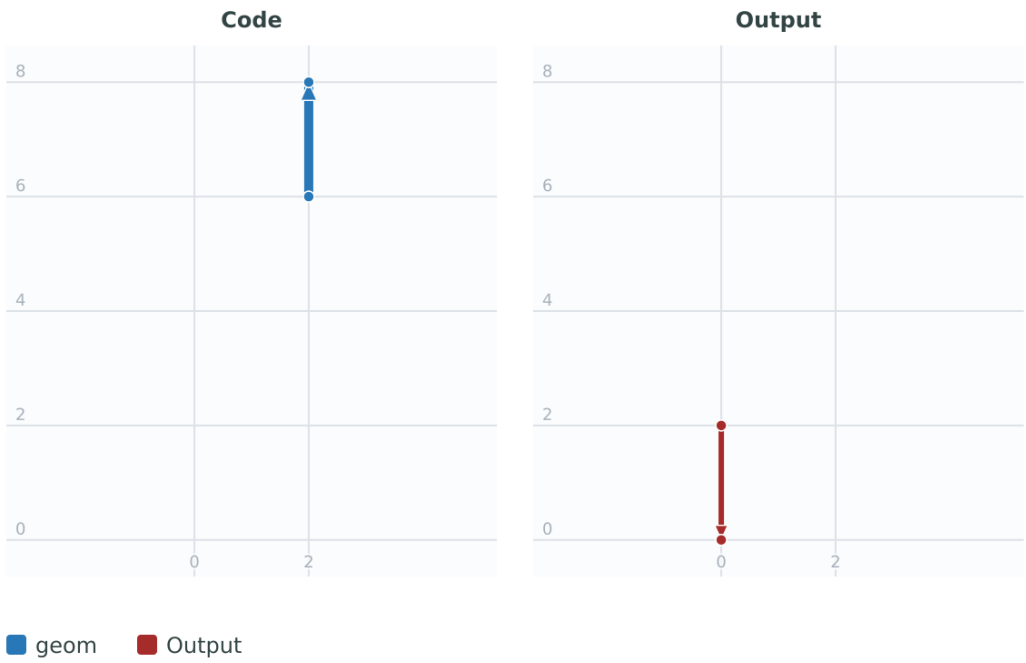
Figure

Rotate a 3D line 180 degrees in both the x and z axis.

Code

```
SELECT ST_Affine(geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), -sin(pi()), 0, sin( ↩
    pi()), cos(pi()), 0, 0, 0)
    FROM (SELECT 'LINESTRING(1 2 3,1 4 3)'::geometry As geom) As foo;
```

Figure



Se även

ST_Rotate, ST_Scale, ST_Translate, ST_TransScale

7.16.2 ST_Rotate

ST_Rotate — Roterar en geometri runt en ursprungspunkt.

Synopsis

geometry **ST_Rotate**(geometry geomA, float rotRadians);
geometry **ST_Rotate**(geometry geomA, float rotRadians, float x0, float y0);
geometry **ST_Rotate**(geometry geomA, float rotRadians, geometry pointOrigin);

Beskrivning

Roterar geometrin rotRadians moturs runt ursprungspunkten. Rotationsursprunget kan anges antingen som en POINT-geometri eller som x- och y-koordinater. Om origo inte anges roteras geometrin kring POINT(0 0).

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Förbättrad: 2.0.0 ytterligare parametrar för att ange rotationens ursprung lades till.

Tillgänglighet: 1.1.2. Namn ändrat från Rotate till ST_Rotate i 1.2.2

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

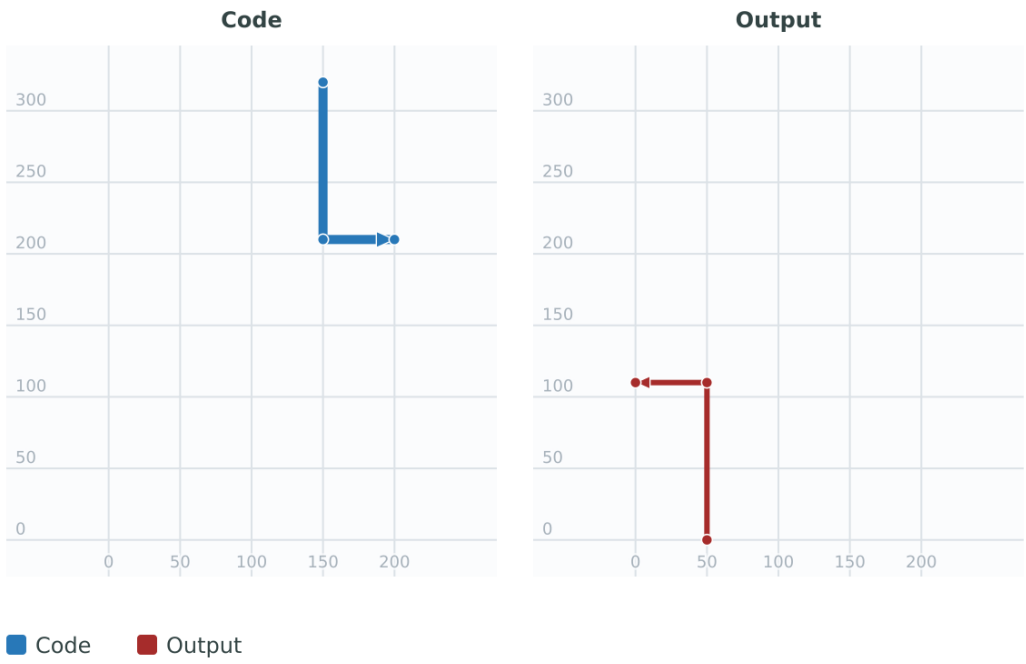
Exempel

This example rotates a geometry by 180 degrees.

Code

```
SELECT ST_Rotate('LINESTRING (50 160,50 50,100 50)'::geometry, pi());
```

Figure

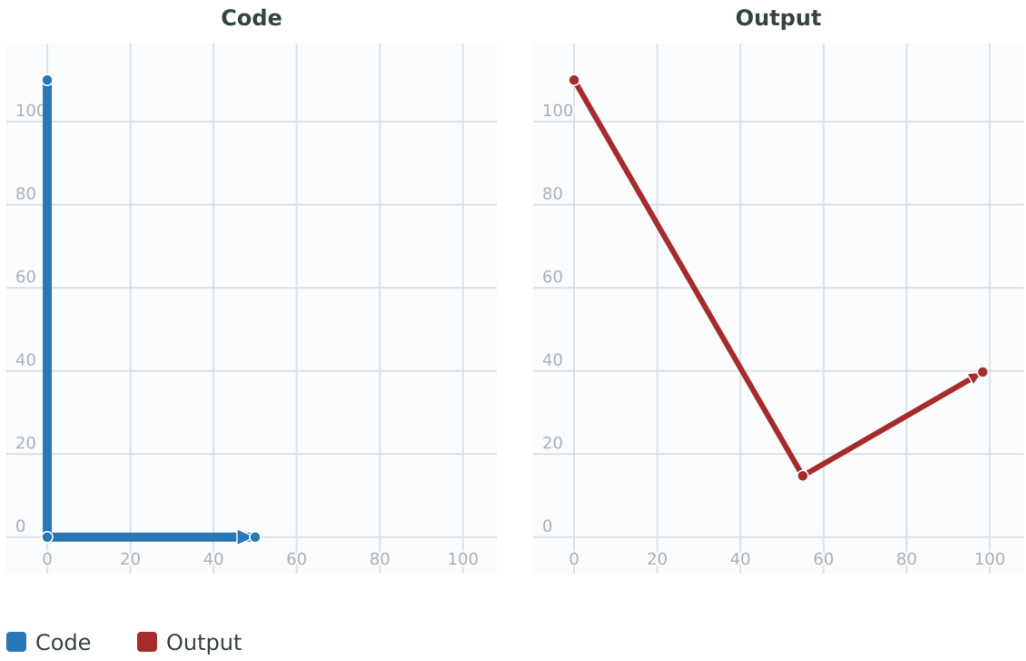


This example rotates a geometry 30 degrees counter-clockwise around x=50, y=160.

Code

```
SELECT ST_Rotate('LINESTRING (50 160,50 50,100 50)'::geometry, pi()/6, 50, 160);
```

Figure

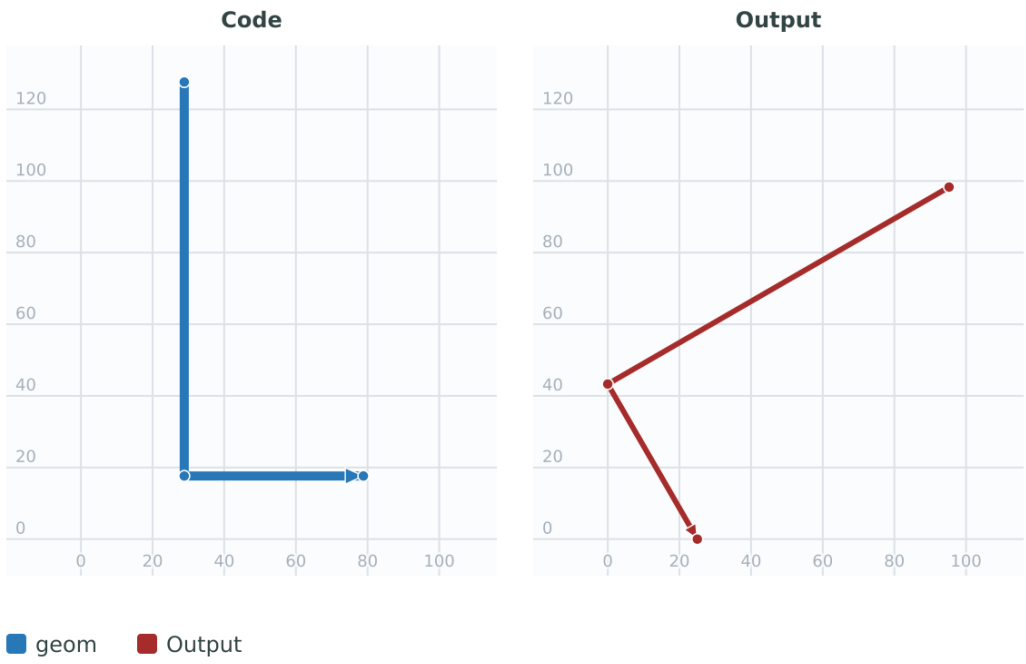


This example rotates a geometry 60 degrees clockwise around its centroid.

Code

```
SELECT ST_Rotate(geom, -pi()/3, ST_Centroid(geom))
FROM (SELECT 'LINESTRING (50 160,50 50,100 50)'::geometry AS geom) AS foo;
```

Figure



Se även

ST_Affine, ST_RotateX, ST_RotateY, ST_RotateZ

7.16.3 ST_RotateX

ST_RotateX — Roterar en geometri runt X-axeln.

Synopsis

geometry **ST_RotateX**(geometry geomA, float rotRadians);

Beskrivning

Roterar en geometri geomA - rotRadians runt X-axeln.

> **Notera**
> ST_RotateX(geomA, rotRadians) är en kortform av ST_Affine(geomA, 1, 0, 0, 0, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0).

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.
Tillgänglighet: 1.1.2. Namn ändrat från RotateX till ST_RotateX i 1.2.2

Denna funktion stöder polyedriska ytor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

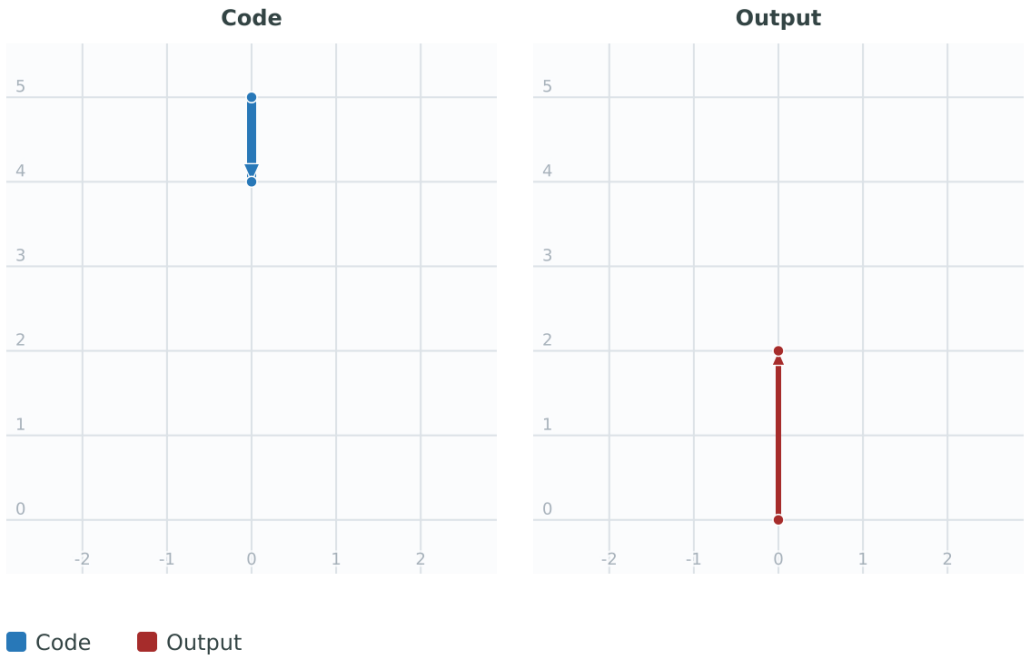
Exempel

This example rotates a line 90 degrees around the X axis.

Code

```
SELECT ST_RotateX('LINESTRING(1 2 3,1 1 1)', pi()/2);
```

Figure



Se även

ST_Affine, ST_RotateY, ST_RotateZ

7.16.4 ST_RotateY

ST_RotateY — Roterar en geometri runt Y-axeln.

Synopsis

geometry **ST_RotateY**(geometry geomA, float rotRadians);

Beskrivning

Roterar en geometri geomA - rotRadians runt y-axeln.

> **Notera**
> `ST_RotateY(geomA, rotRadians )` är en förkortning av `ST_Affine(geomA, cos(rotRadians), 0, sin(rotRadians), 0, 1, 0, -sin(rotRadians), 0, cos(rotRadians), 0, 0, 0)`.

Tillgänglighet: 1.1.2. Namn ändrat från RotateY till ST_RotateY i 1.2.2

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

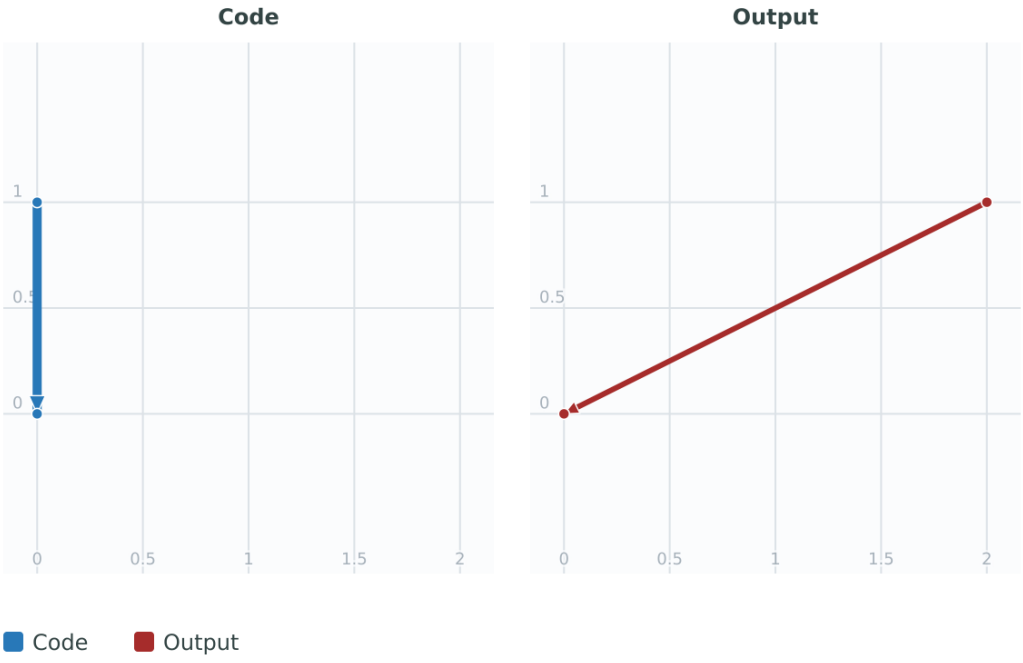
Exempel

This example rotates a line 90 degrees around the Y axis.

Code

```
SELECT ST_RotateY('LINESTRING(1 2 3,1 1 1)', pi()/2);
```

Figure



Se även

ST_Affine, ST_RotateX, ST_RotateZ

7.16.5 ST_RotateZ

ST_RotateZ — Roterar en geometri runt Z-axeln.

Synopsis

geometry **ST_RotateZ**(geometry geomA, float rotRadians);

Beskrivning

Roterar en geometri geomA - rotRadians runt Z-axeln.

Notera
Detta är en synonym för ST_Rotate

Notera
`ST_RotateZ(geomA, rotRadians )` är en förkortning av `SELECT ST_Affine(geomA, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0, 1, 0, 0, 0, 0)`..

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Tillgänglighet: 1.1.2. Namn ändrat från RotateZ till ST_RotateZ i 1.2.2

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

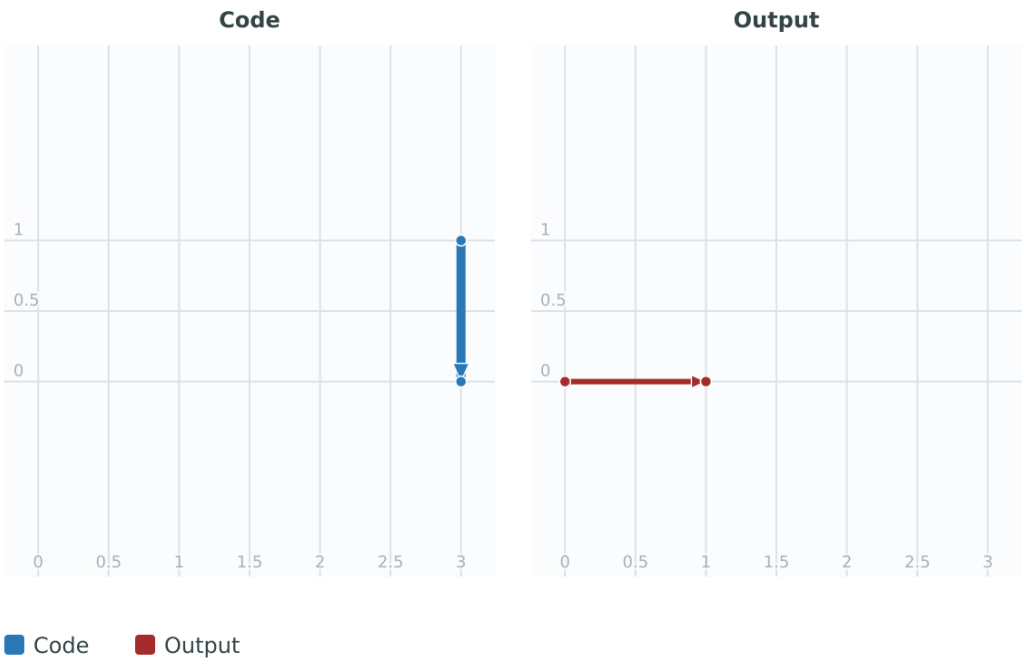
Exempel

This example rotates a line 90 degrees around the Z axis.

Code

```
SELECT ST_RotateZ('LINESTRING(1 2 3,1 1 1)', pi()/2);
```

Figure

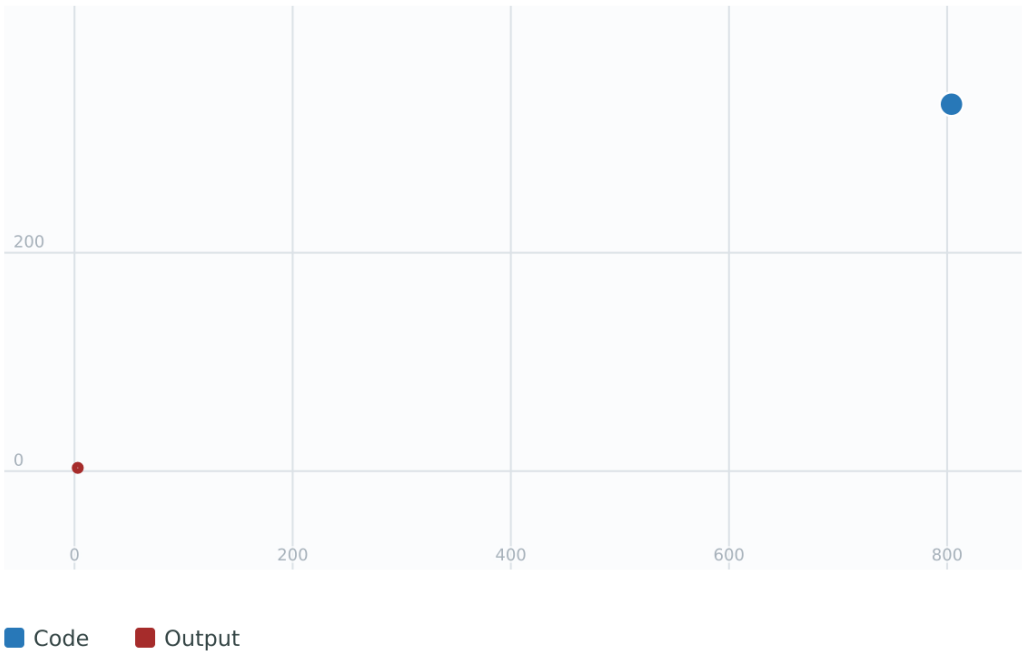


This example rotates a curved circle around the Z axis.

Code

```
SELECT ST_RotateZ(geom, pi()/2)
FROM (SELECT ST_LineToCurve(ST_Buffer('POINT(234 567)'::geometry, 3)) As geom) As foo;
```

Figure



Se även

ST_Affine, ST_RotateX, ST_RotateY

7.16.6 ST_Scale

ST_Scale — Skalar en geometri med givna faktorer.

Synopsis

geometry **ST_Scale**(geometry geomA, float XFactor, float YFactor, float ZFactor);
geometry **ST_Scale**(geometry geomA, float XFactor, float YFactor);
geometry **ST_Scale**(geometry geom, geometry factor);
geometry **ST_Scale**(geometry geom, geometry factor, geometry origin);

Beskrivning

Skalar geometrin till en ny storlek genom att multiplicera ordinaterna med motsvarande faktorparametrar.

Den version som tar en geometri som `faktorparameter` gör det möjligt att skicka en 2d-, 3dm-, 3dz- eller 4d-punkt för att ställa in skalningsfaktorn för alla dimensioner som stöds. Saknade dimensioner i `faktorpunkten` motsvarar ingen skalning av motsvarande dimension.

Varianten med tre geometrier gör det möjligt att ange ett falskt ursprungför skalningen. Detta möjliggör skalning på plats", t.ex. genom att geometrins centroid används som falskt ursprung. Utan ett falskt ursprung sker skalningen i förhållande till det faktiska ursprunget, så alla koordinater multipliceras bara med skalfaktorn.

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Tillgänglighet: 1.1.0.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Förbättrad: 2.2.0 stöd för skalning av alla dimensioner(`faktorparameter`) infördes.

Förbättrad: 2.5.0 stöd för skalning i förhållande till ett lokalt ursprung (`origin`-parametern) infördes.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder M-koordinater.

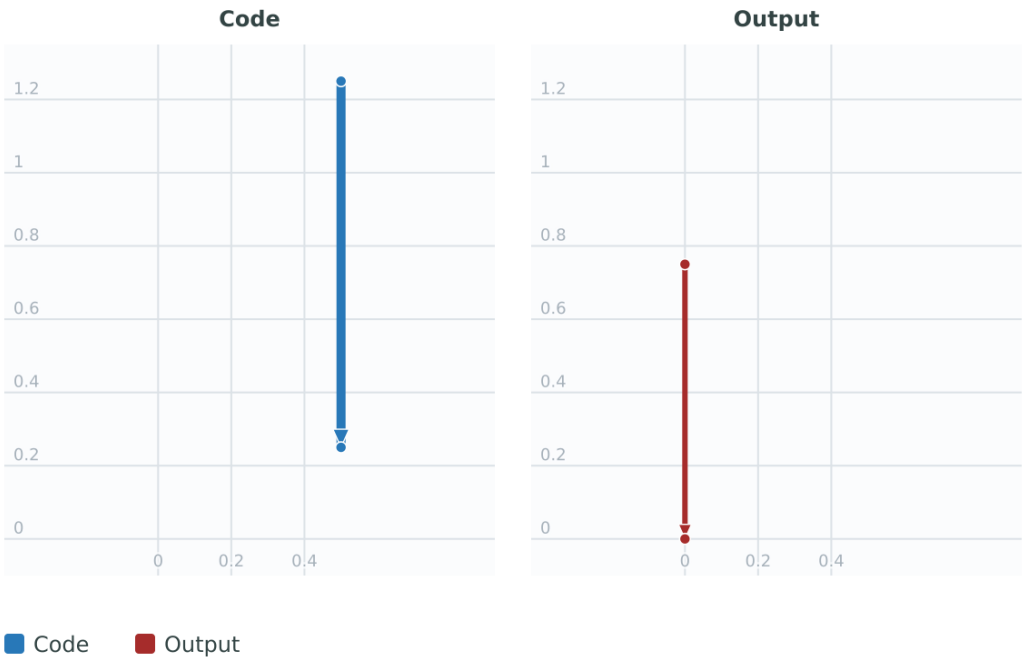
Exempel

This example scales X, Y, and Z values.

Code

```
SELECT ST_Scale('LINESTRING(1 2 3,1 1 1)', 0.5, 0.75, 0.8);
```

Figure

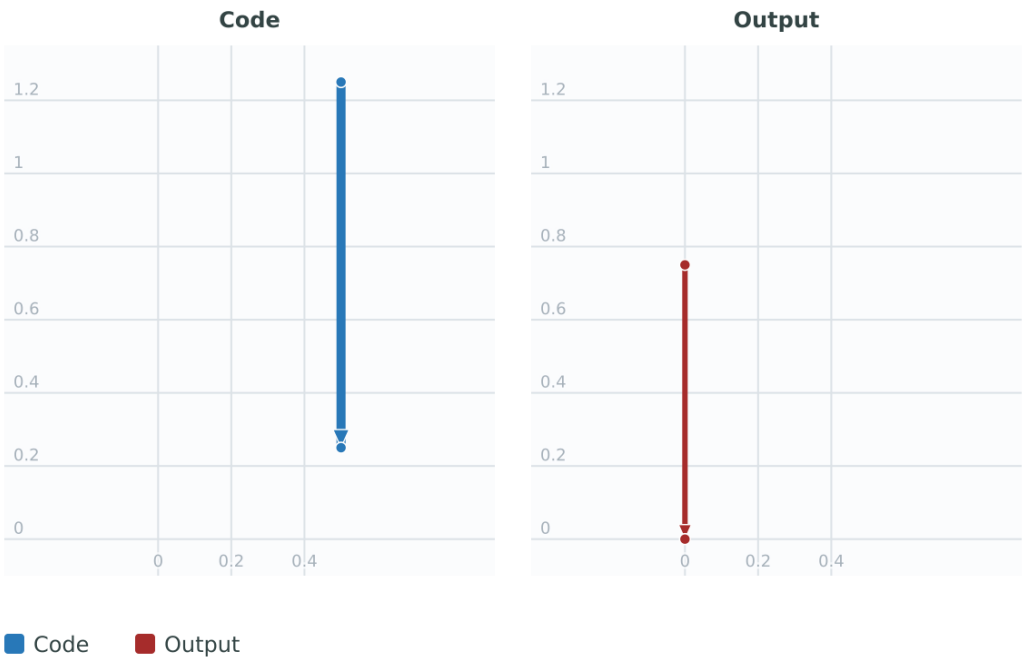


This example scales X and Y values.

Code

```
SELECT ST_Scale('LINESTRING(1 2 3,1 1 1)', 0.5, 0.75);
```

Figure

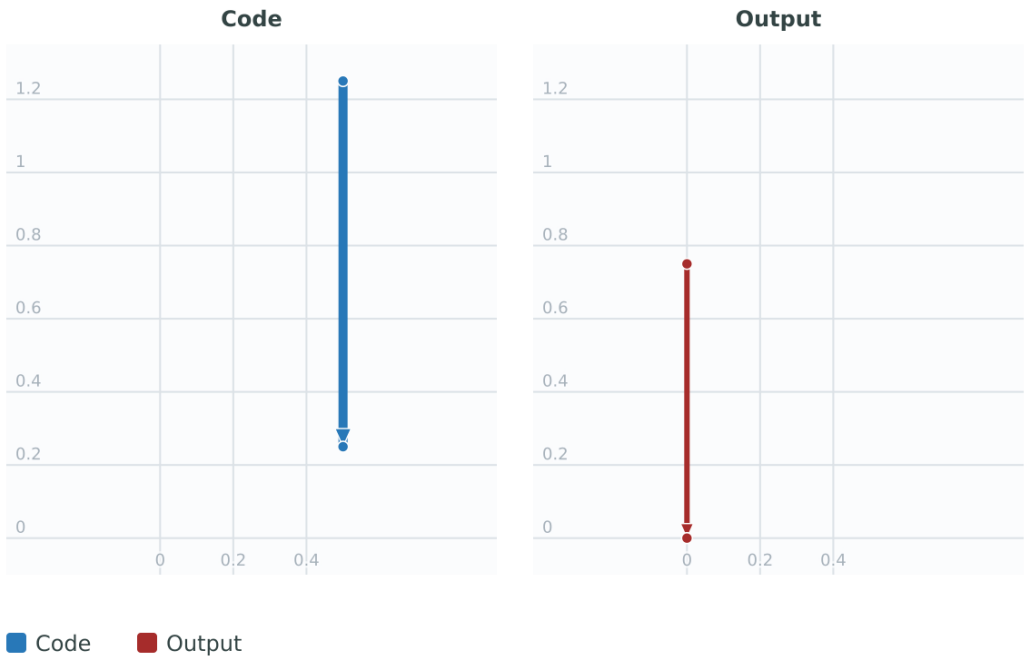


This example scales X, Y, Z, and M values.

Code

```
SELECT ST_Scale('LINESTRING(1 2 3 4,1 1 1 1)',
    ST_MakePoint(0.5, 0.75, 2, -1));
```

Figure

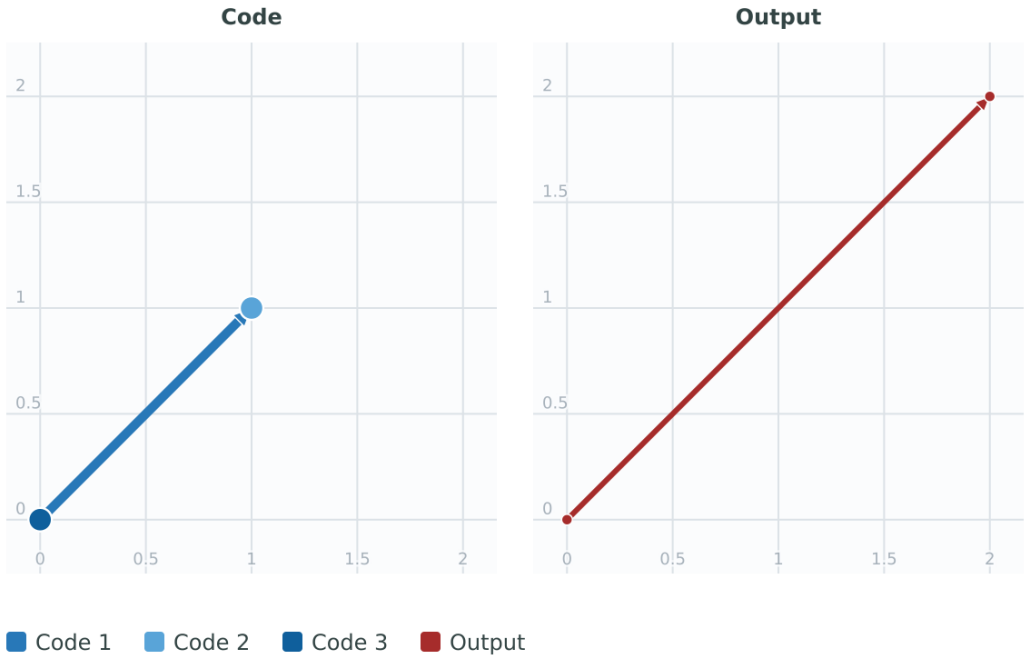


This example scales X and Y values using a false origin.

Code

```
SELECT ST_Scale('LINESTRING(1 1,2 2)', 'POINT(2 2)', 'POINT(1 1)'::geometry);
```

Figure



This example constructs an ellipse centered at (10, 20), with X radius 8, Y radius 4, and rotated 30 degrees counter-clockwise. Scaling and rotation use the center as their origin, so the ellipse keeps both its position and SRID.

Code

```
WITH params AS (
  SELECT
    ST_SetSRID(ST_Point(10, 20), 3857) AS center,
    8.0 AS x_radius,
    4.0 AS y_radius,
    radians(30) AS angle
)
SELECT ST_Rotate(
         ST_Scale(
           ST_Buffer(center, 1, 'quad_segs=32'),
           ST_MakePoint(x_radius, y_radius),
           center
         ),
         angle,
         center
       ) AS ellipse
FROM params;
```

Se även

ST_Affine, ST_Buffer, ST_MakePoint, ST_Rotate, ST_TransScale

7.16.7 ST_Translate

ST_Translate — Translaterar en geometri med givna offsets.

Synopsis

geometry **ST_Translate**(geometry g1, float deltax, float deltay);
geometry **ST_Translate**(geometry g1, float deltax, float deltay, float deltaz);

Beskrivning

Returns a new geometry whose coordinates are translated delta x,delta y,delta z units. Units are based on the units defined in spatial reference (SRID) for this geometry. M coordinates are preserved.

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Tillgänglighet: 1.2.2

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder M-koordinater.

Exempel

Flytta en punkt 1 grads longitud

Code

```
SELECT ST_Translate(ST_GeomFromText('POINT(-71.01 42.37)', 4326), 1, 0) AS wgs_transgeomtxt ↩
    ;
```

Output

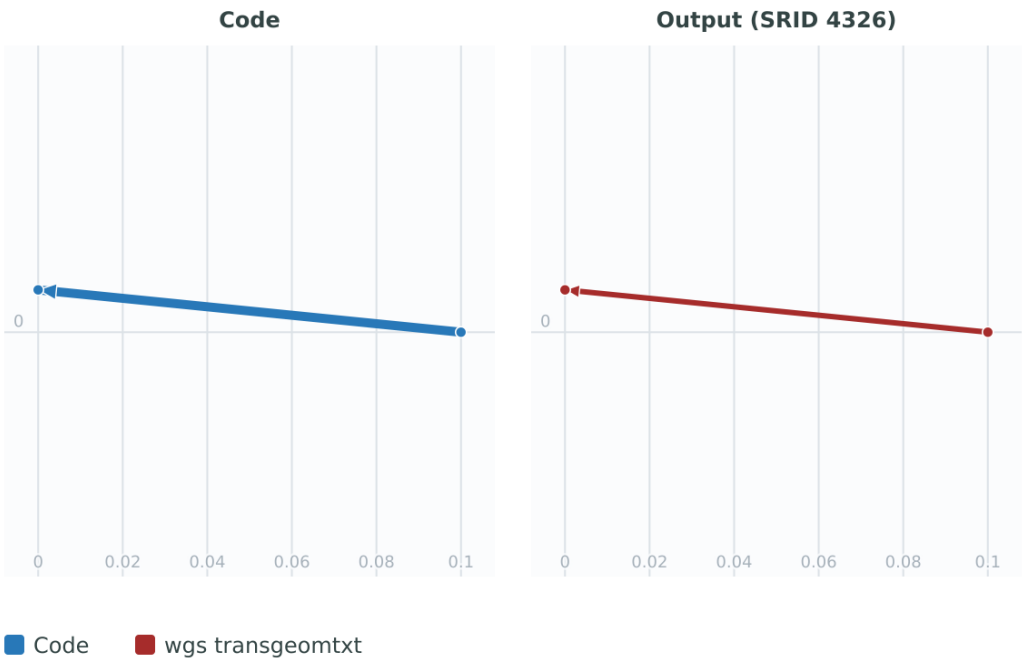
```
POINT(-70.01 42.37)
```

Flytta en linestrings 1 grads longitud och 1/2 grads latitud

Code

```
SELECT ST_Translate(ST_GeomFromText('LINESTRING(-71.01 42.37,-71.11 42.38)', 4326), 1, 0.5) ↩
    AS wgs_transgeomtxt;
```

Figure



Flytta en 3d-punkt

Code

```
SELECT ST_Translate(CAST('POINT(0 0 0)' AS geometry), 5, 12, 3);
```

Output

```
POINT(5 12 3)
```

Move points with a measure coordinate

Code

```
SELECT ST_Translate('POINTM(0 0 11)', 5, 12);
```

Figure

Code

```
SELECT ST_Translate('POINT(0 0 7 11)', 5, 12, 3);
```

Output

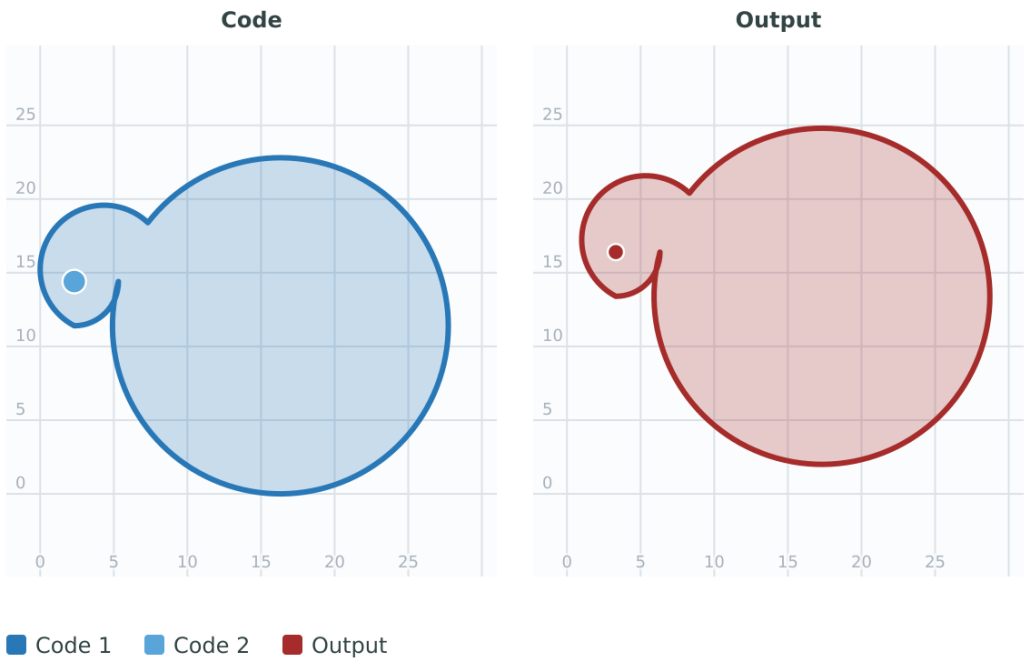
```
POINT(5 12 10 11)
```

Flytta en kurva och en punkt

Code

```
SELECT ST_Translate(ST_Collect('CURVEPOLYGON(CIRCULARSTRING(4 3,3.12 0.878,1 0,-1.121 ↩
    5.1213,6 7,8 9,4 3))', 'POINT(1 3)'), 1, 2);
```

Figure



Se även

ST_Affine, ST_AsText, ST_GeomFromText

7.16.8 ST_TransScale

ST_TransScale — Translaterar och skalar en geometri med hjälp av givna offsets och faktorer.

Synopsis

geometry **ST_TransScale**(geometry geomA, float deltaX, float deltaY, float XFactor, float YFactor);

Beskrivning

Translaterar geometrin med hjälp av argen deltaX och deltaY och skalar den sedan med hjälp av argen XFactor och YFactor, endast i 2D.

Notera
`ST_TransScale(geomA, deltaX, deltaY, XFactor, YFactor)` är en förkortning av `ST_Affine(geomA, XFactor, 0, 0, 0, YFactor, 0, 0, 0, 1, deltaX*XFactor, deltaY*YFactor, 0)`.

Notera
Före 1.3.4 kraschade den här funktionen om den användes med geometrier som innehåller CURVES. Detta är åtgärdat i 1.3.4+

Tillgänglighet: 1.1.0.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_TransScale('LINESTRING(1 2 3,1 1 1)', 0.5, 1, 1, 2);
```

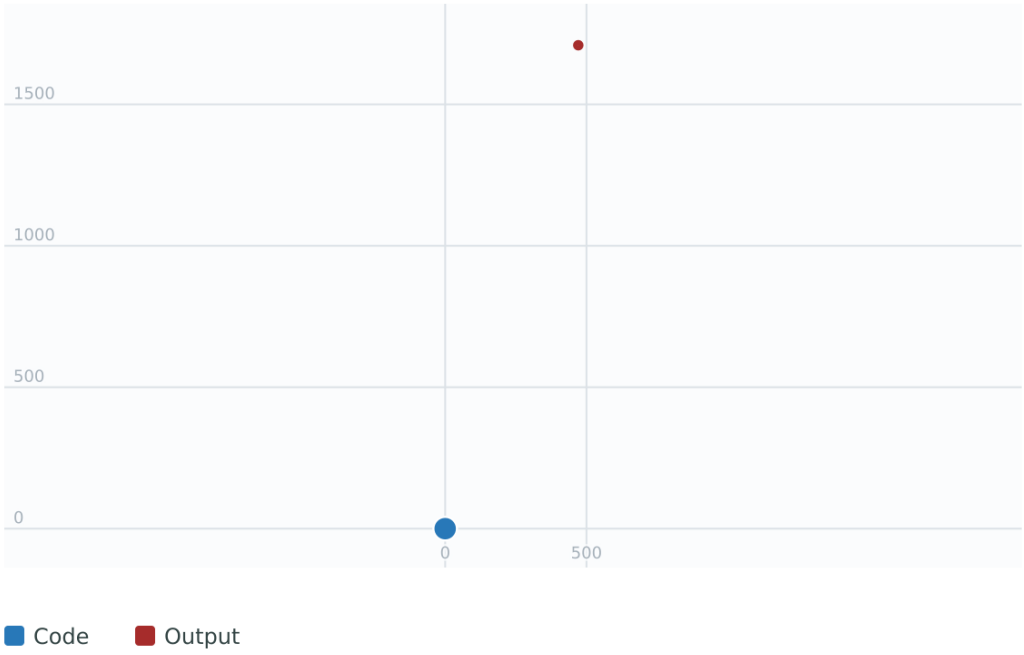
Figure

This example buffers a point to approximate a circle, converts it to a curve, translates it by 1,2, and scales it by 3,4.

Code

```
SELECT ST_TransScale(ST_LineToCurve(ST_Buffer('POINT(234 567)', 3)), 1, 2, 3, 4);
```

Figure



Se även

ST_Affine, ST_Translate

7.17 Klustringsfunktioner

7.17.1 ST_ClusterDBSCAN

ST_ClusterDBSCAN — Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av DBSCAN-algoritmen.

Synopsis

integer **ST_ClusterDBSCAN**(geometry winset geom, float8 eps, integer minpoints);

Beskrivning

En fönsterfunktion som returnerar ett klusternummer för varje indatageometri med hjälp av DBSCAN-algoritmen (2D Density-based spatial clustering of applications with noise). Till skillnad från ST_ClusterKMe kräver den inte att antalet kluster anges, utan använder istället parametrarna för önskat avstånd (`eps`) och densitet (`minpoints`) för att bestämma varje kluster.

En indatageometri läggs till i ett kluster om den är antingen:

- En ”kärn-geometri, som ligger inom `eps` avstånd från minst `minpoints` indatageometrier (inklusive sig själv); eller
- En gräns-geometri som ligger inom `eps` avstånd från en kärngeometri.

Observera att gränsgeometrier kan ligga inom `eps-avstånd` från kärngeometrier i mer än ett kluster. Båda tilldelningarna skulle vara korrekta, så gränsgeometrin kommer godtyckligt att tilldelas ett av de tillgängliga klustren. I den här situationen är det möjligt att ett korrekt kluster genereras med färre än `minpunktsgeometrier`. För att säkerställa deterministisk tilldelning av gränsgeometrier (så att upprepade anrop till ST_ClusterDBSCAN ger identiska resultat), använd en `ORDER BY-klausul` i fönsterdefinitionen. Tvetydiga klustertilldelningar kan skilja sig från andra DBSCAN-implementeringar.

Notera
Geometrier som inte uppfyller kriterierna för att ingå i något kluster tilldelas ett klusternummer som är NULL.

Tillgänglighet: 2.3.0

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Cluster polygons within 50 units of each other, requiring at least two polygons per cluster. The isolated polygon is reported as noise.

Code

```
WITH input(name, geom) AS (
  VALUES
    ('A1', ST_MakeEnvelope(0, 0, 10, 10)),
    ('A2', ST_MakeEnvelope(30, 0, 40, 10)),
    ('B1', ST_MakeEnvelope(100, 0, 110, 10)),
    ('B2', ST_MakeEnvelope(130, 0, 140, 10)),
    ('noise', ST_MakeEnvelope(250, 0, 260, 10))
), clustered AS (
  SELECT name,
         geom,
         ST_ClusterDBSCAN(geom, eps => 50, minpoints => 2)
           OVER (ORDER BY name) AS cid
  FROM input
)
SELECT COALESCE(cid::text, 'noise') AS cluster,
       string_agg(name, ',' ORDER BY name) AS members,
       ST_Collect(geom ORDER BY name) AS cluster_geom
FROM clustered
GROUP BY cid
ORDER BY cid NULLS LAST;
```

Output

```
0 | A1,A2 | MULTIPOLYGON(((0 0,0 10,10 10,10 0,0 0)),((30 0,30 10,40 10,40 0,30 0)))
1 | B1,B2 | MULTIPOLYGON(((100 0,100 10,110 10,110 0,100 0)),((130 0,130 10,140 10,140 0,130 0)))
noise | noise | MULTIPOLYGON(((250 0,250 10,260 10,260 0,250 0)))
```

Figure

Ett exempel som visar hur man kombinerar skiften med samma klusternummer till geometriska samlingar.

Code

```
SELECT cid, ST_Collect(geom) AS cluster_geom, array_agg(parcel_id) AS ids_in_cluster FROM (
    SELECT parcel_id, ST_ClusterDBSCAN(geom, eps => 0.5, minpoints => 5) over () AS cid,
    geom
    FROM parcels) sq
GROUP BY cid;
```

Se även

ST_DWithin, ST_ClusterKMeans, ST_ClusterIntersecting, ST_ClusterIntersectingWin, ST_ClusterWithin, ST_ClusterWithinWin

7.17.2 ST_ClusterIntersecting

ST_ClusterIntersecting — Aggregerad funktion som klustrar inmatade geometrier till sammanhängande mängder.

Synopsis

geometry[] **ST_ClusterIntersecting**(geometry set g);

Beskrivning

En aggregerad funktion som returnerar en array av GeometryCollections som delar upp indatageometrierna i sammanhängande kluster som är disjunkta. Varje geometri i ett kluster skär minst en annan geometri i klustret, och skär inte någon geometri i andra kluster.

Tillgänglighet: 2.2.0

Exempel

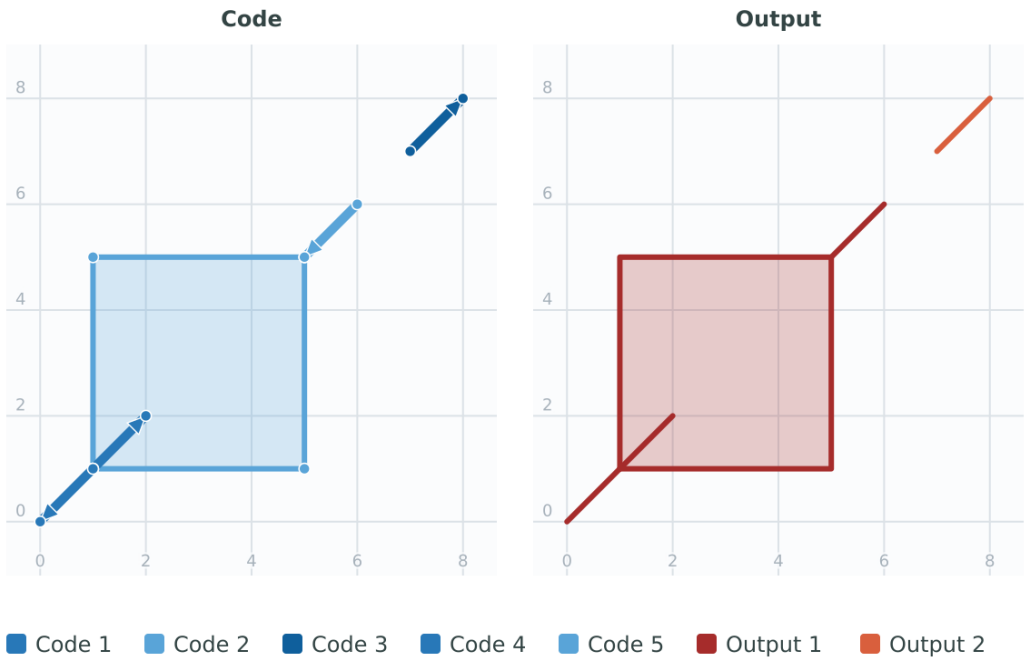
Code

```
WITH testdata AS
  (SELECT unnest(
            ARRAY['LINESTRING (0 0,1 1)'::geometry,
            'LINESTRING (5 5,4 4)'::geometry,
            'LINESTRING (6 6,7 7)'::geometry,
            'LINESTRING (0 0,-1 -1)'::geometry,
            'POLYGON ((0 0,4 0,4 4,0 4,0 0))'::geometry]) AS geom)

SELECT unnest(ST_ClusterIntersecting(geom)) FROM testdata;
```

Output

```
GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),LINESTRING(5 5,4 4),LINESTRING(0 0,-1 -1),POLYGON((0 0,4 0,4 4,0 4,0 0)))
GEOMETRYCOLLECTION(LINESTRING(6 6,7 7))
```

Figure



Se även

ST_ClusterIntersectingWin, ST_ClusterWithin, ST_ClusterWithinWin

7.17.3 ST_ClusterIntersectingWin

ST_ClusterIntersectingWin — Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri och klustrar indatageometrier i sammanhängande uppsättningar.

Synopsis

integer **ST_ClusterIntersectingWin**(geometry winset geom);

Beskrivning

En fönsterfunktion som bygger sammanhängande kluster av geometrier som korsar varandra. Det är möjligt att traversera alla geometrier i ett kluster utan att lämna klustret. Returvärdet är det klusternummer som geometriargumentet deltar i, eller null för null-indata.

Tillgänglighet: 3.4.0

Exempel

Return an explicit cluster id for each input geometry. Using the conventional `cid` alias keeps the geometry column ready for color-by-cluster rendering.

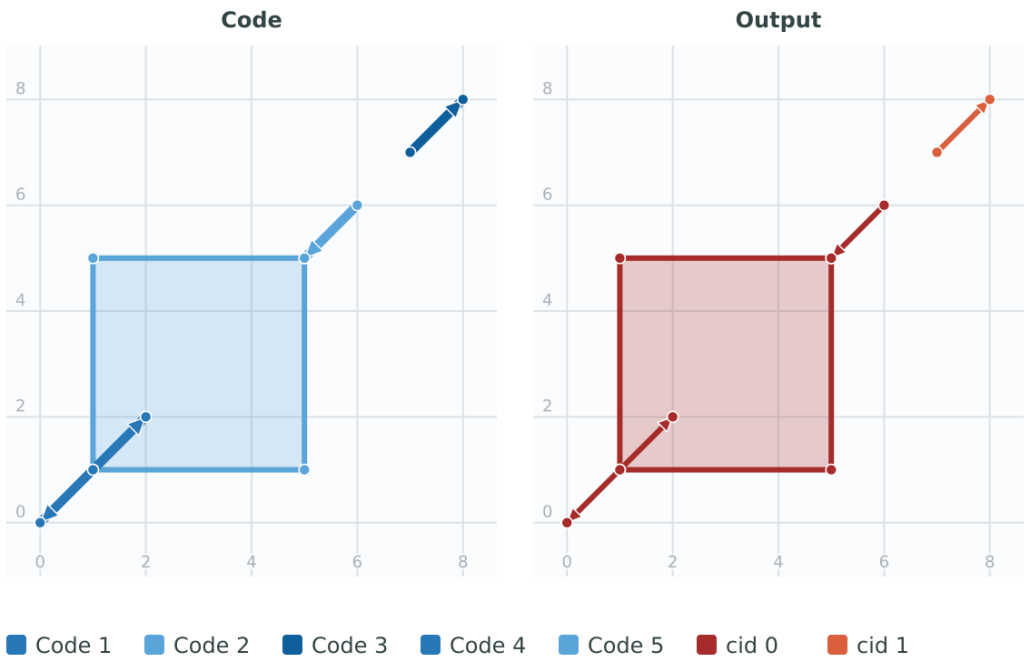
Code

```
WITH testdata(id, geom) AS (
  VALUES (1, 'LINESTRING (0 0,1 1)'::geometry),
         (2, 'LINESTRING (5 5,4 4)'::geometry),
         (3, 'LINESTRING (6 6,7 7)'::geometry),
         (4, 'LINESTRING (0 0,-1 -1)'::geometry),
         (5, 'POLYGON ((0 0,4 0,4 4,0 4,0 0))'::geometry)
), clustered AS (
  SELECT id,
         geom,
         ST_ClusterIntersectingWin(geom) OVER () AS cid
  FROM testdata
)
SELECT id,
       geom AS geom_wkt,
       cid
FROM clustered
ORDER BY id;
```

Output

```
id |            geom_wkt             | cid
---+---------------------------------+-----
  1 | LINESTRING(0 0,1 1)             |   0
  2 | LINESTRING(5 5,4 4)             |   0
  3 | LINESTRING(6 6,7 7)             |   1
  4 | LINESTRING(0 0,-1 -1)           |   0
  5 | POLYGON((0 0,4 0,4 4,0 4,0 0)) |   0
```

Figure



Se även

ST_ClusterIntersecting, ST_ClusterWithin, ST_ClusterWithinWin

7.17.4 ST_ClusterRelateWin

ST_ClusterRelateWin — Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.

Synopsis

integer **ST_ClusterRelateWin**(geometry winset geom, text relate_matrix);

Beskrivning

A window function that builds connected clusters of geometries that intersect. Geometries are added to a cluster if they share a pairwise DE9IM relationship with another member of the cluster. With this function it is possible to build a cluster of all objects that touch at boundaries, but exclude those that merely overlap.

Availability: 3.7.0

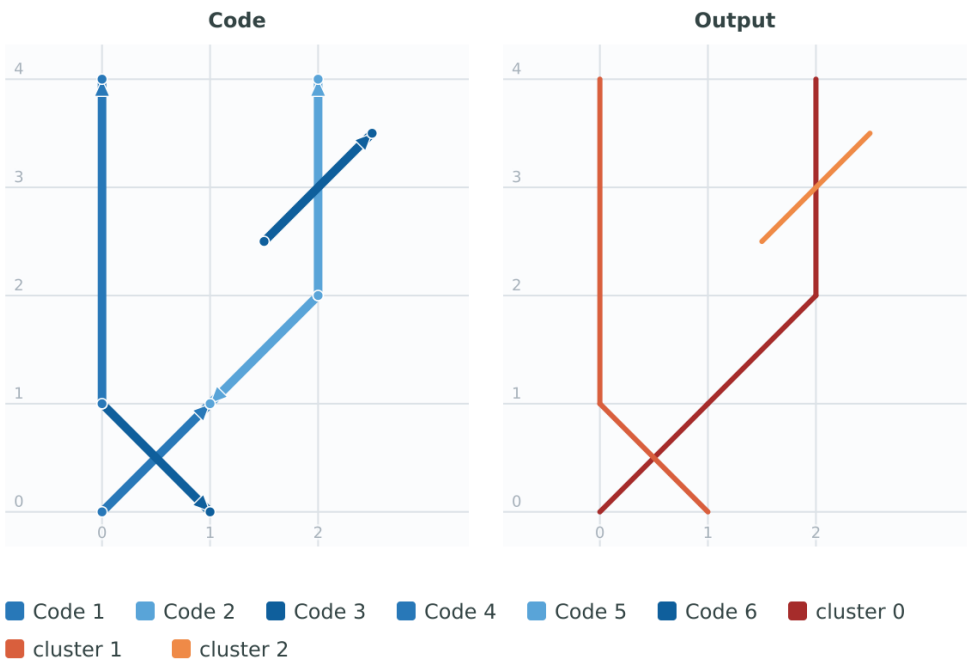
Exempel

This collection of line strings would form a single cluster using ST_ClusterIntersectingWin, but using ST_ClusterRelateWin can be clustered into three groups that connect only via their end points.

Code

```
WITH data(id, geom) AS (
  VALUES (1, 'LINESTRING(0 0,1 1)'::geometry),
         (2, 'LINESTRING(2 2,1 1)'::geometry),
         (3, 'LINESTRING(0 1,1 0)'::geometry),
         (4, 'LINESTRING(0 1,0 4)'::geometry),
         (5, 'LINESTRING(2 2,2 4)'::geometry),
         (6, 'LINESTRING(1.5 2.5,2.5 3.5)'::geometry)
), clustered AS (
  SELECT id, geom,
         ST_ClusterRelateWin(geom, '****0****') OVER () AS cluster
  FROM data
)
SELECT cluster,
       string_agg(id::text, ',' ORDER BY id) AS members,
       ST_Collect(geom ORDER BY id) AS cluster_geom
FROM clustered
GROUP BY cluster
ORDER BY cluster;
```

Figure



Se även

ST_Relate, ST_ClusterIntersectingWin, ST_ClusterWithinWin

7.17.5 ST_MinimumSpanningTree

ST_MinimumSpanningTree — Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.

Synopsis

integer **ST_MinimumSpanningTree**(geometry winset geom);

Beskrivning

A window function that builds connected graphs of line strings based on the Minimum Spanning Tree (MST) of the input geometries. The return value is the cluster number that the geometry argument participates in, or zero if it is not part of the minimum tree.

The Minimum Spanning Tree connects all geometries in the window partition such that the total length of the connecting lines is minimized. If the graph is not fully connected (e.g. infinite distance between some geometries), it produces a Minimum Spanning Forest, and each connected component is assigned a unique tree ID.

Availability: 3.7.0

Requires GEOS >= 3.15.0

Exempel

These six edges describe a square with both diagonals. The minimum spanning tree keeps three edges and excludes the rest.

Code

```
WITH edges(id, geom) AS (
  VALUES
    (1, 'LINESTRING(0 0,1 0)'::geometry),
    (2, 'LINESTRING(0 0,0 1)'::geometry),
    (3, 'LINESTRING(1 1,0 1)'::geometry),
    (4, 'LINESTRING(1 1,1 0)'::geometry),
    (5, 'LINESTRING(0 0,1 1)'::geometry),
    (6, 'LINESTRING(1 0,0 1)'::geometry)
), marked AS (
  SELECT id,
         geom,
         ST_MinimumSpanningTree(geom) OVER (ORDER BY id) AS tree_id
  FROM edges
)
SELECT string_agg(id::text, ',' ORDER BY id)
         FILTER (WHERE tree_id > 0) AS tree_edge_ids,
       ST_Collect(geom ORDER BY id)
         FILTER (WHERE tree_id > 0) AS tree_edges,
       string_agg(id::text, ',' ORDER BY id)
         FILTER (WHERE tree_id = 0) AS excluded_edge_ids,
       ST_Collect(geom ORDER BY id)
         FILTER (WHERE tree_id = 0) AS excluded_edges
FROM marked;
```

Output

```
-[ RECORD 1 ]-----
tree_edge_ids     | 1,2,3
tree_edges        | MULTILINESTRING((0 0,1 0),(0 0,0 1),(1 1,0 1))
excluded_edge_ids | 4,5,6
excluded_edges    | MULTILINESTRING((1 1,1 0),(0 0,1 1),(1 0,0 1))
```

Se även

ST_ClusterDBSCAN, ST_ClusterKMeans, ST_ClusterIntersectingWin

7.17.6 ST_ClusterKMeans

ST_ClusterKMeans — Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av K-means-algoritmen.

Synopsis

integer **ST_ClusterKMeans**(geometry winset geom , integer k , float8 max_radius);

Beskrivning

Returnerar K-means klusternummer för varje indatageometri. Avståndet som används för klustring är avståndet mellan centroiderna för 2D-geometrier och avståndet mellan bounding box-centren för 3D-geometrier. För POINT-indata kommer M-koordinaten att behandlas som indataens vikt och måste vara större än 0.

`max_radius`, om den är inställd, gör att ST_ClusterKMeans genererar fler kluster än `k` och säkerställer att inget kluster i utdata har en radie som är större än `max_radius`. Detta är användbart vid räckviddsanalys.

Förbättrad: 3.2.0 Stöd för `max_radius`

Förbättrad: 3.1.0 Stöd för 3D-geometrier och vikter

Tillgänglighet: 2.3.0

Exempel

Define the parcel geometries once in a CTE and reuse them across the executable examples.

Parcels color-coded by cluster number (`cid`).

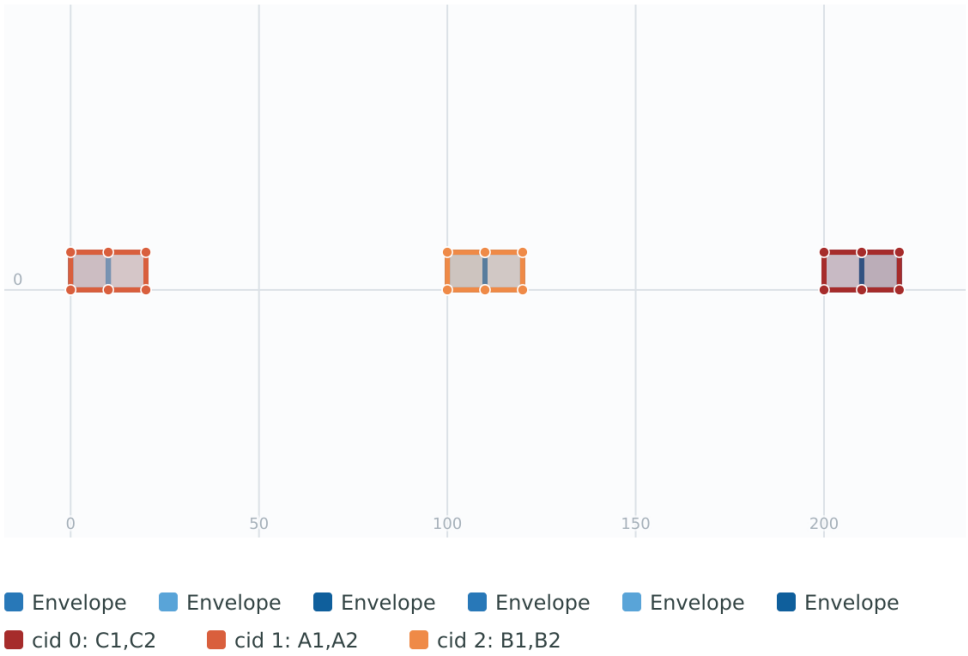
Code

```
WITH parcels(parcel_id, geom) AS (
  VALUES
    ('A1', ST_MakeEnvelope(0, 0, 10, 10)),
    ('A2', ST_MakeEnvelope(10, 0, 20, 10)),
    ('B1', ST_MakeEnvelope(100, 0, 110, 10)),
    ('B2', ST_MakeEnvelope(110, 0, 120, 10)),
    ('C1', ST_MakeEnvelope(200, 0, 210, 10)),
    ('C2', ST_MakeEnvelope(210, 0, 220, 10))
), clustered AS (
  SELECT parcel_id,
         geom,
         ST_ClusterKMeans(geom, 3) OVER (ORDER BY parcel_id) AS cid
  FROM parcels
)
SELECT cid,
       string_agg(parcel_id, ',' ORDER BY parcel_id) AS parcels,
       ST_Union(geom) AS cluster_geom
FROM clustered
GROUP BY cid
ORDER BY cid;
```

Output

```
0 | C1,C2 | POLYGON((200 10,210 10,220 10,220 0,210 0,200 0,200 10))
1 | A1,A2 | POLYGON((0 10,10 10,20 10,20 0,10 0,0 0,0 10))
2 | B1,B2 | POLYGON((100 10,110 10,120 10,120 0,110 0,100 0,100 10))
```

Figure



Uppdelning av paketkluster efter typ:

Code

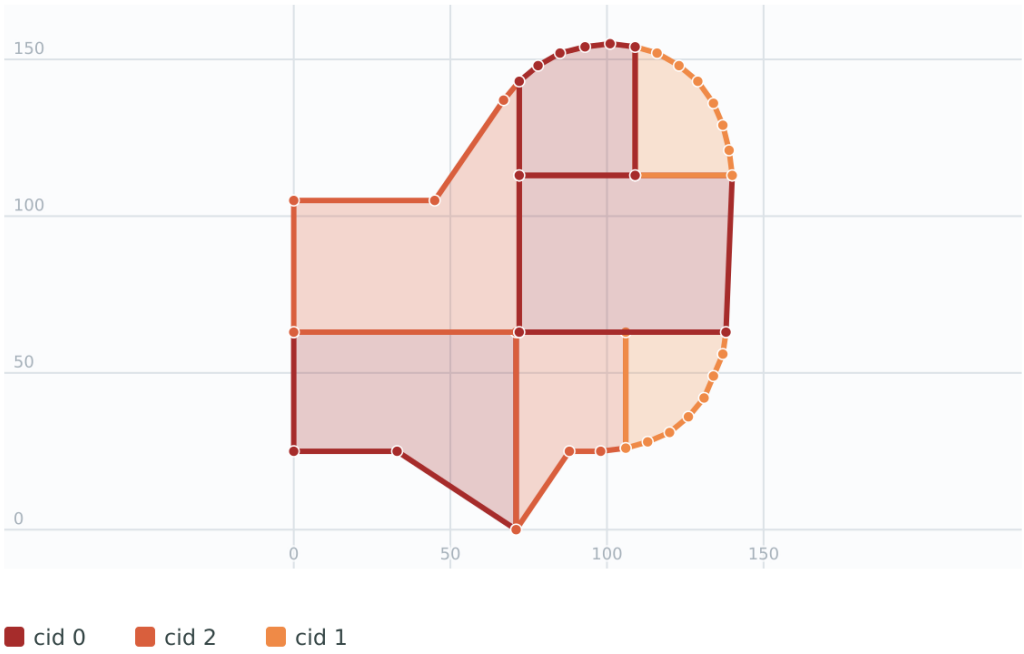
```
WITH parcel_geoms AS (
  SELECT geom
  FROM ST_Subdivide(
    ST_Buffer('SRID=3857;LINESTRING(40 100,98 100,100 150,60 90)'::geometry,
              40, 'endcap=square'),
    12) AS geom
), parcels AS (
  SELECT lpad(row_number() OVER (ORDER BY ST_YMin(geom), ST_XMin(geom))::text, 3, '0') AS ↩
    parcel_id,
         geom,
         ('{residential,commercial}'::text[])[1 + mod(row_number() OVER (ORDER BY ST_YMin( ↩
    geom), ST_XMin(geom)), 2)] AS type
  FROM parcel_geoms
)
SELECT ST_ClusterKMeans(geom, 3) over (PARTITION BY type) AS cid,
       parcel_id,
       type,
       ST_SnapToGrid(geom, 1) AS parcel_geom
FROM parcels
ORDER BY type, parcel_id;
```

Output

```
cid | parcel_id |    type     | parcel_geom
----+-----------+-------------+-----------------------------------------------------------------
   0 | 001       | commercial  | POLYGON((33 60,0 60,0 98,71 98,71 35,33 60))
   2 | 003       | commercial  | POLYGON((106 98,138 98,137 91,134 84,131 77,126 71,120 66,113 63,106 61,106 98))
   2 | 005       | commercial  | POLYGON((140 148,138 98,72 98,72 148,140 148))
```

```
   1 | 007       | commercial  | POLYGON((109 189,116 187,123 183,129 178,134 171,137 164,139 156,140 148,109 148,109 ↩
     189))
   0 | 002       | residential | POLYGON((71 98,106 98,106 61,98 60,88 60,71 35,71 98))
   2 | 004       | residential | POLYGON((0 98,0 140,45 140,67 172,72 178,72 98,0 98))
   1 | 006       | residential | POLYGON((72 178,78 183,85 187,93 189,101 190,109 189,109 148,72 148,72 178))
```

Figure



Example: Clustering a preaggregated planetary-scale data population dataset using 3D clustering and weighting. Identify at least 20 regions based on Kontur Population Data that do not span more than 3000 km from their center:

Code

```
CREATE TABLE kontur_population_3000km_clusters AS
SELECT
    geom,
    ST_ClusterKMeans(
        ST_Force4D(
            ST_Transform(ST_Force3D(geom), 4978),
            mvalue => population
        ),
        20,
        max_radius => 3000000
    ) OVER () AS cid
FROM kontur_population;
```

The clustering produces 46 regions. Clusters are centered at well-populated regions such as New York and Moscow. Greenland forms one cluster, several island clusters span the antimeridian, and cluster edges follow the Earth's curvature.

Kontur population clustered with 3000 km maximum radius.

Cluster weighted city populations on the Earth in geocentric coordinates. The maximum radius is specified in meters, while the returned geometries stay in WGS 84 for display.

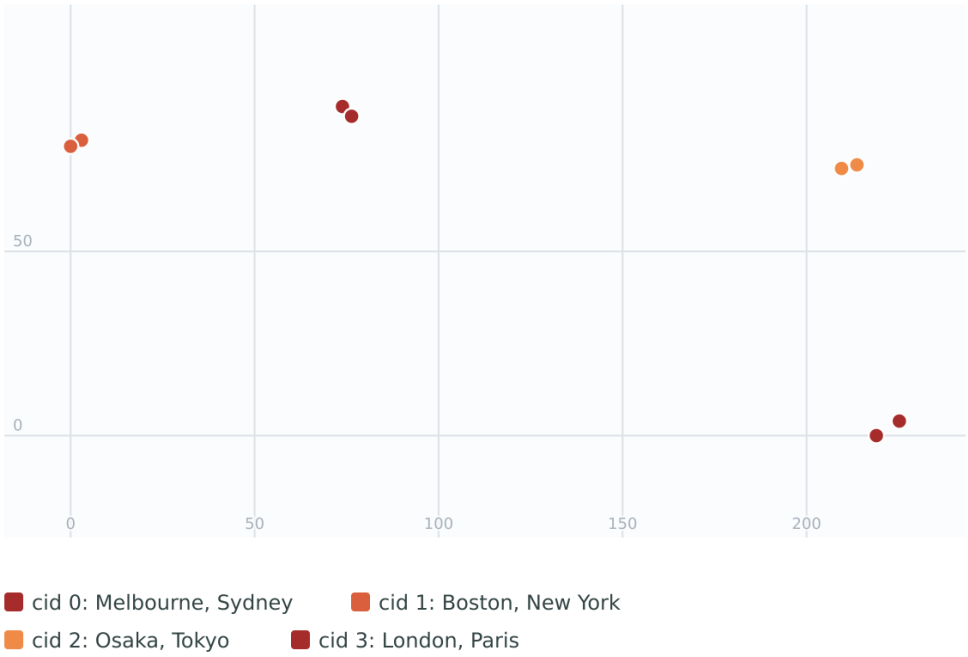
Code

```
WITH population(place, population, geom) AS (
  VALUES
    ('Boston', 5.0, ST_SetSRID(ST_Point(-71.06, 42.36), 4326)),
    ('New York', 20.0, ST_SetSRID(ST_Point(-74.01, 40.71), 4326)),
    ('London', 15.0, ST_SetSRID(ST_Point(-0.13, 51.51), 4326)),
    ('Paris', 11.0, ST_SetSRID(ST_Point(2.35, 48.86), 4326)),
    ('Osaka', 19.0, ST_SetSRID(ST_Point(135.50, 34.69), 4326)),
    ('Tokyo', 37.0, ST_SetSRID(ST_Point(139.69, 35.68), 4326)),
    ('Melbourne', 5.0, ST_SetSRID(ST_Point(144.96, -37.81), 4326)),
    ('Sydney', 5.0, ST_SetSRID(ST_Point(151.21, -33.87), 4326))
), clustered AS (
  SELECT place,
         geom,
         ST_ClusterKMeans(
           ST_Force4D(
             ST_Transform(ST_Force3D(geom), 4978),
             mvalue => population
           ),
           4,
           max_radius => 3000000
         ) OVER (ORDER BY place) AS cid
  FROM population
)
SELECT cid,
       string_agg(place, ', ' ORDER BY place) AS places,
       ST_Collect(geom ORDER BY place) AS cluster_geom
FROM clustered
GROUP BY cid
ORDER BY cid;
```

Output

```
0 | Melbourne, Sydney | MULTIPOINT((144.96 -37.81),(151.21 -33.87))
1 | Boston, New York | MULTIPOINT((-71.06 42.36),(-74.01 40.71))
2 | Osaka, Tokyo | MULTIPOINT((135.5 34.69),(139.69 35.68))
3 | London, Paris | MULTIPOINT((-0.13 51.51),(2.35 48.86))
```

Figure



Se även

ST_ClusterDBSCAN, ST_ClusterIntersectingWin, ST_ClusterWithinWin, ST_ClusterIntersecting, ST_Cluster
ST_Subdivide, ST_Force3D, ST_Force4D,

7.17.7 ST_ClusterWithin

ST_ClusterWithin — Aggregatfunktion som klustrar geometrier efter separationsavstånd.

Synopsis

geometry[] **ST_ClusterWithin**(geometry set g, float8 distance);

Beskrivning

En aggregerad funktion som returnerar en array av GeometryCollections, där varje samling är ett kluster som innehåller vissa indatageometrier. Klustring delar in indatageometrierna i uppsättningar där varje geometri ligger inom det angivna *avståndet* från minst en annan geometri i samma kluster. Avstånden är kartesiska avstånd i SRID:ens enheter.

ST_ClusterWithin är likvärdigt med att köra ST_ClusterDBSCAN med `minpoints=> 0`.

Tillgänglighet: 2.2.0

Denna metod stöder cirkulära strängar och kurvor.

Exempel

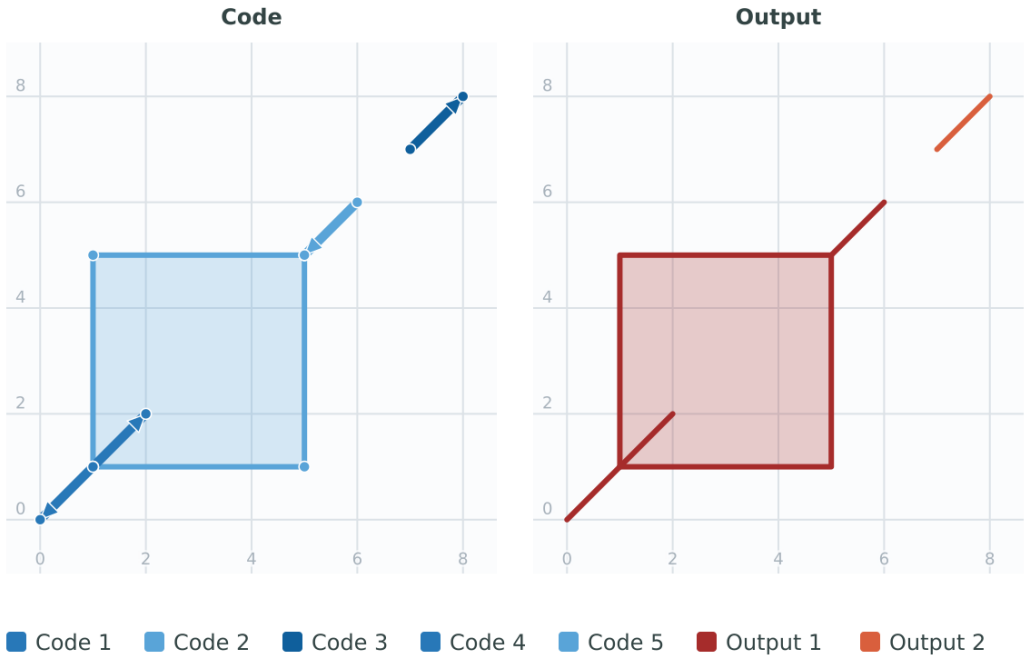
Code

```
WITH testdata AS
  (SELECT unnest(ARRAY['LINESTRING (0 0,1 1)'::geometry,
                'LINESTRING (5 5,4 4)'::geometry,
                'LINESTRING (6 6,7 7)'::geometry,
                'LINESTRING (0 0,-1 -1)'::geometry,
                'POLYGON ((0 0,4 0,4 4,0 4,0 0))'::geometry]) AS geom)

SELECT unnest(ST_ClusterWithin(geom, 1.4)) FROM testdata;
```

Output

```
GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),LINESTRING(5 5,4 4),LINESTRING(0 0,-1 -1),POLYGON((0 0,4 0,4 4,0 4,0 0)))
GEOMETRYCOLLECTION(LINESTRING(6 6,7 7))
```

Figure



Se även

ST_ClusterWithinWin, ST_ClusterDBSCAN, ST_ClusterIntersecting, ST_ClusterIntersectingWin

7.17.8 ST_ClusterWithinWin

ST_ClusterWithinWin — Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri, klustring med hjälp av separationsavstånd.

Synopsis

integer **ST_ClusterWithinWin**(geometry winset geom, float8 distance);

Beskrivning

En fönsterfunktion som returnerar ett klusternummer för varje indatageometri. Klusterindelningen delar in geometrierna i uppsättningar där varje geometri ligger inom det angivna `avståndet` från minst en annan geometri i samma kluster. Avstånden är kartesiska avstånd i SRID-enheterna.

ST_ClusterWithinWin är likvärdigt med att köra ST_ClusterDBSCAN med `minpoints=> 0`.

Tillgänglighet: 3.4.0

Denna metod stöder cirkulära strängar och kurvor.

Exempel

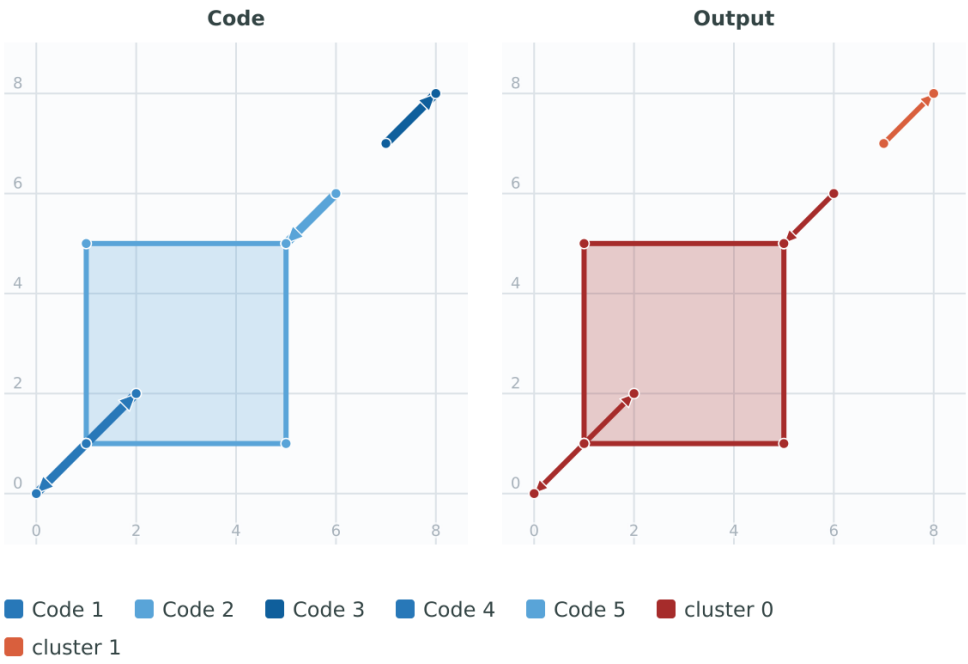
Code

```
WITH testdata AS (
  SELECT id, geom::geometry FROM (
  VALUES  (1, 'LINESTRING (0 0,1 1)'),
          (2, 'LINESTRING (5 5,4 4)'),
          (3, 'LINESTRING (6 6,7 7)'),
          (4, 'LINESTRING (0 0,-1 -1)'),
          (5, 'POLYGON ((0 0,4 0,4 4,0 4,0 0))')) AS t(id, geom)
)
SELECT id,
  geom,
  ST_ClusterWithinWin(geom, 1.4) OVER () AS cluster
FROM testdata;
```

Output

```
id |          st_astext            | cluster
----+-------------------------------+---------
  1 | LINESTRING(0 0,1 1)           |       0
  2 | LINESTRING(5 5,4 4)           |       0
  3 | LINESTRING(6 6,7 7)           |       1
  4 | LINESTRING(0 0,-1 -1)         |       0
  5 | POLYGON((0 0,4 0,4 4,0 4,0 0)) |       0
```

Figure



Se även

ST_ClusterWithin, ST_ClusterDBSCAN, ST_ClusterIntersecting, ST_ClusterIntersectingWin,

7.18 Funktioner för avgränsande box

7.18.1 Box2D

Box2D — Returnerar en BOX2D som representerar 2D-utbredningen av en geometri.

Synopsis

box2d **Box2D**(geometry geom);

Beskrivning

Returnerar en box2d som representerar geometrins 2D-utbredning.
Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

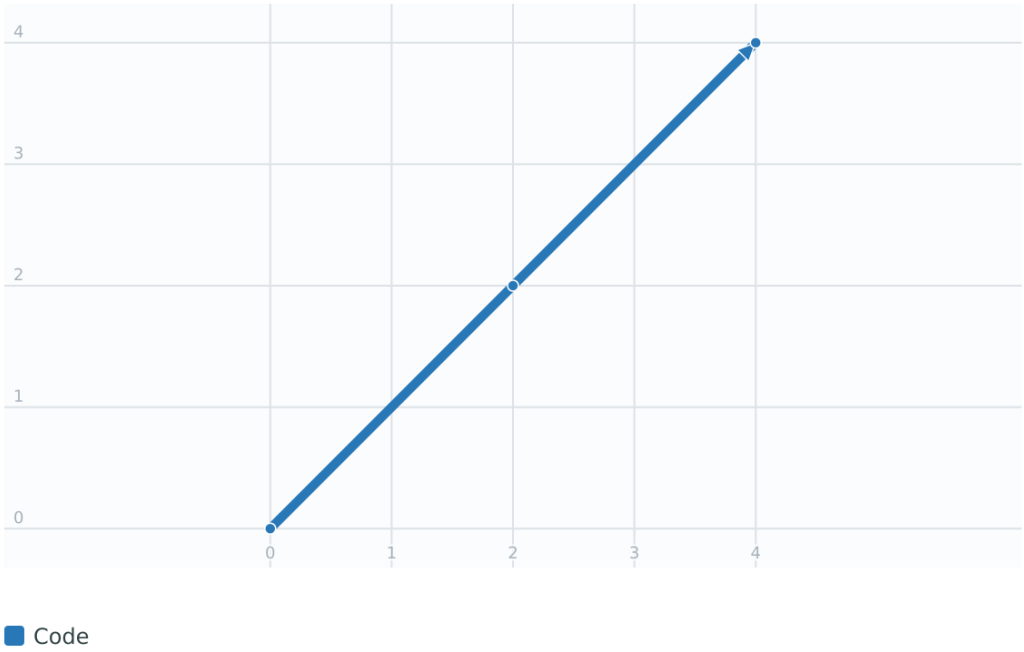
Code

```
SELECT Box2D('LINESTRING(1 2,3 4,5 6)'::geometry);
```

Output

```
BOX(1 2,5 6)
```

Figure



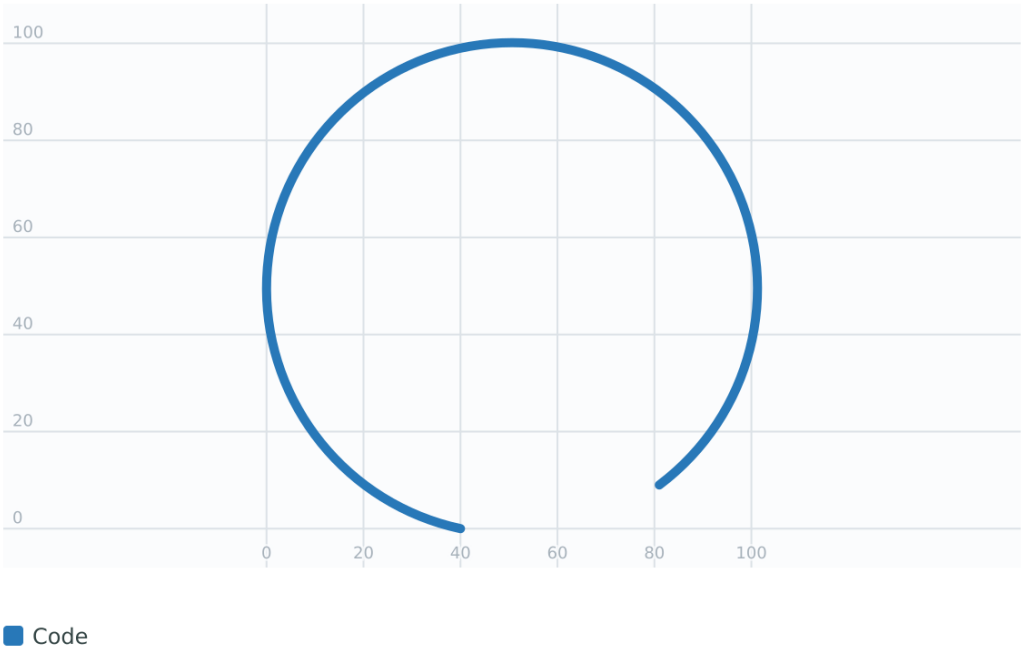
Code

```
SELECT Box2D('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'::geometry);
```

Output

```
BOX(220186.99512189245 150406,220288.24878054656 150506.12682932706)
```

Figure



Se även

Box3D, ST_GeomFromText

7.18.2 Box3D

Box3D — Returnerar en BOX3D som representerar 3D-utbredningen av en geometri.

Synopsis

box3d **Box3D**(geometry geom);

Beskrivning

Returnerar en box3d som representerar geometrins 3D-utbredning.
Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

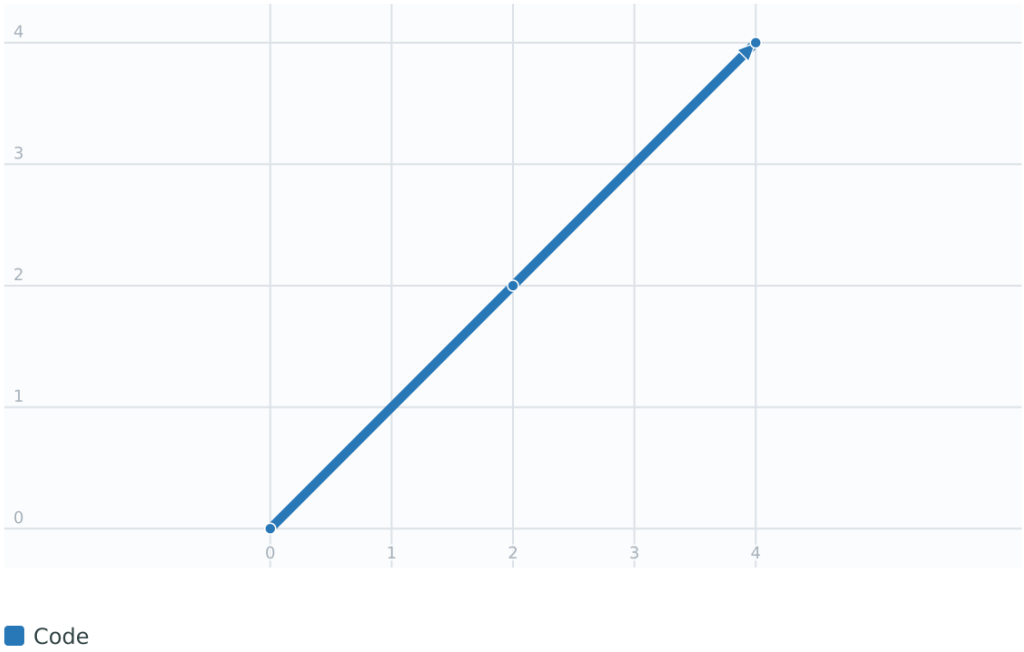
Code

```
SELECT Box3D('LINESTRING(1 2 3,3 4 5,5 6 5)'::geometry);
```

Output

```
BOX3D(1 2 3,5 6 5)
```

Figure



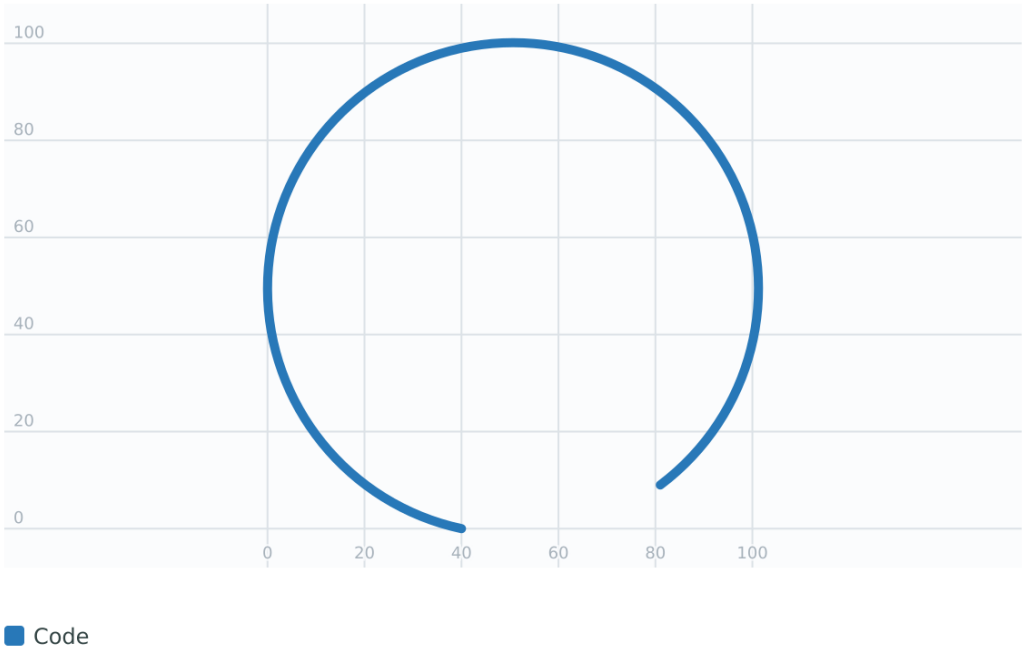
Code

```
SELECT Box3D('CIRCULARSTRING(220268 150415 1,220227 150505 1,220227 150406 1)'::geometry);
```

Output

```
BOX3D(220186.99512189245 150406 1,220288.24878054656 150506.12682932706 1)
```

Figure



Se även

Box2D, ST_GeomFromEWKT

7.18.3 ST_EstimatedExtent

ST_EstimatedExtent — Returnerar den uppskattade omfattningen av en spatial tabell.

Synopsis

box2d **ST_EstimatedExtent**(text schema_name, text table_name, text geocolumn_name, boolean parent_only);
box2d **ST_EstimatedExtent**(text schema_name, text table_name, text geocolumn_name);
box2d **ST_EstimatedExtent**(text table_name, text geocolumn_name);

Beskrivning

Returnerar den beräknade omfattningen av en spatial tabell som en box2d. Det aktuella schemat används om det inte anges. Den uppskattade omfattningen hämtas från geometrikolumnens statistik. Detta är vanligtvis mycket snabbare än att beräkna den exakta omfattningen av tabellen med ST_Extent eller ST_3DExtent.

Standardbeteendet är att även använda statistik som samlats in från underordnade tabeller (tabeller med INHERITS) om sådan finns tillgänglig. Om `parent_only` är satt till TRUE används endast statistik för den angivna tabellen och underordnade tabeller ignoreras.

För PostgreSQL> = 8.0.0 samlas statistik in av VACUUM ANALYZE och resultatomfattningen kommer att vara cirka 95% of den faktiska. För PostgreSQL <8.0.0 samlas statistik genom att köra `update_geometry_()` och resultatomfattningen är exakt.

Notera
Om det inte finns någon statistik (tom tabell eller ingen ANALYZE anropad) returnerar denna funktion NULL. Före version 1.5.4 kastades ett undantag i stället.

Notera
Escaping av namn för tabeller och/eller namnrymder som innehåller specialtecken och citattecken kan kräva särskild hantering. En användare noterar: För scheman och tabeller, använd identifieringsregler för escaping för att producera en dubbelciterad sträng och ta sedan bort det första och sista dubbelciteringstecknet. För geometrikolumner, skicka som de är.”

Tillgänglighet: 1.0.0

Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Estimated_Extent.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_EstimatedExtent('ny', 'edges', 'geom');
```

Output

```
st_estimatedextent
----------------------------------------------------
 BOX(-8877653 4912316,-8010225.5 5589284)
(1 row)
```

Code

```
SELECT ST_EstimatedExtent('feature_poly', 'geom');
```

Output

```
st_estimatedextent
----------------------------------------------------------------------
 BOX(-124.659652709961 24.6830825805664,-67.7798080444336 49.0012092590332)
(1 row)
```

Se även

ST_Extent, ST_3DExtent

7.18.4 ST_Expand

ST_Expand — Returnerar en bounding box som expanderats från en annan bounding box eller en geometri.

Synopsis

geometry **ST_Expand**(geometry geom, float units_to_expand);
geometry **ST_Expand**(geometry geom, float dx, float dy, float dz=0, float dm=0);
box2d **ST_Expand**(box2d box, float units_to_expand);
box2d **ST_Expand**(box2d box, float dx, float dy);
box3d **ST_Expand**(box3d box, float units_to_expand);
box3d **ST_Expand**(box3d box, float dx, float dy, float dz=0);

Beskrivning

Returnerar en bounding box som expanderats från indatans bounding box, antingen genom att ange ett enda avstånd med vilket boxen ska expanderas på båda axlarna, eller genom att ange ett expansionsavstånd för varje axel. Använder dubbel precision. Kan användas för avståndsfrågor eller för att lägga till ett bounding box-filter i en fråga för att dra nytta av ett spatialt index.

Förutom den version av ST_Expand som accepterar och returnerar en geometri, finns det varianter som accepterar och returnerar datatyperna box2d och box3d.

Avstånden anges i enheterna i det spatiala referenssystemet för inmatningen.

ST_Expand liknar ST_Buffer, men medan buffring expanderar en geometri i alla riktningar, expanderar ST_Expand den avgränsande rutan längs varje axel.

Notera
Före version 1.3 användes ST_Expand tillsammans med ST_Distance för att göra indexerbara avståndsfrågor. Till exempel `geom && ST_Expand('POINT(10 20)', 10) AND ST_Distance(geom, 'POINT(10 20)') < 10`. Detta har ersatts av den enklare och mer effektiva funktionen ST_DWithin.

Tillgänglighet: 1.5.0 ändrat beteende för att mata ut dubbel precision istället för float4-koordinater.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Förbättrad: 2.3.0 stöd har lagts till för att expandera en ruta med olika belopp i olika dimensioner.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Notera
Exemplen nedan använder US National Atlas Equal Area (SRID=2163) som är en meterprojektion

10-meter expanded box around bbox of a linestring.

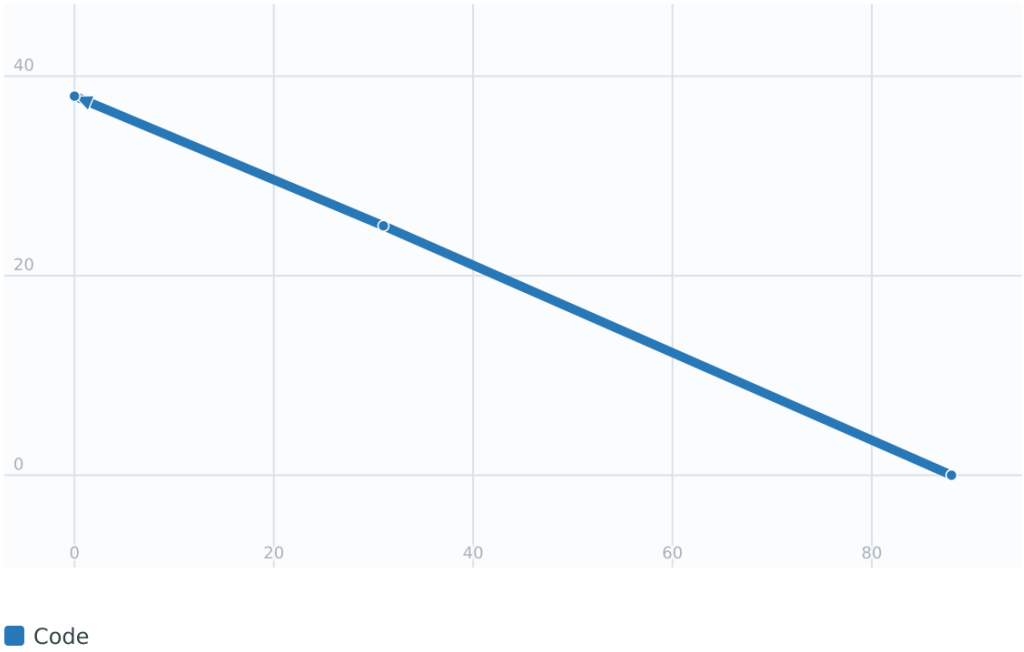
Code

```
SELECT CAST(ST_Expand(ST_GeomFromText('LINESTRING(2312980 110676,2312923 110701,2312892 ↩
    110714)', 2163), 10) As box2d);
```

Output

```
BOX(2312882 110666,2312990 110724)
```

Figure



This example expands a 3D box by 10 meters.

Code

```
SELECT ST_Expand(CAST('BOX3D(778783 2951741 1,794875 2970042.61545891 10)' AS box3d), 10)
```

Output

```
BOX3D(778773 2951731 -9,794885 2970052.61545891 20)
```

This example renders the text representation of a 10-meter expanded box around a point geometry.

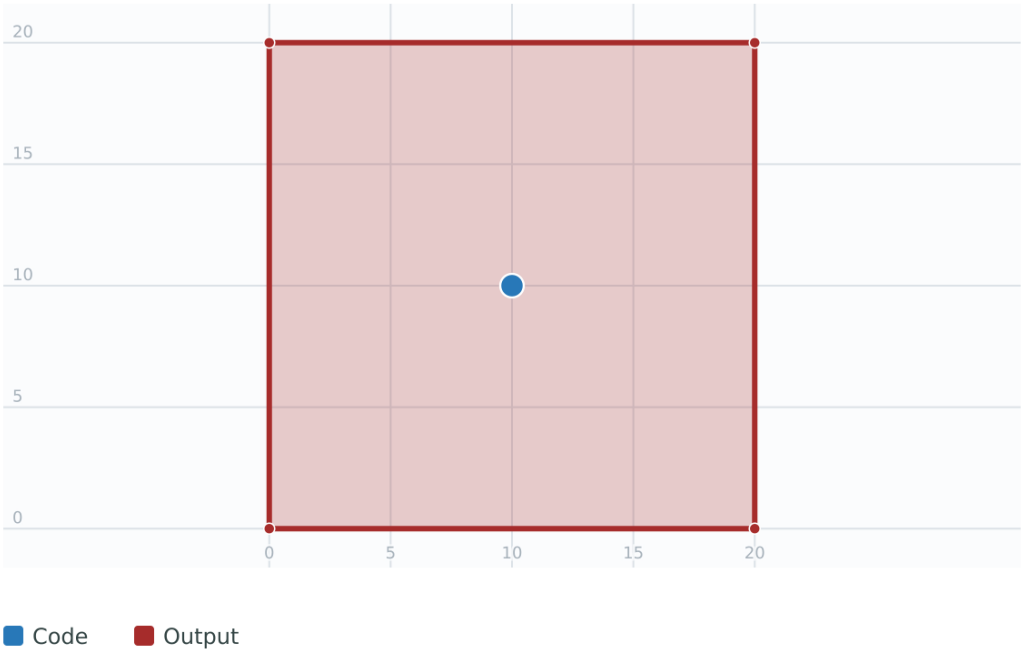
Code

```
SELECT ST_Expand('SRID=2163;POINT(2312980 110676)'::geometry, 10);
```

Output

```
SRID=2163;POLYGON((2312970 110666,2312970 110686,2312990 110686,2312990 110666,2312970 110666))
```

Figure



Se även

ST_Buffer, ST_DWithin, ST_SRID

7.18.5 ST_Extent

ST_Extent — Aggregerad funktion som returnerar geometriernas avgränsande box.

Synopsis

box2d **ST_Extent**(geometry set geomfield);

Beskrivning

En aggregerad funktion som returnerar en box2d bounding box som avgränsar en uppsättning geometrier.

Koordinaterna för begränsningsrutan är i det spatiala referenssystemet för de inmatade geometrierna.

ST_Extent är ett liknande koncept som Oracle Spatial/Locators SDO_AGGR_MBR.

Notera
ST_Extent returnerar rutor med endast X- och Y-ordinater även med 3D-geometrier. Om du vill returnera XYZ-ordinater använder du ST_3DExtent.

Notera
Det returnerade box3d-värdet innehåller inte någon SRID. Använd ST_SetSRID för att konvertera det till en geometri med SRID-metadata. SRID:en är densamma som för indatageometrierna.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Notera
I exemplen nedan används Massachusetts State Plane ft (SRID=2249)

Code

```
SELECT ST_Extent(geom) AS bextent FROM sometable;
```

Output

```
BOX(739651.875 2908247.25,794875.8125 2970042.75)
```

Return extent of each geometry category.

Code

```
SELECT ST_Extent(geom) as bextent
FROM sometable
GROUP BY category ORDER BY category;
```

Output

```
bextent                       |        name
------------------------------------------------------+----------------
 BOX(778783.5625 2951741.25,794875.8125 2970042.75) | A
 BOX(751315.8125 2919164.75,765202.6875 2935417.25) | B
 BOX(739651.875 2917394.75,756688.375 2935866)      | C
```

This example converts the extent back into a geometry and renders its extended text representation.

Code

```
SELECT ST_SetSRID(ST_Extent(geom), 2249) AS bextent FROM sometable;
```

Output

```
SRID=2249;POLYGON((739651.875 2908247.25,739651.875 2970042.75,794875.8125 2970042.75,
794875.8125 2908247.25,739651.875 2908247.25))
```

Se även

ST_EstimatedExtent, ST_3DExtent, ST_SetSRID

7.18.6 ST_3DExtent

ST_3DExtent — Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.

Synopsis

box3d **ST_3DExtent**(geometry set geomfield);

Beskrivning

En aggregerad funktion som returnerar en box3d (inkluderar Z-ordinat) avgränsningsbox som avgränsar en uppsättning geometrier.

Koordinaterna för begränsningsrutan är i det spatiala referenssystemet för de inmatade geometrierna.

Notera
Det returnerade box3d-värdet innehåller inte någon SRID. Använd ST_SetSRID för att konvertera det till en geometri med SRID-metadata. SRID:en är densamma som för indatageometrierna.

Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes.

Ändrad: 2.0.0 I tidigare versioner hette detta ST_Extent3D

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Code

```
SELECT ST_3DExtent(foo.geom) As b3extent
FROM (SELECT ST_MakePoint(x, y, z) As geom
    FROM generate_series(1, 3) As x
        CROSS JOIN generate_series(1, 2) As y
        CROSS JOIN generate_series(0, 2) As Z) As foo;
```

Output

```
BOX3D(1 1 0,3 2 2)
```

This example returns the extent of several elevated CircularStrings.

Code

```
SELECT ST_3DExtent(foo.geom) As b3extent
FROM (SELECT ST_Translate(ST_Force3DZ(ST_LineToCurve(ST_Buffer(ST_Point(x, y), 1))), 0, 0,  ↩
    z) As geom
    FROM generate_series(1, 3) As x
        CROSS JOIN generate_series(1, 2) As y
        CROSS JOIN generate_series(0, 2) As Z) As foo;
```

Output

```
BOX3D(0 0 0,4 3 2)
```

Se även

ST_Extent, ST_Force3DZ, ST_SetSRID

7.18.7 ST_MakeBox2D

ST_MakeBox2D — Skapar en BOX2D som definieras av två 2D-punktgeometrier.

Synopsis

box2d **ST_MakeBox2D**(geometry pointLowLeft, geometry pointUpRight);

Beskrivning

Skapar en box2d som definieras av två punktgeometrier. Detta är användbart för att göra intervallfrågor.

Exempel

Return all features that fall within or partly reside in a US National Atlas coordinate bounding box. It is assumed here that the geometries are stored with SRID=2163 (US National Atlas equal area).

Code

```
SELECT feature_id, feature_name, geom
FROM features
WHERE geom && ST_SetSRID(
  ST_MakeBox2D(
    ST_Point(-989502.1875, 528439.5625),
    ST_Point(-987121.375, 529933.1875)
  ),
  2163
);
```

Se även

ST_Point, ST_SetSRID, ST_SRID

7.18.8 ST_3DMakeBox

ST_3DMakeBox — Skapar en BOX3D som definieras av två 3D-punktgeometrier.

Synopsis

box3d **ST_3DMakeBox**(geometry point3DLowLeftBottom, geometry point3DUpRightTop);

Beskrivning

Skapar en box3d som definieras av två 3D-punktgeometrier.

Denna funktion stöder 3D och släpper inte z-index.

Ändrad: 2.0.0 I tidigare versioner brukade detta kallas ST_MakeBox3D

Exempel

Code

```
SELECT ST_3DMakeBox(ST_MakePoint(-989502.1875, 528439.5625, 10),
    ST_MakePoint(-987121.375 , 529933.1875, 10))
```

Output

```
BOX3D(-989502.1875 528439.5625 10,-987121.375 529933.1875 10)
```

Se även

ST_MakePoint, ST_SetSRID, ST_SRID

7.18.9 ST_XMax

ST_XMax — Returnerar X-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.

Synopsis

float **ST_XMax**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returnerar X-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.

Notera
Även om denna funktion endast är definierad för box3d fungerar den också för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_XMax('BOX3D(1 2 3,4 5 6)');
```

Output

```
4
```

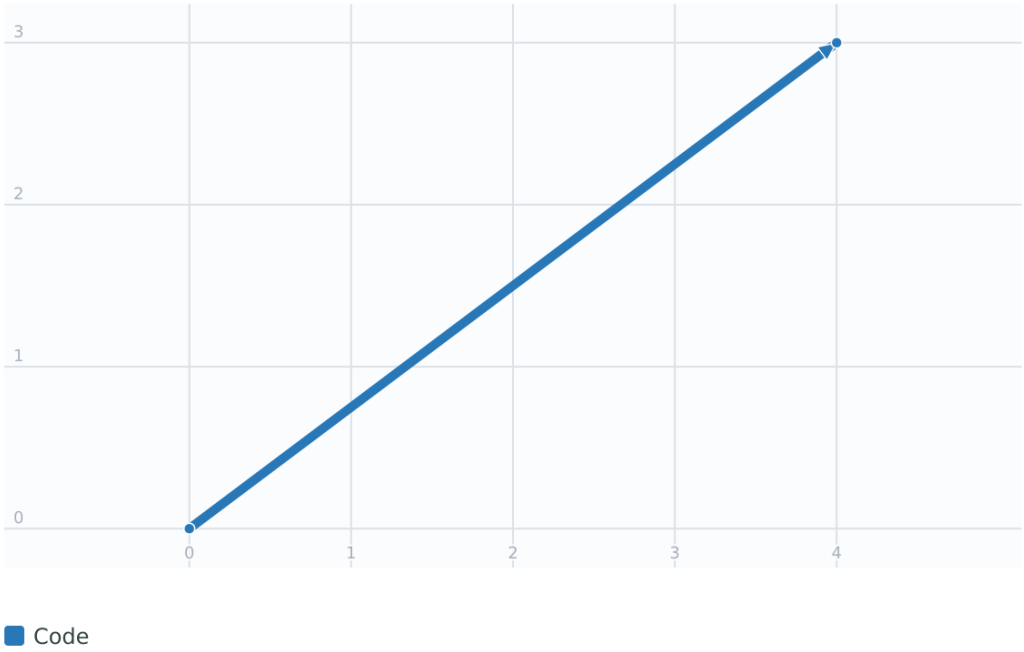
Code

```
SELECT ST_XMax('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
5
```

Figure



Code

```
SELECT ST_XMax(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
3
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_XMax('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

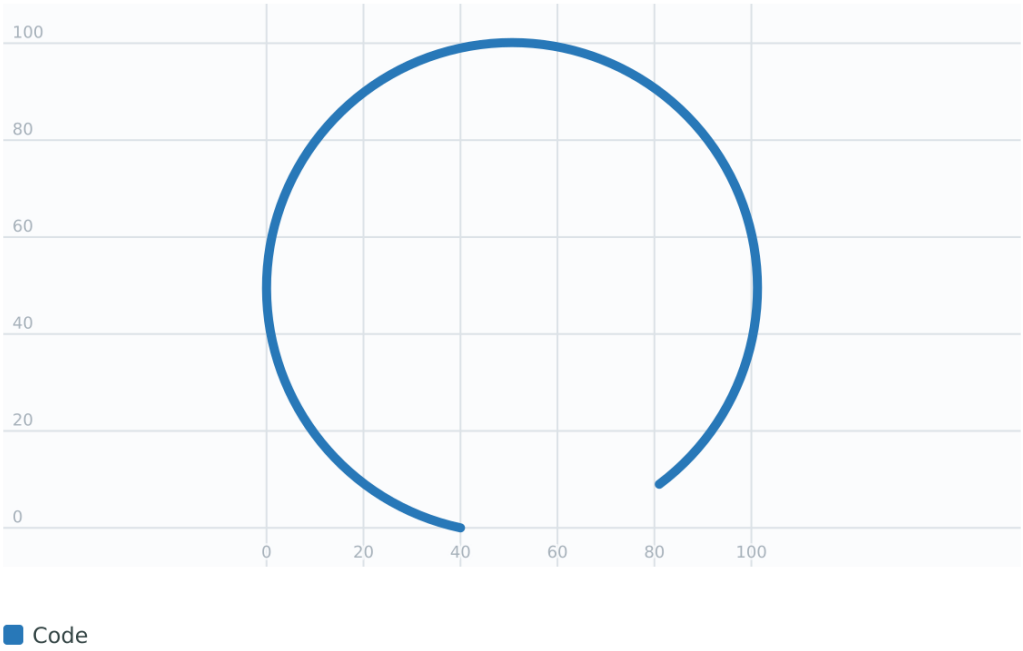
Code

```
SELECT ST_XMax('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
220288.248780547
```

Figure



Se även

ST_XMin, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

7.18.10 ST_XMin

ST_XMin — Returnerar X-minima för en 2D- eller 3D-begränsningsbox eller en geometri.

Synopsis

float **ST_XMin**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returnerar X-minima för en 2D- eller 3D-begränsningsbox eller en geometri.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_XMin('BOX3D(1 2 3,4 5 6)');
```

Output

```
1
```

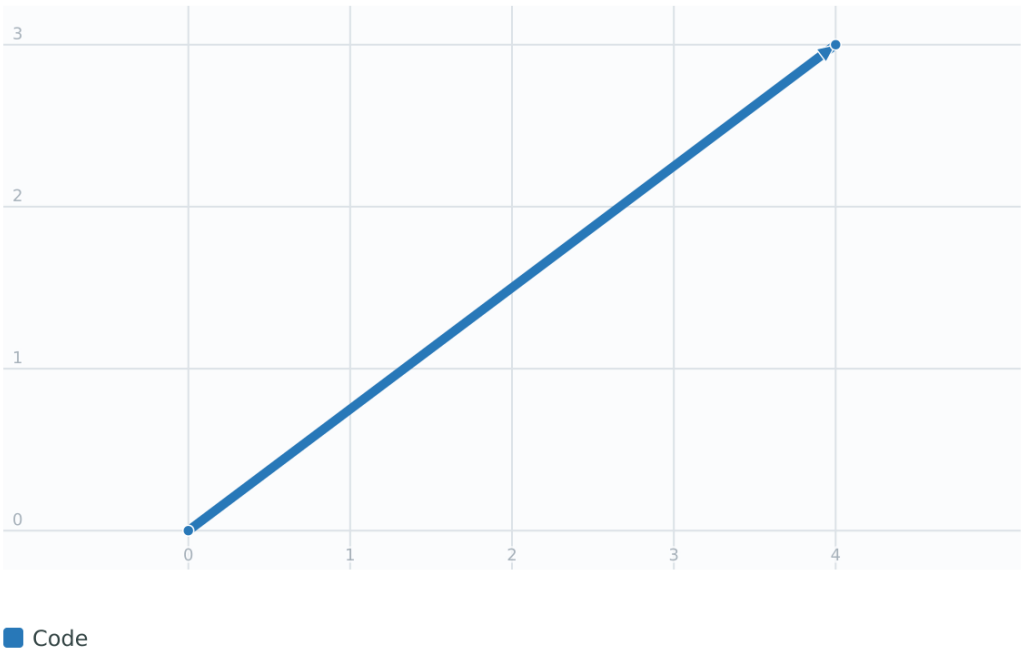
Code

```
SELECT ST_XMin('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
1
```

Figure



Code

```
SELECT ST_XMin(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
-3
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_XMin('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

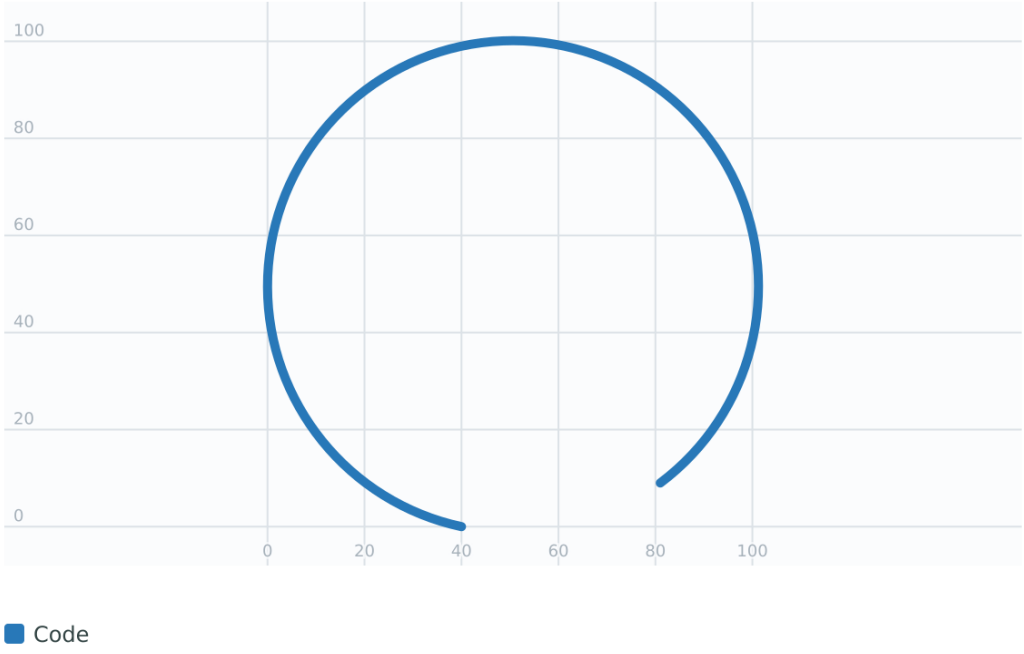
Code

```
SELECT ST_XMin('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
220186.995121892
```

Figure



Se även

ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

7.18.11 ST_YMax

ST_YMax — Returnerar Y-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.

Synopsis

float **ST_YMax**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returnerar Y-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_YMax('BOX3D(1 2 3,4 5 6)');
```

Output

```
5
```

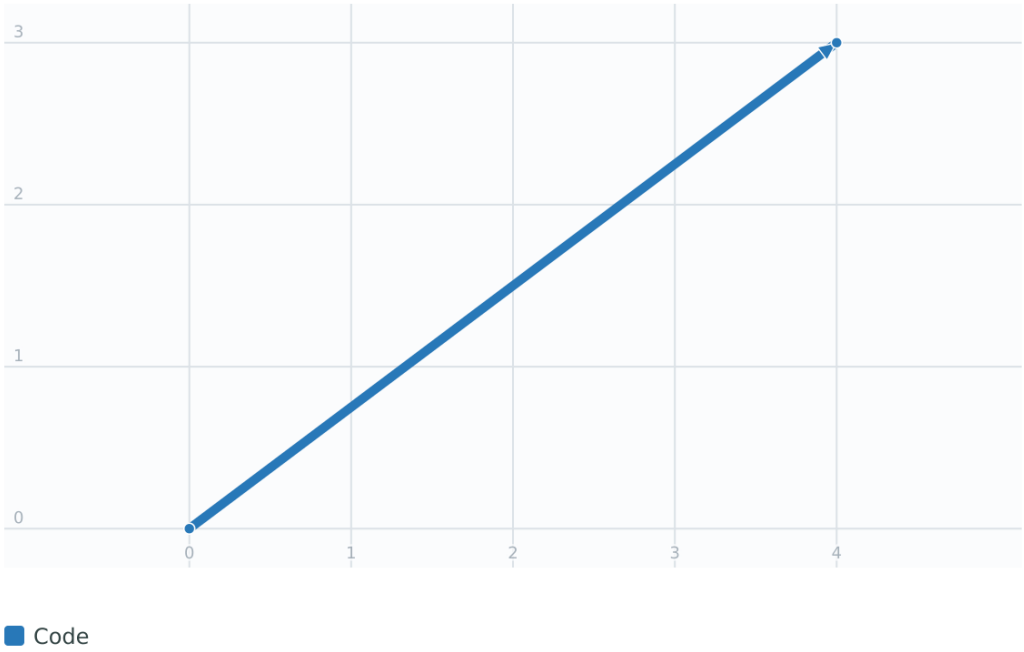
Code

```
SELECT ST_YMax('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
6
```

Figure



Code

```
SELECT ST_YMax(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
4
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_YMax('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

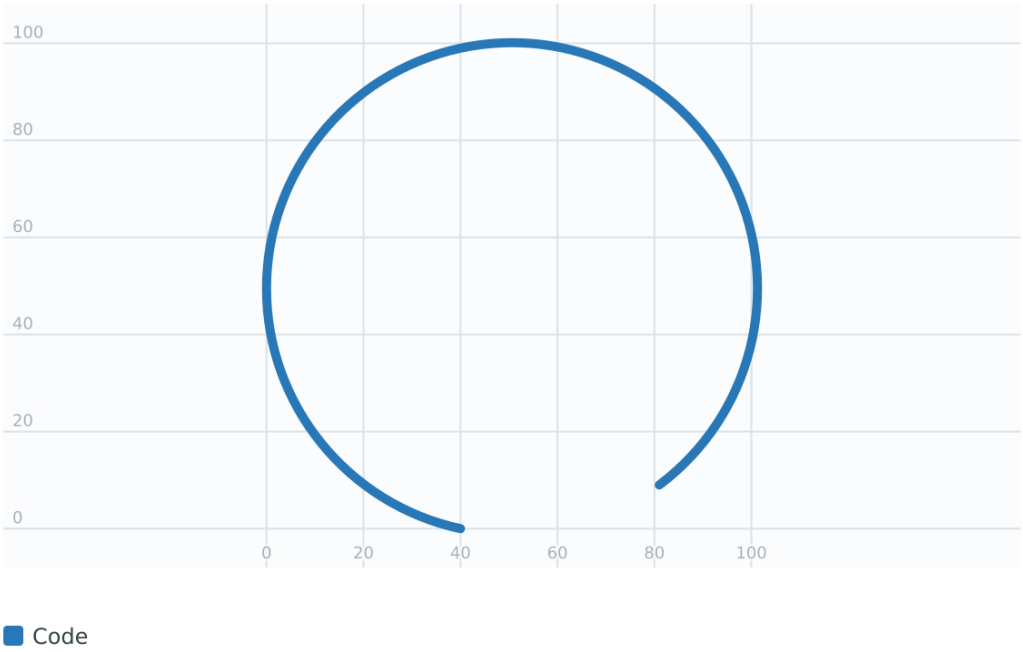
Code

```
SELECT ST_YMax('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
150506.126829327
```

Figure



Se även

ST_XMin, ST_XMax, ST_YMin, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

7.18.12 ST_YMin

ST_YMin — Returnerar Y-minima för en 2D- eller 3D-begränsningsbox eller en geometri.

Synopsis

float **ST_YMin**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returnerar Y-minima för en 2D- eller 3D-begränsningsbox eller en geometri.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_YMin('BOX3D(1 2 3,4 5 6)');
```

Output

```
2
```

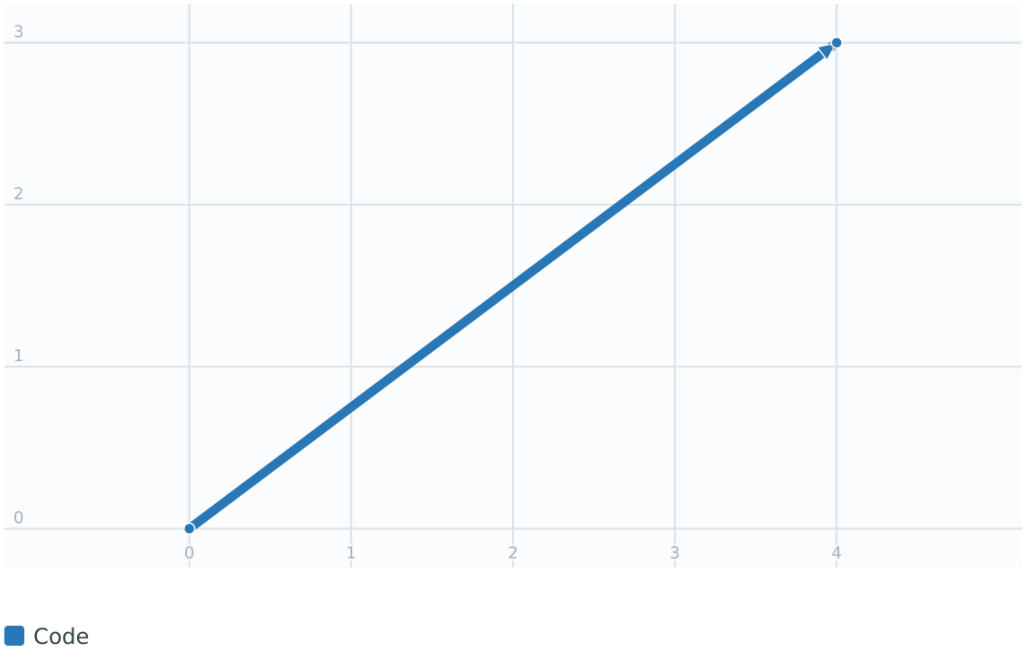
Code

```
SELECT ST_YMin('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
3
```

Figure



Code

```
SELECT ST_YMin(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
2
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_YMin('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

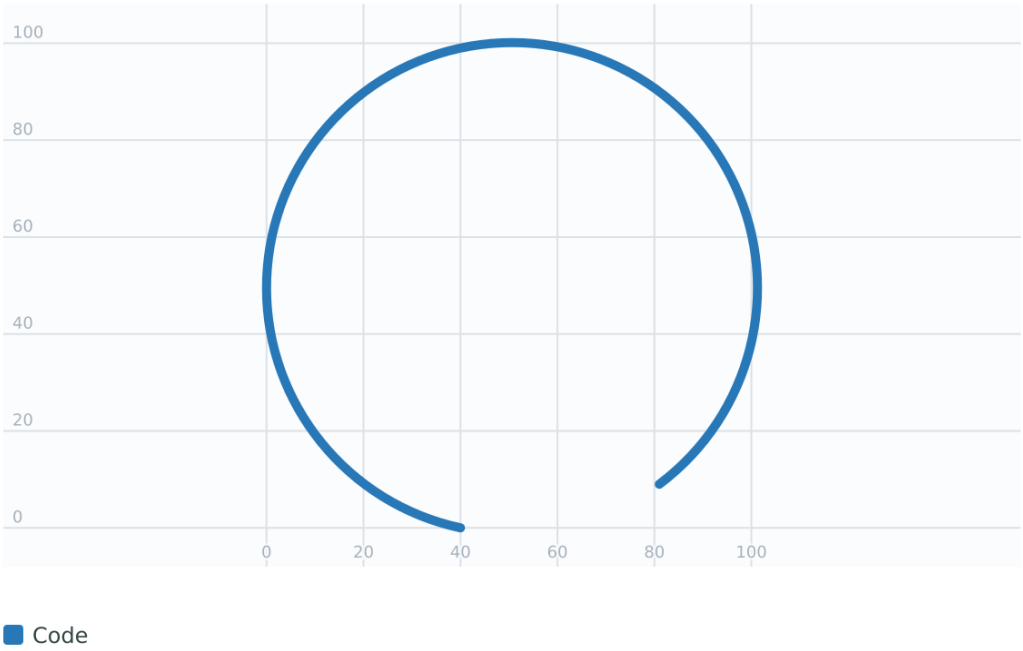
Code

```
SELECT ST_YMin('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
150406
```

Figure



Se även

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

7.18.13 ST_ZMax

ST_ZMax — Returnerar Z-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.

Synopsis

float **ST_ZMax**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returnerar Z-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_ZMax('BOX3D(1 2 3,4 5 6)');
```

Output

```
6
```

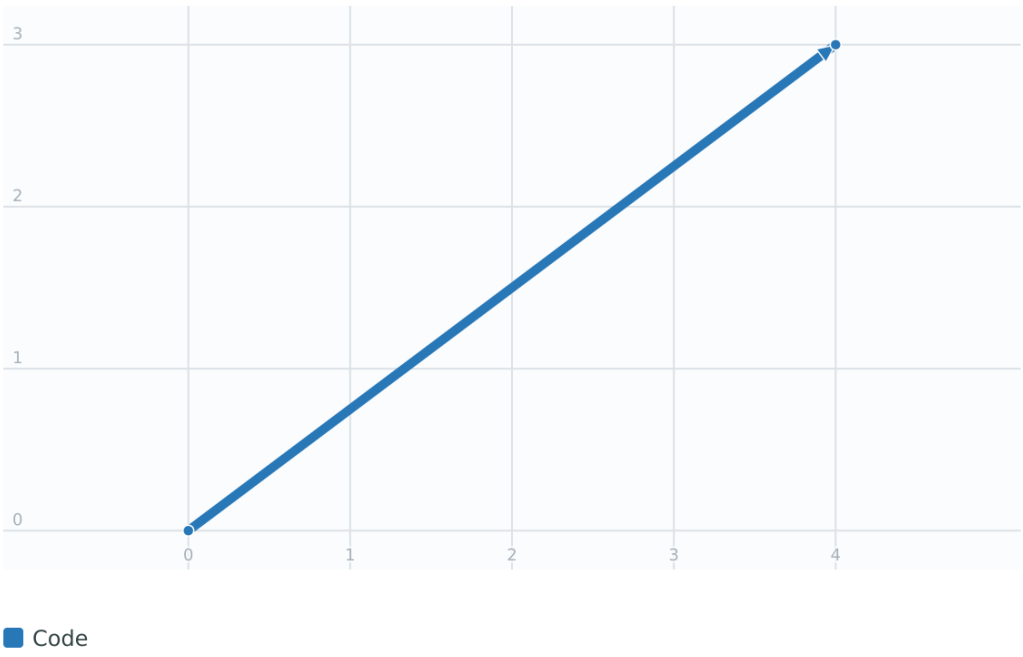
Code

```
SELECT ST_ZMax('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
7
```

Figure



Code

```
SELECT ST_ZMax('BOX3D(-3 2 1,3 4 1)');
```

Output

```
1
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_ZMax('LINESTRING(1 3 4,5 6 7)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

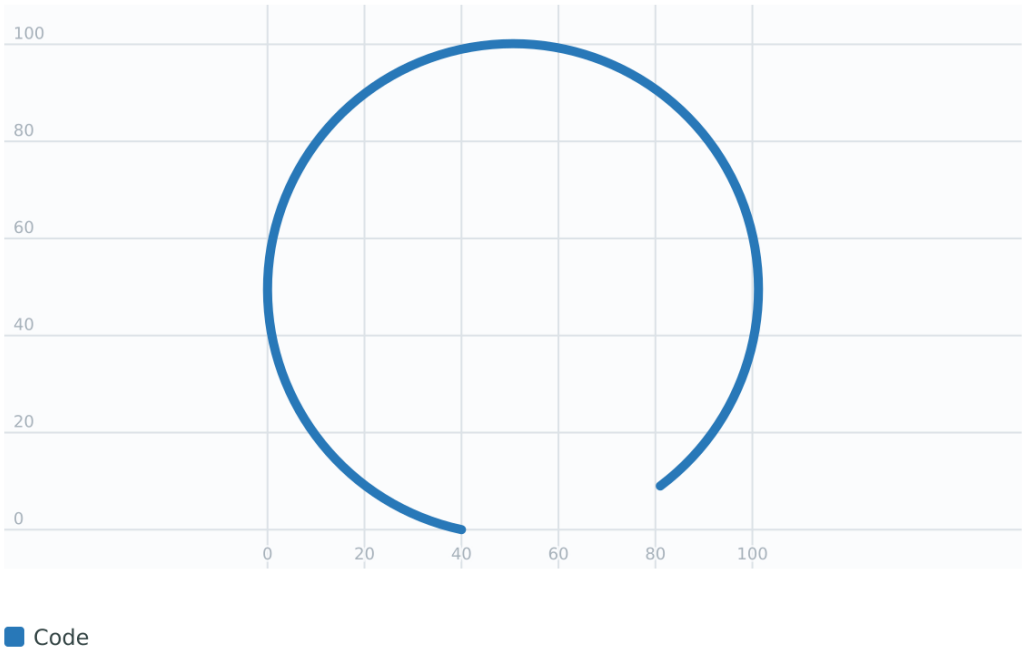
Code

```
SELECT ST_ZMax('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
3
```

Figure



Se även

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_MMin, ST_MMax

7.18.14 ST_ZMin

ST_ZMin — Returnerar Z-minima för en 2D- eller 3D-begränsningsbox eller en geometri.

Synopsis

float **ST_ZMin**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returnerar Z-minima för en 2D- eller 3D-begränsningsbox eller en geometri.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_ZMin('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

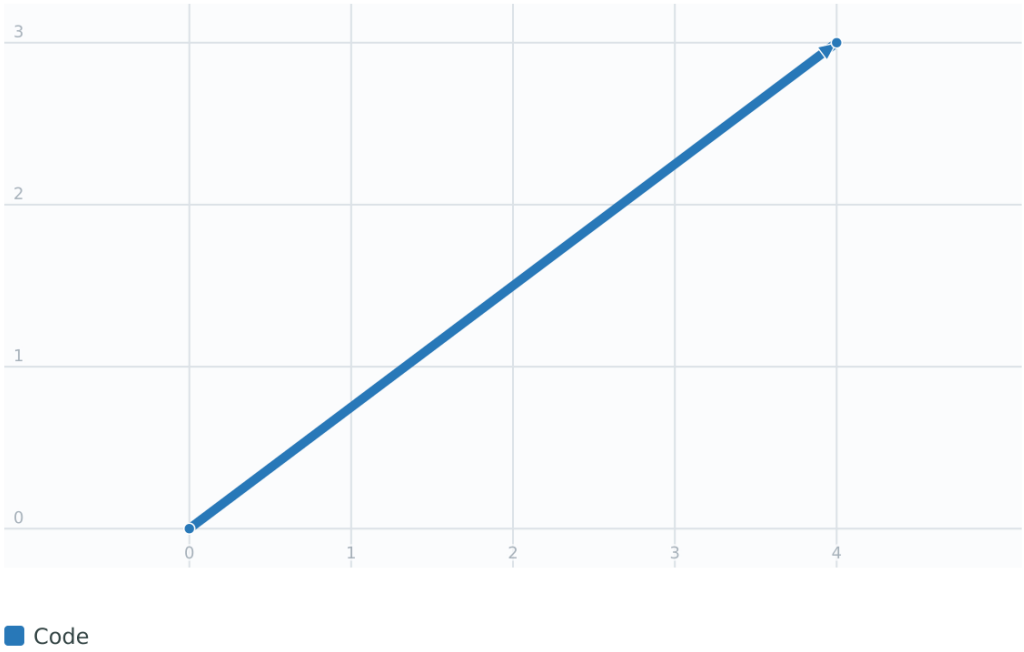
Code

```
SELECT ST_ZMin('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
4
```

Figure



Code

```
SELECT ST_ZMin('BOX3D(-3 2 1,3 4 1)');
```

Output

```
1
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_ZMin('LINESTRING(1 3 4,5 6 7)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

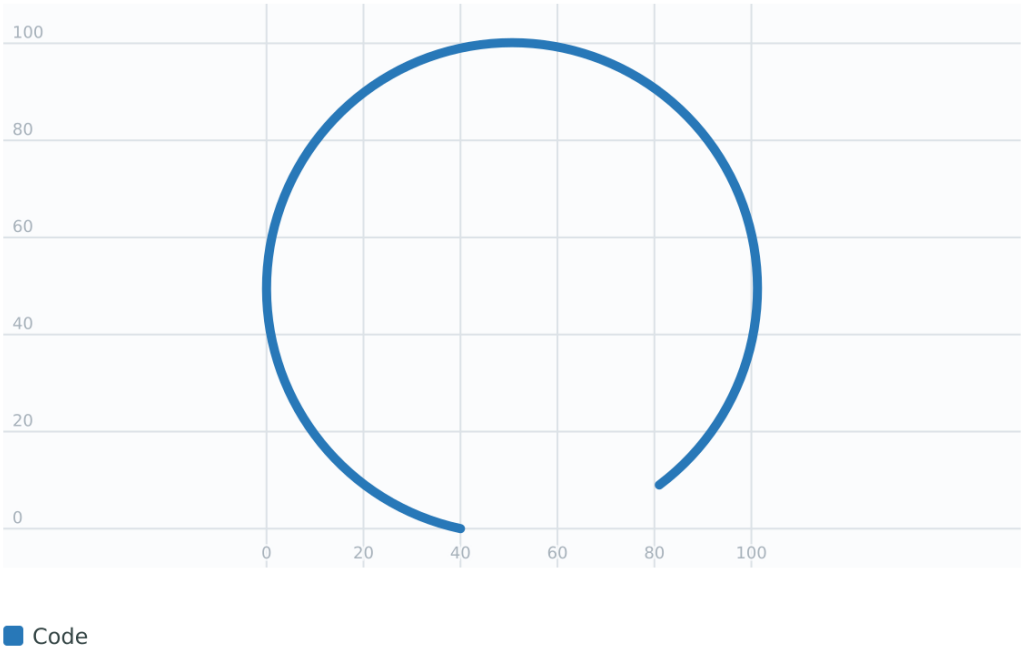
Code

```
SELECT ST_ZMin('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
1
```

Figure



Se även

ST_GeomFromEWKT, ST_GeomFromText, ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMax, ST_MMin, ST_MMax

7.18.15 ST_XSize

ST_XSize — Returns the X size of a 2D or 3D bounding box or a geometry.

Synopsis

float **ST_XSize**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returns the X size of a 2D or 3D bounding box or a geometry, computed as `ST_XMax(aGeomorBox2DorBox3D) - ST_XMin(aGeomorBox2DorBox3D)`.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Availability: 3.7.0

Exempel

Code

```
SELECT ST_XSize('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

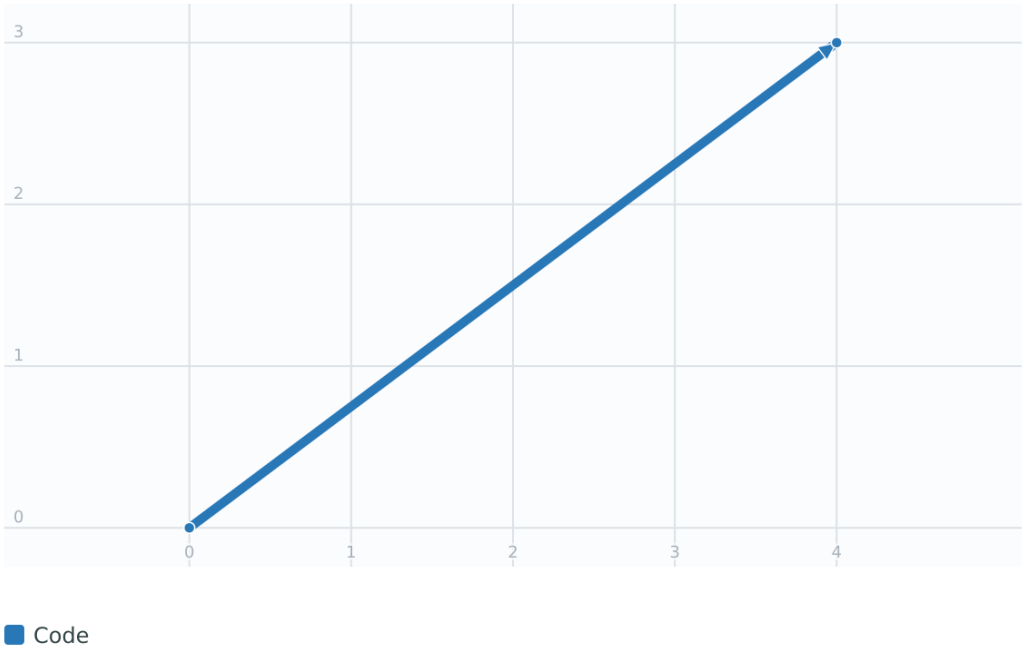
Code

```
SELECT ST_XSize('LINESTRING(1 3,5 6)'::geometry);
```

Output

```
4
```

Figure



Se även

ST_XMin, ST_XMax, ST_YSize, ST_ZSize, ST_MSize

7.18.16 ST_YSize

ST_YSize — Returns the Y size of a 2D or 3D bounding box or a geometry.

Synopsis

float **ST_YSize**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returns the Y size of a 2D or 3D bounding box or a geometry, computed as `ST_YMax(aGeomorBox2DorBox3D) - ST_YMin(aGeomorBox2DorBox3D)`.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Availability: 3.7.0

Exempel

Code

```
SELECT ST_YSize('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

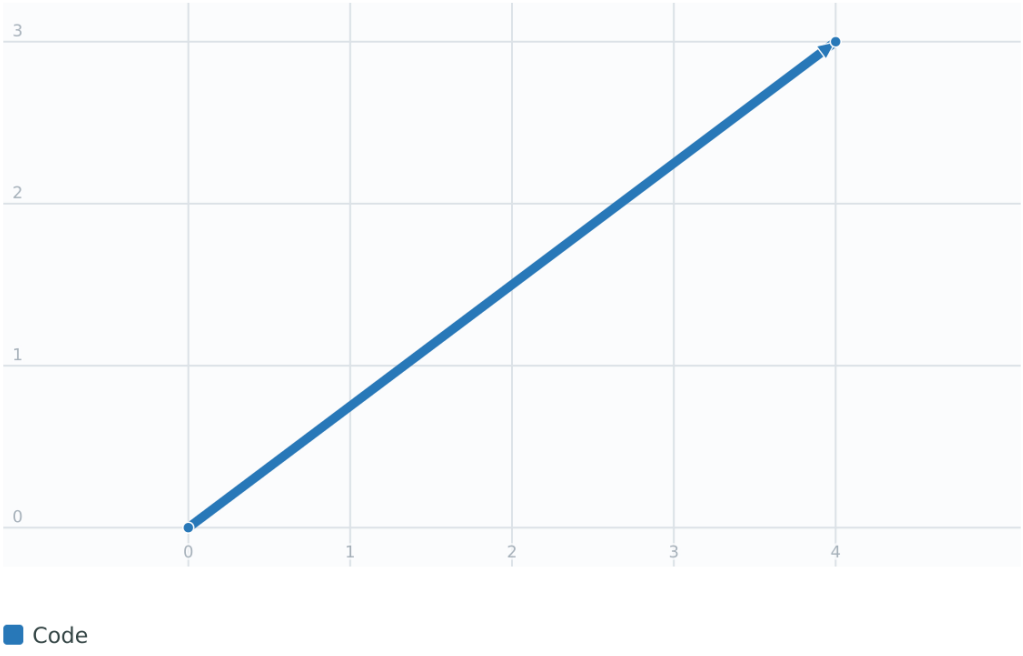
Code

```
SELECT ST_YSize('LINESTRING(1 3,5 6)'::geometry);
```

Output

```
3
```

Figure



Se även

ST_YMin, ST_YMax, ST_XSize, ST_ZSize, ST_MSize

7.18.17 ST_ZSize

ST_ZSize — Returns the Z size of a 2D or 3D bounding box or a geometry.

Synopsis

float **ST_ZSize**(box3d aGeomorBox2DorBox3D);

Beskrivning

Returns the Z size of a 2D or 3D bounding box or a geometry, computed as `ST_ZMax(aGeomorBox2DorBox3D) - ST_ZMin(aGeomorBox2DorBox3D)`.

Notera
Även om denna funktion endast är definierad för box3d, fungerar den även för box2d- och geometrivärden på grund av automatisk casting. Den kommer dock inte att acceptera en geometri- eller box2d-textrepresentation, eftersom dessa inte har automatisk casting.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Availability: 3.7.0

Exempel

Code

```
SELECT ST_ZSize('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

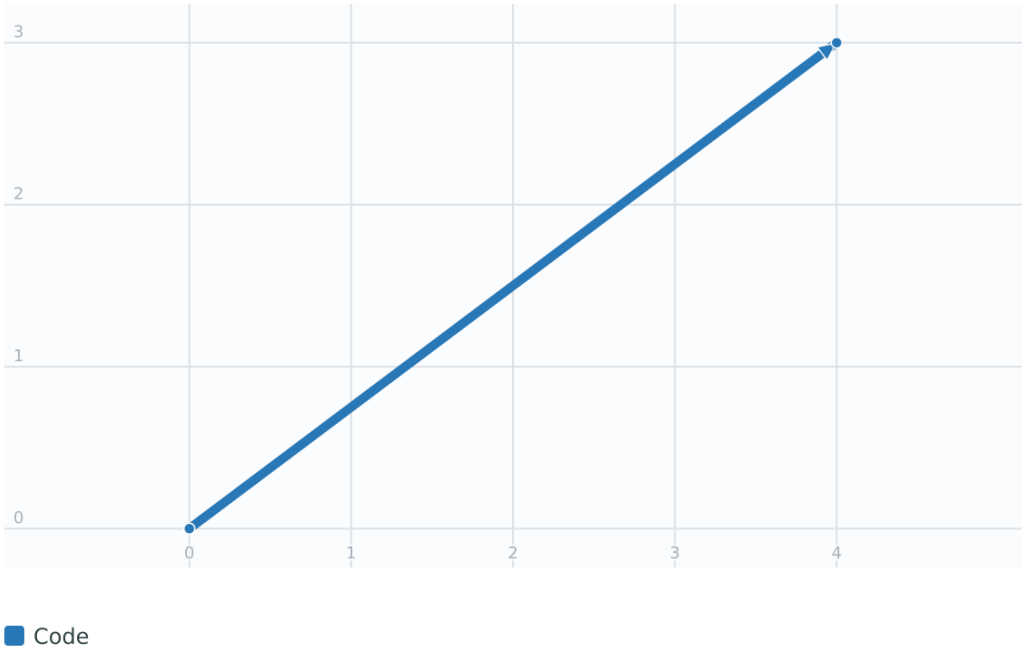
Code

```
SELECT ST_ZSize('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
3
```

Figure



Se även

ST_ZMin, ST_ZMax, ST_XSize, ST_YSize, ST_MSize

7.18.18 ST_MMin

ST_MMin — Returns the M minima of a geometry.

Synopsis

float **ST_MMin**(geometry geom);

Beskrivning

Returns the M minima of a geometry, or null if the geometry lacks M values.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_MMin('POINT M (1 2 3)');
```

Output

```
3
```

Se även

ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMax, ST_MSize

7.18.19 ST_MMax

ST_MMax — Returns the M maxima of a geometry.

Synopsis

float **ST_MMax**(geometry geom);

Beskrivning

Returns the M maxima of a geometry, or null if the geometry lacks M values.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT ST_MMax('POINT M (1 2 3)');
```

Output

```
3
```

Se även

ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MSize

7.18.20 ST_MSize

ST_MSize — Returns the M size of a geometry.

Synopsis

float **ST_MSize**(geometry geom);

Beskrivning

Returns the M size of a geometry, computed as `ST_MMax(geom) - ST_MMin(geom)`, or null if the geometry lacks M values.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna metod stöder cirkulära strängar och kurvor.

Availability: 3.7.0

Exempel

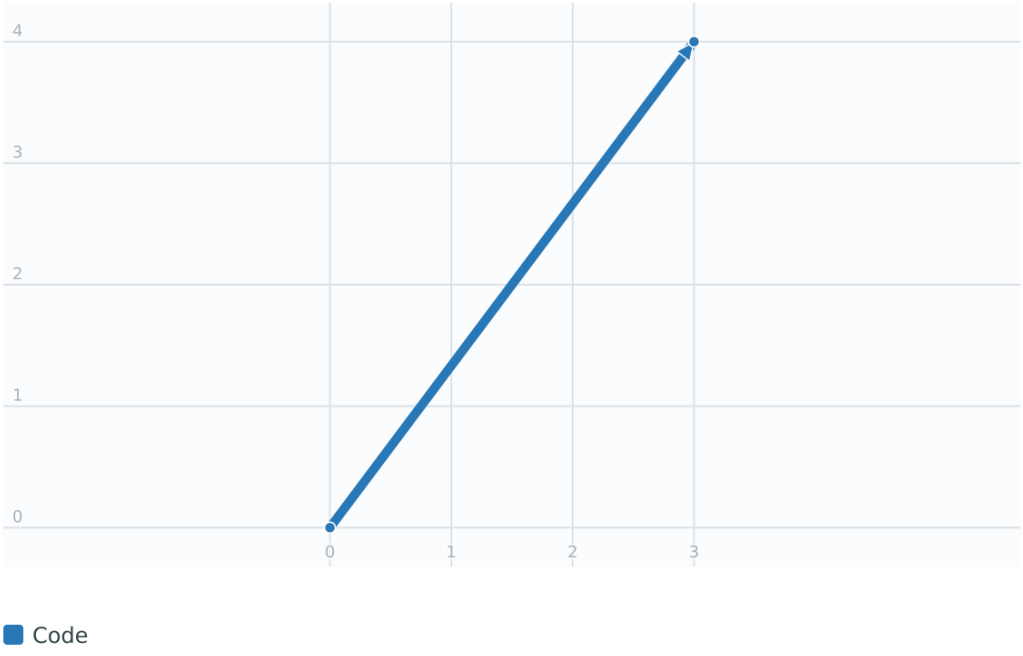
Code

```
SELECT ST_MSize('LINESTRING M (1 2 3,4 6 8)');
```

Output

```
5
```

Figure



Se även

ST_MMin, ST_MMax, ST_XSize, ST_YSize, ST_ZSize

7.19 Linjär referenstagning

7.19.1 ST_LineInterpolatePoint

ST_LineInterpolatePoint — Returnerar en punkt som interpolerats längs en linje på en fraktionerad plats.

Synopsis

geometry **ST_LineInterpolatePoint**(geometry a_linestring, float8 a_fraction);
geography **ST_LineInterpolatePoint**(geography a_linestring, float8 a_fraction, boolean use_spheroid = true);

Beskrivning

Returnerar en punkt som interpolerats längs en linje på en fraktionerad plats. Första argumentet måste vara en LINESTRING. Det andra argumentet är ett flyttal mellan 0 och 1 som representerar den fraktion av linjelängden där punkten ska placeras. Z- och M-värdena interpoleras om de finns.

Se ST_LineLocatePoint för att beräkna linjepositionen närmast en punkt.

Notera
Den här funktionen beräknar punkter i 2D och interpolerar sedan värden för Z och M, medan ST_3DLineInterpolatePoint beräknar punkter i 3D och endast interpolerar M-värdet.

Notera
Sedan version 1.1.1 interpolerar denna funktion även M- och Z-värden (om sådana finns), medan tidigare versioner satte dem till 0,0.

Tillgänglighet: 0.8.2, stöd för Z och M tillkom i 1.1.1

Ändrad: 2.1.0. Fram till 2.0.x hette detta ST_Line_Interpolate_Point.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

This example returns the point 20 percent along a line.

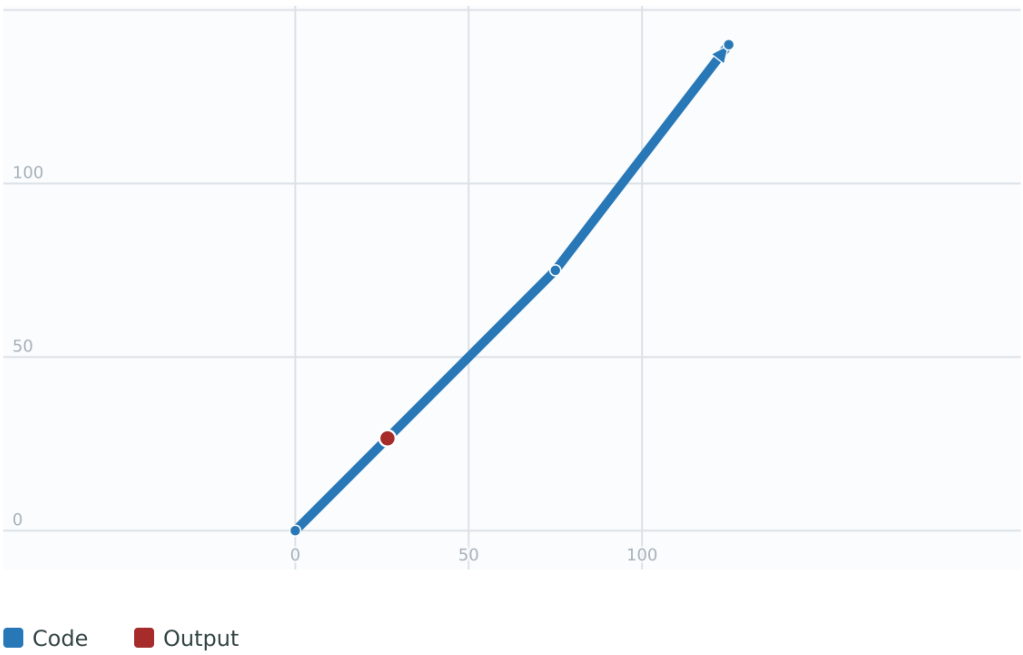
Code

```
SELECT ST_LineInterpolatePoint(
'LINESTRING(25 50,100 125,150 190)',
0.2);
```

Output

```
POINT(51.5974135047432 76.5974135047432)
```

Figure



Mittpunkten för en 3D-linje:

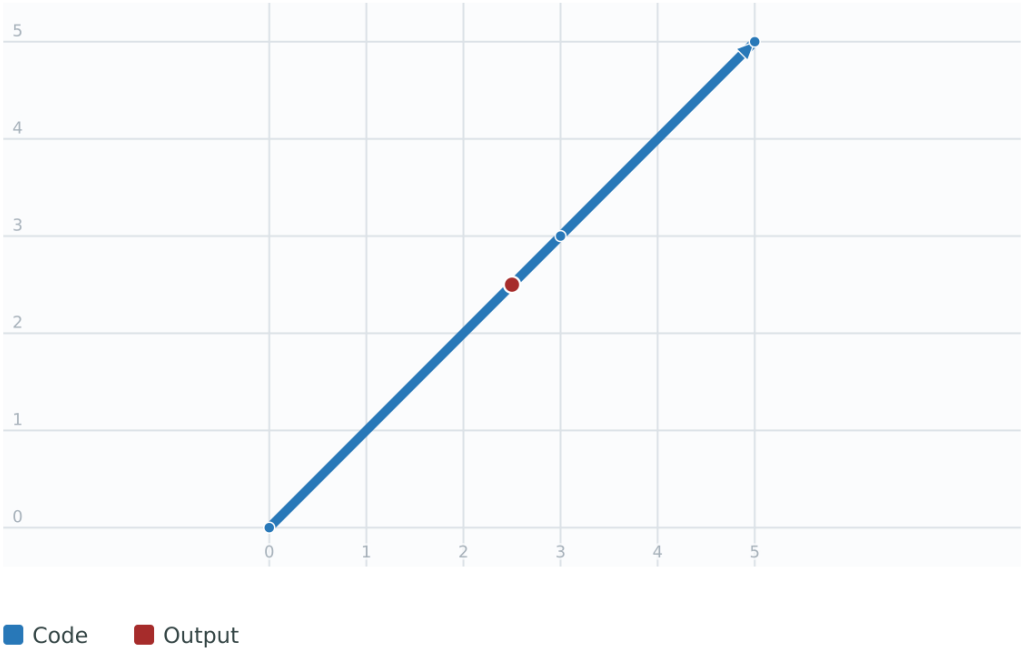
Code

```
SELECT ST_LineInterpolatePoint(
'LINESTRING(1 2 3,4 5 6,6 7 8)',
0.5);
```

Output

```
POINT(3.5 4.5 5.5)
```

Figure



Den punkt på en linje som ligger närmast en punkt:

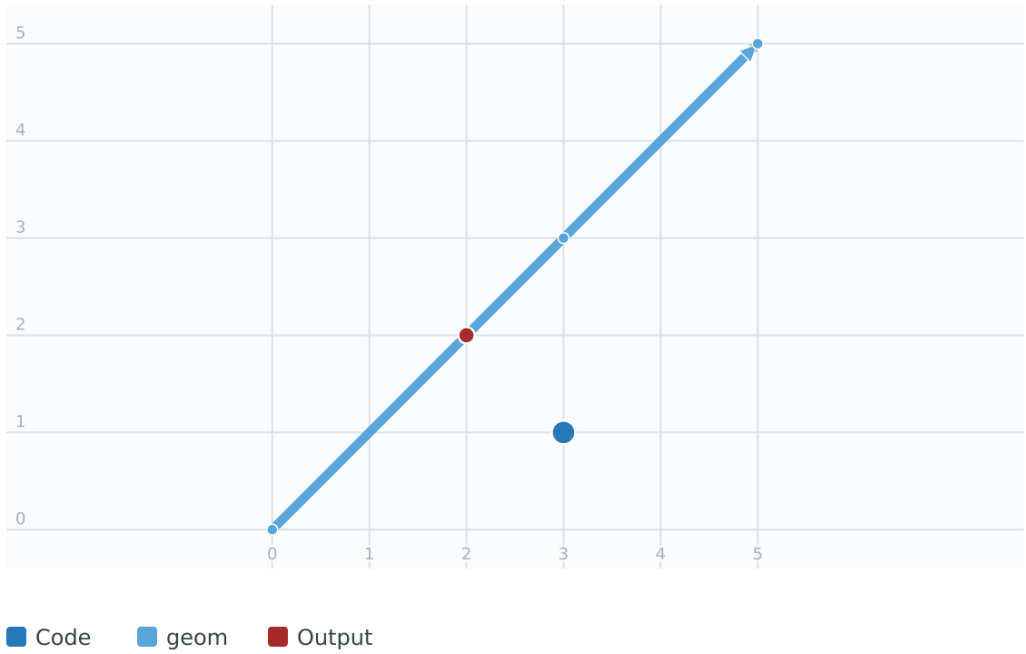
Code

```
SELECT ST_LineInterpolatePoint(
    line.geom,
    ST_LineLocatePoint(line.geom, 'POINT(4 3)'))
FROM (SELECT 'LINESTRING(1 2,4 5,6 7)'::geometry As geom) AS line;
```

Output

```
POINT(3 4)
```

Figure

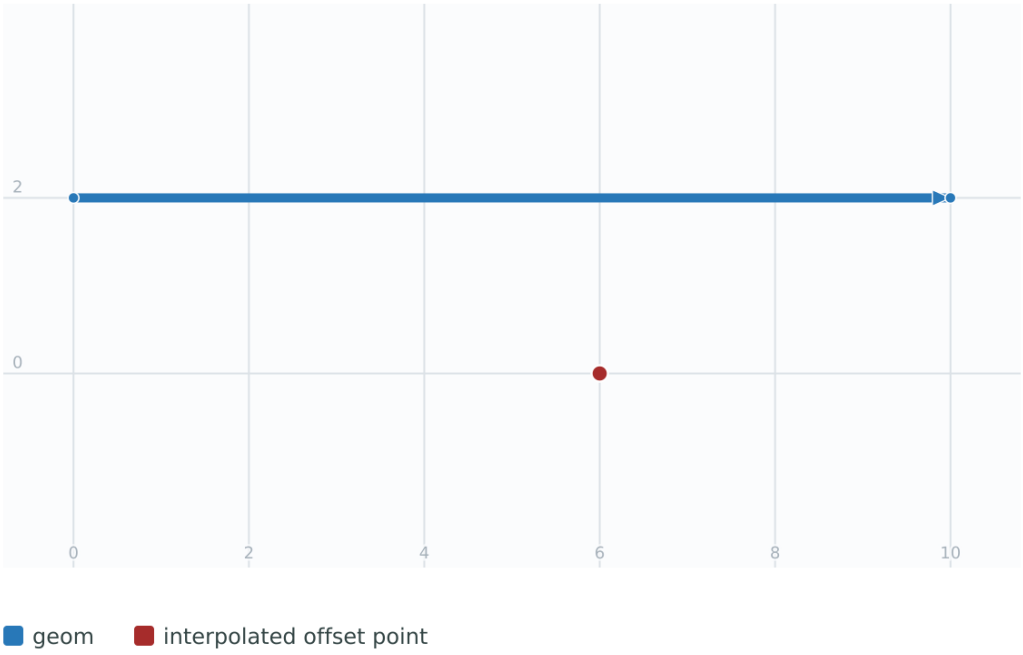


An interpolated point can be offset to the left or right of a line by using ST_OffsetCurve before interpolation. This is useful for locating an address point beside a street centerline.

Code

```
WITH street AS (
    SELECT 'LINESTRING(0 0, 10 0)'::geometry AS geom,
           100 AS from_num,
           200 AS to_num
)
SELECT
    ST_LineInterpolatePoint(
        ST_OffsetCurve(geom, -2),
        (160 - from_num)::float8 / (to_num - from_num)
    ) AS interpolated_offset_point
FROM street;
```

Figure



Se även

ST_LineInterpolatePoints, ST_3DLineInterpolatePoint, ST_LineLocatePoint, ST_OffsetCurve

7.19.2 ST_3DLineInterpolatePoint

ST_3DLineInterpolatePoint — Returnerar en punkt som interpolerats längs en 3D-linje på en fraktionerad plats.

Synopsis

geometry **ST_3DLineInterpolatePoint**(geometry a_linestring, float8 a_fraction);

Beskrivning

Returnerar en punkt som interpolerats längs en 3D-linje på en fraktionerad plats. Första argumentet måste vara en LINESTRING. Det andra argumentet är ett flyttal mellan 0 och 1 som representerar punktens läge som en fraktion av linjens längd. M-värdet interpoleras om det finns.

Notera
ST_LineInterpolatePoint beräknar punkter i 2D och interpolerar sedan värdena för Z och M, medan den här funktionen beräknar punkter i 3D och endast interpolerar M-värdet.

Tillgänglighet: 3.0.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Returpunkt 20% alång 3D-linje

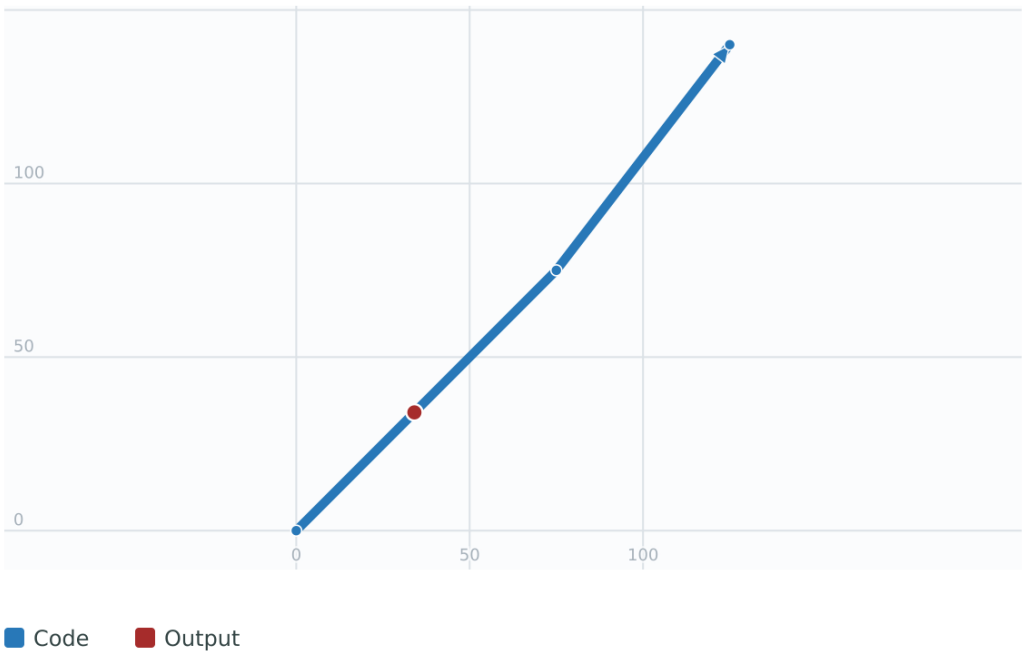
Code

```
SELECT ST_3DLineInterpolatePoint(
'LINESTRING(25 50 70,100 125 90,150 190 200)',
0.20);
```

Output

```
POINT Z (59.06758929108215 84.06758929108216 79.0846904776219)
```

Figure



Se även

ST_LineInterpolatePoint, ST_LineInterpolatePoints, ST_LineLocatePoint

7.19.3 ST_LineInterpolatePoints

ST_LineInterpolatePoints — Returnerar punkter interpolerade längs en linje med ett fraktionerat intervall.

Synopsis

geometry **ST_LineInterpolatePoints**(geometry a_linestring, float8 a_fraction, boolean repeat);
geography **ST_LineInterpolatePoints**(geography a_linestring, float8 a_fraction, boolean use_spheroid = true, boolean repeat = true);

Beskrivning

Returnerar en eller flera punkter interpolerade längs en linje med ett fraktionerat intervall. Det första argumentet måste vara en LINESTRING. Det andra argumentet är en float8 mellan 0 och 1 som representerar avståndet mellan punkterna som en fraktion av linjens längd. Om det tredje argumentet är false kommer högst en punkt att konstrueras (vilket är likvärdigt med ST_LineInterpolatePoint.)

Om resultatet har noll eller en punkt returneras det som en POINT. Om det har två eller fler punkter returneras det som en MULTIPOINT.

Tillgänglighet: 2.5.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Exempel

This example returns points every 20 percent along a two-dimensional line.

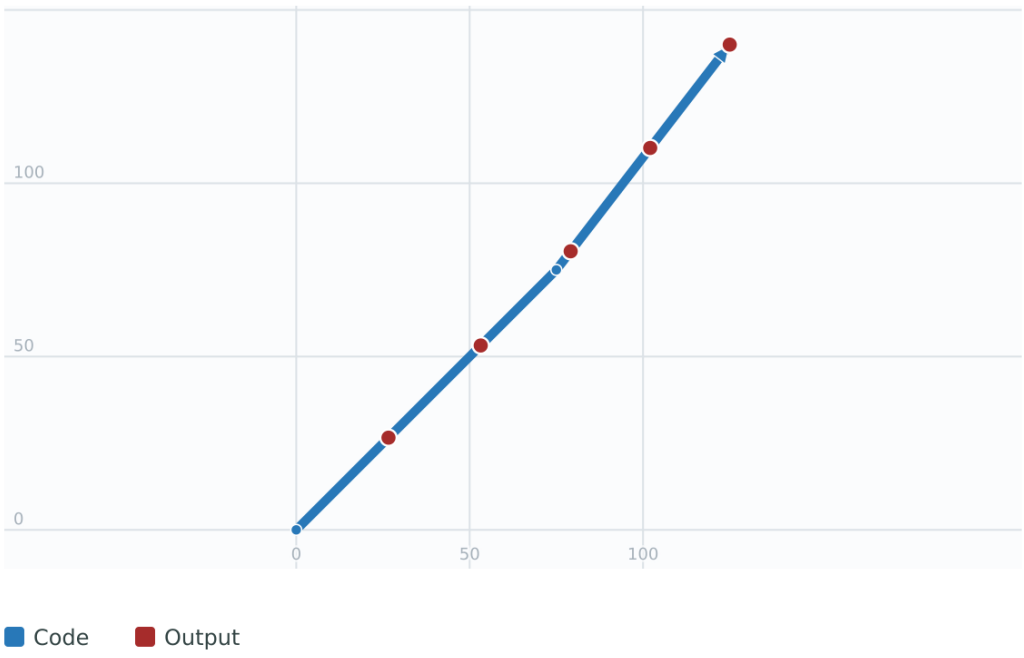
Code

```
SELECT ST_LineInterpolatePoints(
'LINESTRING(25 50,100 125,150 190)',
0.20);
```

Output

```
MULTIPOINT((51.5974135047432 76.5974135047432),(78.1948270094864 103.194827009486),(104.132163186446 130.37181214238) ↩
    ,(127.066081593223 160.18590607119),(150 190))
```

Figure



Se även

ST_LineInterpolatePoint, ST_LineLocatePoint

7.19.4 ST_LineLocatePoint

ST_LineLocatePoint — Returnerar den fraktionerade positionen för den punkt på en linje som ligger närmast en punkt.

Synopsis

float8 **ST_LineLocatePoint**(geometry a_linestring, geometry a_point);
float8 **ST_LineLocatePoint**(geography a_linestring, geography a_point, boolean use_spheroid = true);

Beskrivning

Returnerar en flottör mellan 0 och 1 som representerar platsen för den punkt på en LineString som ligger närmast den angivna Point, som en bråkdel av 2d-linjens längd.

Du kan använda den returnerade platsen för att extrahera en punkt (ST_LineInterpolatePoint) eller en delsträng (ST_LineSubstring).

Detta är användbart för att uppskatta antalet adresser

Tillgänglighet: 1.1.0

Ändrad: 2.1.0. Fram till 2.0.x hette detta ST_Line_Locate_Point.

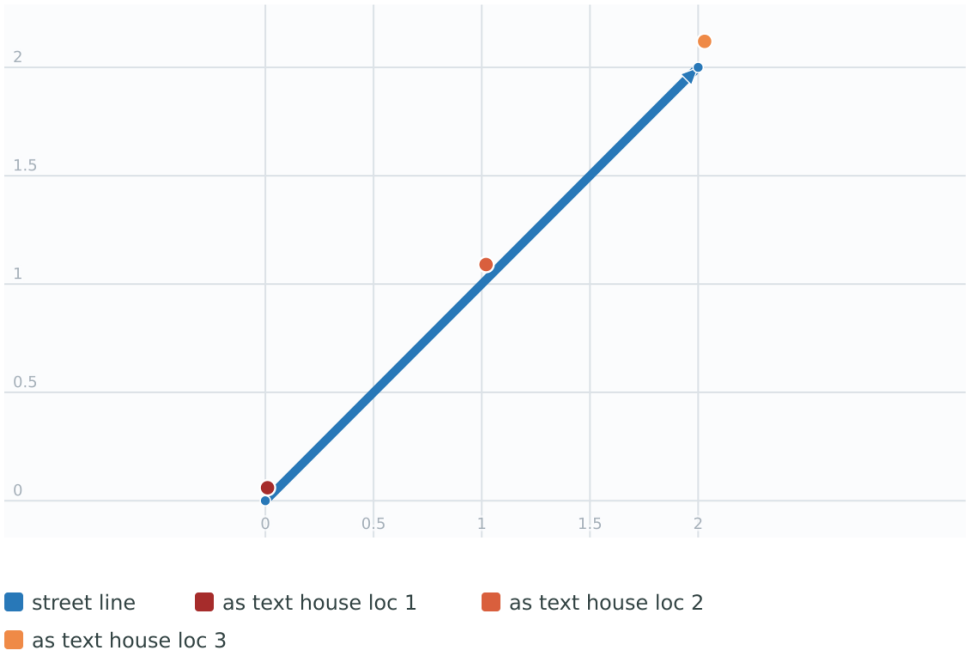
Exempel

This example approximates the street number of a house by locating the closest point on a street line. The subquery generates sample house and street data, and `ST_DWithin` excludes houses that are too far from the street.

Code

```
SELECT house_loc As as_text_house_loc,
    start_house_num +
        CAST( (end_house_num - start_house_num)
            * ST_LineLocatePoint(street_line, house_loc) As integer) As house_num
FROM
(SELECT 'LINESTRING(1 2,3 4)'::geometry As street_line,
    ST_Point(x*1.01, y*1.03) As house_loc, 10 As start_house_num,
        20 As end_house_num
FROM generate_series(1, 3) x CROSS JOIN generate_series(2, 4) As y)
As foo
WHERE ST_DWithin(street_line, house_loc, 0.2);
```

Figure



This example finds the closest point on a line to a point or other geometry.

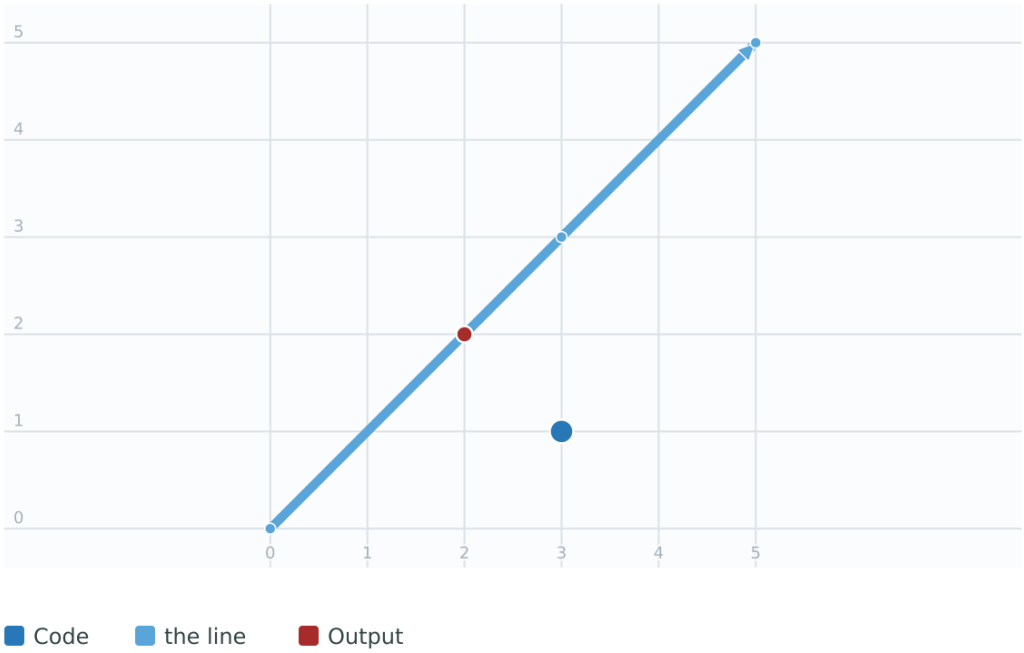
Code

```
SELECT ST_LineInterpolatePoint(
    foo.the_line,
    ST_LineLocatePoint(foo.the_line, 'POINT(4 3)'::geometry))
FROM (SELECT 'LINESTRING(1 2,4 5,6 7)'::geometry As the_line) As foo;
```

Output

```
POINT(3 4)
```

Figure

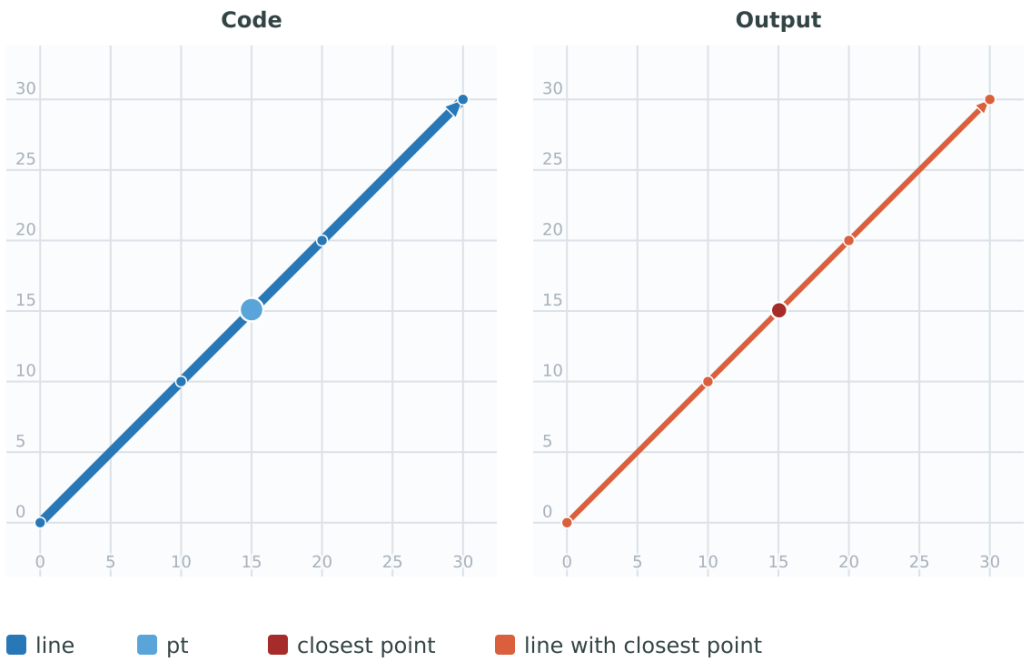


This example locates the closest segment and point on a LineString.

Code

```
WITH data AS (
  SELECT
    'LINESTRING (0 0, 10 10, 20 20, 30 30)'::geometry AS line,
    'POINT(15 15.1)'::geometry AS pt
),
segments AS (
  SELECT
    s.i,
    d.line,
    d.pt,
    ST_MakeLine(ST_PointN(d.line, s.i), ST_PointN(d.line, s.i + 1)) AS segment
  FROM data AS d
  CROSS JOIN LATERAL generate_series(1, ST_NumPoints(d.line) - 1) AS s(i)
)
SELECT
  i AS segment_number,
  round(ST_Distance(segment, pt)::numeric, 3) AS distance,
  ST_ClosestPoint(segment, pt) AS closest_point,
  ST_AddPoint(line, ST_ClosestPoint(segment, pt), i) AS line_with_closest_point
FROM segments
ORDER BY ST_Distance(segment, pt)
LIMIT 1;
```

Figure



Se även

ST_AddPoint, ST_ClosestPoint, ST_DWithin, ST_Length2D, ST_LineInterpolatePoint, ST_LineSubstring

7.19.5 ST_LineSubstring

ST_LineSubstring — Returnerar delen av en linje mellan två fraktionerade platser.

Synopsis

geometry **ST_LineSubstring**(geometry a_linestring, float8 startfraction, float8 endfraction);
geography **ST_LineSubstring**(geography a_linestring, float8 startfraction, float8 endfraction);

Beskrivning

Computes the line which is the section of the input line starting and ending at the given fractional locations. The first argument must be a LINESTRING. The second and third arguments are values in the range [0, 1] representing the start and end locations. For geometry inputs, the fractions are measured in 2D line length. The Z and M values are interpolated for added endpoints if present.

Om `startfraktion` och `slutfraktion` har samma värde är detta likvärdigt med ST_LineInterpolatePoint.

Notera
Detta fungerar endast med LINESTRINGs. För att använda på sammanhängande MULTILINESTRINGs måste du först sammanfoga dem med ST_LineMerge.

Notera
Sedan release 1.1.1 interpolerar denna funktion M- och Z-värden. Tidigare utgåvor satte Z och M till ospecificerade värden.

Förbättrad: 3.4.0 - Stöd för geografi infördes.

Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Line_Substring.

Tillgänglighet: 1.1.0, stöd för Z och M tillkommer i 1.1.1

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Extract the middle third of a LineString, from 0.333 to 0.666.

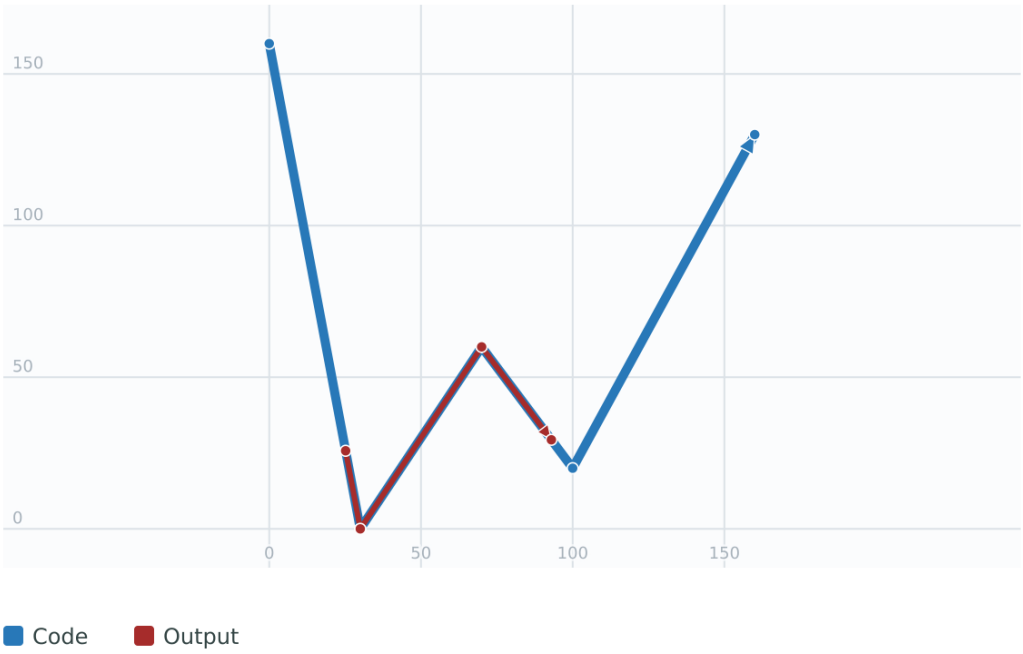
Code

```
SELECT ST_LineSubstring(
'LINESTRING (20 180,50 20,90 80,120 40,180 150)',
0.333,
0.666);
```

Output

```
LINESTRING (45.17311810399485 45.74337011202746, 50 20, 90 80, 112.97593050157862 49.36542599789519)
```

Figure



Om start- och slutplatserna är desamma blir resultatet en POINT.

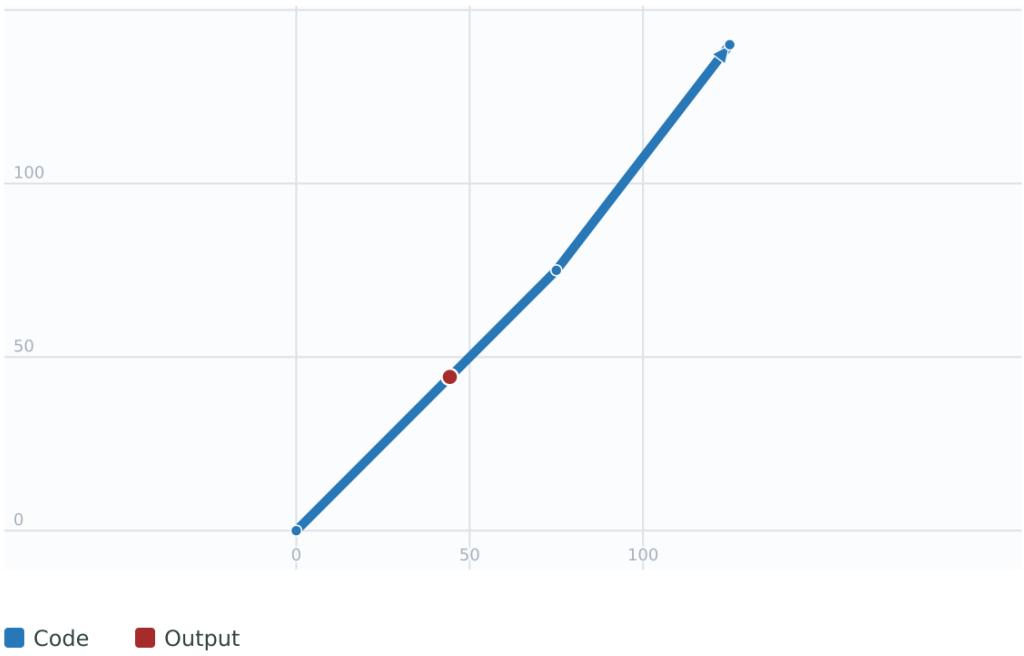
Code

```
SELECT ST_LineSubstring(
'LINESTRING(25 50,100 125,150 190)',
0.333,
0.333);
```

Output

```
POINT(69.28469348539744 94.28469348539744)
```

Figure



En fråga för att klippa en LineString i sektioner med längden 100 eller kortare. Den använder `generate_seri` med en CROSS JOIN LATERAL för att producera motsvarigheten till en FOR-loop.

The final generated step is filtered out when the line length is an exact multiple of the requested section length.

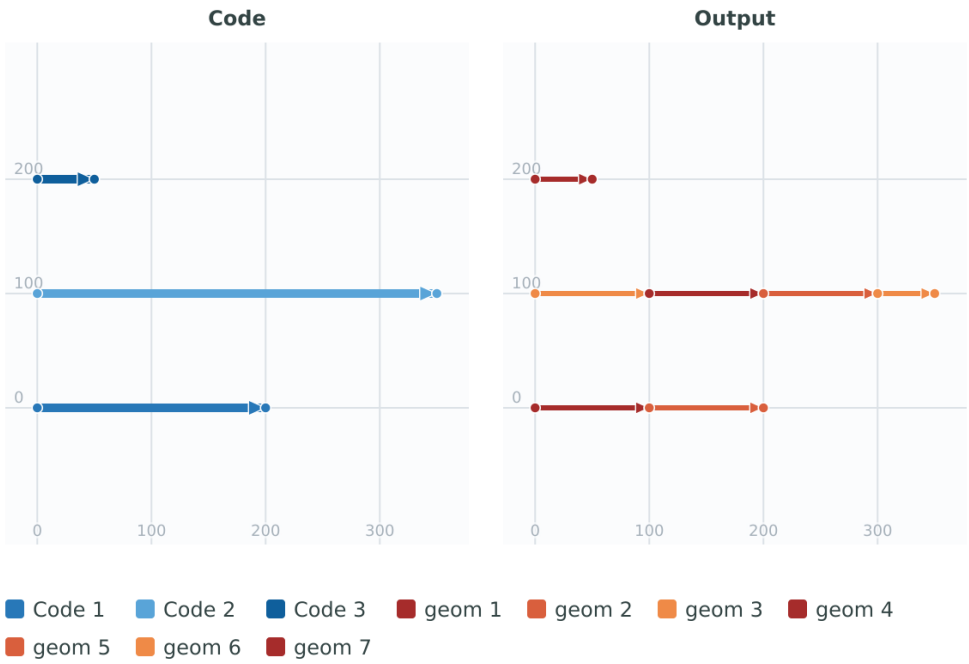
Code

```
WITH data(id, geom) AS (VALUES
        ( 'A', 'LINESTRING( 0 0, 200 0)'::geometry ),
        ( 'B', 'LINESTRING( 0 100, 350 100)'::geometry ),
        ( 'C', 'LINESTRING( 0 200, 50 200)'::geometry )
    )
SELECT id, i,
       ST_LineSubstring(geom, startfrac, LEAST( endfrac, 1 )) AS geom
FROM (
    SELECT id, geom, ST_Length(geom) len, 100 sublen FROM data
    ) AS d
CROSS JOIN LATERAL (
    SELECT i, (sublen * i) / len AS startfrac,
              (sublen * (i+1)) / len AS endfrac
    FROM generate_series(0, floor( len / sublen )::integer ) AS t(i)
    WHERE (sublen * i) / len <
> 1.0
    ) AS d2;
```

Output

```
id | i |            geom
----+---+----------------------------
 A  | 0 | LINESTRING(0 0,100 0)
 A  | 1 | LINESTRING(100 0,200 0)
 B  | 0 | LINESTRING(0 100,100 100)
 B  | 1 | LINESTRING(100 100,200 100)
 B  | 2 | LINESTRING(200 100,300 100)
 B  | 3 | LINESTRING(300 100,350 100)
 C  | 0 | LINESTRING(0 200,50 200)
```

Figure



Split a LineString by points by locating the points along the line, adding the 0 and 1 endpoints, and extracting substrings between adjacent locations.

Code

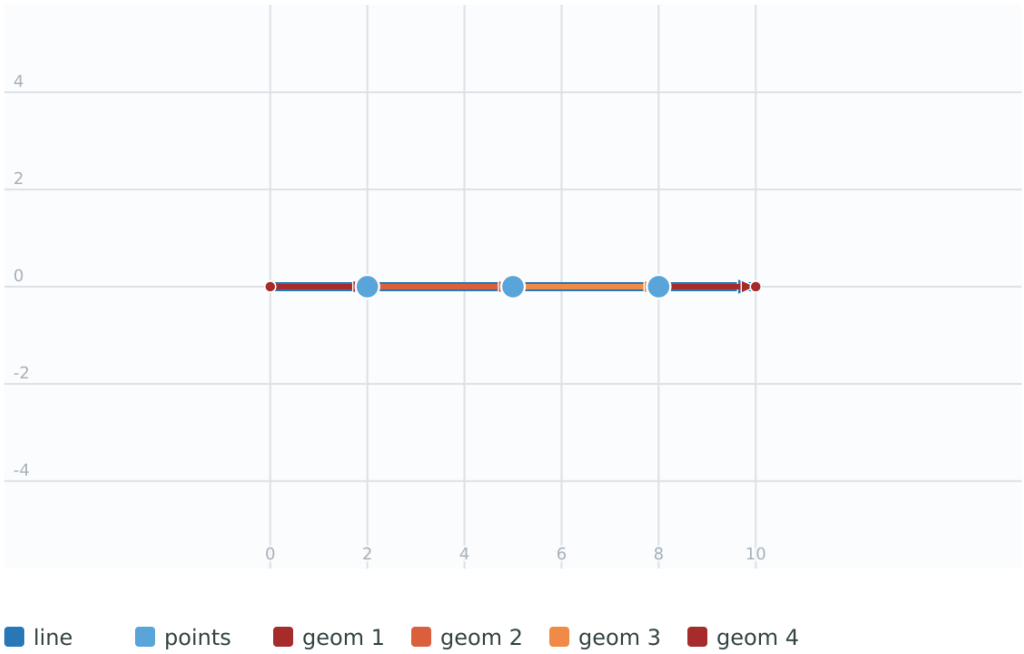
```
WITH data AS (
  SELECT 'LINESTRING(0 0, 10 0)'::geometry AS line,
         'MULTIPOINT((2 0),(5 0),(8 0))'::geometry AS points
),
fractions AS (
  SELECT 0.0 AS fraction
  UNION ALL
  SELECT 1.0
  UNION ALL
  SELECT ST_LineLocatePoint(line, (dp).geom)
  FROM data
  CROSS JOIN LATERAL ST_Dump(points) AS dp
),
ordered AS (
  SELECT DISTINCT fraction
  FROM fractions
),
segments AS (
  SELECT fraction AS start_fraction,
         lead(fraction) OVER (ORDER BY fraction) AS end_fraction
  FROM ordered
)
```

```
SELECT row_number() OVER (ORDER BY start_fraction) AS segment_id,
       ST_LineSubstring(line, start_fraction, end_fraction) AS geom
FROM data
CROSS JOIN segments
WHERE end_fraction IS NOT NULL
  AND start_fraction <
> end_fraction
ORDER BY start_fraction;
```

Output

```
segment_id |         geom
-----------+----------------------
         1 | LINESTRING(0 0,2 0)
         2 | LINESTRING(2 0,5 0)
         3 | LINESTRING(5 0,8 0)
         4 | LINESTRING(8 0,10 0)
```

Figure



Geografiska implementeringsåtgärder längs en sfäroid, geometri längs en linje

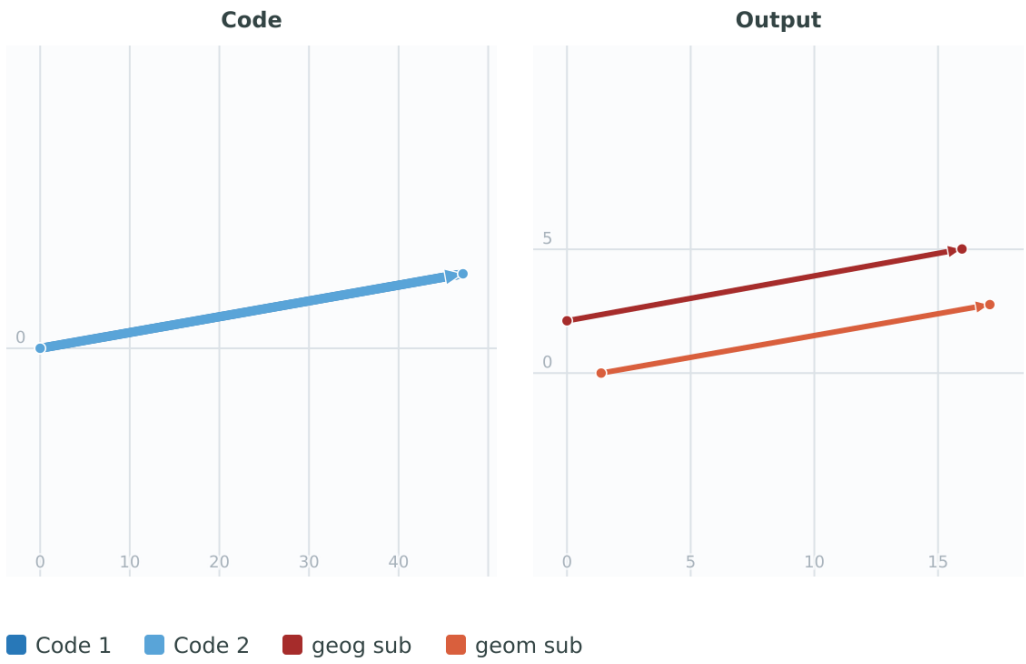
Code

```
SELECT ST_LineSubstring(
'LINESTRING(-118.2436 34.0522,-71.0570 42.3611)'::geography,
0.333,
0.666) AS geog_sub,
ST_LineSubstring(
    'LINESTRING(-118.2436 34.0522,-71.0570 42.3611)'::geometry,
    0.333,
    0.666) AS geom_sub;
```

Output

```
-[ RECORD 1 ]----------
geog_sub | LINESTRING(-103.911641 38.931128,-87.941787 41.831072)
geom_sub | LINESTRING(-102.530462 36.819064,-86.817324 39.585927)
```

Figure



For geometry inputs, the fractional locations are based on the 2D length of the line, even when the input has Z values. The resulting Z values are interpolated along the selected 2D location.

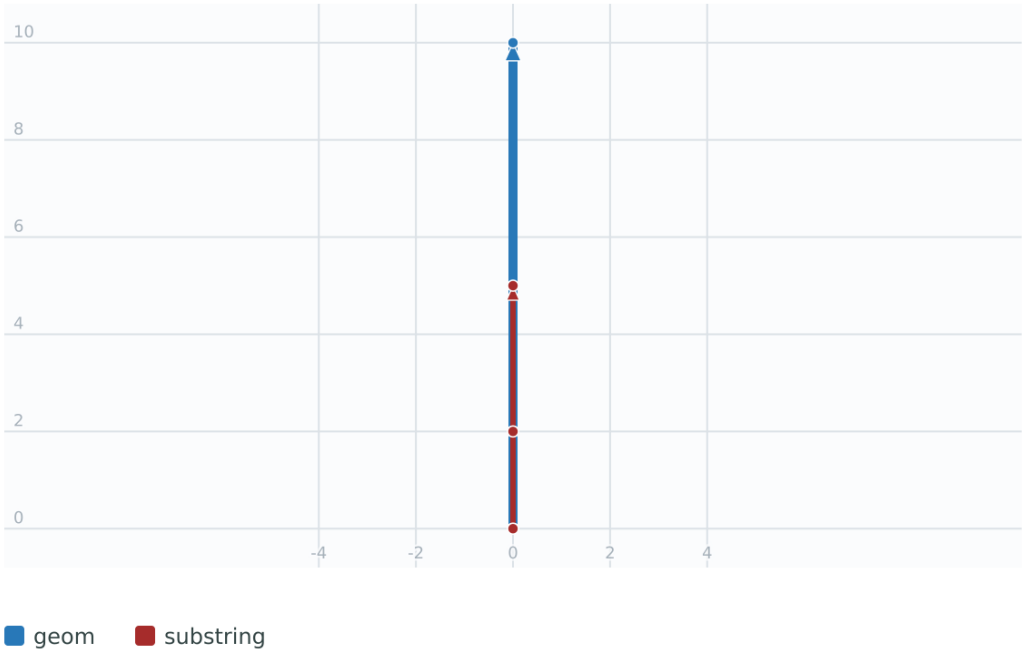
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (0 0 0,0 2 5,0 10 10)'::geometry AS geom
)
SELECT ST_Length(geom) AS length_2d,
       ST_3DLength(geom) AS length_3d,
       ST_LineSubstring(geom, 0, 0.5) AS substring
FROM data;
```

Output

```
length_2d |    length_3d     |              substring
----------+------------------+-------------------------------------
        10 | 14.819145939191106 | LINESTRING Z (0 0 0,0 2 5,0 5 6.875)
```

Figure



Se även

ST_Length, ST_LineExtend, ST_LineInterpolatePoint, ST_LineLocatePoint, ST_LineMerge, ST_Split

7.19.6 ST_LocateAlong

ST_LocateAlong — Returnerar den eller de punkter på en geometri som matchar ett mätvärde.

Synopsis

geometry **ST_LocateAlong**(geometry geom_with_measure, float8 measure, float8 offset = 0);

Beskrivning

Returnerar den eller de platser längs en uppmätt geometri som har de angivna mätvärdena. Resultatet är en Point eller MultiPoint. Polygonala indata stöds inte.

Om `offset` anges, förskjuts resultatet till vänster eller höger om inmatningsraden med det angivna avståndet. En positiv förskjutning blir till vänster och en negativ till höger.

Notera
Använd denna funktion endast för linjära geometrier med en M-komponent

Semantiken specificeras av standarden *ISO/IEC 13249-3 SQL/MM Spatial*.

Tillgänglighet: 1.1.0 med det gamla namnet ST_Locate_Along_Measure.

Ändrad: 2.0.0 i tidigare versioner kallades detta för ST_Locate_Along_Measure.

Denna funktion stöder M-koordinater.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1.13

Exempel

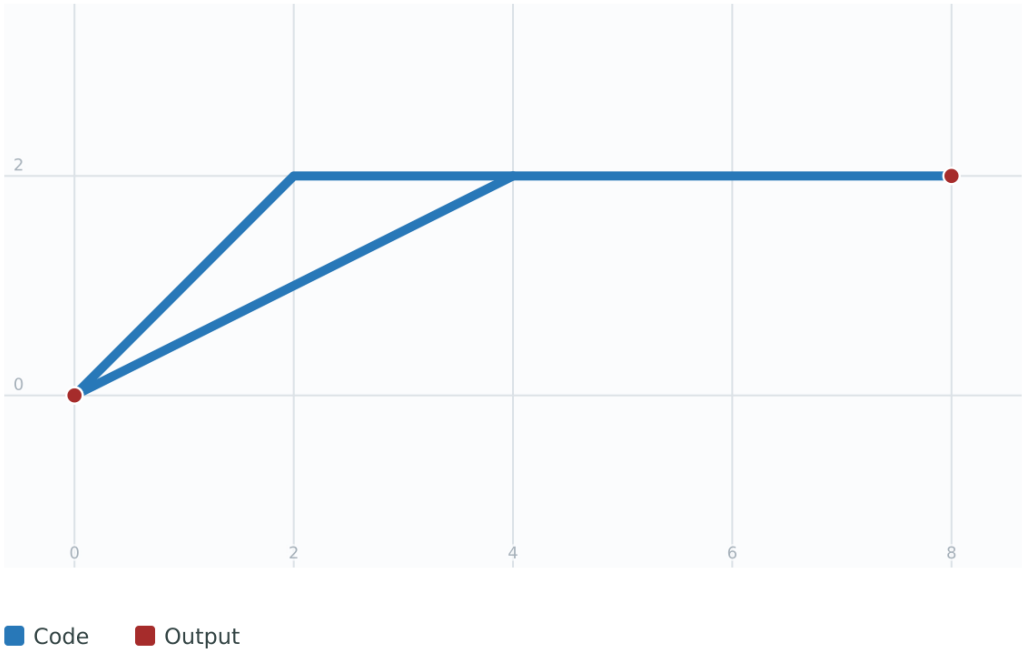
Code

```
SELECT ST_LocateAlong(
'MULTILINESTRINGM((1 2 3, 3 4 2, 9 4 3),(1 2 3, 5 4 5))'::geometry,
3);
```

Output

```
MULTIPOINT M ((1 2 3),(9 4 3),(1 2 3))
```

Figure



Se även

ST_LocateBetween, ST_LocateBetweenElevations, ST_InterpolatePoint

7.19.7 ST_LocateBetween

ST_LocateBetween — Returnerar de delar av en geometri som matchar ett mätintervall.

Synopsis

geometry **ST_LocateBetween**(geometry geom, float8 measure_start, float8 measure_end, float8 offset = 0);

Beskrivning

Returnerar en geometri (samling) med de delar av den inmatade uppmätta geometrin som matchar det angivna mätområdet (inklusive).

Om `offset` anges kommer resultatet att förskjutas till vänster eller höger om inmatningsraden med det angivna avståndet. En positiv förskjutning blir till vänster och en negativ till höger.

Att klippa en icke-konvex POLYGON kan ge ogiltig geometri.

Semantiken specificeras av standarden *ISO/IEC 13249-3 SQL/MM Spatial*.

Tillgänglighet: 1.1.0 med det gamla namnet ST_Locate_Between_Measures.

Ändrad: 2.0.0 - i tidigare versioner hette detta ST_Locate_Between_Measures.

Förbättrad: 3.0.0 - stöd för POLYGON, TIN, TRIANGLE har lagts till.

Denna funktion stöder M-koordinater.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1

Exempel

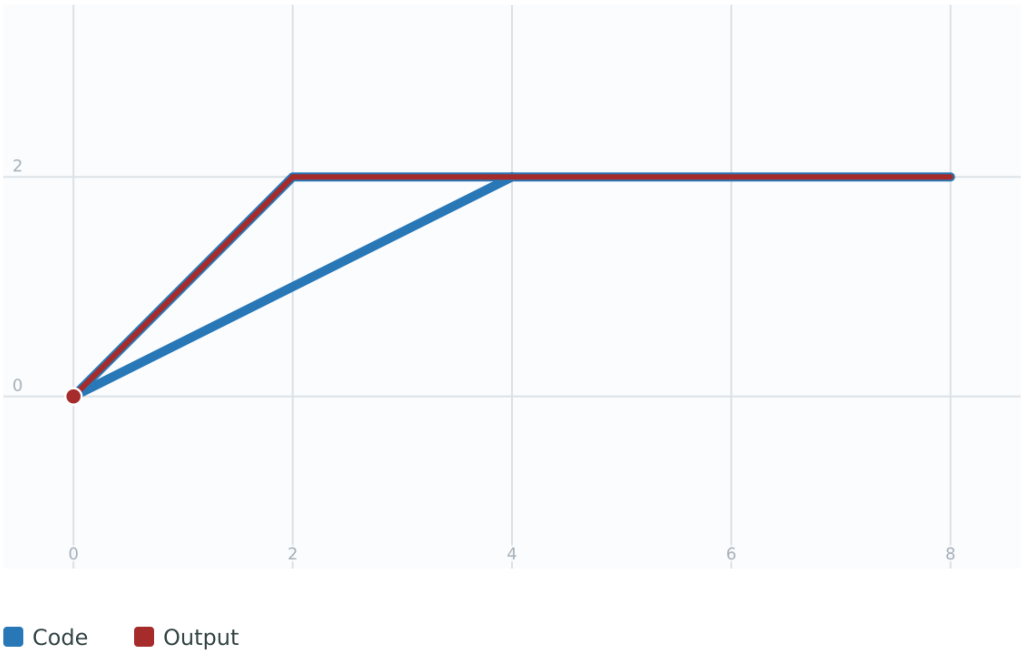
Code

```
SELECT ST_LocateBetween(
'MULTILINESTRING M ((1 2 3,3 4 2,9 4 3),(1 2 3,5 4 5))'::geometry,
1.5,
3);
```

Output

```
GEOMETRYCOLLECTION M (LINESTRING M (1 2 3,3 4 2,9 4 3),POINT M (1 2 3))
```

Figure



Extract the section between measures 2 and 8, offset to the left. Because the clipped line and the offset result share the same coordinate space, this example is intended for a single overlay panel rather than separate side-by-side figures.

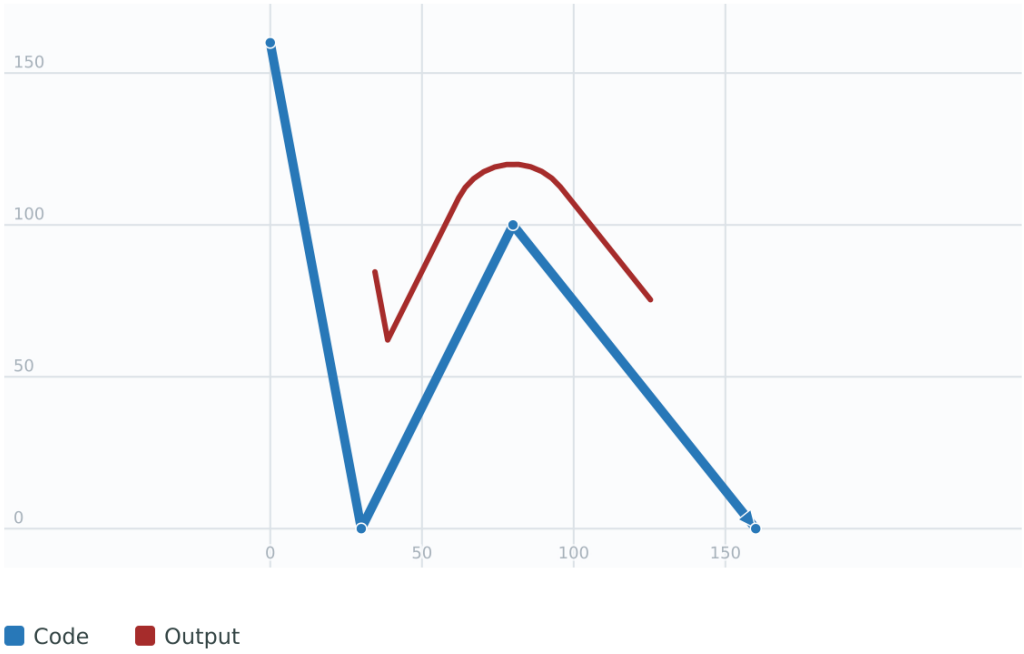
Code

```
SELECT ST_LocateBetween(
ST_AddMeasure(
    'LINESTRING (20 180,50 20,100 120,180 20)',
    0,
    10),
2,
8,
20);
```

Output

```
MULTILINESTRING((54.49835 104.53427,58.700561 82.122481,82.111456 128.944272,84.248937 132.324934,87.01637
    135.21267,90.303073 137.491987,93.977598 139.071724,97.892984 139.888702,101.892639 139.910247,105.816599
    139.135495,109.507928 137.595434,112.818995 135.351657,115.617376 132.493901,145.310173 95.377905))
```

Figure



Se även

ST_LocateAlong, ST_LocateBetweenElevations

7.19.8 ST_LocateBetweenElevations

ST_LocateBetweenElevations — Returnerar de delar av en geometri som ligger inom ett höjdintervall (Z).

Synopsis

geometry **ST_LocateBetweenElevations**(geometry geom, float8 elevation_start, float8 elevation_end);

Beskrivning

Returnerar en geometri (samling) med de delar av en geometri som ligger i ett höjdintervall (Z).

Att klippa en icke-konvex POLYGON kan ge ogiltig geometri.

Tillgänglighet: 1.4.0

Förbättrad: 3.0.0 - stöd för POLYGON, TIN, TRIANGLE har lagts till.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

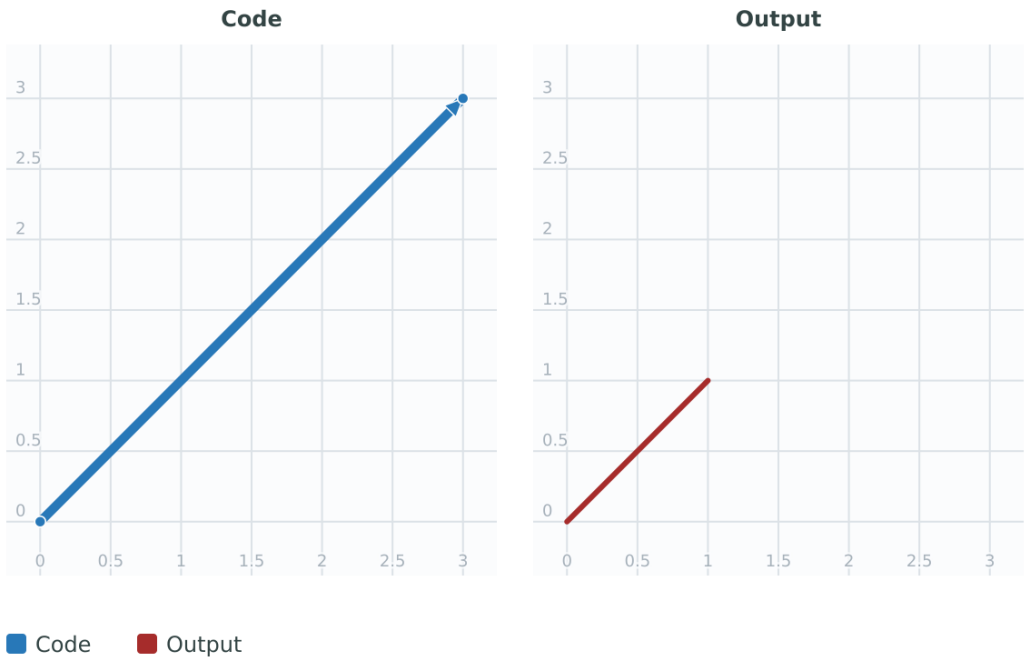
Code

```
SELECT ST_LocateBetweenElevations(
'LINESTRING(1 2 3,4 5 6)'::geometry,
2,
4);
```

Output

```
MULTILINESTRING Z ((1 2 3,2 3 4))
```

Figure



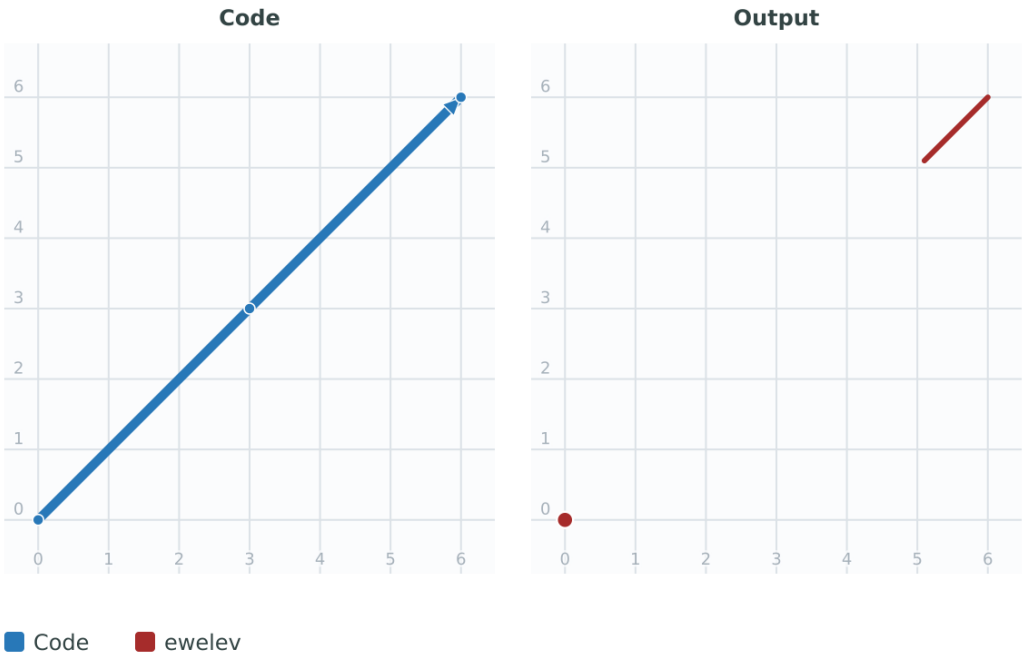
Code

```
SELECT ST_LocateBetweenElevations(
'LINESTRING(1 2 6,4 5 -1,7 8 9)',
6,
9) As ewelev;
```

Output

```
GEOMETRYCOLLECTION Z (POINT Z (1 2 6),LINESTRING Z (6.1 7.1 6,7 8 9))
```

Figure



Se även

ST_Dump, ST_LocateBetween

7.19.9 ST_InterpolatePoint

ST_InterpolatePoint — Returnerar det interpolerade måttet för en geometri som ligger närmast en punkt.

Synopsis

float8 **ST_InterpolatePoint**(geometry linear_geom_with_measure, geometry point);

Beskrivning

Returnerar ett interpolerat mätvärde för en linjärt uppmätt geometri på den plats som ligger närmast den angivna punkten.

> **Notera**
> Använd denna funktion endast för linjära geometrier med en M-komponent

Tillgänglighet: 2.0.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

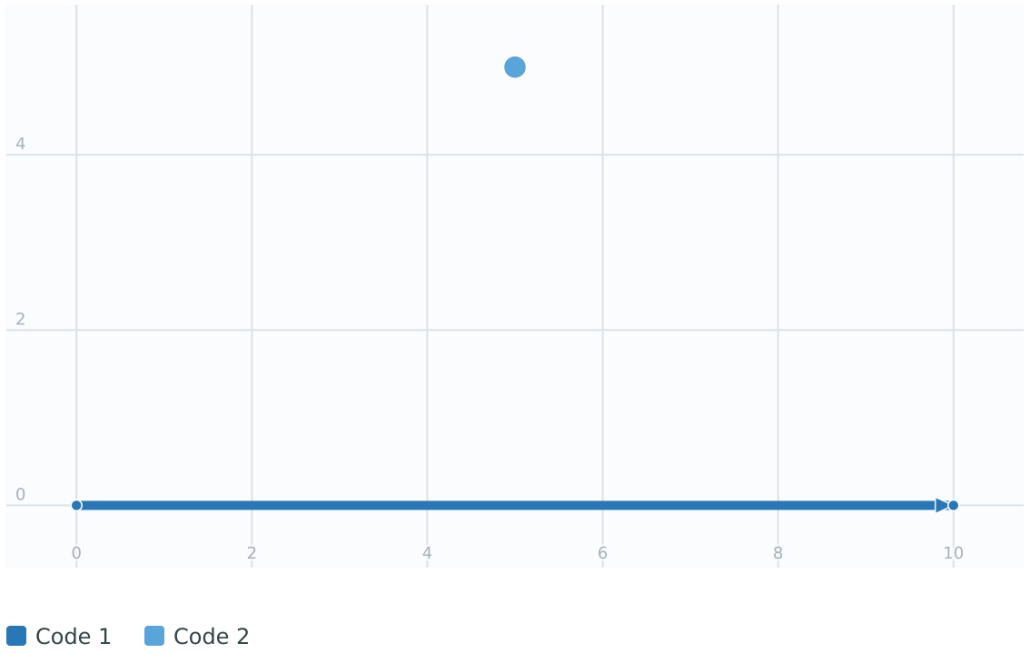
Code

```
SELECT ST_InterpolatePoint(
'LINESTRING M (0 0 0,10 0 20)',
'POINT(5 5)');
```

Output

```
10
```

Figure



Se även

ST_AddMeasure, ST_LocateAlong, ST_LocateBetween

7.19.10 ST_3DInterpolatePoint

ST_3DInterpolatePoint — Returns the interpolated measure of a geometry closest to a point in 3D.

Synopsis

float8 **ST_3DInterpolatePoint**(geometry linear_geom_with_measure, geometry point);

Beskrivning

Returns the interpolated measure value of a linear ZM geometry at the location closest to the given point, using 3D (XYZ) distance for the projection. Use this function when the geometry has significant Z variation, such as flight trajectories, where ST_InterpolatePoint would give incorrect results by ignoring the Z dimension.

Notera
The line must have both Z and M dimensions. The point should have a Z dimension.

Availability: 3.7.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

This example interpolates on a line rising diagonally in Z, using a point at the 3D midpoint.

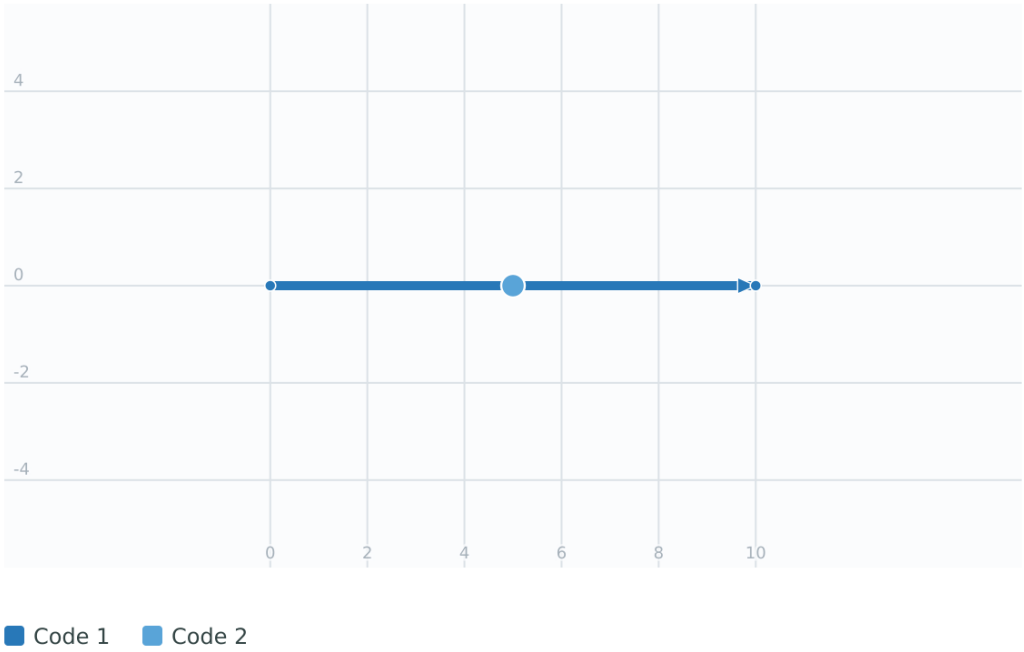
Code

```
SELECT ST_3DInterpolatePoint(
'LINESTRING ZM (0 0 0 0,10 0 10 20)',
'POINT Z (5 0 5)');
```

Output

```
10
```

Figure



This example uses a vertical line with zero XY extent, where 3D projection works but 2D projection does not.

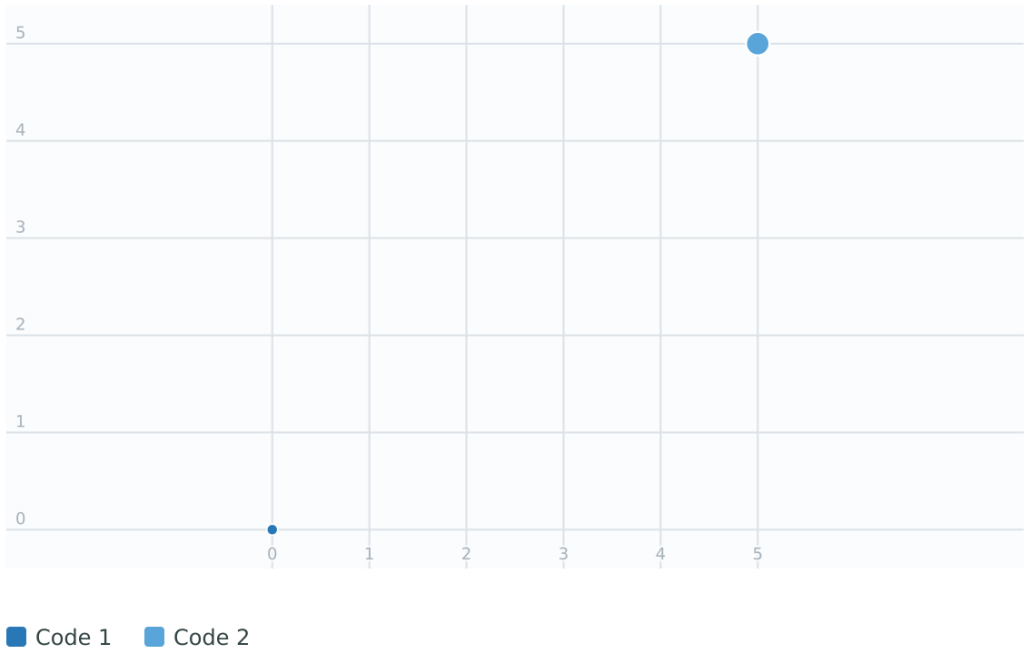
Code

```
SELECT ST_3DInterpolatePoint(
'LINESTRING ZM (0 0 0 0,0 0 10 100)',
'POINT Z (5 5 5)');
```

Output

```
50
```

Figure



Se även

ST_InterpolatePoint, ST_AddMeasure, ST_LocateAlong, ST_LocateBetween

7.19.11 ST_AddMeasure

ST_AddMeasure — Interpolerar mått längs en linjär geometri.

Synopsis

geometry **ST_AddMeasure**(geometry geom_mline, float8 measure_start, float8 measure_end);

Beskrivning

Returnerar en härledd geometri med mätvärden linjärt interpolerade mellan start- och slutpunkterna. Om geometrin inte har någon måttdimension läggs en till. Om geometrin har en måttdimension skrivs den över med nya värden. Endast LINESTRINGS och MULTILINESTRINGS stöds.

Tillgänglighet: 1.5.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

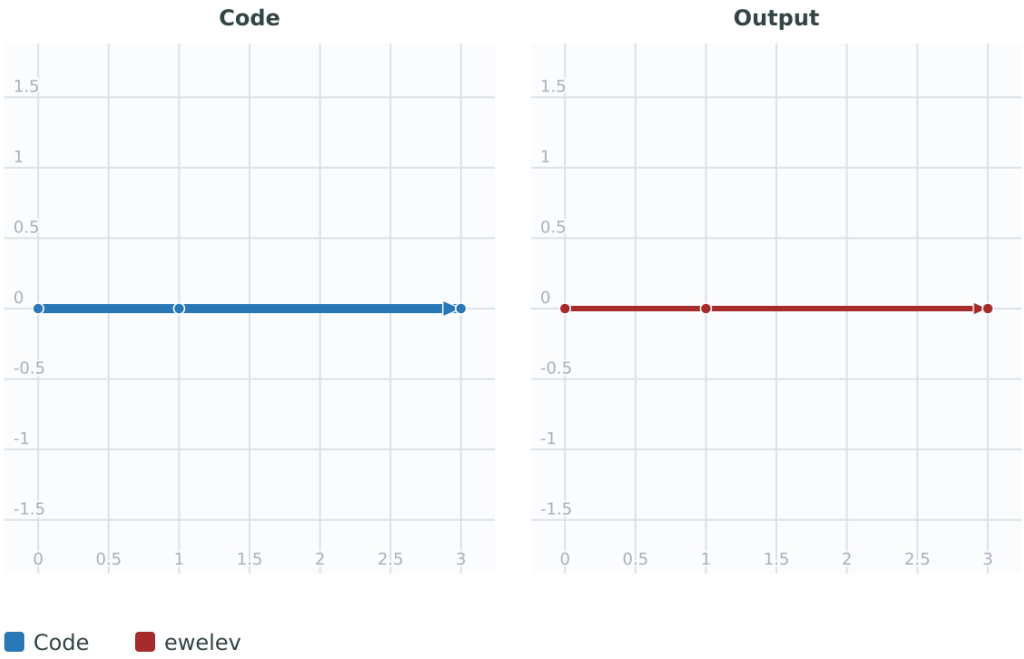
Code

```
SELECT ST_AddMeasure(
'LINESTRING(1 0,2 0,4 0)',
1,
4) As ewelev;
```

Output

```
LINESTRING M (1 0 1,2 0 2,4 0 4)
```

Figure



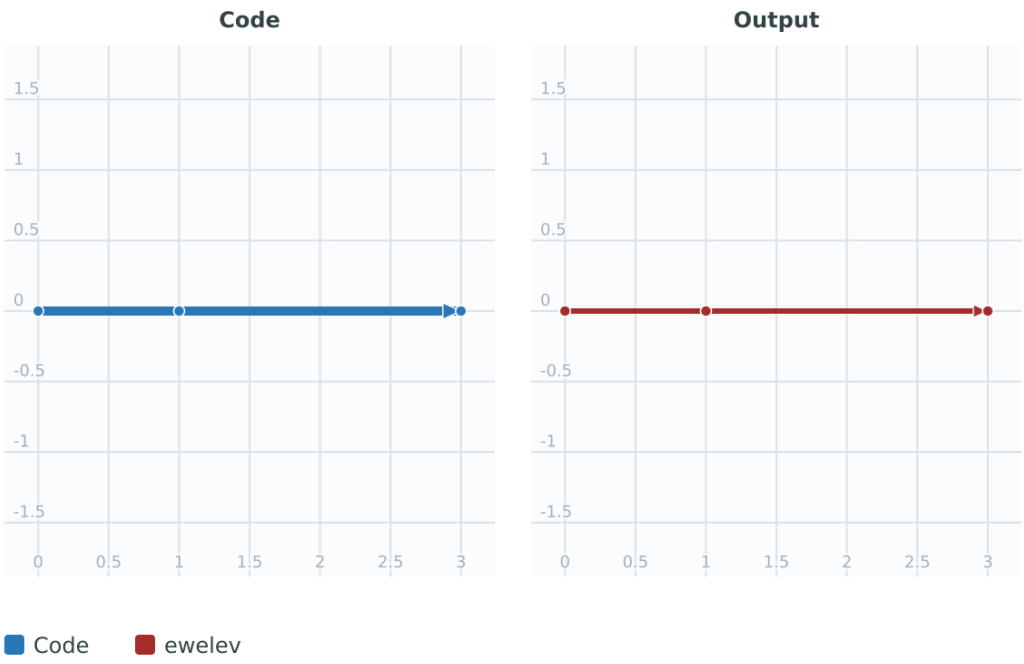
Code

```
SELECT ST_AddMeasure(
'LINESTRING(1 0 4,2 0 4,4 0 4)',
10,
40) As ewelev;
```

Output

```
LINESTRING ZM (1 0 4 10,2 0 4 20,4 0 4 40)
```

Figure



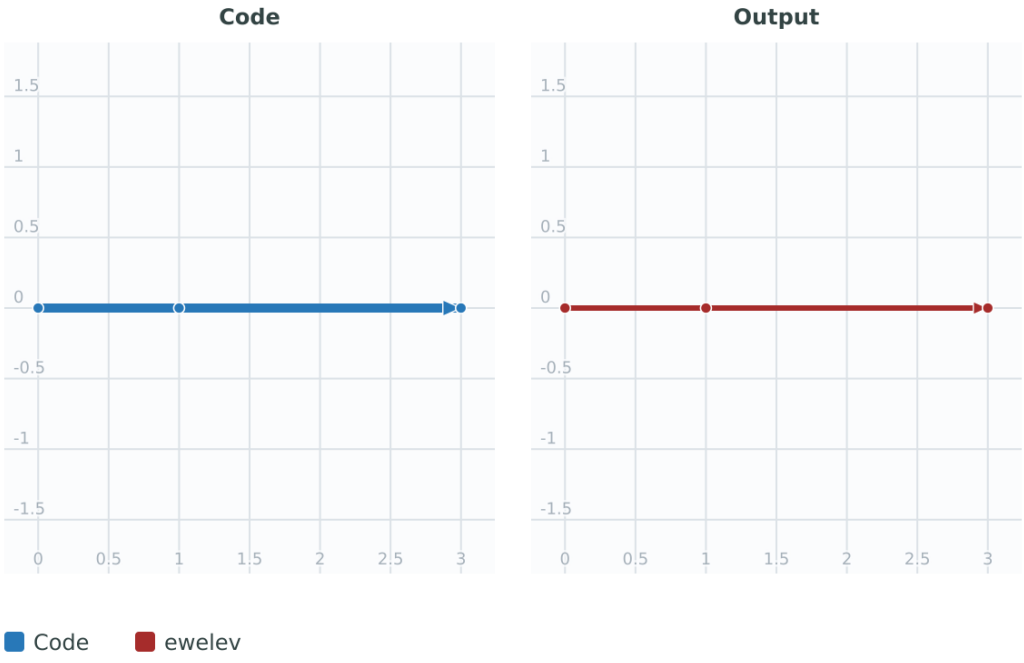
Code

```
SELECT ST_AddMeasure(
'LINESTRINGM(1 0 4, 2 0 4, 4 0 4)',
10,
40) As ewelev;
```

Output

```
LINESTRING M (1 0 10,2 0 20,4 0 40)
```

Figure



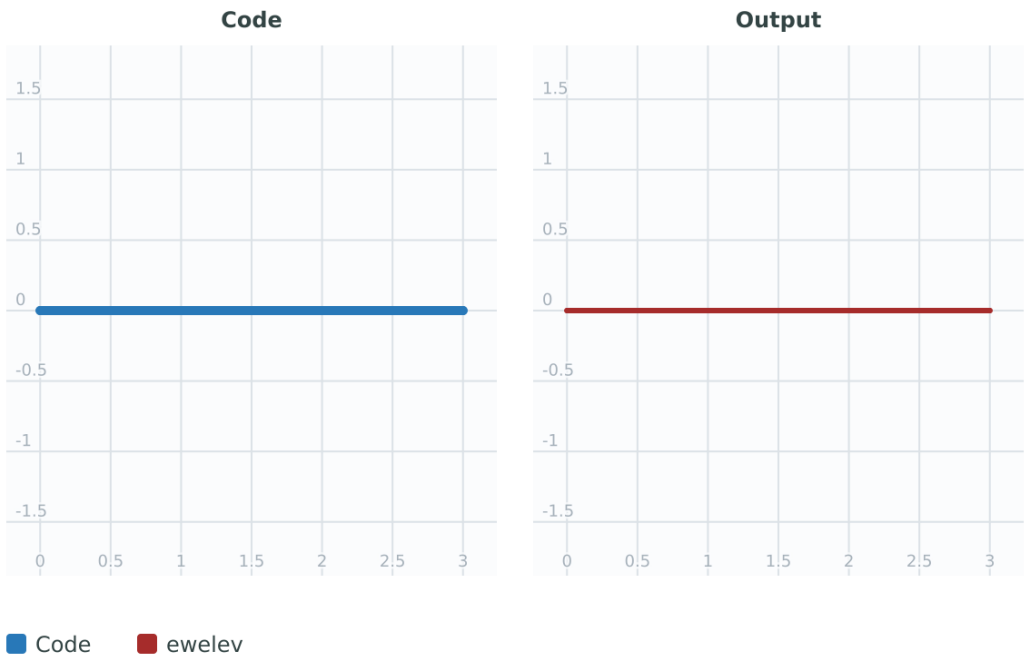
Code

```
SELECT ST_AddMeasure(
'MULTILINESTRINGM((1 0 4, 2 0 4, 4 0 4),(1 0 4, 2 0 4, 4 0 4))',
10,
70) As ewelev;
```

Output

```
MULTILINESTRING M ((1 0 10,2 0 20,4 0 40),(1 0 40,2 0 50,4 0 70))
```

Figure



7.20 Trajektoriefunktioner

7.20.1 ST_IsValidTrajectory

ST_IsValidTrajectory — Testar om geometrin är en giltig bana.

Synopsis

boolean **ST_IsValidTrajectory**(geometry line);

Beskrivning

Testar om en geometri kodar för en giltig bana. En giltig bana representeras som en `LINESTRING` med mått (M-värden). Mätvärdena måste öka från varje toppunkt till nästa.

Giltiga banor förväntas som indata till spatio-temporala funktioner som ST_ClosestPointOfApproach

Tillgänglighet: 2.2.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

This example constructs a valid trajectory.

Code

```
SELECT ST_IsValidTrajectory(ST_MakeLine(ST_MakePointM(0, 0, 1),
ST_MakePointM(0, 1, 2)));
```

Output

```
t
```

This example constructs an invalid trajectory.

Code

```
SELECT ST_IsValidTrajectory(ST_MakeLine(ST_MakePointM(0, 0, 1), ST_MakePointM(0, 1, 0)));
```

Output

```
NOTICE:  Measure of vertex 1 (0) not bigger than measure of vertex 0 (1)
f
```

Se även

ST_ClosestPointOfApproach

7.20.2 ST_ClosestPointOfApproach

ST_ClosestPointOfApproach — Returnerar ett mått på den närmaste närmandepunkten för två banor.

Synopsis

float8 **ST_ClosestPointOfApproach**(geometry track1, geometry track2);

Beskrivning

Returnerar det minsta mått vid vilket punkter interpolerade längs de givna banorna har det minsta avståndet från varandra.

Ingångarna måste vara giltiga banor som kontrolleras av ST_IsValidTrajectory. Null returneras om banorna inte överlappar varandra i sina M-intervall.

För att få de faktiska punkterna vid det beräknade måttet använder du ST_LocateAlong.

Tillgänglighet: 2.2.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Return the time in which two objects moving between 10:00 and 11:00 are closest to each other and their distance at that point.

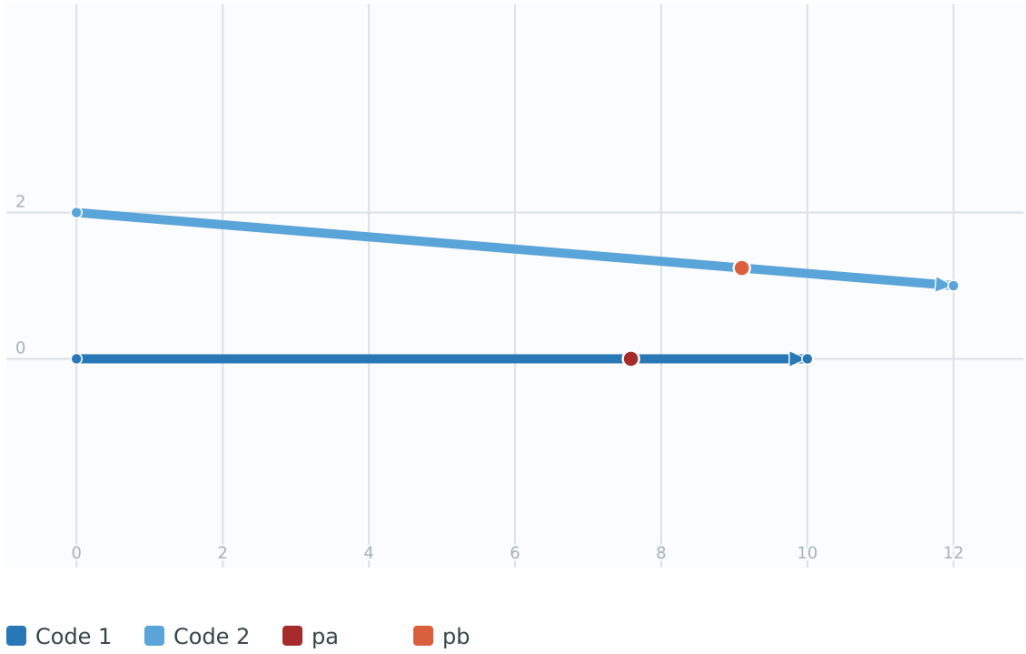
Code

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0,10 0 5)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10,12 1 2)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) b
), cpa AS (
  SELECT ST_ClosestPointOfApproach(a, b) m FROM inp
), points AS (
  SELECT ST_GeometryN(ST_LocateAlong(a, m), 1) pa,
         ST_GeometryN(ST_LocateAlong(b, m), 1) pb
  FROM inp, cpa
)
SELECT to_timestamp(m) AT TIME ZONE 'UTC' t,
       ST_3DDistance(pa, pb) distance,
       pa AS pa, pb AS pb
FROM points, cpa;
```

Output

```
-[ RECORD 1 ]---------------------------
t        | 2015-05-26 10:45:31.034483
distance | 1.9652147377620688
pa       | POINT ZM (7.59 0 3.79 1432637131.03)
pb       | POINT ZM (9.1 1.24 3.93 1432637131.03)
```

Figure



Se även

ST_IsValidTrajectory, ST_DistanceCPA, ST_LocateAlong, ST_AddMeasure

7.20.3 ST_DistanceCPA

ST_DistanceCPA — Returnerar avståndet mellan de två banornas närmaste närmandepunkter.

Synopsis

float8 **ST_DistanceCPA**(geometry track1, geometry track2);

Beskrivning

Returnerar avståndet (i 2D) mellan två banor vid deras närmaste närmandepunkt.

Ingångarna måste vara giltiga banor som kontrolleras av ST_IsValidTrajectory. Null returneras om banorna inte överlappar varandra i sina M-intervall.

Tillgänglighet: 2.2.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Return the minimum distance of two objects moving between 10:00 and 11:00.

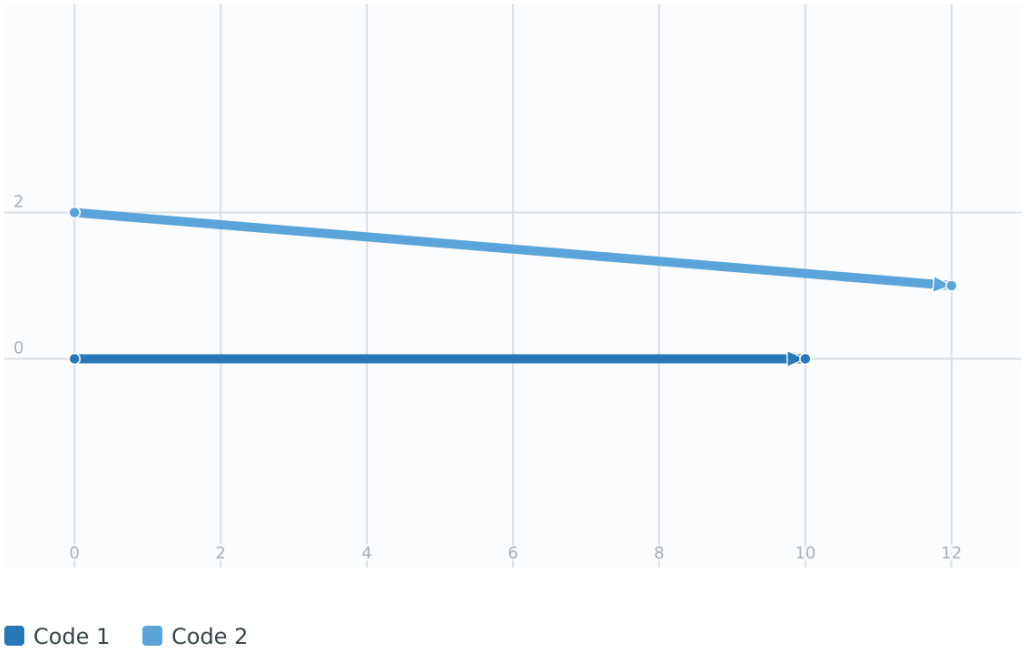
Code

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0,10 0 5)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10,12 1 2)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) b
)
SELECT ST_DistanceCPA(a, b) distance FROM inp;
```

Output

```
1.9652147377620688
```

Figure



Se även

ST_IsValidTrajectory, ST_ClosestPointOfApproach, ST_AddMeasure, |=|

7.20.4 ST_CPAWithin

ST_CPAWithin — Testar om den närmaste punkten för två banor ligger inom det angivna avståndet.

Synopsis

boolean **ST_CPAWithin**(geometry track1, geometry track2, float8 dist);

Beskrivning

Testar om två rörliga objekt någonsin har varit närmare än det angivna avståndet.

Ingångarna måste vara giltiga banor som kontrolleras av ST_IsValidTrajectory. False returneras om banorna inte överlappar varandra i sina M-områden.

Tillgänglighet: 2.2.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

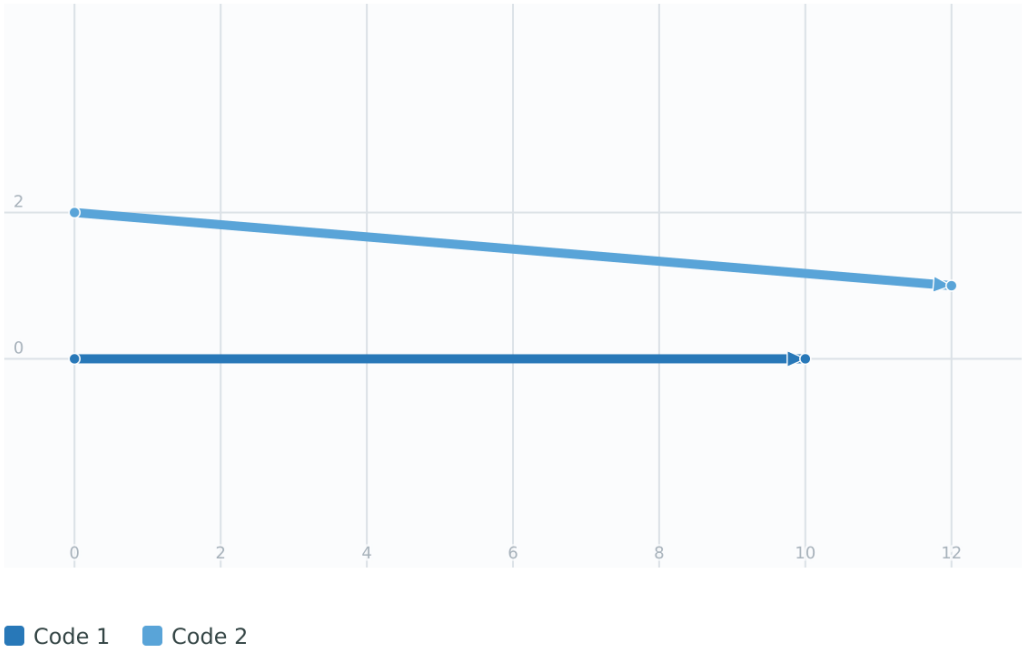
Code

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0,10 0 5)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10,12 1 2)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) b
)
SELECT ST_CPAWithin(a, b, 2), ST_DistanceCPA(a, b) distance FROM inp;
```

Output

```
st_cpawithin |     distance
-------------+------------------
 t           | 1.9652147377620688
```

Figure



Se även

ST_IsValidTrajectory, ST_ClosestPointOfApproach, ST_DistanceCPA, |=|

7.21 Versionsfunktioner

7.21.1 PostGIS_Extensions_Upgrade

PostGIS_Extensions_Upgrade — Paketerar och uppgraderar PostGIS-tillägg (t.ex. postgis_raster, postgis_topology, postgis_sfcgal) till given eller senaste version.

Synopsis

text **PostGIS_Extensions_Upgrade**(text target_version=null);

Beskrivning

Packages and upgrades PostGIS extensions to given or latest version. Only extensions you have installed in the database will be packaged and upgraded if needed. Reports full PostGIS version and build configuration infos after. This is short-hand for doing multiple CREATE EXTENSION .. FROM unpackaged and ALTER EXTENSION .. UPDATE for each PostGIS extension. Currently only tries to upgrade extensions postgis, postgis_raster, postgis_sfcgal, and postgis_topology.

Tillgänglighet: 2.5.0

Ändrad: 3.4.0 för att lägga till argumentet target_version.

Ändrat: 3.3.0 stöd för uppgraderingar från alla PostGIS-versioner. Fungerar inte på alla system.

Ändrad: 3.0.0 för att packa om lösa tillägg och stödja postgis_raster.

Exempel

Code

```
SELECT PostGIS_Extensions_Upgrade();
```

Output

```
NOTICE:  Packaging extension postgis
NOTICE:  Packaging extension postgis_raster
NOTICE:  Packaging extension postgis_sfcgal
NOTICE:  Extension postgis_topology is not available or not packagable for some reason
```

Se även

Avsnitt 3.4, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version PostGIS_Version

7.21.2 PostGIS_Full_Version

PostGIS_Full_Version — Rapporterar fullständig information om PostGIS-version och byggkonfiguration.

Synopsis

text **PostGIS_Full_Version**();

Beskrivning

Rapporterar fullständig information om PostGIS-version och byggkonfiguration. Informerar också om synkronisering mellan bibliotek och skript som föreslår uppgraderingar vid behov.

Förbättrad: 3.4.0 innehåller nu extra PROJ-konfigurationer NETWORK_ENABLED, URL_ENDPOINT och DATABASE_PATH för proj.db-platsen

Exempel

Code

```
SELECT PostGIS_Full_Version();
```

Output

```
POSTGIS="3.7.0dev 3.7.0alpha1-477-ga9004e778" [EXTENSION] PGSQL="180" GEOS="3.15.0beta1" SFCGAL="2.3.0" PROJ="9.8.1 ↩
     NETWORK_ENABLED=OFF URL_ENDPOINT=https://cdn.proj.org DATABASE_PATH=/usr/share/proj/proj.db" GDAL="GDAL 3.13.1, ↩
     released 2026/06/01" LIBXML="2.15.3"
```

Se även

Avsnitt 3.4, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version PostGIS_Wagyu_Version, PostGIS_Version

7.21.3 PostGIS_GEOS_Version

PostGIS_GEOS_Version — Returnerar versionsnumret för GEOS-biblioteket.

Synopsis

text **PostGIS_GEOS_Version**();

Beskrivning

Returnerar versionsnumret för GEOS-biblioteket, eller NULL om GEOS-stöd inte är aktiverat.

Exempel

Code

```
SELECT PostGIS_GEOS_Version();
```

Output

```
3.12.0dev-CAPI-1.18.0
```

Se även

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Version

7.21.4 PostGIS_GEOS_Compiled_Version

PostGIS_GEOS_Compiled_Version — Returnerar versionsnumret för GEOS-biblioteket som PostGIS byggdes mot.

Synopsis

text **PostGIS_GEOS_Compiled_Version**();

Beskrivning

Returnerar versionsnumret för GEOS-biblioteket, eller mot vilket PostGIS byggdes.

Tillgänglighet: 3.4.0

Exempel

Code

```
SELECT PostGIS_GEOS_Compiled_Version();
```

Output

```
3.12.0
```

Se även

PostGIS_GEOS_Version, PostGIS_Full_Version

7.21.5 PostGIS_Liblwgeom_Version

PostGIS_Liblwgeom_Version — Returnerar versionsnumret för liblwgeom-biblioteket. Detta bör matcha versionen av PostGIS.

Synopsis

text **PostGIS_Liblwgeom_Version**();

Beskrivning

Returnerar versionsnumret för liblwgeom-biblioteket/

Exempel

Code

```
SELECT PostGIS_Liblwgeom_Version();
```

Output

```
3.4.0dev 3.3.0rc2-993-g61bdf43a7
```

Se även

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Version

7.21.6 PostGIS_LibXML_Version

PostGIS_LibXML_Version — Returnerar versionsnumret för libxml2-biblioteket.

Synopsis

text **PostGIS_LibXML_Version**();

Beskrivning

Returnerar versionsnumret för LibXML2-biblioteket.

Tillgänglighet: 1,5

Exempel

Code

```
SELECT PostGIS_LibXML_Version();
```

Output

```
2.9.10
```

Se även

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_PROJ_Version, PostGIS_GEOS_Version, PostGIS_Versio

7.21.7 PostGIS_LibJSON_Version

PostGIS_LibJSON_Version — Returnerar versionsnumret för libjson-c-biblioteket.

Synopsis

text **PostGIS_LibJSON_Version**();

Beskrivning

Returnerar versionsnumret för LibJSON-C-biblioteket.

Exempel

Code

```
SELECT PostGIS_LibJSON_Version();
```

Output

```
0.17
```

Se även

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_PROJ_Version, PostGIS_GEOS_Version, PostGIS_Versio

7.21.8 PostGIS_Lib_Build_Date

PostGIS_Lib_Build_Date — Returnerar byggdatum för PostGIS-biblioteket.

Synopsis

text **PostGIS_Lib_Build_Date**();

Beskrivning

Returnerar byggdatum för PostGIS-biblioteket.

Exempel

Code

```
SELECT PostGIS_Lib_Build_Date();
```

Output

```
2023-06-22 03:56:11
```

7.21.9 PostGIS_Lib_Version

PostGIS_Lib_Version — Returnerar versionsnumret för PostGIS-biblioteket.

Synopsis

text **PostGIS_Lib_Version**();

Beskrivning

Returnerar versionsnumret för PostGIS-biblioteket.

Exempel

Code

```
SELECT PostGIS_Lib_Version();
```

Output

```
3.4.0dev
```

Se även

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Version

7.21.10 PostGIS_PROJ_Version

PostGIS_PROJ_Version — Returnerar versionsnumret för PROJ4-biblioteket.

Synopsis

text **PostGIS_PROJ_Version**();

Beskrivning

Returnerar versionsnumret för PROJ-biblioteket och vissa konfigurationsalternativ för proj.

Förbättrad: 3.4.0 inkluderar nu NETWORK_ENABLED, URL_ENDPOINT och DATABASE_PATH för proj.db-platsen

Exempel

Code

```
SELECT PostGIS_PROJ_Version();
```

Output

```
7.2.1 NETWORK_ENABLED=OFF URL_ENDPOINT=https://cdn.proj.org USER_WRITABLE_DIRECTORY=/tmp/ ↩
    proj DATABASE_PATH=/usr/share/proj/proj.db
```

Se även

PostGIS_PROJ_Compiled_Version, PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_Version

7.21.11 PostGIS_PROJ_Compiled_Version

PostGIS_PROJ_Compiled_Version — Returnerar versionsnumret för PROJ-biblioteket som PostGIS byggdes mot.

Synopsis

text **PostGIS_PROJ_Compiled_Version**();

Beskrivning

Returnerar versionsnumret för PROJ-biblioteket, eller mot vilket PostGIS byggdes.

Tillgänglighet: 3.5.0

Exempel

Code

```
SELECT PostGIS_PROJ_Compiled_Version();
```

Output

```
9.1.1
```

Se även

PostGIS_PROJ_Version, PostGIS_Full_Version

7.21.12 PostGIS_Wagyu_Version

PostGIS_Wagyu_Version — Returnerar versionsnumret för det interna Wagyu-biblioteket.

Synopsis

text **PostGIS_Wagyu_Version**();

Beskrivning

Returnerar versionsnumret för det interna Wagyu-biblioteket, eller `NULL` om Wagyu-stöd inte är aktiverat.

Exempel

Code

```
SELECT PostGIS_Wagyu_Version();
```

Output

```
0.5.0 (Internal)
```

Se även

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_PROJ_Version, PostGIS_Lib_Version, PostGIS_LibXM
PostGIS_Version

7.21.13 PostGIS_Scripts_Build_Date

PostGIS_Scripts_Build_Date — Returnerar byggdatum för PostGIS-skript.

Synopsis

text **PostGIS_Scripts_Build_Date**();

Beskrivning

Returnerar byggdatum för PostGIS-skript.

Tillgänglighet: 1.0.0RC1

Exempel

Code

```
SELECT PostGIS_Scripts_Build_Date();
```

Output

```
2023-06-22 03:56:11
```

Se även

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_Version

7.21.14 PostGIS_Scripts_Installed

PostGIS_Scripts_Installed — Returnerar versionen av de PostGIS-skript som är installerade i databasen.

Synopsis

text **PostGIS_Scripts_Installed**();

Beskrivning

Returnerar versionen av de PostGIS-skript som är installerade i databasen.

Notera
Om utdata från denna funktion inte stämmer överens med utdata från PostGIS_Scripts_Released har du förmodligen missat att uppgradera en befintlig databas på rätt sätt. Se avsnittet Uppgradering för mer information.

Tillgänglighet: 0.9.0

Exempel

Code

```
SELECT PostGIS_Scripts_Installed();
```

Output

```
3.4.0dev 3.3.0rc2-993-g61bdf43a7
```

Se även

PostGIS_Full_Version, PostGIS_Scripts_Released, PostGIS_Version

7.21.15 PostGIS_Scripts_Released

PostGIS_Scripts_Released — Returnerar versionsnumret för skriptet postgis.sql som släpptes med det installerade PostGIS-biblioteket.

Synopsis

text **PostGIS_Scripts_Released**();

Beskrivning

Returnerar versionsnumret för skriptet postgis.sql som släpptes med det installerade PostGIS-biblioteket.

Notera
Från och med version 1.1.0 returnerar denna funktion samma värde som PostGIS_Lib_Version. Bevaras för bakåtkompatibilitet.

Tillgänglighet: 0.9.0

Exempel

Code

```
SELECT PostGIS_Scripts_Released();
```

Output

```
3.4.0dev 3.3.0rc2-993-g61bdf43a7
```

Se även

PostGIS_Full_Version, PostGIS_Scripts_Installed, PostGIS_Lib_Version

7.21.16 PostGIS_Version

PostGIS_Version — Returnerar PostGIS versionsnummer och kompileringstidsalternativ.

Synopsis

text **PostGIS_Version**();

Beskrivning

Returnerar PostGIS versionsnummer och kompileringstidsalternativ.

Exempel

Code

```
SELECT PostGIS_Version();
```

Output

```
3.4 USE_GEOS=1 USE_PROJ=1 USE_STATS=1
```

Se även

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version

7.22 Grand Unified Custom Variables (GUC)

7.22.1 postgis.gdal_datapath

postgis.gdal_datapath — Ett konfigurationsalternativ för att tilldela värdet för GDAL:s GDAL_DATA-alternativ. Om den inte är inställd används den miljöinställda variabeln GDAL_DATA.

Beskrivning

En PostgreSQL GUC-variabel för att ställa in värdet på GDAL: s GDAL_DATA-alternativ. Värdet `postgis.gdal` bör vara den fullständiga fysiska sökvägen till GDAL:s datafiler.

Detta konfigurationsalternativ är mest användbart för Windows-plattformar där GDAL:s datafilsökväg inte är hårdkodad. Detta alternativ bör också ställas in när GDAL:s datafiler inte finns i GDAL:s förväntade sökväg.

Notera
Detta alternativ kan ställas in i PostgreSQL: s konfigurationsfil postgresql.conf. Det kan också ställas in via anslutning eller transaktion.

Tillgänglighet: 2.2.0

Notera
Ytterligare information om GDAL_DATA finns på GDAL:s konfigurationsalternativ.

Exempel

Ställ in och återställ `postgis.gdal_datapath`

Code

```
SET postgis.gdal_datapath TO '/usr/local/share/gdal.hidden';
SET postgis.gdal_datapath TO default;
```

Inställning av fönster för en viss databas

Code

```
ALTER DATABASE gisdb
SET postgis.gdal_datapath = 'C:/Program Files/PostgreSQL/9.3/gdal-data';
```

Se även

PostGIS_GDAL_Version, ST_Transform

7.22.2 postgis.gdal_enabled_drivers

postgis.gdal_enabled_drivers — Ett konfigurationsalternativ för att ställa in de aktiverade GDAL-drivrutinerna i PostGIS-miljön. Påverkar GDAL-konfigurationsvariabeln GDAL_SKIP.

Beskrivning

Ett konfigurationsalternativ för att ställa in de aktiverade GDAL-drivrutinerna i PostGIS-miljön. Påverkar GDAL-konfigurationsvariabeln GDAL_SKIP. Det här alternativet kan ställas in i PostgreSQL: s konfigurationsfil: postgresql.conf. Det kan också ställas in via anslutning eller transaktion.

Det initiala värdet för `postgis.gdal_enabled_drivers` kan också ställas in genom att skicka miljövariabeln `POSTGIS_GDAL_ENABLED_DRIVERS` med listan över aktiverade drivrutiner till processen som startar PostgreSQL.

Aktiverade GDAL specificerade drivrutiner kan specificeras med drivrutinens kortnamn eller kod. Kortnamn eller koder för drivrutiner finns på GDAL Raster Formats. Flera drivrutiner kan specificeras genom att lägga till ett mellanslag mellan varje drivrutin.

Notera
Det finns tre specialkoder tillgängliga för `postgis.gdal_enabled_drivers`. Koderna är skiftlägeskänsliga.

- `DISABLE_ALL` inaktiverar alla GDAL-drivrutiner. Om `DISABLE_ALL` finns där åsidosätter den alla andra värden i `postgis.gdal_enabled_drivers`.
- `ENABLE_ALL` aktiverar alla GDAL-drivrutiner.
- `VSICURL` aktiverar GDAL:s virtuella filsystem `/vsicurl/`.

När `postgis.gdal_enabled_drivers` är satt till DISABLE_ALL kommer försök att använda out-db rasters, ST_FromGDALRaster(), ST_AsGDALRaster(), ST_AsTIFF(), ST_AsJPEG() och ST_AsPNG() att resultera i felmeddelanden.

Notera
I standardinstallationen av PostGIS är `postgis.gdal_enabled_drivers` inställd på DISABLE_ALL.

Notera
Ytterligare information om GDAL_SKIP finns på GDAL:s konfigurationsalternativ.

Tillgänglighet: 2.2.0

Exempel

Set and reset `postgis.gdal_enabled_drivers` for the current session.

Code

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.gdal_enabled_drivers = default;
```

Set specific drivers for all new connections to a database.

Code

```
ALTER DATABASE mygisdb SET postgis.gdal_enabled_drivers TO 'GTiff PNG JPEG';
```

Set the default for the whole database cluster to enable all drivers. This requires superuser access. Also note that database, session, and user settings override this.

This example writes the setting to postgres.auto.conf.

Code

```
ALTER SYSTEM SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Reload the PostgreSQL configuration so the changed setting is applied.

Code

```
SELECT pg_reload_conf();
```

Se även

ST_FromGDALRaster, ST_AsGDALRaster, ST_AsTIFF, ST_AsPNG, ST_AsJPEG, postgis.enable_outdb_raster

7.22.3 postgis.enable_outdb_rasters

postgis.enable_outdb_rasters — Ett booleskt konfigurationsalternativ för att aktivera åtkomst till out-db-rasterband.

Beskrivning

Ett boolskt konfigurationsalternativ för att aktivera åtkomst till out-db rasterband. Detta alternativ kan ställas in i PostgreSQL: s konfigurationsfil: postgresql.conf. Det kan också ställas in via anslutning eller transaktion.

Det initiala värdet för `postgis.enable_outdb_rasters` kan också ställas in genom att skicka miljövariabeln `POSTGIS_ENABLE_OUTDB_RASTERS` med ett icke-nollvärde till processen som startar PostgreSQL.

Notera
Även om `postgis.enable_outdb_rasters` är True, avgör GUC `postgis.gdal_enabled_drivers` vilka rasterformat som är tillgängliga.

Notera
I standardinstallationen av PostGIS är `postgis.enable_outdb_rasters` inställd på False.

Tillgänglighet: 2.2.0

Exempel

Set and reset `postgis.enable_outdb_rasters` for the current session.

Code

```
SET postgis.enable_outdb_rasters TO True;
SET postgis.enable_outdb_rasters = default;
SET postgis.enable_outdb_rasters = True;
SET postgis.enable_outdb_rasters = False;
```

Set the default for all new connections to a database.

Code

```
ALTER DATABASE gisdb SET postgis.enable_outdb_rasters = true;
```

Set the default for the whole database cluster. This requires superuser access. Also note that database, session, and user settings override this.

This example writes the setting to postgres.auto.conf.

Code

```
ALTER SYSTEM SET postgis.enable_outdb_rasters = true;
```

Reload the PostgreSQL configuration so the changed setting is applied.

Code

```
SELECT pg_reload_conf();
```

Se även

postgis.gdal_enabled_drivers postgis.gdal_vsi_options

7.22.4 postgis.gdal_vsi_options

postgis.gdal_vsi_options — En strängkonfiguration för att ställa in alternativ som används när du arbetar med ett out-db-raster.

Beskrivning

En strängkonfiguration för att ställa in alternativ som används när du arbetar med en out-db-raster. Konfigurationsalternativ styr saker som hur mycket utrymme GDAL tilldelar lokal datacache, om översikter ska läsas och vilka åtkomstnycklar som ska användas för fjärranslutna out-db-datakällor.

Tillgänglighet: 3.2.0

Exempel

Ställ in `postgis.gdal_vsi_options` för aktuell session:

Code

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx AWS_SECRET_ACCESS_KEY= ↩
    yyyyyyyyyyyyyyyyyyyyyyyyyyy';
```

Ställ in `postgis.gdal_vsi_options` bara för den *aktuella transaktionen* med hjälp av nyckelordet `LOCAL`:

Code

```
SET LOCAL postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx ↩
    AWS_SECRET_ACCESS_KEY=yyyyyyyyyyyyyyyyyyyyyyyyyy';
```

Se även

postgis.enable_outdb_rasters postgis.gdal_enabled_drivers

7.22.5 postgis.gdal_cpl_debug

postgis.gdal_cpl_debug — En boolesk konfiguration för att aktivera eller inaktivera loggning av GDAL-felsökningsmeddelanden.

Beskrivning

Som standard skrivs GDAL-loggning ut till stderr, och felsökningsmeddelanden på lägre nivå skrivs inte ut alls. Att göra denna GUC till sant kommer att orsaka att GDAL-loggning skickas till PostgreSQL-loggningsströmmen, så att du kan se mer eller mindre av det genom att ändra `client_min_message` PostgreSQL GUC.

Tillgänglighet: 3.6.0

Se även

postgis.enable_outdb_rasters postgis.gdal_enabled_drivers

7.23 Felsökningsfunktioner

7.23.1 PostGIS_AddBBox

PostGIS_AddBBox — Lägg till en begränsningsbox till geometrin.

Synopsis

geometry **PostGIS_AddBBox**(geometry geomA);

Beskrivning

Lägg till en begränsningsbox i geometrin. Detta skulle göra frågor baserade på bounding box snabbare, men kommer att öka geometrins storlek.

Notera
Begränsningsrutor läggs automatiskt till geometrier så i allmänhet behövs inte detta om inte den genererade begränsningsrutan på något sätt blir skadad eller om du har en gammal installation som saknar begränsningsrutor. Då måste du ta bort de gamla och lägga till dem igen.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
UPDATE sometable
SET geom =  PostGIS_AddBBox(geom)
WHERE PostGIS_HasBBox(geom) = false;
```

Se även

PostGIS_DropBBox, PostGIS_HasBBox

7.23.2 PostGIS_DropBBox

PostGIS_DropBBox — Ta bort begränsningsboxens cache från geometrin.

Synopsis

geometry **PostGIS_DropBBox**(geometry geomA);

Beskrivning

Ta bort bounding box-cachen från geometrin. Detta minskar geometrins storlek, men gör frågor baserade på bounding box långsammare. Det används också för att släppa en korrupt bounding box. Ett tecken på en korrupt cachelagrad bounding box är när ST_Intersects och andra relationsfrågor utelämnar geometrier som rätteligen borde returnera true.

Notera
Begränsningsrutor läggs automatiskt till geometrier och förbättrar hastigheten på frågor så i allmänhet behövs detta inte om inte den genererade begränsningsrutan på något sätt blir skadad eller om du har en gammal installation som saknar begränsningsrutor. Då måste du ta bort den gamla och lägga till den på nytt. Denna typ av korruption har observerats i 8.3-8.3.6-serien varigenom cachade bboxar inte alltid omräknades när en geometri ändrades och uppgradering till en nyare version utan en omladdning av dumpning kommer inte att korrigera redan korrupta boxar. Så man kan korrigera manuellt med hjälp av nedan och lägga till bboxen igen eller göra en dump-omladdning.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

This example drops cached bounding boxes where the stored box is wrong. Applying `Box2D` to `ST_AsBinary` forces a fresh box calculation, while `Box2D` on the table geometry reads the cached box.

Code

```
UPDATE sometable
SET geom =  PostGIS_DropBBox(geom)
WHERE Not (Box2D(ST_AsBinary(geom)) = Box2D(geom));
```

Output

```
UPDATE ...
```

After dropping corrupt cached boxes, re-add them where needed.

Code

```
UPDATE sometable
SET geom =  PostGIS_AddBBox(geom)
WHERE Not PostGIS_HasBBOX(geom);
```

Output

```
UPDATE ...
```

Se även

PostGIS_AddBBox, PostGIS_HasBBox, Box2D

7.23.3 PostGIS_HasBBox

PostGIS_HasBBox — Returnerar TRUE om bboxen för denna geometri är cachad, FALSE annars.

Synopsis

boolean **PostGIS_HasBBox**(geometry geomA);

Beskrivning

Returnerar TRUE om bboxen för denna geometri är cachelagrad, FALSE annars. Använd PostGIS_AddBBox och PostGIS_DropBBox för att styra cachelagring.

Denna metod stöder cirkulära strängar och kurvor.

Exempel

Code

```
SELECT geom
FROM sometable WHERE PostGIS_HasBBox(geom) = false;
```

Se även

PostGIS_AddBBox, PostGIS_DropBBox

Kapitel 8

SFCGAL Funktioner Referens

SFCGAL är ett C++-bibliotek runt CGAL som tillhandahåller avancerade 2D- och 3D-spatiala funktioner. För robusthetens skull har geometrikoordinaterna en exakt representation av rationella tal.

Installationsanvisningar för biblioteket finns på SFCGAL:s hemsida(http://www.sfcgal.org). För att aktivera funktionerna använd `create extension postgis_sfcgal`.

8.1 SFCGAL-hanteringsfunktioner

8.1.1 postgis_sfcgal_version

postgis_sfcgal_version — Returnerar den version av SFCGAL som används

Synopsis

text **postgis_sfcgal_version**(void);

Beskrivning

Returnerar den version av SFCGAL som används

Tillgänglighet: 2.1.0

Denna metod behöver SFCGAL-backend.

Se även

postgis_sfcgal_full_version

8.1.2 postgis_sfcgal_full_version

postgis_sfcgal_full_version — Returnerar den fullständiga versionen av SFCGAL som används, inklusive CGAL- och Boost-versioner

Synopsis

text **postgis_sfcgal_full_version**(void);

Beskrivning

Returnerar den fullständiga versionen av SFCGAL som används, inklusive CGAL- och Boost-versioner

Tillgänglighet: 3.3.0

Denna metod behöver SFCGAL-backend.

Se även

postgis_sfcgal_version

8.2 SFCGAL-accessorer och Setters

8.2.1 CG_ForceLHR

CG_ForceLHR — Tvinga fram LHR-orientering

Synopsis

geometry **CG_ForceLHR**(geometry geom);

Beskrivning

Returns an equivalent geometry whose polygonal components follow the left-hand rule: exterior rings are counter-clockwise and interior rings are clockwise. Rings are reversed as needed; coordinates and dimensionality are preserved. The input must be valid.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Reorient a polygon and its hole to the left-hand rule.

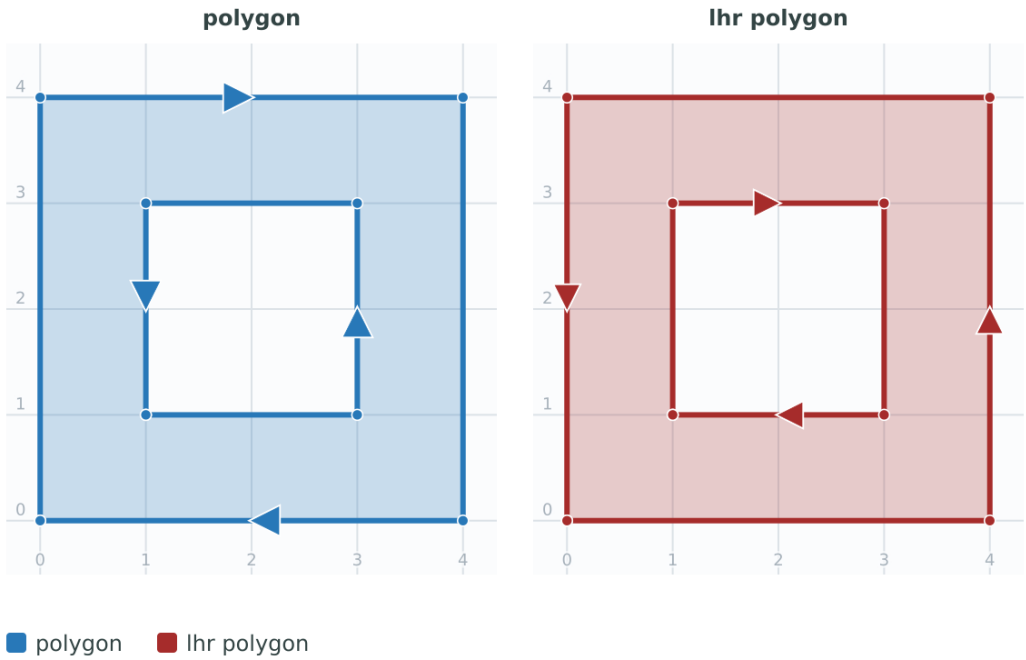
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,0 4,4 4,4 0,0 0),
                  (1 1,3 1,3 3,1 3,1 1))'::geometry AS geom
)
SELECT geom AS input_polygon,
       CG_ForceLHR(geom) AS lhr_polygon,
       ST_IsPolygonCCW(geom) AS input_is_lhr,
       ST_IsPolygonCCW(CG_ForceLHR(geom)) AS output_is_lhr
FROM data;
```

Output

```
POLYGON((0 0,0 4,4 4,4 0,0 0),(1 1,3 1,3 3,1 3,1 1)) | POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,1 3,3 3,3 1,1 1)) | f | t
```

Figure



8.2.2 CG_IsPlanar

CG_IsPlanar — Kontrollera om en yta är plan eller inte

Synopsis

boolean **CG_IsPlanar**(geometry geom);

Beskrivning

Tests whether the input geometry lies in a single plane in 3D.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

8.2.3 CG_IsSolid

CG_IsSolid — Testar om geometrin är en solid. Ingen validitetskontroll utförs.

Synopsis

boolean **CG_IsSolid**(geometry geom1);

Beskrivning

Returns `true` when the input carries the solid flag, such as a geometry produced by CG_MakeSolid. A closed `PolyhedralSurface` remains a surface until it is explicitly converted. This tests the representation, not shell closure or validity; no validity check is performed.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Compare the same closed shell before and after converting it to a solid.

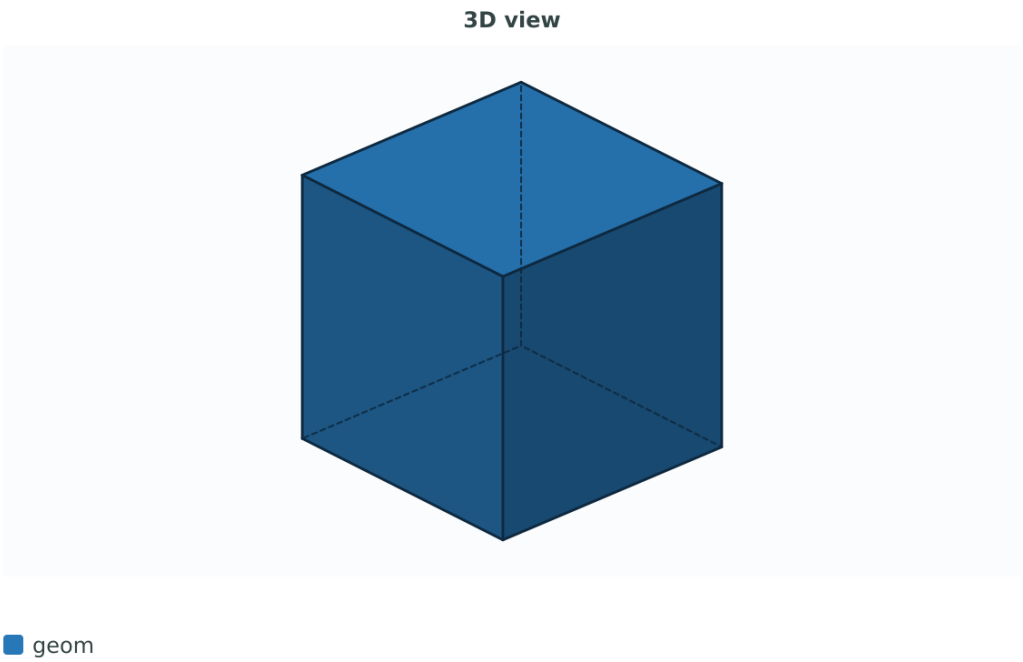
Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z (
    ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
    ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
    ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
  )'::geometry AS geom
)
SELECT CG_IsSolid(geom) AS shell_is_solid,
       CG_IsSolid(CG_MakeSolid(geom)) AS solid_is_solid
FROM data;
```

Output

```
f | t
```

Figure



8.2.4 CG_MakeSolid

CG_MakeSolid — Casta geometrin till en solid. Ingen kontroll utförs. För att få en giltig solid måste indatageometrin vara en sluten polyedrisk yta eller en sluten TIN.

Synopsis

geometry **CG_MakeSolid**(geometry geom1);

Beskrivning

Marks a PolyhedralSurface or TIN as a solid. The input is expected to describe a closed shell; the function sets the solid representation and does not validate the shell.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

8.2.5 CG_Orientation

CG_Orientation — Bestäm ytans orientering

Synopsis

integer **CG_Orientation**(geometry geom);

Beskrivning

Funktionen gäller endast för polygoner. Den returnerar -1 om polygonen är orienterad moturs och 1 om polygonen är orienterad medurs.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

8.2.6 CG_Area

CG_Area — Beräknar arean av en geometri

Synopsis

double precision **CG_Area**(geometry geom);

Beskrivning

Beräknar arean av en geometri.

Utförs av SFCGAL-modulen

> **Notera**
> OBS: Denna funktion returnerar ett värde med dubbel precision som representerar området.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Exempel på geometri

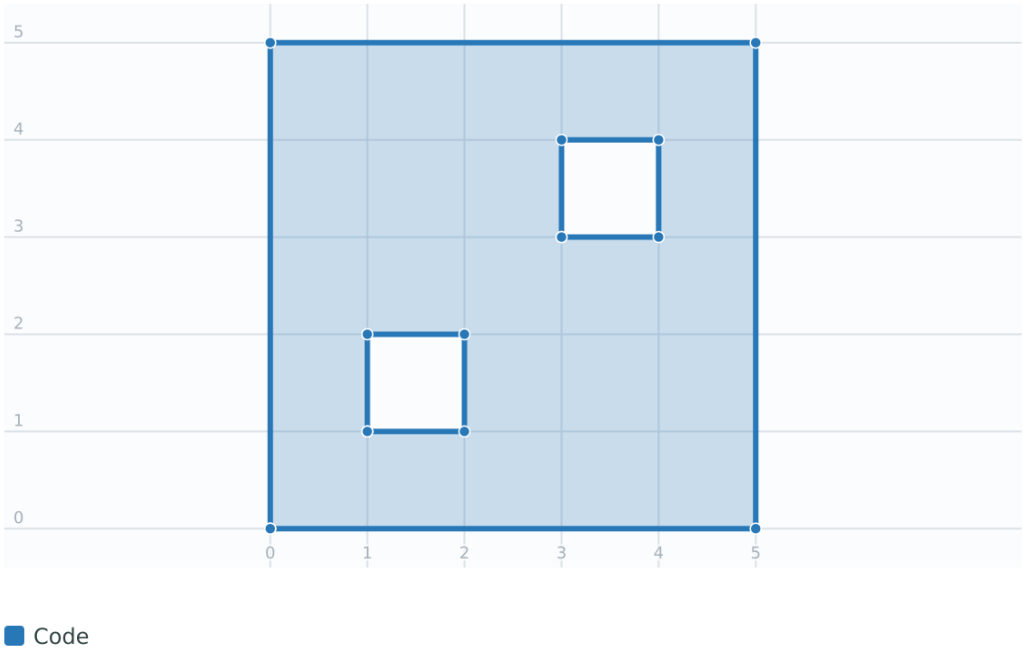
Code

```
SELECT CG_Area('Polygon ((0 0,0 5,5 5,5 0,0 0),(1 1,2 1,2 2,1 2,1 1),(3 3,4 3,4 4,3 4,3 3))
    ');
```

Output

```
23
```

Figure



Se även

CG_3DArea, ST_Area

8.2.7 CG_3DArea

CG_3DArea — Beräknar area för 3D-ytgeometrier. Returnerar 0 för solider.

Synopsis

float**CG_3DArea**(geometry geom1);

Beskrivning

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 8.1, 10.5

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Obs: Som standard är en PolyhedralSurface som byggts från WKT en ytgeometri, inte en solid. Den har därför en ytarea. När den har konverterats till en solid har den ingen area.

Code

```
SELECT CG_3DArea(geom) As cube_surface_area,
CG_3DArea(CG_MakeSolid(geom)) As solid_surface_area
FROM (SELECT 'POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry) As f(geom);
```

Output

```
cube_surface_area | solid_surface_area
------------------+--------------------
6                 | 0
```

Se även

CG_Area, CG_MakeSolid, CG_IsSolid, CG_Area

8.2.8 CG_Volume

CG_Volume — Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.

Synopsis

float **CG_Volume**(geometry geom1);

Beskrivning

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 9.1 (samma som CG_3DVolume)

Closed surface geometries can have non-zero volume. Use CG_MakeSolid when the geometry value itself must carry the solid flag.

Exempel

A closed polyhedral surface has volume. CG_MakeSolid makes the solid representation explicit.

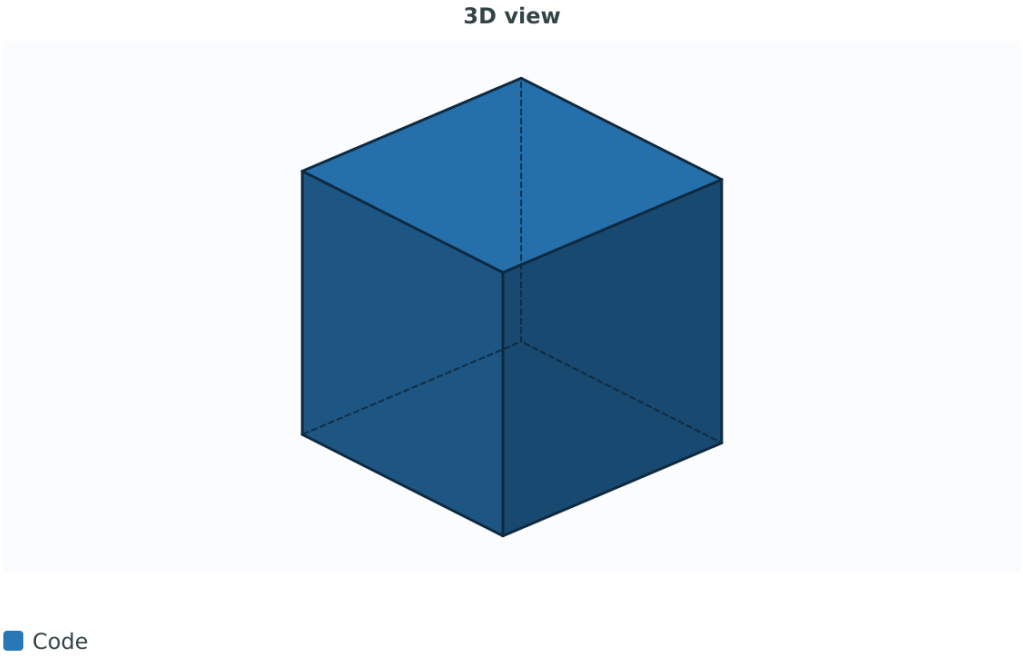
Code

```
SELECT CG_Volume(geom) As cube_surface_vol,
CG_Volume(CG_MakeSolid(geom)) As solid_surface_vol
FROM (SELECT 'POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry) As f(geom);
```

Output

```
cube_surface_vol | solid_surface_vol
-----------------+------------------
1                | 1
```

Figure



Se även

CG_3DArea, CG_MakeSolid, CG_IsSolid

8.3 SFCGAL bearbetnings- och relationsfunktioner

8.3.1 CG_Intersection

CG_Intersection — Beräknar skärningspunkten mellan två geometrier

Synopsis

geometry **CG_Intersection**(geometry geomA , geometry geomB);

Beskrivning

Beräknar skärningspunkten mellan två geometrier.

Utförs av SFCGAL-modulen

Notera
OBS: Denna funktion returnerar en geometri som representerar korsningen.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Exempel på geometri

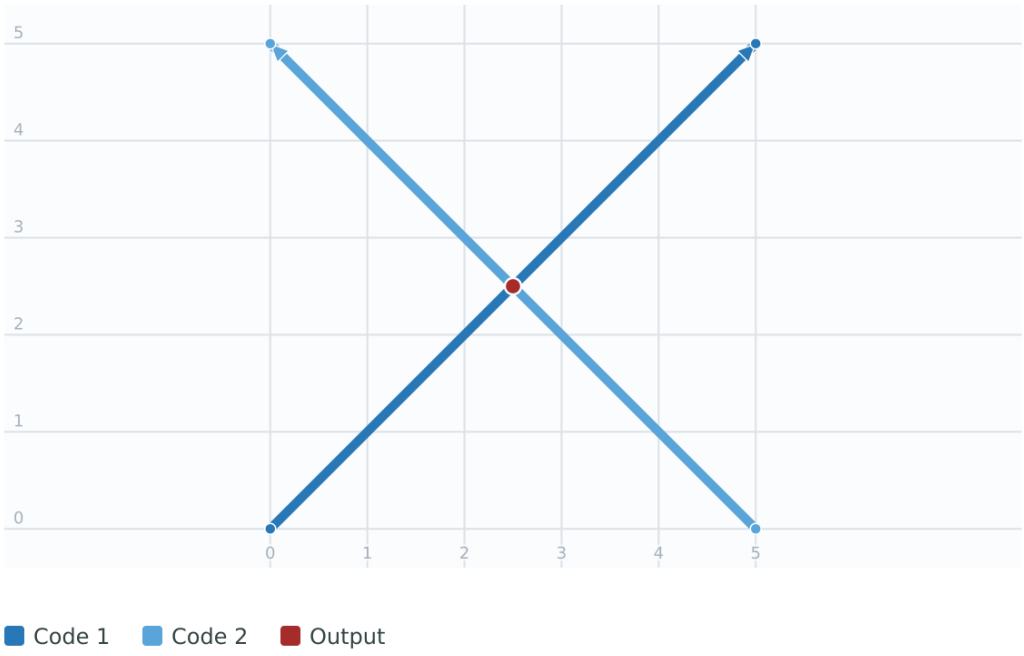
Code

```
SELECT CG_Intersection('LINESTRING(0 0,5 5)', 'LINESTRING(5 0,0 5)');
```

Output

```
POINT(2.5 2.5)
```

Figure



Se även

CG_3DIntersection, ST_Intersection

8.3.2 CG_Intersects

CG_Intersects — Testar om två geometrier skär varandra (de har minst en gemensam punkt)

Synopsis

boolean **CG_Intersects**(geometry geomA , geometry geomB);

Beskrivning

Returnerar `true` om två geometrier korsar varandra. Geometrier korsar varandra om de har någon gemensam punkt.

Utförs av SFCGAL-modulen

Notera
OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel på geometri

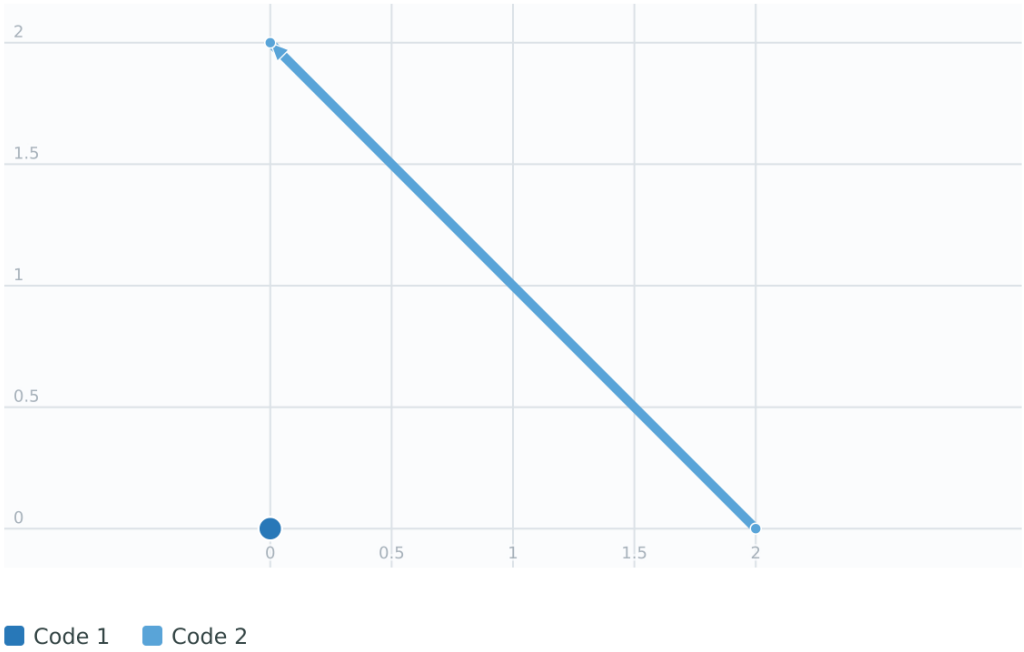
Code

```
SELECT CG_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
f
```

Figure



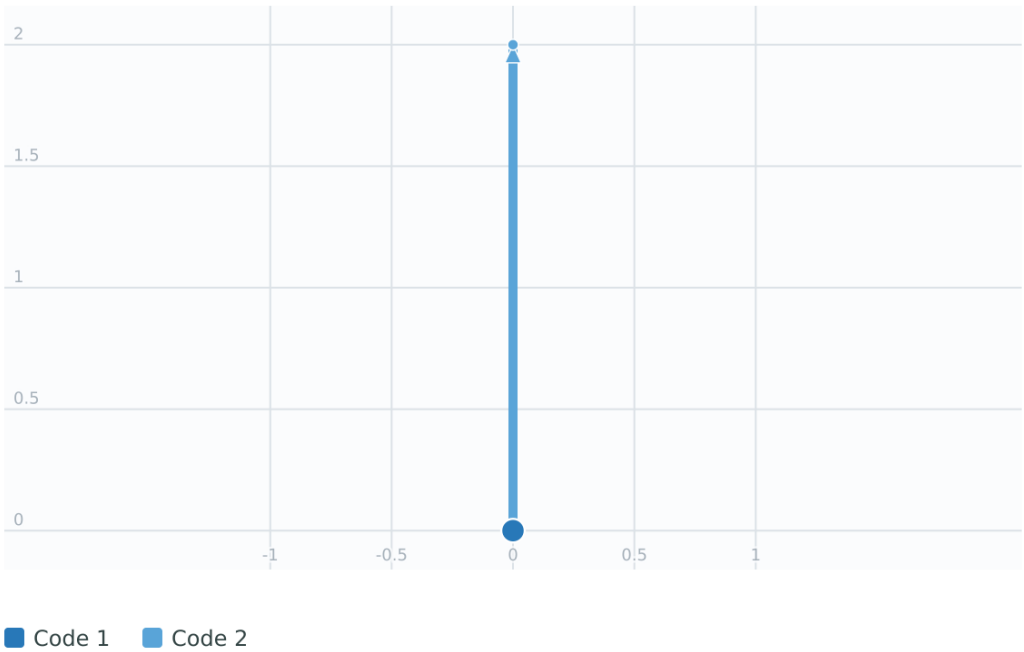
Code

```
SELECT CG_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Output

```
t
```

Figure



Se även

CG_3DIntersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.3 CG_3DIntersects

CG_3DIntersects — Testar om två 3D-geometrier korsar varandra

Synopsis

boolean **CG_3DIntersects**(geometry geomA , geometry geomB);

Beskrivning

Testar om två 3D-geometrier skär varandra. 3D-geometrier skär varandra om de har någon gemensam punkt i det tredimensionella rummet.

Utförs av SFCGAL-modulen

Notera
OBS: detta är den tillåtna”versionen som returnerar ett boolean, inte ett heltal.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel på geometri

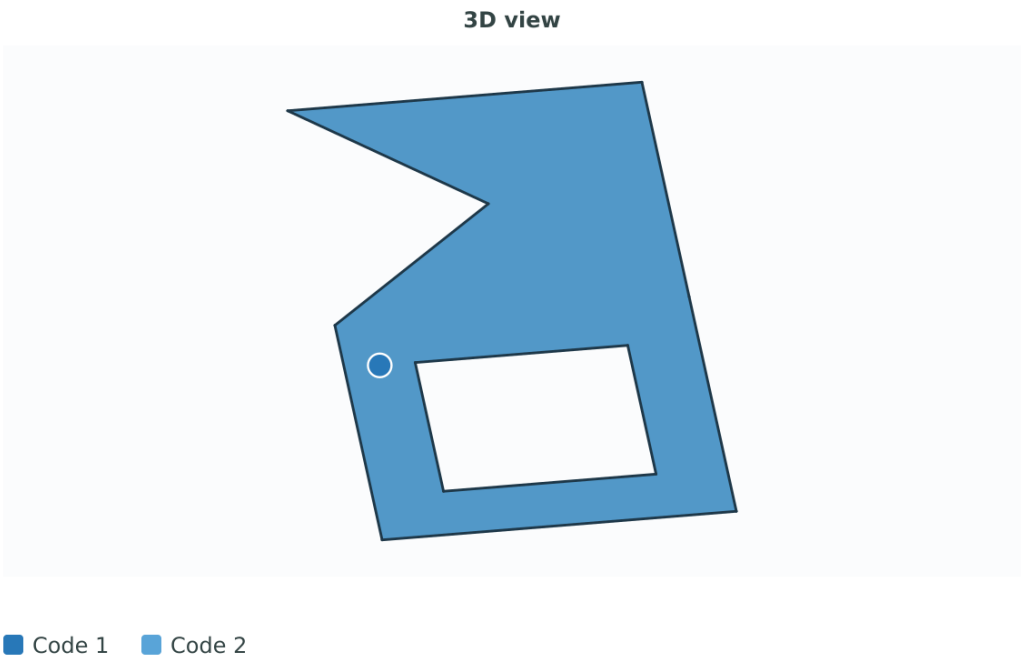
Code

```
SELECT CG_3DIntersects('POINT(1.2 0.1 0)', 'POLYHEDRALSURFACE(((0 0 0,0.5 0.5 0,1 0 0,1 1 ↩
    0,0 1 0,0 0 0)),((1 0 0,2 0 0,2 1 0,1 1 0,1 0 0),(1.2 0.2 0,1.2 0.8 0,1.8 0.8 0,1.8 0.2 ↩
    0,1.2 0.2 0)))');
```

Output

```
t
```

Figure



Se även

CG_Intersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.4 CG_Difference

CG_Difference — Beräknar den geometriska skillnaden mellan två geometrier

Synopsis

geometry **CG_Difference**(geometry geomA , geometry geomB);

Beskrivning

Beräknar den geometriska skillnaden mellan två geometrier. Den resulterande geometrin är en uppsättning punkter som finns i geomA men inte i geomB.

Utförs av SFCGAL-modulen

Notera
OBS: denna funktion returnerar en geometri.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel på geometri

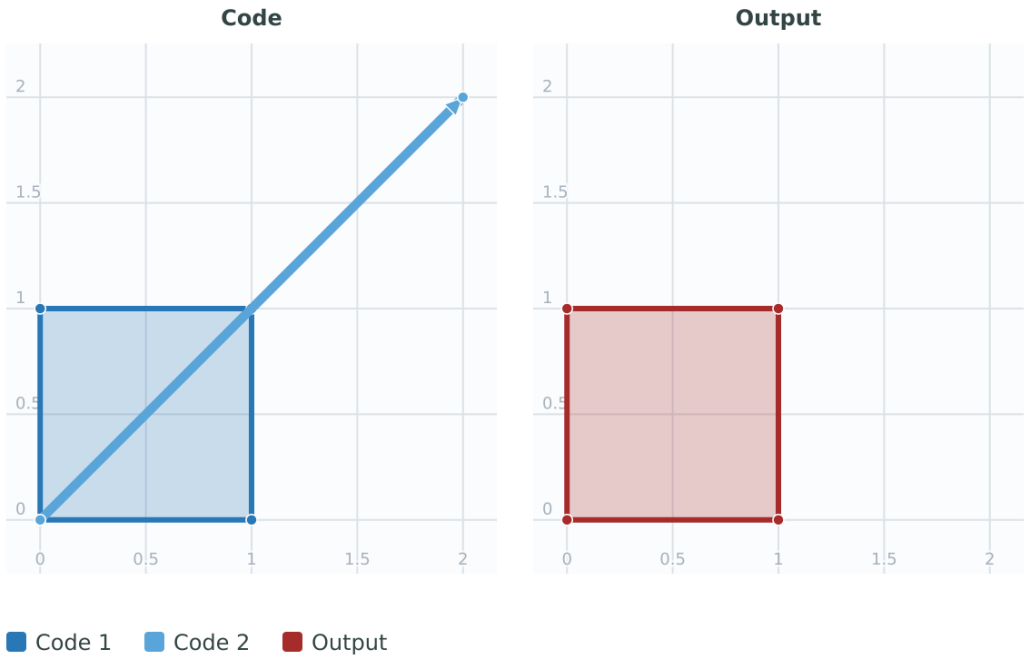
Code

```
SELECT CG_Difference('POLYGON((0 0,0 1,1 1,1 0,0 0))'::geometry, 'LINESTRING(0 0,2 2)':: ↩
    geometry);
```

Output

```
POLYGON((0 0,1 0,1 1,0 1,0 0))
```

Figure



Se även

CG_3DDifference, ST_Difference

8.3.5 CG_3DDifference

CG_3DDifference — Utföra 3D-differens

Synopsis

geometry **CG_3DDifference**(geometry geom1, geometry geom2);

Beskrivning

Returnerar den del av geom1 som inte är en del av geom2.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

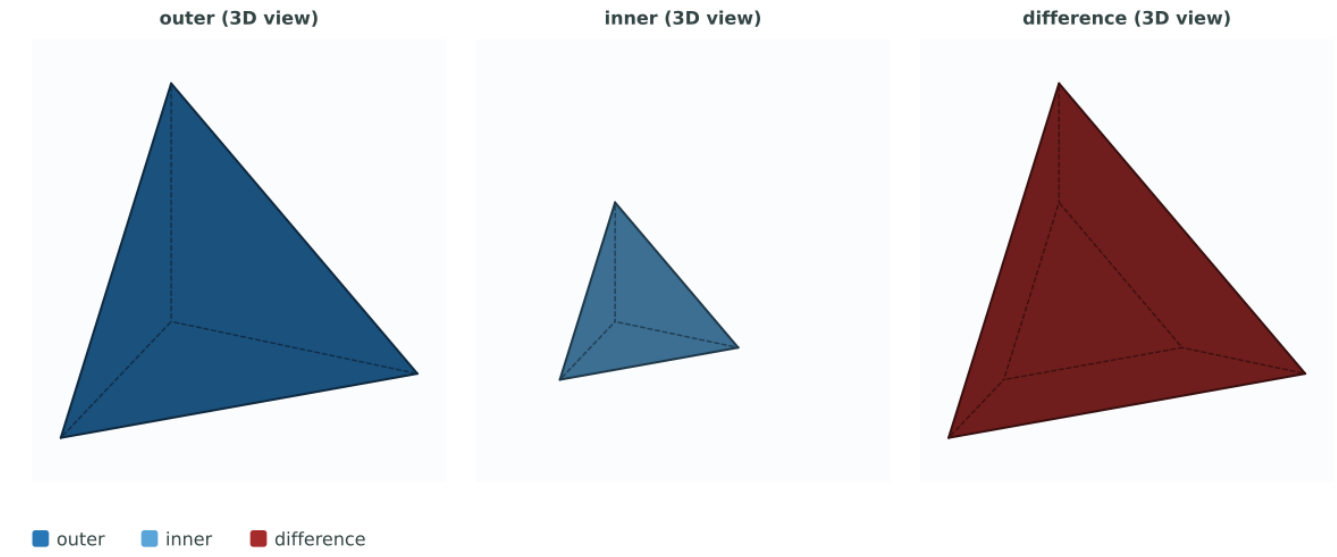
Exempel

Subtract a smaller tetrahedral solid and render the two inputs and the actual remaining solid separately.

Code

```
WITH data AS (
  SELECT CG_MakeSolid('POLYHEDRALSURFACE Z (
    ((0 0 0,0 4 0,4 0 0,0 0 0)),
    ((0 0 0,4 0 0,0 0 4,0 0 0)),
    ((0 0 0,0 0 4,0 4 0,0 0 0)),
    ((4 0 0,0 4 0,0 0 4,4 0 0))
  )'::geometry) AS outer_solid
), solids AS (
  SELECT outer_solid,
         ST_Scale(outer_solid, 0.5, 0.5, 0.5) AS inner_solid
  FROM data
)
SELECT outer_solid AS input_outer,
       inner_solid AS input_inner,
       CG_3DDifference(outer_solid, inner_solid) AS difference
FROM solids;
```

Figure



Se även

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DUnion

8.3.6 CG_Distance

CG_Distance — Beräknar det minsta avståndet mellan två geometrier

Synopsis

double precision **CG_Distance**(geometry geomA , geometry geomB);

Beskrivning

Beräknar det minsta avståndet mellan två geometrier.
Utförs av SFCGAL-modulen

Notera
OBS: Denna funktion returnerar ett värde med dubbel precision som representerar avståndet.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel på geometri

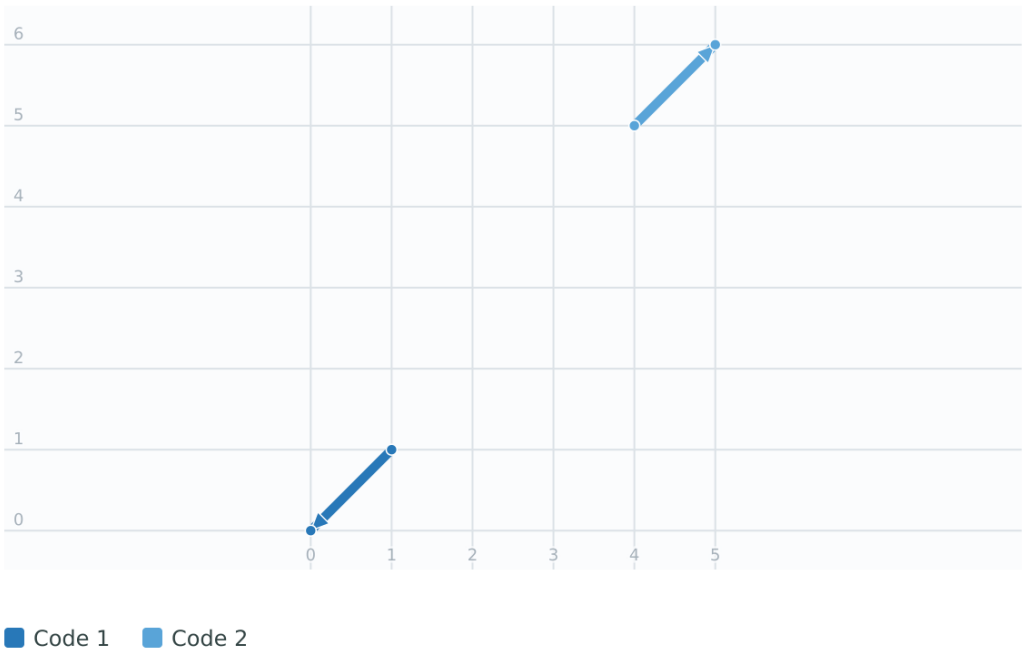
Code

```
SELECT CG_Distance('LINESTRING(0.0 0.0,-1.0 -1.0)', 'LINESTRING(3.0 4.0,4.0 5.0)');
```

Output

```
5
```

Figure



Se även

CG_3DDistance, CG_Distance

8.3.7 CG_3DDistance

CG_3DDistance — Beräknar det minsta 3D-avståndet mellan två geometrier

Synopsis

double precision **CG_3DDistance**(geometry geomA , geometry geomB);

Beskrivning

Beräknar det minsta 3D-avståndet mellan två geometrier.
Utförs av SFCGAL-modulen

> **Notera**
> OBS: Denna funktion returnerar ett värde med dubbel precision som representerar 3D-avståndet.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel på geometri

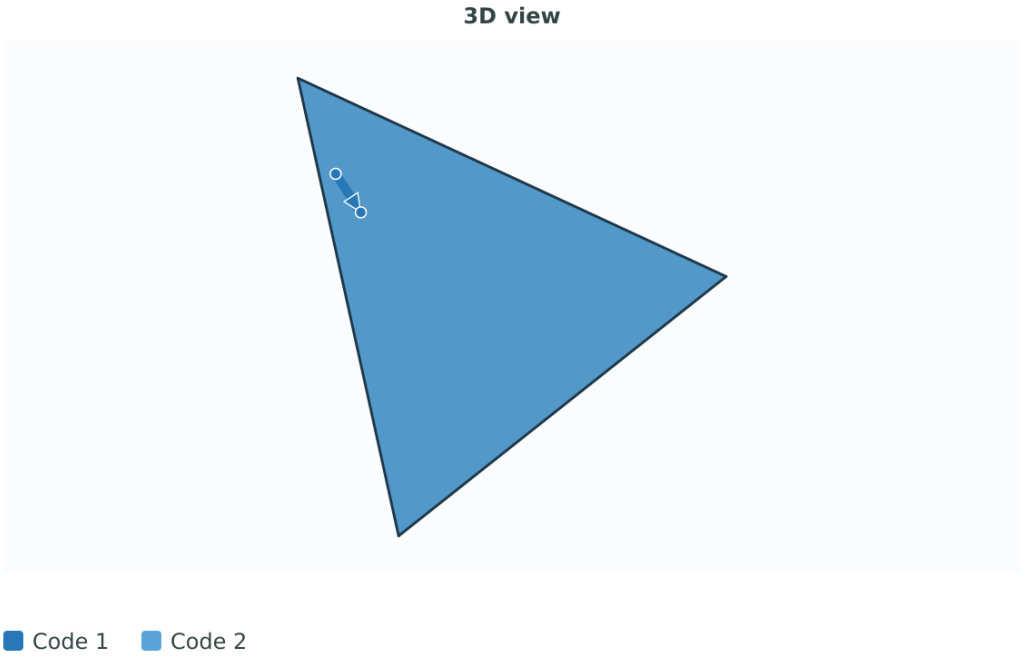
Code

```
SELECT CG_3DDistance('LINESTRING(-1.0 0.0 2.0,1.0 0.0 3.0)', 'TRIANGLE((-4.0 0.0 1.0,4.0 ↩
    0.0 1.0,0.0 4.0 1.0,-4.0 0.0 1.0))');
```

Output

```
1
```

Figure



Se även

CG_Distance, ST_3DDistance

8.3.8 CG_3DConvexHull

CG_3DConvexHull — Beräknar den konvexa 3D-skålen för en geometri.

Synopsis

geometry **CG_3DConvexHull**(geometry geom1);

Beskrivning

Computes the smallest convex three-dimensional geometry that contains the input. Unlike ST_ConvexHull, this function uses Z coordinates when constructing the hull.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Compute the 3D convex hull of a 3D LineString.

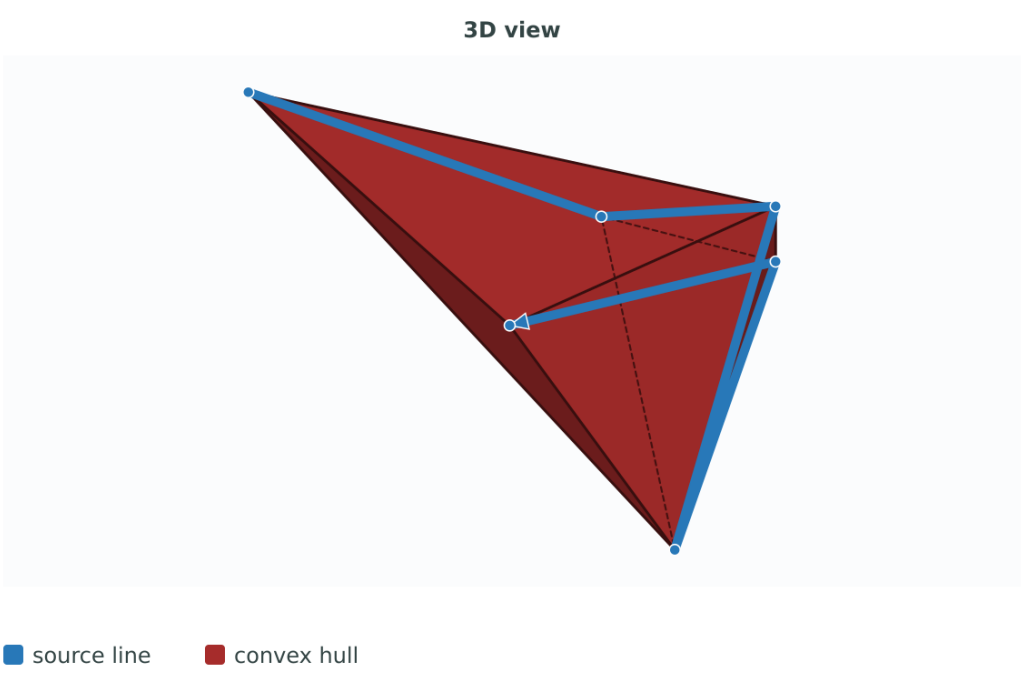
Code

```
WITH data AS (
  SELECT 'LINESTRING Z(0 0 5,1 5 3,5 7 6,9 5 3,5 7 5,6 3 5)'::geometry AS source_line
)
SELECT CG_3DConvexHull(source_line) AS convex_hull
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((1 5 3,9 5 3,0 0 5,1 5 3)),((1 5 3,0 0 5,5 7 6,1 5 3)),((5 7 6,5 7 5,1 5 3,5 7 6)),((0 0 5,6 3 ↩
    5,5 7 6,0 0 5)),((6 3 5,9 5 3,5 7 6,6 3 5)),((0 0 5,9 5 3,6 3 5,0 0 5)),((9 5 3,5 7 5,5 7 6,9 5 3)),((1 5 3,5 7 ↩
    5,9 5 3,1 5 3)))
```

Figure



Compare the extruded source geometry with its 3D convex hull.

Code

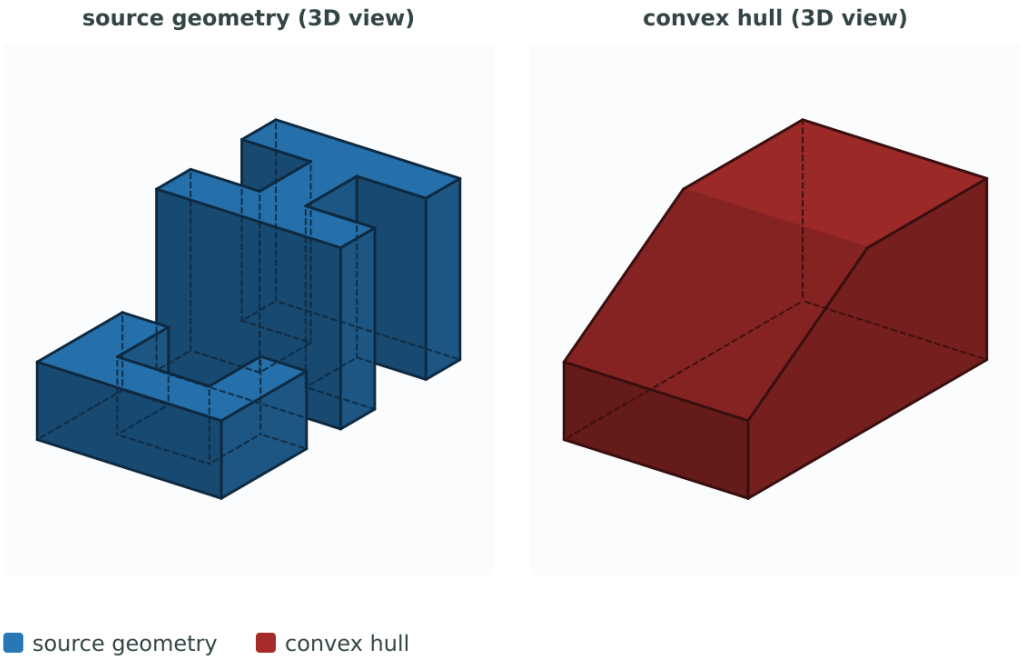
```
WITH footprints(label, geom, height) AS (VALUES
  ('C', 'POLYGON((0 0,0 8,5 8,5 6,2 6,2 2,5 2,5 0,0 0))'::geometry, 3),
  ('H', 'POLYGON((7 0,9 0,9 3,12 3,12 0,14 0,14 8,12 8,12 5,9 5,9 8,7 8,7 0))'::geometry, ↩
    7)
```

```
), parts AS (
  SELECT label, CG_Extrude(geom, 0, 0, height) AS geom
  FROM footprints
), shapes AS (
  SELECT ST_Collect(geom ORDER BY label) AS source_geometry
  FROM parts
)
SELECT
  source_geometry AS input_source_geometry,
  CG_3DConvexHull(source_geometry) AS convex_hull
FROM shapes;
```

Output

```
-[ RECORD 1 ]----------
input_source_geometry | GEOMETRYCOLLECTION(POLYHEDRALSURFACE(((0 0 0,0 8 0,5 8 0,5 6 0,2 6 0,2 2 0,5 2 0,5 0 0,0 0 0)) ↩
     ,((0 0 3,5 0 3,5 2 3,2 2 3,2 6 3,5 6 3,5 8 3,0 8 3,0 0 3)),((0 0 0,0 0 3,0 8 3,0 8 0,0 0 0)),((0 8 0,0 8 3,5 8  ↩
     3,5 8 0,0 8 0)),((5 8 0,5 8 3,5 6 3,5 6 0,5 8 0)),((5 6 0,5 6 3,2 6 3,2 6 0,5 6 0)),((2 6 0,2 6 3,2 2 3,2 2 0,2 6 ↩
      0)),((2 2 0,2 2 3,5 2 3,5 2 0,2 2 0)),((5 2 0,5 2 3,5 0 3,5 0 0,5 2 0)),((5 0 0,5 0 3,0 0 3,0 0 0,5 0 0))),  ↩
     POLYHEDRALSURFACE(((7 0 0,7 8 0,9 8 0,9 5 0,12 5 0,12 8 0,14 8 0,14 0 0,12 0 0,12 3 0,9 3 0,9 0 0,7 0 0)),((7 0  ↩
     7,9 0 7,9 3 7,12 3 7,12 0 7,14 0 7,14 8 7,12 8 7,12 5 7,9 5 7,9 8 7,7 8 7,7 0 7)),((7 0 0,7 0 7,7 8 7,7 8 0,7 0  ↩
     0)),((7 8 0,7 8 7,9 8 7,9 8 0,7 8 0)),((9 8 0,9 8 7,9 5 7,9 5 0,9 8 0)),((9 5 0,9 5 7,12 5 7,12 5 0,9 5 0)),((12  ↩
     5 0,12 5 7,12 8 7,12 8 0,12 5 0)),((12 8 0,12 8 7,14 8 7,14 8 0,12 8 0)),((14 8 0,14 8 7,14 0 7,14 0 0,14 8 0)) ↩
     ,((14 0 0,14 0 7,12 0 7,12 0 0,14 0 0)),((12 0 0,12 0 7,12 3 7,12 3 0,12 0 0)),((12 3 0,12 3 7,9 3 7,9 3 0,12 3  ↩
     0)),((9 3 0,9 3 7,9 0 7,9 0 0,9 3 0)),((9 0 0,9 0 7,7 0 7,7 0 0,9 0 0))))
convex_hull           | POLYHEDRALSURFACE(((14 8 7,14 0 0,14 8 0,14 8 7)),((14 0 7,7 8 7,7 0 7,14 0 7)),((0 8 3,7 8  ↩
     7,0 8 0,0 8 3)),((0 0 3,7 8 7,0 8 3,0 0 3)),((0 0 3,0 8 3,0 0 0,0 0 3)),((7 0 7,7 8 7,0 0 3,7 0 7)),((14 0 7,14 0 ↩
      0,14 8 7,14 0 7)),((7 0 7,14 0 0,14 0 7,7 0 7)),((14 8 7,7 8 7,14 0 7,14 8 7)),((0 0 0,0 8 3,0 8 0,0 0 0)),((0 0 ↩
      0,14 0 0,0 0 3,0 0 0)),((0 0 3,14 0 0,7 0 7,0 0 3)),((0 8 0,14 0 0,0 0 0,0 8 0)),((14 8 0,14 0 0,0 8 0,14 8 0)) ↩
     ,((14 8 0,7 8 7,14 8 7,14 8 0)),((0 8 0,7 8 7,14 8 0,0 8 0)))
```

Figure



Se även

ST_Letters, ST_AsX3D

8.3.9 CG_3DIntersection

CG_3DIntersection — Utför 3D-intersektion

Synopsis

geometry **CG_3DIntersection**(geometry geom1, geometry geom2);

Beskrivning

Returnerar en geometri som är den delade delen mellan geom1 och geom2.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

3D linestrings and polygons can intersect in a 3D segment.

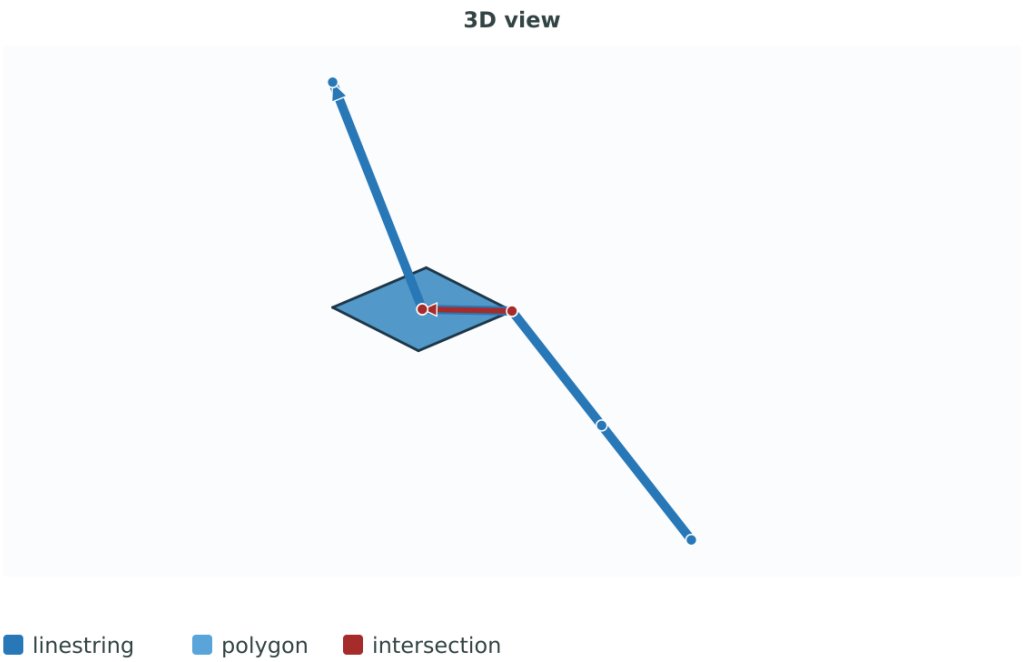
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)'::geometry AS  ↩
    input_linestring,
         'POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8))'::geometry AS input_polygon
)
SELECT input_linestring AS input_linestring,
       input_polygon AS input_polygon,
       CG_3DIntersection(input_linestring, input_polygon) AS intersection
FROM data;
```

Output

```
LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10) | POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8)) | LINESTRING Z (1  ↩
    1 8,0.5 0.5 8)
```

Figure



Intersect a cube shell with a vertical polygon.

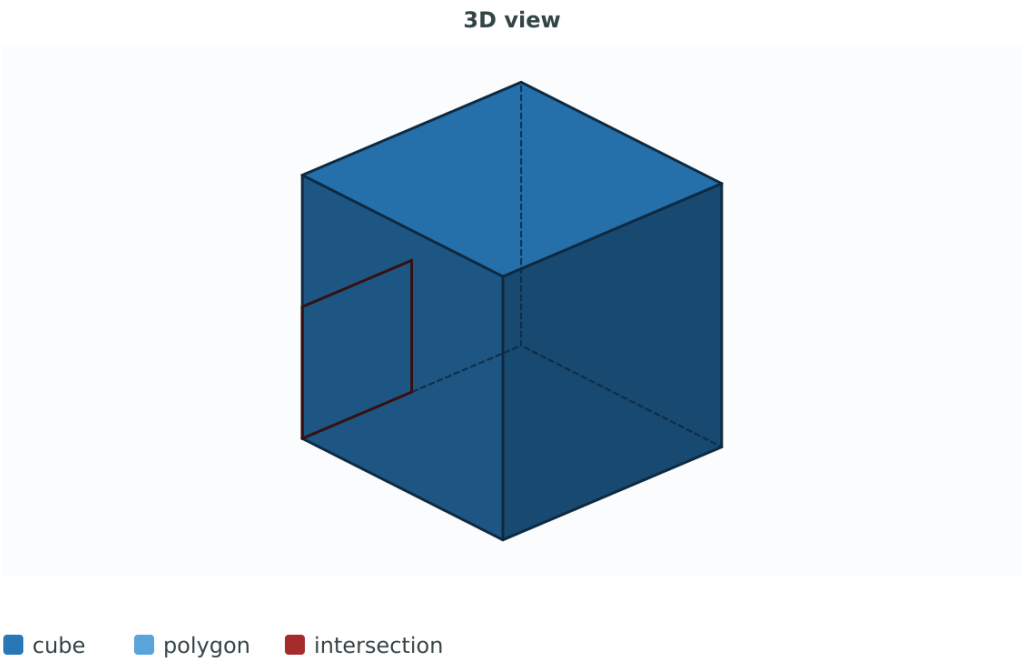
Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z(
    ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
    ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
    ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
  )'::geometry AS input_cube,
  'POLYGON Z ((0 0 0,0 0 0.5,0 0.5 0.5,0 0.5 0,0 0 0))'::geometry AS input_polygon
)
SELECT input_cube AS input_cube,
       input_polygon AS input_polygon,
       CG_3DIntersection(input_cube, input_polygon) AS intersection
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 ↩
     0)),((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))) |  ↩
    POLYGON Z ((0 0 0,0 0 0.5,0 0.5 0.5,0 0.5 0,0 0 0)) | TIN Z (((0 0 0,0 0 0.5,0 0.5 0.5,0 0 0)),((0 0.5 0,0 0 0,0  ↩
    0.5 0.5,0 0.5 0)))
```

Figure



A smaller solid contained by a larger solid remains volumetric after intersection.

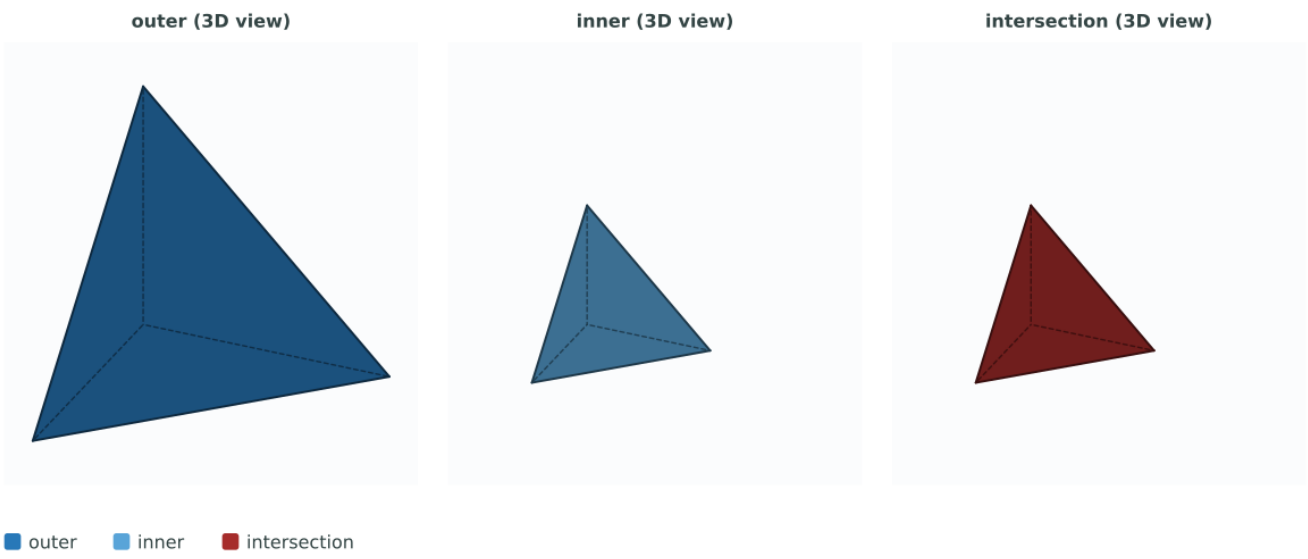
Code

```
WITH data AS (
  SELECT CG_MakeSolid('POLYHEDRALSURFACE Z (
    ((0 0 0,0 4 0,4 0 0,0 0 0)),
    ((0 0 0,4 0 0,0 0 4,0 0 0)),
    ((0 0 0,0 0 4,0 4 0,0 0 0)),
    ((4 0 0,0 4 0,0 0 4,4 0 0))
  )'::geometry) AS outer_solid
), solids AS (
  SELECT outer_solid,
         ST_Scale(outer_solid, 0.5, 0.5, 0.5) AS inner_solid
  FROM data
)
SELECT outer_solid AS input_outer,
       inner_solid AS input_inner,
       CG_3DIntersection(outer_solid, inner_solid) AS intersection
FROM solids;
```

Output

```
POLYHEDRALSURFACE Z (((0 0 0,0 4 0,4 0 0,0 0 0)),((0 0 0,4 0 0,0 0 4,0 0 0)),((0 0 0,0 0 4,0 4 0,0 0 0)),((4 0 0,0 4 ↩
    0,0 0 4,4 0 0))) | POLYHEDRALSURFACE Z (((0 0 0,0 2 0,2 0 0,0 0 0)),((0 0 0,2 0 0,0 0 2,0 0 0)),((0 0 0,0 0 2,0 2 ↩
     0,0 0 0)),((2 0 0,0 2 0,0 0 2,2 0 0))) | POLYHEDRALSURFACE Z (((0 0 0,0 2 0,2 0 0,0 0 0)),((0 0 0,0 0 2,0 2 0,0 ↩
    0 0)),((2 0 0,0 0 2,0 0 0,2 0 0)),((2 0 0,0 2 0,0 0 2,2 0 0)))
```

Figure



8.3.10 CG_Union

CG_Union — Beräknar föreningen av två geometrier

Synopsis

geometry **CG_Union**(geometry geomA , geometry geomB);

Beskrivning

Beräknar föreningen av två geometrier.
Utförs av SFCGAL-modulen

> **Notera**
> OBS: Denna funktion returnerar en geometri som representerar unionen.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Exempel på geometri

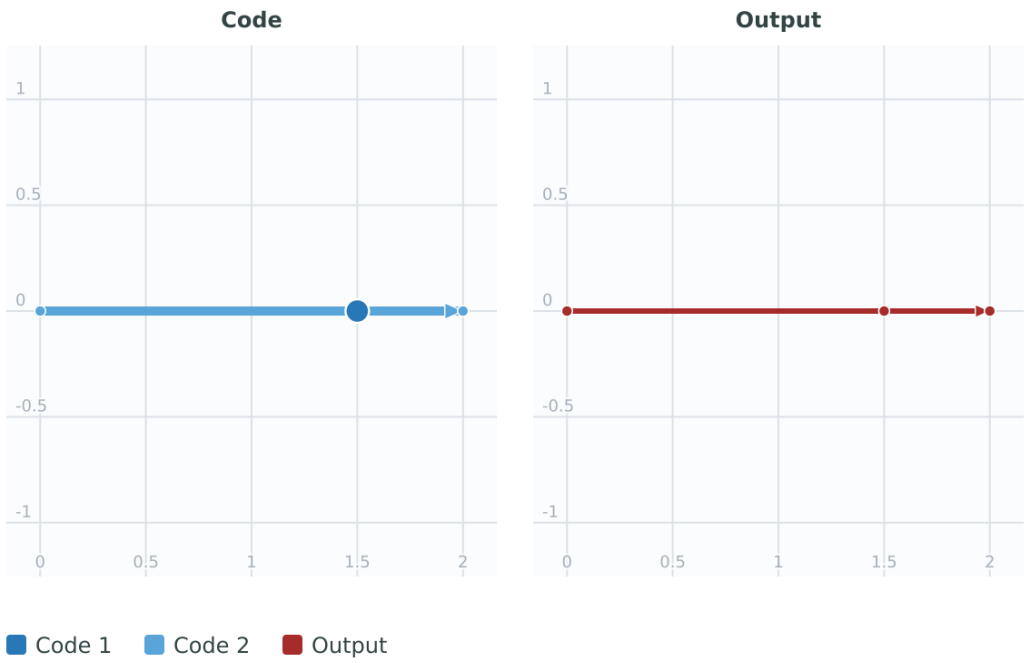
Code

```
SELECT CG_Union('POINT(.5 0)', 'LINESTRING(-1 0,1 0)');
```

Output

```
LINESTRING(-1 0,0.5 0,1 0)
```

Figure



Se även

CG_3DUnion, ST_Union

8.3.11 CG_3DUnion

CG_3DUnion — Utför 3D-union med hjälp av postgis_sfcgal.

Synopsis

geometry **CG_3DUnion**(geometry geom1, geometry geom2);
geometry **CG_3DUnion**(geometry set g1field);

Beskrivning

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna metod implementerar SQL/MM-specifikationen. SQL-MM IEC 13249-3: 5.1

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Aggregerad variant: returnerar en geometri som är 3D-unionen av en raduppsättning geometrier. CG_3DUnion () -funktionen är en äggregeradfunktion i terminologin för PostgreSQL. Det betyder att den fungerar på rader med data, på samma sätt som SUM () och AVG () -funktionerna gör och som de flesta aggregat ignorerar den också NULL-geometrier.

Exempel

Union two overlapping tetrahedral solids.

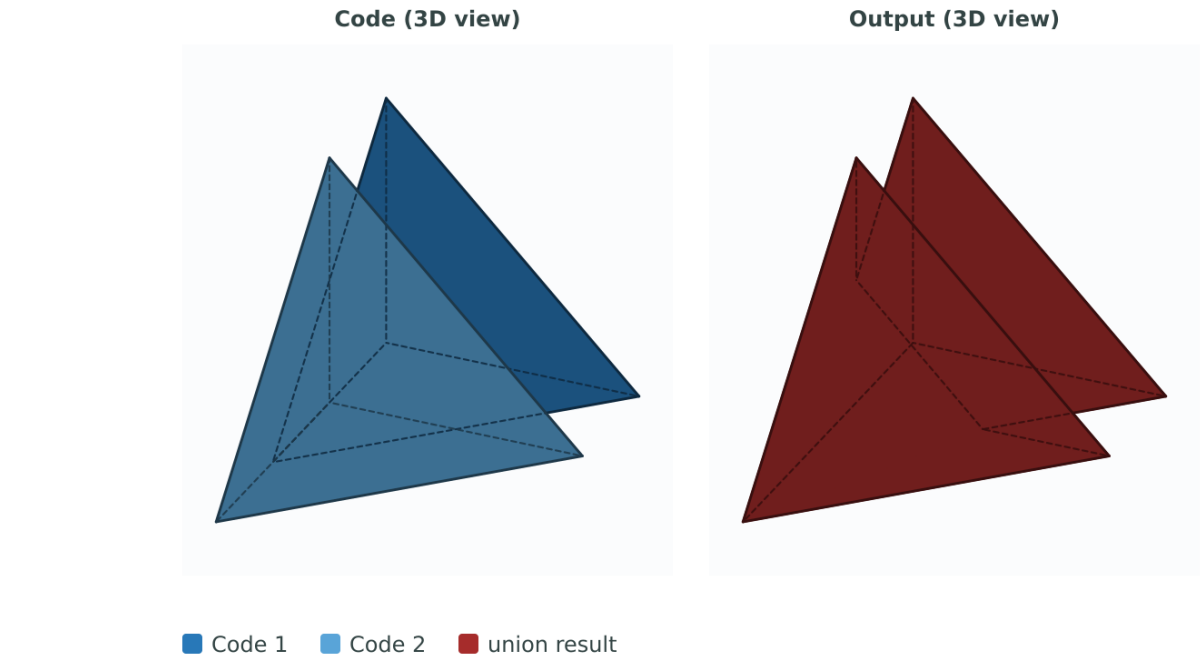
Code

```
WITH solids AS (
  SELECT CG_MakeSolid('POLYHEDRALSURFACE Z (
            ((0 0 0,4 0 0,0 4 0,0 0 0)),
            ((0 0 0,0 0 4,4 0 0,0 0 0)),
            ((0 0 0,0 4 0,0 0 4,0 0 0)),
            ((4 0 0,0 0 4,0 4 0,4 0 0)))'::geometry) AS geom1,
         CG_MakeSolid('POLYHEDRALSURFACE Z (
            ((2 0 0,6 0 0,2 4 0,2 0 0)),
            ((2 0 0,2 0 4,6 0 0,2 0 0)),
            ((2 0 0,2 4 0,2 0 4,2 0 0)),
            ((6 0 0,2 0 4,2 4 0,6 0 0)))'::geometry) AS geom2
)
SELECT CG_3DUnion(geom1, geom2) AS union_result
FROM solids;
```

Output

```
POLYHEDRALSURFACE Z (((2 0 0,0 0 0,2 2 0,2 0 0)),((2 0 2,0 0 0,2 0 0,2 0 2)),((0 0 4,0 4 0,0 0 0,0 0 4)),((2 0 2,0 4 ↩
    0,0 0 4,2 0 2)),((0 4 0,2 2 0,0 0 0,0 4 0)),((2 2 0,0 4 0,2 0 2,2 2 0)),((0 0 0,2 0 2,0 0 4,0 0 0)),((4 0 0,2 0 ↩
    0,2 2 0,4 0 0)),((4 0 0,2 0 2,2 0 0,4 0 0)),((4 0 0,2 2 0,6 0 0,4 0 0)),((2 0 2,4 0 0,2 0 4,2 0 2)),((2 0 2,2 4 ↩
    0,2 2 0,2 0 2)),((2 4 0,2 0 4,6 0 0,2 4 0)),((2 2 0,2 4 0,6 0 0,2 2 0)),((2 4 0,2 0 2,2 0 4,2 4 0)),((4 0 0,6 0 ↩
    0,2 0 4,4 0 0)))
```

Figure



Se även

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DDifference

8.3.12 CG_AlphaShape

CG_AlphaShape — Beräknar en Alpha-form som omsluter en geometri

Synopsis

geometry **CG_AlphaShape**(geometry geom, float alpha, boolean allow_holes = false);

Beskrivning

Beräknar Alpha-Shape för punkterna i en geometri. En alfa-form är en (vanligtvis) konkav polygonal geometri som innehåller alla indatavärdenas hörnpunkter och vars hörnpunkter är en delmängd av indatavärdenas hörnpunkter. En alfa-form ger en bättre anpassning till indatans form än den form som produceras av det konvexa skrovet.

Passningsgradenstyrs av `alfaparametern`, som kan ha värden från 0 till oändligt. Mindre alfavärden ger mer konkava resultat. Alphavärden som är större än ett visst databeroende värde ger ett konvext skrov av indata.

Notera
Enligt CGAL-implementeringen är alfavärdet *kvadraten* på radien på den skiva som används i Alpha-Shape-algoritmen för att ”eroderaDelaunay-trianguleringen av inmatningspunkterna. Se CGAL Alpha-Shapes för mer information. Detta skiljer sig från den ursprungliga definitionen av alpha-former, som definierar alfa som radien på den eroderande skivan.

Den beräknade formen innehåller inga hål om inte det valfria argumentet `allow_holes` anges som true.

Denna funktion beräknar effektivt ett konkavt skrov av en geometri på ett liknande sätt som ST_ConcaveHull men använder CGAL och en annan algoritm.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.4.1.

Denna metod behöver SFCGAL-backend.

Exempel

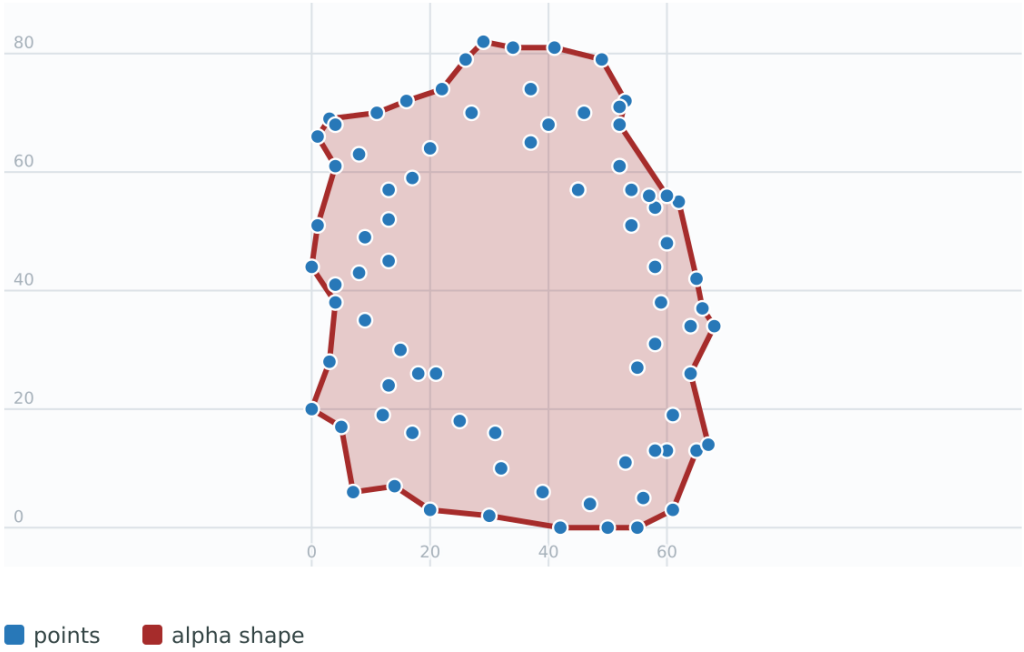
The original point cloud and the alpha-shape share one comparison frame.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),
(88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 ↩
    65),
(81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
    98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
    88),(60 81),
```

```
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
    90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
    59),(34 86),(45 90),(64 97))'::geometry AS input_points
)
SELECT input_points AS input_points,

        CG_AlphaShape(input_points, 80.2) AS alpha_shape
FROM data;
```

Figure

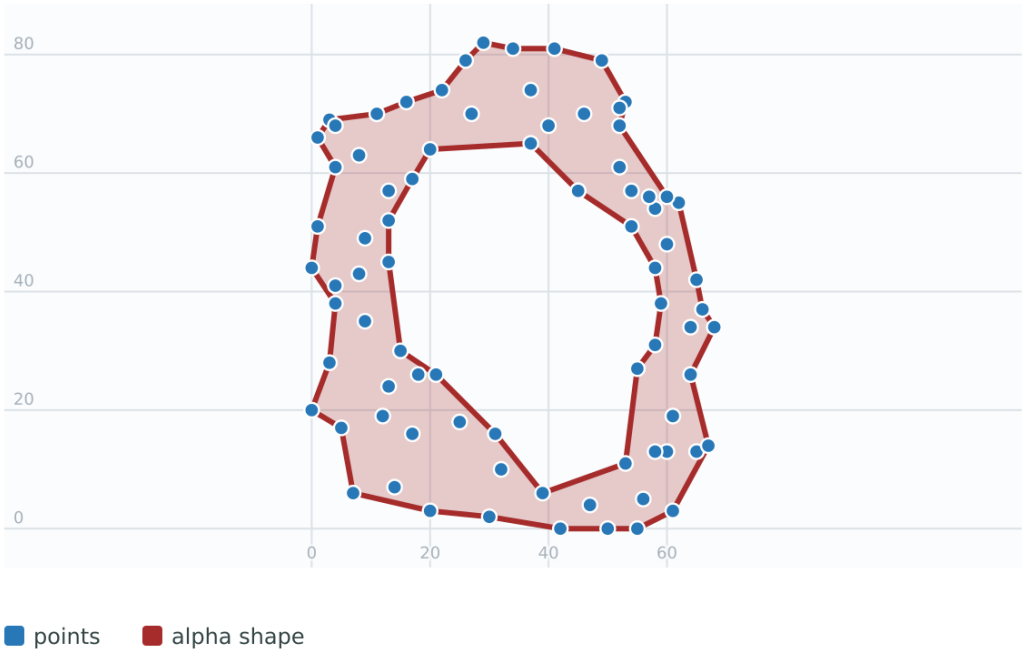


The same point cloud can produce a shape with a hole when holes are allowed.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),(88 29),(24 82),(32  ↩
    51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 65),(81 47),(88  ↩
    58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
    98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
    88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
    90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
    59),(34 86),(45 90),(64 97))'::geometry AS input_points
)
SELECT input_points AS input_points,

        CG_AlphaShape(input_points, 100.1, true) AS alpha_shape
FROM data;
```

Figure

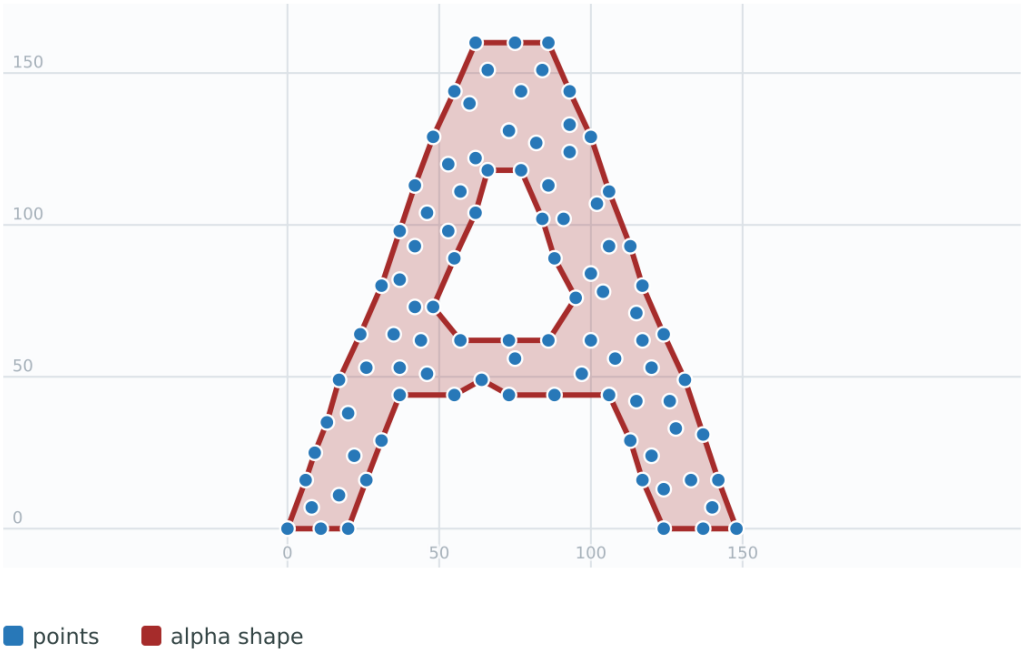


The second documented point cloud also produces a holed alpha-shape.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT ((132 64),(114 64),(99 64),(81 64),(63 64),(57 49),(52 36),(46 20),(37 ↩
     20),(26 20),(32 36),(39 55),(43 69),(50 84),(57 100),(63 118),(68 133),(74 149),(81  ↩
    164),(88 180),(101 180),(112 180),(119 164),(126 149),(132 131),(139 113),(143 100),(150 ↩
     84),(157 69),(163 51),(168 36),(174 20),(163 20),(150 20),(143 36),(139 49),(132 64) ↩
    ,(99 151),(92 138),(88 124),(81 109),(74 93),(70 82),(83 82),(99 82),(112 82),(126 82) ↩
    ,(121 96),(114 109),(110 122),(103 138),(99 151),(34 27),(43 31),(48 44),(46 58),(52 73) ↩
    ,(63 73),(61 84),(72 71),(90 69),(101 76),(123 71),(141 62),(166 27),(150 33),(159 36) ↩
    ,(146 44),(154 53),(152 62),(146 73),(134 76),(143 82),(141 91),(130 98),(126 104),(132  ↩
    113),(128 127),(117 122),(112 133),(119 144),(108 147),(119 153),(110 171),(103 164),(92 ↩
     171),(86 160),(88 142),(79 140),(72 124),(83 131),(79 118),(68 113),(63 102),(68 93) ↩
    ,(35 45))'::geometry AS input_points
)
SELECT input_points AS input_points,

         CG_AlphaShape(input_points, 102.2, true) AS alpha_shape
FROM data;
```

Figure



Se även

ST_ConcaveHull, CG_OptimalAlphaShape

8.3.13 CG_ApproxConvexPartition

CG_ApproxConvexPartition — Beräknar approximal konvex partition av polygongeometrin

Synopsis

geometry **CG_ApproxConvexPartition**(geometry geom);

Beskrivning

Beräknar approximal konvex partition av polygongeometrin (med hjälp av en triangulering).

Notera
En partition av en polygon P är en uppsättning polygoner som är sådana att polygonernas inre inte skär varandra och att polygonernas förening är lika med den ursprungliga polygonen P:s inre. Funktionerna CG_ApproxConvexPartition och CG_GreeneApproxConvexPartition producerar ungefärligt optimala konvexa partitioner. Båda dessa funktioner producerar konvexa uppdelningar genom att först dela upp polygonen i enklare polygoner; CG_ApproxConvexPartition använder en triangulering och CG_GreeneApproxConvexPartition en monoton uppdelning. Dessa två funktioner garanterar båda att de inte kommer att producera mer än fyra gånger det optimala antalet konvexa bitar, men de skiljer sig åt i deras körtidskomplexitet. Även om den trianguleringsbaserade approximationsalgoritmen ofta resulterar i färre konvexa bitar är detta inte alltid fallet.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0.

Kräver SFCGAL >= 1.5.0

Denna metod behöver SFCGAL-backend.

Exempel

Approximate convex partition of the same polygon used by the other partition examples.

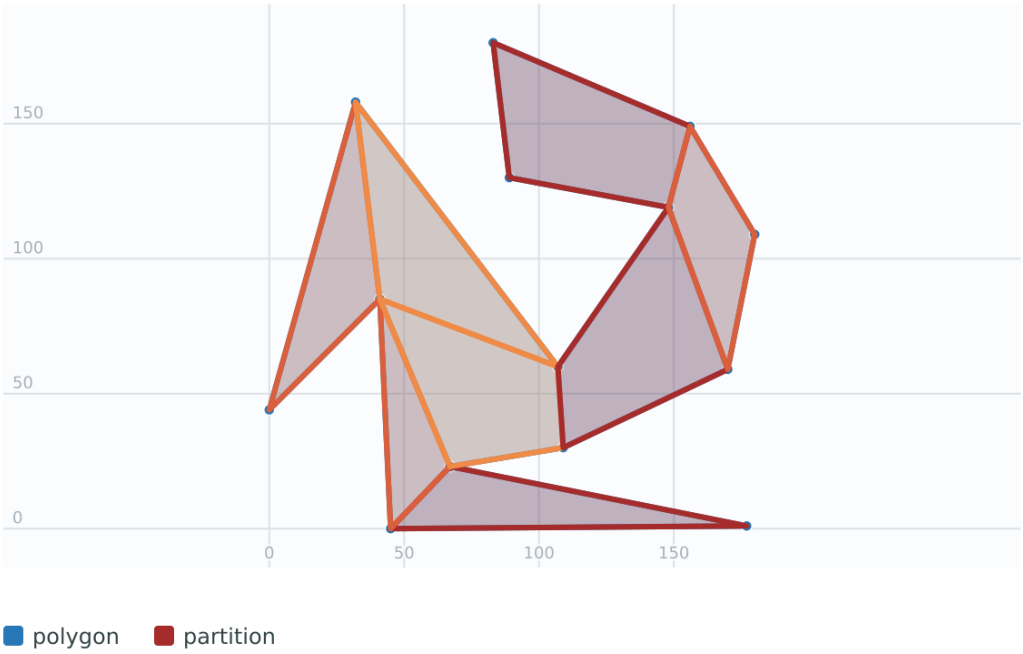
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67  ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_ApproxConvexPartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) |  ↩
    GEOMETRYCOLLECTION(POLYGON((156 150,83 181,89 131,148 120,156 150)), POLYGON((32 159,0 45,41 86,32 159)), POLYGON ↩
    ((107 61,32 159,41 86,107 61)), POLYGON((45 1,177 2,67 24,45 1)), POLYGON((41 86,45 1,67 24,41 86)), POLYGON((107 ↩
     61,41 86,67 24,109 31,107 61)), POLYGON((148 120,107 61,109 31,170 60,148 120)), POLYGON((156 150,148 120,170  ↩
    60,180 110,156 150)))
```

Figure



Se även

CG_YMonotonePartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

8.3.14 CG_ApproximateMedialAxis

CG_ApproximateMedialAxis — Beräkna den ungefärliga mediala axeln för en arealgeometri.

Synopsis

geometry **CG_ApproximateMedialAxis**(geometry geom);
geometry **CG_ApproximateMedialAxis**(geometry geom, boolean projected);

Beskrivning

Returnerar en ungefärlig medial axel för den areella inmatningen baserat på dess raka skelett. Använder ett SFCGAL-specifikt API när det byggs mot en kapabel version (1.2.0+). Annars är funktionen bara ett omslag runt CG_StraightSkeleton (långsammare fall).

When `projected` is `true`, free endpoints of the medial axis are extended to reach the polygon boundary (projected medial axis). Requires SFCGAL 2.3.0+. When built against an older SFCGAL version, a notice is emitted and the non-projected result is returned instead.

Tillgänglighet: 3.5.0

Availability: 3.7.0 - `projected` parameter. Requires SFCGAL >= 2.3.0 for projected result; falls back to non-projected with a notice on older versions.

Notera
Denna funktion ignorerar Z-dimensionen. Den ger alltid ett 2D-resultat även när den används på en 3D-geometri.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

A polygon and its approximate medial axis.

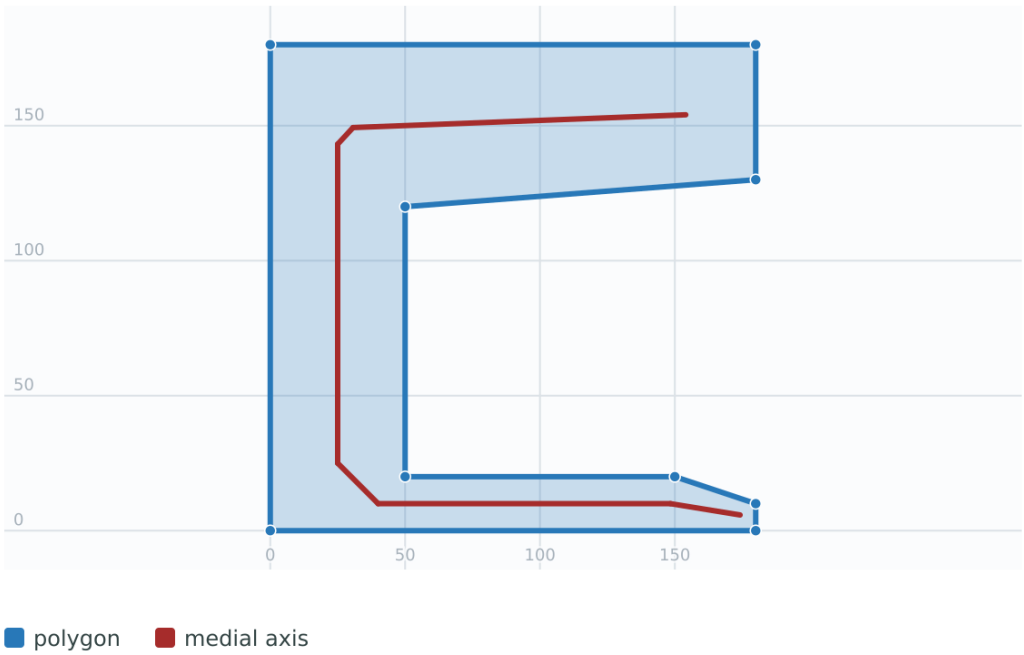
Code

```
WITH data AS (
  SELECT 'POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190)) ↩
    '::geometry AS polygon
)
SELECT polygon AS input_polygon,
       CG_ApproximateMedialAxis(polygon) AS medial_axis
FROM data;
```

Output

```
POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190)) | MULTILINESTRING((184.189 ↩
    15.811,158.377 20),(50 20,158.377 20),(50 20,35 35),(35 153.151,35 35),(35 153.151,40.697 159.303),(164.04 ↩
    164.04,40.697 159.303))
```

Figure



This example computes the projected medial axis with free endpoints extended to the polygon boundary.

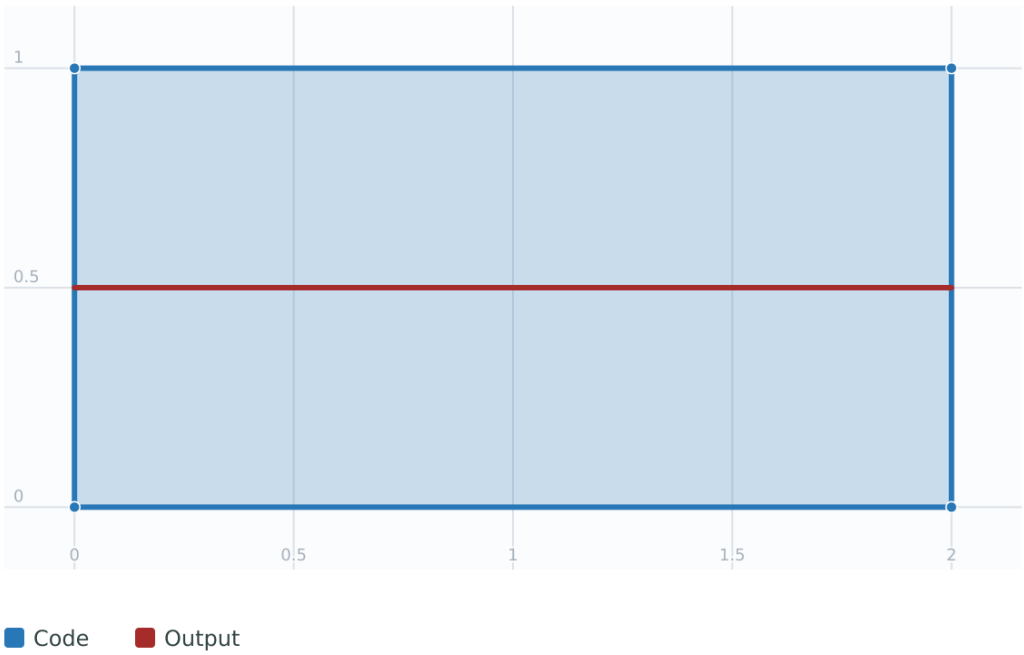
Code

```
SELECT CG_ApproximateMedialAxis('POLYGON((0 0,2 0,2 1,0 1,0 0))', true);
```

Output

```
MULTILINESTRING((0 0.5,0.5 0.5,1.5 0.5,2 0.5))
```

Figure



Se även

CG_StraightSkeleton, CG_StraightSkeletonPartition

8.3.15 CG_GenerateFlatRoof

CG_GenerateFlatRoof — Generate a flat (box) roof from a footprint polygon.

Synopsis

geometry **CG_GenerateFlatRoof**(geometry geom, float8 height);

Beskrivning

Generate a flat (box-shaped) roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. The roof height above the ground plane is given by `height` (default 3.0).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

Exempel

Generate a flat roof two units above a rectangular footprint.

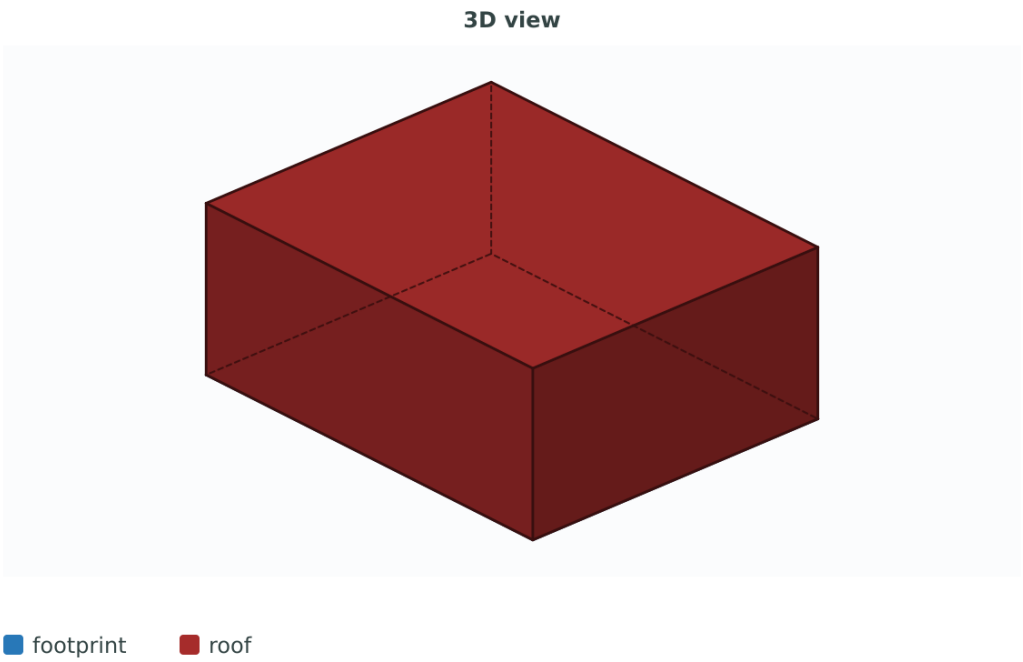
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS footprint
)
SELECT CG_GenerateFlatRoof(footprint, 2.0) AS roof
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((0 0 0,0 4 0,5 4 0,5 0 0,0 0 0)),((0 0 2,5 0 2,5 4 2,0 4 2,0 0 2)),((0 0 0,0 0 2,0 4 2,0 4 0,0 0 ↩
     0)),((0 4 0,0 4 2,5 4 2,5 4 0,0 4 0)),((5 4 0,5 4 2,5 0 2,5 0 0,5 4 0)),((5 0 0,5 0 2,0 0 2,0 0 0,5 0 0)))
```

Figure



Se även

CG_GenerateHippedRoof, CG_GenerateGableRoof, CG_GenerateSkillionRoof, CG_GenerateRoof

8.3.16 CG_GenerateHippedRoof

CG_GenerateHippedRoof — Generate a hipped roof from a footprint polygon.

Synopsis

geometry **CG_GenerateHippedRoof**(geometry geom, float8 height);

Beskrivning

Generate a hipped roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. All sides slope upward to meet at a ridge or apex. The roof height is given by `height` (default 3.0).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

Exempel

The footprint and hipped roof are shown side-by-side so the generated 3D faces do not hide behind the input footprint.

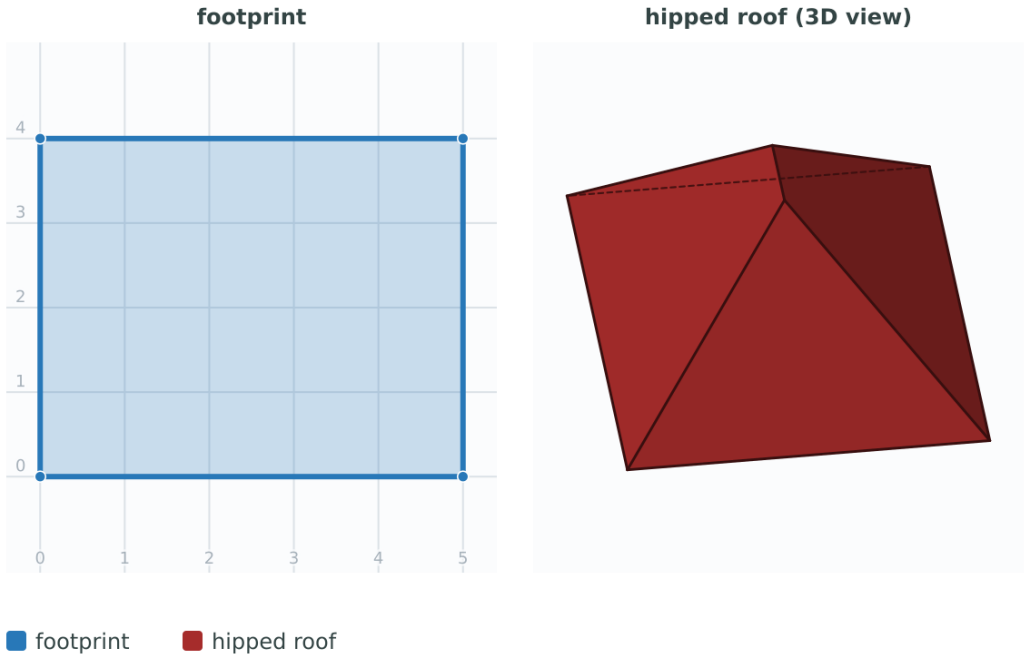
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS input_footprint
)
SELECT input_footprint AS input_footprint,
       CG_GenerateHippedRoof(input_footprint, 2.0) AS hipped_roof
FROM data;
```

Output

```
POLYGON((0 0,5 0,5 4,0 4,0 0)) | POLYHEDRALSURFACE Z (((0 4 0,5 4 0,5 0 0,0 0 0,0 4 0)),((0 4 0,0 0 0,2 2 2,0 4 0)) ↩
    ,((0 0 0,5 0 0,3 2 2,2 2 2,0 0 0)),((5 0 0,5 4 0,3 2 2,5 0 0)),((5 4 0,0 4 0,2 2 2,3 2 2,5 4 0)))
```

Figure



Se även

CG_GenerateFlatRoof, CG_GenerateGableRoof, CG_GenerateSkillionRoof, CG_GenerateRoof

8.3.17 CG_GenerateGableRoof

CG_GenerateGableRoof — Generate a gable roof from a footprint polygon.

Synopsis

geometry **CG_GenerateGableRoof**(geometry geom, float8 height, float8 slope_angle);

Beskrivning

Generate a gable roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. Two opposite sides are vertical gable ends; the other two sides slope to a horizontal ridge. `height` sets the ridge height

above the ground (default 3.0) and `slope_angle` controls the pitch in degrees (default 30.0). The SRID of the input geometry is preserved.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

Exempel

Generate a gable roof using a height of 2 and a 30 degree slope.

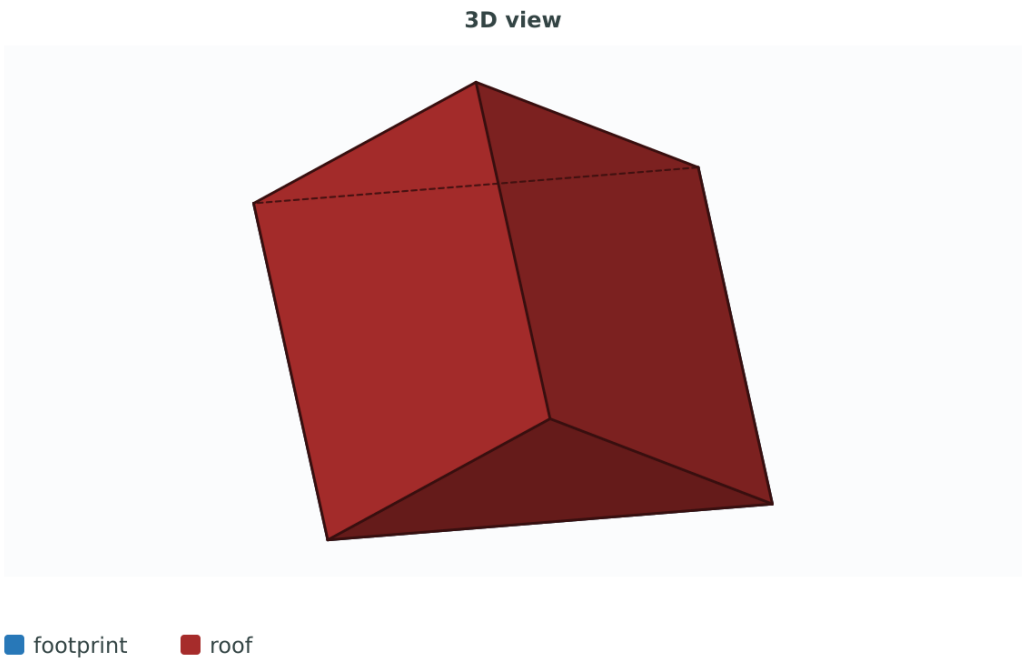
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS footprint
)
SELECT CG_GenerateGableRoof(footprint, 2.0, 30.0) AS roof
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((5 4 0,5 0 0,0 0 0,0 4 0,5 4 0)),((0 4 0,0 0 0,0 2 1.155,0 4 0)),((0 0 0,5 0 0,5 2 1.155,0 2 ↩
    1.155,0 0 0)),((5 0 0,5 4 0,5 2 1.155,5 0 0)),((5 4 0,0 4 0,0 2 1.155,5 2 1.155,5 4 0)))
```

Figure



Se även

CG_GenerateFlatRoof, CG_GenerateHippedRoof, CG_GenerateSkillionRoof, CG_GenerateRoof

8.3.18 CG_GenerateSkillionRoof

CG_GenerateSkillionRoof — Generate a skillion (single-slope) roof from a footprint polygon.

Synopsis

geometry **CG_GenerateSkillionRoof**(geometry geom, float8 height, float8 slope_angle, integer primary_edge_index);

Beskrivning

Generate a skillion (single-slope, shed-style) roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. The entire roof surface slopes in one direction. `height` sets the maximum ridge height (default 3.0), `slope_angle` sets the pitch in degrees (default 30.0), and `primary_edge_index` selects which polygon edge defines the high side (0-based, default 0).

When the combination of `height` and `slope_angle` would cause a side wall to vanish (the slope reaches the far edge before the full height is achieved), that face is omitted from the result. This means the number of output patches can vary depending on the parameters — a lower height relative to the footprint depth produces more faces than a taller, steeper configuration.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

Exempel

Height 2.0 produces 6 faces (all walls present); height 4.0 produces 5 faces (one side wall is filtered out because the slope reaches the far edge before the full height).

Code

```
SELECT
  ST_NumPatches(CG_GenerateSkillionRoof(
    'POLYGON((0 0,5 0,5 4,0 4,0 0))', 2.0)) AS "height 2",
  ST_NumPatches(CG_GenerateSkillionRoof(
    'POLYGON((0 0,5 0,5 4,0 4,0 0))', 4.0)) AS "height 4";
```

Output

```
6 | 5
```

The height 2.0 result is shown with its footprint.

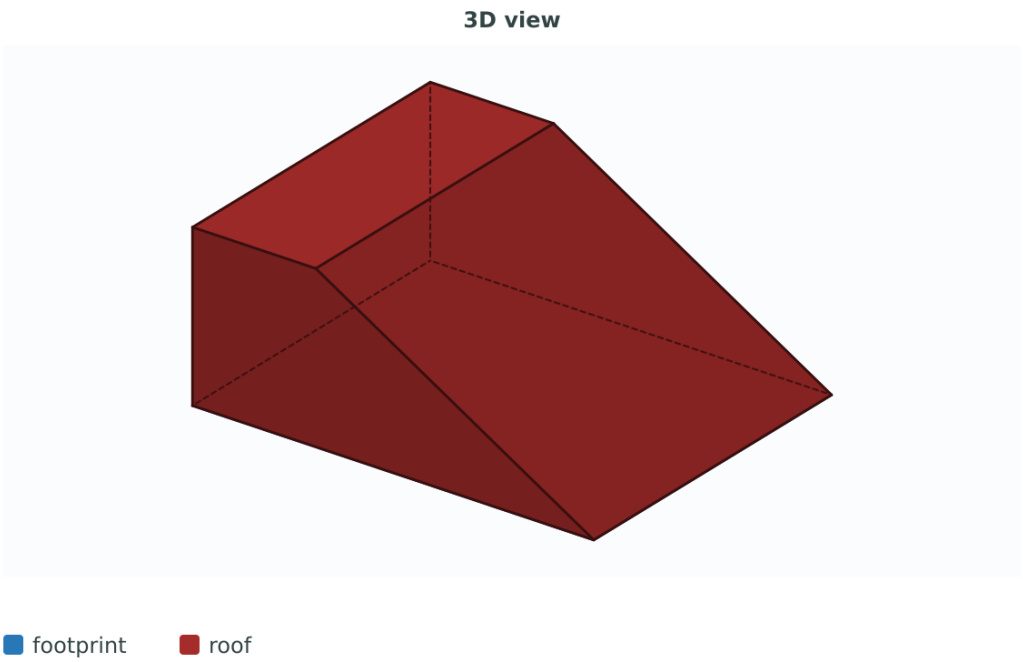
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS footprint
)
SELECT CG_GenerateSkillionRoof(footprint, 2.0) AS roof
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((0 4 0,5 4 0,5 0 0,0 0 0,0 4 0)),((5 0 2,5 4 2,3.464 4 2,3.464 0 2,5 0 2)),((0 4 0,0 0 0,3.464 0 ↩
     2,3.464 4 2,0 4 0)),((0 0 0,5 0 0,5 0 2,3.464 0 2,0 0 0)),((5 0 0,5 4 0,5 4 2,5 0 2,5 0 0)),((5 4 0,0 4 0,3.464  ↩
    4 2,5 4 2,5 4 0)))
```

Figure



Se även

CG_GenerateFlatRoof, CG_GenerateHippedRoof, CG_GenerateGableRoof, CG_GenerateRoof

8.3.19 CG_GenerateRoof

CG_GenerateRoof — Generate a roof of the requested type from a footprint polygon.

Synopsis

geometry **CG_GenerateRoof**(geometry geom, text roof_type, float8 height, float8 slope_angle, integer primary_edge_index);

Beskrivning

Generate a roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. The `roof_type` parameter selects the roof style (default `'HIPPED'`):

- `FLAT` — flat box roof.
- `HIPPED` — hipped roof (all sides slope to apex).
- `GABLE` — gable roof (two sloping sides, two vertical ends).
- `SKILLION` — single-slope (shed) roof.

`height` sets the roof height (default 3.0), `slope_angle` sets the pitch in degrees (default 30.0), and `primary_edge_index` selects the reference edge for skillion roofs (default 0).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

Exempel

The generic dispatcher returns the same 3D roofs as the specialized helpers for the same footprint.

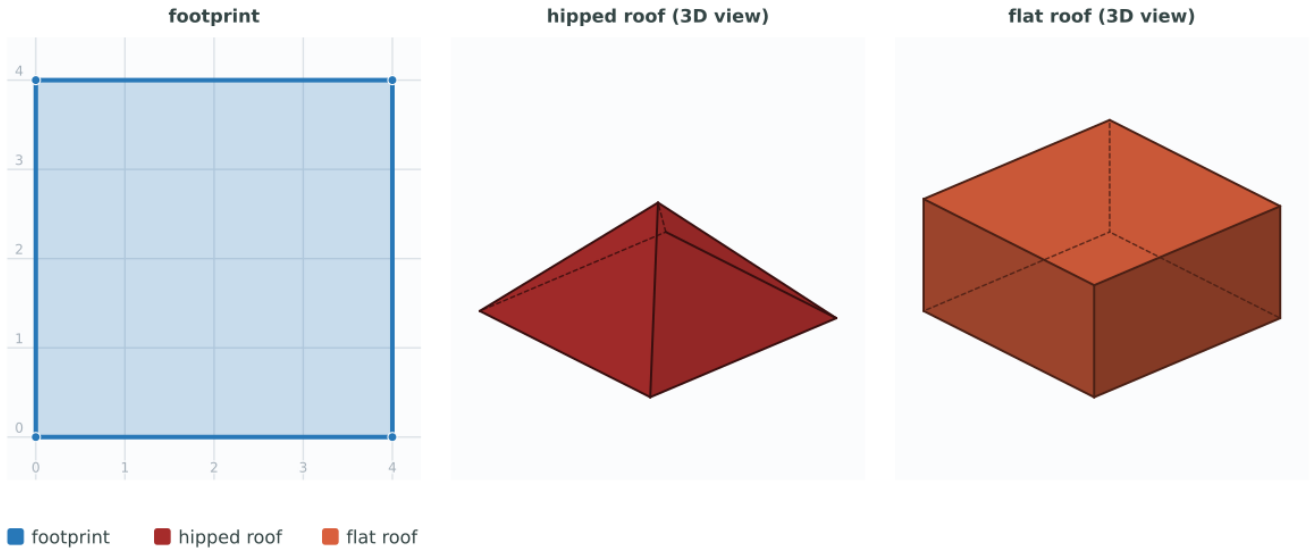
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,4 0,4 4,0 4,0 0))'::geometry AS input_footprint
)
SELECT input_footprint AS input_footprint,
       CG_GenerateRoof(input_footprint, 'HIPPED', 2.0, 30.0, 0) AS hipped_roof,
       CG_GenerateRoof(input_footprint, 'FLAT', 2.0, 30.0, 0) AS flat_roof
FROM data;
```

Output

```
POLYGON((0 0,4 0,4 4,0 4,0 0)) | POLYHEDRALSURFACE Z (((0 4 0,4 4 0,4 0 0,0 0 0,0 4 0)),((0 4 0,0 0 0,2 2 2,0 4 0)) ↩
    ,((0 0 0,4 0 0,2 2 2,0 0 0)),((4 0 0,4 4 0,2 2 2,4 0 0)),((4 4 0,0 4 0,2 2 2,4 4 0))) | POLYHEDRALSURFACE Z (((0  ↩
    0 0,0 4 0,4 4 0,4 0 0,0 0 0)),((0 0 2,4 0 2,4 4 2,0 4 2,0 0 2)),((0 0 0,0 0 2,0 4 2,0 4 0,0 0 0)),((0 4 0,0 4 2,4 ↩
     4 2,4 4 0,0 4 0)),((4 4 0,4 4 2,4 0 2,4 0 0,4 4 0)),((4 0 0,4 0 2,0 0 2,0 0 0,4 0 0)))
```

Figure



Se även

CG_GenerateFlatRoof, CG_GenerateHippedRoof, CG_GenerateGableRoof, CG_GenerateSkillionRoof

8.3.20 CG_PolygonRepair

CG_PolygonRepair — Repair an invalid polygon or multipolygon.

Synopsis

geometry **CG_PolygonRepair**(geometry geom, text rule);

Beskrivning

Repair an invalid polygon or multipolygon using CGAL's 2D Polygon Repair algorithm. Returns a `MultiPolygon`.

The `rule` parameter controls the repair strategy:

- `EVEN_ODD` (default) — areas covered an odd number of times are kept. Available with CGAL 6.0+.
- `NON_ZERO` — areas with non-zero winding number are kept. Requires CGAL 6.1+.
- `UNION` — union of all input polygons. Requires CGAL 6.1+.
- `INTERSECTION` — intersection of all input polygons. Requires CGAL 6.1+.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0 and CGAL >= 6.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

Exempel

Repair a self-intersecting bowtie polygon with the default `EVEN_ODD` rule.

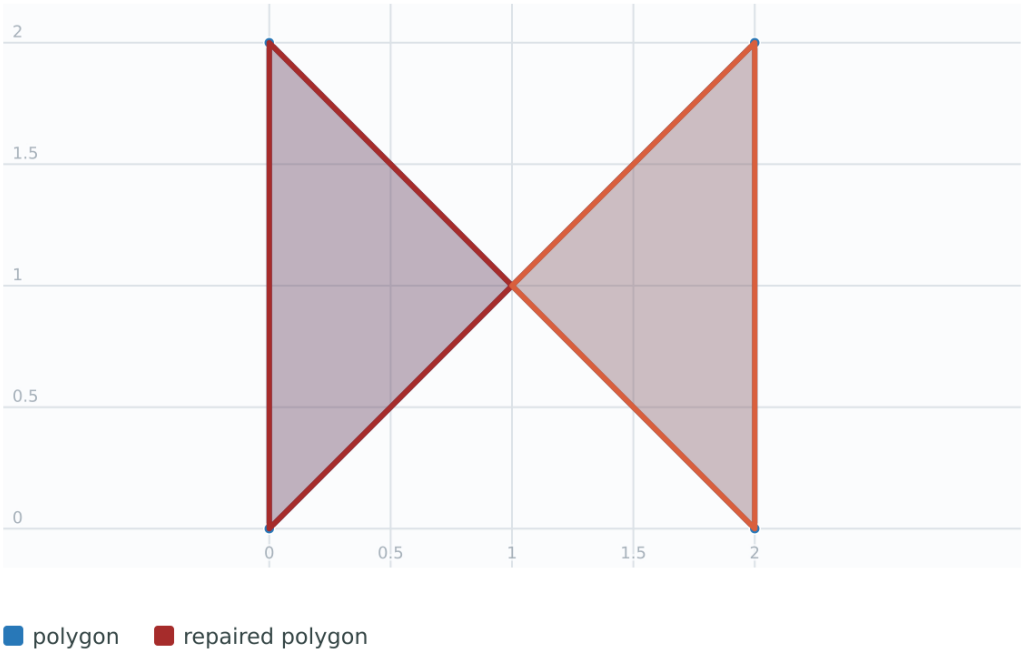
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,2 2,2 0,0 2,0 0))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_PolygonRepair(input_polygon) AS repaired_polygon
FROM data;
```

Output

```
POLYGON((0 0,2 2,2 0,0 2,0 0)) | MULTIPOLYGON(((0 0,1 1,0 2,0 0)),((1 1,2 0,2 2,1 1)))
```

Figure



With the `UNION` rule, all winding area is kept. This rule requires CGAL 6.1 or later.

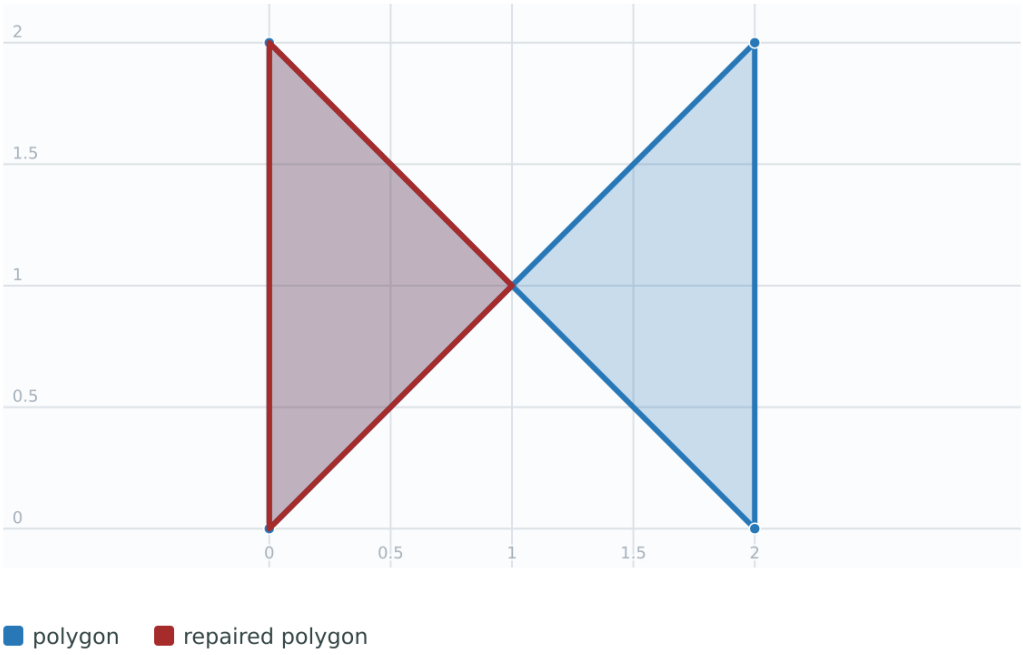
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,2 2,2 0,0 2,0 0))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_PolygonRepair(input_polygon, 'UNION') AS repaired_polygon
FROM data;
```

Output

```
POLYGON((0 0,2 2,2 0,0 2,0 0)) | MULTIPOLYGON(((0 0,1 1,0 2,0 0)))
```

Figure



Se även

ST_IsValid, ST_MakeValid

8.3.21 CG_ConstrainedDelaunayTriangles

CG_ConstrainedDelaunayTriangles — Returnerar en begränsad Delaunay-triangulering runt den givna indatageometrin.

Synopsis

geometry **CG_ConstrainedDelaunayTriangles**(geometry g1);

Beskrivning

Returnerar en begränsad Delaunay-triangulering runt topparna i indatageometrin. Utdata är en TIN.

Denna metod behöver SFCGAL-backend.

Tillgänglighet: 3.0.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

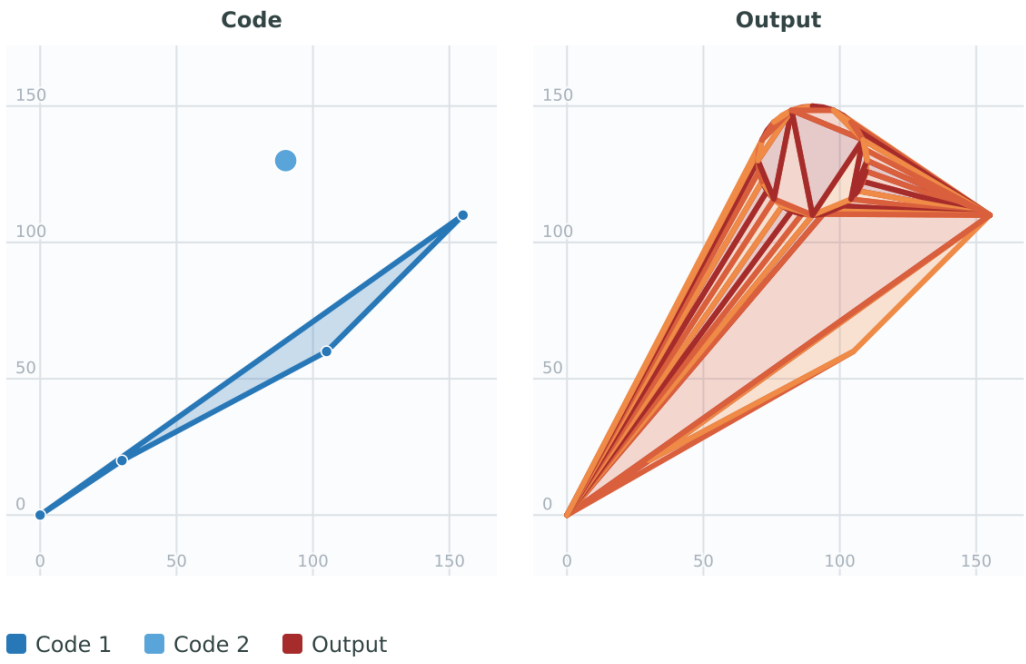
Exempel

Constrained Delaunay triangulation of the polygon and buffered point.

Code

```
SELECT
  CG_ConstrainedDelaunayTriangles(
    ST_Union(
      'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry,
      ST_Buffer('POINT(110 170)'::geometry, 20)
    )
  );
```

Figure

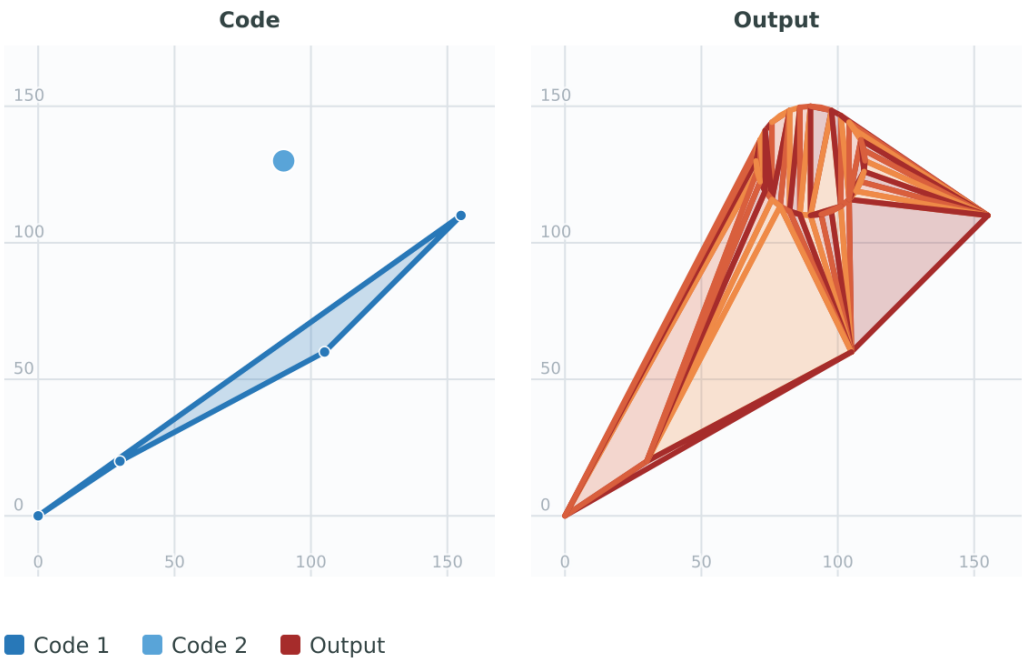


ST_DelaunayTriangles applied to the same input. Triangle edges cross polygon boundaries.

Code

```
SELECT
  ST_DelaunayTriangles(
    ST_Union(
      'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry,
      ST_Buffer('POINT(110 170)'::geometry, 20)
    )
  );
```

Figure



Se även

ST_DelaunayTriangles, ST_TriangulatePolygon, CG_Tesselate, ST_ConcaveHull, ST_Dump

8.3.22 CG_Extrude

CG_Extrude — Extrudera en yta till en relaterad volym

Synopsis

geometry **CG_Extrude**(geometry geom, float x, float y, float z);

Beskrivning

Extrudes the input along the vector (*x*, *y*, *z*) and connects it to its translated copy. Extrusion can raise the dimensionality of the result, for example from a surface to a volume.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

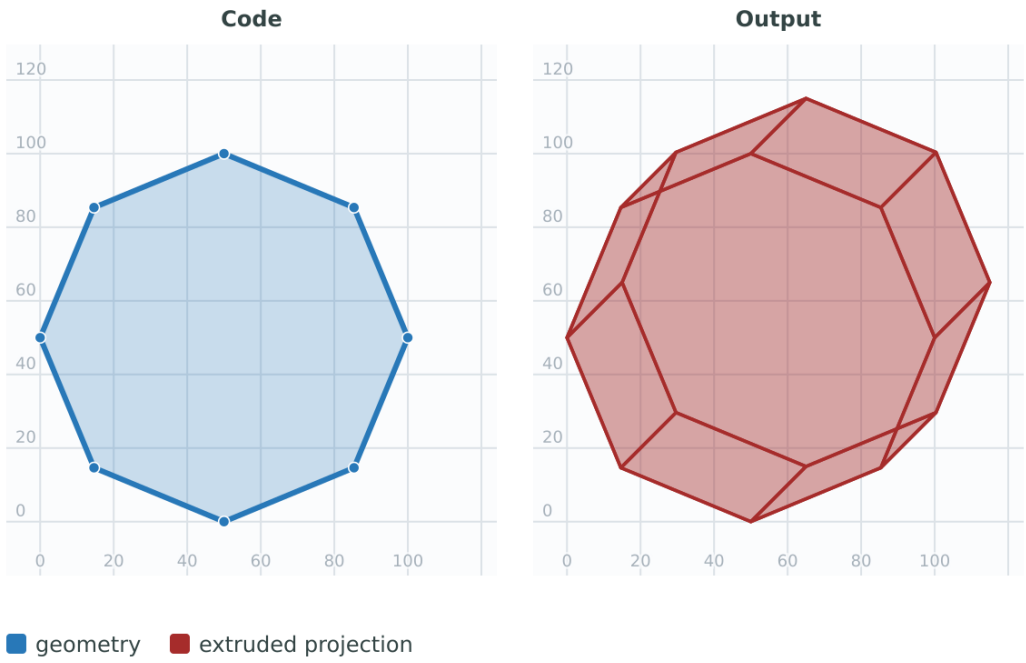
The figures use ST_Affine to project Z diagonally into XY after extrusion. The returned geometry from CG_Extrude remains a 3D PolyhedralSurface.

Extruding an octagonal Polygon 30 units along Z.

Code

```
WITH data AS (
  SELECT ST_Buffer('POINT(100 90)'::geometry, 50, 'quad_segs=2') AS geom
), projected AS (
  SELECT ST_Force2D(ST_Affine(geom,
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS input_geom,
         ST_Force2D(ST_Affine(CG_Extrude(geom, 0, 0, 30),
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS output_geom
  FROM data
)
SELECT input_geom AS input_geometry,
       output_geom AS extruded_projection
FROM projected;
```

Figure



Extruding a LineString 10 units along Z.

Code

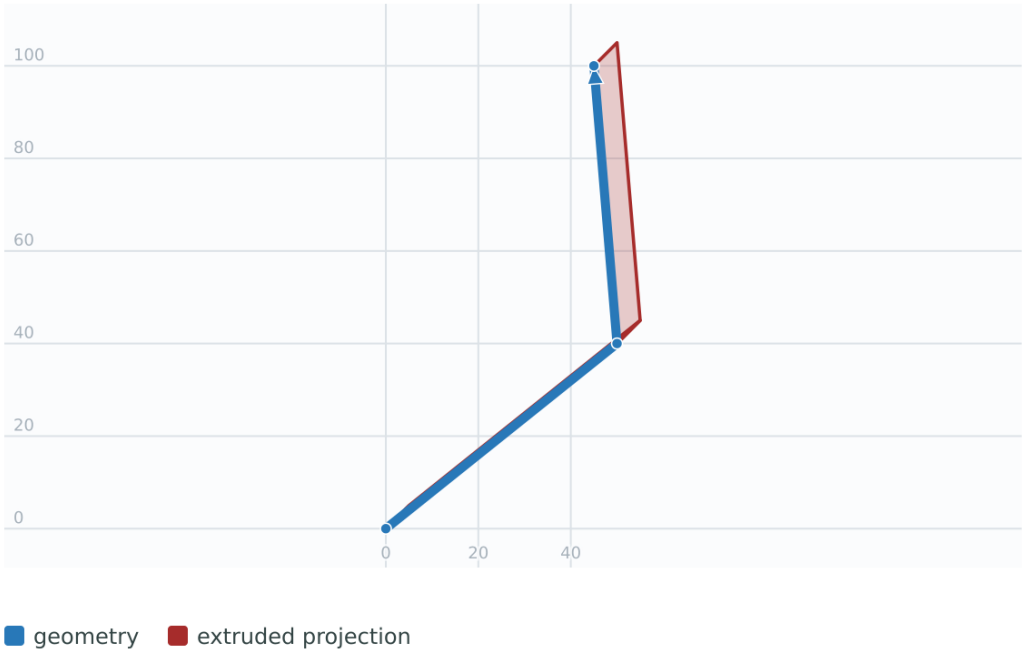
```
WITH data AS (
  SELECT 'LINESTRING(50 50,100 90,95 150)'::geometry AS geom
), projected AS (
  SELECT ST_Force2D(ST_Affine(geom,
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS input_geom,
         ST_Force2D(ST_Affine(CG_Extrude(geom, 0, 0, 10),
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS output_geom
  FROM data
)
SELECT input_geom AS input_geometry,
```

```
       output_geom AS extruded_projection
FROM projected;
```

Output

```
LINESTRING(50 50,100 90,95 150) | POLYHEDRALSURFACE(((50 50,100 90,105 95,55 55,50 50)),((100 90,95 150,100 155,105 ↩
    95,100 90)))
```

Figure



Se även

ST_AsX3D, CG_ExtrudeStraightSkeleton

8.3.23 CG_ExtrudeStraightSkeleton

CG_ExtrudeStraightSkeleton — Extrudering av raka skelett

Synopsis

geometry **CG_ExtrudeStraightSkeleton**(geometry geom, float roof_height, float body_height = 0);

Beskrivning

Beräknar en profil med en maximal höjd på polygongeometrin.

Notera
Kanske det första (historiskt sett) användningsområdet för raka skelett: givet ett polygonalt tak ger det raka skelettet direkt layouten för varje tält. Om varje skelettkant lyfts från planet med en höjd som är lika med dess offsetavstånd, är det resulterande taket ”korrekt”eftersom vatten alltid kommer att falla ner till konturkanterna (takets kant), oavsett var det faller på taket. Funktionen beräknar denna extrudering, även kallat tak”, på en polygon. Om argumentet body_height > 0, så extruderas polygonen på samma sätt som med CG_Extrude(polygon, 0, 0, body_height). Resultatet är en förening av dessa polyhedralsurfaces.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0.

Kräver SFCGAL >= 1.5.0

Denna metod behöver SFCGAL-backend.

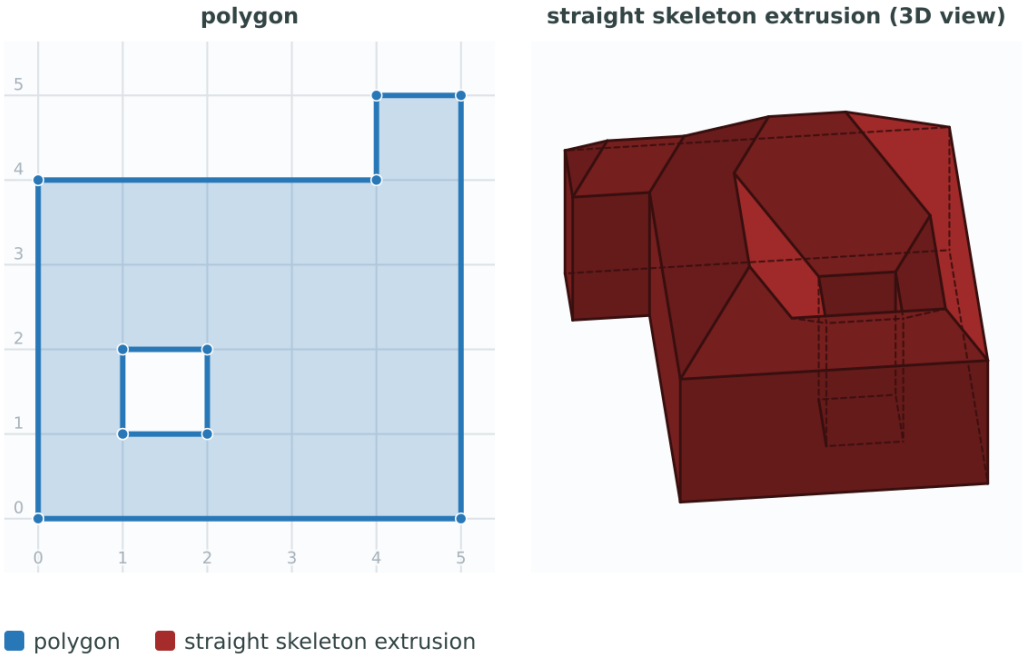
Exempel

Extrude a polygon with a hole into a body and straight-skeleton roof.

Code

```
WITH data AS (
  SELECT 'POLYGON ((0 0,5 0,5 5,4 5,4 4,0 4,0 0),(1 1,1 2,2 2,2 1,1 1))'::geometry AS ↩
    input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_ExtrudeStraightSkeleton(input_polygon, 3.0, 2.0) AS straight_skeleton_extrusion
FROM data;
```

Figure



Se även

CG_Extrude, CG_ExtrudeStraightSkeleton, CG_StraightSkeleton, CG_StraightSkeletonPartition

8.3.24 CG_GreeneApproxConvexPartition

CG_GreeneApproxConvexPartition — Beräknar approximal konvex partition av polygongeometrin

Synopsis

geometry **CG_GreeneApproxConvexPartition**(geometry geom);

Beskrivning

Beräknar approximal monoton konvex partition av polygongeometrin.

Notera
En partition av en polygon P är en uppsättning polygoner som är sådana att polygonernas inre inte skär varandra och att polygonernas förening är lika med den ursprungliga polygonen P:s inre. Funktionerna CG_ApproxConvexPartition och CG_GreeneApproxConvexPartition producerar ungefärligt optimala konvexa partitioner. Båda dessa funktioner producerar konvexa uppdelningar genom att först dela upp polygonen i enklare polygoner; CG_ApproxConvexPartition använder en triangulering och CG_GreeneApproxConvexPartition en monoton uppdelning. Dessa två funktioner garanterar båda att de inte kommer att producera mer än fyra gånger det optimala antalet konvexa bitar, men de skiljer sig åt i deras körtidskomplexitet. Även om den trianguleringsbaserade approximationsalgoritmen ofta resulterar i färre konvexa bitar är detta inte alltid fallet.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0.

Kräver SFCGAL >= 1.5.0

Denna metod behöver SFCGAL-backend.

Exempel

Greene’s monotone-based approximation uses the same polygon and stays directly comparable in one panel.

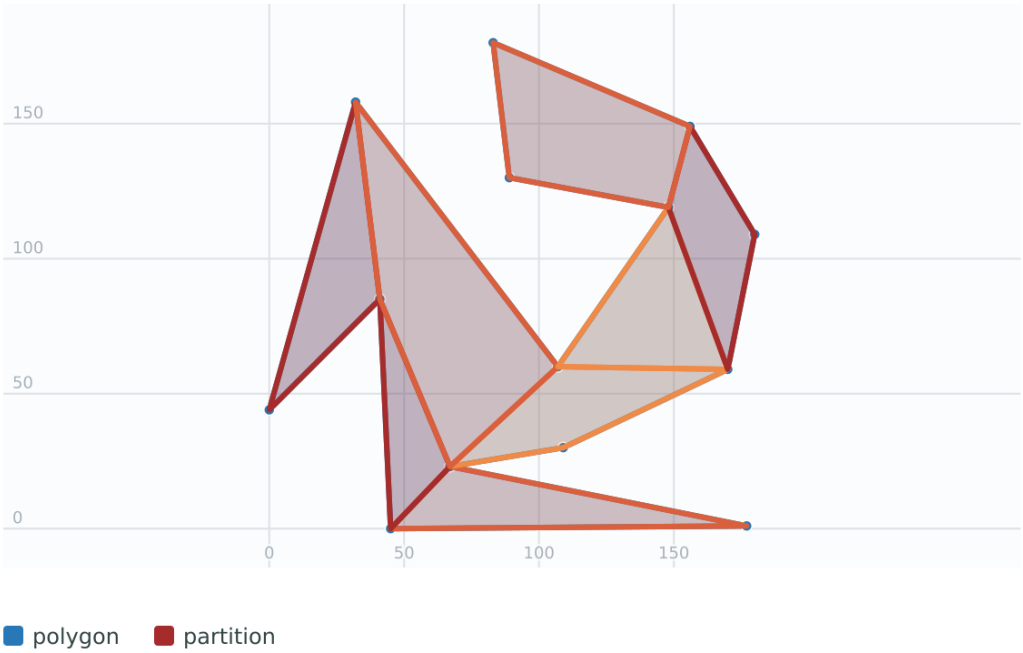
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_GreeneApproxConvexPartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) | ↩
    GEOMETRYCOLLECTION(POLYGON((32 159,0 45,41 86,32 159)), POLYGON((45 1,177 2,67 24,45 1)), POLYGON((67 24,109 ↩
    31,170 60,107 61,67 24)), POLYGON((41 86,45 1,67 24,41 86)), POLYGON((107 61,32 159,41 86,67 24,107 61)), POLYGON ↩
    ((148 120,107 61,170 60,148 120)), POLYGON((148 120,170 60,180 110,156 150,148 120)), POLYGON((156 150,83 181,89 ↩
    131,148 120,156 150)))
```

Figure



Se även

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_OptimalConvexPartition

8.3.25 CG_MinkowskiSum

CG_MinkowskiSum — Utför Minkowski-summa

Synopsis

geometry **CG_MinkowskiSum**(geometry geom1, geometry geom2);

Beskrivning

Denna funktion utför en 2D minkowski-summa av en punkt, linje eller polygon med en polygon.

En Minkowski-summa av två geometrier A och B är mängden av alla punkter som är summan av alla punkter i A och B. Minkowski-summor används ofta vid rörelseplanering och datorstödd design. Mer information på Wikipedia Minkowski addition.

Den första parametern kan vara en 2D-geometri (punkt, linestrings, polygon). Om en 3D-geometri skickas in kommer den att konverteras till 2D genom att tvinga Z till 0, vilket kan leda till ogiltiga fall. Den andra parametern måste vara en 2D-polygon.

Implementeringen använder CGAL 2D Minkowskisum.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

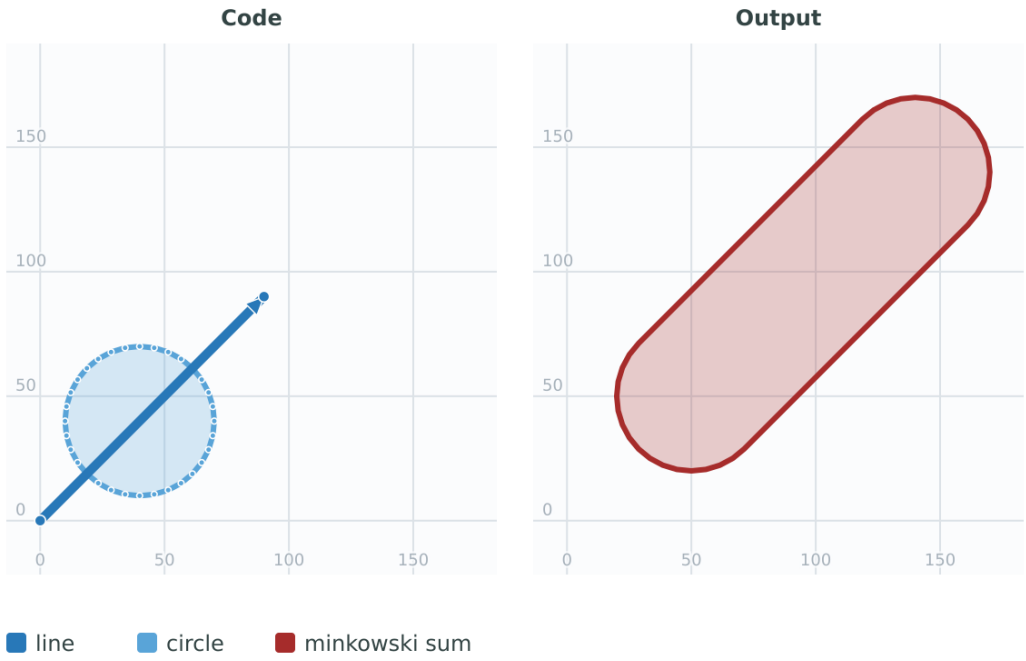
Exempel

Minkowski sum of a LineString and a circular polygon. The line crosses the polygon, and the result sweeps the polygon along the line.

Code

```
WITH data AS (
  SELECT ST_MakeLine(ST_Point(10, 10), ST_Point(100, 100)) AS line,
         ST_Buffer('POINT(50 50)'::geometry, 30) AS circle
)
SELECT line AS input_line,
       circle AS input_circle,
       CG_MinkowskiSum(line, circle) AS minkowski_sum
FROM data;
```

Figure



Minkowski sum of a MultiPoint and a polygon. The result contains one translated copy of the polygon for each point.

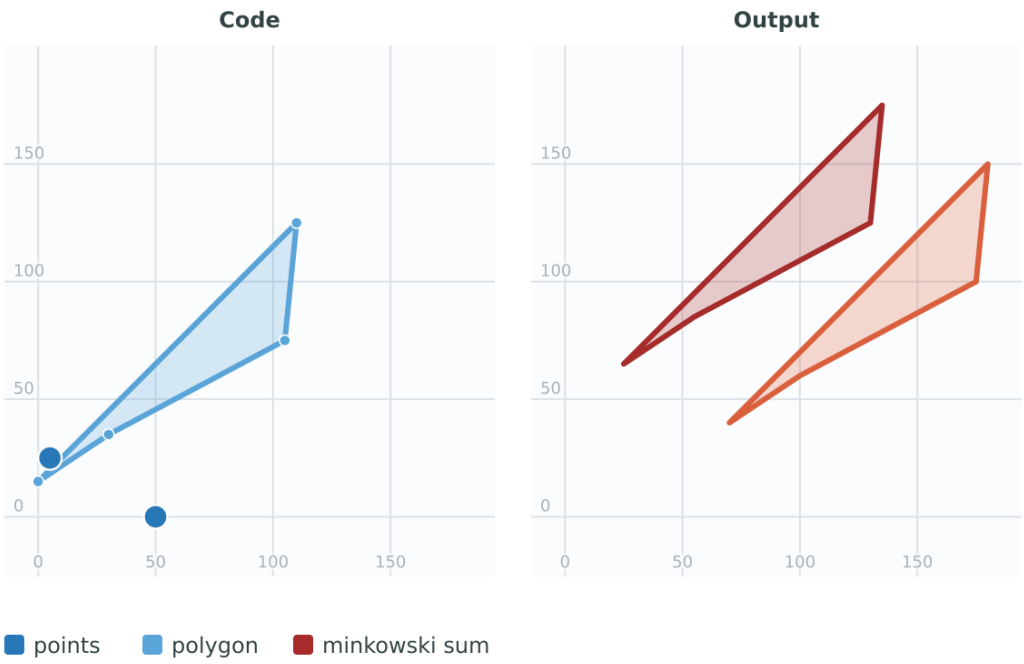
Code

```
WITH data AS (
  SELECT 'MULTIPOINT(25 50,70 25)'::geometry AS points,
         'POLYGON((130 150,20 40,50 60,125 100,130 150))'::geometry AS polygon
)
SELECT points AS input_points,
       polygon AS input_polygon,
       CG_MinkowskiSum(points, polygon) AS minkowski_sum
FROM data;
```

Output

```
MULTIPOINT((25 50),(70 25)) | POLYGON((130 150,20 40,50 60,125 100,130 150)) | MULTIPOLYGON(((45 90,75 110,150 150,155 ↩
     200,45 90)),((90 65,120 85,195 125,200 175,90 65)))
```

Figure



8.3.26 CG_OptimalAlphaShape

CG_OptimalAlphaShape — Beräknar en alpha-form som omsluter en geometri med ett öptimaltälpha-värde.

Synopsis

geometry **CG_OptimalAlphaShape**(geometry geom, boolean allow_holes = false, integer nb_components = 1);

Beskrivning

Beräknar den öptimalaälfaformen för punkterna i en geometri. Alfaformen beräknas med hjälp av ett värde på α som väljs så att:

1. antalet polygonelement är lika med eller mindre än `nb_components` (som standard är 1)
2. alla inmatningspunkter ingår i formen

Resultatet kommer inte att innehålla hål om inte det valfria argumentet `allow_holes` anges som true.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.4.1.

Denna metod behöver SFCGAL-backend.

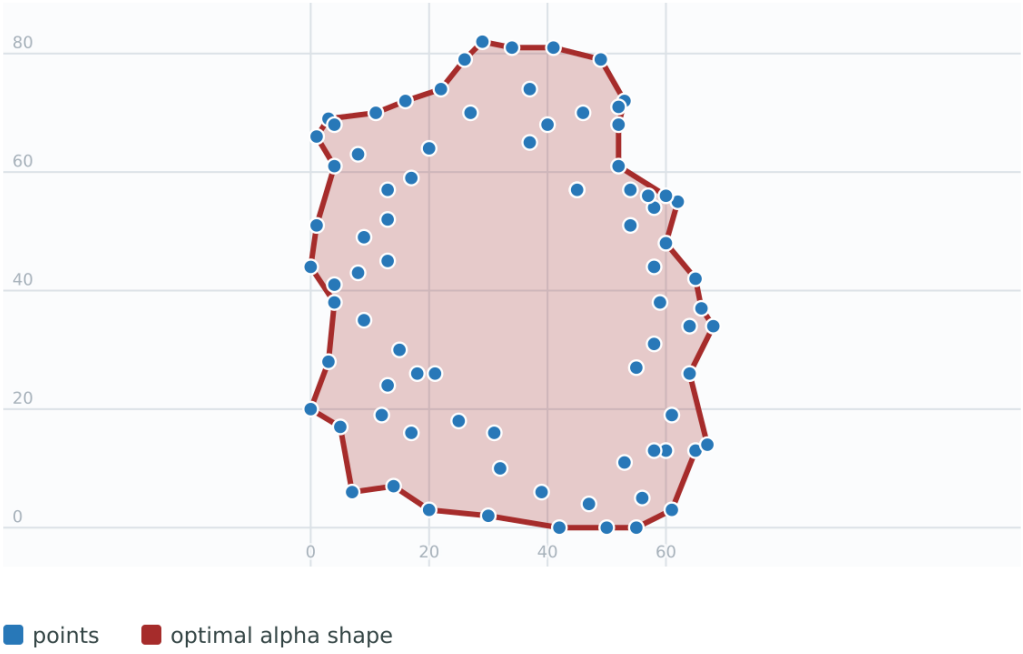
Exempel

The original point cloud and the optimal alpha-shape share one comparison frame.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),
(88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 ↩
    65),
(81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
    98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
    88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
    90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
    59),(34 86),(45 90),(64 97))'::geometry AS input_points
)
SELECT input_points AS input_points,

        CG_OptimalAlphaShape(input_points) AS optimal_alpha_shape
FROM data;
```

Figure



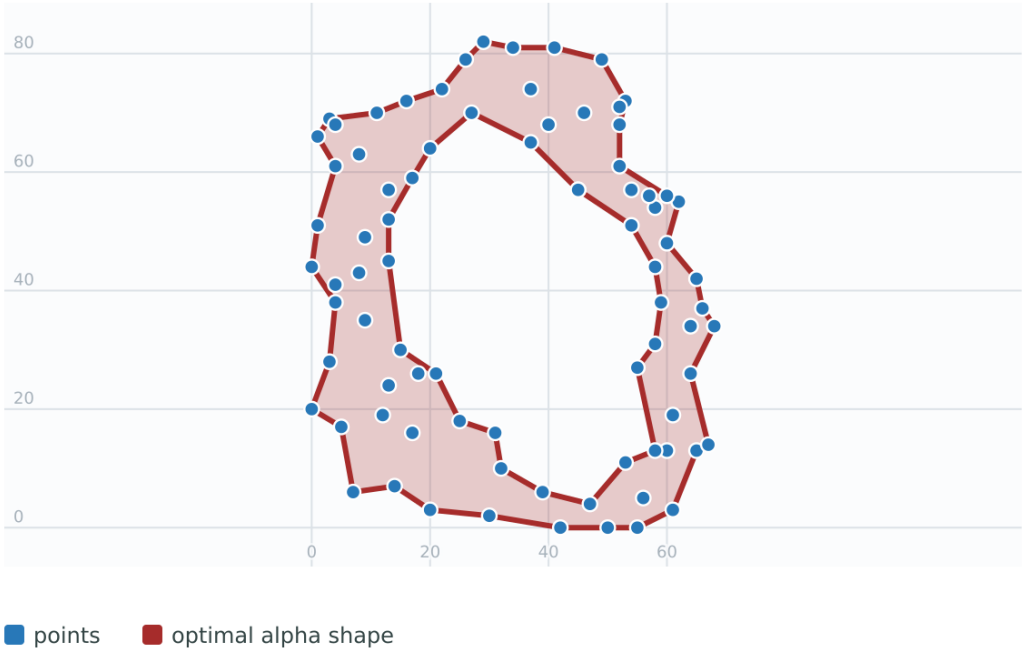
Allowing holes keeps the same outer shell but carves a central hole.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),(88 29),(24 82),(32  ↩
    51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 65),(81 47),(88  ↩
    58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
    98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
    88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
    90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
    59),(34 86),(45 90),(64 97))'::geometry AS input_points
```

```
)
SELECT input_points AS input_points,

         CG_OptimalAlphaShape(input_points, allow_holes => true) AS optimal_alpha_shape
FROM data;
```

Figure



Se även

ST_ConcaveHull, CG_AlphaShape

8.3.27 CG_OptimalConvexPartition

CG_OptimalConvexPartition — Beräknar en optimal konvex partition av polygongeometrin

Synopsis

geometry **CG_OptimalConvexPartition**(geometry geom);

Beskrivning

Beräknar en optimal konvex partition av polygongeometrin.

Notera
En partition av en polygon P är en uppsättning polygoner som är sådana att polygonernas insidor inte skär varandra och att polygonernas förening är lika med insidan av den ursprungliga polygonen P. CG_OptimalConvexPartition producerar en partition som är optimal med avseende på antalet delar.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0.

Kräver SFCGAL >= 1.5.0

Denna metod behöver SFCGAL-backend.

Exempel

The optimal convex partition uses the same polygon and produces the smallest documented piece count.

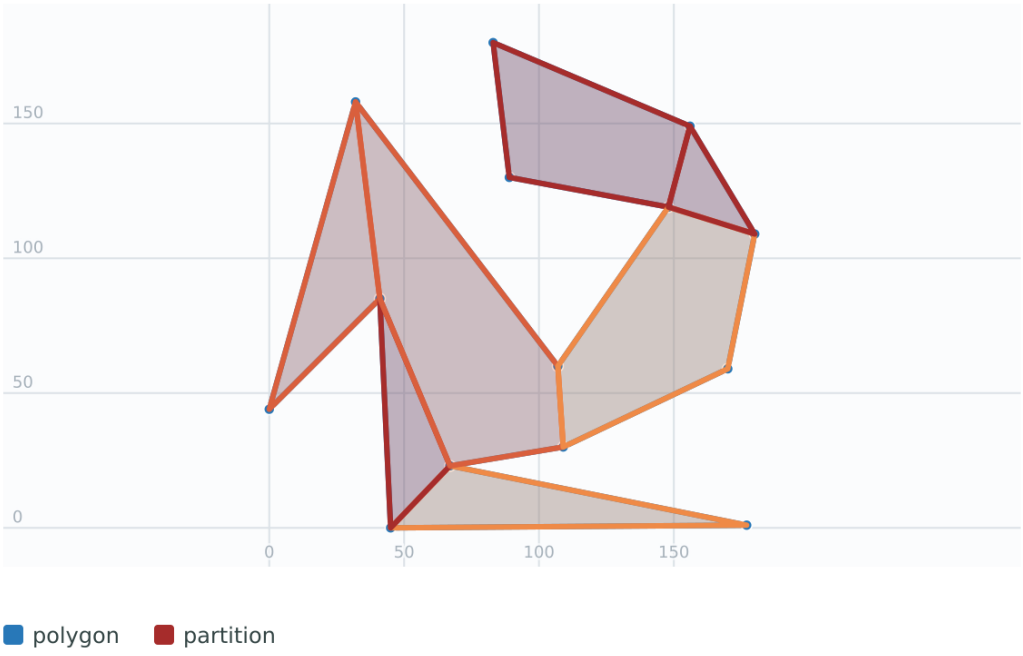
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_OptimalConvexPartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) | ↩
    GEOMETRYCOLLECTION(POLYGON((156 150,83 181,89 131,148 120,156 150)), POLYGON((32 159,0 45,41 86,32 159)), POLYGON ↩
    ((45 1,177 2,67 24,45 1)), POLYGON((41 86,45 1,67 24,41 86)), POLYGON((107 61,32 159,41 86,67 24,109 31,107 61)), ↩
     POLYGON((148 120,107 61,109 31,170 60,180 110,148 120)), POLYGON((156 150,148 120,180 110,156 150)))
```

Figure



Se även

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition

8.3.28 CG_StraightSkeleton

CG_StraightSkeleton — Beräkna ett rakt skelett från en geometri

Synopsis

geometry **CG_StraightSkeleton**(geometry geom, boolean use_distance_as_m = false);

Beskrivning

Computes the straight skeleton of a polygon by tracing the inward propagation of its edges. When *`use_distance_as_m`* is `true`, the M coordinate of each output point stores its distance from the input boundary.

Tillgänglighet: 3.5.0

Kräver SFCGAL >= 1.3.8 för alternativet use_distance_as_m

Notera
Denna funktion ignorerar Z-dimensionen. Den ger alltid ett 2D-resultat även när den används på en 3D-geometri.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder polyedriska ytor.

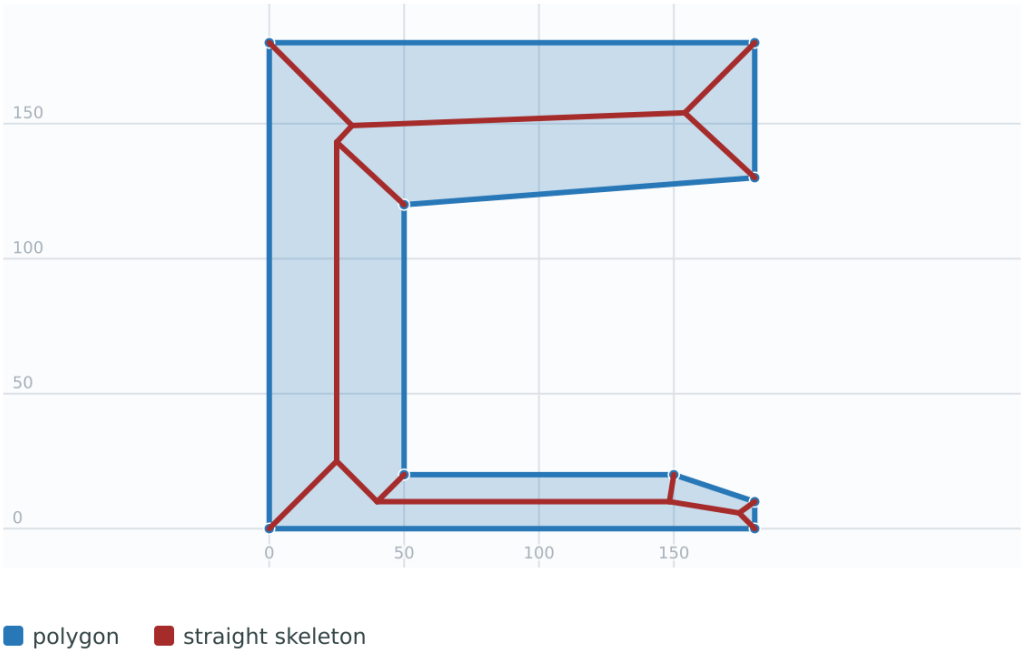
Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Code

```
WITH data AS (
  SELECT 'POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))
    '::geometry AS polygon
)
SELECT CG_StraightSkeleton(polygon) AS straight_skeleton
FROM data;
```

Figure



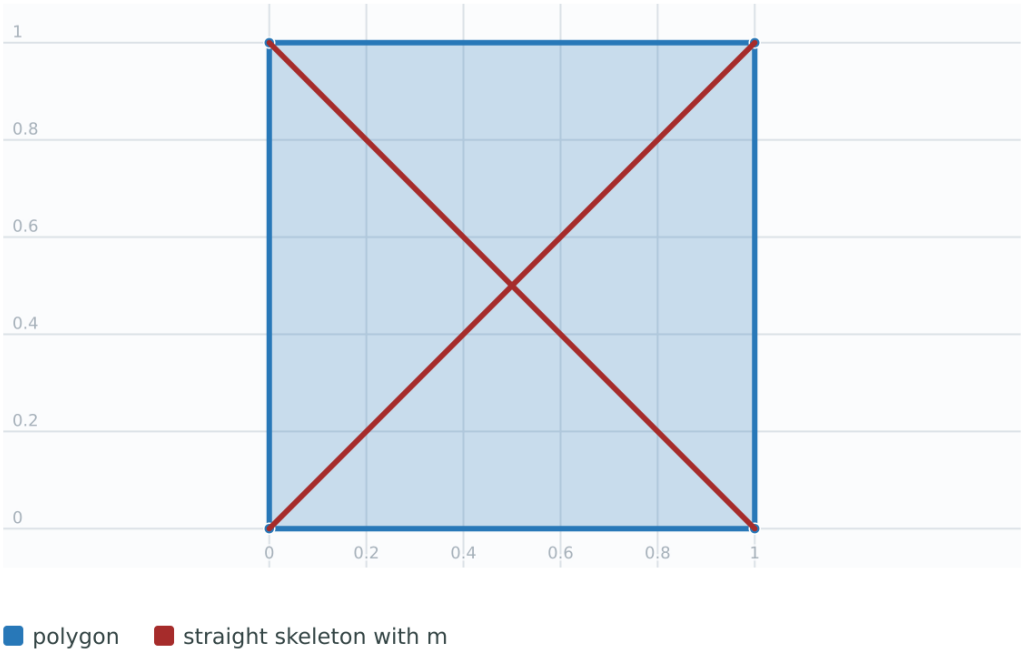
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,1 0,1 1,0 1,0 0))'::geometry AS polygon
)
SELECT CG_StraightSkeleton(polygon, true) AS straight_skeleton_with_m
FROM data;
```

Output

```
MULTILINESTRING M ((0 0 0,0.5 0.5 0.5),(1 0 0,0.5 0.5 0.5),(1 1 0,0.5 0.5 0.5),(0 1 0,0.5 0.5 0.5))
```

Figure



Observera att giltiga inmatningar med ringar som rör vid en enda punkt kommer att ge upphov till ett fel.

Code

```
SELECT CG_StraightSkeleton('POLYGON((0 0,3 0,3 3,0 3,0 0),(0 0,1 2,2 1,0 0))'));
```

Output

```
NOTICE:  During straight_skeleton(A) :
NOTICE:    with A: POLYGON((0/1 0/1,3/1 0/1,3/1 3/1,0/1 3/1,0/1 0/1),(0/1 0/1,1/1 2/1,2/1 1/1,0/1 0/1))
ERROR:  straight skeleton of Polygon with point touching rings is not implemented.
```

Se även

CG_StraightSkeletonPartition, CG_ExtrudeStraightSkeleton

8.3.29 CG_Tesselate

CG_Tesselate — Tessellerar ytan på en polygon eller polyederyta och returnerar som en TIN eller en samling av TINS

Synopsis

geometry **CG_Tesselate**(geometry geom);

Beskrivning

Tar som indata en yta som MULTI(POLYGON) eller POLYHEDRALSURFACE och returnerar en TIN-representation via tessellationsprocessen med hjälp av trianglar.

Notera
ST_TriangulatePolygon liknar denna funktion förutom att den returnerar en geometrisamling av polygoner istället för en TIN och dessutom endast fungerar med 2D-geometrier.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

Exempel

Tessellation divides each face of a cube into triangles.

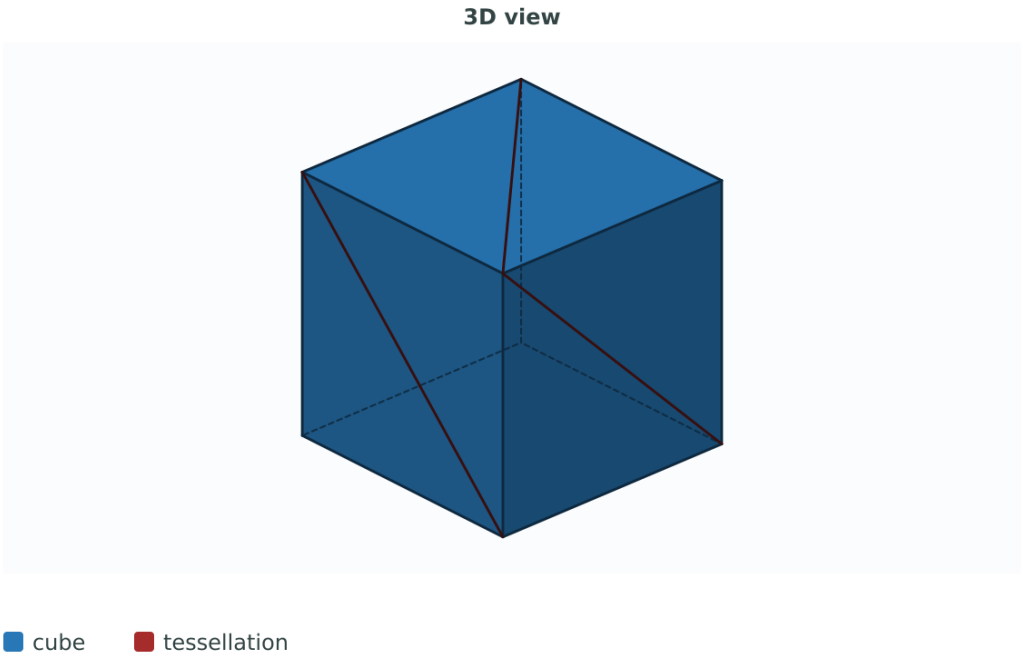
Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry AS cube
)
SELECT CG_Tesselate(cube) AS tessellation
FROM data;
```

Output

```
TIN Z (((0 0 0,0 0 1,0 1 1,0 0 0)),((0 1 0,0 0 0,0 1 1,0 1 0)),((0 0 0,0 1 0,1 1 0,0 0 0)),((1 0 0,0 0 0,1 1 0,1 0 0)) ↩
    ,((0 0 1,1 0 0,1 0 1,0 0 1)),((0 0 1,0 0 0,1 0 0,0 0 1)),((1 1 0,1 1 1,1 0 1,1 1 0)),((1 0 0,1 1 0,1 0 1,1 0 0)) ↩
    ,((0 1 0,0 1 1,1 1 1,0 1 0)),((1 1 0,0 1 0,1 1 1,1 1 0)),((0 1 1,1 0 1,1 1 1,0 1 1)),((0 1 1,0 0 1,1 0 1,0 1 1)))
```

Figure



Tessellation of a Polygon.

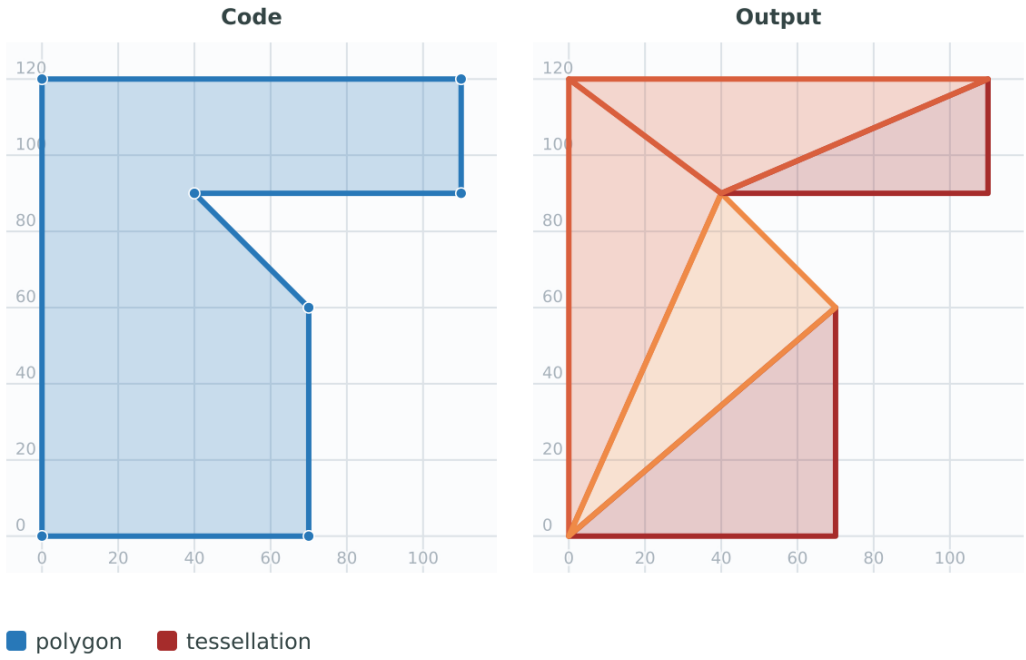
Code

```
WITH data AS (
  SELECT 'POLYGON((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190))'::geometry AS
    polygon
)
SELECT polygon AS input_polygon,
       CG_Tesselate(polygon) AS tessellation
FROM data;
```

Output

```
POLYGON((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190)) | TIN(((80 130,10 70,80 70,80 130)),((50 160,10
    190,10 70,50 160)),((80 130,50 160,10 70,80 130)),((120 160,120 190,50 160,120 160)),((120 190,10 190,50 160,120
    190)))
```

Figure



Se även

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.30 CG_Triangulate

CG_Triangulate — Triangulerar en polygonal geometri

Synopsis

geometry **CG_Triangulate**(geometry geom);

Beskrivning

Triangulerar en polygonal geometri.
Utförs av SFCGAL-modulen

> **Notera**
> OBS: Denna funktion returnerar en geometri som representerar det triangulerade resultatet.

Tillgänglighet: 3.5.0

Denna metod behöver SFCGAL-backend.

Exempel på geometri

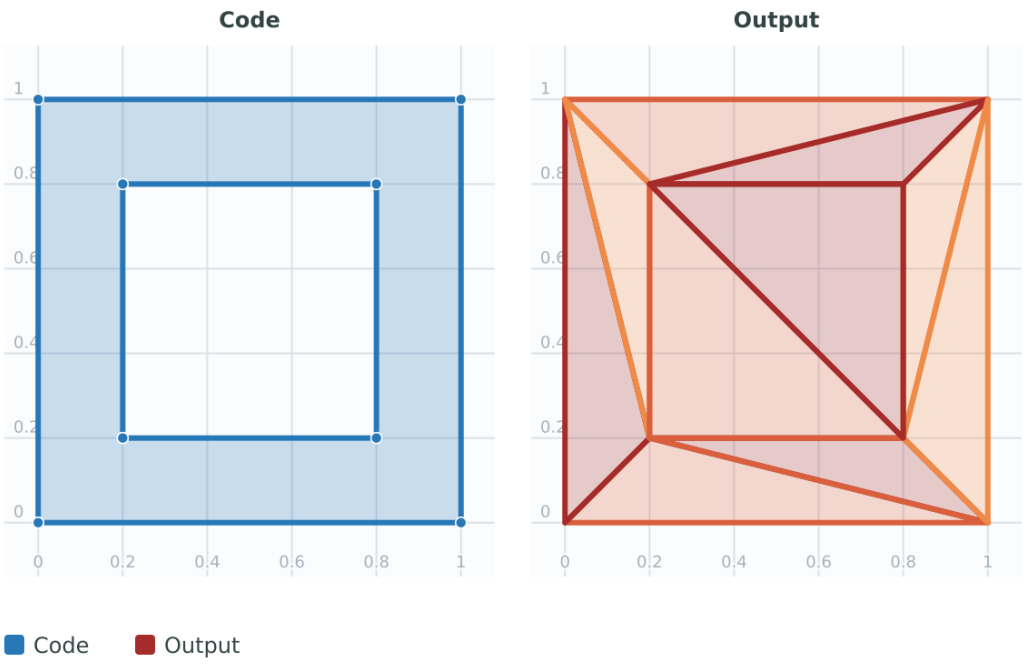
Code

```
SELECT CG_Triangulate(
  'POLYGON((0 0,1 0,1 1,0 1,0 0),
           (0.2 0.2,0.2 0.8,0.8 0.8,0.8 0.2,0.2 0.2))'
);
```

Output

```
TIN(((0.8 0.2,0.2 0.2,1 0,0.8 0.2)),((0.2 0.2,0 0,1 0,0.2 0.2)),((1 1,0.8 0.8,0.8 0.2,1 1)),((0 1,0 0,0.2 0.2,0 1)) ↩
    ,((0 1,0.2 0.8,1 1,0 1)),((0 1,0.2 0.2,0.2 0.8,0 1)),((0.2 0.8,0.8 0.8,1 1,0.2 0.8)),((0.2 0.8,0.2 0.2,0.8 ↩
    0.2,0.2 0.8)),((1 1,0.8 0.2,1 0,1 1)),((0.8 0.8,0.2 0.8,0.8 0.2,0.8 0.8)))
```

Figure



Se även

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.31 CG_Visibility

CG_Visibility — Beräkna en synlighetspolygon från en punkt eller ett segment i en polygongeometri

Synopsis

geometry **CG_Visibility**(geometry polygon, geometry point);
geometry **CG_Visibility**(geometry polygon, geometry pointA, geometry pointB);

Beskrivning

Returns the portion of a polygon visible from an observer point, or from the observer segment defined by *`pointA`* and *`pointB`*.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0.

Kräver SFCGAL >= 1.5.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder polyedriska ytor.

Denna funktion stöder trianglar och triangulerade oregelbundna nätverksytor (TIN).

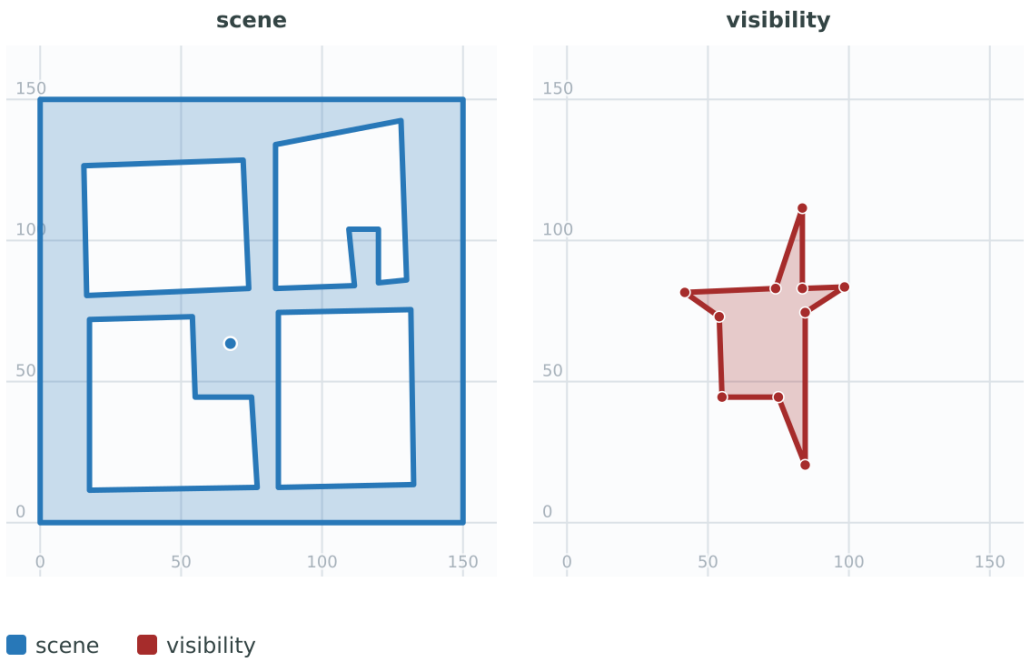
Exempel

Visibility from a point inside a polygon with holes.

Code

```
WITH data AS (
  SELECT 'POLYGON((23.5 23.5,23.5 173.5,173.5 173.5,173.5 23.5,23.5 23.5),(108 98,108 ↩
    36,156 37,155 99,108 98),(107 157.5,107 106.5,135 107.5,133 127.5,143.5 127.5,143.5 ↩
    108.5,153.5 109.5,151.5 166,107 157.5),(41 95.5,41 35,100.5 36,98.5 68,78.5 68,77.5 ↩
    96.5,41 95.5),(39 150,40 104,97.5 106.5,95.5 152,39 150))'::geometry AS polygon,
         'POINT(91 87)'::geometry AS observer
)
SELECT ST_Collect(polygon, observer) AS input_scene,
       CG_Visibility(polygon, observer) AS visibility
FROM data;
```

Figure

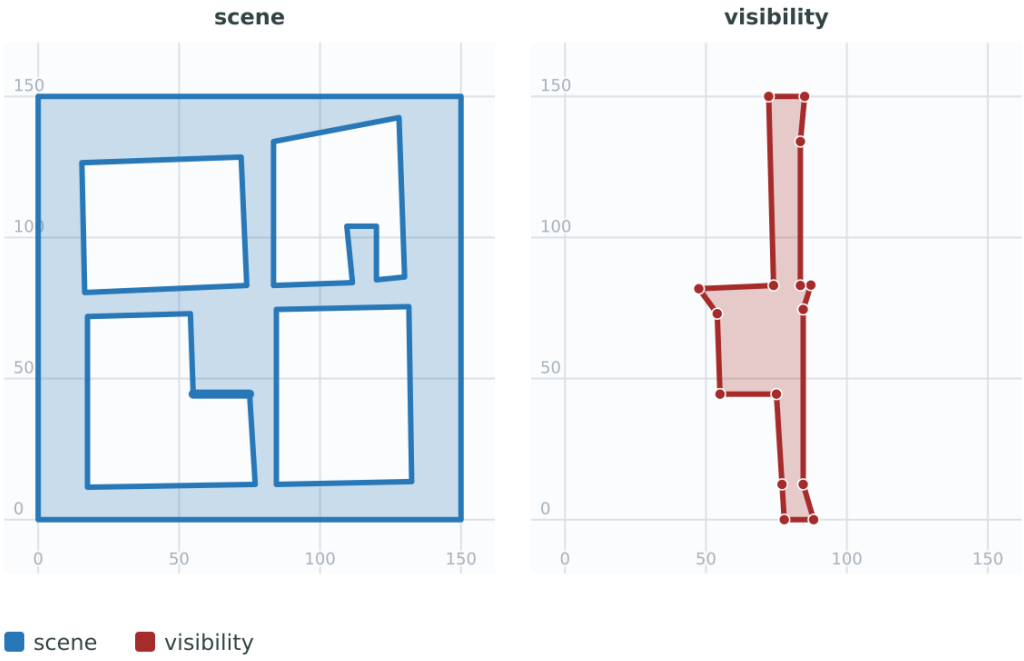


Visibility from a segment inside the same polygon.

Code

```
WITH data AS (
  SELECT 'POLYGON((23.5 23.5,23.5 173.5,173.5 173.5,173.5 23.5,23.5 23.5),(108 98,108 ↩
    36,156 37,155 99,108 98),(107 157.5,107 106.5,135 107.5,133 127.5,143.5 127.5,143.5 ↩
    108.5,153.5 109.5,151.5 166,107 157.5),(41 95.5,41 35,100.5 36,98.5 68,78.5 68,77.5 ↩
    96.5,41 95.5),(39 150,40 104,97.5 106.5,95.5 152,39 150))'::geometry AS polygon,
         'LINESTRING(78.5 68,98.5 68)'::geometry AS observer
)
SELECT ST_Collect(polygon, observer) AS input_scene,
       CG_Visibility(polygon,
                     ST_StartPoint(observer),
                     ST_EndPoint(observer)) AS visibility
FROM data;
```

Figure



8.3.32 CG_YMonotonePartition

CG_YMonotonePartition — Beräknar y-monoton partition av polygongeometrin

Synopsis

geometry **CG_YMonotonePartition**(geometry geom);

Beskrivning

Beräknar y-monoton partition av polygongeometrin.

Notera
En partition av en polygon P är en uppsättning polygoner som är sådana att polygonernas inre delar inte skär varandra och att polygonernas förening är lika med den ursprungliga polygonen P:s inre del. En y-monoton polygon är en polygon vars hörn v1,...,vn kan delas upp i två kedjor v1,...,vk och vk,...,vn,v1, så att varje horisontell linje skär endera kedjan högst en gång. Denna algoritm garanterar inte en gräns för antalet polygoner som produceras i förhållande till det optimala antalet.

Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0.

Kräver SFCGAL >= 1.5.0

Denna metod behöver SFCGAL-backend.

Exempel

The same polygon can also be partitioned into y-monotone pieces for downstream processing.

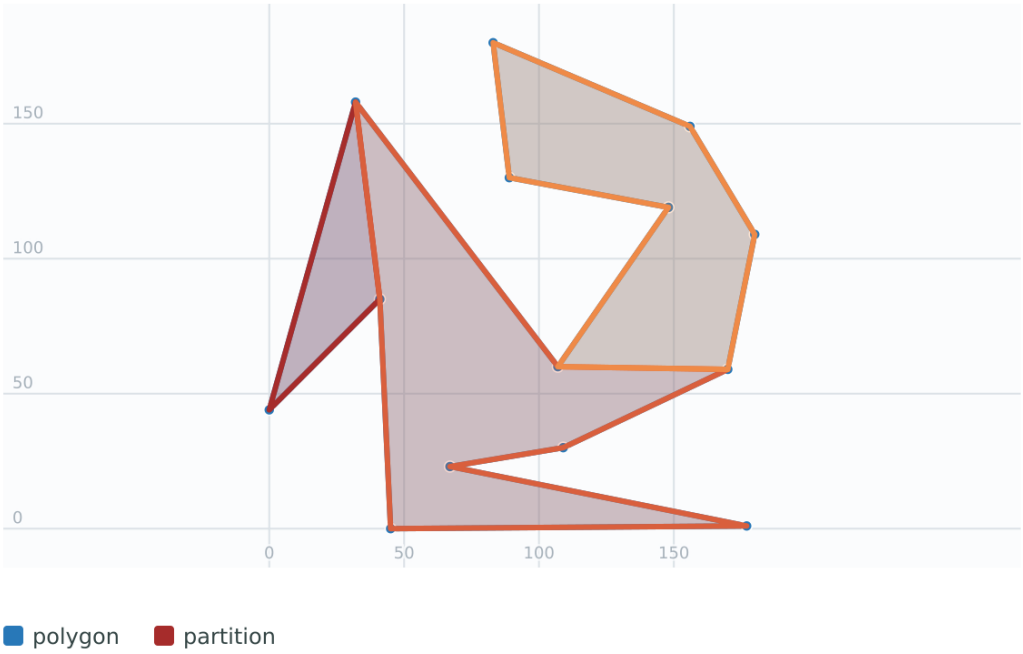
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_YMonotonePartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) | ↩
    GEOMETRYCOLLECTION(POLYGON((32 159,0 45,41 86,32 159)), POLYGON((107 61,32 159,41 86,45 1,177 2,67 24,109 31,170 ↩
    60,107 61)), POLYGON((156 150,83 181,89 131,148 120,107 61,170 60,180 110,156 150)))
```

Figure



Se även

CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

8.3.33 CG_StraightSkeletonPartition

CG_StraightSkeletonPartition — Beräknar den raka skelettpartitionen av en polygon.

Synopsis

geometry **CG_StraightSkeletonPartition**(geometry geom, boolean auto_orientation);

Beskrivning

Beräknar den raka skelettpartitionen av den inmatade polygongeometrin *geom*. Det raka skelettet är en partitionering av polygonen i ytor som bildas genom att spåra kollapsen av dess kanter. Om *auto_orientation* är satt till true kommer funktionen automatiskt att justera indatapolygonens orientering för att säkerställa korrekta resultat.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0.

Denna metod behöver SFCGAL-backend.

Exempel

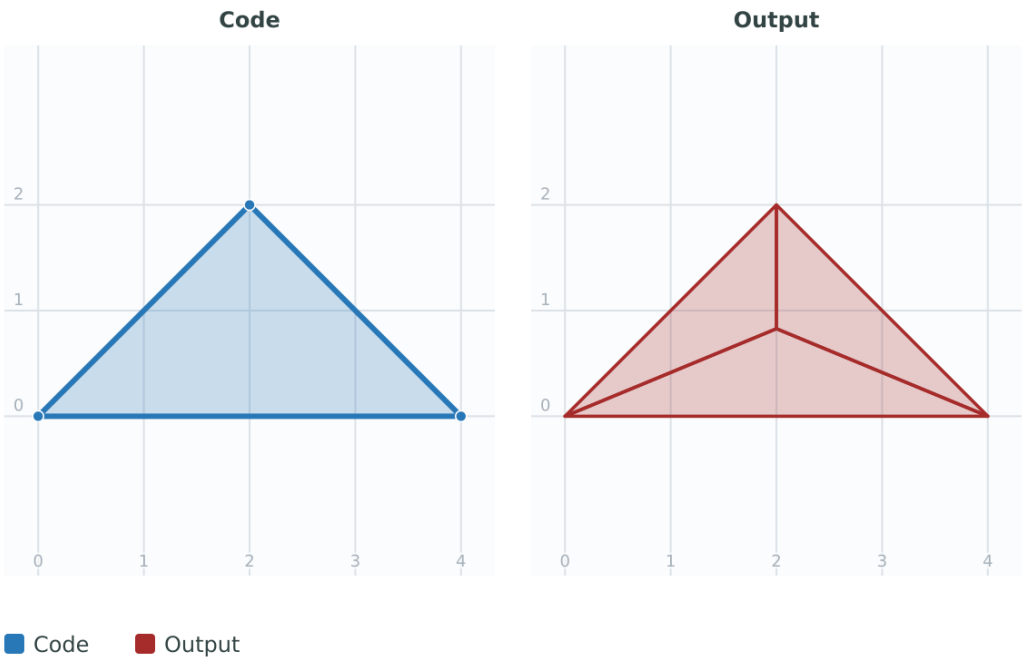
Code

```
SELECT CG_StraightSkeletonPartition('POLYGON((0 0,4 0,2 2,0 0))', true);
```

Output

```
POLYHEDRALSURFACE(((0 0,2 0.82842712474619,2 2,0 0)),((4 0,2 0.82842712474619,0 0,4 0)),((2 2,2 0.82842712474619,4 0,2 ↩
    2)))
```

Figure

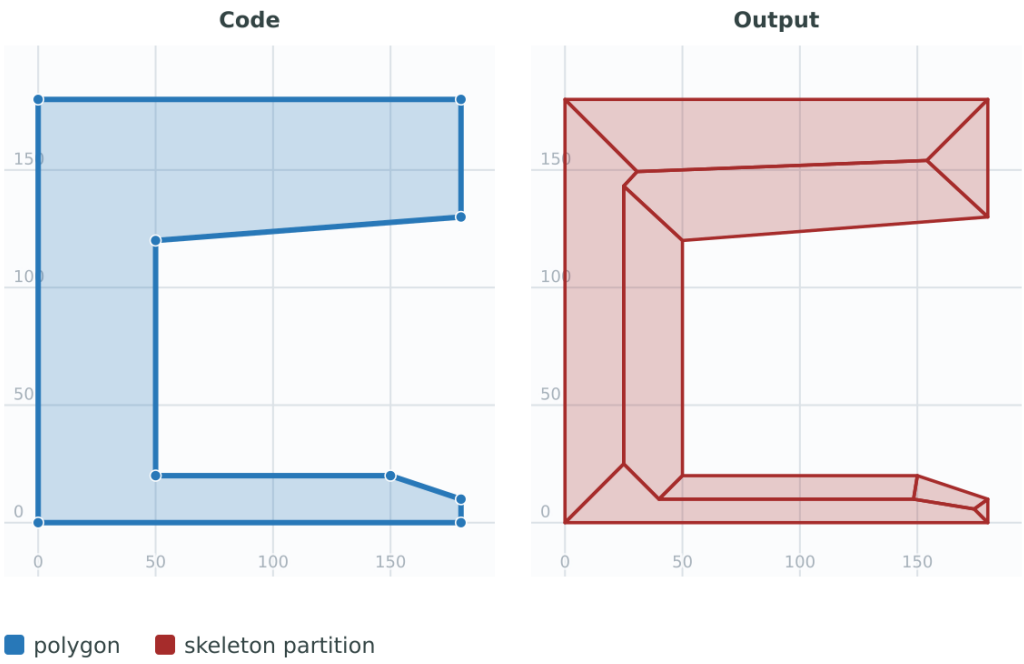


Straight skeleton partition of a polygon.

Code

```
WITH data AS (
  SELECT 'POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))'::geometry AS polygon
)
SELECT polygon AS input_polygon,
       CG_StraightSkeletonPartition(polygon, true) AS skeleton_partition
FROM data;
```

Figure



Se även

CG_StraightSkeleton, ST_Polygonize

8.3.34 CG_3DBuffer

CG_3DBuffer — Beräknar en 3D-buffert runt en geometri.

Synopsis

geometry **CG_3DBuffer**(geometry geom, float8 radius, integer segments, integer buffer_type);

Beskrivning

Genererar en 3D-buffert runt indatageometrin *`geom`* med en angiven *`radie`*. Bufferten konstrueras i 3D-rymd och skapar en volymetrisk representation av geometrins omgivning. Parametern *`segments`* definierar antalet segment som används för att approximera de krökta delarna av bufferten, med ett minimivärde på 4 segment. Parametern *`buffer_type`* anger vilken typ av buffert som ska skapas: 0: Rundad buffert (standard) 1: Platt buffert 2: Kvadratisk buffert

Indatageometrin måste vara en Point eller LineString.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Exempel

Code

```
SELECT CG_3DBuffer('POINT(0 0 0)', 1, 8, 0);
```

Output

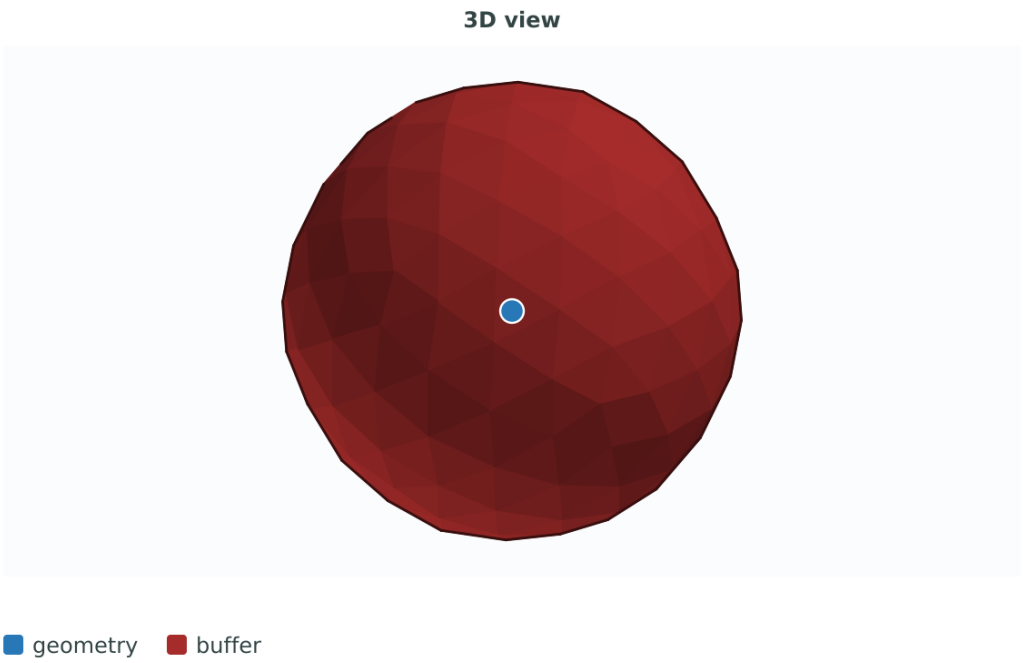
```
POLYHEDRALSURFACE Z (((0 0 1, 0.5 -0.5 0.71, 0 -0.71 0.71, 0 0 1)), ... )
```

A rounded point buffer using 32 segments.

Code

```
SELECT
  'POINT(100 90)'::geometry AS input_geometry,
  CG_3DBuffer('POINT(100 90)'::geometry, 50, 32, 0) AS buffer;
```

Figure

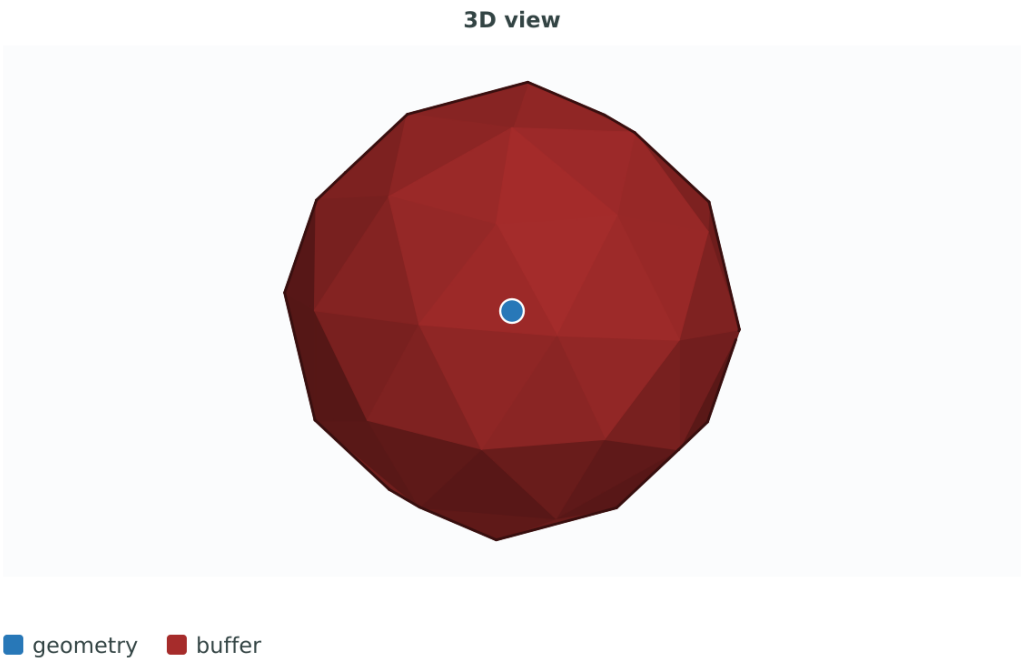


A coarser rounded point buffer using 5 segments.

Code

```
SELECT
  'POINT(100 90)'::geometry AS input_geometry,
  CG_3DBuffer('POINT(100 90)'::geometry, 50, 5, 0) AS buffer;
```

Figure

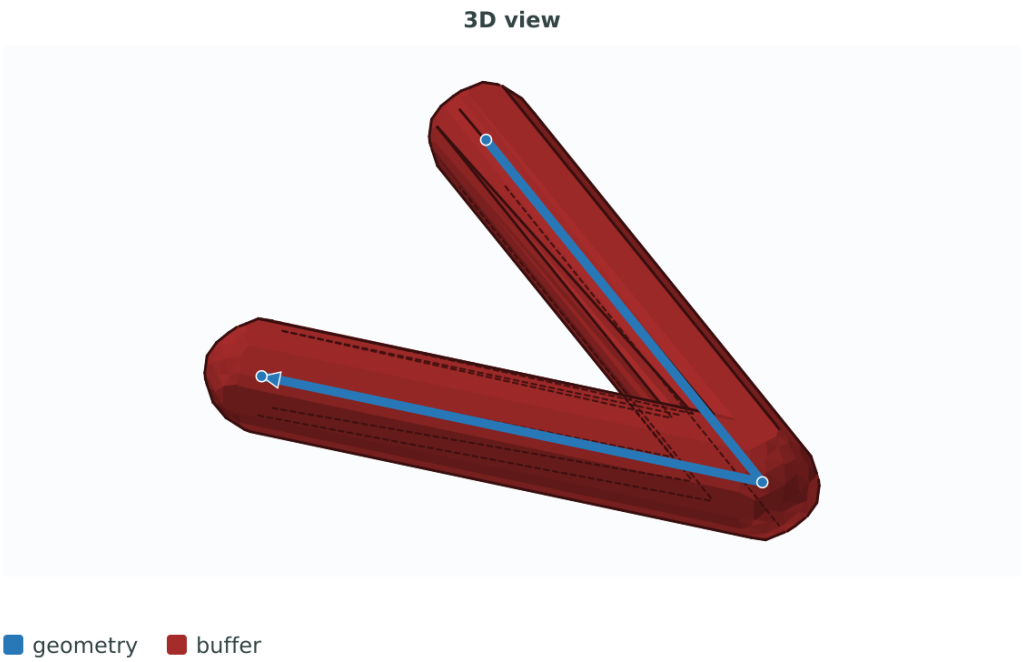


A rounded buffer around a line using 32 segments.

Code

```
SELECT
  'LINESTRING(50 50,150 150,150 50)'::geometry AS input_geometry,
  CG_3DBuffer('LINESTRING(50 50,150 150,150 50)'::geometry, 10, 32, 0) AS buffer;
```

Figure

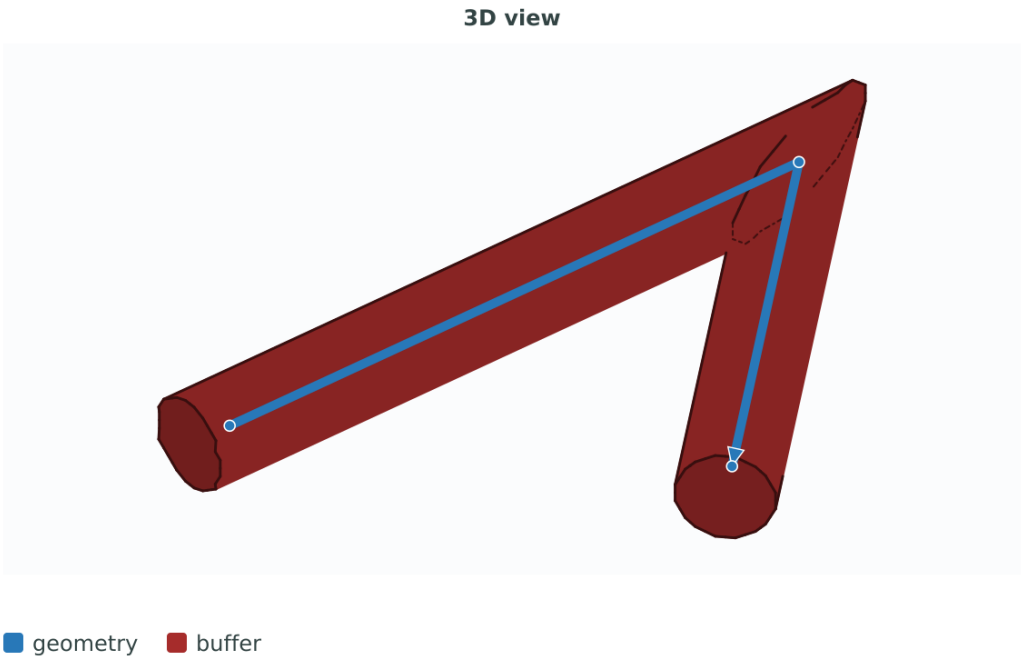


A square buffer around the same line using 32 segments.

Code

```
SELECT
  'LINESTRING(50 50,150 150,150 50)'::geometry AS input_geometry,
  CG_3DBuffer('LINESTRING(50 50,150 150,150 50)'::geometry, 10, 32, 2) AS buffer;
```

Figure



Se även

ST_Buffer, CG_3DConvexHull, ST_AsX3D

8.3.35 CG_Rotate

CG_Rotate — Roterar en geometri med en given vinkel runt origo (0,0).

Synopsis

geometry **CG_Rotate**(geometry geom, float8 angle);

Beskrivning

Roterar indatageometrin *`geom`* med *`vinkelradianer`* runt ursprungspunkten (0,0). Rotationen utförs i 2D-rymden; Z-koordinaterna ändras inte. Positiva vinklar roterar geometrin moturs.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Exempel

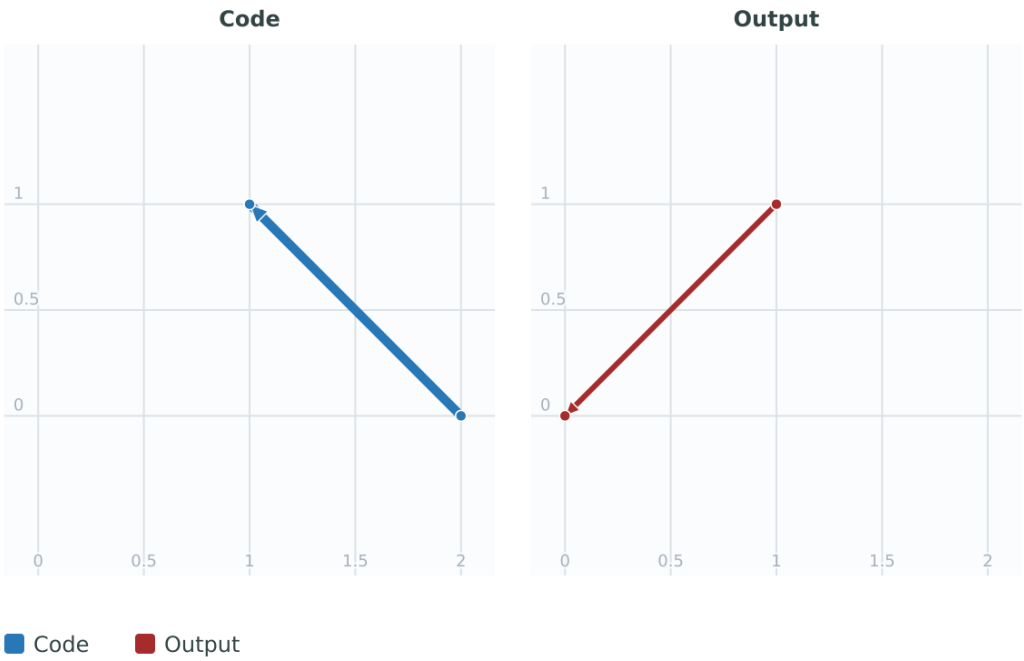
Code

```
SELECT CG_Rotate('LINESTRING(1 0,0 1)', pi()/2);
```

Output

```
LINESTRING(6.123233995736766e-17 1,-1 6.123233995736766e-17)
```

Figure



Se även

CG_2DRotate, ST_Rotate

8.3.36 CG_2DRotate

CG_2DRotate — Roterar en geometri med en given vinkel runt en angiven punkt i 2D.

Synopsis

geometry **CG_2DRotate**(geometry geom, float8 angle, float8 cx, float8 cy);

Beskrivning

Roterar indatageometrin *geom* med *angle*-radianer runt punkten (*cx*, *cy*). Rotationen utförs i 2D-rymd; Z-koordinater utelämnas. Positiva vinklar roterar geometrin moturs.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Exempel

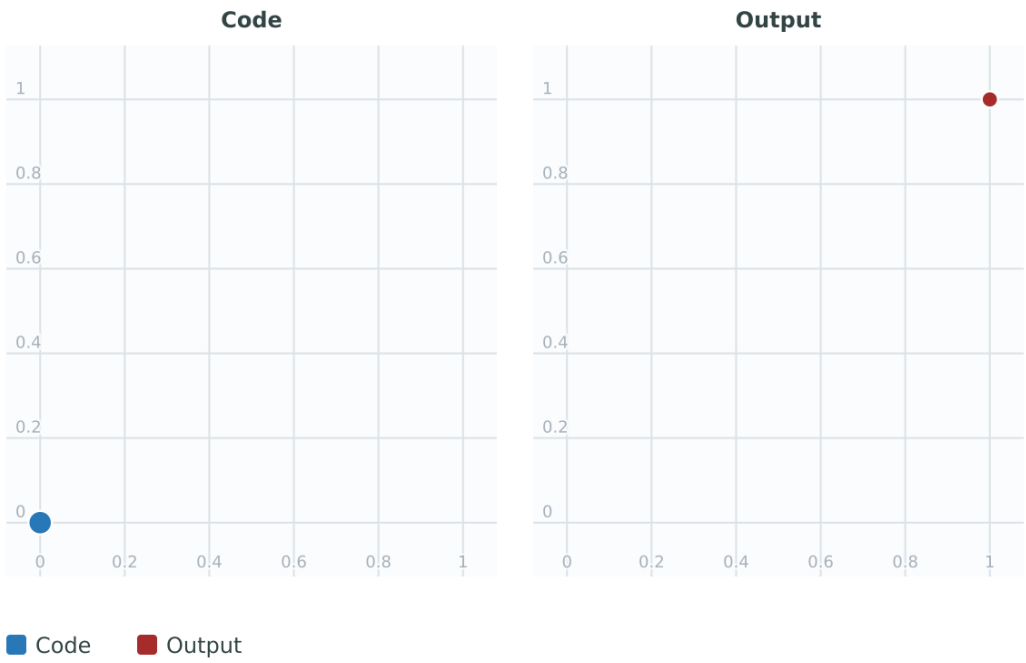
Code

```
SELECT CG_2DRotate('POINT(1 0)', pi()/2, 1, 1);
```

Output

```
POINT(2 1)
```

Figure



Se även

CG_Rotate, CG_3DRotate

8.3.37 CG_3DRotate

CG_3DRotate — Roterar en geometri i 3D-rymden runt en axelvektor.

Synopsis

geometry **CG_3DRotate**(geometry geom, float8 angle, float8 ax, float8 ay, float8 az);

Beskrivning

Roterar indatageometrin *geom* med *vinkelradianer* runt en axel som definieras av vektorn*(ax, ay, az)* som passerar genom origo (0,0,0).

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

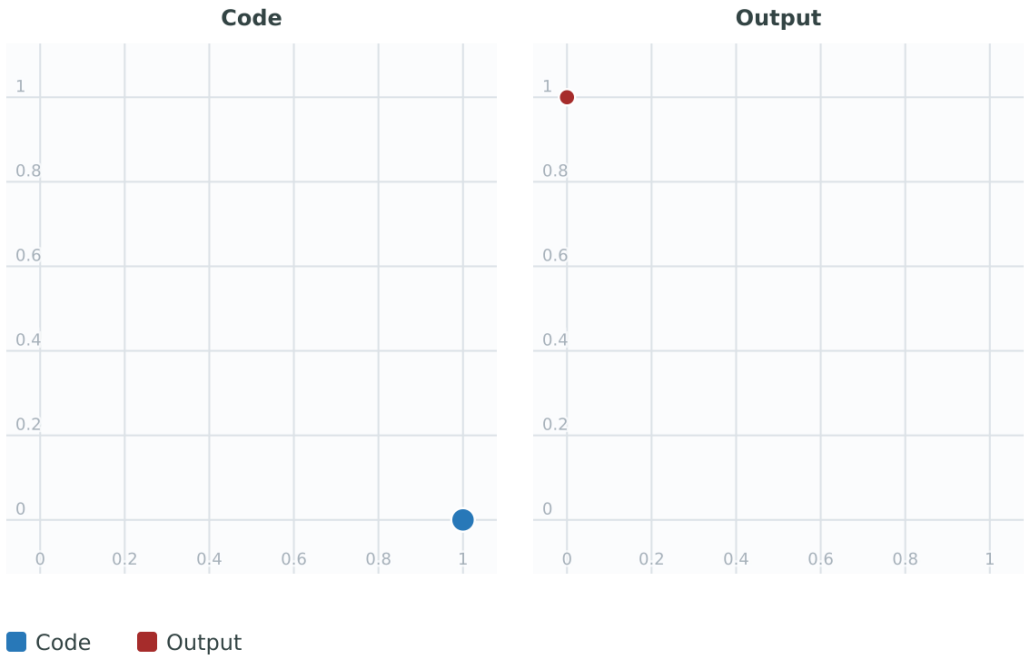
Code

```
SELECT CG_3DRotate('POINT(1 0 0)', pi()/2, 0, 0, 1);
```

Output

```
POINT Z (6.123233995736766e-17 1 0)
```

Figure



Se även

CG_RotateX, CG_RotateY, CG_RotateZ

8.3.38 CG_RotateX

CG_RotateX — Roterar en geometri runt X-axeln med en given vinkel.

Synopsis

geometry **CG_RotateX**(geometry geom, float8 angle);

Beskrivning

Roterar indatageometrin *`geom`* med *`vinkelradianer`* runt X-axeln.
Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

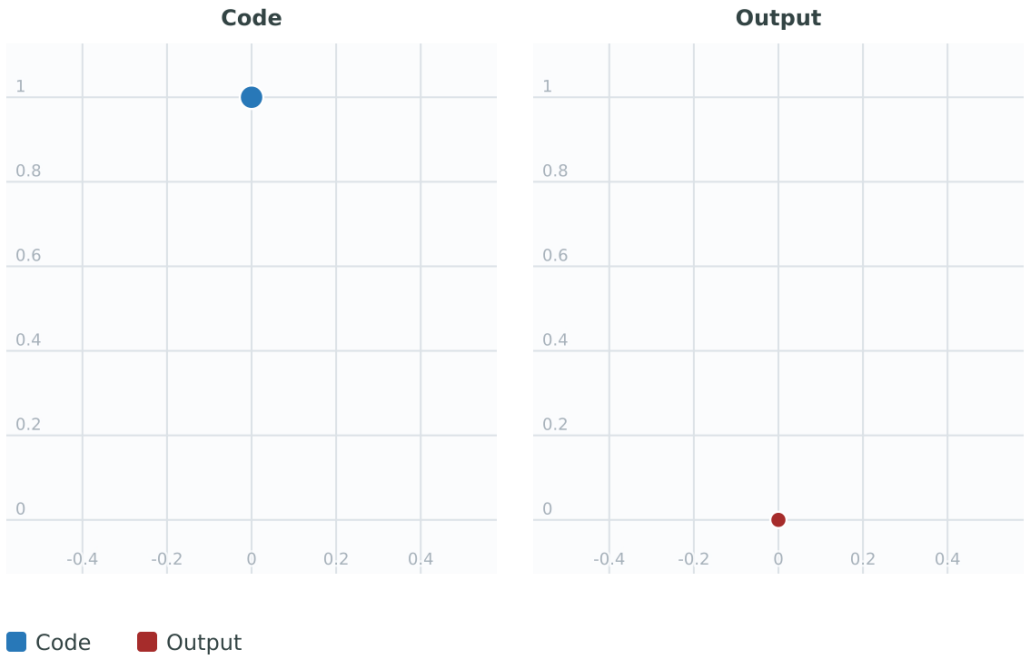
Code

```
SELECT CG_RotateX('POINT(0 1 0)', pi()/2);
```

Output

```
POINT Z (0 6.123233995736766e-17 1)
```

Figure



Se även

CG_RotateY, CG_RotateZ, CG_3DRotate

8.3.39 CG_RotateY

CG_RotateY — Roterar en geometri runt Y-axeln med en given vinkel.

Synopsis

geometry **CG_RotateY**(geometry geom, float8 angle);

Beskrivning

Roterar indatageometrin *`geom`* med *`vinkelradianer`* runt Y-axeln passerar.
Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

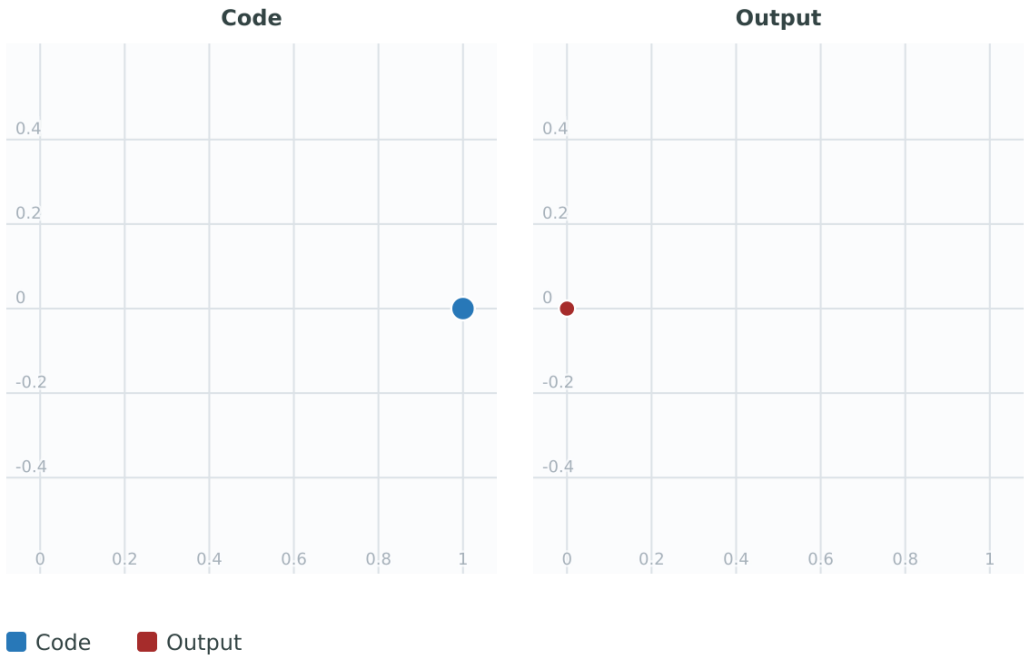
Code

```
SELECT CG_RotateY('POINT(1 0 0)', pi()/2);
```

Output

```
POINT Z (6.123233995736766e-17 0 -1)
```

Figure



Se även

CG_RotateX, CG_RotateZ, CG_3DRotate

8.3.40 CG_RotateZ

CG_RotateZ — Roterar en geometri runt Z-axeln med en given vinkel.

Synopsis

geometry **CG_RotateZ**(geometry geom, float8 angle);

Beskrivning

Roterar indatageometrin *geom* med *vinkelradianer* runt Z-axeln.
Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

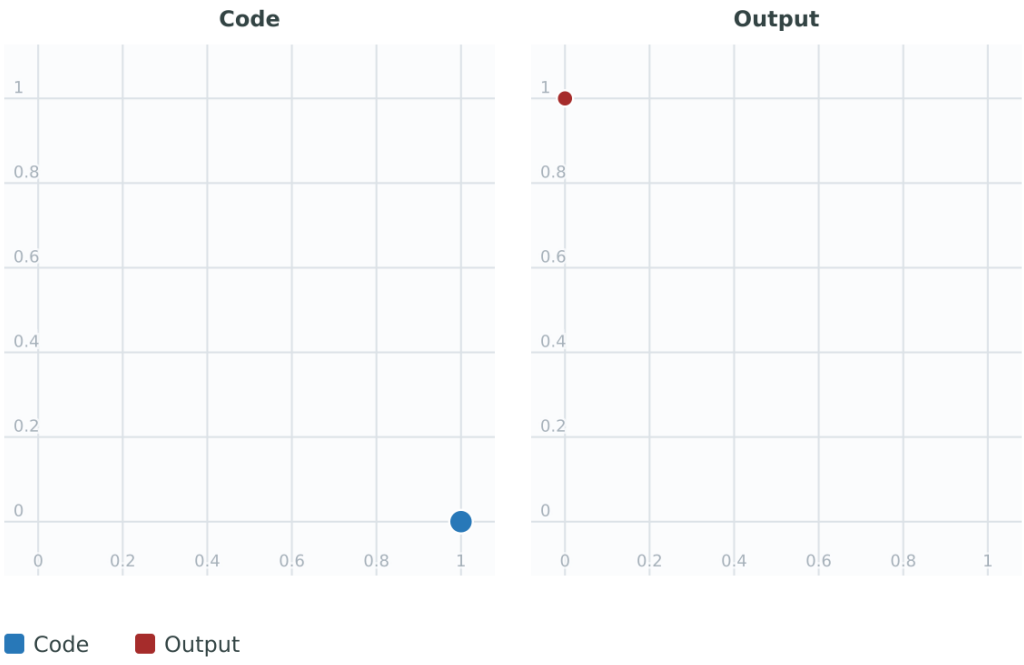
Code

```
SELECT CG_RotateZ('POINT(1 0 0)', pi()/2);
```

Output

```
POINT Z (6.123233995736766e-17 1 0)
```

Figure



Se även

CG_RotateX, CG_RotateY, CG_3DRotate

8.3.41 CG_Scale

CG_Scale — Skalar en geometri enhetligt i alla dimensioner med en given faktor.

Synopsis

geometry **CG_Scale**(geometry geom, float8 factor);

Beskrivning

Skalar inmatningsgeometrin *`geom`* med en enhetlig *`skalfaktor`* i alla dimensioner (X, Y och Z). Skalningen utförs i förhållande till ursprungspunkten (0,0,0).

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Exempel

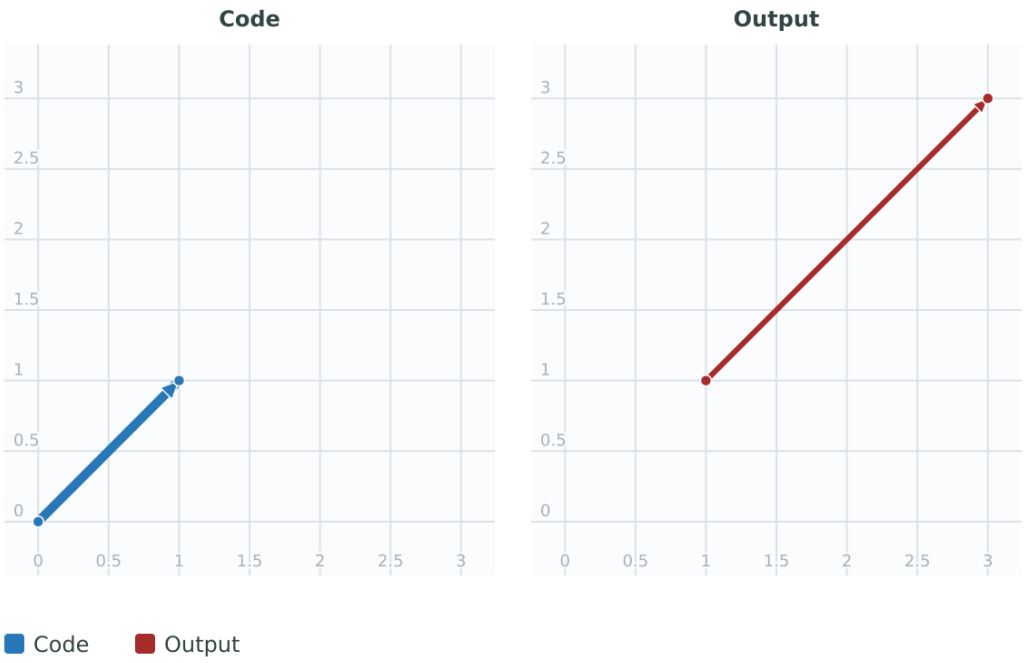
Code

```
SELECT CG_Scale('LINESTRING(1 1,2 2)', 2);
```

Output

```
LINESTRING(2 2,4 4)
```

Figure



Se även

CG_3DScale, CG_3DScaleAroundCenter, ST_Scale

8.3.42 CG_3DScale

CG_3DScale — Skalar en geometri med separata faktorer längs X-, Y- och Z-axlarna.

Synopsis

geometry **CG_3DScale**(geometry geom, float8 factorX, float8 factorY, float8 factorZ);

Beskrivning

Skalar inmatningsgeometrin *geom* med olika faktorer längs X-, Y- och Z-axlarna. Skalningen utförs i förhållande till ursprungspunkten (0,0,0).

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT CG_3DScale('POINT(1 1 1)', 2, 3, 4);
```

Output

```
POINT Z (2 3 4)
```

Figure

Se även

CG_Scale, CG_3DScaleAroundCenter

8.3.43 CG_3DScaleAroundCenter

CG_3DScaleAroundCenter — Skalar en geometri i 3D-rymden runt en angiven mittpunkt.

Synopsis

geometry **CG_3DScaleAroundCenter**(geometry geom, float8 factorX, float8 factorY, float8 factorZ, float8 centerX, float8 centerY, float8 centerZ);

Beskrivning

Skalar indatageometrin *geom* med olika faktorer längs X-, Y- och Z-axlarna i förhållande till en angiven mittpunkt (*centerX*, *centerY*, *centerZ*).

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

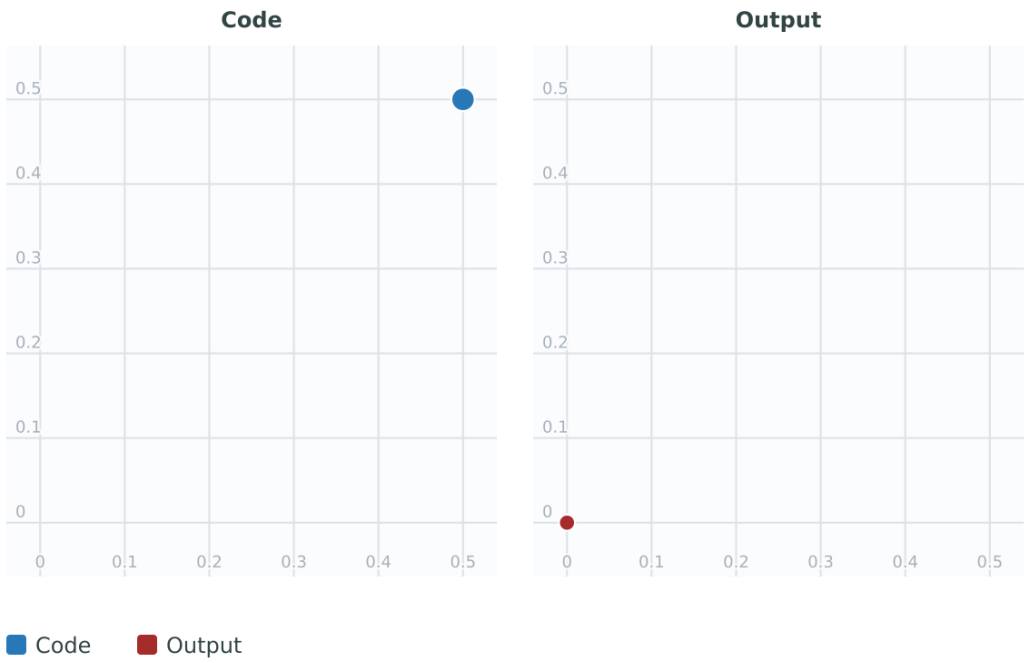
Code

```
SELECT CG_3DScaleAroundCenter('POINT(2 2 2)', 0.5, 0.5, 0.5, 1, 1, 1);
```

Output

```
POINT Z (1.5 1.5 1.5)
```

Figure



Se även

CG_Scale, CG_3DScale

8.3.44 CG_Translate

CG_Translate — Translaterar (flyttar) en geometri med hjälp av givna offsets i 2D-rymden.

Synopsis

geometry **CG_Translate**(geometry geom, float8 deltaX, float8 deltaY);

Beskrivning

Translaterar indatageometrin *geom* genom att lägga till *deltaX* till X-koordinaterna och *deltaY* till Y-koordinaterna. Z-koordinater utelämnas.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Exempel

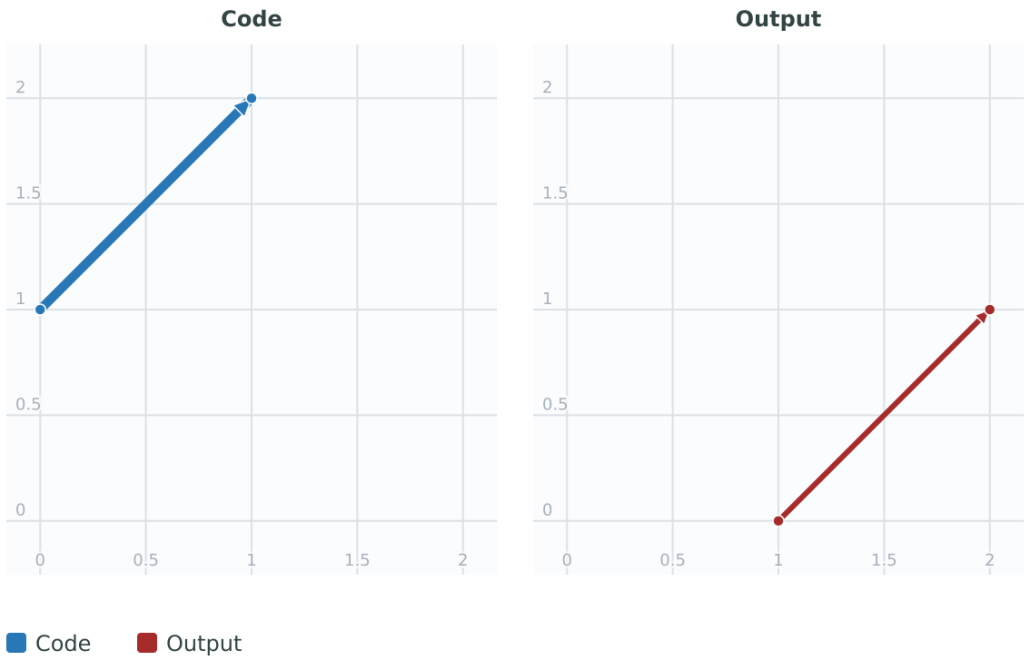
Code

```
SELECT CG_Translate('LINESTRING(1 1,2 2)', 1, -1);
```

Output

```
LINESTRING(2 0,3 1)
```

Figure



Se även

CG_3DTranslate, ST_Translate

8.3.45 CG_3DTranslate

CG_3DTranslate — Translaterar (flyttar) en geometri med hjälp av givna offsets i 3D-rymden.

Synopsis

geometry **CG_3DTranslate**(geometry geom, float8 deltaX, float8 deltaY, float8 deltaZ);

Beskrivning

Translaterar indatageometrin *geom* genom att lägga till *deltaX* till X-koordinaterna, *deltaY* till Y-koordinaterna och *deltaZ* till Z-koordinaterna.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Code

```
SELECT CG_3DTranslate('POINT(1 1 1)', 1, -1, 2);
```

Output

```
POINT Z (2 0 3)
```

Figure

Se även

CG_Translate, ST_Translate

8.3.46 CG_Simplify

CG_Simplify — Minskar komplexiteten i en geometri samtidigt som viktiga egenskaper och Z/M-värden bevaras.

Synopsis

geometry **CG_Simplify**(geometry geom, double precision threshold, boolean preserveTopology = false);

Beskrivning

Förenklar en geometri med hjälp av SFCGAL:s förenklingsalgoritm, som minskar antalet punkter eller hörn samtidigt som geometrins väsentliga egenskaper bevaras. Denna funktion bevarar Z- och M-värden under förenklingen.

Algoritmen är baserad på begränsad triangulering och använder CGAL Polyline Simplification 2-biblioteket med ytterligare hantering för att bevara Z- och M-koordinater. När topologin bevaras och geometrierna korsar varandra interpoleras Z- och M-värdena vid skärningspunkterna.

Denna funktion fungerar bra med 3D-terrängliknande geometrier (2,5D) men är inte utformad för vertikala ytor som väggar.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.1.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Denna funktion stöder M-koordinater.

Parametrar

geom Geometri för inmatning

tröskelvärde Tröskelvärde för maximalt avstånd (i geometrisk enhet) för förenkling. Ju högre detta värde är, desto mer förenklad blir den resulterande geometrin.

bevaraTopologi Om funktionen är inställd på true säkerställer den att geometrins topologi bevaras. När geometrier korsas i detta läge interpoleras Z- och M-värdena vid skärningspunkterna. Standardvärdet är false.

Returvärde

Returnerar en förenklad geometri med bevarade Z- och M-värden.

Exempel

This example simplifies a polygon with a threshold of 0.5.

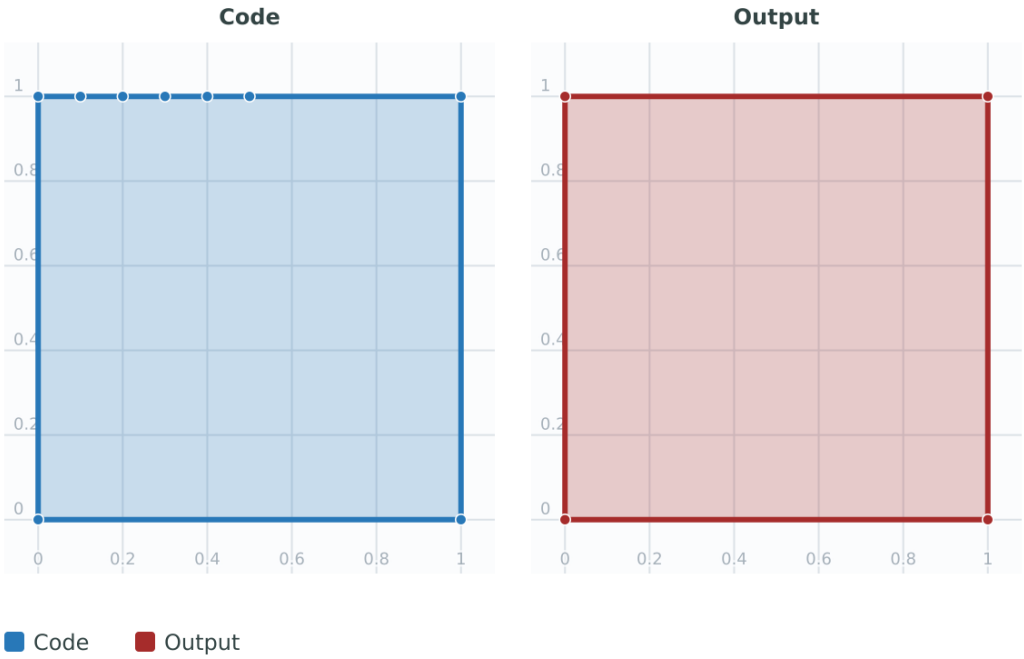
Code

```
SELECT CG_Simplify(
    'POLYGON((0 0,0 1,0.1 1,0.2 1,0.3 1,0.4 1,0.5 1,1 1,1 0,0 0))',
    0.5
  );
```

Output

```
POLYGON((0 0,0 1,1 1,0 0))
```

Figure



Preserving topology can keep additional support points in a mixed ZM collection.

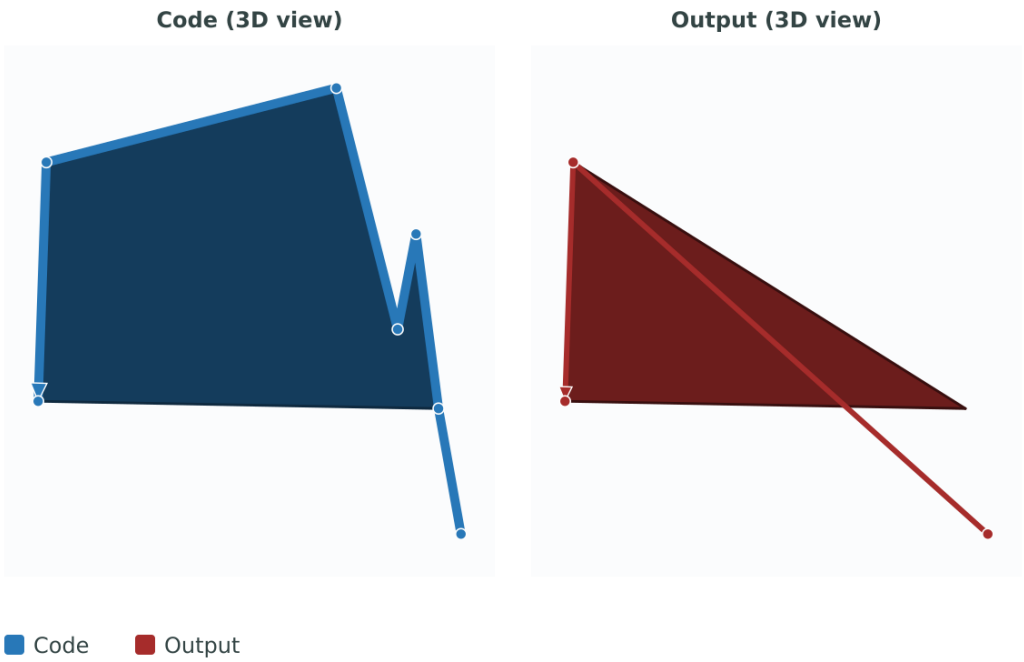
Code

```
SELECT CG_Simplify(
    'GEOMETRYCOLLECTION ZM(
      LINESTRING ZM(-1 -1 3 4,0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 ↩
     15 150),
      POLYGON ZM((0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 15 150,0 0 ↩
    10 100))
    )',
    2,
    false
  );
```

Output

```
GEOMETRYCOLLECTION ZM (LINESTRING ZM (-1 -1 3 4,2 19 25 250,-4 20 15 150),POLYGON ZM ((0 0 10 100,2 19 25 250,-4 20 15 ↩
     150,0 0 10 100)))
```

Figure



The topology-preserving variant keeps the shared anchor point at the collection intersection.

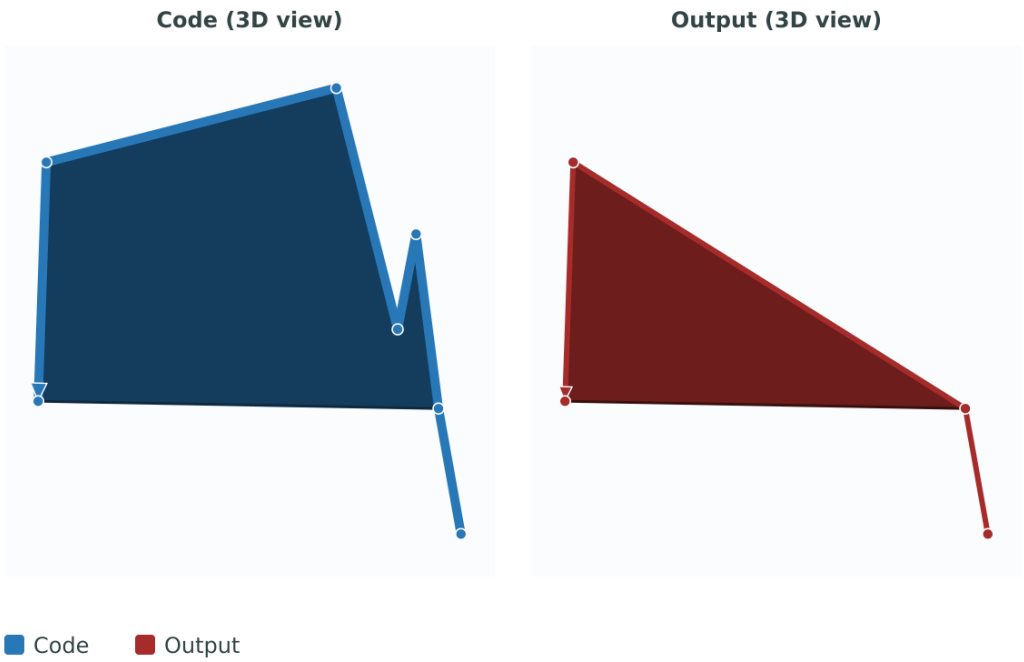
Code

```
SELECT CG_Simplify(
    'GEOMETRYCOLLECTION ZM(
      LINESTRING ZM(-1 -1 3 4,0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 ↩
     15 150),
      POLYGON ZM((0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 15 150,0 0  ↩
    10 100))
    )',
    2,
    true
  );
```

Output

```
GEOMETRYCOLLECTION ZM (LINESTRING ZM (-1 -1 3 4,0 0 10 100,2 19 25 250,-4 20 15 150),POLYGON ZM ((0 0 10 100,2 19 25  ↩
     250,-4 20 15 150,0 0 10 100)))
```

Figure

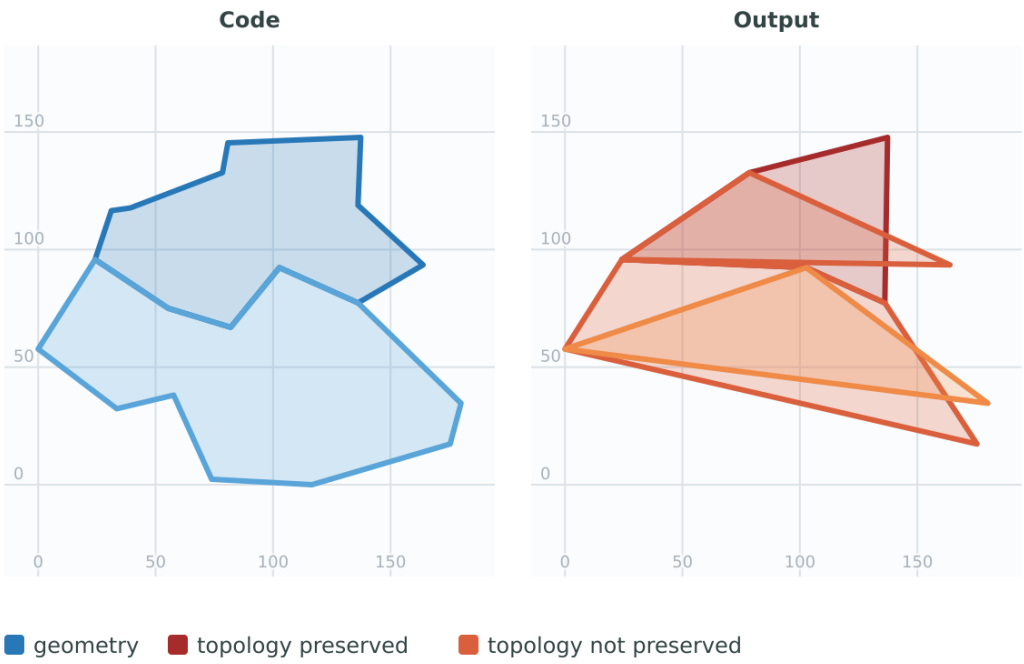


Compare simplification with and without topology preservation, using a threshold of 50.

Code

```
WITH data AS (
  SELECT ST_GeomFromText('GEOMETRYCOLLECTION(
    POLYGON((88.46 158.85,90.77 171.54,147.31 173.85,146.15 145,173.85 119.62,146.15 ↩
    103.46,112.69 118.46,91.92 93.08,65.38 101.15,34.23 121.92,41.15 142.69,49.23 ↩
    143.85,88.46 158.85)),
    POLYGON((112.69 118.46,146.15 103.46,190 60.77,185.38 43.46,126.54 26.15,83.85 ↩
    28.46,67.69 64.23,43.46 58.46,10 83.85,34.23 121.92,65.38 101.15,91.92 93.08,112.69 ↩
    118.46)))
  ') AS geom
)
SELECT geom AS input_geometry,
       CG_Simplify(geom, 50, true) AS topology_preserved,
       CG_Simplify(geom, 50, false) AS topology_not_preserved
FROM data;
```

Figure



Se även

ST_Simplify, ST_SimplifyPreserveTopology

8.3.47 CG_3DAlphaWrapping

CG_3DAlphaWrapping — Beräknar en 3D Alpha-wrapping som strikt omsluter en geometri.

Synopsis

geometry **CG_3DAlphaWrapping**(geometry geom, integer relative_alpha, integer relative_offset);

Beskrivning

Beräknar 3D-alfaomslaget för punkterna i en geometri. En alfaförpackning är ett vattentätt och orienterbart ytnät som strikt omsluter indata. Det kan ses som en utvidgning eller förfining av en alfa-form.

The `relative_alpha` parameter controls which features will appear in the output. It can have values from 0 to infinity. Higher relative_alpha values produce coarser wraps; lower values retain more detail.

The `relative_offset` parameter controls the tightness of the result. It can have values from 0 to infinity. If this parameter is set to 0, its value is automatically determined based on the relative_alpha parameter.

Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.1.0

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

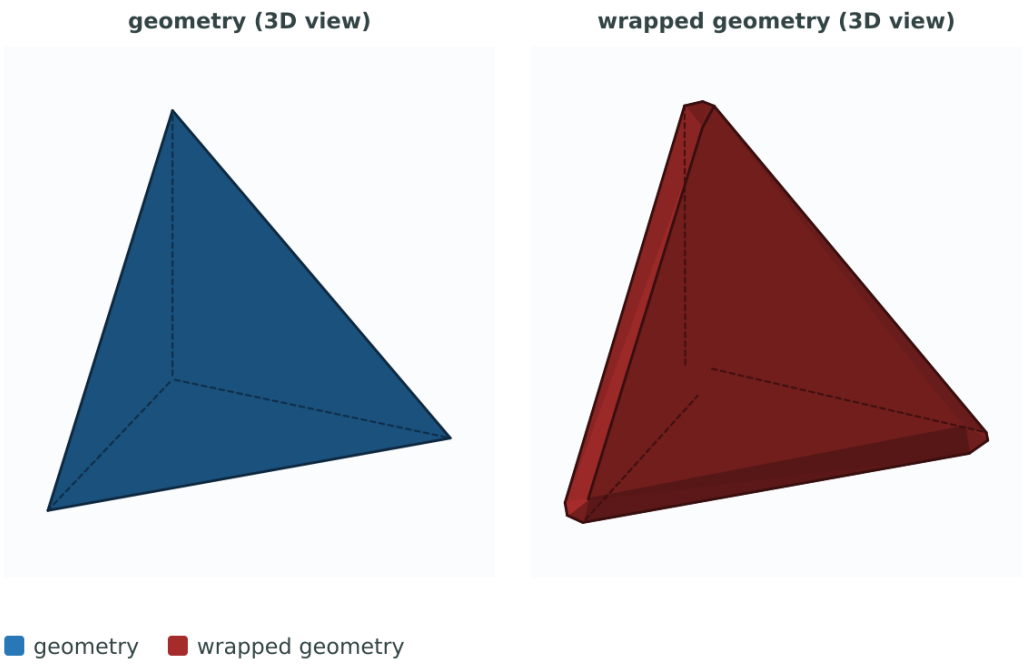
Exempel

Wrap a tetrahedral surface and show the native 3D input beside the actual enclosing mesh.

Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z (
    ((0 0 0,4 0 0,0 4 0,0 0 0)),
    ((0 0 0,0 0 4,4 0 0,0 0 0)),
    ((0 0 0,0 4 0,0 0 4,0 0 0)),
    ((4 0 0,0 0 4,0 4 0,4 0 0))
  )'::geometry AS input_geometry
)
SELECT input_geometry AS input_geometry,
       CG_3DAlphaWrapping(input_geometry, 1)
         AS wrapped_geometry
FROM data;
```

Figure



Se även

CG_AlphaShape

8.3.48 CG_NurbsCurveInterpolate

CG_NurbsCurveInterpolate — Creates an interpolating NURBS curve passing through all given data points

Synopsis

geometry **CG_NurbsCurveInterpolate**(geometry data_points, integer degree);

Beskrivning

Creates a NURBS curve that interpolates through all provided data points. The resulting curve passes exactly through each point in the input geometry.

Parameters:

- `data_points` - A LINESTRING geometry containing the points to interpolate through.
- `degree` - Polynomial degree of the NURBS curve (typically 2 or 3).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Interpolate a degree-3 NURBS curve through four 2D points.

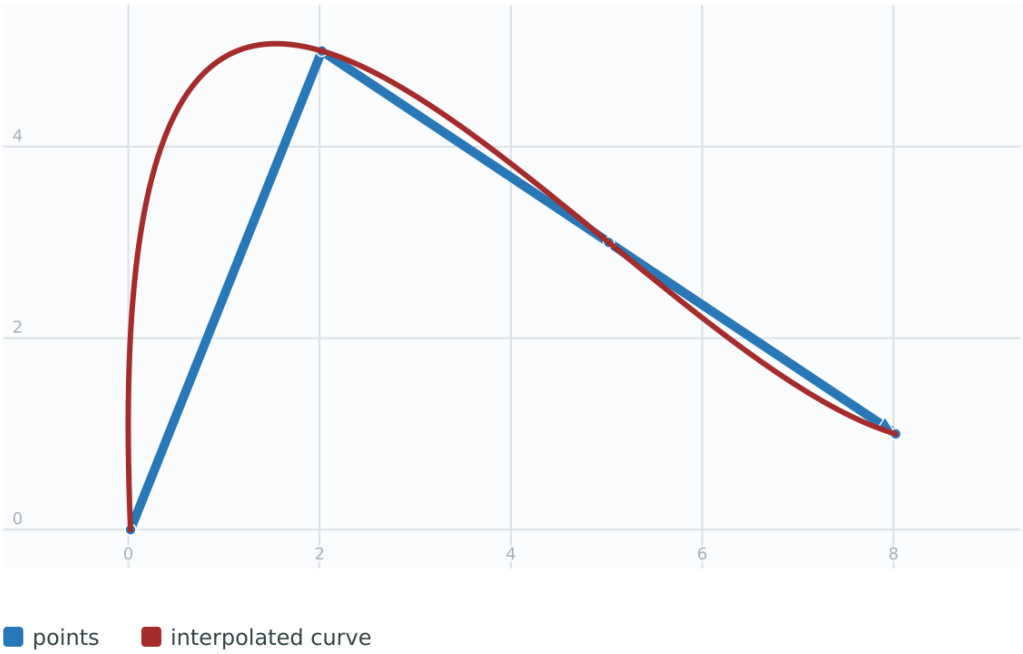
Code

```
WITH data AS (
  SELECT 'LINESTRING(0 0,2 5,5 3,8 1)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveInterpolate(input_points, 3) AS interpolated_curve
FROM data;
```

Output

```
LINESTRING(0 0,2 5,5 3,8 1) | NURBSCURVE(DEGREE 3, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT( ↩
    WEIGHTEDPOINT(-0.43 10.29),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(4.95 1.9),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(8 1) ↩
    ,WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(1,4)))
```

Figure



Interpolate a 3D degree-3 NURBS curve through the documented control points.

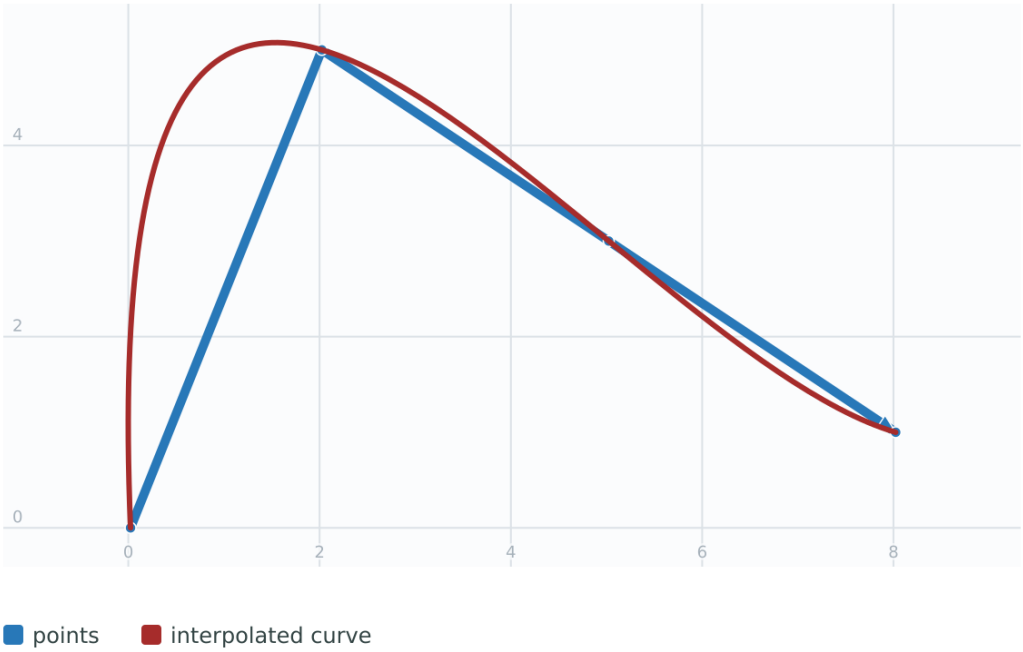
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (0 0 0,2 5 2,5 3 1,8 1 0)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveInterpolate(input_points, 3) AS interpolated_curve
FROM data;
```

Output

```
LINESTRING Z (0 0 0,2 5 2,5 3 1,8 1 0) | NURBSCURVE Z (DEGREE 3,CONTROLPOINTS Z (NURBSPOINT(WEIGHTEDPOINT Z (0 0 0), ↩
    WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (-0.43 10.29 4.36),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (4.95 1.9 0.54), ↩
    WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (8 1 0),WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(1,4)))
```

Figure



Se även

CG_NurbsCurveApproximate, ST_MakeNurbsCurve

8.3.49 CG_NurbsCurveApproximate

CG_NurbsCurveApproximate — Creates an approximating NURBS curve fitting data points within a tolerance

Synopsis

geometry **CG_NurbsCurveApproximate**(geometry data_points, integer degree, float8 tolerance, integer max_control_points=100);

Beskrivning

Creates a NURBS curve that approximates the given data points within the specified tolerance. Unlike interpolation, the curve does not necessarily pass through all points but provides a smooth fit with fewer control points.

Parameters:

- `data_points` - A LINESTRING geometry containing the points to approximate.
- `degree` - Polynomial degree of the NURBS curve.
- `tolerance` - Maximum allowed distance between the curve and data points.
- `max_control_points` - Optional maximum number of control points (default: 100). Limits curve complexity.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

Approximate noisy 2D data with a degree-2 curve and a tolerance of 0.5.

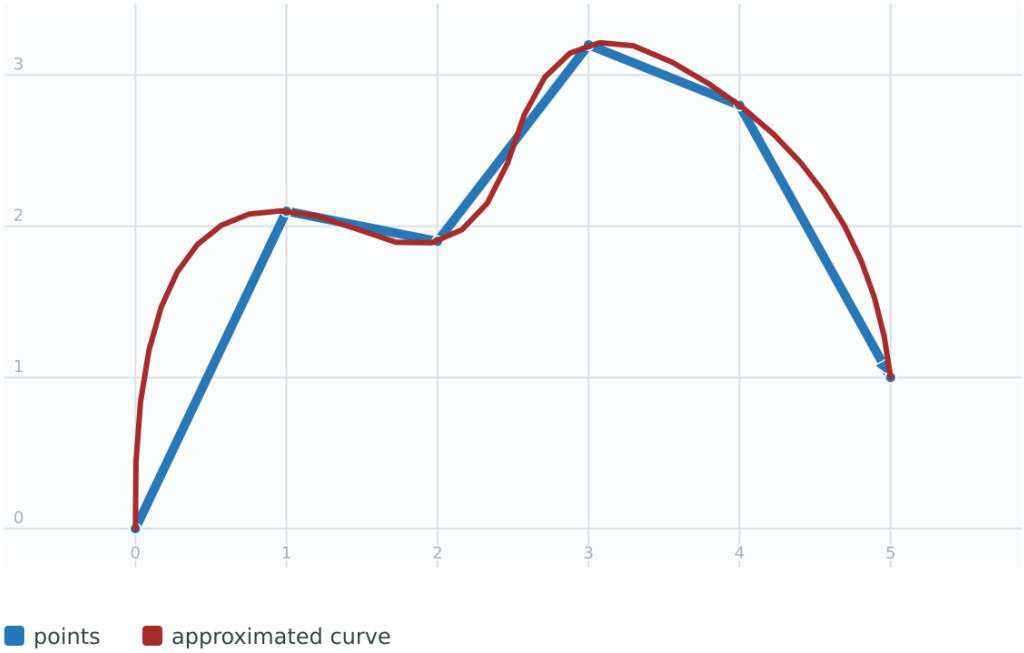
Code

```
WITH data AS (
  SELECT 'LINESTRING(0 0,1 2.1,2 1.9,3 3.2,4 2.8,5 1)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveApproximate(input_points, 2, 0.5) AS approximated_curve
FROM data;
```

Output

```
LINESTRING(0 0,1 2.1,2 1.9,3 3.2,4 2.8,5 1) | NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT ↩
    1), NURBSPOINT(WEIGHTEDPOINT(-0.05 2.64),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(2.24 1.65),WEIGHT 1), NURBSPOINT( ↩
    WEIGHTEDPOINT(2.79 3.48),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(4.84 2.43),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 1), ↩
    WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(0.35,1), KNOT(0.51,1), KNOT(0.68,1), KNOT(1,3)))
```

Figure



Approximate a 3D point set while preserving the Z dimension in the fitted curve.

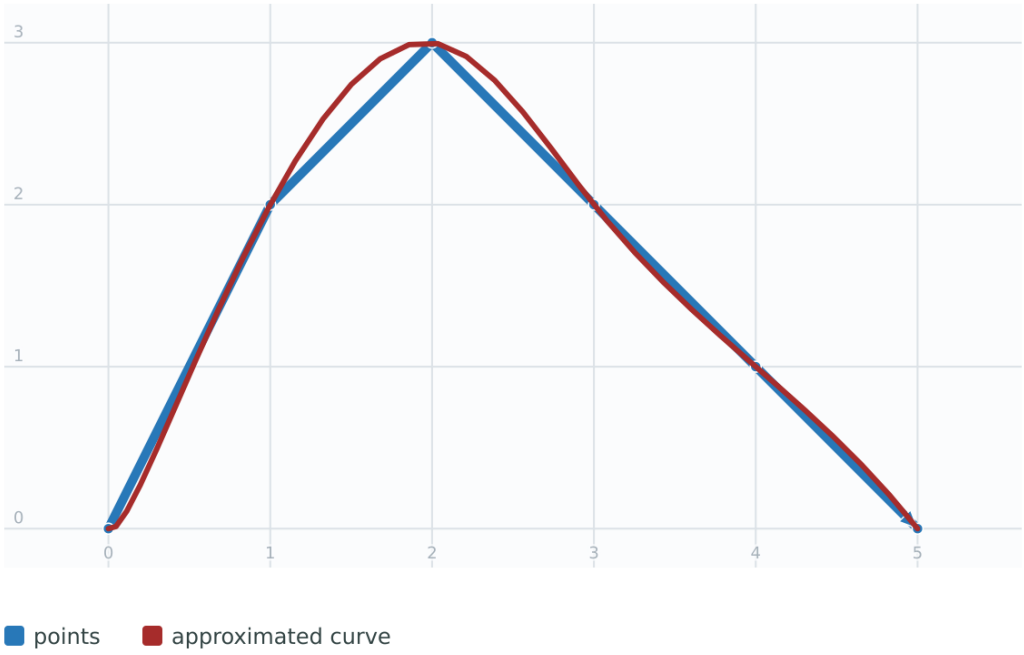
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (0 0 0,1 2 1,2 3 2,3 2 1,4 1 0,5 0 0)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveApproximate(input_points, 3, 0.1) AS approximated_curve
FROM data;
```

Output

```
LINESTRING Z (0 0 0,1 2 1,2 3 2,3 2 1,4 1 0,5 0 0) | NURBSCURVE Z (DEGREE 3,CONTROLPOINTS Z (NURBSPOINT(WEIGHTEDPOINT ↩
    Z (0 0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (0.18 -0.15 -1.29),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (1.49 ↩
    4.36 3.35),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (3.33 1.33 0.38),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (4.34 ↩
    0.83 -0.44),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (5 0 0),WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(0.46,1), KNOT(0.64,1) ↩
    , KNOT(1,4)))
```

Figure



Se även

CG_NurbsCurveInterpolate, ST_MakeNurbsCurve

8.3.50 CG_NurbsCurveDerivative

CG_NurbsCurveDerivative — Computes the derivative (tangent, curvature) of a NURBS curve at a given parameter

Synopsis

geometry **CG_NurbsCurveDerivative**(geometry nurbscurve, float8 parameter, integer derivative_order);

Beskrivning

Computes the derivative of a NURBS curve at a specific parameter value. The derivative represents the rate of change at that point:

- Order 1: Tangent vector (direction and speed)
- Order 2: Curvature vector (how the tangent changes)
- Order 3: Third-order derivative (supported maximum)

Parameters:

- `nurbscurve` - A NURBS curve geometry.
- `parameter` - Parameter value for evaluation (typically 0.0 to 1.0).

- `derivative_order` - Order of derivative to compute (1, 2, or 3).

Returns a POINT geometry representing the derivative vector at the specified parameter.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

Denna metod behöver SFCGAL-backend.

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

The first derivative gives the tangent vector at the evaluation parameter.

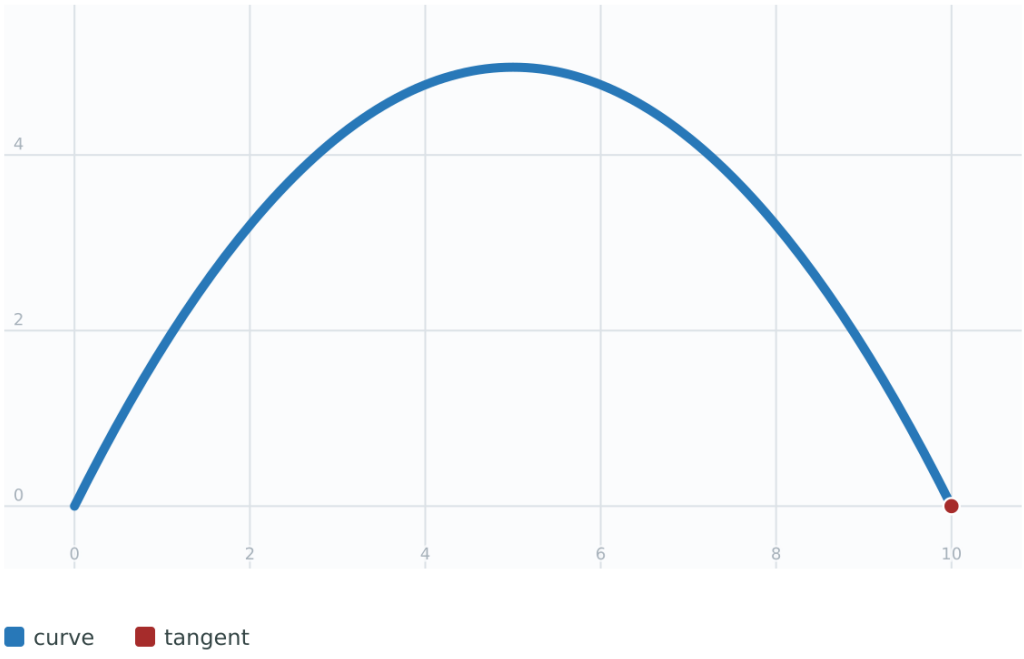
Code

```
WITH data AS (
  SELECT 'NURBSCURVE (2, (0.00 0.00, 5.00 10.00, 10.00 0.00))'::geometry AS input_curve
)
SELECT input_curve AS input_curve,
       CG_NurbsCurveDerivative(input_curve, 0.5, 1) AS tangent
FROM data;
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 10),WEIGHT 1),  ↩
    NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3))) | POINT Z (10 0 0)
```

Figure



The second derivative exposes curvature change for a higher-degree curve.

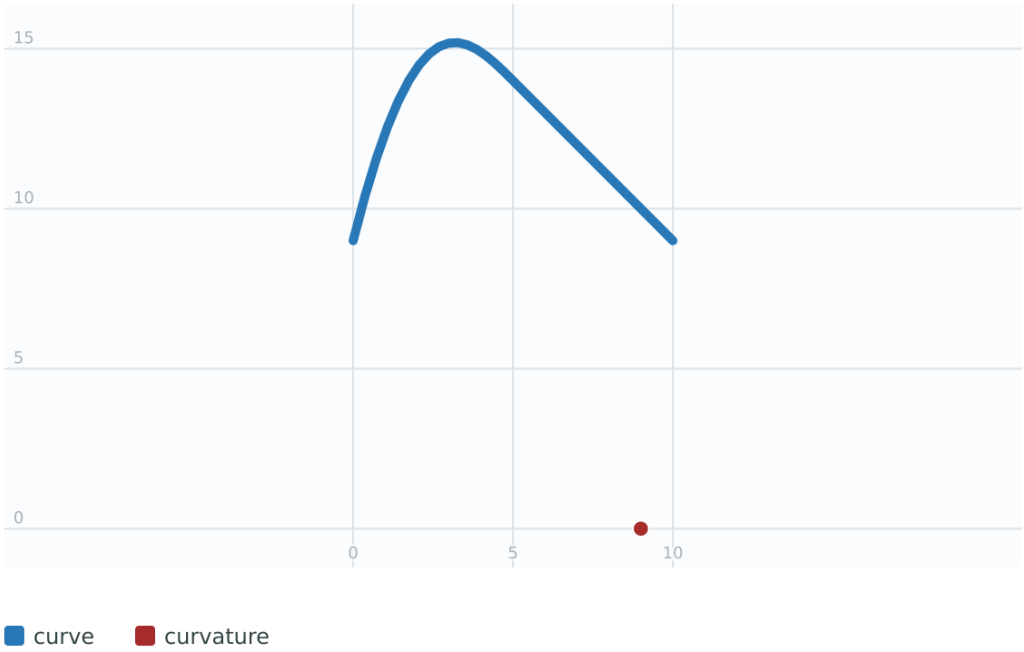
Code

```
WITH data AS (
  SELECT 'NURBSCURVE (3, (0.00 0.00, 2.00 8.00, 5.00 5.00, 8.00 2.00, 10.00 0.00))':: ↩
    geometry AS input_curve
)
SELECT input_curve AS input_curve,
       CG_NurbsCurveDerivative(input_curve, 0.5, 2) AS curvature
FROM data;
```

Output

```
NURBSCURVE(DEGREE 3, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(2 8),WEIGHT 1),  ↩
    NURBSPOINT(WEIGHTEDPOINT(5 5),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(8 2),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(10 0), ↩
    WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(0.5,1), KNOT(1,4))) | POINT Z (9 -9 0)
```

Figure



The same derivative call also works for 3D NURBS curves.

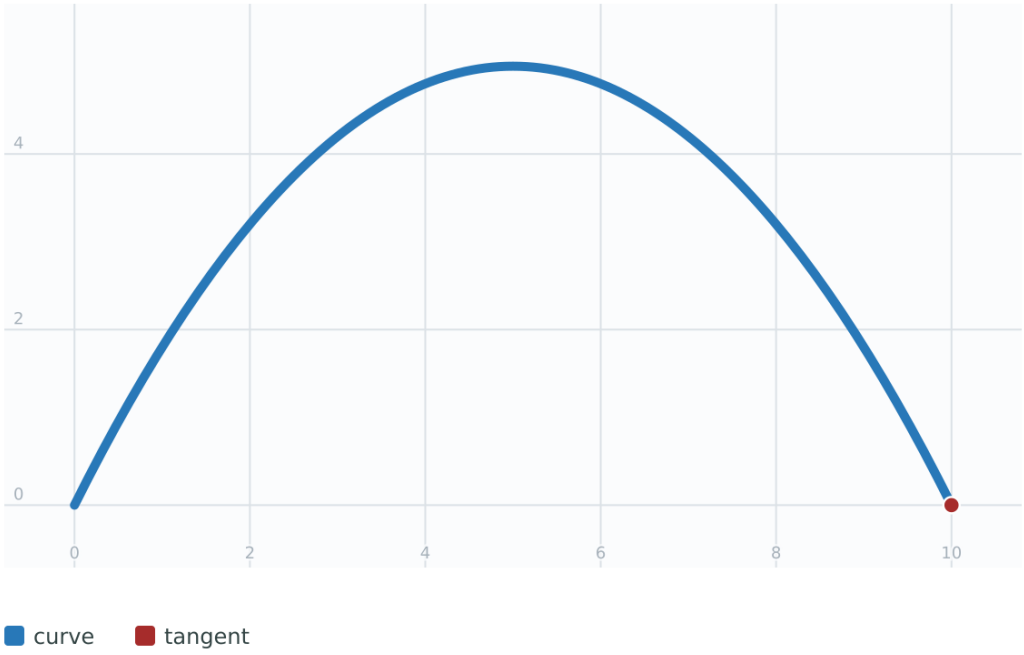
Code

```
WITH data AS (
  SELECT 'NURBSCURVE Z (2, (0.00 0.00 0.00, 5.00 10.00 5.00, 10.00 0.00 0.00))'::geometry  ↩
    AS input_curve
)
SELECT input_curve AS input_curve,
       CG_NurbsCurveDerivative(input_curve, 0.5, 1) AS tangent
FROM data;
```

Output

```
NURBSCURVE Z (DEGREE 2,CONTROLPOINTS Z (NURBSPOINT(WEIGHTEDPOINT Z (0 0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (5 10 ↩
     5),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (10 0 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3))) | POINT Z (10 0 0)
```

Figure



Se även

ST_Evaluate, ST_MakeNurbsCurve

8.4 Deprecated SFCGAL Function Aliases

These ST_ names remain available for compatibility and emit deprecation notices. Their entries contain only the legacy signature and migration link. Use the corresponding CG_ pages for behavior, examples, and geometry support details.

8.4.1 ST_3DIntersection

ST_3DIntersection — Deprecated alias for `CG_3DIntersection`.

Synopsis

geometry **ST_3DIntersection**(geometry geom1, geometry geom2);

Description

Varning
`ST_3DIntersection` is deprecated as of 3.5.0. Use CG_3DIntersection instead.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.2 ST_3DDifference

ST_3DDifference — Deprecated alias for `CG_3DDifference`.

Synopsis

geometry **ST_3DDifference**(geometry geom1, geometry geom2);

Description

Varning
`ST_3DDifference` is deprecated as of 3.5.0. Use CG_3DDifference instead.

Availability: 2.2.0

Denna metod behöver SFCGAL-backend.

8.4.3 ST_3DUnion

ST_3DUnion — Deprecated alias for `CG_3DUnion`.

Synopsis

geometry **ST_3DUnion**(geometry geom1, geometry geom2);
geometry **ST_3DUnion**(geometry set g1field);

Description

Varning
`ST_3DUnion` is deprecated as of 3.5.0. Use CG_3DUnion instead.

Availability: 2.2.0

Denna metod behöver SFCGAL-backend.

8.4.4 ST_Tesselate

ST_Tesselate — Deprecated alias for `CG_Tesselate`.

Synopsis

geometry **ST_Tesselate**(geometry geom);

Description

Varning
`ST_Tesselate` is deprecated as of 3.5.0. Use CG_Tesselate instead.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.5 ST_3DArea

ST_3DArea — Deprecated alias for `CG_3DArea`.

Synopsis

float**ST_3DArea**(geometry geom1);

Description

Varning
`ST_3DArea` is deprecated as of 3.5.0. Use CG_3DArea instead.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.6 ST_Extrude

ST_Extrude — Deprecated alias for `CG_Extrude`.

Synopsis

geometry **ST_Extrude**(geometry geom, float x, float y, float z);

Description

Varning
ST_Extrude is deprecated as of 3.5.0. Use CG_Extrude instead.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.7 ST_ForceLHR

ST_ForceLHR — Deprecated alias for CG_ForceLHR.

Synopsis

geometry **ST_ForceLHR**(geometry geom);

Description

Varning
ST_ForceLHR is deprecated as of 3.5.0. Use CG_ForceLHR instead.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.8 ST_Orientation

ST_Orientation — Deprecated alias for CG_Orientation.

Synopsis

integer **ST_Orientation**(geometry geom);

Description

Varning
ST_Orientation is deprecated as of 3.5.0. Use CG_Orientation instead.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.9 ST_MinkowskiSum

ST_MinkowskiSum — Deprecated alias for `CG_MinkowskiSum`.

Synopsis

geometry **ST_MinkowskiSum**(geometry geom1, geometry geom2);

Description

Varning
`ST_MinkowskiSum` is deprecated as of 3.5.0. Use CG_MinkowskiSum instead.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.10 ST_StraightSkeleton

ST_StraightSkeleton — Deprecated alias for `CG_StraightSkeleton`.

Synopsis

geometry **ST_StraightSkeleton**(geometry geom);

Description

Varning
`ST_StraightSkeleton` is deprecated as of 3.1.0. Use CG_StraightSkeleton instead.

Notera
Like `CG_StraightSkeleton`, this function ignores the Z dimension and always gives a 2D result, even when used on a 3D geometry.

Availability: 2.1.0

Denna metod behöver SFCGAL-backend.

8.4.11 ST_ApproximateMedialAxis

ST_ApproximateMedialAxis — Deprecated alias for `CG_ApproximateMedialAxis`.

Synopsis

geometry **ST_ApproximateMedialAxis**(geometry geom);

Description

Varning
`ST_ApproximateMedialAxis` is deprecated as of 3.5.0. Use CG_ApproximateMedialAxis instead.

Notera
Like `CG_ApproximateMedialAxis`, this function ignores the Z dimension and always gives a 2D result, even when used on a 3D geometry.

Availability: 2.2.0

Denna metod behöver SFCGAL-backend.

8.4.12 ST_IsPlanar

ST_IsPlanar — Deprecated alias for `CG_IsPlanar`.

Synopsis

boolean **ST_IsPlanar**(geometry geom);

Description

Varning
`ST_IsPlanar` is deprecated as of 3.5.0. Use CG_IsPlanar instead.

Availability: 2.2.0: This was documented in 2.1.0 but got accidentally left out in 2.1 release.

Denna metod behöver SFCGAL-backend.

8.4.13 ST_Volume

ST_Volume — Deprecated alias for `CG_Volume`.

Synopsis

float **ST_Volume**(geometry geom1);

Description

Varning
`ST_Volume` is deprecated as of 3.5.0. Use CG_Volume instead.

Availability: 2.2.0

Denna metod behöver SFCGAL-backend.

8.4.14 ST_MakeSolid

ST_MakeSolid — Deprecated alias for `CG_MakeSolid`.

Synopsis

geometry **ST_MakeSolid**(geometry geom1);

Description

Varning
`ST_MakeSolid` is deprecated as of 3.5.0. Use CG_MakeSolid instead.

Availability: 2.2.0

Denna metod behöver SFCGAL-backend.

8.4.15 ST_IsSolid

ST_IsSolid — Deprecated alias for `CG_IsSolid`.

Synopsis

boolean **ST_IsSolid**(geometry geom1);

Description

Varning
`ST_IsSolid` is deprecated as of 3.5.0. Use CG_IsSolid instead.

Availability: 2.2.0

Denna metod behöver SFCGAL-backend.

8.4.16 ST_ConstrainedDelaunayTriangles

ST_ConstrainedDelaunayTriangles — Deprecated alias for `CG_ConstrainedDelaunayTriangles`.

Synopsis

geometry **ST_ConstrainedDelaunayTriangles**(geometry g1);

Description

Varning
`ST_ConstrainedDelaunayTriangles` is deprecated as of 3.5.0. Use CG_ConstrainedDelaunayTriangles instead.

Availability: 3.0.0

Denna metod behöver SFCGAL-backend.

8.4.17 ST_3DConvexHull

ST_3DConvexHull — Deprecated alias for `CG_3DConvexHull`.

Synopsis

geometry **ST_3DConvexHull**(geometry geom1);

Description

Varning
`ST_3DConvexHull` is deprecated as of 3.5.0. Use CG_3DConvexHull instead.

Availability: 3.3.0

Denna metod behöver SFCGAL-backend.

8.4.18 ST_AlphaShape

ST_AlphaShape — Deprecated alias for `CG_AlphaShape`.

Synopsis

geometry **ST_AlphaShape**(geometry geom, float alpha , boolean allow_holes = false);

Description

Varning
ST_AlphaShape is deprecated as of 3.5.0. Use CG_AlphaShape instead.

Denna metod behöver SFCGAL-backend.

8.4.19 ST_OptimalAlphaShape

ST_OptimalAlphaShape — Deprecated alias for `CG_OptimalAlphaShape`.

Synopsis

geometry **ST_OptimalAlphaShape**(geometry geom, boolean allow_holes = false , integer nb_components = 1);

Description

Varning
ST_OptimalAlphaShape is deprecated as of 3.5.0. Use CG_OptimalAlphaShape instead.

Availability: 3.3.0 - requires SFCGAL >= 1.4.1.

Denna metod behöver SFCGAL-backend.

Kapitel 9

Topologi

PostGIS Topology-typer och -funktioner används för att hantera topologiska objekt som ytor, kanter och noder.

Sandro Santillis presentation på konferensen PostGIS Day Paris 2011 ger en bra sammanfattning av PostGIS Topology och vart det är på väg Topology with PostGIS 2.0 slide deck..

Vincent Picavet ger en bra sammanfattning och översikt över vad Topology är, hur det används och olika FOSS4G-verktyg som stöder det i PostGIS Topology PGConf EU 2012.

An example of a topologically based GIS database is the US Census Topologically Integrated Geographic Encoding and Referencing System (TIGER) database. If you want to experiment with PostGIS topology and need some data.

The PostGIS topology module has existed for a long time but was not always part of the official documentation. Extensive cleanup removed deprecated functions, fixed known usability issues, documented the features and functions, added new functionality, and improved SQL-MM compliance.

This chapter is the maintained reference for the topology module, including its primitive tables, TopoGeometry type, topology management functions, editing functions, validation functions, and conversion helpers.

Alla funktioner och tabeller som är kopplade till denna modul är installerade i ett schema som kallas `topologi`.

Funktioner som är definierade i SQL/MM-standarden har prefixet ST_ och funktioner som är specifika för PostGIS har inget prefix.

Topology support is built by default and can be disabled by specifying the --without-topology configure option at build time as described in Kapitel 2

9.1 Building Faces from Linework

This example builds topology faces from noded linework and associates point attributes with the resulting faces. It uses ST_CreateTopoGeo to populate an empty topology, ValidateTopology to check the result, GetFaceByPoint to locate each attribute point, and ST_GetFaceGeometry to materialize ordinary polygon geometries.

Create an empty topology and populate it with linework that encloses two faces:

Code

```
CREATE EXTENSION IF NOT EXISTS postgis_topology;

SELECT topology.CreateTopology('topo_example', 3857);

WITH lines(geom) AS (VALUES
    ('SRID=3857;LINESTRING(0 0, 2 0)'::geometry),
    ('SRID=3857;LINESTRING(0 1, 2 1)'::geometry),
    ('SRID=3857;LINESTRING(0 0, 0 1)'::geometry),
    ('SRID=3857;LINESTRING(1 0, 1 1)'::geometry),
    ('SRID=3857;LINESTRING(2 0, 2 1)'::geometry)
)
SELECT topology.ST_CreateTopoGeo(
    'topo_example',
    ST_Collect(geom)
)
FROM lines;

SELECT * FROM topology.ValidateTopology('topo_example');
```

An empty result from `ValidateTopology` means that no topology errors were found. Attribute points can now be matched to faces. A face identifier of `0` denotes the universal face, so the point is outside every bounded face.

Code

```
WITH labels(name, geom) AS (VALUES
    ('west', 'SRID=3857;POINT(0.5 0.5)'::geometry),
    ('east', 'SRID=3857;POINT(1.5 0.5)'::geometry),
    ('outside', 'SRID=3857;POINT(3 0.5)'::geometry)
),
located AS (
    SELECT name,
           topology.GetFaceByPoint('topo_example', geom, 0) AS face_id
    FROM labels
)
SELECT name,
       face_id,
       CASE WHEN face_id = 0 THEN NULL
            ELSE topology.ST_GetFaceGeometry('topo_example', face_id)
       END AS geom
FROM located
ORDER BY name;
```

Before copying attributes into a face table, check both for points outside bounded faces and for multiple source points assigned to the same face:

Code

```
WITH labels(name, geom) AS (VALUES
    ('west', 'SRID=3857;POINT(0.5 0.5)'::geometry),
    ('east', 'SRID=3857;POINT(1.5 0.5)'::geometry),
    ('outside', 'SRID=3857;POINT(3 0.5)'::geometry)
),
located AS (
    SELECT name,
           topology.GetFaceByPoint('topo_example', geom, 0) AS face_id
    FROM labels
)
SELECT 'outside face' AS problem,
       array_agg(name ORDER BY name) AS labels
```

```
FROM located
WHERE face_id = 0
HAVING count(*) > 0
UNION ALL
SELECT 'duplicate face ' || face_id,
       array_agg(name ORDER BY name)
FROM located
WHERE face_id != 0
GROUP BY face_id
HAVING count(*) > 1;
```

Use DropTopology to remove the example topology when it is no longer needed:

Code

```
SELECT topology.DropTopology('topo_example');
```

9.2 Topology Primitive Tables

The core primitives of any topology are stored in the `edge_data`, `node`, and `face` tables that live in the schema created by CreateTopology. Each row of `edge_data` represents an oriented edge: it records a directed curve from `start_node` to `end_node` together with the identifier of the face encountered on the left of that direction (`left_face`) and the face encountered on the right (`right_face`). The same geometric segment may therefore appear twice—once for each orientation—when it belongs to two faces.

The `next_left_edge` and `next_right_edge` columns complete this orientation information by encoding how to keep walking around a face. They store signed integers whose absolute value is the identifier of the next oriented edge and whose sign determines whether the stored orientation has to be followed as-is or reversed when traversing. Formally, the following rules hold for every edge `e`:

- `abs(next_left_edge)` is the identifier of the edge reached by continuing around the face that lies to the left of `e`. If the value is positive the walk continues from the end node of `e` along the stored orientation of the referenced edge; if it is negative the referenced edge must be followed backwards so that the shared face remains on the walker's left.
- `abs(next_right_edge)` analogously follows the boundary of the face located on the right of `e`. A positive value means that the next edge is taken with its recorded orientation starting from the end node of `e`, whereas a negative value instructs to traverse the referenced edge in reverse, starting from its end node, so that the right-hand face is preserved.
- A zero value indicates that the edge is dangling on the corresponding side (for example an isolated edge whose incident face is the universal face `0`). The `abs_next_left_edge` and `abs_next_right_edge` columns exposed by the `edge` view are convenience projections of these absolute values.

This representation is a variant of a doubly connected edge list and is exploited by many topology routines. Functions such as GetRingEdges and ValidateTopology rely on it to reconstruct face boundaries and to diagnose inconsistencies—hence the "invalid next_left_edge" and "invalid next_right_edge" diagnostics reported during validation. Constructors like AddEdge initialise the `next_*` attributes with trivial self references, while editing routines including ST_AddEdgeModFace and ST_RemEdgeModFace update the links as edges are inserted or removed. Other bulk operations (for example Polygonize) may intentionally leave the fields unset, which is why the documentation flags their behaviour explicitly.

9.3 Topologityper

9.3.1 getfaceedges_returntype

getfaceedges_returntype — En sammansatt typ som består av ett sekvensnummer och ett kantnummer.

Description

En sammansatt typ som består av ett sekvensnummer och ett kantnummer. Detta är returtypen för funktionerna `ST_GetFaceEdges` och `GetNodeEdges`.

1. `sequence` är ett heltal: Hänvisar till en topologi som definieras i tabellen topology.topology som definierar topologins schema och srid.
2. `edge` är ett heltal: Identifieraren för en kant.

9.3.2 TopoGeometry

TopoGeometry — En sammansatt typ som representerar en topologiskt definierad geometri.

Description

En sammansatt typ som hänvisar till en topologisk geometri i ett specifikt topologiskt lager, med en specifik typ och ett specifikt id. Elementen i en TopoGeometry är egenskaperna: topology_id, layer_id, id integer, type integer.

1. `topology_id` är ett heltal: Hänvisar till en topologi som definieras i tabellen topology.topology som definierar topologins schema och srid.
2. `layer_id` är ett heltal: Det layer_id i layer-tabellen som TopoGeometry hör till. Kombinationen av topology_id och layer_id ger en unik referens i tabellen topology.layers.
3. `id` är ett heltal: Id är det autogenererade sekvensnumret som unikt definierar topogeometrin i respektive topologilager.
4. `type` heltal mellan 1 - 4 som definierar geometritypen: 1:[multi]punkt, 2:[multi]linje, 3:[multi]poly, 4:samling

Casting-beteende

I detta avsnitt listas de automatiska såväl som explicita kast som är tillåtna för denna datatyp

Kasta till	Beteende
geometri	automatisk

Se även

CreateTopoGeom

9.3.3 validatetopology_returntype

validatetopology_returntype — En sammansatt typ som består av ett felmeddelande samt id1 och id2 för att ange var felet finns. Detta är returtypen för `ValidateTopology`.

Description

En sammansatt typ som består av ett felmeddelande och två heltal. Funktionen ValidateTopology returnerar en uppsättning av dessa för att beteckna valideringsfel och id1 och id2 för att beteckna id för de topologiobjekt som är inblandade i felet.

Field	Type	Description
`error`	varchar	Denotes the type of error. Current error descriptors are: coincident nodes, edge crosses node, edge not simple, edge end node geometry mismatch, edge start node geometry mismatch, face overlaps face, face within face.
`id1`	integer	Denotes the identifier of the edge, face, or node in error.
`id2`	integer	For errors that involve two objects, denotes the secondary edge or node.

Se även

ValidateTopology

9.4 Topologidomäner

9.4.1 TopoElement

TopoElement — En matris med 2 heltal som vanligtvis används för att identifiera en TopoGeometry-komponent.

Description

En matris med 2 heltal som används för att representera en komponent i en enkel eller hierarkisk TopoGeometry.

När det gäller en enkel TopoGeometry representerar det första elementet i matrisen identifieraren för en topologisk primitiv och det andra elementet representerar dess typ (1:nod, 2:kant, 3:yta). När det gäller en hierarkisk TopoGeometry representerar det första elementet i matrisen identifieraren för en underordnad TopoGeometry och det andra elementet representerar dess lageridentifierare.

Notera
För varje given hierarkisk TopoGeometry kommer alla underordnade TopoGeometry-element att komma från samma underordnade lager, enligt vad som anges i topology.layer-posten för lagret i den TopoGeometry som definieras.

Exempel

Code

```
SELECT te[1] AS id, te[2] AS type FROM
( SELECT ARRAY[1, 2]::topology.topoelement AS te ) f;
```

Output

```
id | type
----+------
  1 |    2
```

Code

```
SELECT ARRAY[1, 2]::topology.topoelement;
```

Output

```
te
-------
 {1,2}
```

This example shows what happens when casting a three-element array to topology.topoelement. The domain requires a two-element array, so the dimension check fails.

Code

```
SELECT ARRAY[1, 2, 3]::topology.topoelement;
```

Output

```
ERROR:  value for domain topology.topoelement violates check constraint "dimensions"
```

Se även

GetTopoGeomElements, TopoElementArray, TopoGeometry, TopoGeom_addElement, TopoGeom_remEleme

9.4.2 TopoElementArray

TopoElementArray — En array av TopoElement-objekt.

Description

En array med 1 eller flera TopoElement-objekt, som vanligtvis används för att skicka runt komponenter i TopoGeometry-objekt.

Exempel

Code

```
SELECT '{{1,2},{4,3}}'::topology.topoelementarray AS tea;
```

Output

```
{{1,2},{4,3}}
```

This is the more verbose equivalent.

Code

```
SELECT ARRAY[ARRAY[1, 2], ARRAY[4, 3]]::topology.topoelementarray AS tea;
```

Output

```
{{1,2},{4,3}}
```

This example uses the array aggregate function packaged with topology.

Code

```
SELECT topology.TopoElementArray_Agg(ARRAY[e, t] ORDER BY e, t) As tea
FROM generate_series(1, 4) As e CROSS JOIN generate_series(1, 3) As t;
```

Output

```
{{1,1},{1,2},{1,3},{2,1},{2,2},{2,3},{3,1},{3,2},{3,3},{4,1},{4,2},{4,3}}
```

Code

```
SELECT '{{1,2,4},{3,4,5}}'::topology.topoelementarray AS tea;
```

Output

```
ERROR:  value for domain topology.topoelementarray violates check constraint "dimensions"
```

Se även

TopoElement, GetTopoGeomElementArray, TopoElementArray_Agg

9.5 Hantering av topologi och topogeometri

9.5.1 AddTopoGeometryColumn

AddTopoGeometryColumn — Lägger till en topogeometrikolumn i en befintlig tabell, registrerar den nya kolumnen som ett lager i topology.layer och returnerar det nya layer_id.

Synopsis

integer **AddTopoGeometryColumn**(name topology_name, name schema_name, name table_name, name column_name, varchar feature_type, integer child_layer);
integer **AddTopoGeometryColumn**(name topology_name, regclass tab, name column_name, integer layer_id, varchar feature_type, integer child_layer);

Description

Varje TopoGeometry-objekt hör till ett specifikt lager i en specifik topologi. Innan du skapar ett TopoGeometry objekt måste du skapa dess TopologyLayer. Ett Topology Layer är en association av en feature-tabell med topologin. Det innehåller också information om typ och hierarki. Vi skapar ett lager med hjälp av funktionen AddTopoGeometryColumn():

Denna funktion kommer både att lägga till den begärda kolumnen i tabellen och lägga till en post i tabellen topology.layer med all angiven information.

Om du inte anger [child_layer] (eller sätter det till NULL) kommer detta lager att innehålla grundläggande TopoGeometries (sammansatta av primitiva topologielement). Annars kommer detta lager att innehålla hierarkiska TopoGeometries (sammansatta av TopoGeometries från child_layer).

När lagret har skapats (dess id returneras av funktionen AddTopoGeometryColumn) är du redo att konstruera TopoGeometry-objekt i det

Giltiga `feature_typesär`: PUNKT, MULTIPOINT, LINE, MULTILINE, POLYGON, MULTIPOLYGON, COLLECTION

Tillgänglighet: 1.1

Exempel

This example creates the new table in the `ma_topo` schema. The table could be created in a different schema, in which case `topology_name` and `schema_name` would be different.

Code

```
CREATE SCHEMA ma;
CREATE TABLE ma.parcels(gid serial, parcel_id varchar(20) PRIMARY KEY, address text);
SELECT topology.AddTopoGeometryColumn('ma_topo', 'ma', 'parcels', 'topo', 'POLYGON');
```

Code

```
CREATE SCHEMA ri;
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);
SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

Se även

DropTopoGeometryColumn, toTopoGeom, CreateTopology, CreateTopoGeom

9.5.2 RenameTopoGeometryColumn

RenameTopoGeometryColumn — Byter namn på en topogeometri-kolumn

Synopsis

topology.layer **RenameTopoGeometryColumn**(regclass layer_table, name feature_column, name new_nam

Description

Denna funktion ändrar namnet på en befintlig TopoGeometry-kolumn och ser till att metadatainformationen om den uppdateras i enlighet med detta.

Tillgänglighet: 3.4.0

Exempel

Code

```
SELECT topology.RenameTopoGeometryColumn('public.parcels', 'topogeom', 'tgeom');
```

Se även

AddTopoGeometryColumn, RenameTopology

9.5.3 DropTopology

DropTopology — Använd med försiktighet: Tar bort ett topologischema och raderar dess referens från tabellen topology.topology och referenser till tabeller i det schemat från tabellen geometry_columns.

Synopsis

integer **DropTopology**(varchar topology_schema_name);

Description

Tar bort ett topologischema och raderar dess referens från tabellen topology.topology och referenser till tabeller i schemat från tabellen geometry_columns. Denna funktion bör användas med försiktighet, eftersom den kan förstöra data som du bryr dig om. Om schemat inte finns tar den bara bort referensposter i det namngivna schemat.

Tillgänglighet: 1.1

Exempel

Cascade släpper ma_topo-schemat och tar bort alla referenser till det i topology.topology och geometry_columns.

Code

```
SELECT topology.DropTopology('ma_topo');
```

Se även

DropTopoGeometryColumn

9.5.4 RenameTopology

RenameTopology — Byter namn på en topologi

Synopsis

varchar **RenameTopology**(varchar old_name, varchar new_name);

Description

Byter namn på ett topologischema och uppdaterar dess metadatapost i tabellen topology . topology.

Tillgänglighet: 3.4.0

Exempel

Byt namn på en topologi från `topo_stage` till `topo_prod`.

Code

```
SELECT topology.RenameTopology('topo_stage', 'topo_prod');
```

Se även

CopyTopology, RenameTopoGeometryColumn

9.5.5 DropTopoGeometryColumn

DropTopoGeometryColumn — Tar bort kolumnen topogeometry från tabellen med namnet `table_name` i schema `schema_name` och avregistrerar kolumnerna från tabellen topology.layer.

Synopsis

text **DropTopoGeometryColumn**(varchar schema_name, varchar table_name, varchar column_name);

Description

Tar bort kolumnen topogeometry från tabellen med namnet `table_name` i schema `schema_name` och avregistrerar kolumnerna från tabellen topology.layer. Returnerar en sammanfattning av borttagningsstatus. OBS: den sätter först alla värden till NULL innan den släpper för att kringgå kontroller av referensintegritet.

Tillgänglighet: 1.1

Exempel

Code

```
SELECT topology.DropTopoGeometryColumn('ma_topo', 'parcel_topo', 'topo');
```

Se även

AddTopoGeometryColumn

9.5.6 FixCorruptTopoGeometryColumn

FixCorruptTopoGeometryColumn — Fixes topogeometry corruption caused by upgrade to postgis_topology 3.6.0 and higher

Synopsis

text **FixCorruptTopoGeometryColumn**(name layerSchema, name layerTable, name layerColumn);

Description

When upgrading from PostGIS topology <3.6.0 to version >3.6.0+, the topogeometry column definition was changed. This caused corruption in topogeometries created before the upgrade. This function fixes this corruption in affected tables.

The function refuses to process tables with expression or partial indexes; drop or rebuild those indexes manually around the repair.

Availability: 3.6.1

Exempel

Fix all topology columns

Code

```
SELECT topology.FixCorruptTopoGeometryColumn(schema_name, table_name, feature_column)
    FROM topology.layer;
```

Se även

UpgradeTopology

9.5.7 Populate_Topology_Layer

Populate_Topology_Layer — Lägger till saknade poster i tabellen topology.layer genom att läsa metadata från topotabeller.

Synopsis

setof record **Populate_Topology_Layer**();

Description

Lägger till saknade poster i tabellen `topology.layer` genom att inspektera topologibegränsningar för tabeller. Denna funktion är användbar för att åtgärda poster i topologikatalogen efter återställning av scheman med topodata.

Den returnerar en lista över skapade poster. Kolumner som returneras är `schema_name`, `table_name`, `feature_column`.

Tillgänglighet: 2.3.0

Exempel

Code

```
SELECT CreateTopology('strk_topo');
CREATE SCHEMA strk;
CREATE TABLE strk.parcels(gid serial, parcel_id varchar(20) PRIMARY KEY, address text);
SELECT topology.AddTopoGeometryColumn('strk_topo', 'strk', 'parcels', 'topo', 'POLYGON');
```

Output

```
createtopology
----------------
              1
(1 row)

CREATE SCHEMA
CREATE TABLE
 addtopogeometrycolumn
-----------------------
                     1
(1 row)
```

This query returns no records because the feature is already registered.

Code

```
SELECT *
FROM topology.Populate_Topology_Layer();
```

Output

```
(0 rows)
```

Rebuild the topology layer catalog.

Code

```
TRUNCATE TABLE topology.layer;
```

Output

```
TRUNCATE TABLE
```

Code

```
SELECT *
FROM topology.Populate_Topology_Layer();
```

Output

```
schema_name | table_name | feature_column
-------------+------------+----------------
 strk        | parcels    | topo
(1 row)
```

Code

```
SELECT topology_id, layer_id,
       schema_name AS sn, table_name AS tn, feature_column AS fc
FROM topology.layer;
```

Output

```
topology_id | layer_id |  sn  |   tn    |  fc
------------+----------+------+---------+------
           1 |        1 | strk | parcels | topo
(1 row)
```

Se även

AddTopoGeometryColumn

9.5.8 TopologySummary

TopologySummary — Tar ett topologinamn och ger sammanfattande totalsummor för olika typer av objekt i topologin.

Synopsis

text **TopologySummary**(varchar topology_schema_name);

Description

Tar ett topologinamn och ger sammanfattande totalsummor för olika typer av objekt i topologin.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT topology.TopologySummary('city_data');
```

Output

```
Topology city_data (329), SRID 4326, precision: 0
22 nodes, 24 edges, 10 faces, 29 topogeoms in 5 layers
Layer 1, type Polygonal (3), 9 topogeoms
 Deploy: features.land_parcels.feature
Layer 2, type Puntal (1), 8 topogeoms
 Deploy: features.traffic_signs.feature
Layer 3, type Lineal (2), 8 topogeoms
 Deploy: features.city_streets.feature
Layer 4, type Polygonal (3), 3 topogeoms
 Hierarchy level 1, child layer 1
 Deploy: features.big_parcels.feature
Layer 5, type Puntal (1), 1 topogeoms
 Hierarchy level 1, child layer 2
 Deploy: features.big_signs.feature
```

9.5.9 ValidateTopology

ValidateTopology — Returnerar en uppsättning validatetopology_returntype-objekt som beskriver problem med topologin.

Synopsis

setof validatetopology_returntype **ValidateTopology**(varchar toponame, geometry bbox);

Description

Returnerar en uppsättning validatetopology_returntype -objekt som beskriver problem med topologin och eventuellt begränsar kontrollen till det område som anges av parametern `bbox`.

Nedan visas en lista över möjliga fel, vad de betyder och vad de returnerade ID:na representerar:

Fel	id1	id2	Betydelse
sammanfallande noder	Identifierare för första noden.	Identifierare för den andra noden.	Två noder har samma geometri.
kant korsar nod	Identifierare av kanten.	Identifierare för noden.	En kant har en nod i sitt inre. Se ST_Relate.
ogiltig flank	Identifierare av kanten.		En kantgeometri är ogiltig. Se ST_IsValid.
kant inte enkel	Identifierare av kanten.		En kantgeometri har självskärningar. Se ST_IsSimple.
kant korsar kant	Identifierare av första kanten.	Identifierare av andra kanten.	Två kanter har en inre skärningspunkt. Se ST_Relate.
kant start nod geometri missmatchning	Identifierare av kanten.	Identifierare för den angivna startnoden.	Geometrin för den nod som anges som startnod för en kant stämmer inte överens med den första punkten i kantgeometrin. Se ST_StartPoint.
kant ändnod geometri missanpassning	Identifierare av kanten.	Identifierare för den angivna slutnoden.	Geometrin för den nod som anges som slutnod för en kant stämmer inte överens med den sista punkten i kantgeometrin. Se ST_EndPoint.
yta utan kanter	Identifierare av den föräldralösa ytan.		Ingen kant rapporterar en befintlig yta på någon av sina sidor (left_face, right_face).
ytan har inga ringar	Identifierare av den delvis definierade ytan.		Kanter som rapporterar en yta på sina sidor bildar inte en ring.

Fel	id1	id2	Betydelse
ytan har fel mbr	Identifierare av yta med felaktig mbr-cache.		Minsta avgränsande rektangel för en yta stämmer inte överens med minsta avgränsande box för samlingen av kanter som rapporterar ytan på sina sidor.
hål inte i annonserad yta	Signerad identifierare av en kant som identifierar ringen. Se GetRingEdges.		En ring av kanter som rapporterar en yta på sin utsida är innesluten i en annan yta.
ej isolerad nod har ej innehållande_yta	Identifierare för den odefinierade noden.		En nod som rapporteras ligga på gränsen till en eller flera kanter indikerar en innehållande yta.
isolerad nod har innehållande_yta	Identifierare för den odefinierade noden.		En nod som inte rapporteras som liggande på gränsen till någon kant saknar indikationen för en innehållande yta.
isolerad nod har fel containing_face	Identifierare för den felaktigt presenterade noden.		En nod som inte rapporteras som liggande på gränsen till någon kant indikerar en innehållande yta som inte är den faktiska yta som innehåller den. Se GetFaceContainingPoint.
ogiltig next_right_edge	Identifierare av den felrepresenterade kanten.	Signerad id för den kant som ska anges som nästa högerkant.	Den kant som anges som nästa kant som man stöter på när man går på höger sida av en kant är fel.
ogiltig nästa_vänstra_kant	Identifierare av den felrepresenterade kanten.	Signerad id för den kant som ska anges som nästa vänstra kant.	Den kant som anges som nästa kant som man stöter på när man går på vänster sida av en kant är fel.
blandad ytmärkning i ring	Signerad identifierare av en kant som identifierar ringen. Se GetRingEdges.		Kanter i en ring indikerar motstridiga ytor på gångsidan. Detta kallas också för en Side Location Conflict”.
ej sluten ring	Signerad identifierare av en kant som identifierar ringen. Se GetRingEdges.		En ring av kanter som bildas genom att följa next_left_edge/next_right_edg attributen börjar och slutar på olika noder.

Fel	id1	id2	Betydelse
ytan har flera skal	Identifierare av den konturerade ytan.	Signerad identifierare av en kant som identifierar ringen. Se GetRingEdges.	Mer än en ring av kanter indikerar samma yta på dess insida.

Tillgänglighet: 1.0.0

Förbättrad: 2.0.0 effektivare detektering av kantkorsningar och korrigeringar för falska positiva som fanns i tidigare versioner.

Ändrad: 2.2.0 värdena för id1 och id2 byttes ut mot ”edge crosses nodeför att stämma överens med felbeskrivningen.

Ändrad: 3.2.0 lade till valfri bbox-parameter, utför kontroller av ytmärkning och kantlänkning.

Exempel

Code

```
SELECT * FROM topology.ValidateTopology('ma_topo');
```

Output

```
error        | id1 | id2
-------------------+-----+-----
face without edges |   1 |
```

Se även

validatetopology_returntype

9.5.10 ValidateTopologyRelation

ValidateTopologyRelation — Returnerar information om ogiltiga topologirelationsposter

Synopsis

setof record **ValidateTopologyRelation**(varchar toponame);

Description

Returnerar en uppsättning poster som ger information om ogiltigheter i topologins relationstabell.

Tillgänglighet: 3.2.0

Se även

ValidateTopology

9.5.11 ValidateTopologyPrecision

ValidateTopologyPrecision — Returnerar icke-precisa toppunkter i topologin.

Synopsis

geometry **ValidateTopologyPrecision**(name toponame, geometry bbox, float8 gridSize);

Description

Returnerar alla hörn som inte är avrundade till topologin eller given `gridSize` som en punktgeometri, eventuellt med begränsning av kontrollen till det område som anges av parametern `bbox`.

Tillgänglighet: 3.6.0

Exempel

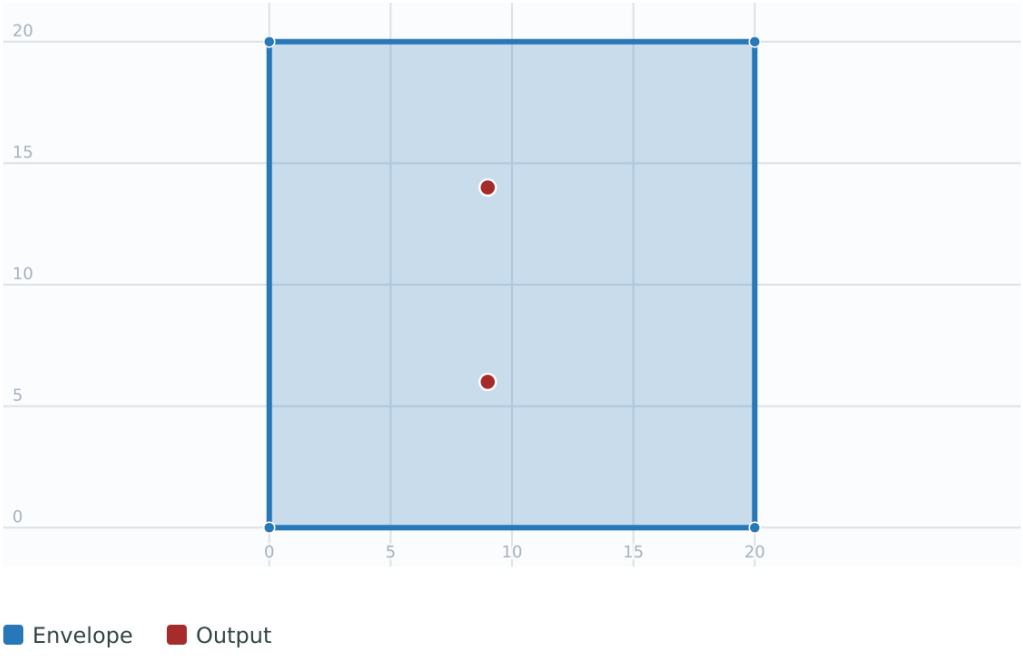
Code

```
SELECT g
FROM topology.ValidateTopologyPrecision(
    'city_data',
    gridSize =
> 2,
    bbox =
> ST_MakeEnvelope(0, 0, 20, 20)
) g;
```

Output

```
MULTIPOINT(9 6,9 14)
```

Figure



Se även

MakeTopologyPrecise

9.5.12 MakeTopologyPrecise

MakeTopologyPrecise — Fäst topologivinklar till precisionsrutnätet.

Synopsis

void **MakeTopologyPrecise**(name toponame, geometry bbox, float8 gridSize);

Description

Snäpper alla hörn i en topologi till topologins precisionsraster eller till det raster vars storlek anges med parametern `gridSize`, och begränsar eventuellt operationen till de objekt som skär det område som anges med parametern `bbox`.

Using a `gridSize` larger than the smallest edge extent raises an exception rather than collapsing the edge into an empty geometry. This safeguards adjacent topological relationships by preventing precision snapping from removing primitives altogether.

Notera
Snapping kan göra topologin ogiltig, så det rekommenderas att du kontrollerar resultatet av operationen med ValidateTopology.

Tillgänglighet: 3.6.0

Exempel

Code

```
SELECT topology.MakeTopologyPrecise(
    'city_data',
    gridSize =
> 2
);
```

Se även

ValidateTopologyPrecision, ValidateTopology

9.5.13 FindTopology

FindTopology — Returnerar en topologipost på olika sätt.

Synopsis

topology **FindTopology**(topogeometry topogeom);
topology **FindTopology**(regclass layerTable, name layerColumn);
topology **FindTopology**(name layerSchema, name layerTable, name layerColumn);
topology **FindTopology**(text topoName);
topology **FindTopology**(int id);

Description

Tar en topologiidentifierare eller identifieraren för ett topologirelaterat objekt och returnerar en topology.topology-post.

Tillgänglighet: 3.2.0

Exempel

Code

```
SELECT name(FindTopology('features.land_parcels', 'feature'));
```

Output

```
city_data
```

Se även

FindLayer

9.5.14 FindLayer

FindLayer — Returnerar en topology.layer-post på olika sätt.

Synopsis

topology.layer **FindLayer**(topogeometry tg);
topology.layer **FindLayer**(regclass layer_table, name feature_column);
topology.layer **FindLayer**(name schema_name, name table_name, name feature_column);
topology.layer **FindLayer**(integer topology_id, integer layer_id);

Description

Tar en lageridentifierare eller identifieraren för ett topologirelaterat objekt och returnerar en topology.layer-post.

Tillgänglighet: 3.2.0

Exempel

Code

```
SELECT layer_id(FindLayer('features.land_parcels', 'feature'));
```

Output

```
1
```

Se även

FindTopology

9.5.15 TotalTopologySize

TotalTopologySize — Totalt diskutrymme som används av den angivna topologin, inklusive alla index och TOAST-data.

Synopsis

int8 **TotalTopologySize**(name toponame);

Description

Tar ett topologinamn och ger det totala diskutrymmet som används av alla dess tabeller, inklusive index och TOAST-data.

Tillgänglighet: 3.6.0

Exempel

Code

```
SELECT topology.TopologySummary('city_data');
```

Output

```
Topology city_data (329), SRID 4326, precision: 0
22 nodes, 24 edges, 10 faces, 29 topogeoms in 5 layers
Layer 1, type Polygonal (3), 9 topogeoms
 Deploy: features.land_parcels.feature
Layer 2, type Puntal (1), 8 topogeoms
 Deploy: features.traffic_signs.feature
Layer 3, type Lineal (2), 8 topogeoms
 Deploy: features.city_streets.feature
Layer 4, type Polygonal (3), 3 topogeoms
 Hierarchy level 1, child layer 1
 Deploy: features.big_parcels.feature
Layer 5, type Puntal (1), 1 topogeoms
 Hierarchy level 1, child layer 2
 Deploy: features.big_signs.feature
```

9.5.16 UpgradeTopology

UpgradeTopology — Uppgraderar den angivna topologin till att stödja stora ids (int8) för topologi och primitiva ids.

Synopsis

void **UpgradeTopology**(name toponame);

Description

Tar ett topologinamn och uppgraderar det så att det stöder stora id:n (int8) för topologi och primitiva id:n. Funktionen uppgraderar följande: - face (face_id kolumn från int4 till int8, face_id_seq från int4 till int8) - node (node_id kolumn från int4 till int8, containing_face kolumn från int4 till int8, node_id_seq från int4 till int8) - edge_data (edge_id kolumn från int4 till int8, edge_data_edge_id_seq från int4 till int8, left_face och right_face kolumner från int4 till int8, start_node och end_node kolumner från int4 till int8, next_left_edge och next_right_edge kolumner från int4 till int8) - relation (topogeo_id kolumn från int4 till int8, element_id från int4 till int8) - topologi (useslargeids kolumn satt till true)

Tillgänglighet: 3.6.0

Exempel

Code

```
SELECT topology.UpgradeTopology('city_data');
```

9.6 Hantering av topologistatistik

Att lägga till element i en topologi utlöser många databasfrågor för att hitta befintliga kanter som ska delas, lägga till noder och uppdatera kanter som ska knytas till det nya linjeverket. Därför är det bra om statistiken över data i topologitabellerna är uppdaterad.

PostGIS topologipopulation och redigeringsfunktioner uppdaterar inte statistiken automatiskt eftersom det skulle vara onödigt att uppdatera statistiken efter varje ändring i en topologi, så det är uppringarens skyldighet att ta hand om det.

Notera
Att statistiken som uppdateras av autovacuum INTE kommer att vara synlig för transaktioner som startade innan autovacuum-processen slutfördes, så långvariga transaktioner måste köra ANALYZE själva för att använda uppdaterad statistik.

9.7 Topology Constructors

9.7.1 CreateTopology

CreateTopology — Skapar ett nytt topologischema och registrerar det i tabellen topology.topology.

Synopsis

integer **CreateTopology**(name topology_schema_name, integer srid, double precision prec=0, boolean hasz=false, integer topoid=0, boolean useslargeids=false);

Description

Skapar ett nytt topologischema med namnet `topology_name` och registrerar det i tabellen topology.topology. Topologierna måste ha unika namn. Topologitabellerna (`edge_data`, `face`, `node` och `relation`) skapas i schemat. Den returnerar id för topologin.

`Srid` är det spatiala referenssystemets SRID för topologin. Standardvärdet för SRID är -1 (okänt) om det inte anges.

Toleransen `prec` mäts i enheterna i det spatiala referenssystemet. Toleransen är 0 som standard.

`hasz` är som standard false om det inte anges.

`topoid` valfri explicit identifierare (möjliggör deterministisk tilldelning av topologi-id, måste vara unik)

`useslargeids` valfritt, standardvärde false. Om true, kommer topologin att skapas för att stödja stora id (int8) för topologi- och primitiva id.

Detta liknar SQL/MM ST_InitTopoGeo men har fler funktioner.

Tillgänglighet: 1.1

Enhanced: 2.0 added support for `hasZ`.

Enhanced: 3.6.0 added `topoid` and `usesLargeIDs` parameters.

Exempel

Skapa ett topologischema som heter `ma_topo` som lagrar kanter och noder i Massachusetts State Plane-meters (SRID = 26986). Toleransen representerar 0,5 meter eftersom det spatiala referenssystemet är meterbaserat.

Code

```
SELECT topology.CreateTopology('ma_topo', 26986, 0.5);
```

Skapa en topologi för Rhode Island med namnet `ri_topo` i det spatiala referenssystemet State Plane-feet (SRID = 3438)

Code

```
SELECT topology.CreateTopology('ri_topo', 3438) AS topoid;
```

Output

```
2
```

Se även

Avsnitt 4.5, ST_InitTopoGeo

9.7.2 CopyTopology

CopyTopology — Gör en kopia av en topologi (noder, kanter, ytor, lager och TopoGeometries) till ett nytt schema

Synopsis

integer **CopyTopology**(varchar existing_topology_name, varchar new_name);

Description

Skapar en ny topologi med namnet `new_name`, med SRID och precision kopierade från `existing_topology_n`
Noder, kanter och ytor i `existing_topology_name` kopieras till den nya topologin, liksom lager och deras associerade TopoGeometries.

Notera
De nya raderna i tabellen `topology.layer` innehåller syntetiska värden för schema_name, table_name och feature_column. Detta beror på att TopoGeometry-objekten endast finns som en definition och ännu inte är tillgängliga i en användardefinierad tabell.

Tillgänglighet: 2.0.0

Exempel

Gör en säkerhetskopia av en topologi som heter `ma_topo`.

Code

```
SELECT topology.CopyTopology('ma_topo', 'ma_topo_backup');
```

Se även

Avsnitt 4.5, CreateTopology, RenameTopology

9.7.3 ST_InitTopoGeo

ST_InitTopoGeo — Skapar ett nytt topologischema och registrerar det i tabellen topology.topology.

Synopsis

text **ST_InitTopoGeo**(varchar topology_schema_name);

Description

Detta är SQL-MM:s motsvarighet till CreateTopology. Den saknar alternativ för spatialt referenssystem och tolerans. den returnerar en textbeskrivning av topologins skapande, i stället för topologins id.

Tillgänglighet: 1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3 Topo-Geo och Topo-Net 3: Rutinbeskrivningar: X.3.17

Exempel

Code

```
SELECT topology.ST_InitTopoGeo('topo_schema_to_create') AS topocreation;
```

Output

```
Topology-Geometry 'topo_schema_to_create' (id:7) created.
```

Se även

CreateTopology

9.7.4 ST_CreateTopoGeo

ST_CreateTopoGeo — Lägger till en samling geometrier till en given tom topologi och returnerar ett meddelande om det lyckas.

Synopsis

text **ST_CreateTopoGeo**(varchar atopology, geometry acollection);

Description

Lägger till en samling geometrier till en given tom topologi och returnerar ett meddelande om det lyckas.

Användbar för att fylla i en tom topologi.

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer -- X.3.18

Exempel

Populate the topology.

Code

```
SELECT topology.ST_CreateTopoGeo(
'ri_topo',
ST_GeomFromText(
 'MULTILINESTRING((384744 236928,384750 236923,384769 236911,384799 236895,384811 ↩
    236890,384833 236884,
 384844 236882,384866 236881,384879 236883,384954 236898,385087 236932,385117 236938,
 385167 236938,385203 236941,385224 236946,385233 236950,385241 236956,385254 236971,
 385260 236979,385268 236999,385273 237018,385273 237037,385271 237047,385267 237057,
 385225 237125,385210 237144,385192 237161,385167 237192,385162 237202,385159 237214,
 385159 237227,385162 237241,385166 237256,385196 237324,385209 237345,385234 237375,
 385237 237383,385238 237399,385236 237407,385227 237419,385213 237430,385193 237439,
 385174 237451,385170 237455,385169 237460,385171 237475,385181 237503,385190 237521,
 385200 237533,385206 237538,385213 237541,385221 237542,385235 237540,385242 237541,
 385249 237544,385260 237555,385270 237570,385289 237584,385292 237589,385291 237596,385284 ↩
    237630))',
 3438)
 );
```

Output

```
Topology ri_topo populated
```

Create the tables and topogeometries.

Code

```
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);

SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

Se även

TopoGeo_LoadGeometry, AddTopoGeometryColumn, CreateTopology, DropTopology

9.7.5 TopoGeo_AddPoint

TopoGeo_AddPoint — Lägger till en punkt i en befintlig topologi med hjälp av en tolerans och eventuellt genom att dela en befintlig kant.

Synopsis

bigint **TopoGeo_AddPoint**(varchar atopology, geometry apoint, float8 tolerance);

Description

Lägger till en punkt till en befintlig topologi och returnerar dess identifierare. Den givna punkten snäpper till befintliga noder eller kanter inom given tolerans. En befintlig kant kan delas av den snäppta punkten.

Tillgänglighet: 2.0.0

Se även

TopoGeo_AddLineString, TopoGeo_AddPolygon, TopoGeo_LoadGeometry, AddNode, CreateTopology

9.7.6 TopoGeo_AddLineString

TopoGeo_AddLineString — Lägger till en linestrings till en befintlig topologi med hjälp av en tolerans och eventuellt delning av befintliga kanter/ytor.

Synopsis

SETOF bigint **TopoGeo_AddLineString**(varchar atopology, geometry aline, float8 tolerance, int max_edges

Description

Adds a linestring to an existing topology and returns a set of signed edge identifiers forming it up (negative identifies mean the edge goes in the opposite direction of the input linestring).

The given line will snap to existing nodes or edges within given tolerance. Existing edges and faces may be split by the line. New nodes and faces may be added, in addition to new edges.

The returned edge identifiers may be either existing edges or newly created edges as needed to fully represent the input line as closely as possible.

The number of newly created edges (either covering space previously uncovered or resulting from split of existing edges) may be limited by the `max_edges` parameter.

Notera
When adding many lines, the load order can have a large impact on performance because each call checks the line against topology elements already loaded. In bulk workflows, adding large closed or enclosing lines before smaller contained lines can avoid expensive later splits. For loading a collection in one call, see TopoGeo_LoadGeometry.

Notera
Uppdatering av statistik om topologier som laddas via denna funktion är upp till den som anropar, se maintaining statistics during topology editing and population.

Tillgänglighet: 2.0.0

Förbättrad: 3.2.0 lade till stöd för att returnera signerad identifierare.

Enhanced: 3.7.0 added support for limiting the number of new edges created in the topology.

Se även

TopoGeo_AddPoint, TopoGeo_AddPolygon, TopoGeo_LoadGeometry, AddEdge, CreateTopology

9.7.7 TopoGeo_AddPolygon

TopoGeo_AddPolygon — Lägger till en polygon till en befintlig topologi med hjälp av en tolerans och eventuellt delning av befintliga kanter/ytor. Returnerar ytidentifierare.

Synopsis

SETOF bigint **TopoGeo_AddPolygon**(varchar atopology, geometry apoly, float8 tolerance);

Description

Lägger till en polygon till en befintlig topologi och returnerar en uppsättning ytidentifierare som bildar den. Gränsen för den givna polygonen kommer att snäppa till befintliga noder eller kanter inom given tolerans. Befintliga kanter och ytor kan delas av den nya polygonens gräns.

Notera
Uppdatering av statistik om topologier som laddas via denna funktion är upp till den som anropar, se maintaining statistics during topology editing and population.

Tillgänglighet: 2.0.0

Se även

TopoGeo_AddPoint, TopoGeo_AddLineString, TopoGeo_LoadGeometry, AddFace, CreateTopology

9.7.8 TopoGeo_LoadGeometry

TopoGeo_LoadGeometry — Läs in en geometri i en befintlig topologi, snappa och dela efter behov.

Synopsis

void **TopoGeo_LoadGeometry**(varchar atopology, geometry ageom, float8 tolerance);

Description

Läser in en geometri i en befintlig topologi. Den givna geometrin kommer att snäppa till befintliga noder eller kanter inom given tolerans. Befintliga kanter och ytor kan delas som en följd av laddningen.

Notera
Uppdatering av statistik om topologier som laddas via denna funktion är upp till den som anropar, se maintaining statistics during topology editing and population.

Tillgänglighet: 3.5.0

Se även

TopoGeo_AddPoint, TopoGeo_AddLineString, TopoGeo_AddPolygon, CreateTopology

9.8 Topologiredigerare

9.8.1 ST_AddIsoNode

ST_AddIsoNode — Lägger till en isolerad nod till en face i en topologi och returnerar nodeid för den nya noden. Om face är null skapas noden ändå.

Synopsis

bigint **ST_AddIsoNode**(varchar atopology, bigint aface, geometry apoint);

Description

Lägger till en isolerad nod med punktläge `apoint` till en befintlig yta med faceid `aface` till en topologi `atopologi` och returnerar den nya nodens nodeid.

Om det spatiala referenssystemet (srid) för punktgeometrin inte är detsamma som topologin , `apoint` inte är en punktgeometri, punkten är null eller punkten skär en befintlig kant (även vid gränserna) kommer ett undantag att utlösas. Om punkten redan existerar som en nod, kommer ett undantag att kastas.

Om `aface` inte är null och `apoint` inte ligger inom ytan, utlöses ett undantag.

Tillgänglighet: 1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Net Routines: X+1.3.1

Exempel

Se även

AddNode, CreateTopology, DropTopology, ST_Intersects

9.8.2 ST_AddIsoEdge

ST_AddIsoEdge — Lägger till en isolerad kant definierad av geometrin `alinestring` till en topologi som förbinder två befintliga isolerade noder `anode` och `anothernode` och returnerar kant-ID för den nya kanten.

Synopsis

bigint **ST_AddIsoEdge**(varchar atopology, bigint anode, bigint anothernode, geometry alinestring);

Description

Lägger till en isolerad kant definierad av geometrin `alinestring` till en topologi som förbinder två befintliga isolerade noder `anode` och `anothernode` och returnerar kant-ID för den nya kanten.

Om det spatiala referenssystemet (srid) för `alinestring-geometrin` inte är detsamma som topologin, om något av indatagumenten är null, om noderna finns i mer än en yta eller om noderna är start- eller slutnoder för en befintlig kant, så kastas ett undantag.

Om `alinestringen` inte ligger inom ytan för den yta som `anoden` och `anothernoden` tillhör, kastas ett undantag.

Om `anoden` och `anothernoden` inte är start- och slutpunkterna för `alinestringen` kastas ett undantag.

Tillgänglighet: 1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.4

Exempel

Se även

ST_AddIsoNode, ST_IsSimple, ST_Within

9.8.3 ST_AddEdgeNewFaces

ST_AddEdgeNewFaces — Lägg till en ny kant och, om den delar en yta, ta bort den ursprungliga ytan och ersätt den med två nya ytor.

Synopsis

bigint **ST_AddEdgeNewFaces**(varchar atopology, bigint anode, bigint anothernode, geometry acurve);

Description

Lägg till en ny kant och, om den delar en yta, ta bort den ursprungliga ytan och ersätt den med två nya ytor. Returnerar id för den nyligen tillagda kanten.

Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta.

Om något argument är null, de givna noderna är okända (måste redan finnas i `nodtabellen` i topologischemat), `acurve` inte är en `LINESTRING`, `anode` och `anothernode` inte är start- och slutpunkterna för `acurve`, så kastas ett fel.

Om det spatiala referenssystemet (srid) för `acurve-geometrin` inte är detsamma som topologin kastas ett undantag.

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.12

Exempel

Se även

ST_RemEdgeNewFace

ST_AddEdgeModFace

9.8.4 ST_AddEdgeModFace

ST_AddEdgeModFace — Lägg till en ny kant och, om den delar en yta, modifiera den ursprungliga ytan och lägg till en ny yta.

Synopsis

bigint **ST_AddEdgeModFace**(varchar atopology, bigint anode, bigint anothernode, geometry acurve);

Description

Lägg till en ny kant och, om detta splittrar en yta, modifiera den ursprungliga ytan och lägg till en ny.

Notera
Om möjligt skapas den nya ytan på vänster sida av den nya kanten. Detta är inte möjligt om ytan på vänster sida måste vara universums yta (obegränsad).

Returnerar id för den nyligen tillagda kanten.

Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta.

Om något argument är null, de givna noderna är okända (måste redan finnas i `nodtabellen` i topologischemat), `acurve` inte är en `LINESTRING`, `anode` och `anothernode` inte är start- och slutpunkterna för `acurve`, så kastas ett fel.

Om det spatiala referenssystemet (srid) för `acurve-geometrin` inte är detsamma som topologin kastas ett undantag.

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.13

Exempel

Se även

ST_RemEdgeModFace

ST_AddEdgeNewFaces

9.8.5 ST_RemEdgeNewFace

ST_RemEdgeNewFace — Tar bort en kant och, om den borttagna kanten separerade två ytor, tar bort de ursprungliga ytorna och ersätter dem med en ny yta.

Synopsis

bigint **ST_RemEdgeNewFace**(varchar atopology, bigint anedge);

Description

Tar bort en kant och, om den borttagna kanten separerade två ytor, tar bort de ursprungliga ytorna och ersätter dem med en ny yta.

Returnerar id för en nyskapad yta eller NULL, om ingen ny yta skapats. Ingen ny yta skapas när den borttagna kanten är dinglande eller isolerad eller begränsad med universums yta (vilket kan göra att universum flyter in i ytan på andra sidan).

Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta.

Vägrar att ta bort en kant som deltar i definitionen av en befintlig TopoGeometry. Vägrar att läka två ytor om någon TopoGeometry definieras av endast en av dem (och inte den andra).

Om något av argumenten är null, den angivna kanten är okänd (måste redan finnas i `kanttabellen` i topologischemat), topologinamnet är ogiltigt, kastas ett fel.

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinbeskrivningar: X.3.14

Exempel

Se även

ST_RemEdgeModFace

ST_AddEdgeNewFaces

9.8.6 ST_RemEdgeModFace

ST_RemEdgeModFace — Tar bort en kant, och om kanten separerar två ytor tas den ena ytan bort och den andra ytan modifieras så att den täcker utrymmet för båda ytorna.

Synopsis

bigint **ST_RemEdgeModFace**(varchar atopology, bigint anedge);

Description

Tar bort en kant, och om den borttagna kanten separerar två ytor tas den ena ytan bort och den andra ytan modifieras så att den täcker utrymmet för båda ytorna. Företrädesvis behålls ytan till höger, för att vara konsekvent med ST_AddEdgeModFace. Returnerar id för den yta som bevaras.

Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta.

Vägrar att ta bort en kant som deltar i definitionen av en befintlig TopoGeometry. Vägrar att läka två ytor om någon TopoGeometry definieras av endast en av dem (och inte den andra).

Om något av argumenten är null, den angivna kanten är okänd (måste redan finnas i `kanttabellen` i topologischemat), topologinamnet är ogiltigt, kastas ett fel.

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.15

Exempel

Se även

ST_AddEdgeModFace

ST_RemEdgeNewFace

9.8.7 ST_ChangeEdgeGeom

ST_ChangeEdgeGeom — Ändrar formen på en kant utan att påverka topologins struktur.

Synopsis

text **ST_ChangeEdgeGeom**(varchar atopology, bigint anedge, geometry acurve);

Description

Ändrar formen på en kant utan att påverka topologins struktur.

Om något av argumenten är null, den angivna kanten inte finns i `kanttabellen` i topologischemat, `acurvan` inte är en `LINESTRING` eller om modifieringen skulle ändra den underliggande topologin, så kastas ett fel.

Om det spatiala referenssystemet (srid) för `acurve-geometrin` inte är detsamma som topologin kastas ett undantag.

Om den nya `akurvan` inte är enkel, kastas ett fel.

Om förflyttningen av kanten från den gamla till den nya positionen skulle stöta på ett hinder, uppstår ett fel.

Tillgänglighet: 1.1.0

Förbättrad: 2.0.0 lägger till upprätthållande av topologisk konsistens

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer X.3.6

Exempel

Code

```
SELECT topology.ST_ChangeEdgeGeom(
    'ma_topo',
    1,
        ST_GeomFromText(
            'LINESTRING(227591.9 893900.4,227622.6 893844.3,227641.6 893816.6,227704.5 ↩
    893778.5)',
            26986)
);
```

Output

```
Edge 1 changed
```

Se även

ST_AddEdgeModFace

ST_RemEdgeModFace

ST_ModEdgeSplit

9.8.8 ST_ModEdgeSplit

ST_ModEdgeSplit — Dela en kant genom att skapa en ny nod längs en befintlig kant, modifiera den ursprungliga kanten och lägga till en ny kant.

Synopsis

bigint **ST_ModEdgeSplit**(varchar atopology, bigint anedge, geometry apoint);

Description

Delar en kant genom att skapa en ny nod längs en befintlig kant, modifiera den ursprungliga kanten och lägga till en ny kant. Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta. Returnerar identifieraren för den nyligen tillagda noden.

Tillgänglighet: 1.1

Ändrad: 2.0 - I tidigare versioner var detta felaktigt benämnt ST_ModEdgesSplit

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.9

Exempel

Add an edge to split.

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227592 893910,227600 893910) ↩
    ', 26986)) AS edgeid;
```

Output

```
3
```

Split the edge with a new node.

Code

```
SELECT topology.ST_ModEdgeSplit('ma_topo', 3, ST_SetSRID(ST_Point(227594, 893910), 26986)) ↩
    AS node_id;
```

Output

```
7
```

Se även

ST_NewEdgesSplit, ST_ModEdgeHeal, ST_NewEdgeHeal, AddEdge

9.8.9 ST_ModEdgeHeal

ST_ModEdgeHeal — Läker två kanter genom att ta bort den nod som förbinder dem, modifiera den första kanten och ta bort den andra kanten. Returnerar id för den borttagna noden.

Synopsis

bigint **ST_ModEdgeHeal**(varchar atopology, bigint anedge, bigint anotheredge);

Description

Läker två kanter genom att ta bort den nod som förbinder dem, modifiera den första kanten och ta bort den andra kanten. Returnerar id för den borttagna noden. Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta.

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.9

Se även

ST_ModEdgeSplit ST_NewEdgesSplit

9.8.10 ST_NewEdgeHeal

ST_NewEdgeHeal — Läker två kanter genom att ta bort noden som förbinder dem, ta bort båda kanterna och ersätta dem med en kant vars riktning är densamma som den första tillhandahållna kanten.

Synopsis

bigint **ST_NewEdgeHeal**(varchar atopology, bigint anedge, bigint anotheredge);

Description

Läker två kanter genom att ta bort noden som förbinder dem, ta bort båda kanterna och ersätta dem med en kant vars riktning är densamma som den första angivna kanten. Returnerar id för den nya kanten som ersätter de läkta. Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta.

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.9

Se även

ST_ModEdgeHeal ST_ModEdgeSplit ST_NewEdgesSplit

9.8.11 ST_MoveIsoNode

ST_MoveIsoNode — Flyttar en isolerad nod i en topologi från en punkt till en annan. Om den nya `apoint-geometrin` existerar som en nod kastas ett fel. Returnerar beskrivning av förflyttning.

Synopsis

text **ST_MoveIsoNode**(varchar atopology, bigint anode, geometry apoint);

Description

Flyttar en isolerad nod i en topologi från en punkt till en annan. Om den nya `apoint-geometrin` finns som en nod kommer ett fel att uppstå.

Om något argument är null, `apoint` inte är en punkt, den befintliga noden inte är isolerad (är en start- eller slutpunkt för en befintlig kant), den nya nodpositionen skär en befintlig kant (även vid slutpunkterna) eller den nya positionen är i en annan yta (sedan 3.2.0) så kastas ett undantag.

Om det spatiala referenssystemet (srid) för punktgeometrin inte är detsamma som topologin kastas ett undantag.

Tillgänglighet: 2.0.0

Förbättrad: 3.2.0 säkerställer att nicken inte kan flyttas till en annan sida

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Net Routines: X.3.2

Exempel

Add an isolated node with no face.

Code

```
SELECT topology.ST_AddIsoNode('ma_topo', NULL, ST_GeomFromText('POINT(227579 893916)', ↩
    26986)) AS nodeid;
```

Output

```
7
```

Move the new node.

Code

```
SELECT topology.ST_MoveIsoNode('ma_topo', 7, ST_GeomFromText('POINT(227579.5 893916.5)', ↩
    26986)) AS descrip;
```

Output

```
Isolated Node 7 moved to location 227579.5,893916.5
```

Se även

ST_AddIsoNode

9.8.12 ST_NewEdgesSplit

ST_NewEdgesSplit — Dela en kant genom att skapa en ny nod längs en befintlig kant, ta bort den ursprungliga kanten och ersätta den med två nya kanter. Returnerar id för den nya nod som skapats och som sammanfogar de nya kanterna.

Synopsis

bigint **ST_NewEdgesSplit**(varchar atopology, bigint anedge, geometry apoint);

Description

Delar en kant med kant-id `anedge` genom att skapa en ny nod med punktläge `apoint` längs aktuell kant, ta bort den ursprungliga kanten och ersätta den med två nya kanter. Returnerar id för den nya nod som skapats och som sammanfogar de nya kanterna. Uppdaterar alla befintliga sammanfogade kanter och relationer i enlighet med detta.

Om det spatiala referenssystemet (srid) för punktgeometrin inte är detsamma som topologin, `apoint` inte är en punktgeometri, punkten är null, punkten redan finns som en nod, kanten inte motsvarar en befintlig kant eller punkten inte ligger inom kanten så kastas ett undantag.

Tillgänglighet: 1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Net Routines: X.3.8

Exempel

Add an edge to split.

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575 893917,227592 893900) ↩
    ', 26986)) AS edgeid;
```

Output

```
2
```

Split the new edge.

Code

```
SELECT topology.ST_NewEdgesSplit('ma_topo', 2, ST_GeomFromText('POINT(227578.5 893913.5)', ↩
    26986)) AS newnodeid;
```

Output

```
6
```

Se även

ST_ModEdgeSplit ST_ModEdgeHeal ST_NewEdgeHeal AddEdge

9.8.13 ST_RemoveIsoNode

ST_RemoveIsoNode — Tar bort en isolerad nod och returnerar en beskrivning av åtgärden. Om noden inte är isolerad (är början eller slutet på en kant), kastas ett undantag.

Synopsis

text **ST_RemoveIsoNode**(varchar atopology, bigint anode);

Description

Tar bort en isolerad nod och returnerar en beskrivning av åtgärden. Om noden inte är isolerad (är början eller slutet på en kant), kastas ett undantag.

Tillgänglighet: 1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X+1.3.3

Exempel

Remove an isolated node with no face.

Code

```
SELECT topology.ST_RemoveIsoNode('ma_topo', 7) AS result;
```

Output

```
Isolated node 7 removed
```

Se även

ST_AddIsoNode

9.8.14 ST_RemoveIsoEdge

ST_RemoveIsoEdge — Tar bort en isolerad kant och returnerar en beskrivning av åtgärden. Om kanten inte är isolerad kastas ett undantag.

Synopsis

text **ST_RemoveIsoEdge**(varchar atopology, bigint anedge);

Description

Tar bort en isolerad kant och returnerar en beskrivning av åtgärden. Om kanten inte är isolerad kastas ett undantag.

Tillgänglighet: 1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM: Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X+1.3.3

Exempel

Remove an isolated edge with no face.

Code

```
SELECT topology.ST_RemoveIsoEdge('ma_topo', 7) AS result;
```

Output

```
Isolated edge 7 removed
```

Se även

ST_AddIsoNode

9.9 Topologi-accessorer

9.9.1 GetEdgeByPoint

GetEdgeByPoint — Hittar kant-id för en kant som skär en given punkt.

Synopsis

bigint **GetEdgeByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

Hämtar id för en kant som skär en Point.

Funktionen returnerar ett heltal (id-edge) givet en topologi, en POINT och en tolerans. Om tolerans = 0 måste punkten skära kanten.

Om en `punkt` inte korsar en kant returneras 0 (noll).

Om use tolerance > 0 och det finns mer än en kant nära punkten kastas ett undantag.

Notera
Om toleransen = 0 använder funktionen ST_Intersects, annars används ST_DWithin.

Utförs av GEOS-modulen.

Tillgänglighet: 2.0.0

Exempel

I dessa exempel används kanter som vi skapade i AddEdge

Code

```
SELECT topology.GetEdgeByPoint('ma_topo', geom, 1) As with1mtol, topology.GetEdgeByPoint(' ↩
    ma_topo', geom, 0) As withnotol
FROM ST_GeomFromEWKT('SRID=26986;POINT(227622.6 893843)') As geom;
```

Output

```
with1mtol | withnotol
-----------+-----------
         2 |         0
```

This query errors because two or more edges are found.

Code

```
SELECT topology.GetEdgeByPoint('ma_topo', geom, 1) As nearnode
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
```

Output

```
ERROR:  Two or more edges found
```

Se även

AddEdge, GetNodeByPoint, GetFaceByPoint

9.9.2 GetFaceByPoint

GetFaceByPoint — Hitta en yta som skär en given punkt.

Synopsis

bigint **GetFaceByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

Hittar en yta som refereras av en punkt, med given tolerans.

Funktionen letar effektivt efter en yta som skär en cirkel med punkten som centrum och toleransen som radie.

Om ingen yta korsar den angivna sökplatsen returneras 0 (universell yta).

Om mer än en yta korsar den plats där frågan ställs uppstår ett undantag.

Tillgänglighet: 2.0.0

Förbättrad: 3.2.0 effektivare implementering och tydligare kontrakt, slutar fungera med ogiltiga topologier.

Exempel

Code

```
SELECT topology.GetFaceByPoint('ma_topo', geom, 10) As with1mtol, topology.GetFaceByPoint(' ↩
    ma_topo', geom, 0) As withnotol
FROM ST_GeomFromEWKT('POINT(234604.6 899382.0)') As geom;
```

Output

```
with1mtol | withnotol
-----------+-----------
         1 |         0
```

This query errors because two or more faces are found.

Code

```
SELECT topology.GetFaceByPoint('ma_topo', geom, 1) As nearnode
FROM ST_GeomFromEWKT('POINT(227591.9 893900.4)') As geom;
```

Output

```
ERROR:  Two or more faces found
```

Se även

GetFaceContainingPoint, AddFace, GetNodeByPoint, GetEdgeByPoint

9.9.3 GetFaceContainingPoint

GetFaceContainingPoint — Hittar den yta som innehåller en punkt.

Synopsis

bigint **GetFaceContainingPoint**(text atopology, geometry apoint);

Description

Returnerar id för den yta som innehåller en punkt.
Ett undantag utlöses om punkten faller på en ytgräns.

Notera
Funktionen bygger på en giltig topologi med hjälp av kantlänkning och ytmärkning.

Tillgänglighet: 3.2.0

Se även

ST_GetFaceGeometry

9.9.4 GetNodeByPoint

GetNodeByPoint — Hittar nod-id för en nod vid en punktposition.

Synopsis

bigint **GetNodeByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

Hämtar id för en nod på en punktplats.

Funktionen returnerar ett heltal (id-node) givet en topologi, en POINT och en tolerans. Om tolerans = 0 betyder det exakt intersektion, annars hämtas noden från ett intervall.

Om en punkt inte korsar en nod returneras 0 (noll).

Om use tolerance > 0 och det finns mer än en nod nära punkten kastas ett undantag.

Notera
Om toleransen = 0 använder funktionen ST_Intersects, annars används ST_DWithin.

Utförs av GEOS-modulen.

Tillgänglighet: 2.0.0

Exempel

I dessa exempel används kanter som vi skapade i AddEdge

Code

```
SELECT topology.GetNodeByPoint('ma_topo', geom, 1) As nearnode
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
```

Output

```
2
```

This query errors because the tolerance matches two or more nodes.

Code

```
SELECT topology.GetNodeByPoint('ma_topo', geom, 1000) As too_much_tolerance
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
```

Output

```
ERROR:  Two or more nodes found
```

Se även

AddEdge, GetEdgeByPoint, GetFaceByPoint

9.9.5 GetTopologyID

GetTopologyID — Returnerar id för en topologi i tabellen topology.topology givet topologins namn.

Synopsis

integer **GetTopologyID**(varchar toponame);

Description

Returnerar id för en topologi i tabellen topology.topology givet topologins namn.

Tillgänglighet: 1.1

Exempel

Code

```
SELECT topology.GetTopologyID('ma_topo') AS topo_id;
```

Output

```
1
```

Se även

CreateTopology, DropTopology, GetTopologyName, GetTopologySRID

9.9.6 GetTopologySRID

GetTopologySRID — Returnerar SRID för en topologi i tabellen topology.topology med topologins namn.

Synopsis

integer **GetTopologyID**(varchar toponame);

Description

Returnerar det spatiala referens-id:t för en topologi i tabellen topology.topology givet namnet på topologin.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT topology.GetTopologySRID('ma_topo') AS srid;
```

Output

```
4326
```

Se även

CreateTopology, DropTopology, GetTopologyName, GetTopologyID

9.9.7 GetTopologyName

GetTopologyName — Returnerar namnet på en topologi (schema) givet topologins id.

Synopsis

varchar **GetTopologyName**(integer topology_id);

Description

Returnerar topologinamnet (schemat) för en topologi från tabellen topology.topology med topologins id.

Tillgänglighet: 1.1

Exempel

Code

```
SELECT topology.GetTopologyName(1) AS topo_name;
```

Output

```
ma_topo
```

Se även

CreateTopology, DropTopology, GetTopologyID, GetTopologySRID

9.9.8 ST_GetFaceEdges

ST_GetFaceEdges — Returnerar en uppsättning ordnade kanter som `avgränsar en yta.`.

Synopsis

getfaceedges_returntype **ST_GetFaceEdges**(varchar atopology, bigint aface);

Description

Returnerar en uppsättning ordnade kanter som `avgränsar en yta.` Varje utdata består av en sekvens och edgeid. Sekvensnummer börjar med värde 1.

Uppräkningen av varje ringkant börjar med kanten med den minsta identifieraren. Ordningen på kanterna följer en vänsterhands-regel (den bundna ytan är till vänster om varje riktad kant).

Tillgänglighet: 2.0

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3 Topo-Geo och Topo-Net 3: Rutinmässiga detaljer: X.3.5

Exempel

This query returns the edges that bound face 1.

Code

```
SELECT (topology.ST_GetFaceEdges('tt', 1)).*;
```

Output

```
sequence | edge
----------+------
        1 |   -4
        2 |    5
        3 |    7
        4 |   -6
        5 |    1
        6 |    2
        7 |    3
```

This query returns the sequence, edge identifier, and geometry of the edges that bound face 1. Use ST_GetFaceGeometry if only the geometry and sequence are needed.

Code

```
SELECT t.seq, t.edge, geom
FROM topology.ST_GetFaceEdges('tt', 1) As t(seq, edge)
    INNER JOIN tt.edge AS e ON abs(t.edge) = e.edge_id;
```

Se även

GetRingEdges, AddFace, ST_GetFaceGeometry

9.9.9 ST_GetFaceGeometry

ST_GetFaceGeometry — Returnerar polygonen i den angivna topologin med det angivna ytans id.

Synopsis

geometry **ST_GetFaceGeometry**(varchar atopology, bigint aface);

Description

Returnerar polygonen i den angivna topologin med det angivna ytans id. Bygger polygonen från de kanter som utgör ytan.

Tillgänglighet: 1.1

Denna metod implementerar SQL/MM-specifikationen. SQL-MM 3 Topo-Geo och Topo-Net 3: Rutinbeskrivningar: X.3.16

Exempel

This query returns the WKT of the polygon added with AddFace.

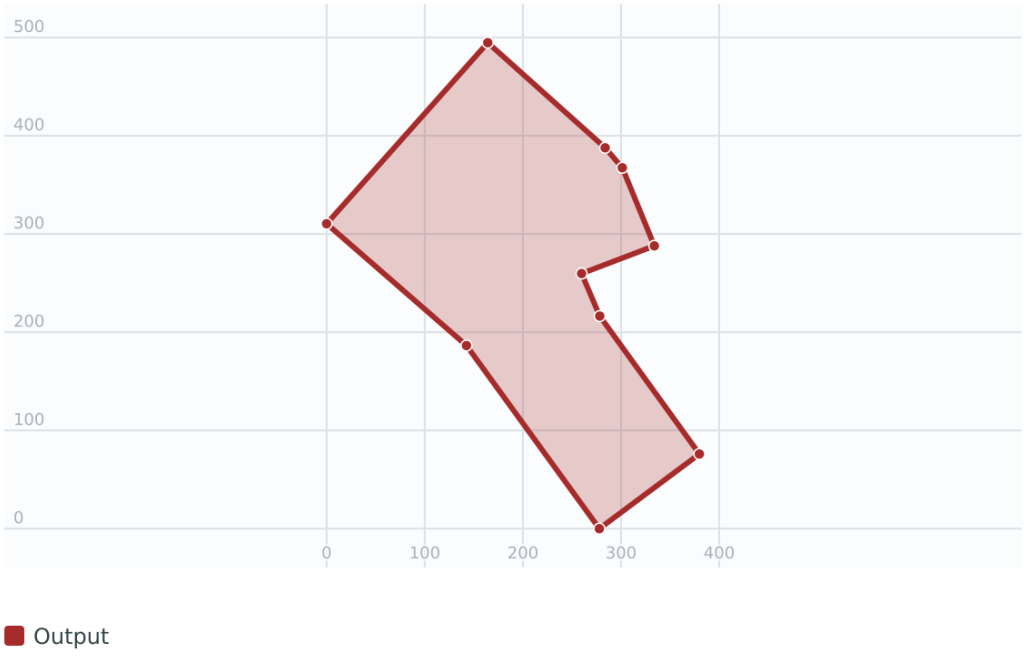
Code

```
SELECT topology.ST_GetFaceGeometry('ma_topo', 1) AS facegeomwkt;
```

Output

```
POLYGON((234776.9 899563.7,234896.5 899456.7,234914 899436.4,234946.6 899356.9,
234872.5 899328.7,234891 899285.4,234992.5 899145,234890.6 899069,
234755.2 899255.4,234612.7 899379.4,234776.9 899563.7))
```

Figure



Se även

AddFace

9.9.10 GetRingEdges

GetRingEdges — Returnerar den ordnade uppsättningen av signerade kantidentifierare som man möter när man går på en given kantsida.

Synopsis

getfaceedges_returntype **GetRingEdges**(varchar atopology, bigint aring, integer max_edges=null);

Description

Returnerar den ordnade uppsättningen signerade kantidentifierare som man möter när man går på en given kantsida. Varje utdata består av en sekvens och ett signerat kant-ID. Sekvensnummer börjar med värde 1.

Om du passerar ett positivt kant-ID börjar promenaden på vänster sida av motsvarande kant och följer kantriktningen. Om du passerar ett negativt kant-id startar promenaden på höger sida av den och går bakåt.

Om `max_edges` inte är null returneras inte fler än dessa poster av den funktionen. Detta är tänkt att vara en säkerhetsparameter vid hantering av eventuellt ogiltiga topologier.

Notera
Denna funktion använder metadata för kantringslänkning.

Tillgänglighet: 2.0.0

Se även

ST_GetFaceEdges, GetNodeEdges

9.9.11 GetNodeEdges

GetNodeEdges — Returnerar en ordnad uppsättning kanter som är kopplade till den angivna noden.

Synopsis

getfaceedges_returntype **GetNodeEdges**(varchar atopology, bigint anode);

Description

Returnerar en ordnad uppsättning kanter som är kopplade till den angivna noden. Varje utdata består av en sekvens och ett signerat kant-ID. Sekvensnummer börjar med värdet 1. En positiv kant börjar vid den givna noden. En negativ kant slutar i den givna noden. Slutna kanter kommer att visas två gånger (med båda tecknen). Ordningen är medurs med start från norrgående.

Notera
Denna funktion beräknar ordning i stället för att härleda från metadata och kan därför användas för att bygga kantringslänkning.

Tillgänglighet: 2.0

Se även

getfaceedges_returntype, GetRingEdges, ST_Azimuth

9.9.12 FindVertexSegmentPairsBelowDistance

FindVertexSegmentPairsBelowDistance — Find pairs of topology vertex/segment that are closer than tolerated distance

Synopsis

setof record **FindVertexSegmentPairsBelowDistance**(varchar atopology, float8 tolerance);

Description

Returns a relation in which each tuple is a pair of vertex/segment where the vertex is NOT an endpoint of the segment but the distance between them is below the given tolerance.

Availability: 3.7

Se även

ValidateTopologyPrecision

9.10 Bearbetning av topologi

9.10.1 Polygonize

Polygonize — Hittar och registrerar alla ytor som definieras av topologikanter.

Synopsis

text **Polygonize**(varchar toponame);

Description

Registrerar alla ytor som kan byggas upp av en topologis kantprimitiver.

Måltopologin antas inte innehålla några självskärande kanter.

Notera
Redan kända ytan känns igen, så det är säkert att anropa Polygonize flera gånger på samma topologi.

Notera
Denna funktion varken använder eller ställer in fälten next_left_edge och next_right_edge i kanttabellen.

Tillgänglighet: 2.0.0

Se även

AddFace, ST_Polygonize

9.10.2 AddNode

AddNode — Lägger till en punktnod i nodtabellen i det angivna topologischemat och returnerar den nya nodens nodid. Om punkten redan finns som nod returneras det befintliga nodid.

Synopsis

bigint **AddNode**(varchar toponame, geometry apoint, boolean allowEdgeSplitting=false, boolean computeContainingFace=false);

Description

Lägger till en punktnod i nodtabellen i det angivna topologischemat. Funktionen AddEdge lägger automatiskt till start- och slutpunkter för en kant när den anropas, så det är inte nödvändigt att uttryckligen lägga till noder för en kant.

Om en kant som korsar noden påträffas, utlöses antingen ett undantag eller så delas kanten, beroende på parametervärdet för `allowEdgeSplitting`.

Om `computeContainingFace` är true kommer en nytillagd nod att få rätt ”containing faceberäknad.

Notera
Om `apoint-geometrin` redan finns som en nod, läggs inte noden till utan det befintliga nodeid returneras.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT topology.AddNode('ma_topo', ST_GeomFromText('POINT(227641.6 893816.5)', 26986)) AS ↩
    nodeid;
```

Output

```
4
```

Se även

AddEdge, CreateTopology

9.10.3 AddEdge

AddEdge — Lägger till en linestringskant i tabellen edge och associerade start- och slutpunkter i tabellen point nodes i det angivna topologischemat med hjälp av den angivna linestringsgeometrin och returnerar edgeid för den nya (eller befintliga) kanten.

Synopsis

bigint **AddEdge**(varchar toponame, geometry aline);

Description

Lägger till en edge i edge-tabellen och associerade noder i nodes-tabellen i det angivna `toponame-schemat` med hjälp av den angivna geometrin för linestrings och returnerar edgeid för den nya eller befintliga posten. Den nyligen tillagda kanten har ”universe-yta på båda sidor och länkar till sig själv.

Notera
Om `aline-geometrin` korsar, överlappar, innehåller eller är innesluten av en befintlig linestringskant, så kastas ett fel och kanten läggs inte till.

Notera
Geometrin för `aline` måste ha samma `srid` som definierats för topologin, annars kommer ett ogiltigt sys-fel för spatial referens att kastas.

Utförs av GEOS-modulen.

Varning
AddEdge är föråldrad från och med 3.5.0. Använd TopoGeo_AddLineString istället.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575.8 893917.2,227591.9 ↩
    893900.4)', 26986)) AS edgeid;
```

Output

```
1
```

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.9 893900.4,227622.6 ↩
    893844.2,227641.6 893816.5,227704.5 893778.5)', 26986)) AS edgeid;
```

Output

```
2
```

This query errors because the edge intersects an existing edge away from the endpoints.

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.2 893900,227591.9 ↩
    893900.4,227704.5 893778.5)', 26986)) AS edgeid;
```

Output

```
ERROR:  Edge intersects (not on endpoints) with existing edge 1
```

Se även

TopoGeo_AddLineString, CreateTopology, Avsnitt 4.5

9.10.4 AddFace

AddFace — Registrerar en ytprimitiv till en topologi och hämtar dess identifierare.

Synopsis

bigint **AddFace**(varchar toponame, geometry apolygon, boolean force_new=false);

Description

Registrerar en ytprimitiv till en topologi och hämtar dess identifierare.

För en nytillagd yta kommer de kanter som utgör dess gränser och de som ingår i ytan att uppdateras så att de har korrekta värden i fälten left_face och right_face. Isolerade noder som ingår i ytan kommer också att uppdateras så att de får ett korrekt värde i fältet containing_face.

Notera
Denna funktion varken använder eller ställer in fälten next_left_edge och next_right_edge i kanttabellen.

Måltopologin antas vara giltig (innehåller inga självskärande kanter). Ett undantag uppstår om: Polygongränsen inte är helt definierad av befintliga kanter eller polygonen överlappar en befintlig yta.

Om `apolygongeometrin` redan finns som en yta, gäller följande: om `force_new` är false (standard) returneras den befintliga ytans id; om `force_new` är true tilldelas en ny id till den nyregistrerade ytan.

Notera
När en ny registrering av en befintlig yta utförs (force_new=true), kommer inga åtgärder att vidtas för att lösa hängande referenser till den befintliga ytan i tabellerna för edge, node och relation, och MBR-fältet i posten för den befintliga ytan kommer inte heller att uppdateras. Det är upp till den som anropar att hantera detta.

Notera
`Apolygongeometrin` måste ha samma `srid` som definierats för topologin, annars kommer ett ogiltigt sys-fel för spatial referens att uppstå.

Tillgänglighet: 2.0.0

Exempel

First add the edges using `generate_series` as an iterator. This query only works for polygons with fewer than 10000 points because of the maximum value in `generate_series`.

Code

```
SELECT topology.AddEdge('ma_topo', ST_MakeLine(ST_PointN(geom,i), ST_PointN(geom, i + 1) )) ↩
     As edgeid
FROM (
  SELECT ST_NPoints(geom) AS npt, geom
  FROM (
    SELECT ST_Boundary(ST_GeomFromText('POLYGON((234896.5 899456.7,234914 899436.4,234946.6 ↩
     899356.9,234872.5 899328.7,
                234891 899285.4,234992.5 899145, 234890.6 899069,234755.2 899255.4,
                234612.7 899379.4,234776.9 899563.7,234896.5 899456.7))', 26986) )  As geom
  ) AS geoms
) AS facen
CROSS JOIN generate_series(1,10000) AS i
WHERE i < npt;
```

Output

```
3
4
5
6
7
8
9
10
11
12
```

Then add the face.

Code

```
SELECT topology.AddFace(
    'ma_topo',
    ST_GeomFromText(
    'POLYGON((234896.5 899456.7,234914 899436.4,234946.6 899356.9,234872.5 899328.7,
    234891 899285.4,234992.5 899145, 234890.6 899069,234755.2 899255.4,
    234612.7 899379.4,234776.9 899563.7,234896.5 899456.7))',
    26986) ) As faceid;
```

Output

```
1
```

Se även

AddEdge, CreateTopology, Avsnitt 4.5

9.10.5 ST_Simplify

ST_Simplify — Returnerar en förenkladgeometriversion av den angivna TopoGeometry med hjälp av Douglas-Peucker-algoritmen.

Synopsis

geometry **ST_Simplify**(topogeometry tg, float8 tolerance);

Description

Returnerar en förenkladgeometrisk version av den angivna TopoGeometry med Douglas-Peucker-algoritmen på varje komponentkant.

Notera
Den returnerade geometrin kan vara icke-enkel eller icke-giltig.
Att dela upp komponentkanter kan bidra till att bibehålla enkelhet/validitet.

Utförs av GEOS-modulen.

Tillgänglighet: 2.1.0

Se även

Geometri ST_Simplify, ST_IsSimple, ST_IsValid, ST_ModEdgeSplit

9.10.6 RemoveUnusedPrimitives

RemoveUnusedPrimitives — Tar bort topologiprimitiver som inte behövs för att definiera befintliga TopoGeometry-objekt.

Synopsis

bigint **RemoveUnusedPrimitives**(text topology_name, geometry bbox);

Description

Hittar alla primitiver (noder, kanter, ytor) som inte är absolut nödvändiga för att representera befintliga TopoGeometry-objekt och tar bort dem, med bibehållen topologigiltighet (kantlänkning, ytmärkning) och TopoGeometry-rymdbeläggning.

Inga nya primitiva identifierare skapas, utan befintliga primitiver utvidgas till att omfatta sammanslagna ytor (när kanter tas bort) eller läkta kanter (när noder tas bort).

Tillgänglighet: 3.3.0

Se även

ST_ModEdgeHeal, ST_RemEdgeModFace

9.11 TopoGeometry Constructors

9.11.1 CreateTopoGeom

CreateTopoGeom — Skapar ett nytt topogeometriobjekt från en array av topoelement - tg_type: 1:[multi]punkt, 2:[multi]linje, 3:[multi]poly, 4:samling

Synopsis

topogeometry **CreateTopoGeom**(varchar toponame, integer tg_type, integer layer_id, topoelementarray tg_objs, bigint tg_id);
topogeometry **CreateTopoGeom**(varchar toponame, integer tg_type, integer layer_id);

Description

Skapar ett topogeometriobjekt för det lager som betecknas med `layer_id` och registrerar det i relationstabellen i `toponame-schemat`.

`tg_type` är ett heltal: 1:[multi]punkt (punctal), 2:[multi]linje (lineal), 3:[multi]poly (areal), 4:samling. `layer_id` är lagrets id i tabellen topology.layer.

punktlager bildas av en uppsättning noder, linjelager bildas av en uppsättning kanter, areallager bildas av en uppsättning ytor och samlingar kan bildas av en blandning av noder, kanter och ytor.

Om du utelämnar komponentuppsättningen genereras ett tomt TopoGeometry-objekt.

Tillgänglighet: 1.1

Exempel

Form from existing edges.

Skapa en topogeom i ri_topo-schema för lager 2 (våra ri_roads), av typ (2) LINE, för den första kanten (vi laddade i `ST_CreateTopoGeo`).

Code

```
INSERT INTO ri.ri_roads(road_name, topo)
VALUES (
    'Unknown',
    topology.CreateTopoGeom(
        'ri_topo',
        2,
        2,
        '{{1,2}}'::topology.topoelementarray
    )
);
```

Convert an areal geometry to best guess topogeometry.

Låt oss säga att vi har geometrier som ska bildas från en samling ytor. Vi har t.ex. tabellen blockgroups och vill veta topogeometrin för varje blockgrupp. Om våra data var perfekt inriktade skulle vi kunna göra detta:

Create the topogeometry column.

Code

```
SELECT topology.AddTopoGeometryColumn(
'topo_boston',
'boston', 'blockgroups', 'topo', 'POLYGON');
```

Output

```
1
```

Update the column assuming everything is perfectly aligned with the edges.

Code

```
UPDATE boston.blockgroups AS bg
SET topo = topology.CreateTopoGeom(
    'topo_boston',
    3,
    1,
    foo.bfaces
)
FROM (
    SELECT
        b.gid,
        topology.TopoElementArray_Agg(ARRAY[f.face_id, 3]) AS bfaces
    FROM boston.blockgroups AS b
    INNER JOIN topo_boston.face AS f
        ON b.geom && f.mbr
    WHERE ST_Covers(
        b.geom,
        topology.ST_GetFaceGeometry('topo_boston', f.face_id)
    )
    GROUP BY b.gid
) AS foo
WHERE foo.gid = bg.gid;
```

The world is rarely perfect, so this version allows some error. It counts a face if 50 percent of it falls within the expected block group boundary.

Code

```
WITH candidate_faces AS (
    SELECT
        b.gid,
        b.geom AS block_geom,
        f.face_id,
        topology.ST_GetFaceGeometry('topo_boston', f.face_id) AS face_geom
    FROM boston.blockgroups AS b
    INNER JOIN topo_boston.face AS f
        ON b.geom && f.mbr
),
block_faces AS (
    SELECT
        gid,
        topology.TopoElementArray_Agg(ARRAY[face_id, 3]) AS bfaces
    FROM candidate_faces
    WHERE ST_Covers(block_geom, face_geom)
        OR (
            ST_Intersects(block_geom, face_geom)
            AND ST_Area(ST_Intersection(block_geom, face_geom))

> ST_Area(face_geom) * 0.5
        )
    GROUP BY gid
)
UPDATE boston.blockgroups AS bg
SET topo = topology.CreateTopoGeom(
    'topo_boston',
    3,
```

```
    1,
    block_faces.bfaces
)
FROM block_faces
WHERE block_faces.gid = bg.gid;
```

To convert the topogeometry back to a denormalized geometry aligned with faces and edges, cast the topogeometry to a geometry. The resulting geometries are aligned with the TIGER street centerlines.

Code

```
UPDATE boston.blockgroups SET new_geom = topo::geometry;
```

Se även

AddTopoGeometryColumn, toTopoGeom ST_CreateTopoGeo, ST_GetFaceGeometry, TopoElementArray, TopoElementArray_Agg

9.11.2 toTopoGeom

toTopoGeom — Konverterar en enkel geometri till en topogeometri.

Synopsis

topogeometry **toTopoGeom**(geometry geom, varchar toponame, integer layer_id, float8 tolerance=-1);
topogeometry **toTopoGeom**(geometry geom, topogeometry topogeom, float8 tolerance=-1);

Description

Konverterar en enkel geometri till en TopoGeometry.

De topologiska primitiver som krävs för att representera indatageometrin läggs till i den underliggande topologin, eventuellt genom att dela upp befintliga primitiver, och de associeras med utdata TopoGeometry i `relationstabellen`.

Befintliga TopoGeometry-objekt (med möjligt undantag för `topogeom`, om det anges) kommer att behålla sina former.

When `tolerance` is given it will be used to snap the input geometry to existing primitives. The special value -1 (default) means use the topology precision as tolerance.

I det första formuläret skapas en ny TopoGeometry för det givna lagret`(layer_id) i` den givna topologin`(toponame)`.

I den andra formen kommer de primitiver som konverteringen resulterar i att läggas till den redan existerande TopoGeometry`(topogeom)`, och eventuellt lägga till utrymme i den slutliga formen. För att få den nya formen att helt ersätta den gamla, se clearTopoGeom.

Tillgänglighet: 2.0

Förbättrad: 2.1.0 lägger till versionen som tar en befintlig TopoGeometry.

Changed: 3.7.0 changes meaning of tolerance 0 to really mean 0 and -1 (new default) to request using the topology precision.

Exempel

Detta är ett helt fristående arbetsflöde

Create the topology without allowing any tolerance.

Code

```
SELECT topology.CreateTopology('topo_boston_test', 2249);
```

Create a new table.

Code

```
CREATE TABLE nei_topo(gid serial primary key, nei varchar(30));
```

Add a topogeometry column to the new table.

Code

```
SELECT topology.AddTopoGeometryColumn('topo_boston_test', 'public', 'nei_topo', 'topo', ' ↩
    MULTIPOLYGON') AS new_layer_id;
```

Output

```
1
```

Use the new layer identifier to populate the topogeometry column.

Code

```
INSERT INTO nei_topo(nei, topo)
SELECT nei, topology.toTopoGeom(geom, 'topo_boston_test', 1)
FROM neighborhoods
WHERE gid BETWEEN 1 and 15;
```

Verify what happened.

Code

```
SELECT * FROM
    topology.TopologySummary('topo_boston_test');
```

Output

```
Topology topo_boston_test (5), SRID 2249, precision 0
61 nodes, 87 edges, 35 faces, 15 topogeoms in 1 layers
Layer 1, type Polygonal (3), 15 topogeoms
 Deploy: public.nei_topo.topo
```

Shrink all TopoGeometry polygons by 10 meters.

Code

```
UPDATE nei_topo SET topo = toTopoGeom(ST_Buffer(topo, -10), clearTopoGeom(topo), 0);
```

Find faces that are left without any TopoGeometry reference after the shrink operation.

Code

```
SELECT ST_GetFaceGeometry('topo_boston_test', f.face_id)
  FROM topo_boston_test.face f
  WHERE f.face_id
> 0
  AND NOT EXISTS (
    SELECT * FROM topo_boston_test.relation
    WHERE layer_id = 1 AND element_id = f.face_id
  );
```

Se även

CreateTopology, AddTopoGeometryColumn, CreateTopoGeom, TopologySummary, clearTopoGeom

9.11.3 TopoElementArray_Agg

TopoElementArray_Agg — `Returnerar en topoelementarray` för en uppsättning element_id, typ arrayer (topoelements).

Synopsis

topoelementarray **TopoElementArray_Agg**(topoelement set tefield);

Description

Används för att skapa en TopoElementArray från en uppsättning av TopoElement.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT topology.TopoElementArray_Agg(ARRAY[e, t] ORDER BY e, t) As tea
FROM generate_series(1, 3) As e CROSS JOIN generate_series(1, 4) As t;
```

Output

```
{{1,1},{1,2},{1,3},{1,4},{2,1},{2,2},{2,3},{2,4},{3,1},{3,2},{3,3},{3,4}}
```

Se även

TopoElement, TopoElementArray

9.11.4 TopoElement

TopoElement — Konverterar en topogeometri till ett topoelement.

Synopsis

topoelement **TopoElement**(topogeometry topo);

Description

Omvandlar en TopoGeometry till en TopoElement.
Tillgänglighet: 3.4.0

Exempel

Detta är ett helt fristående arbetsflöde
Return the TopoElement representation of each topogeometry.

Code

```
SELECT TopoElement(topo)
FROM neighborhoods;
```

Use a cast to aggregate topogeometries into a TopoElementArray.

Code

```
SELECT topology.TopoElementArray_Agg(topo::topoelement)
FROM neighborhoods
GROUP BY city;
```

Se även

TopoElementArray_Agg, TopoGeometry, TopoElement

9.12 TopoGeometry-redigerare

9.12.1 clearTopoGeom

clearTopoGeom — Rensar innehållet i en topogeometri.

Synopsis

topogeometry **clearTopoGeom**(topogeometry topogeom);

Description

Rensar innehållet i TopoGeometry och förvandlar det till ett tomt. Används främst i kombination med toTopoGeom för att ersätta formen på befintliga objekt och alla beroende objekt i högre hierarkiska nivåer.
Tillgänglighet: 2.1

Exempel

Shrink all TopoGeometry polygons by 10 meters.

Code

```
UPDATE nei_topo SET topo = toTopoGeom(ST_Buffer(topo, -10), clearTopoGeom(topo), 0);
```

Se även

toTopoGeom

9.12.2 TopoGeom_addElement

TopoGeom_addElement — Lägger till ett element till definitionen av en TopoGeometry.

Synopsis

topogeometry **TopoGeom_addElement**(topogeometry tg, topoelement el);

Description

Lägger till en TopoElement till definitionen av ett TopoGeometry-objekt. Det blir inget fel om elementet redan är en del av definitionen.
Tillgänglighet: 2.3

Exempel

Add edge 5 to TopoGeometry `tg`.

Code

```
UPDATE mylayer SET tg = TopoGeom_addElement(tg, '{5,2}');
```

Se även

TopoGeom_remElement, CreateTopoGeom

9.12.3 TopoGeom_remElement

TopoGeom_remElement — Tar bort ett element från definitionen av en TopoGeometry.

Synopsis

topogeometry **TopoGeom_remElement**(topogeometry tg, topoelement el);

Description

Tar bort TopoElement från definitionen av ett TopoGeometry-objekt.
Tillgänglighet: 2.3

Exempel

Remove face 43 from TopoGeometry `tg`.

Code

```
UPDATE mylayer SET tg = TopoGeom_remElement(tg, '{43,3}');
```

Se även

TopoGeom_addElement, CreateTopoGeom

9.12.4 TopoGeom_addTopoGeom

TopoGeom_addTopoGeom — Lägger till element i en TopoGeometry till definitionen av en annan TopoGeometry.

Synopsis

topogeometry **TopoGeom_addTopoGeom**(topogeometry tgt, topogeometry src);

Description

Lägger till elementen i en TopoGeometry till definitionen av en annan TopoGeometry, eventuellt genom att ändra dess cachade typ (attributet type) till en samling, om det behövs för att innehålla alla element i källobjektet.

De två TopoGeometry-objekten måste definieras mot *samma* topologi och, om de är hierarkiskt definierade, måste de bestå av element i samma underordnade lager.

Tillgänglighet: 3.2

Exempel

Set an overall TopoGeometry value to be composed of all elements from specific TopoGeometry values.

Code

```
UPDATE mylayer SET tg_overall = TopoGeom_addTopoGeom(
    TopoGeom_addTopoGeom(
        clearTopoGeom(tg_overall),
        tg_specific1
    ),
    tg_specific2
);
```

Se även

TopoGeom_addElement, clearTopoGeom, CreateTopoGeom

9.12.5 toTopoGeom

toTopoGeom — Lägger till en geometrisk form till en befintlig topogeometri.

Description

Se toTopoGeom.

9.13 TopoGeometry-accessorer

9.13.1 GetTopoGeomElementArray

GetTopoGeomElementArray — `Returnerar en topoelementarray` (en array av topoelements) som innehåller de topologiska elementen och typen för den givna TopoGeometry (primitiva element).

Synopsis

topoelementarray **GetTopoGeomElementArray**(varchar toponame, integer layer_id, bigint tg_id);

topoelementarray **GetTopoGeomElementArray**(topogeometry tg);

Description

Returnerar en TopoElementArray som innehåller de topologiska elementen och typen för den givna TopoGeometry (primitiva element). Detta liknar GetTopoGeomElements, men returnerar elementen som en array i stället för som en datamängd.

tg_id är topogeometri-id för topogeometriobjektet i topologin i det lager som betecknas med `layer_id` i tabellen topology.layer.

Tillgänglighet: 1.1

Exempel

Se även

GetTopoGeomElements, TopoElementArray

9.13.2 GetTopoGeomElements

GetTopoGeomElements — Returnerar en uppsättning `topoelement-objekt` som innehåller topologiska element_id,element_type för den givna TopoGeometry (primitiva element).

Synopsis

setof topoelement **GetTopoGeomElements**(varchar toponame, integer layer_id, bigint tg_id);

setof topoelement **GetTopoGeomElements**(topogeometry tg);

Description

Returnerar en uppsättning element_id,element_type (topoelements) som motsvarar primitiva topologielement TopoElement (1: noder, 2: kanter, 3: ytor) som ett givet topogeometriobjekt i `toponame-schema` består av.

tg_id är topogeometri-id för topogeometriobjektet i topologin i det lager som betecknas med `layer_id` i tabellen topology.layer.

Tillgänglighet: 2.0.0

Exempel

Se även

GetTopoGeomElementArray, TopoElement, TopoGeom_addElement, TopoGeom_remElement

9.13.3 ST_SRID

ST_SRID — Returnerar den spatiala referensidentifieraren för en topogeometri.

Synopsis

integer **ST_SRID**(topogeometry tg);

Description

Returns the spatial reference identifier of the topology containing the TopoGeometry.

The identifier is recorded for the topology when it is created and refers to an entry in Avsnitt 4.5.

Tillgänglighet: 3.2.0

Exempel

Return the SRID of a TopoGeometry stored in the `ri_topo` topology.

Code

```
SELECT topology.ST_SRID(topo) AS srid
FROM ri.roads
WHERE road_name = 'Unknown';
```

Output

```
srid
------
 3438
```

Se även

Avsnitt 4.5, ST_SetSRID, ST_Transform, ST_SRID

9.14 TopoGeometry-utdata

9.14.1 AsGML

AsGML — Returnerar GML-representationen av en topogeometri.

Synopsis

text **AsGML**(topogeometry tg);
text **AsGML**(topogeometry tg, text nsprefix_in);
text **AsGML**(topogeometry tg, regclass visitedTable);
text **AsGML**(topogeometry tg, regclass visitedTable, text nsprefix);
text **AsGML**(topogeometry tg, text nsprefix_in, integer precision, integer options);
text **AsGML**(topogeometry tg, text nsprefix_in, integer precision, integer options, regclass visitedTable);
text **AsGML**(topogeometry tg, text nsprefix_in, integer precision, integer options, regclass visitedTable, text idprefix);
text **AsGML**(topogeometry tg, text nsprefix_in, integer precision, integer options, regclass visitedTable, text idprefix, int gmlversion);

Description

Returnerar GML-representationen av en topogeometri i version GML3-format. Om inget `nsprefix_in` anges används `gml`. Ange en tom sträng för nsprefix för att få en icke-kvalificerad namnrymd. Parametrarna precision (standard: 15) och options (standard 1), om de anges, skickas orörda till det underliggande anropet till ST_AsGML.

Parametern `visitedTable`, om den anges, används för att hålla reda på de besökta Node- och Edge-elementen så att korsreferenser (xlink:xref) kan användas i stället för att duplicera definitioner. Tabellen förväntas ha (minst) två heltalsfält: 'element_type' och 'element_id'. Den anropande användaren måste ha både läs- och skrivrättigheter för den angivna tabellen. För bästa prestanda bör ett index definieras på `element_type` och `element_id`, i nämnd ordning. Ett sådant index skapas automatiskt genom att en unik begränsning läggs till i fälten. Ett exempel:

Code

```
CREATE TABLE visited (
  element_type integer, element_id integer,
  unique(element_type, element_id)
);
```

Parametern `idprefix`, om den anges, kommer att läggas till Edge- och Node-taggidentifierare.

Parametern `gmlver`, om den anges, kommer att skickas till den underliggande ST_AsGML. Standardvärdet är 3.

Tillgänglighet: 2.0.0

Exempel

Här används den topogeometri som vi skapade i CreateTopoGeom

Code

```
SELECT topology.AsGML(topo) As rdgml
  FROM ri.roads
  WHERE road_name = 'Unknown';
```

Output

```
<gml:TopoCurve>
    <gml:directedEdge>
        <gml:Edge gml:id="E1">
            <gml:directedNode orientation="-">
                <gml:Node gml:id="N1"/>
            </gml:directedNode>
            <gml:directedNode
></gml:directedNode>
            <gml:curveProperty>
                <gml:Curve srsName="urn:ogc:def:crs:EPSG::3438">
                    <gml:segments>
                        <gml:LineStringSegment>
                            <gml:posList srsDimension="2"
>384744 236928 384750 236923 384769 236911 384799 236895 384811 236890
                            384833 236884 384844 236882 384866 236881 384879 236883 384954 236898 385087 236932 385117 ↩
      236938
                            385167 236938 385203 236941 385224 236946 385233 236950 385241 236956 385254 236971
                            385260 236979 385268 236999 385273 237018 385273 237037 385271 237047 385267 237057 385225 ↩
      237125
                            385210 237144 385192 237161 385167 237192 385162 237202 385159 237214 385159 237227 385162 ↩
      237241
                            385166 237256 385196 237324 385209 237345 385234 237375 385237 237383 385238 237399 385236 ↩
      237407
                            385227 237419 385213 237430 385193 237439 385174 237451 385170 237455 385169 237460 385171 ↩
      237475
                            385181 237503 385190 237521 385200 237533 385206 237538 385213 237541 385221 237542 385235 ↩
      237540 385242 237541
                            385249 237544 385260 237555 385270 237570 385289 237584 385292 237589 385291 237596 385284 ↩
      237630</gml:posList>
                        </gml:LineStringSegment>
                    </gml:segments>
                </gml:Curve>
            </gml:curveProperty>
        </gml:Edge>
    </gml:directedEdge>
</gml:TopoCurve>
```

Samma övning som tidigare utan namnrymd

Code

```
SELECT topology.AsGML(topo,'') As rdgml
  FROM ri.roads
  WHERE road_name = 'Unknown';
```

Output

```
<TopoCurve>
    <directedEdge>
        <Edge id="E1">
            <directedNode orientation="-">
                <Node id="N1"/>
            </directedNode>
            <directedNode
></directedNode>
            <curveProperty>
                <Curve srsName="urn:ogc:def:crs:EPSG::3438">
                    <segments>
                        <LineStringSegment>
                            <posList srsDimension="2"
>384744 236928 384750 236923 384769 236911 384799 236895 384811 236890
                            384833 236884 384844 236882 384866 236881 384879 236883 384954 236898 385087 236932 385117 ↩
      236938
                            385167 236938 385203 236941 385224 236946 385233 236950 385241 236956 385254 236971
                            385260 236979 385268 236999 385273 237018 385273 237037 385271 237047 385267 237057 385225 ↩
      237125
                            385210 237144 385192 237161 385167 237192 385162 237202 385159 237214 385159 237227 385162 ↩
      237241
                            385166 237256 385196 237324 385209 237345 385234 237375 385237 237383 385238 237399 385236 ↩
      237407
```

```
                            385227 237419 385213 237430 385193 237439 385174 237451 385170 237455 385169 237460 385171 ↩
      237475
                            385181 237503 385190 237521 385200 237533 385206 237538 385213 237541 385221 237542 385235 ↩
      237540 385242 237541
                            385249 237544 385260 237555 385270 237570 385289 237584 385292 237589 385291 237596 385284 ↩
      237630</posList>
                        </LineStringSegment>
                    </segments>
                </Curve>
            </curveProperty>
        </Edge>
    </directedEdge>
</TopoCurve>
```

Se även

CreateTopoGeom, ST_CreateTopoGeo

9.14.2 AsTopoJSON

AsTopoJSON — Returnerar TopoJSON-representationen av en topogeometri.

Synopsis

text **AsTopoJSON**(topogeometry tg, regclass edgeMapTable);

Description

Returnerar TopoJSON-representationen av en topogeometri. Om `edgeMapTable` inte är null kommer den att användas som en mappning för uppslagning/lagring av kantidentifierare till bågindex. Detta för att möjliggöra en kompakt ärcs-matris i det slutliga dokumentet.

Om tabellen anges förväntas den ha ett ärc_id-fält av typen serialöch ett ”edge_idäv typen integer; koden kommer att fråga efter ”edge_id”i tabellen, så det rekommenderas att lägga till ett index på det fältet.

Notera
Arc-index i TopoJSON-utdata är 0-baserade men de är 1-baserade i tabellen ”edgeMapTable”.

Ett fullständigt TopoJSON-dokument måste, utöver de utdrag som returneras av denna funktion, innehålla de faktiska bågarna plus några rubriker. Se specifikationen för TopoJSON.

Tillgänglighet: 2.1.0

Förbättrad: 2.2.1 har lagt till stöd för puntal-indata

Se även

ST_AsGeoJSON

Exempel

Create a temporary edge map for the TopoJSON output.

Code

```
CREATE TEMP TABLE edgemap(arc_id serial, edge_id int unique);
```

Build the TopoJSON header, objects, arcs, and footer.

Code

```
SELECT '{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects ↩
    ": {'

UNION ALL SELECT '"' || feature_name || '": ' || AsTopoJSON(feature, 'edgemap')
FROM features.big_parcels WHERE feature_name = 'P3P4';

WITH edges AS (
  SELECT m.arc_id, e.geom FROM edgemap m, city_data.edge e
  WHERE e.edge_id = m.edge_id
), points AS (
  SELECT arc_id, (ST_DumpPoints(geom)).* FROM edges
), compare AS (
  SELECT p2.arc_id,
         CASE WHEN p1.path IS NULL THEN p2.geom
              ELSE ST_Translate(p2.geom, -ST_X(p1.geom), -ST_Y(p1.geom))
         END AS geom
  FROM points p2 LEFT OUTER JOIN points p1
  ON ( p1.arc_id = p2.arc_id AND p2.path[1] = p1.path[1]+1 )
  ORDER BY arc_id, p2.path
), arcsdump AS (
  SELECT arc_id, (regexp_matches(ST_AsGeoJSON(geom), '\[.*\]'))[1] as t
  FROM compare
), arcs AS (
  SELECT arc_id, '[' || array_to_string(array_agg(t), ',') || ']' as a FROM arcsdump
  GROUP BY arc_id
  ORDER BY arc_id
)
SELECT '}, "arcs": [' UNION ALL
SELECT array_to_string(array_agg(a), E',\n') from arcs

UNION ALL SELECT ']}'::text as t;
```

Output

```
{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects": {
"P3P4": { "type": "MultiPolygon", "arcs": [[[-1]],[[6,5,-5,-4,-3,1]]]}
}, "arcs": [
 [[25,30],[6,0],[0,10],[-14,0],[0,-10],[8,0]],
 [[35,6],[0,8]],
 [[35,6],[12,0]],
 [[47,6],[0,8]],
 [[47,14],[0,8]],
 [[35,22],[12,0]],
 [[35,14],[0,8]]
 ]}
```

9.15 Spatiala relationer mellan topologier

9.15.1 Equals

Equals — Returnerar true om två topogeometrier består av samma topologiprimitiver.

Synopsis

boolean **Equals**(topogeometry tg1, topogeometry tg2);

Description

Returnerar true om två topogeometrier består av samma topologiprimitiver: ytor, kanter, noder.

Notera
Denna funktion stöds inte för topogeometrier som är geometrisamlingar. Den kan inte heller jämföra topogeometrier från olika topologier.

Tillgänglighet: 1.1.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

The following example uses the `nei_topo` table created in the toTopoGeom examples.

Code

```
SELECT topology.Equals(a.topo, b.topo)
FROM nei_topo AS a
JOIN nei_topo AS b ON b.gid = a.gid
WHERE a.gid = 1;
```

Output

```
t
```

Se även

GetTopoGeomElements, ST_Equals

9.15.2 Intersects

Intersects — Returnerar true om något par av primitiver från de två topogeometrierna korsar varandra.

Synopsis

boolean **Intersects**(topogeometry tg1, topogeometry tg2);

Description

Returnerar true om något par av primitiver från de två topogeometrierna korsar varandra.

Notera
Denna funktion stöds inte för topogeometrier som är geometrisamlingar. Den kan inte heller jämföra topogeometrier från olika topologier. För närvarande stöds inte heller hierarkiska topogeometrier (topogeometrier som består av andra topogeometrier).

Tillgänglighet: 1.1.0

Denna funktion stöder 3d och kommer inte att tappa z-index.

Exempel

The following example uses the `nei_topo` table created in the toTopoGeom examples.

Code

```
SELECT topology.Intersects(a.topo, b.topo)
FROM nei_topo AS a
JOIN nei_topo AS b ON b.gid = a.gid
WHERE a.gid = 1;
```

Output

```
t
```

Se även

ST_Intersects

9.16 Importera och exportera topologier

När du har skapat topologier och kanske tillhörande topologiska lager kanske du vill exportera dem till ett filbaserat format för säkerhetskopiering eller överföring till en annan databas.

Att använda standardverktygen för dumpning / återställning av PostgreSQL är problematiskt eftersom topologier består av en uppsättning tabeller (4 för primitiver, ett godtyckligt antal för lager) och poster i metadatatabeller (topology.topology och topology.layer). Dessutom är topologiidentifierare inte univoque över databaser så att parametern för din topologi måste ändras när du återställer den.

För att förenkla export/återställning av topologier tillhandahålls ett par körbara program: `pgtopo_export` och `pgtopo_import`. Exempel på användning:

Code

```
pgtopo_export dev_db topo1 | pgtopo_import topo1 | psql staging_db
```

9.16.1 Använda topologiexportören

Skriptet `pgtopo_export` tar namnet på en databas och en topologi och matar ut en dumpfil som kan användas för att importera topologin (och tillhörande lager) till en ny databas.

Som standard skriver `pgtopo_export` dumpfilen till standardutdata så att den kan pipas till `pgtopo_import` eller omdirigeras till en fil (vägrar att skriva till terminalen). Du kan ange ett filnamn för utdata med kommandoradsalternativet `-f.`.

Som standard inkluderar `pgtopo_export` en dumpning av alla lager som definierats mot den givna topologin. Detta kan vara mer data än du behöver, eller kanske inte fungerar (om dina lagertabeller har komplexa beroenden), i vilket fall du kan begära att lagren hoppas över med omkopplaren `--skip-layers` och hantera dem separat.

Om `pgtopo_export an` ropas med kommandot `--help` (eller `-h` förkortat) skrivs alltid en kort användningssträng ut.

Dumpfilformatet är ett komprimerat tar-arkiv med en `pgtopo_export-katalog` som innehåller minst en `pgtopo_dump_version-fil` med information om formatversion. Från och med version 1 innehåller katalogen tabbavgränsade CSV-filer med data från de primitiva tabellerna för topologi (nod, kantdata, yta, relation), topologi- och lagerposterna som är associerade med den och (om inte `--skip-layers` ges) en PostgreSQL-dump i anpassat format av tabeller som rapporteras som lager i den givna topologin.

9.16.2 Använda topologiimportören

Skriptet `pgtopo_import` tar en topologidump i `pgtopo_export-format` och ett namn på den topologi som ska skapas och matar ut ett SQL-skript som rekonstruerar topologin och tillhörande lager.

Den genererade SQL-filen kommer att innehålla satser som skapar en topologi med det angivna namnet, laddar primitiva data i den, återställer och registrerar alla topologilager genom att korrekt länka alla TopoGeometry-värden till deras korrekta topologi.

Som standard läser `pgtopo_import` dumpningen från standardinmatningen så att den kan användas tillsammans med `pgtopo_export` i en pipeline. Du kan eventuellt ange ett filnamn för indata med kommandoradsalternativet `-f`.

Som standard inkluderar `pgtopo_import` i SQL-filen koden för att återställa alla lager som finns i dumpningen.

Detta kan vara oönskat eller inte fungera om din måldatabas redan har tabeller med samma namn som de i dumpningen. I så fall kan du begära att lagren hoppas över med `--skip-layers` och hantera dem separat (eller senare).

SQL för att endast läsa in och länka lager till en namngiven topologi kan genereras med hjälp av omkopplaren `--only-layers`. Detta kan vara användbart för att läsa in lager EFTER att namnkonflikterna har lösts eller för att länka lager till en annan topologi (t.ex. en spatialt förenklad version av starttopologin).

Om måltopologin redan finns och du vill att den ska tas bort i förväg kan du använda kommandot `--drop-topology` (sedan PostGIS-3.6.0).

Kapitel 10

Rasterdatahantering, frågor och applikationer

10.1 Läs in och skapa raster

I de flesta användningsfall skapar du PostGIS-raster genom att ladda befintliga rasterfiler med hjälp av den paketerade rasterladdaren `raster2pgsql`.

10.1.1 Använda raster2pgsql för att läsa in raster

`Raster2pgsql` är en körbar rasterläsa inre som läsa inr GDAL-stödda rasterformat till SQL som lämpar sig för laddning i en PostGIS-rastertabell. Den kan läsa in mappar med rasterfiler samt skapa översikter över raster.

Eftersom raster2pgsql oftast kompileras som en del av PostGIS (om du inte kompilerar ditt eget GDAL-bibliotek), kommer de rastertyper som stöds av den körbara filen att vara desamma som de som kompileras i GDAL-beroendebiblioteket. För att få en lista över rastertyper som just din `raster2pgsql` stöder, använd `-G-omkopplaren`.

Notera
När du skapar översikter över en viss faktor från en uppsättning raster som är anpassade är det möjligt att översikterna inte är anpassade. Besök http://trac.osgeo.org/postgis/ticket/1764 för ett exempel där översikterna inte är i linje.

10.1.1.1 Exempel på användning

Ett exempel på en session där laddaren används för att skapa en inmatningsfil och ladda upp den i 100x100 bitar kan se ut så här:

The loader assigns SRID 4326, creates an index and standard constraints, analyzes the loaded table, records source filenames, and stores the input rasters in 100 by 100 pixel tiles.

Code

```
raster2pgsql -s 4326 -I -C -M -F -t 100x100 *.tif public.demelevation
> elev.sql
psql -d gisdb -f elev.sql
```

Notera
Om du inte anger schemat som en del av måltabellens namn kommer tabellen att skapas i standardschemat för den databas eller användare som du ansluter till.

En konvertering och uppladdning kan göras i ett steg med hjälp av UNIX pipes:

Code

```
raster2pgsql -s 4326 -I -C -M -F -t 100x100 \
  *.tif public.demelevation |
  psql -d gisdb
```

Läs in raster Massachusetts statliga planmätare flygplattor i ett schema som heter `aerial` och skapa en fullständig vy, 2 och 4 nivå översiktstabeller, använd kopieringsläge för att infoga (ingen mellanliggande fil bara direkt till db) och -e tvingar inte allt i en transaktion (bra om du vill se data i tabeller direkt utan att vänta). Dela upp rastren i 128x128 pixelplattor och tillämpa rasterbegränsningar. Använd kopieringsläge i stället för tabellinsättning. (-F) Inkludera ett fält som heter filename för att innehålla namnet på filen som plattorna klipptes ut från.

Code

```
raster2pgsql -I -C -e -Y -F -s 26986 -t 128x128 -l 2,4 \
  bostonaerials2008/*.jpg aerials.boston |
  psql -U postgres -d gisdb -h localhost -p 5432
```

List the raster formats supported by this build:

Code

```
raster2pgsql -G
```

The `-G` command outputs a list something like:

Output

```
Available GDAL raster formats:
  Virtual Raster
  GeoTIFF
  National Imagery Transmission Format
  Raster Product Format TOC format
  ECRG TOC format
  Erdas Imagine Images (.img)
  CEOS SAR Image
  CEOS Image
  ...
  Arc/Info Export E00 GRID
  ZMap Plus Grid
  NOAA NGS Geoid Height Grids
```

10.1.1.2 raster2pgsql-alternativ

Long options that take values also accept the `--option=value` spelling.

`-?, --help` Visa hjälpskärm. Hjälp visas också om du inte skickar in några argument.

`-G, --gdal-formats` Skriv ut de rasterformat som stöds.

(c|a|d|p) Dessa är ömsesidigt uteslutande alternativ:

`-c` Skapa en ny tabell och fyll i den med raster, *detta är standardläget*

-a Lägg till raster till en befintlig tabell.

-d Ta bort tabell, skapa en ny och fyll den med raster(s)

-p Förbered läge, skapa bara bordet.

--if-not-exists Use `IF NOT EXISTS` for table creation in `-c` and `-p` modes. When `-I` or `--create-index` is also specified, use `IF NOT EXISTS` for index creation too. This option cannot be used with `-d`. Append mode requires an explicit creation action.

Atomic loader actions The short modes are presets over explicit actions. The equivalent long options are `--drop-table`, `--create-table`, `--load-data`, `--create-index`, `--add-constraints`, `--vacuum`, `--analyze`, and `--no-transaction`. Add `--if-not-exists` to make active creation actions use `IF NOT EXISTS`. `--drop-table` emits `DROP TABLE IF EXISTS` before the selected actions. With no mode specified, the default create/load actions still apply.

Rasterbearbetning: Tillämpa begränsningar för korrekt registrering i rasterkataloger

-C, --add-constraints Tillämpa rasterbegränsningar - srid, pixelstorlek etc. för att säkerställa att raster är korrekt registrerat i vyn `raster_columns`.

-x, --no-extent Disable setting the max extent constraint. Only applied if -C or --add-constraints is also used.

-r, --regular-blocking Set the constraints (spatially unique and coverage tile) for regular blocking. Only applied if -C or --add-constraints is also used.

Rasterbearbetning: Valfria parametrar som används för att manipulera inmatad rasterdataup

-s, --srid [<FROM_SRID>:]<SRID> Assign the output raster to the specified SRID. When FROM_SRID is also provided, raster2pgsql marks the input raster with FROM_SRID and emits SQL that reprojects the raster to the target SRID. If the source SRID is omitted or is zero, the raster metadata will be checked to determine an appropriate source SRID. Reprojection cannot be used with copy mode.

-b, --band BAND Index (1-baserat) för det band som ska extraheras från rastret. För mer än ett bandindex, separera med kommatecken (,). Om inget anges kommer alla band i rastret att extraheras.

-t, --tile-size TILE_SIZE Skär rastret i brickor som ska infogas, en per tabellrad. `TILE_SIZE` uttrycks som WIDTHxHEIGHT eller sätts till värdet äutoför att låta inläsaren beräkna en lämplig brickstorlek med hjälp av det första rastret och tillämpas på alla raster.

-P, --pad Fyll på längst till höger och längst ner för att garantera att alla brickor har samma bredd och höjd.

-R, --register Registrera rastret som ett filsystemraster (out-db).

Endast metadata för rastret och sökvägen till rastret lagras i databasen (inte pixlarna).

-l, --overview-factor OVERVIEW_FACTOR Create overview of the raster. For more than one factor, separate with comma(,). Overview table name follows the pattern o_`overview factor`_`table` where `overview factor` is a placeholder for numerical overview factor and `table` is replaced with the base table name. If a source raster already has an overview with matching dimensions, that overview is reused. Created overview is stored in the database and is not affected by -R. Note that your generated sql file will contain both the main table and overview tables.

-N, --nodata NODATA NODATA-värde som ska användas på band utan NODATA-värde.

Valfria parametrar som används för att manipulera databasobjekt

-f, --raster-column COLUMN Ange namn på målrasterkolumn, standard är 'rast'

-F, --filename Lägg till en kolumn med namnet på filen

-n, --filename-column COLUMN Ange namnet på kolumnen för filnamn. Innebär -F.

-q, --quote Packa in PostgreSQL-identifierare i citat.

-I Alias for `--create-index`.

--create-index Create a GiST index on the raster column at the end of this raster2pgsql run. With repeated `-a` append runs, create the index on the final run or after loading; add `--if-not-exis` to make reruns tolerate an existing index.

-M Vakuumanalysera rastertabellen.

-k, --skip-nodata-check Behåller tomma plattor och hoppar över NODATA-värdeskontroller för varje rasterband. Observera att du sparar tid på att kontrollera, men kan sluta med mycket fler skräprader i din databas och dessa skräprader markeras inte som tomma rutor.

-T, --tablespace tablespace Ange tablespace för den nya tabellen. Observera att index (inklusive primärnyckeln) fortfarande kommer att använda standardtabellutrymmet om inte flaggan -X också används.

-X, --index-tablespace tablespace Ange tablespace för tabellens nya index. Detta gäller för primärnyckeln och det spatiala indexet om flaggan -I används.

-Y, --copy [MAX_ROWS_PER_COPY] Använd copy-satser i stället för insert-satser. Ange valfritt `max_rows_per_copy`; standard 50 om inget anges.

-e, --no-transaction Utför varje villkor individuellt, använd inte en transaktion.

-E, --endian ENDIAN Styr endianness för genererad binär utdata av raster; ange 0 för XDR och 1 för NDR (standard); endast NDR-utdata stöds nu

-V, --wkb-version version Ange version av utdataformat. Standard är 0. Endast 0 stöds för närvarande.

10.1.2 Skapa raster med hjälp av PostGIS rasterfunktioner

Vid många tillfällen vill du skapa raster och rastertabeller direkt i databasen. Det finns en uppsjö av funktioner för att göra det. De allmänna stegen att följa.

1. Skapa en tabell med en rasterkolumn för att hålla de nya rasterposterna, vilket kan åstadkommas med:

   Code

   ```
   CREATE TABLE myrasters(rid serial primary key, rast raster);
   ```

2. Det finns många funktioner som hjälper till med det målet. Om du skapar raster som inte är ett derivat av andra raster, bör du börja med: ST_MakeEmptyRaster, följt av ST_AddBand

   Du kan också skapa raster från geometrier. För att uppnå det vill du använda ST_AsRaster kanske tillsammans med andra funktioner som ST_Union eller ST_MapAlgebraFct eller någon av familjen av andra map algebra-funktioner.

   Det finns även många fler alternativ för att skapa nya rastertabeller från befintliga tabeller. Du kan t.ex. skapa en rastertabell i en annan projektion än en befintlig med hjälp av ST_Transform

3. När du är klar med att fylla i din tabell initialt vill du skapa ett spatialt index på rasterkolumnen med något liknande:

   Code

   ```
   CREATE INDEX myrasters_rast_st_convexhull_idx ON myrasters USING gist(ST_ConvexHull( ↩
       rast));
   ```

   Observera användningen av ST_ConvexHull eftersom de flesta rasteroperatorer baseras på rastrernas konvexa skrov.

Notera
Pre-2.0-versioner av PostGIS-raster baserades på kuvertet snarare än det konvexa skrovet. För att de spatiala indexen ska fungera korrekt måste du ta bort dem och ersätta dem med index baserade på konvexa skrov.

4. Tillämpa rasterbegränsningar med hjälp av AddRasterConstraints

10.1.3 Använda öut db-molnraster

Verktyget `raster2pgsql` använder GDAL för att få åtkomst till rasterdata och kan dra nytta av en viktig GDAL-funktion: möjligheten att läsa från raster som lagras på distans i molnbaserade öbjektlager”(t.ex. AWS S3, Google Cloud Storage).

Effektiv användning av molnlagrade raster kräver att man använder ett molnoptimeratformat. Det mest välkända och använda är detmolnoptimerade GeoTIFF-formatet. Om du använder ett format som inte är molnbaserat, t.ex. JPEG eller en TIFF som inte är kaklad, får du mycket dåliga prestanda eftersom systemet måste ladda ner hela rastret varje gång det behöver komma åt en delmängd.

Först laddar du ditt raster i den molnlagring du väljer. När den har laddats kommer du att ha en URI för att komma åt den, antingen en ”http-URI eller ibland en URI som är specifik för tjänsten. (t.ex. s3://bucket/object”). Om du vill komma åt icke-offentliga buckets måste du ange GDAL-konfigurationsalternativ för att autentisera din anslutning. Observera att det här kommandot *läser* från molnrastret och *skriver* till databasen.

Code

```
AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxxxx \
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx \
raster2pgsql \
  -s 990000 \
  -t 256x256 \
  -I \
  -R \
  /vsis3/your.bucket.com/your_file.tif \
  your_table \
  | psql your_db
```

När tabellen har laddats måste du ge databasen behörighet att läsa från fjärraster genom att ange två behörigheter, postgis.enable_outdb_rasters och postgis.gdal_enabled_drivers.

Code

```
SET postgis.enable_outdb_rasters = true;
SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

För att ändringarna ska bli bestående kan du göra dem direkt i din databas. Du måste återansluta för att uppleva de nya inställningarna.

Code

```
ALTER DATABASE your_db SET postgis.enable_outdb_rasters = true;
ALTER DATABASE your_db SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

För icke-publika raster kan du behöva ange åtkomstnycklar för att läsa från molnrastern. Samma nycklar som du använde för att skriva `raster2pgsql-anropet` kan ställas in för användning i databasen med postgis.gdal_vsi_options -konfigurationen. Observera att flera alternativ kan ställas in genom att mellanslag separerar `nyckel=värde-paren`.

Code

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
```

När du har laddat data och ställt in behörigheter kan du interagera med rastertabellen som med vilken annan rastertabell som helst och använda samma funktioner. Databasen hanterar all mekanik för att ansluta till molndata när den behöver läsa pixeldata.

10.2 Raster Coverages and Raster/Vector Analysis

A coverage is a way to interpret spatial data as a mapping from a *domain* of locations to a *range* of values. Evaluating a coverage means asking for the value at a location; an inverse query asks for the locations whose values match a condition. PostGIS does not expose a generic ISO 19123 coverage SQL type, but geometry tables and raster values provide the building blocks for both directions.

A coverage can be continuous, with values calculated or interpolated for arbitrary locations, or discrete, with values supplied by stored spatial objects or grid cells. The coverage model describes how locations map to values, independently of whether those values come from an equation, a table, or one or more raster values. A time column or function argument can add a temporal dimension without changing that distinction.

In a vector coverage, a geometry column supplies the domain and other columns supply attributes in the range. A vector range can contain text, numbers, or another spatial value, provided its meaning and type are consistent wherever the coverage is defined. In a raster, each cell location is part of the domain and the band values at that cell form the range. Other columns in the same table row describe the raster value or tile as a whole; they do not attach separate values to individual cells.

A logical raster coverage may be stored as one raster value or as many tiles in a table. Tiles that belong to one coverage should have compatible SRIDs, scales, alignment, band counts, pixel types, and NODATA conventions. The constraints exposed by Avsnitt 10.3.1 record and validate those properties.

Raster table arrangements. A raster column can represent an image warehouse, a tiled coverage, or a collection of rasterized objects. These arrangements have different assumptions about tile size, alignment, gaps, and overlaps. PostGIS does not infer the intended arrangement from the column type alone.

Arrangement	Row relationship	Typical use
Image warehouse	Independent rasters; sizes, alignment, and extents may differ or overlap	Cataloging unrelated source images
Irregular tiled coverage	One logical coverage with gaps allowed, variable tile sizes, and normally no overlap	Incremental or source-driven mosaics
Regular tiled coverage	One alignment and tile size with no overlap; gaps may remain at the boundary or inside the coverage	Indexed analysis over a large coverage
Rectangular regular coverage	Equal aligned tiles fill a rectangular extent without gaps or overlaps	Predictable block processing and overview generation
Separate tiled images	Each image is internally regular but stored in its own table or partition	Keeping source-image identity and lifecycle separate
Raster-object coverage	Each row is a rasterized feature; extents and sizes may differ and overlap	Object models where cells replace or supplement vector geometry

Use AddRasterConstraints to record common SRID, scale, alignment, band, extent, and regular-blocking properties, and use spatial indexes and application constraints for relationships that the catalog cannot express.

Queries against a tiled coverage normally have two stages:

1. Use the raster spatial index and ST_Intersects to select candidate tiles.
2. Read, clip, aggregate, or vectorize values inside those tiles with functions such as ST_Value, ST_Clip, ST_SummaryStatsAgg, and ST_DumpAsPolygons.

ST_DumpAsPolygons returns geomval rows that pair geometries with cell values. This allows ordinary geometry overlay and filtering to answer inverse coverage queries, but polygonizing many cells is usually more expensive than keeping the operation in raster space.

Choose boundary semantics deliberately. Clipping and aggregating raster pixels is usually the shorter and faster path. If a partial boundary cell must contribute in proportion to its covered area, intersect the geometry with the geomval rows and calculate an area-weighted result in an appropriate projected coordinate system. That exact partial-cell method costs more because it vectorizes the selected cells.

Tile size affects both sides of the indexed query: smaller tiles reduce the amount of raster data inspected for each candidate row, while increasing row and index overhead. Benchmark representative queries instead of treating one tile size as a universal default.

For a current end-to-end raster/vector pattern, load and tile data with Avsnitt 10.1.1, then see the building example in ST_SummaryStats. It selects intersecting tiles, clips them to each polygon, and combines per-tile statistics into one result per feature. For point observations such as animal locations, first buffer the points in an appropriate projected coordinate system, then apply the same tile selection, clipping, and aggregation pattern to an elevation or other continuous raster coverage.

10.3 Rasterkataloger

Det finns två rasterkatalogvyer som levereras med PostGIS. Båda vyerna använder information som är inbäddad i begränsningarna i rastertabellerna. Som ett resultat är katalogvyerna alltid konsekventa med rasterdata i tabellerna eftersom begränsningarna verkställs.

1. `raster_columns` Denna vy katalogiserar alla rastertabellkolumner i databasen.
2. `raster_overviews` Denna vy katalogiserar alla rastertabellkolumner i din databas som fungerar som översikter för en mer finkornig tabell. Tabeller av den här typen genereras när du använder växeln `-l` under inläsning.

10.3.1 Rasterkolumner Katalog

`Raster_columns` är en katalog över alla kolumner i rastertabeller i databasen som är av typen raster. Det är en vy som utnyttjar begränsningarna i tabellerna så att informationen alltid är konsekvent även om du återställer en rastertabell från en säkerhetskopia av en annan databas. Följande kolumner finns i katalogen `raster_columns`.

Om du inte skapade dina tabeller med inläsningsprogrammet eller glömde att ange flaggan `-C` under inläsningen kan du genomdriva begränsningarna i efterhand med hjälp av AddRasterConstraints så att `raster_columns-katalogen` registrerar den gemensamma informationen om dina rasterplattor.

- `r_table_catalog` Den databas som tabellen finns i. Detta kommer alltid att läsa den aktuella databasen.

- `r_table_schema` Det databasschema som rastertabellen tillhör.
- `r_table_namn` rastertabell
- `r_raster_column` den kolumn i tabellen `r_table_name` som är av typen raster. Det finns inget i PostGIS som hindrar dig från att ha flera rasterkolumner per tabell, så det är möjligt att ha en rastertabell listad flera gånger med en annan rasterkolumn för varje.
- `srid` Den spatiala referensidentifieraren för rastret. Bör vara en post i Avsnitt 4.5.
- `scale_x` Skalningen mellan geometriska rumskoordinater och pixel. Detta är endast tillgängligt om alla tiles i rasterkolumnen har samma `scale_x` och denna begränsning tillämpas. Se ST_ScaleX för mer information.
- `scale_y` Skalningen mellan geometriska rumskoordinater och pixel. Detta är endast tillgängligt om alla tiles i rasterkolumnen har samma `scale_y` och `scale_y-begränsningen` tillämpas. Se ST_ScaleY för mer information.
- `blocksize_x` Bredden (antal pixlar tvärs över) för varje rasterplatta. Se ST_Width för mer information.
- `blocksize_y` Bredden (antal pixlar nedåt) för varje rasterplatta. Se ST_Height för mer information.
- `same_alignment` En boolean som är sann om alla rasterplattor har samma inriktning . Se ST_SameAlignme för mer information.
- `regular_blocking` Om rasterkolumnen har begränsningarna spatially unique och coverage tile är värdet TRUE. Annars kommer det att vara FALSE.
- `num_bands` Antalet band i varje platta i rasteruppsättningen. Detta är samma information som den som tillhandahålls av ST_NumBands
- `pixel_types` En array som definierar pixeltypen för varje band. Du kommer att ha samma antal element i denna array som du har antal band. Pixel_types är en av följande definierade i ST_BandPixelType.
- `nodata_values` En matris med tal med dubbel precision som anger `nodata_värdet` för varje band. Du kommer att ha samma antal element i den här matrisen som du har antal band. Dessa siffror definierar pixelvärdet för varje band som bör ignoreras för de flesta operationer. Detta är liknande information som tillhandahålls av ST_BandNoDataValue.
- `out_db` En array med booleska flaggor som anger om rasterbanddata sparas utanför databasen. Du kommer att ha samma antal element i denna array som du har antal band.
- `extent` Detta är omfattningen av alla rasterrader i din rasteruppsättning. Om du planerar att läsa in mer data som kommer att ändra uppsättningens utsträckning, bör du köra funktionen DropRaster-Constraints före laddningen och sedan tillämpa begränsningar på nytt med AddRasterConstraints efter laddningen.
- `spatial_index` En boolean som är sann om rasterkolumnen har ett spatialt index.

10.3.2 Raster-översikter

`raster_overviews` katalogiserar information om rastertabellkolumner som används för översikter och ytterligare information om dem som är bra att känna till när man använder översikter. Översiktstabeller katalogiseras i både `raster_columns` och `raster_overviews` eftersom de är rastertabeller i sig själva men också har ett särskilt syfte, nämligen att vara en karikatyr med lägre upplösning av en tabell med högre upplösning. Dessa genereras tillsammans med huvudrastertabellen när du använder `-l-omkopplaren` i rasterladdningen eller kan genereras manuellt med AddOverviewConstraints.

Översiktstabeller innehåller samma begränsningar som andra rastertabeller samt ytterligare begränsningar som endast gäller information och som är specifika för översikter.

Notera
Informationen i `raster_overviews` duplicerar inte informationen i `raster_columns`. Om du behöver information om en översiktstabell som finns i `raster_columns` kan du koppla ihop `raster_overviews` och `raster_columns` för att få den fullständiga information du behöver.

Två huvudskäl till översikter är:

1. Lågupplöst representation av kärntabellerna som ofta används för snabb utzoomning av kartor.
2. Det går i allmänhet snabbare att göra beräkningar på dem än på deras högupplösta förlagor eftersom det finns färre poster och varje pixel täcker ett större område. Även om beräkningarna inte är lika exakta som de högupplösta tabeller de stöder, kan de vara tillräckliga i många tumregelberäkningar.

Katalogen `raster_overviews` innehåller följande kolumner med information.

- `o_table_catalog` Den databas som översiktstabellen finns i. Detta kommer alltid att läsa den aktuella databasen.
- `o_table_schema` Det databasschema som översiktsrastertabellen tillhör.
- `o_table_name` namn på tabell för rasteröversikt
- `o_raster_column` rasterkolumnen i översiktstabellen.
- `r_table_catalog` Databasen där rastertabellen för denna översiktstjänst finns. Detta kommer alltid att läsa den aktuella databasen.
- `r_table_schema` Databasschemat för den rastertabell som denna översiktstjänst tillhör.
- `r_table_name` rastertabell som denna översikt hanterar.
- `r_raster_column` den rasterkolumn som denna översiktskolumn hanterar.
- `overview_factor` - detta är pyramidnivån för översiktstabellen. Ju högre siffra desto lägre upplösning på tabellen. raster2pgsql kommer, om den får en mapp med bilder, att beräkna översikten för varje bildfil och läsa in den separat. Nivå 1 antas alltid vara originalfilen. Nivå 2 är att varje platta representerar 4 av originalet. Så om du till exempel har en mapp med bildfiler på 5000x5000 pixlar som du valt att dela upp i 125x125, kommer din bastabell för varje bildfil att ha (5000*5000)/(125*125) poster = 1600, din (l=2) `o_2-tabell` kommer att ha tak(1600/Power(2,2)) = 400 rader, din (l=3) `o_3` kommer att ha tak(1600/Power(2,3)) = 200 rader. Om dina pixlar inte är delbara med storleken på dina brickor kommer du att få några skrotbrickor (brickor som inte är helt fyllda). Observera att varje översiktsplatta som genereras av raster2pgsql har samma antal pixlar som sin förälder, men har en lägre upplösning där varje pixel i den representerar (Power(2,overview_factor) pixlar av originalet).

10.4 Bygga anpassade applikationer med PostGIS Raster

Det faktum att PostGIS raster ger dig SQL-funktioner för att rendera raster i kända bildformat ger dig många alternativ för att rendera dem. Du kan till exempel använda OpenOffice / LibreOffice för rendering, vilket visas i Rendering av PostGIS Raster-grafik med LibreOffice Base Reports. Dessutom kan du använda en mängd olika språk som visas i det här avsnittet.

10.4.1 PHP-exempel Utdata med ST_AsPNG i kombination med andra rasterfunktioner

I det här avsnittet kommer vi att visa hur man använder PHP PostgreSQL-drivrutinen och ST_AsGDALRaster -familjen av funktioner för att mata ut band 1,2,3 av en raster till en PHP-förfrågningsström som sedan kan bäddas in i en img src html-tagg.

Exempelfrågan visar hur man kombinerar ett helt gäng rasterfunktioner tillsammans för att ta tag i alla plattor som skär en viss wgs 84 avgränsningsbox och sedan förenar med ST_Union de korsande plattorna tillsammans och returnerar alla band, transformerar till användarens angivna projektion med ST_Transform och sedan matar ut resultaten som en png med ST_AsPNG.

You would call the script with `http://mywebserver/test_raster.php?srid=2249` to get the raster image in Massachusetts state plane feet.

Code

```
<?php
$conn_str = 'dbname=mydb host=localhost port=5432 user=myuser password=mypwd';
$dbconn = pg_connect($conn_str);
header('Content-Type: image/png');

if (!empty($_REQUEST['srid']) &amp;amp;&amp;amp; is_numeric($_REQUEST['srid'])) {
    $input_srid = intval($_REQUEST['srid']);
} else {
    $input_srid = 26986;
}

$sql = "SET bytea_output = 'escape';
SELECT ST_AsPNG(
    ST_Transform(
        ST_AddBand(
            ST_Union(rast, 1),
            ARRAY[ST_Union(rast, 2), ST_Union(rast, 3)]
        ),
        $input_srid
    )
) AS new_rast
FROM aerials.boston
WHERE ST_Intersects(
    rast,
    ST_Transform(
        ST_MakeEnvelope(-71.1217, 42.227, -71.1210, 42.218, 4326),
        26986
    )
)";

$result = pg_query($sql);
$row = pg_fetch_row($result);
pg_free_result($result);
if ($row === false) return;
echo pg_unescape_bytea($row[0]);
?>
```

10.4.2 ASP.NET C# Exempel Utdata med hjälp av ST_AsPNG tillsammans med andra rasterfunktioner

In this section, we'll demonstrate how to use Npgsql PostgreSQL .NET driver and the ST_AsGDALRaster family of functions to output band 1,2,3 of a raster to the HTTP response stream so it can be embedded in an `img` tag.

Du behöver npgsql .NET PostgreSQL-drivrutinen för den här övningen som du kan få det senaste från http://npgsql.projects.postgresql.org/. Ladda bara ner det senaste och släpp in i din ASP.NET bin-mapp så är du redo att gå.

Exempelfrågan visar hur man kombinerar ett helt gäng rasterfunktioner tillsammans för att ta tag i alla plattor som skär en viss wgs 84 avgränsningsbox och sedan förenar med ST_Union de korsande plattorna tillsammans och returnerar alla band, transformerar till användarens angivna projektion med ST_Transform och sedan matar ut resultaten som en png med ST_AsPNG.

Detta är samma exempel som Avsnitt 10.4.1 men implementerat i C#.

You would call the handler with `http://mywebserver/TestRaster.ashx?srid=2249` to get the raster image in Massachusetts state plane feet.

web.config connection string section:

Code

```
<connectionStrings>
    <add name="DSN"
        connectionString="server=localhost;database=mydb;Port=5432;User Id=myuser;password= ↩
    mypwd"/>
</connectionStrings>
```

Code for `TestRaster.ashx`:

Code

```
<%@ WebHandler Language="C#" Class="TestRaster" %>
using System;
using System.Data;
using System.Web;
using Npgsql;

public class TestRaster : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "image/png";
        context.Response.BinaryWrite(GetResults(context));
    }

    public bool IsReusable {
        get { return false; }
    }

    public byte[] GetResults(HttpContext context)
    {
        byte[] result = null;
        NpgsqlCommand command;
        string sql = null;
        int input_srid = 26986;
        try {
            string connectionString = System.Configuration.ConfigurationManager
                .ConnectionStrings["DSN"].ConnectionString;
            using (NpgsqlConnection conn = new NpgsqlConnection(connectionString)) {
                conn.Open();

                if (context.Request["srid"] != null)
                {
                    input_srid = Convert.ToInt32(context.Request["srid"]);
                }
                sql = @"SELECT ST_AsPNG(
                            ST_Transform(
```

```
                                ST_AddBand(
                                    ST_Union(rast, 1),
                                    ARRAY[ST_Union(rast, 2), ST_Union(rast, 3)]
                                ),
                                :input_srid
                            )
                        ) As new_rast
                        FROM aerials.boston
                        WHERE ST_Intersects(
                            rast,
                            ST_Transform(
                                ST_MakeEnvelope(
                                    -71.1217, 42.227,
                                    -71.1210, 42.218,
                                    4326
                                ),
                                26986
                            )
                        )";
                command = new NpgsqlCommand(sql, conn);
                command.Parameters.Add(new NpgsqlParameter("input_srid", input_srid));


                result = (byte[]) command.ExecuteScalar();
                conn.Close();
            }

        }
        catch (Exception ex)
        {
            result = null;
            context.Response.Write(ex.Message.Trim());
        }
        return result;
    }
}
```

10.4.3 Java-konsolapp som matar ut rasterfråga som bildfil

Detta är en enkel java-konsolapp som tar en fråga som returnerar en bild och matar ut till den angivna filen.

Du kan ladda ner de senaste PostgreSQL JDBC-drivrutinerna från http://jdbc.postgresql.org/download.html

You can compile the following code with commands like:

Code

```
CLASSPATH=".:../postgresql.jar" javac SaveQueryImage.java
jar cfm SaveQueryImage.jar Manifest.txt *.class
```

And call it from the command line with something like:

Code

```
java -jar SaveQueryImage.jar \
  "SELECT ST_AsPNG(ST_AsRaster(ST_Buffer(ST_Point(1, 5), 10, 'quad_segs=2'), 150, 150, '8 ↩
    BUI', 100));" \
  "test.png"
```

`Manifest.txt`:

Code

```
Class-Path: postgresql-9.0-801.jdbc4.jar
Main-Class: SaveQueryImage
```

Code for `SaveQueryImage.java`:

Code

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.io.*;

public class SaveQueryImage {
  public static void main(String[] argv) {
      System.out.println("Checking if Driver is registered with DriverManager.");

      try {
        //java.sql.DriverManager.registerDriver (new org.postgresql.Driver());
        Class.forName("org.postgresql.Driver");
      }
      catch (ClassNotFoundException cnfe) {
        System.out.println("Couldn't find the driver!");
        cnfe.printStackTrace();
        System.exit(1);
      }

      Connection conn = null;

      try {
        conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/mydb", "myuser ↩
   ", "mypwd");
        conn.setAutoCommit(false);

        PreparedStatement sGetImg = conn.prepareStatement(argv[0]);

        ResultSet rs = sGetImg.executeQuery();

        FileOutputStream fout;
        try
        {
            rs.next();
            /** Output to file name requested by user **/
            fout = new FileOutputStream(new File(argv[1]) );
            fout.write(rs.getBytes(1));
            fout.close();
        }
        catch(Exception e)
        {
            System.out.println("Can't create file");
            e.printStackTrace();
        }

        rs.close();
        sGetImg.close();
        conn.close();
      }
      catch (SQLException se) {
        System.out.println("Couldn't connect: print out a stack trace and exit.");
        se.printStackTrace();
        System.exit(1);
```

```
      }
  }
}
```

10.4.4 Använd PLPython för att dumpa bilder via SQL

Detta är en plpython-lagrad funktion som skapar en fil i serverkatalogen för varje post. Kräver att du har plpython installerat. Bör fungera bra med både plpythonu och plpython3u.

Code

```
CREATE OR REPLACE FUNCTION write_file (param_bytes bytea, param_filepath text)
RETURNS text
AS $$
f = open(param_filepath, 'wb+')
f.write(param_bytes)
return param_filepath
$$ LANGUAGE plpythonu;
```

Write out 5 images to the PostgreSQL server in varying sizes. The PostgreSQL daemon account needs write access to the folder; this echoes back the file names created.

Code

```
SELECT write_file(
    ST_AsPNG(
        ST_AsRaster(
            ST_Buffer(ST_Point(1, 5), j * 5, 'quad_segs=2'),
            150 * j,
            150 * j,
            '8BUI',
            100
        )
    ),
    'C:/temp/slices' || j || '.png'
)
FROM generate_series(1, 5) AS j;
```

Output

```
write_file
--------------------
 C:/temp/slices1.png
 C:/temp/slices2.png
 C:/temp/slices3.png
 C:/temp/slices4.png
 C:/temp/slices5.png
```

10.4.5 Utdata av raster med PSQL

Sadly PSQL doesn't have easy to use built-in functionality for outputting binaries. This is a bit of a hack that piggy backs on PostgreSQL somewhat legacy large object support. To use it, first launch your **psql** command line connected to your database.

Till skillnad från pythonmetoden skapas filen här på din lokala dator.

Code

```
SELECT
    oid,
    lowrite(lo_open(oid, 131072), png) AS num_bytes
FROM (VALUES (
    lo_create(0),
    ST_AsPNG((SELECT rast FROM aerials.boston WHERE rid = 1))
)) AS v(oid, png);
```

Output

```
oid   | num_bytes
---------+-----------
 2630819 |     74860
```

Note the large object OID and export it to a file on your local computer, replacing the example path with the desired file path.

Code

```
\lo_export 2630819 'C:/temp/aerial_samp.png'
```

This deletes the file from large object storage in the database.

Code

```
SELECT lo_unlink(2630819);
```

Kapitel 11

Referens för raster

`raster` är en PostGIS-typ för lagring och analys av rasterdata.

För att läsa in raster från rasterfiler, se Avsnitt 10.1

I några exempel i denna referens används en rastertabell med dummy-data som skapats med följande kod:

Code

```
CREATE TABLE dummy_rast(rid integer, rast raster);
INSERT INTO dummy_rast(rid, rast)
VALUES (1,
('01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0000' -- nBands (uint16 0)
||
'0000000000000040' -- scaleX (float64 2)
||
'0000000000000840' -- scaleY (float64 3)
||
'000000000000E03F' -- ipX (float64 0.5)
||
'000000000000E03F' -- ipY (float64 0.5)
||
'0000000000000000' -- skewX (float64 0)
||
'0000000000000000' -- skewY (float64 0)
||
'00000000' -- SRID (int32 0)
||
'0A00' -- width (uint16 10)
||
'1400' -- height (uint16 20)
)::raster
),
-- Raster: 5 x 5 pixels, 3 bands, PT_8BUI pixel type, NODATA = 0
(2, ('01000003009A9999999999A93F9A9999999999A9BF000000E02B274A' ||
'4100000000771956410000000000000000000000000000000000 ↩
    FFFFFFFF050005000400FDFEFDFEFEFDFEFEFDF9FAFEF' ||
' ↩
    EFCF9FBFDFEFEFDFCFAFEFEFE04004E627AADD16076B4F9FE6370A9F5FE59637AB0E54F5861708704004656648 7A1506
    ')::raster);
```

Funktionerna nedan är sådana som en användare av PostGIS Raster sannolikt kommer att behöva. Det finns andra stödfunktioner för raster som inte är av intresse för en allmän användare.

11.1 Rasterstöd Datatyper

11.1.1 geomval

geomval — En spatial datatyp med två fält - geom (som innehåller ett geometriobjekt) och val (som innehåller ett pixelvärde med dubbel precision från ett rasterband).

Description

geomval är en sammansatt datatyp som består av ett geometriobjekt som refereras av fältet .geom och val, ett värde med dubbel precision som representerar pixelvärdet på en viss geometrisk plats i ett rasterband. Den används av funktionerna ST_DumpAsPolygon och Raster intersection som en utdatatyp för att explodera ett rasterband till geometripolygoner.

Se även

Avsnitt 13.6

11.1.2 addbandarg

addbandarg — En sammansatt typ som används som indata till funktionen ST_AddBand och som definierar det nya bandets attribut och initiala värde.

Description

En sammansatt typ som används som indata till funktionen ST_AddBand och som definierar det nya bandets attribut och initiala värde.

Field	Type	Description
`index`	integer	1-baserat värde som anger positionen där det nya bandet ska läggas till bland rastrets band. Om NULL, kommer det nya bandet att läggas till i slutet av rastrets band.
`pixeltype`	text	Pixel type of the new band. One of the pixel types described in ST_BandPixelType.
`initialvalue`	double precision	Initial value that all pixels of the new band will be set to.
`nodataval`	double precision	NODATA-värde för det nya bandet. Om NULL, kommer det nya bandet inte att tilldelas något NODATA-värde.

Se även

ST_AddBand

11.1.3 rastbandarg

rastbandarg — En sammansatt typ som kan användas när man behöver uttrycka ett raster och ett bandindex för detta raster.

Description

En sammansatt typ som kan användas när man behöver uttrycka ett raster och ett bandindex för detta raster.

Field	Type	Description
rast	raster	The raster in question.
nband	integer	1-based value indicating the raster band.

Se även

ST_MapAlgebra (callback function version)

11.1.4 raster

raster — spatial rasterdatatyp.

Description

raster är en spatial datatyp som används för att representera rasterdata som de som importeras från JPEGs, TIFFs, PNGs, digitala höjdmodeller. Varje raster har 1 eller flera band som vart och ett har en uppsättning pixelvärden. Raster kan georefereras.

Notera
Kräver att PostGIS är kompilerat med GDAL-stöd. För närvarande kan raster implicit konverteras till geometrityp, men konverteringen returnerar ST_ConvexHull för rastret. Denna automatiska casting kan tas bort inom en snar framtid, så lita inte på det.

Casting-beteende

I detta avsnitt listas de automatiska såväl som explicita kast som är tillåtna för denna datatyp

Kasta till	Beteende
geometri	automatisk

Se även

Kapitel 11

11.1.5 reclassarg

reclassarg — En sammansatt typ som används som indata till funktionen ST_Reclass och som definierar beteendet vid omklassificering.

Description

En sammansatt typ som används som indata till funktionen ST_Reclass som definierar beteendet vid omklassificering.

Field	Type	Description
`nband`	integer	Bandnumret på bandet för att omklassificera.
`reclassexpr`	text	Comma-delimited `range:map_range` mappings that define how old band values map to new band values.
`pixeltype`	text	One of defined pixel types as described in ST_BandPixelType.
`nodataval`	double precision	Värde som ska behandlas som ingen data. För bildutdata som stöder transparens kommer dessa att vara tomma.

The `reclassexpr` range delimiters mean: `(` greater than, `)` less than, `]` less than or equal, and `[` greater than or equal.

- `[a-b]` means `a <= x <= b`.
- `(a-b]` means `a < x <= b`.
- `[a-b)` means `a <= x < b`.
- `(a-b)` means `a < x < b`.

Opening-parenthesis notation is optional, so `a-b` means the same as `(a-b)`.

Exempel

Reclassify band 2 as an 8BUI where 255 is nodata value.

Code

```
SELECT ROW(2, '0-100:1-10, 101-500:11-150,501 - 10000: 151-254', '8BUI', 255)::reclassarg;
```

Reclassify band 1 as an 1BB and no nodata value defined.

Code

```
SELECT ROW(1, '0-100]:0, (100-255:1', '1BB', NULL)::reclassarg;
```

Se även

ST_Reclass

11.1.6 summarystats

summarystats — En sammansatt typ som returneras av funktionerna ST_SummaryStats och ST_SummarySta

Description

En sammansatt typ som returneras av funktionerna ST_SummaryStats och ST_SummaryStatsAgg.

antal heltal Antal pixlar som räknats för den sammanfattande statistiken.

summa dubbel precision Summan av alla räknade pixelvärden.

medelvärde dubbel precision Aritmetiskt medelvärde av alla räknade pixelvärden.

stddev dubbel precision Standardavvikelse för alla räknade pixelvärden.

min dubbel precision Minimivärde för räknade pixelvärden.

max dubbel precision Maximalt värde för räknade pixelvärden.

Se även

ST_SummaryStats, ST_SummaryStatsAgg

11.1.7 unionarg

unionarg — En sammansatt typ som används som indata till ST_Union-funktionen och som definierar de band som ska bearbetas och beteendet hos UNION-operationen.

Description

En sammansatt typ som används som indata till ST_Union-funktionen och som definierar de band som ska bearbetas och beteendet hos UNION-operationen.

Field	Type	Description
`nband`	integer	1-baserat värde som anger bandet för varje inmatningsraster som ska bearbetas.
`uniontype`	text	Type of union operation. One of the types described in ST_Union.

Se även

ST_Union

11.2 Rasterhantering

11.2.1 AddRasterConstraints

AddRasterConstraints — Lägger till rasterbegränsningar till en laddad rastertabell för en specifik kolumn som begränsar spatial ref, skalning, blockstorlek, inriktning, band, bandtyp och en flagga för att ange om rasterkolumnen blockeras regelbundet. Tabellen måste vara laddad med data för att begränsningarna ska kunna härledas. Returnerar true om begränsningsinställningen har utförts och utfärdar ett meddelande i annat fall.

Synopsis

boolean **AddRasterConstraints**(name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=true, boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true, boolean same_alignment=true, boolean regular_blocking=false, boolean num_bands=true , boolean pixel_types=true , boolean nodata_values=true , boolean out_db=true , boolean extent=true);
boolean **AddRasterConstraints**(name rasttable, name rastcolumn, text[] VARIADIC constraints);
boolean **AddRasterConstraints**(name rastschema, name rasttable, name rastcolumn, text[] VARIADIC constraints);
boolean **AddRasterConstraints**(name rastschema, name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=true, boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true, boolean same_alignment=true, boolean regular_blocking=false, boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true , boolean out_db=true , boolean extent=true);

Description

Skapar begränsningar för en rasterkolumn som används för att visa information i rasterkatalogen `raster_columns`. `Rastschema` är namnet på det tabellschema som tabellen finns i. `Srid` måste vara en heltalsreferens till en post i tabellen SPATIAL_REF_SYS.

`raster2pgsql-laddaren` använder denna funktion för att registrera rastertabeller

Giltiga namn på begränsningar som kan anges: se Avsnitt 10.3.1 för mer information.

- `blocksize` ställer in både X- och Y-blocksize
- `blocksize_x` ställer in X-tile (bredd i pixlar för varje tile)
- `blocksize_y` anger Y-platta (höjd i pixlar för varje platta)
- `extent` beräknar hela tabellens utsträckning och tillämpar begränsningen att alla raster måste ligga inom denna utsträckning
- `num_bands` antal band
- `pixel_types` läser matris med pixeltyper för varje band säkerställer att alla band n har samma pixeltyp
- `regular_blocking` anger spatialt unika (inga två raster kan vara spatialt lika) och täckningsplatta (raster är anpassat till en täckning) begränsningar
- `same_alignment` säkerställer att de alla har samma inriktning, vilket innebär att alla två brickor du jämför kommer att returnera sant för. Hänvisa till ST_SameAlignment.
- `srid` säkerställer att alla har samma srid
- Mer -- alla som anges som indata i ovanstående funktioner

Notera
Denna funktion härleder begränsningarna från de data som redan finns i tabellen. För att den ska fungera måste du därför först skapa rasterkolumnen och sedan läsa in den med data.

Notera
Om du behöver läsa in mer data i dina tabeller efter att du redan har tillämpat begränsningar kan du köra DropRasterConstraints om omfattningen av dina data har ändrats.

Tillgänglighet: 2.0.0

Exempel

Apply all possible constraints on column based on data.

Verify the raster table is registered correctly in the raster_columns view.

Code

```
CREATE TABLE myrasters(rid SERIAL primary key, rast raster);
INSERT INTO myrasters(rast)
SELECT ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0.3, -0.3, 2, 2, 0, 0, 4326), 1, '8BSI':: ↩
    text, -129, NULL);

SELECT AddRasterConstraints('myrasters'::name, 'rast'::name);
SELECT srid, scale_x, scale_y, blocksize_x, blocksize_y, num_bands, pixel_types,  ↩
    nodata_values
    FROM raster_columns
    WHERE r_table_name = 'myrasters';
```

Output

```
srid | scale_x | scale_y | blocksize_x | blocksize_y | num_bands | pixel_types| nodata_values
------+---------+---------+-------------+-------------+-----------+-------------+---------------
 4326 |       2 |       2 |        1000 |        1000 |         1 | {8BSI}      | {0}
```

Apply single constraint.

Code

```
CREATE TABLE public.myrasters2(rid SERIAL primary key, rast raster);
INSERT INTO myrasters2(rast)
SELECT ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0.3, -0.3, 2, 2, 0, 0, 4326), 1, '8BSI':: ↩
    text, -129, NULL);

SELECT AddRasterConstraints('public'::name, 'myrasters2'::name, 'rast'::name, ' ↩
    regular_blocking', 'blocksize');
```

Output

```
NOTICE:  Adding regular blocking constraint
NOTICE:  Adding blocksize-X constraint
NOTICE:  Adding blocksize-Y constraint
```

Se även

Avsnitt 10.3.1, ST_AddBand, ST_MakeEmptyRaster, DropRasterConstraints, ST_BandPixelType, ST_SRID

11.2.2 DropRasterConstraints

DropRasterConstraints — Tar bort PostGIS-rasterbegränsningar som hänvisar till en rastertabellkolumn. Användbart om du behöver ladda om data eller uppdatera dina rasterkolumndata.

Synopsis

boolean **DropRasterConstraints**(name rasttable, name rastcolumn, boolean srid, boolean scale_x, boolean scale_y, boolean blocksize_x, boolean blocksize_y, boolean same_alignment, boolean regular_blocking, boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true, boolean out_db=true , boolean extent=true);
boolean **DropRasterConstraints**(name rastschema, name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=true, boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true, boolean same_alignment=true, boolean regular_blocking=false, boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true, boolean out_db=true , boolean extent=true);
boolean **DropRasterConstraints**(name rastschema, name rasttable, name rastcolumn, text[] constraints);

Description

Tar bort PostGIS-rasterbegränsningar som hänvisar till en rastertabellkolumn som lades till av AddRasterConstraints. Användbart om du behöver läsa in mer data eller uppdatera dina rasterkolumndata. Du behöver inte göra detta om du vill bli av med en rastertabell eller en rasterkolumn.

To drop a raster table use standard SQL: `DROP TABLE mytable`.

To drop just a raster column and leave the rest of the table, use standard SQL: `ALTER TABLE mytable DROP COLUMN rast`.

kommer tabellen att försvinna från katalogen `raster_columns` om kolumnen eller tabellen tas bort. Men om endast begränsningarna tas bort kommer rasterkolumnen fortfarande att listas i katalogen `raster_columns`, men det kommer inte att finnas någon annan information om den förutom kolumnnamnet och tabellen.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT DropRasterConstraints ('myrasters', 'rast');
```

Output

```
droprasterconstraints
----------------------
 t
(1 row)
```

Verify change in raster_columns.

Code

```
SELECT srid, scale_x, scale_y, blocksize_x, blocksize_y, num_bands, pixel_types,  ↩
    nodata_values
    FROM raster_columns
    WHERE r_table_name = 'myrasters';
```

Output

```
srid | scale_x | scale_y | blocksize_x | blocksize_y | num_bands | pixel_types| nodata_values
------+---------+---------+-------------+-------------+-----------+-------------+---------------
    0 |         |         |             |             |           |             |
```

Se även

AddRasterConstraints

11.2.3 AddOverviewConstraints

AddOverviewConstraints — Märk en rasterkolumn som en översikt av en annan.

Synopsis

boolean **AddOverviewConstraints**(name ovschema, name ovtable, name ovcolumn, name refschema, name reftable, name refcolumn, int ovfactor);
boolean **AddOverviewConstraints**(name ovtable, name ovcolumn, name reftable, name refcolumn, int ovfactor);

Description

Lägger till begränsningar för en rasterkolumn som används för att visa information i rasterkatalogen `raster_overviews`.

Parametern `ovfactor` representerar skalmultiplikatorn i översiktskolumnen: högre översiktsfaktorer ger lägre upplösning.

När parametrarna `ovschema` och `refschema` utelämnas används den första tabellen som hittas när `sökvägen` skannas.

Tillgänglighet: 2.0.0

Exempel

Verify the raster table is registered correctly in the raster_overviews view.

Code

```
CREATE TABLE res1 AS SELECT
ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 2),
  1, '8BSI'::text, -129, NULL
) r1;

CREATE TABLE res2 AS SELECT
ST_AddBand(ST_MakeEmptyRaster(500, 500, 0, 0, 4),
  1, '8BSI'::text, -129, NULL
) r2;

SELECT AddOverviewConstraints('res2', 'r2', 'res1', 'r1', 2);
SELECT o_table_name ot, o_raster_column oc,
       r_table_name rt, r_raster_column rc,
       overview_factor f
FROM raster_overviews WHERE o_table_name = 'res2';
```

Output

```
ot  | oc |  rt  | rc | f
------+----+------+----+---
 res2 | r2 | res1 | r1 | 2
(1 row)
```

Se även

Avsnitt 10.3.2, DropOverviewConstraints, ST_CreateOverview, AddRasterConstraints

11.2.4 DropOverviewConstraints

DropOverviewConstraints — Avmarkera en rasterkolumn från att vara en översikt av en annan.

Synopsis

boolean **DropOverviewConstraints**(name ovschema, name ovtable, name ovcolumn);
boolean **DropOverviewConstraints**(name ovtable, name ovcolumn);

Description

Ta bort de begränsningar som används för att visa en rasterkolumn som en översikt över en annan i rasterkatalogen `raster_overviews` från en rasterkolumn.

Om parametern `ovschema` utelämnas används den första tabellen som hittas när `sökvägen` skannas.

Tillgänglighet: 2.0.0

Se även

Avsnitt 10.3.2, AddOverviewConstraints, DropRasterConstraints

11.2.5 PostGIS_GDAL_Version

PostGIS_GDAL_Version — Rapporterar den version av GDAL-biblioteket som används av PostGIS.

Synopsis

text **PostGIS_GDAL_Version**();

Description

Rapporterar den version av GDAL-biblioteket som används av PostGIS. Kommer också att kontrollera och rapportera om GDAL kan hitta sina datafiler.

Exempel

Code

```
SELECT PostGIS_GDAL_Version();
```

Output

```
GDAL 1.11dev, released 2013/04/13
```

Se även

postgis.gdal_datapath

11.2.6 PostGIS_Raster_Lib_Build_Date

PostGIS_Raster_Lib_Build_Date — Rapporterar hela rasterbibliotekets byggnadsdatum.

Synopsis

text **PostGIS_Raster_Lib_Build_Date**();

Description

Rapporterar rastrets byggnadsdatum

Exempel

Code

```
SELECT PostGIS_Raster_Lib_Build_Date();
```

Output

```
2010-04-28 21:15:10
```

Se även

PostGIS_Raster_Lib_Version

11.2.7 PostGIS_Raster_Lib_Version

PostGIS_Raster_Lib_Version — Rapporterar fullständig information om rasterversion och byggkonfiguration.

Synopsis

text **PostGIS_Raster_Lib_Version**();

Description

Rapporterar fullständig information om rasterversion och byggkonfiguration.

Exempel

Code

```
SELECT PostGIS_Raster_Lib_Version();
```

Output

```
2.0.0
```

Se även

PostGIS_Lib_Version

11.2.8 ST_GDALDrivers

ST_GDALDrivers — Returnerar en lista över rasterformat som stöds av PostGIS via GDAL. Endast de format med can_write=True kan användas av ST_AsGDALRaster

Synopsis

setof record **ST_GDALDrivers**(integer OUT idx, text OUT short_name, text OUT long_name, text OUT can_read, text OUT can_write, text OUT create_options);

Description

Returnerar en lista med rasterformat short_name,long_name och skapande alternativ för varje format som stöds av GDAL. Använd short_name som indata i `format`-parametern för ST_AsGDALRaster. Alternativen varierar beroende på vilka drivrutiner som libgdal kompilerades med. `create_options` returnerar en xml-formaterad uppsättning CreationOptionList/Option som består av namn och valfri `type`, `description` och uppsättning `VALUE` för varje skaparalternativ för den specifika drivrutinen.

Ändrad: 2.5.0 - lägg till kolumnerna can_read och can_write.

Ändrad: 2.0.6, 2.1.3 - som standard är inga drivrutiner aktiverade, såvida inte GUC eller miljövariabeln gdal_enabled_drivers är inställd.

Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0.

Exempel

List of Drivers.

Code

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SELECT short_name, long_name, can_write
FROM ST_GDALDrivers()
ORDER BY short_name;
```

Output

```
short_name    |                          long_name                          | can_write
----------------+-------------------------------------------------------------+-----------
 AAIGrid        | Arc/Info ASCII Grid                                         | t
 ACE2           | ACE2                                                        | f
 ADRG           | ARC Digitized Raster Graphics                               | f
 AIG            | Arc/Info Binary Grid                                        | f
 AirSAR         | AirSAR Polarimetric Image                                   | f
 ARG            | Azavea Raster Grid format                                   | t
 BAG            | Bathymetry Attributed Grid                                  | f
 BIGGIF         | Graphics Interchange Format (.gif)                          | f
 BLX            | Magellan topo (.blx)                                        | t
 BMP            | MS Windows Device Independent Bitmap                        | f
 BSB            | Maptech BSB Nautical Charts                                 | f
 PAux           | PCI .aux Labelled                                           | f
 PCIDSK         | PCIDSK Database File                                        | f
 PCRaster       | PCRaster Raster File                                        | f
 PDF            | Geospatial PDF                                              | f
 PDS            | NASA Planetary Data System                                  | f
 PDS4           | NASA Planetary Data System 4                                | t
 PLMOSAIC       | Planet Labs Mosaics API                                     | f
 PLSCENES       | Planet Labs Scenes API                                      | f
 PNG            | Portable Network Graphics                                   | t
 PNM            | Portable Pixmap Format (netpbm)                             | f
 PRF            | Racurs PHOTOMOD PRF                                         | f
 R              | R Object Data Store                                         | t
 Rasterlite     | Rasterlite                                                  | t
 RDA            | DigitalGlobe Raster Data Access driver                      | f
 RIK            | Swedish Grid RIK (.rik)                                     | f
 RMF            | Raster Matrix Format                                        | f
 ROI_PAC        | ROI_PAC raster                                              | f
 RPFTOC         | Raster Product Format TOC format                            | f
 RRASTER        | R Raster                                                    | f
 RS2            | RadarSat 2 XML Product                                      | f
 RST            | Idrisi Raster A.1                                           | t
 SAFE           | Sentinel-1 SAR SAFE Product                                 | f
 SAGA           | SAGA GIS Binary Grid (.sdat, .sg-grd-z)                     | t
 SAR_CEOS       | CEOS SAR Image                                              | f
 SDTS           | SDTS Raster                                                 | f
 SENTINEL2      | Sentinel 2                                                  | f
 SGI            | SGI Image File Format 1.0                                   | f
 SNODAS         | Snow Data Assimilation System                               | f
 SRP            | Standard Raster Product (ASRP/USRP)                         | f
 SRTMHGT        | SRTMHGT File Format                                         | t
 Terragen       | Terragen heightfield                                        | f
 TIL            | EarthWatch .TIL                                             | f
 TSX            | TerraSAR-X Product                                          | f
 USGSDEM        | USGS Optional ASCII DEM (and CDED)                          | t
 VICAR          | MIPL VICAR file                                             | f
 VRT            | Virtual Raster                                              | t
 WCS            | OGC Web Coverage Service                                    | f
 WMS            | OGC Web Map Service                                         | t
 WMTS           | OGC Web Map Tile Service                                    | t
 XPM            | X11 PixMap Format                                           | t
 XYZ            | ASCII Gridded XYZ                                           | t
 ZMap           | ZMap Plus Grid                                              | t
```

List of options for each driver.

Output the create options XML column of JPEG as a table. These creator options can be used in the ST_AsGDALRaster options argument.

Code

```
SELECT (xpath('@name', g.opt))[1]::text As oname,
       (xpath('@type', g.opt))[1]::text As otype,
       (xpath('@description', g.opt))[1]::text As descrip
FROM (SELECT unnest(xpath('/CreationOptionList/Option', create_options::xml)) As opt
FROM  ST_GDALDrivers()
WHERE short_name = 'JPEG') As g;
```

Output

```
oname        |  otype  |      descrip
-------------------+---------+-------------------------------------------------
 PROGRESSIVE       | boolean | whether to generate a progressive JPEG
 QUALITY           | int     | good=100, bad=0, default=75
 WORLDFILE         | boolean | whether to generate a worldfile
```

```
 INTERNAL_MASK      | boolean | whether to generate a validity mask
 COMMENT            | string  | Comment
 SOURCE_ICC_PROFILE | string  | ICC profile encoded in Base64
 EXIF_THUMBNAIL     | boolean | whether to generate an EXIF thumbnail(overview).
                                By default its max dimension will be 128
 THUMBNAIL_WIDTH    | int     | Forced thumbnail width
 THUMBNAIL_HEIGHT   | int     | Forced thumbnail height
(9 rows)
```

This example turns a few common GeoTIFF creator options into a compact table.

Code

```
WITH gtiff_options AS (
    SELECT
        option_name,
        option_type,
        default_value
    FROM ST_GDALDrivers() AS driver
    CROSS JOIN LATERAL XMLTABLE(
        '/CreationOptionList/Option'
        PASSING (driver.create_options::xml)
        COLUMNS
            option_name text PATH '@name',
            option_type text PATH '@type',
            default_value text PATH '@default'
    ) AS options
    WHERE driver.short_name = 'GTiff'
)
SELECT option_name AS name, option_type AS type, coalesce(default_value, '') AS ↩
    default_value
FROM gtiff_options
WHERE option_name = ANY (ARRAY[
    'COMPRESS',
    'INTERLEAVE',
    'TILED',
    'BLOCKXSIZE',
    'BLOCKYSIZE',
    'JPEG_QUALITY'
])
ORDER BY 1;
```

Output

```
name      |     type      | default_value
--------------+---------------+---------------
 BLOCKXSIZE    | int           |
 BLOCKYSIZE    | int           |
 COMPRESS      | string-select |
 INTERLEAVE    | string-select | PIXEL
 JPEG_QUALITY  | int           | 75
 TILED         | boolean       |
(6 rows)
```

Use XMLTABLE to expose selected GTiff creation options as rows. The same pattern can be used without the name filter to inspect the full XML document.

Code

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';

SELECT option_name, option_type
FROM ST_GDALDrivers() AS driver
CROSS JOIN LATERAL XMLTABLE(
    '/CreationOptionList/Option'
    PASSING (driver.create_options::xml)
```

```
    COLUMNS
        option_name text PATH '@name',
        option_type text PATH '@type'
) AS options
WHERE driver.short_name = 'GTiff'
  AND option_name IN ('COMPRESS', 'TILED', 'BIGTIFF')
ORDER BY option_name;
```

Output

```
option_name |  option_type
------------+--------------
 BIGTIFF    | string-select
 COMPRESS   | string-select
 TILED      | boolean
(3 rows)
```

Se även

ST_AsGDALRaster, ST_SRID, postgis.gdal_enabled_drivers

11.2.9 UpdateRasterSRID

UpdateRasterSRID — Ändra SRID för alla raster i den användarspecifika kolumnen och tabellen.

Synopsis

raster **UpdateRasterSRID**(name schema_name, name table_name, name column_name, integer new_srid);
raster **UpdateRasterSRID**(name table_name, name column_name, integer new_srid);

Description

Ändra SRID för alla raster i den användarspecificerade kolumnen och tabellen. Funktionen släpper alla lämpliga kolumnbegränsningar (utsträckning, inriktning och SRID) innan SRID ändras för den angivna kolumnens raster.

Notera
Data (bandpixelvärden) i rastrerna berörs inte av denna funktion. Endast rastrets metadata ändras.

Tillgänglighet: 2.1.0

Se även

UpdateGeometrySRID

11.2.10 ST_CreateOverview

ST_CreateOverview — Skapa en version med reducerad upplösning av en given rastertäckning.

Synopsis

regclass **ST_CreateOverview**(regclass tab, name col, int factor, text algo='NearestNeighbor');

Description

Skapa en översiktstabell med omsamplade plattor från källtabellen. Utdatabrickorna kommer att ha samma storlek som indatabrickorna och täcka samma spatiala utsträckning med en lägre upplösning (pixelstorleken kommer att vara 1/`faktor` av originalet i båda riktningarna).

Översiktstabellen kommer att göras tillgänglig i katalogen `raster_overviews` och kommer att ha rasterbegränsningar.

Algoritmalternativen är: NearestNeighbor", Bilinear", "Cubic", "CubicSplineöch Lanczos". Hänvisa till: GDAL Warp resampling methods för mer information.

Tillgänglighet: 2.2.0

Exempel

Generellt bättre kvalitet på utdata, men långsammare till produktformat

Code

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2, 'Lanczos');
```

Utdata till snabbare för att bearbeta standard närmaste granne

Code

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2);
```

Se även

ST_Retile, AddOverviewConstraints, AddRasterConstraints, Avsnitt 10.3.2

11.3 Raster Constructors

11.3.1 ST_AddBand

ST_AddBand — Returnerar ett raster med det eller de nya banden av given typ som lagts till med givet initialvärde på den givna indexplatsen. Om inget index anges läggs bandet till i slutet.

Synopsis

(1) raster **ST_AddBand**(raster rast, addbandarg[] addbandargset);
(2) raster **ST_AddBand**(raster rast, integer index, text pixeltype, double precision initialvalue=0, double precision nodataval=NULL);
(3) raster **ST_AddBand**(raster rast, text pixeltype, double precision initialvalue=0, double precision nodataval=NULL);
(4) raster **ST_AddBand**(raster torast, raster fromrast, integer fromband=1, integer torastindex=at_end);
(5) raster **ST_AddBand**(raster torast, raster[] fromrasts, integer fromband=1, integer torastindex=at_end);
(6) raster **ST_AddBand**(raster rast, integer index, text outdbfile, integer[] outdbindex, double precision nodataval=NULL);
(7) raster **ST_AddBand**(raster rast, text outdbfile, integer[] outdbindex, integer index=at_end, double precision nodataval=NULL);

Description

Returnerar ett raster med ett nytt band tillagt i given position (index), av given typ, med givet initialt värde och med givet nodatavärde. Om inget index anges läggs bandet till i slutet. Om inget `fromband` anges antas band 1. Pixel type är en strängrepresentation av en av de pixeltyper som anges i ST_BandPixelType. Om ett befintligt index anges ökas alla efterföljande band >= det indexet med 1. Om ett startvärde som är större än pixeltypens max anges, sätts startvärdet till det högsta värde som pixeltypen tillåter.

För den variant som tar en array av addbandarg (Variant 1) är indexvärdet för ett specifikt addbandarg relativt till rastret vid den tidpunkt då det band som beskrivs av det addbandarg läggs till i rastret. Se exemplet Multiple New Bands nedan.

För den variant som tar en array av raster (variant 5), om `torast` är NULL, ackumuleras `fromband`-bandet för varje raster i arrayen till ett nytt raster.

För de varianter som tar `utdbfile` (variant 6 och 7) måste värdet innehålla den fullständiga sökvägen till rasterfilen. Filen måste också vara tillgänglig för postgres-serverprocessen.

Förbättrad: 2.1.0-stöd för addbandarg har lagts till.

Förbättrad: 2.1.0 stöd för nya out-db band tillagt.

Exempel

Single New Band.

This example adds another 8-bit unsigned integer band with pixels initialized to 200.

Code

```
UPDATE dummy_rast
    SET rast = ST_AddBand(rast, '8BUI'::text, 200)
WHERE rid = 1;
```

Create an empty raster 100x100 units, with upper left right at 0, add 2 bands (band 1 is 0/1 boolean bit switch, band2 allows values 0-15) Uses addbandargs.

Output metadata for the raster bands to verify they are correct.

Code

```
INSERT INTO dummy_rast(rid, rast)
    VALUES(10, ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 1, -1, 0, 0, 0),
    ARRAY[
        ROW(1, '1BB'::text, 0, NULL),
        ROW(2, '4BUI'::text, 0, NULL)
            ]::addbandarg[]
     )
    );

SELECT  (bmd).*
FROM (SELECT ST_BandMetaData(rast, generate_series(1, 2)) As bmd
    FROM dummy_rast WHERE rid = 10) AS foo;
```

Output

```
pixeltype | nodatavalue | isoutdb | path
-----------+---------------+-------------+---------+------
 1BB       |             | f       |
 4BUI      |             | f       |
```

This example outputs metadata for the raster.

Code

```
SELECT  (rmd).width, (rmd).height, (rmd).numbands
FROM (SELECT ST_MetaData(rast) As rmd
    FROM dummy_rast WHERE rid = 10) AS foo;
```

Output

```
width | height | numbands
-------+--------+----------
   100 |    100 |        2
```

Multiple New Bands.

Code

```
SELECT
    *
FROM ST_BandMetaData(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1, 0, 0, 0),
        ARRAY[
            ROW(NULL, '8BUI', 255, 0),
            ROW(NULL, '16BUI', 1, 2),
            ROW(2, '32BUI', 100, 12),
            ROW(2, '32BF', 3.14, -1)
        ]::addbandarg[]
    ),
    ARRAY[]::integer[]
);
```

Output

```
bandnum | pixeltype | nodatavalue | isoutdb | path | outdbbandnum | filesize | filetimestamp
---------+-----------+-------------+---------+------+--------------+----------+---------------
       1 | 8BUI      |           0 | f       | null |         null |     null |          null
       2 | 32BF      |          -1 | f       | null |         null |     null |          null
       3 | 32BUI     |          12 | f       | null |         null |     null |          null
       4 | 16BUI     |           2 | f       | null |         null |     null |          null
```

Aggregate the 1st band of a table of like rasters into a single raster. With as many bands as there are test_types and as many rows (new rasters) as there are mice. Note that The ORDER BY test_type clause relies on aggregate ORDER BY support. The resulting raster will have a band for each test_type alphabetical by test_type. For mouse lovers: No mice were harmed in this exercise.

Code

```
SELECT
    mouse,
    ST_AddBand(NULL, array_agg(rast ORDER BY test_type), 1) As rast
FROM mice_studies
GROUP BY mouse;
```

New Out-db band.

Code

```
SELECT
    *
FROM ST_BandMetaData(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1, 0, 0, 0),
        '/home/raster/mytestraster.tif'::text, NULL::int[]
    ),
    ARRAY[]::integer[]
);
```

Output

```
bandnum | pixeltype | nodatavalue | isoutdb | path
---------+-----------+-------------+---------+------
       1 | 8BUI      |             | t       | /home/raster/mytestraster.tif
       2 | 8BUI      |             | t       | /home/raster/mytestraster.tif
       3 | 8BUI      |             | t       | /home/raster/mytestraster.tif
```

Se även

ST_BandMetaData, ST_BandPixelType, ST_MakeEmptyRaster, ST_MetaData, ST_NumBands, ST_Reclass

11.3.2 ST_AsRaster

ST_AsRaster — Konverterar en PostGIS-geometri till ett PostGIS-raster.

Synopsis

raster **ST_AsRaster**(geometry geom, raster ref, text pixeltype, double precision value=1, double precision nodataval=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, raster ref, text[] pixeltype=ARRAY['8BUI'], double precision[] value=ARRAY[1], double precision[] nodataval=ARRAY[0], boolean touched=false);
raster **ST_AsRaster**(geometry geom, double precision scalex, double precision scaley, double precision gridx, double precision gridy, text pixeltype, double precision value=1, double precision nodataval=0, double precision skewx=0, double precision skewy=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, double precision scalex, double precision scaley, double precision gridx=NULL, double precision gridy=NULL, text[] pixeltype=ARRAY['8BUI'], double precision[] value=ARRAY[1], double precision[] nodataval=ARRAY[0], double precision skewx=0, double precision skewy=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, double precision scalex, double precision scaley, text pixeltype, double precision value=1, double precision nodataval=0, double precision upperleftx=NULL, double precision upperlefty=NULL, double precision skewx=0, double precision skewy=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, double precision scalex, double precision scaley, text[] pixeltype, double precision[] value=ARRAY[1], double precision[] nodataval=ARRAY[0], double precision upperleftx=NULL, double precision upperlefty=NULL, double precision skewx=0, double precision skewy=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, integer width, integer height, double precision gridx, double precision gridy, text pixeltype, double precision value=1, double precision nodataval=0, double precision skewx=0, double precision skewy=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, integer width, integer height, double precision gridx=NULL, double precision gridy=NULL, text[] pixeltype=ARRAY['8BUI'], double precision[] value=ARRAY[1], double precision[] nodataval=ARRAY[0], double precision skewx=0, double precision skewy=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, integer width, integer height, text pixeltype, double precision value=1, double precision nodataval=0, double precision upperleftx=NULL, double precision upperlefty=NULL, double precision skewx=0, double precision skewy=0, boolean touched=false);
raster **ST_AsRaster**(geometry geom, integer width, integer height, text[] pixeltype, double precision[] value=ARRAY[1], double precision[] nodataval=ARRAY[0], double precision upperleftx=NULL, double precision upperlefty=NULL, double precision skewx=0, double precision skewy=0, boolean touched=false);

Description

Konverterar en PostGIS-geometri till ett PostGIS-raster. De många varianterna erbjuder tre grupper av möjligheter att ställa in justering och pixelstorlek för det resulterande rastret.

Den första gruppen, som består av de två första varianterna, producerar ett raster med samma inriktning (`scalex`, `scaley`, `gridx` och `gridy`), pixeltyp och nodatavärde som det tillhandahållna referensrastret. Du skickar i allmänhet detta referensraster genom att koppla samman tabellen som innehåller geometrin med tabellen som innehåller referensrastret.

Den andra gruppen, som består av fyra varianter, låter dig ställa in rastrets dimensioner genom att ange parametrarna för en pixelstorlek(`scalex` & `scaley` och `skewx` & `skewy`). `Bredden` och `höjden` på det resulterande rastret kommer att justeras för att passa geometrins omfattning. I de flesta fall måste du kasta heltal `scalex` & `scaley-argument` till dubbel precision så att PostgreSQL väljer rätt variant.

Den tredje gruppen, som består av fyra varianter, låter dig fastställa rastrets dimensioner genom att ange rastrets dimensioner (`width` & `height`). Parametrarna för pixelstorleken (`scalex` & `scaley` och `skewx` & `skewy`) i det resulterande rastret kommer att justeras för att passa geometrins utsträckning.

De två första varianterna av var och en av de två sista grupperna låter dig ange inriktningen med ett godtyckligt hörn av inriktningsrutnätet (`gridx` & `gridy`) och de två sista varianterna tar det övre vänstra hörnet (`upperleftx` & `upperlefty`).

Varje variantgrupp gör det möjligt att producera ett enbandsraster eller ett flerbandsraster. För att producera ett raster med flera band måste du tillhandahålla en array med pixeltyper(`pixeltype[]`), en array med initiala värden (`value`) och en array med`nodatavärden (nodataval)`. Om pixeltyped inte tillhandahålls är standardvärdet 8BUI, värdena 1 och nodataval 0.

Utdatarastret kommer att ha samma spatiala referens som källgeometrin. Det enda undantaget är för varianter med ett referensraster. I detta fall kommer det resulterande rastret att få samma SRID som referensrastret.

Den valfria parametern `touched` är som standard false och mappar till rasteriseringsalternativet GDAL ALL_TOUCHED, som avgör om pixlar som berörs av linjer eller polygoner ska brännas. Inte bara de som ligger på linjens renderbana eller vars mittpunkt ligger inom polygonen.

Detta är särskilt användbart för att rendera jpeg- och png-filer av geometrier direkt från databasen när de används i kombination med ST_AsPNG och andra funktioner i ST_AsGDALRaster -familjen.

Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0.

Curved geometry inputs are linearized before rasterization, including NURBSCurve inputs handled by PostGIS. TIN and PolyhedralSurface inputs are not supported.

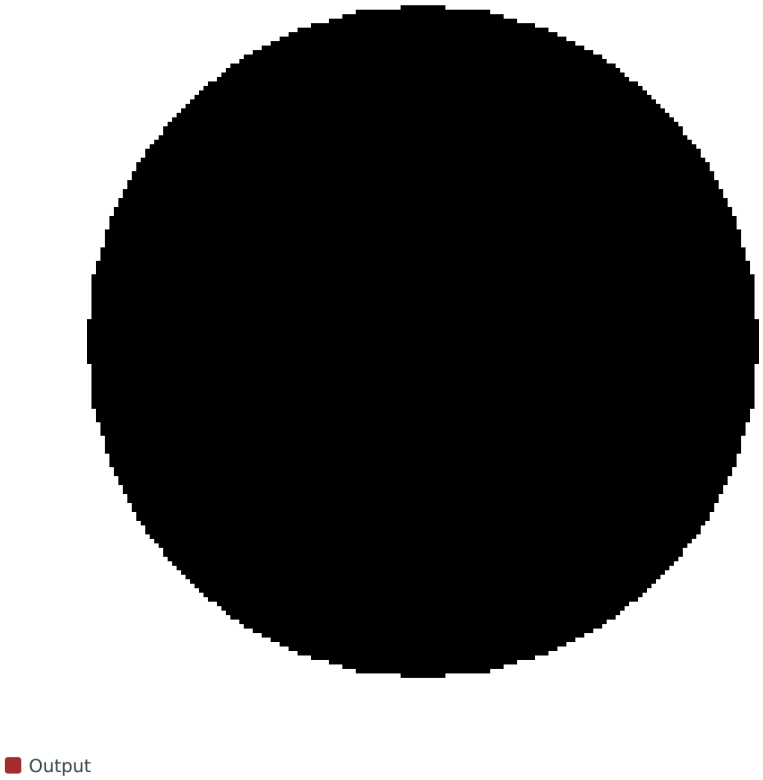
Exempel: Utdata av geometrier som PNG-filer

This example outputs a black circle on a white background, occupying 150 by 150 pixels.

Code

```
SELECT ST_AsPNG(
    ST_AsRaster(
        ST_Buffer(ST_Point(1, 5), 10),
        150, 150,
        ARRAY['8BUI', '8BUI', '8BUI'],
        ARRAY[0, 0, 0],
        ARRAY[255, 255, 255]
    )
) AS image;
```

Figure

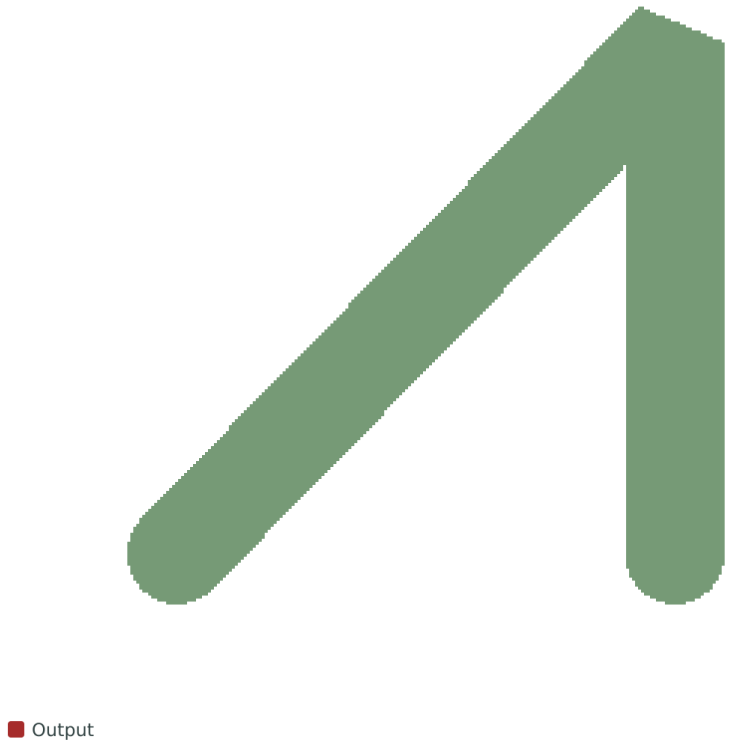


This example maps bands to RGB channels, producing the teal value (118,154,118).

Code

```
SELECT ST_AsPNG(
    ST_AsRaster(
        ST_Buffer('LINESTRING(50 50,150 150,150 50)'::geometry, 10, 'join=bevel'),
        200, 200,
        ARRAY['8BUI', '8BUI', '8BUI'],
        ARRAY[118, 154, 118],
        ARRAY[255, 255, 255]
    )
) AS image;
```

Figure



Se även

ST_BandPixelType, ST_Buffer, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsJPEG, ST_SRID

11.3.3 ST_AsRasterAgg

ST_AsRasterAgg — Aggregera. Renderar PostGIS-geometrier till ett nytt raster.

Synopsis

raster **ST_AsRasterAgg**(geometry geom, double precision val, raster ref, text pixeltype, double precision nodataval, text uniontype, boolean touched);

Description

Returnerar ett enbandsraster som innehåller den renderade versionen av alla inkommande geometrier, var och en med sitt associerade värde.

Tillgänglighet: 3.6.0

Exempel

Code

```
WITH inp(g, v) AS (
    VALUES
        ( ST_Buffer(ST_MakePoint(10, 0), 10), 1 ),
        ( ST_Buffer(ST_MakePoint(20, 0), 10), 2 )
),
agg AS (
    SELECT ST_AsRasterAgg(
        g,
        v,
        ST_MakeEmptyRaster(0, 0, 0, 0, 1.0),
        '8BUI',
        99,
        'SUM',
        true
    ) r
    FROM inp
)
SELECT
    ST_Width(r) w,
    ST_Height(r) h,
    ST_Value(r, 'POINT(5 0)') v5_0,
    ST_Value(r, 'POINT(15 0)') v15_0,
    ST_Value(r, 'POINT(25 0)') v25_0
FROM agg;
```

Output

```
w  | h  | v5_0 | v15_0 | v25_0
----+----+------+-------+-------
 30 | 20 |    1 |     3 |     2
(1 row)
```

Se även

ST_AsRaster, ST_DumpAsPolygons, ST_Union

11.3.4 ST_Band

ST_Band — Returnerar ett eller flera band från ett befintligt raster som ett nytt raster. Användbart för att bygga nya raster från befintliga raster.

Synopsis

raster **ST_Band**(raster rast, integer[] nbands = ARRAY[1]);
raster **ST_Band**(raster rast, integer nband);
raster **ST_Band**(raster rast, text nbands, character delimiter=,);

Description

Returnerar ett eller flera band i ett befintligt raster som ett nytt raster. Användbar för att bygga nya raster från befintliga raster eller exportera endast utvalda band i ett raster eller omorganisera ordningen på banden i ett raster. Om inget band anges eller om något av de angivna banden inte finns i rastret returneras alla band. Används som en hjälpfunktion i olika funktioner, t.ex. för att radera ett band.

Varning
För funktionen nbands as textvariant är standardavgränsaren , vilket innebär att du kan be om ”1,2,3” och om du vill använda en annan avgränsare gör du ST_Band(rast, ’1@2@3’, ’@’). För att be om flera band rekommenderar vi starkt att du använder array-formen av denna funktion, t.ex. ST_Band(rast, ’{1,2,3}’::int[]); eftersom textlistan med band kan tas bort i framtida versioner av PostGIS.

Tillgänglighet: 2.0.0

Exempel

This example creates two rasters: one containing band 1 of dummy_rast, and one containing band 2 reclassified as 2BUI.

Code

```
SELECT ST_NumBands(rast1) As numb1, ST_BandPixelType(rast1) As pix1,
 ST_NumBands(rast2) As numb2, ST_BandPixelType(rast2) As pix2
FROM (
    SELECT ST_Band(rast) As rast1, ST_Reclass(ST_Band(rast, 3), '100-200):1, [200-254:2', ' ↩
    2BUI') As rast2
        FROM dummy_rast
        WHERE rid = 2) As foo;
```

Output

```
numb1 | pix1 | numb2 | pix2
------+------+------+------
    1 | 8BUI |     1 | 2BUI
```

Return bands 2 and 3 using array cast syntax.

Code

```
SELECT ST_NumBands(ST_Band(rast, '{2,3}'::int[])) As num_bands
FROM dummy_rast WHERE rid=2;
```

Output

```
2
```

Return bands 2 and 3 using array syntax.

Code

```
SELECT ST_NumBands(ST_Band(rast, ARRAY[2, 3])) As num_bands
    FROM dummy_rast
WHERE rid=2;
```

Output

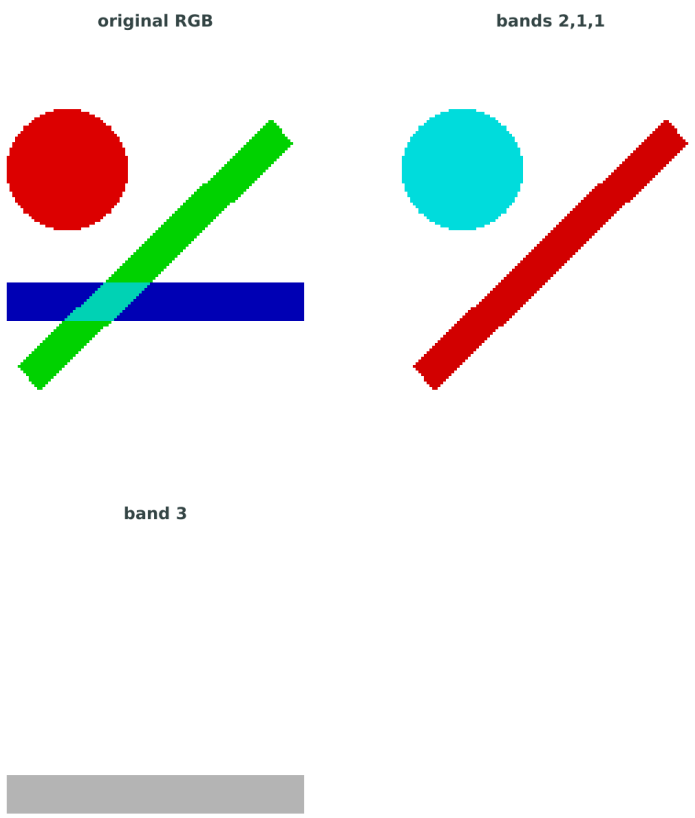
```
2
```

This example uses three distinct shapes as the red, green, and blue bands of a synthetic raster. It compares the original RGB raster with band 2 followed by band 1 twice, and with band 3 alone.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_AddBand(
            ST_AddBand(
                ST_MakeEmptyRaster(128, 128, 0, 128, 1, -1, 0, 0, 0),
                '8BUI'::text, 0::double precision, 0::double precision
            ),
            '8BUI'::text, 0::double precision, 0::double precision
        ),
        '8BUI'::text, 0::double precision, 0::double precision
    ) AS rast
), bands AS (
    SELECT ST_SetValue(
        ST_SetValue(
            ST_SetValue(rast, 1, ST_Buffer('POINT(32 96)'::geometry, 22), 220),
            2, ST_Buffer('LINESTRING(18 20,110 110)'::geometry, 6,
                         'endcap=flat join=bevel'), 210
        ),
        3, ST_Buffer('LINESTRING(10 48,118 48)'::geometry, 7,
                     'endcap=flat'), 180
    ) AS rast
    FROM canvas
)
SELECT ST_AsPNG(rast) AS "original RGB",
    ST_AsPNG(ST_Band(rast, ARRAY[2, 1, 1])) AS "bands 2,1,1",
    ST_AsPNG(ST_Band(rast, 3)) AS "band 3"
FROM bands;
```

Figure



Band selection from a multi-band raster.

original (kolumn rast)

dupe_band

sing_band

Se även

ST_AddBand, ST_NumBands, ST_Reclass, Kapitel 11

11.3.5 ST_MakeEmptyCoverage

ST_MakeEmptyCoverage — Täck georefererat område med ett rutnät av tomma rasterplattor.

Synopsis

raster **ST_MakeEmptyCoverage**(integer tilewidth, integer tileheight, integer width, integer height, double precision upperleftx, double precision upperlefty, double precision scalex, double precision scaley, double precision skewx, double precision skewy, integer srid=unknown);

Description

Skapa en uppsättning rasterplattor med ST_MakeEmptyRaster. Grid-dimensionen är `bredd` och `höjd`. Plattdimensionen är `tilewidth` och `tileheight`. Det täckta georefererade området är från övre vänstra hörnet (`upperleftx`, `upperlefty`) till nedre högra hörnet (upperleftx + `bredd` * `scalex`, `upperlefty` + `höjd` * `scaley`).

Notera
Observera att scaley i allmänhet är negativt för raster och scalex i allmänhet är positivt. Så det nedre högra hörnet kommer att ha ett lägre y-värde och ett högre x-värde än det övre vänstra hörnet.

Tillgänglighet: 2.4.0

Exempel Grundläggande

Skapa 16 plattor i ett 4x4 rutnät för att täcka WGS84-området från övre vänstra hörnet (22, 77) till nedre högra hörnet (55, 33).

Code

```
SELECT (ST_MetaData(tile)).* FROM ST_MakeEmptyCoverage(1, 1, 4, 4, 22, 33, (55 - 22)/(4):: ↩
    float, (33 - 77)/(4)::float, 0., 0., 4326) tile;
```

Output

```
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands
-----------------------------------------------------------------------------------------
         22 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      30.25 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
       38.5 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      46.75 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
         22 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      30.25 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
       38.5 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      46.75 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
         22 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      30.25 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
       38.5 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      46.75 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
         22 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      30.25 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
       38.5 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      46.75 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
```

Se även

ST_MakeEmptyRaster

11.3.6 ST_MakeEmptyRaster

ST_MakeEmptyRaster — Returnerar ett tomt raster (utan band) med givna dimensioner (bredd & höjd), övre vänstra X och Y, pixelstorlek och rotation (scalex, scaley, skewx & skewy) och referenssystem (srid). Om ett raster skickas in returneras ett nytt raster med samma storlek, inriktning och SRID. Om srid utelämnas sätts den spatiala ref till okänd (0).

Synopsis

raster **ST_MakeEmptyRaster**(raster rast);
raster **ST_MakeEmptyRaster**(integer width, integer height, float8 upperleftx, float8 upperlefty, float8 scalex, float8 scaley, float8 skewx, float8 skewy, integer srid=unknown);
raster **ST_MakeEmptyRaster**(integer width, integer height, float8 upperleftx, float8 upperlefty, float8 pixelsize);

Description

Returnerar ett tomt raster (utan band) med givna dimensioner (bredd & höjd) och georefererat i spatiala (eller världsliga) koordinater med övre vänstra X (upperleftx), övre vänstra Y (upperlefty), pixelstorlek och rotation (scalex, scaley, skewx & skewy) och referenssystem (srid).

Den senaste versionen använder en enda parameter för att ange pixelstorleken (pixelsize). scalex sätts till detta argument och scaley sätts till det negativa värdet av detta argument. skewx och skewy sätts till 0.

Om ett befintligt raster skickas in returneras ett nytt raster med samma inställningar för metadata (utan banden).

Om ingen srid anges är standardvärdet 0. När du har skapat ett tomt raster vill du förmodligen lägga till band i det och kanske redigera det. Se ST_AddBand för att definiera band och ST_SetValue för att ställa in initiala pixelvärden.

Exempel

Use an existing raster as a template and output metadata of the rasters we added.

Code

```
INSERT INTO dummy_rast(rid, rast)
VALUES(3, ST_MakeEmptyRaster(100, 100, 0.0005, 0.0005, 1, 1, 0, 0, 4326) );
INSERT INTO dummy_rast(rid, rast)
SELECT 4, ST_MakeEmptyRaster(rast)
FROM dummy_rast WHERE rid = 3;
SELECT
    rid,
    (md).width,
    (md).height,
    (md).scalex,
    (md).scaley,
    (md).srid
FROM (
    SELECT rid, ST_MetaData(rast) AS md
    FROM dummy_rast
    WHERE rid IN (3, 4)
) AS foo
ORDER BY rid;
```

Output

```
rid | width | height | scalex | scaley | srid
-----+-------+--------+--------+--------+------
   3 |   100 |    100 |      1 |      1 | 4326
   4 |   100 |    100 |      1 |      1 | 4326
```

Se även

ST_AddBand, ST_MetaData, ST_ScaleX, ST_ScaleY, ST_SetValue, ST_SkewX, , ST_SkewY

11.3.7 ST_Tile

ST_Tile — Returnerar en uppsättning raster som är resultatet av uppdelningen av inmatningsrastret baserat på de önskade dimensionerna för utdatarastren.

Synopsis

setof raster **ST_Tile**(raster rast, int[] nband, integer width, integer height, boolean padwithnodata=FALSE, double precision nodataval=NULL);
setof raster **ST_Tile**(raster rast, integer nband, integer width, integer height, boolean padwithnodata=FALSE, double precision nodataval=NULL);
setof raster **ST_Tile**(raster rast, integer width, integer height, boolean padwithnodata=FALSE, double precision nodataval=NULL);

Description

Returnerar en uppsättning raster som är resultatet av uppdelningen av inmatningsrastret baserat på de önskade dimensionerna för utdatarastren.

Om `padwithnodata` = FALSE kan kantplattor på rastrets högra och nedre sidor ha andra dimensioner än resten av plattorna. Om `padwithnodata` = TRUE har alla plattor samma dimensioner, men det är möjligt att kantplattor fylls på med NODATA-värden. Om rasterband inte har NODATA-värden angivna kan ett sådant anges genom att ställa in `nodataval`.

Notera
Om ett specificerat band i indatarastret är out-of-db, kommer motsvarande band i utdatarastren också att vara out-of-db.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     1, 0), 2, '8BUI', 10, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     2, 0), 2, '8BUI', 20, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     3, 0), 2, '8BUI', 30, 0) AS rast UNION ALL
```

```
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 4, 0), 2, '8BUI', 40, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 5, 0), 2, '8BUI', 50, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 6, 0), 2, '8BUI', 60, 0) AS rast UNION ALL

    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 7, 0), 2, '8BUI', 70, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 8, 0), 2, '8BUI', 80, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 9, 0), 2, '8BUI', 90, 0) AS rast
), bar AS (
    SELECT ST_Union(rast) AS rast FROM foo
), baz AS (
    SELECT
        row_number() OVER (
            ORDER BY ST_UpperLeftY(rast) DESC, ST_UpperLeftX(rast)
        ) AS tile_id,
        rast
    FROM (
        SELECT ST_Tile(rast, 3, 3, TRUE) AS rast
        FROM bar
    ) AS tiled
)
SELECT
    tile_id,
    ST_UpperLeftX(rast) AS ulx,
    ST_UpperLeftY(rast) AS uly,
    ST_Value(rast, 1, 1, 1) AS band1,
    ST_Value(rast, 2, 1, 1) AS band2
FROM baz;
```

Output

```
tile_id | ulx | uly | band1 | band2
---------+-----+-----+-------+-------
       1 |   0 |   0 |     1 |    10
       2 |   3 |   0 |     2 |    20
       3 |   6 |   0 |     3 |    30
       4 |   0 |  -3 |     4 |    40
       5 |   3 |  -3 |     5 |    50
       6 |   6 |  -3 |     6 |    60
       7 |   0 |  -6 |     7 |    70
       8 |   3 |  -6 |     8 |    80
       9 |   6 |  -6 |     9 |    90
(9 rows)
```

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     1, 0), 2, '8BUI', 10, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     2, 0), 2, '8BUI', 20, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     3, 0), 2, '8BUI', 30, 0) AS rast UNION ALL

    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 4, 0), 2, '8BUI', 40, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 5, 0), 2, '8BUI', 50, 0) AS rast UNION ALL
```

```
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 6, 0), 2, '8BUI', 60, 0) AS rast UNION ALL

    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 7, 0), 2, '8BUI', 70, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 8, 0), 2, '8BUI', 80, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 9, 0), 2, '8BUI', 90, 0) AS rast
), bar AS (
    SELECT ST_Union(rast) AS rast FROM foo
), baz AS (
    SELECT
        row_number() OVER (
            ORDER BY ST_UpperLeftY(rast) DESC, ST_UpperLeftX(rast)
        ) AS tile_id,
        rast
    FROM (
        SELECT ST_Tile(rast, ARRAY[2], 3, 3) AS rast
        FROM bar
    ) AS tiled
)
SELECT
    tile_id,
    ST_UpperLeftX(rast) AS ulx,
    ST_UpperLeftY(rast) AS uly,
    ST_Value(rast, 1, 1, 1) AS band_value
FROM baz;
```

Output

```
tile_id | ulx | uly | band_value
---------+-----+-----+------------
       1 |   0 |   0 |         10
       2 |   3 |   0 |         20
       3 |   6 |   0 |         30
       4 |   0 |  -3 |         40
       5 |   3 |  -3 |         50
       6 |   6 |  -3 |         60
       7 |   0 |  -6 |         70
       8 |   3 |  -6 |         80
       9 |   6 |  -6 |         90
(9 rows)
```

Se även

ST_Union, ST_Retile

11.3.8 ST_Retile

ST_Retile — Returnerar en uppsättning konfigurerade plattor från en godtyckligt uppdelad rastertäckning.

Synopsis

SETOF raster **ST_Retile**(regclass tab, name col, geometry ext, float8 sfx, float8 sfy, int tw, int th, text algo='NearestNeighbor');

Description

Returnerar en uppsättning plattor med angiven skala (`sfx`, `sfy`) och maxstorlek (`tw`, `th`) och som täcker den angivna utsträckningen (`ext`) med data som kommer från den angivna rastertäckningen (`tab`, `col`).

Algoritmalternativen är: NearestNeighbor", Bilinear", "Cubic", "CubicSplineöch Lanczos". Hänvisa till: GDAL Warp resampling methods för mer information.

Tillgänglighet: 2.2.0

Se även

ST_CreateOverview

11.3.9 ST_FromGDALRaster

ST_FromGDALRaster — Returnerar ett raster från en GDAL-rasterfil som stöds.

Synopsis

raster **ST_FromGDALRaster**(bytea gdaldata, integer srid=NULL);

Description

Returnerar ett raster från en GDAL-rasterfil som stöds. `gdaldata` är av typen bytea och bör vara innehållet i GDAL-rasterfilen.

Om `srid` är NULL försöker funktionen automatiskt tilldela SRID från GDAL-rastret. Om `srid` anges kommer det angivna värdet att åsidosätta eventuellt automatiskt tilldelad SRID.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_AsPNG(ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 0.1,  ↩
    -0.1, 0, 0, 4326), 1, '8BUI', 1, 0), 2, '8BUI', 2, 0), 3, '8BUI', 3, 0)) AS png
),
bar AS (
    SELECT 1 AS rid, ST_FromGDALRaster(png) AS rast FROM foo
    UNION ALL
    SELECT 2 AS rid, ST_FromGDALRaster(png, 3310) AS rast FROM foo
)
SELECT
    rid,
    ST_MetaData(rast) AS metadata,
    ST_SummaryStats(rast, 1) AS stats1,
    ST_SummaryStats(rast, 2) AS stats2,
    ST_SummaryStats(rast, 3) AS stats3
FROM bar
ORDER BY rid;
```

Output

```
rid |         metadata          |    stats1     |    stats2     |     stats3
----+---------------------------+---------------+---------------+----------------
   1 | (0,0,2,2,1,-1,0,0,0,3)    | (4,4,1,0,1,1) | (4,8,2,0,2,2) | (4,12,3,0,3,3)
   2 | (0,0,2,2,1,-1,0,0,3310,3) | (4,4,1,0,1,1) | (4,8,2,0,2,2) | (4,12,3,0,3,3)
(2 rows)
```

Se även

ST_AsGDALRaster

11.4 Raster-accessorer

11.4.1 ST_GeoReference

ST_GeoReference — Returnerar metadata för georeferenser i GDAL- eller ESRI-format, vilket är vanligt förekommande i en world-fil. Standard är GDAL.

Synopsis

text **ST_GeoReference**(raster rast, text format=GDAL);

Description

Returnerar metadata för georeferensen inklusive vagnsretur i GDAL- eller ESRI-format, vilket är vanligt förekommande i en world-fil. Standard är GDAL om ingen typ anges. typ är strängen 'GDAL' eller 'ESRI'.

Skillnaden mellan formatrepresentationer är följande:

GDAL:

Output

```
scalex
skewy
skewx
scaley
upperleftx
upperlefty
```

ESRI:

Output

```
scalex
skewy
skewx
scaley
upperleftx + scalex*0.5
upperlefty + scaley*0.5
```

Exempel

Code

```
SELECT
    format,
    replace(ST_GeoReference(rast, format), E'\n', ' | ') AS georef
FROM dummy_rast
CROSS JOIN (VALUES ('ESRI'), ('GDAL')) AS formats(format)
WHERE rid = 1;
```

Output

```
format | georef
--------+-------------------------------------------------------------------------
 ESRI   | 2.0000000000 | 0.0000000000 | 0.0000000000 | 3.0000000000 | 1.5000000000 | 2.0000000000
 GDAL   | 2.0000000000 | 0.0000000000 | 0.0000000000 | 3.0000000000 | 0.5000000000 | 0.5000000000
(2 rows)
```

Se även

ST_SetGeoReference, ST_ScaleX, ST_ScaleY

11.4.2 ST_Height

ST_Height — Returnerar höjden på rastret i pixlar.

Synopsis

integer **ST_Height**(raster rast);

Description

Returnerar höjden på rastret.

Exempel

Code

```
SELECT rid, ST_Height(rast) As rastheight
FROM dummy_rast;
```

Output

```
rid | rastheight
-----+------------
   1 |         20
   2 |          5
```

Se även

ST_Width

11.4.3 ST_IsEmpty

ST_IsEmpty — Returnerar true om rastret är tomt (bredd = 0 och höjd = 0). I annat fall returneras false.

Synopsis

boolean **ST_IsEmpty**(raster rast);

Description

Returnerar true om rastret är tomt (bredd = 0 och höjd = 0). I annat fall returneras false.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(100, 100, 0, 0, 0, 0, 0, 0))
```

Output

```
f
```

Code

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(0, 0, 0, 0, 0, 0, 0, 0))
```

Output

```
t
```

Se även

ST_HasNoBand

11.4.4 ST_MemSize

ST_MemSize — Returnerar den mängd utrymme (i byte) som rastret tar upp.

Synopsis

integer **ST_MemSize**(raster rast);

Description

Returnerar den mängd utrymme (i byte) som rastret tar upp.

Detta är ett trevligt komplement till PostgreSQL inbyggda funktioner pg_column_size, pg_size_pretty, pg_relation_size, pg_total_relation_size.

Notera
pg_relation_size som ger bytestorleken för en tabell kan returnera en bytestorlek som är lägre än ST_MemSize. Detta beror på att pg_relation_size inte lägger till toasted-tabellbidrag och stora geometrier lagras i TOAST-tabeller. pg_column_size kan returnera lägre eftersom den returnerar den komprimerade storleken.
pg_total_relation_size - inkluderar tabellen, de toastade tabellerna och indexen.

Tillgänglighet: 2.2.0

Exempel

Code

```
SELECT ST_MemSize(ST_AsRaster(ST_Buffer(ST_Point(1, 5), 10, 1000), 150, 150, '8BUI')) AS ↩
    rast_mem;
```

Output

```
22568
```

Se även

11.4.5 ST_MetaData

ST_MetaData — Returnerar grundläggande metadata om ett rasterobjekt, t.ex. pixelstorlek, rotation (skew), övre, nedre vänster etc.

Synopsis

record **ST_MetaData**(raster rast);

Description

Returnerar grundläggande metadata om ett rasterobjekt, t.ex. pixelstorlek, rotation (skew), övre och nedre vänster etc. Kolumner som returneras: upperleftx | upperlefty | bredd | höjd | scalex | scaley | skewx | skewy | srid | numbands

Exempel

Code

```
SELECT
    rid,
    format('(%s, %s)', (md).upperleftx, (md).upperlefty) AS upper_left,
    format('%sx%s', (md).width, (md).height) AS size,
    format('(%s, %s)', (md).scalex, (md).scaley) AS scale,
    format('(%s, %s)', (md).skewx, (md).skewy) AS skew,
    (md).srid AS srid,
    (md).numbands AS numbands
FROM (
    SELECT rid, ST_MetaData(rast) AS md
    FROM dummy_rast
) AS foo
ORDER BY rid;
```

Output

```
rid |        upper_left        | size  |      scale      |  skew  | srid | numbands
-----+--------------------------+-------+-----------------+--------+------+----------
   1 | (0.5, 0.5)               | 10x20 | (2, 3)          | (0, 0) |    0 |        0
   2 | (3427927.75, 5793244)    | 5x5   | (0.05, -0.05)   | (0, 0) |    0 |        3
(2 rows)
```

Se även

ST_BandMetaData, ST_NumBands

11.4.6 ST_NumBands

ST_NumBands — Returnerar antalet band i rasterobjektet.

Synopsis

integer **ST_NumBands**(raster rast);

Description

Returnerar antalet band i rasterobjektet.

Exempel

Code

```
SELECT rid, ST_NumBands(rast) As numbands
FROM dummy_rast;
```

Output

```
rid | numbands
----+----------
  1 |        0
  2 |        3
```

Se även

ST_Value

11.4.7 ST_PixelHeight

ST_PixelHeight — Returnerar pixelhöjden i geometriska enheter i det spatiala referenssystemet.

Synopsis

double precision **ST_PixelHeight**(raster rast);

Description

Returnerar höjden på en pixel i geometriska enheter i det spatiala referenssystemet. I det vanliga fallet där det inte finns någon skevhet är pixelhöjden bara skalförhållandet mellan geometriska koordinater och rasterpixlar.

Se ST_PixelWidth för en schematisk visualisering av förhållandet.

Exempel

Rasters with no skew.

Code

```
SELECT ST_Height(rast) As rastheight, ST_PixelHeight(rast) As pixheight,
 ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
        ST_SkewY(rast) As skewy
FROM dummy_rast;
```

Output

```
rastheight | pixheight | scalex | scaley | skewx | skewy
-----------+-----------+--------+--------+-------+----------
        20 |         3 |      2 |      3 |     0 |        0
         5 |      0.05 |   0.05 |  -0.05 |     0 |        0
```

Rasters with skew different than 0.

Code

```
SELECT ST_Height(rast) As rastheight, ST_PixelHeight(rast) As pixheight,
 ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
        ST_SkewY(rast) As skewy
FROM (SELECT ST_SetSkew(rast, 0.5, 0.5) As rast
        FROM dummy_rast) As skewed;
```

Output

```
rastheight |     pixheight     | scalex | scaley | skewx | skewy
-----------+-------------------+--------+--------+-------+----------
        20 |  3.04138126514911 |      2 |      3 |   0.5 |      0.5
         5 | 0.502493781056044 |   0.05 |  -0.05 |   0.5 |      0.5
```

Se även

ST_PixelWidth, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.4.8 ST_PixelWidth

ST_PixelWidth — Returnerar pixelbredden i geometriska enheter i det spatiala referenssystemet.

Synopsis

double precision **ST_PixelWidth**(raster rast);

Description

Returnerar bredden på en pixel i geometriska enheter i det spatiala referenssystemet. I det vanliga fallet där det inte finns någon skevhet är pixelbredden bara skalförhållandet mellan geometriska koordinater och rasterpixlar.

The following executable example draws the two pixel basis vectors on the same coordinate frame as a skewed raster footprint. The `i direction` vector is `(scaleX, skewY)`; the `j direction` vector is `(skewX, scaleY)`.

Exempel

Pixel basis vectors and footprint for a raster with non-zero skew.

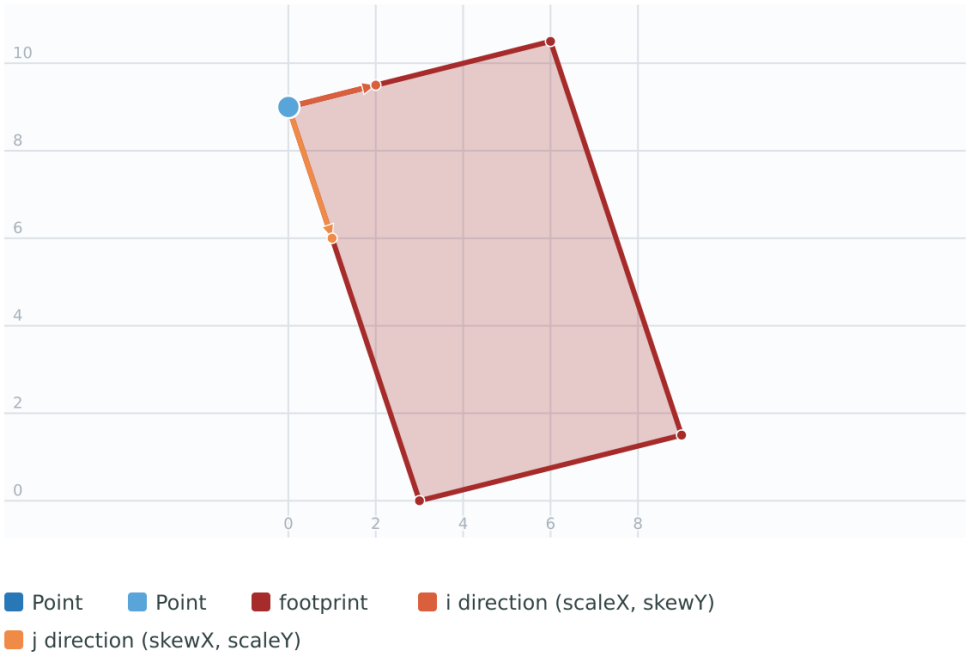
Code

```
WITH raster AS (
    SELECT ST_MakeEmptyRaster(
        3, 3, 0, 0,
        2, -3, 1, 0.5,
        0
    ) AS rast
)
SELECT
    ST_ConvexHull(rast) AS footprint,
    ST_MakeLine(
        ST_Point(0, 0),
        ST_Point(ST_ScaleX(rast), ST_SkewY(rast))
    ) AS "i direction (scaleX, skewY)",
    ST_MakeLine(
        ST_Point(0, 0),
        ST_Point(ST_SkewX(rast), ST_ScaleY(rast))
    ) AS "j direction (skewX, scaleY)"
FROM raster;
```

Output

```
POLYGON((0 0,6 1.5,9 -7.5,3 -9,0 0)) | LINESTRING(0 0,2 0.5) | LINESTRING(0 0,1 -3)
```

Figure



Rasters with no skew.

Code

```
SELECT ST_Width(rast) As rastwidth, ST_PixelWidth(rast) As pixwidth,
ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
ST_SkewY(rast) As skewy
FROM dummy_rast;
```

Output

```
rastwidth | pixwidth | scalex | scaley | skewx | skewy
----------+----------+--------+--------+-------+----------
10 |        2 |      2 |      3 |     0 |        0
 5 |     0.05 |   0.05 |  -0.05 |     0 |        0
```

Rasters with skew different than 0.

Code

```
SELECT ST_Width(rast) As rastwidth, ST_PixelWidth(rast) As pixwidth,
ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
ST_SkewY(rast) As skewy
FROM (SELECT ST_SetSkew(rast, 0.5, 0.5) As rast
FROM dummy_rast) As skewed;
```

Output

```
rastwidth |     pixwidth      | scalex | scaley | skewx | skewy
----------+-------------------+--------+--------+-------+----------
10 |  2.06155281280883 |      2 |      3 |   0.5 |      0.5
 5 | 0.502493781056044 |   0.05 |  -0.05 |   0.5 |      0.5
```

Se även

ST_PixelHeight, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.4.9 ST_ScaleX

ST_ScaleX — Returnerar X-komponenten av pixelbredden i enheter av koordinatreferenssystemet.

Synopsis

float8 **ST_ScaleX**(raster rast);

Description

Returnerar X-komponenten av pixelbredden i enheter av koordinatreferenssystemet. Se World File för mer information.

Ändrad: 2.0.0. I WKTRaster-versioner kallades detta ST_PixelSizeX.

Exempel

Code

```
SELECT rid, ST_ScaleX(rast) As rastpixwidth
FROM dummy_rast;
```

Output

```
rid | rastpixwidth
-----+--------------
   1 |            2
   2 |         0.05
```

Se även

ST_Width

11.4.10 ST_ScaleY

ST_ScaleY — Returnerar Y-komponenten av pixelhöjden i enheter av koordinatreferenssystemet.

Synopsis

float8 **ST_ScaleY**(raster rast);

Description

Returnerar Y-komponenten av pixelhöjden i enheter av koordinatreferenssystemet. Kan vara negativ. Se World File för mer information.

Ändrad: 2.0.0. I WKTRaster-versioner kallades detta ST_PixelSizeY.

Exempel

Code

```
SELECT rid, ST_ScaleY(rast) As rastpixheight
FROM dummy_rast;
```

Output

```
rid | rastpixheight
-----+--------------
   1 |             3
   2 |         -0.05
```

Se även

ST_Height

11.4.11 ST_RasterToWorldCoord

ST_RasterToWorldCoord — Returnerar rastrets övre vänstra hörn som geometriska X och Y (longitud och latitud) givet en kolumn och rad. Kolumn och rad börjar på 1.

Synopsis

record **ST_RasterToWorldCoord**(raster rast, integer xcolumn, integer yrow);

Description

Returnerar det övre vänstra hörnet som geometriska X och Y (longitud och latitud) givet en kolumn och rad. X och Y som returneras är i geometriska enheter för det georefererade rastret. Numreringen av kolumn och rad börjar på 1, men om någon av parametrarna är noll, ett negativt tal eller ett tal som är större än respektive dimension av rastret, kommer den att returnera koordinater utanför rastret förutsatt att rastrets rutnät är tillämpligt utanför rastrets gränser.

Tillgänglighet: 2.1.0

Exempel

This example uses a non-skewed raster.

Code

```
SELECT
    rid,
    (ST_RasterToWorldCoord(rast, 1, 1)).*,
    (ST_RasterToWorldCoord(rast, 2, 2)).*
FROM dummy_rast
```

Output

```
rid | longitude  | latitude | longitude |  latitude
-----+------------+----------+-----------+-----------
   1 |        0.5 |      0.5 |       2.5 |        3.5
   2 | 3427927.75 |  5793244 | 3427927.8 | 5793243.95
```

This example uses a skewed raster.

Code

```
SELECT
    rid,
    (ST_RasterToWorldCoord(rast, 1, 1)).*,
    (ST_RasterToWorldCoord(rast, 2, 3)).*
FROM (
    SELECT
        rid,
        ST_SetSkew(rast, 100.5, 0) As rast
    FROM dummy_rast
) As foo
```

Output

```
rid | longitude  | latitude | longitude | latitude
----+------------+----------+-----------+----------
  1 |        0.5 |      0.5 |     203.5 |       6.5
  2 | 3427927.75 |  5793244 | 3428128.8 | 5793243.9
```

Se även

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SetSkew

11.4.12 ST_RasterToWorldCoordX

ST_RasterToWorldCoordX — Returnerar den geometriska X-koordinaten uppe till vänster i ett raster, en kolumn och en rad. Numreringen av kolumner och rader börjar på 1.

Synopsis

float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn);
float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn, integer yrow);

Description

Returnerar den övre vänstra X-koordinaten för en rasterkolumnrad i geometriska enheter för det georefererade rastret. Numreringen av kolumner och rader börjar på 1, men om du anger ett negativt tal eller ett tal som är högre än antalet kolumner i rastret, får du koordinater utanför rasterfilen till vänster eller höger med antagandet att skevheten och pixelstorlekarna är desamma som i det valda rastret.

Notera
För raster utan skevhet räcker det med att ange X-kolumnen. För skeva raster är den georefererade koordinaten en funktion av ST_ScaleX och ST_SkewX samt rad och kolumn. Ett felmeddelande visas om du bara anger X-kolumnen för ett snedställt raster.

Ändrad: 2.1.0 I tidigare versioner kallades detta ST_Raster2WorldCoordX

Exempel

This example shows that providing a column is sufficient for a non-skewed raster.

Code

```
SELECT rid, ST_RasterToWorldCoordX(rast, 1) As x1coord,
    ST_RasterToWorldCoordX(rast, 2) As x2coord,
    ST_ScaleX(rast) As pixelx
FROM dummy_rast;
```

Output

```
rid |  x1coord   |  x2coord  | pixelx
-----+------------+-----------+--------
   1 |        0.5 |       2.5 |      2
   2 | 3427927.75 | 3427927.8 |   0.05
```

For fun let's skew it.

Code

```
SELECT rid, ST_RasterToWorldCoordX(rast, 1, 1) As x1coord,
    ST_RasterToWorldCoordX(rast, 2, 3) As x2coord,
    ST_ScaleX(rast) As pixelx
FROM (SELECT rid, ST_SetSkew(rast, 100.5, 0) As rast FROM dummy_rast) As foo;
```

Output

```
rid |  x1coord   |  x2coord  | pixelx
-----+------------+-----------+--------
   1 |        0.5 |     203.5 |      2
   2 | 3427927.75 | 3428128.8 |   0.05
```

Se även

ST_ScaleX, ST_RasterToWorldCoordY, ST_SetSkew, ST_SkewX

11.4.13 ST_RasterToWorldCoordY

ST_RasterToWorldCoordY — Returnerar den geometriska Y-koordinaten i övre vänstra hörnet av ett raster, en kolumn och en rad. Numreringen av kolumner och rader börjar på 1.

Synopsis

float8 **ST_RasterToWorldCoordY**(raster rast, integer yrow);
float8 **ST_RasterToWorldCoordY**(raster rast, integer xcolumn, integer yrow);

Description

Returnerar den övre vänstra Y-koordinaten för en rasterkolumnrad i geometriska enheter för det georefererade rastret. Numreringen av kolumner och rader börjar på 1, men om du anger ett negativt tal eller ett tal som är högre än antalet kolumner/rader i rastret, får du koordinater utanför rasterfilen till vänster eller höger med antagandet att skevheten och pixelstorlekarna är desamma som för den valda rasterplattan.

Notera
För raster som inte är snedställda räcker det att ange Y-kolumnen. För skeva raster är den georefererade koordinaten en funktion av ST_ScaleY och ST_SkewY samt rad och kolumn. Ett felmeddelande visas om du bara anger Y-raden för ett snedställt raster.

Ändrad: 2.1.0 I tidigare versioner kallades detta ST_Raster2WorldCoordY

Exempel

Non-skewed raster providing row is sufficient.

Code

```
SELECT rid, ST_RasterToWorldCoordY(rast, 1) As y1coord,
    ST_RasterToWorldCoordY(rast, 3) As y2coord,
    ST_ScaleY(rast) As pixely
FROM dummy_rast;
```

Output

```
rid | y1coord |  y2coord  | pixely
-----+---------+-----------+--------
   1 |     0.5 |       6.5 |      3
   2 | 5793244 | 5793243.9 |  -0.05
```

For fun let's skew it.

Code

```
SELECT rid, ST_RasterToWorldCoordY(rast, 1, 1) As y1coord,
    ST_RasterToWorldCoordY(rast, 2, 3) As y2coord,
    ST_ScaleY(rast) As pixely
FROM (SELECT rid, ST_SetSkew(rast, 0, 100.5) As rast FROM dummy_rast) As foo;
```

Output

```
rid | y1coord |  y2coord  | pixely
-----+---------+-----------+--------
   1 |     0.5 |       107 |      3
   2 | 5793244 | 5793344.4 |  -0.05
```

Se även

ST_ScaleY, ST_RasterToWorldCoordX, ST_SetSkew, ST_SkewY

11.4.14 ST_Rotation

ST_Rotation — Returnerar rastrets rotation i radianer.

Synopsis

float8 **ST_Rotation**(raster rast);

Description

Returnerar rastrets enhetliga rotation i radianer. Om ett raster inte har enhetlig rotation returneras NaN. Se World File för mer information.

Exempel

Code

```
SELECT rid, ST_Rotation(ST_SetScale(ST_SetSkew(rast, sqrt(2)), sqrt(2))) AS rot FROM  ↩
    dummy_rast;
```

Output

```
rid |        rot
----+-------------------
  1 | 0.785398163397448
  2 | 0.785398163397448
```

Se även

ST_SetRotation, ST_SetScale, ST_SetSkew

11.4.15 ST_SkewX

ST_SkewX — Returnerar georeferensens X skew (eller rotationsparameter).

Synopsis

float8 **ST_SkewX**(raster rast);

Description

Returnerar georeferensens X-skevhet (eller rotationsparameter). Se World File för mer information.

Exempel

Code

```
WITH refs AS (
    SELECT
        rid,
        format('(%s, %s)', ST_SkewX(rast), ST_SkewY(rast)) AS skew,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
)
SELECT
    rid,
    skew,
    term,
    georef[idx] AS value
FROM refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY rid, idx;
```

Output

```
rid |  skew  |  term   |       value
-----+--------+---------+--------------------
   1 | (0, 0) | scaleX  | 2.0000000000
   1 | (0, 0) | skewY   | 0.0000000000
   1 | (0, 0) | skewX   | 0.0000000000
   1 | (0, 0) | scaleY  | 3.0000000000
   1 | (0, 0) | originX | 0.5000000000
   1 | (0, 0) | originY | 0.5000000000
   2 | (0, 0) | scaleX  | 0.0500000000
   2 | (0, 0) | skewY   | 0.0000000000
   2 | (0, 0) | skewX   | 0.0000000000
   2 | (0, 0) | scaleY  | -0.0500000000
   2 | (0, 0) | originX | 3427927.7500000000
   2 | (0, 0) | originY | 5793244.0000000000
(12 rows)
```

Se även

ST_GeoReference, ST_SkewY, ST_SetSkew

11.4.16 ST_SkewY

ST_SkewY — Returnerar georeferensens Y skew (eller rotationsparameter).

Synopsis

float8 **ST_SkewY**(raster rast);

Description

Returnerar georeferensens Y-skevhet (eller rotationsparameter). Se World File för mer information.

Exempel

Code

```
WITH refs AS (
    SELECT
        rid,
        format('(%s, %s)', ST_SkewX(rast), ST_SkewY(rast)) AS skew,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
)
SELECT
    rid,
    skew,
    term,
    georef[idx] AS value
FROM refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY rid, idx;
```

Output

```
rid |  skew  |  term   |       value
----+--------+---------+--------------------
   1 | (0, 0) | scaleX  | 2.0000000000
   1 | (0, 0) | skewY   | 0.0000000000
   1 | (0, 0) | skewX   | 0.0000000000
   1 | (0, 0) | scaleY  | 3.0000000000
   1 | (0, 0) | originX | 0.5000000000
   1 | (0, 0) | originY | 0.5000000000
   2 | (0, 0) | scaleX  | 0.0500000000
   2 | (0, 0) | skewY   | 0.0000000000
   2 | (0, 0) | skewX   | 0.0000000000
   2 | (0, 0) | scaleY  | -0.0500000000
   2 | (0, 0) | originX | 3427927.7500000000
   2 | (0, 0) | originY | 5793244.0000000000
(12 rows)
```

Se även

ST_GeoReference, ST_SkewX, ST_SetSkew

11.4.17 ST_SRID

ST_SRID — Returnerar den spatiala referensidentifieraren för rastret enligt definitionen i tabellen spatial_ref_sys.

Synopsis

integer **ST_SRID**(raster rast);

Description

Returnerar den spatiala referensidentifieraren för rasterobjektet enligt definitionen i tabellen spatial_ref_sys.

Notera
Non-georeferenced rasters and geometries use SRID 0.

Exempel

Code

```
SELECT ST_SRID(rast) As srid
FROM dummy_rast WHERE rid=1;
```

Output

```
0
```

Se även

Avsnitt 4.5, ST_SRID

11.4.18 ST_Summary

ST_Summary — Returnerar en textsammanfattning av innehållet i rastret.

Synopsis

text **ST_Summary**(raster rast);

Description

Returnerar en textsammanfattning av innehållet i rastret.
Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT ST_Summary(ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1,  ↩
    0, 0, 0),
                1, '8BUI', 1, 0
            ),
            2, '32BF', 0, -9999
        ),
        3, '16BSI', 0, NULL
    )
);
```

Output

```
Raster of 10x10 pixels has 3 bands and extent of BOX(0 -10,10 0)+
     band 1 of pixtype 8BUI is in-db with NODATA value of 0      +
     band 2 of pixtype 32BF is in-db with NODATA value of -9999  +
     band 3 of pixtype 16BSI is in-db with no NODATA value
(1 row)
```

Se även

ST_MetaData, ST_BandMetaData, ST_Summary ST_Extent

11.4.19 ST_UpperLeftX

ST_UpperLeftX — Returnerar den övre vänstra X-koordinaten för rastret i projicerad spatial ref.

Synopsis

float8 **ST_UpperLeftX**(raster rast);

Description

Returnerar den övre vänstra X-koordinaten för rastret i projicerad spatial ref.

Exempel

Code

```
SELECt rid, ST_UpperLeftX(rast) As ulx
FROM dummy_rast;
```

Output

```
rid |    ulx
-----+-----------
   1 |        0.5
   2 | 3427927.75
```

Se även

ST_UpperLeftY, ST_GeoReference, Box3D

11.4.20 ST_UpperLeftY

ST_UpperLeftY — Returnerar den övre vänstra Y-koordinaten för rastret i projicerad spatial ref.

Synopsis

float8 **ST_UpperLeftY**(raster rast);

Description

Returnerar den övre vänstra Y-koordinaten för rastret i projicerad spatial ref.

Exempel

Code

```
SELECT rid, ST_UpperLeftY(rast) As uly
FROM dummy_rast;
```

Output

```
rid |   uly
-----+---------
   1 |     0.5
   2 | 5793244
```

Se även

ST_UpperLeftX, ST_GeoReference, Box3D

11.4.21 ST_Width

ST_Width — Returnerar rastrets bredd i pixlar.

Synopsis

integer **ST_Width**(raster rast);

Description

Returnerar rastrets bredd i pixlar.

Exempel

Code

```
SELECT ST_Width(rast) As rastwidth
FROM dummy_rast WHERE rid=1;
```

Output

```
10
```

Se även

ST_Height

11.4.22 ST_WorldToRasterCoord

ST_WorldToRasterCoord — Returnerar det övre vänstra hörnet som kolumn och rad givet geometriskt X och Y (longitud och latitud) eller en punktgeometri uttryckt i rastrets spatiala referenskoordinatsystem.

Synopsis

record **ST_WorldToRasterCoord**(raster rast, geometry pt);
record **ST_WorldToRasterCoord**(raster rast, double precision longitude, double precision latitude);

Description

Returnerar det övre vänstra hörnet som kolumn och rad givet geometriskt X och Y (longitud och latitud) eller en punktgeometri. Denna funktion fungerar oavsett om de geometriska X och Y eller punktgeometrin ligger utanför rastrets utsträckning eller inte. Geometriska X och Y måste uttryckas i rastrets spatiala referenskoordinatsystem.

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT
    rid,
    (ST_WorldToRasterCoord(rast, 3427927.8, 20.5)).*,
    (ST_WorldToRasterCoord(rast, ST_GeomFromText('POINT(3427927.8 20.5)', ST_SRID(rast)))) ↩
    .*
FROM dummy_rast;
```

Output

```
rid | columnx |   rowy    | columnx |   rowy
----+---------+-----------+---------+----------
  1 | 1713964 |         7 | 1713964 |         7
  2 |       2 | 115864471 |       2 | 115864471
```

Se även

ST_WorldToRasterCoordX, ST_WorldToRasterCoordY, ST_RasterToWorldCoordX, ST_RasterToWorldCoordY
ST_SRID

11.4.23 ST_WorldToRasterCoordX

ST_WorldToRasterCoordX — Returnerar kolumnen i rastret för punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw) som representeras i rastrets världsspatiala referenssystem.

Synopsis

integer **ST_WorldToRasterCoordX**(raster rast, geometry pt);
integer **ST_WorldToRasterCoordX**(raster rast, double precision xw);
integer **ST_WorldToRasterCoordX**(raster rast, double precision xw, double precision yw);

Description

Returnerar kolumnen i rastret med punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw). En punkt, eller (både xw- och yw-världskoordinater krävs om ett raster är skevt). Om ett raster inte är skevt räcker det med xw. Världskoordinaterna är i rastrets spatiala referenskoordinatsystem.

Ändrad: 2.1.0 I tidigare versioner kallades detta ST_World2RasterCoordX

Exempel

Code

```
SELECT rid, ST_WorldToRasterCoordX(rast, 3427927.8) As xcoord,
        ST_WorldToRasterCoordX(rast, 3427927.8, 20.5) As xcoord_xwyw,
        ST_WorldToRasterCoordX(rast, ST_GeomFromText('POINT(3427927.8 20.5)', ST_SRID(rast) ↩
    )) As ptxcoord
FROM dummy_rast;
```

Output

```
rid | xcoord  |  xcoord_xwyw   | ptxcoord
-----+---------+---------+----------
   1 | 1713964 | 1713964 |  1713964
   2 |       1 |       1 |        1
```

Se även

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

11.4.24 ST_WorldToRasterCoordY

ST_WorldToRasterCoordY — Returnerar raden i rastret för punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw) som representeras i rastrets världsspatiala referenssystem.

Synopsis

integer **ST_WorldToRasterCoordY**(raster rast, geometry pt);
integer **ST_WorldToRasterCoordY**(raster rast, double precision xw);
integer **ST_WorldToRasterCoordY**(raster rast, double precision xw, double precision yw);

Description

Returnerar raden i rastret för punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw). En punkt, eller (både xw- och yw-världskoordinater krävs om ett raster är skevt). Om ett raster inte är skevt räcker det med xw. Världskoordinaterna är i rastrets spatiala referenskoordinatsystem.

Ändrad: 2.1.0 I tidigare versioner kallades detta ST_World2RasterCoordY

Exempel

Code

```
SELECT rid, ST_WorldToRasterCoordY(rast, 20.5) As ycoord,
        ST_WorldToRasterCoordY(rast, 3427927.8, 20.5) As ycoord_xwyw,
        ST_WorldToRasterCoordY(rast, ST_GeomFromText('POINT(3427927.8 20.5)', ST_SRID(rast) ↩
    )) As ptycoord
FROM dummy_rast;
```

Output

```
rid |  ycoord   | ycoord_xwyw | ptycoord
-----+-----------+-------------+-----------
   1 |         7 |           7 |         7
   2 | 115864471 |   115864471 | 115864471
```

Se även

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

11.5 Rasterband-accessorer

11.5.1 ST_BandMetaData

ST_BandMetaData — Returnerar grundläggande metadata för ett specifikt rasterband. bandnummer 1 antas om det inte specificeras.

Synopsis

(1) record **ST_BandMetaData**(raster rast, integer band=1);
(2) record **ST_BandMetaData**(raster rast, integer[] band);

Description

Returnerar grundläggande metadata om ett rasterband. Kolumner som returneras: pixeltyp, nodata-värde, isoutdb, sökväg, outdbbandnummer, filstorlek, filstämpel.

Notera
Om rastret inte innehåller några band utlöses ett felmeddelande.

Notera
Om bandet inte har något NODATA-värde är nodatavärdet NULL.

Notera
Om isoutdb är False är sökväg, outdbbandnum, filstorlek och filetimestamp NULL. Om outdb-åtkomst är inaktiverad kommer även filesize och filetimestamp att vara NULL.

Förbättrad: 2.5.0 för att inkludera *outdbbandnum*, *filstorlek* och *filetimestamp* för outdb-raster.

Exempel

Variant 1.

Code

```
SELECT
    rid,
    (foo.md).*
FROM (
    SELECT
        rid,
        ST_BandMetaData(rast, 1) AS md
    FROM dummy_rast
    WHERE rid=2
) As foo;
```

Output

```
rid | pixeltype | nodatavalue | isoutdb | path | outdbbandnum
-----+-----------+-------------+---------+------+--------------
   2 | 8BUI      |           0 | f       |      |
(1 row)
```

Variant 2.

Code

```
WITH foo AS (
    SELECT
        ST_AddBand(NULL::raster, '/home/pele/devel/geo/postgis-git/raster/test/regress/ ↩
    loader/Projected.tif', NULL::int[]) AS rast
)
SELECT
    bandnum,
    pixeltype,
    isoutdb,
    regexp_replace(path, '^.*/', '') AS file,
    outdbbandnum
FROM ST_BandMetaData((SELECT rast FROM foo),
    ARRAY[1, 3, 2]::int[]
);
```

Output

```
bandnum | pixeltype | isoutdb |     file      | outdbbandnum
---------+-----------+---------+---------------+--------------
       1 | 8BUI      | t       | Projected.tif |            1
       3 | 8BUI      | t       | Projected.tif |            3
       2 | 8BUI      | t       | Projected.tif |            2
```

Se även

ST_MetaData, ST_BandPixelType

11.5.2 ST_BandNoDataValue

ST_BandNoDataValue — Returnerar värdet i ett givet band som inte representerar några data. Om inget band finns antas siffran 1.

Synopsis

double precision **ST_BandNoDataValue**(raster rast, integer bandnum=1);

Description

Returnerar det värde som representerar ingen data för bandet

Exempel

Code

```
SELECT ST_BandNoDataValue(rast, 1) As bnval1,
    ST_BandNoDataValue(rast, 2) As bnval2, ST_BandNoDataValue(rast, 3) As bnval3
FROM dummy_rast
WHERE rid = 2;
```

Output

```
bnval1 | bnval2 | bnval3
--------+--------+--------
      0 |      0 |      0
```

Se även

ST_NumBands

11.5.3 ST_BandIsNoData

ST_BandIsNoData — Returnerar true om bandet är fyllt med endast nodata-värden.

Synopsis

boolean **ST_BandIsNoData**(raster rast, integer band, boolean forceChecking=false);
boolean **ST_BandIsNoData**(raster rast, boolean forceChecking=false);

Description

Returnerar true om bandet är fyllt med endast nodata-värden. Band 1 antas om det inte anges. Om det sista argumentet är TRUE kontrolleras hela bandet pixel för pixel. I annat fall returnerar funktionen helt enkelt värdet på isnodata-flaggan för bandet. Standardvärdet för denna parameter är FALSE, om den inte anges.

Tillgänglighet: 2.0.0

Notera
If the flag is dirty (this is, the result is different using TRUE as last parameter and not using it) you should update the raster to set this flag to true, by using ST_SetBandIsNoData(), or ST_SetBandNoDataValue() with TRUE as last argument. See ST_SetBandIsNoData.

Exempel

This example creates a dummy table with one raster column. The raster has two bands and one pixel per band. In the first band, the NODATA value and pixel value are both 3. In the second band, the NODATA value is 13 and the pixel value is 4.

Code

```
create table dummy_rast (rid integer, rast raster);

insert into dummy_rast values(
1,
(
'01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0200' -- nBands (uint16 0)
||
'17263529ED684A3F' -- scaleX (float64 0.000805965234044584)
||
'F9253529ED684ABF' -- scaleY (float64 -0.00080596523404458)
||
'1C9F33CE69E352C0' -- ipX (float64 -75.5533328537098)
||
'718F0E9A27A44840' -- ipY (float64 49.2824585505576)
||
'ED50EB853EC32B3F' -- skewX (float64 0.000211812383858707)
||
'7550EB853EC32B3F' -- skewY (float64 0.000211812383858704)
||
'E6100000' -- SRID (int32 4326)
||
'0100' -- width (uint16 1)
||
'0100' -- height (uint16 1)
||
'6' -- hasnodatavalue and isnodata value set to true.
||
'2' -- first band type (4BUI)
||
'03' -- novalue==3
||
'03' -- pixel(0,0)==3 (same that nodata)
||
'0' -- hasnodatavalue set to false
||
'5' -- second band type (16BSI)
||
'0D00' -- novalue==13
||
'0400' -- pixel(0,0)==4
)::raster
);

select ST_BandIsNoData(rast, 1) from dummy_rast where rid = 1; -- Expected true
select ST_BandIsNoData(rast, 2) from dummy_rast where rid = 1; -- Expected false
```

Se även

ST_BandNoDataValue, ST_NumBands, ST_SetBandNoDataValue, ST_SetBandIsNoData

11.5.4 ST_BandPath

ST_BandPath — Returnerar systemfilens sökväg till ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.

Synopsis

text **ST_BandPath**(raster rast, integer bandnum=1);

Description

Returnerar systemfilens sökväg till ett band. Kastar ett fel om det anropas med ett in db-band.

Exempel

Code

```
WITH foo AS (
    SELECT ST_AddBand(
        NULL::raster,
        '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif',
        NULL::int[]
    ) AS rast
)
SELECT bandnum, ST_BandPath(rast, bandnum) AS path
FROM foo
CROSS JOIN generate_series(1, 3) AS bandnum;
```

Output

```
bandnum |                                 path
---------+-------------------------------------------------------------------------
       1 | /home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
       2 | /home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
       3 | /home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
```

Se även

ST_BandMetaData, ST_SetBandPath, ST_SetBandIndex

11.5.5 ST_BandFileSize

ST_BandFileSize — Returnerar filstorleken för ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.

Synopsis

bigint **ST_BandFileSize**(raster rast, integer bandnum=1);

Description

Returnerar filstorleken för ett band som lagras i filsystemet. Kastar ett fel om det anropas med ett in db-band, eller om outdb-åtkomst inte är aktiverad.

Denna funktion används vanligtvis tillsammans med ST_BandPath() och ST_BandFileTimestamp() så att en klient kan avgöra om filnamnet på ett outdb-raster som den ser är detsamma som det som servern ser.

Tillgänglighet: 2.5.0

Exempel

Code

```
SELECT ST_BandFileSize(rast, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
240574
```

11.5.6 ST_BandFileTimestamp

ST_BandFileTimestamp — Returnerar filens tidsstämpel för ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.

Synopsis

bigint **ST_BandFileTimestamp**(raster rast, integer bandnum=1);

Description

Returnerar filens tidsstämpel (antal sekunder sedan 1 januari 1970 00:00:00 UTC) för ett band som lagras i filsystemet. Kastar ett fel om det anropas med ett in db-band, eller om outdb-åtkomst inte är aktiverad.

Denna funktion används vanligtvis tillsammans med ST_BandPath() och ST_BandFileSize() så att en klient kan avgöra om filnamnet på ett outdb-raster som den ser är detsamma som det som servern ser.

Tillgänglighet: 2.5.0

Exempel

Code

```
SELECT ST_BandFileTimestamp(rast, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
1521807257
```

11.5.7 ST_BandPixelType

ST_BandPixelType — Returnerar pixeltyp för givet band. Om inget bandnummer anges antas 1.

Synopsis

text **ST_BandPixelType**(raster rast, integer bandnum=1);

Description

Returnerar namn som beskriver datatyp och storlek på värden som lagras i varje cell i ett givet band. Det finns 11 olika pixeltyper. Pixeltyper som stöds är följande:

- 1BB - 1-bitars boolean
- 2BUI - 2-bitars osignerat heltal
- 4BUI - 4-bitars osignerat heltal
- 8BSI - 8-bitars signerat heltal
- 8BUI - 8-bitars osignerat heltal
- 16BSI - 16-bitars signerat heltal
- 16BUI - 16-bitars osignerat heltal
- 32BSI - 32-bitars signerat heltal
- 32BUI - 32-bitars osignerat heltal
- 32BF - 32-bitars float
- 64BF - 64-bitars float

Exempel

Code

```
SELECT ST_BandPixelType(rast, 1) As btype1,
    ST_BandPixelType(rast, 2) As btype2, ST_BandPixelType(rast, 3) As btype3
FROM dummy_rast
WHERE rid = 2;
```

Output

```
btype1 | btype2 | btype3
--------+--------+--------
 8BUI   | 8BUI   | 8BUI
```

Se även

ST_NumBands

11.5.8 ST_MinPossibleValue

ST_MinPossibleValue — Returnerar det lägsta värde som denna pixeltyp kan lagra.

Synopsis

integer **ST_MinPossibleValue**(text pixeltype);

Description

Returnerar det lägsta värde som denna pixeltyp kan lagra.

Exempel

Code

```
SELECT ST_MinPossibleValue('16BSI');
```

Output

```
-32768
```

Code

```
SELECT ST_MinPossibleValue('8BUI');
```

Output

```
0
```

Se även

ST_BandPixelType

11.5.9 ST_HasNoBand

ST_HasNoBand — Returnerar true om det inte finns något band med angivet bandnummer. Om inget bandnummer anges antas bandnummer 1.

Synopsis

boolean **ST_HasNoBand**(raster rast, integer bandnum=1);

Description

Returnerar true om det inte finns något band med angivet bandnummer. Om inget bandnummer anges antas bandnummer 1.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT rid, ST_HasNoBand(rast) As hb1, ST_HasNoBand(rast, 2) as hb2,
ST_HasNoBand(rast, 4) as hb4, ST_NumBands(rast) As numbands
FROM dummy_rast;
```

Output

```
rid | hb1 | hb2 | hb4 | numbands
-----+-----+-----+-----+----------
1 | t   | t   | t   |        0
2 | f   | f   | t   |        3
```

Se även

ST_NumBands

11.6 Raster Pixel-accessorer och Setters

11.6.1 ST_PixelAsPolygon

ST_PixelAsPolygon — Returnerar den polygongeometri som avgränsar pixeln för en viss rad och kolumn.

Synopsis

geometry **ST_PixelAsPolygon**(raster rast, integer columnx, integer rowy);

Description

Returnerar den polygongeometri som avgränsar pixeln för en viss rad och kolumn.

Tillgänglighet: 2.0.0

Exempel

This example returns a raster pixel polygon.

Code

```
SELECT i, j, ST_PixelAsPolygon(foo.rast, i, j) As b1pgeom
FROM dummy_rast As foo
    CROSS JOIN generate_series(1, 2) As i
    CROSS JOIN generate_series(1, 1) As j
WHERE rid=2;
```

Output

```
i | j |                                                    b1pgeom
---+---+-------------------------------------------------------------------------------
 1 | 1 | POLYGON((3427927.75 5793244,3427927.8 5793244,3427927.8 5793243.95,...
 2 | 1 | POLYGON((3427927.8 5793244,3427927.85 5793244,3427927.85 5793243.95, ..
```

Se även

ST_DumpAsPolygons, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid, ST_Pixe
ST_Intersection, ST_AsText

11.6.2 ST_PixelAsPolygons

ST_PixelAsPolygons — Returnerar den polygongeometri som avgränsar varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel.

Synopsis

setof record **ST_PixelAsPolygons**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

Description

Returnerar den polygongeometri som avgränsar varje pixel i ett rasterband tillsammans med värdet (dubbel precision), X- och Y-rasterkoordinaterna (heltal) för varje pixel.

Format för returpost: *geom* geometry, *val* dubbel precision, *x* heltal, *y* heltal.

Notera
When *exclude_nodata_value* is TRUE, only pixels whose values are not NODATA are returned. Unlike ST_DumpAsPolygons, this function returns one polygon per pixel rather than merging adjacent pixels with the same value.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 exclude_nodata_value valfritt argument lades till.

Ändrad: 2.1.1 Ändrat beteende för exclude_nodata_value.

Exempel

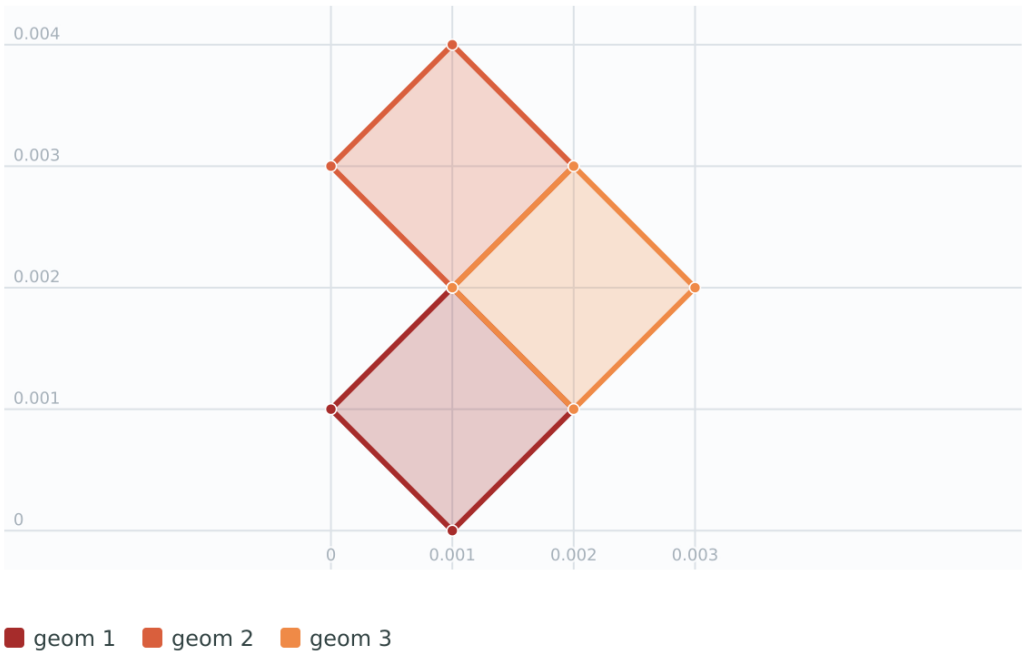
This example returns a raster pixel polygon.

Code

```
WITH raster AS (
    SELECT ST_SetValue(
        ST_SetValue(
            ST_AddBand(
                ST_MakeEmptyRaster(
                    2, 2, 0, 0, 0.001, -0.001, 0.001, 0.001, 4269
                ),
                1, '8BUI', 1, 0
            ),
            2, 2, 10
        ),
        1, 1, NULL
    ) AS rast
), pixels AS (
    SELECT (ST_PixelAsPolygons(rast)).*
    FROM raster
```

```
)
SELECT x, y, val, geom AS geom
FROM pixels
ORDER BY x, y;
```

Figure



Se även

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid, ST_Pixel
ST_AsText

11.6.3 ST_PixelAsPoint

ST_PixelAsPoint — Returnerar en punktgeometri för pixelns övre vänstra hörn.

Synopsis

geometry **ST_PixelAsPoint**(raster rast, integer columnx, integer rowy);

Description

Returnerar en punktgeometri för pixelns övre vänstra hörn.

Tillgänglighet: 2.1.0

Exempel

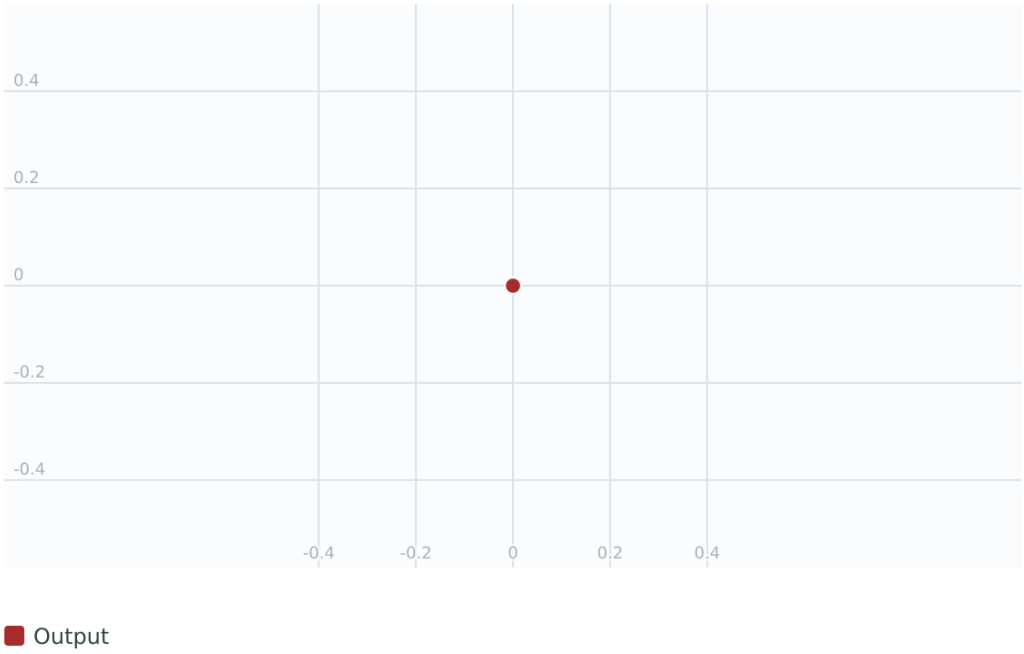
Code

```
SELECT ST_PixelAsPoint(rast, 1, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
POINT(0.5 0.5)
```

Figure



Se även

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids

11.6.4 ST_PixelAsPoints

ST_PixelAsPoints — Returnerar en punktgeometri för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrins koordinater är pixelns övre vänstra hörn.

Synopsis

setof record **ST_PixelAsPoints**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

Description

Returnerar en punktgeometri för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordina
för varje pixel. Punktgeometrins koordinater är pixelns övre vänstra hörn.

Format för returpost: *geom* geometry, *val* dubbel precision, *x* heltal, *y* heltal.

Notera
När *exclude_nodata_value* = TRUE returneras endast de pixlar vars värden inte är NODATA som punkter.

Tillgänglighet: 2.1.0

Ändrad: 2.1.1 Ändrat beteende för exclude_nodata_value.

Exempel

This example returns point features together with a derived marker size that a client can use for value-scaled point rendering.

Code

```
SELECT
    x,
    y,
    val,
    round((val / 32.0)::numeric, 2) AS marker_scale,geom AS geom
FROM (
    SELECT (ST_PixelAsPoints(rast, 1)).*
    FROM dummy_rast
    WHERE rid = 2
) AS foo
ORDER BY y, x;
```

Output

```
x | y | val | marker_scale |          geom
---+---+-----+--------------+----------------------------
 1 | 1 | 253 |         7.91 | POINT(3427927.75 5793244)
 2 | 1 | 254 |         7.94 | POINT(3427927.8 5793244)
 3 | 1 | 253 |         7.91 | POINT(3427927.85 5793244)
 4 | 1 | 254 |         7.94 | POINT(3427927.9 5793244)
 5 | 1 | 254 |         7.94 | POINT(3427927.95 5793244)
 1 | 2 | 253 |         7.91 | POINT(3427927.75 5793243.95)
 2 | 2 | 254 |         7.94 | POINT(3427927.8 5793243.95)
 3 | 2 | 254 |         7.94 | POINT(3427927.85 5793243.95)
 4 | 2 | 253 |         7.91 | POINT(3427927.9 5793243.95)
 5 | 2 | 249 |         7.78 | POINT(3427927.95 5793243.95)
 1 | 3 | 250 |         7.81 | POINT(3427927.75 5793243.9)
 2 | 3 | 254 |         7.94 | POINT(3427927.8 5793243.9)
 3 | 3 | 254 |         7.94 | POINT(3427927.85 5793243.9)
 4 | 3 | 252 |         7.88 | POINT(3427927.9 5793243.9)
 5 | 3 | 249 |         7.78 | POINT(3427927.95 5793243.9)
 1 | 4 | 251 |         7.84 | POINT(3427927.75 5793243.85)
 2 | 4 | 253 |         7.91 | POINT(3427927.8 5793243.85)
 3 | 4 | 254 |         7.94 | POINT(3427927.85 5793243.85)
 4 | 4 | 254 |         7.94 | POINT(3427927.9 5793243.85)
 5 | 4 | 253 |         7.91 | POINT(3427927.95 5793243.85)
 1 | 5 | 252 |         7.88 | POINT(3427927.75 5793243.8)
 2 | 5 | 250 |         7.81 | POINT(3427927.8 5793243.8)
 3 | 5 | 254 |         7.94 | POINT(3427927.85 5793243.8)
 4 | 5 | 254 |         7.94 | POINT(3427927.9 5793243.8)
 5 | 5 | 254 |         7.94 | POINT(3427927.95 5793243.8)
```

Figure

Se även

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsCentroid, ST_PixelAsCentroids

11.6.5 ST_PixelAsCentroid

ST_PixelAsCentroid — Returnerar centroiden (punktgeometri) för det område som representeras av en pixel.

Synopsis

geometry **ST_PixelAsCentroid**(raster rast, integer x, integer y);

Description

Returnerar centroiden (punktgeometri) för det område som representeras av en pixel.

Förbättrad: 3.2.0 Snabbare nu implementerad i C.

Tillgänglighet: 2.1.0

Exempel

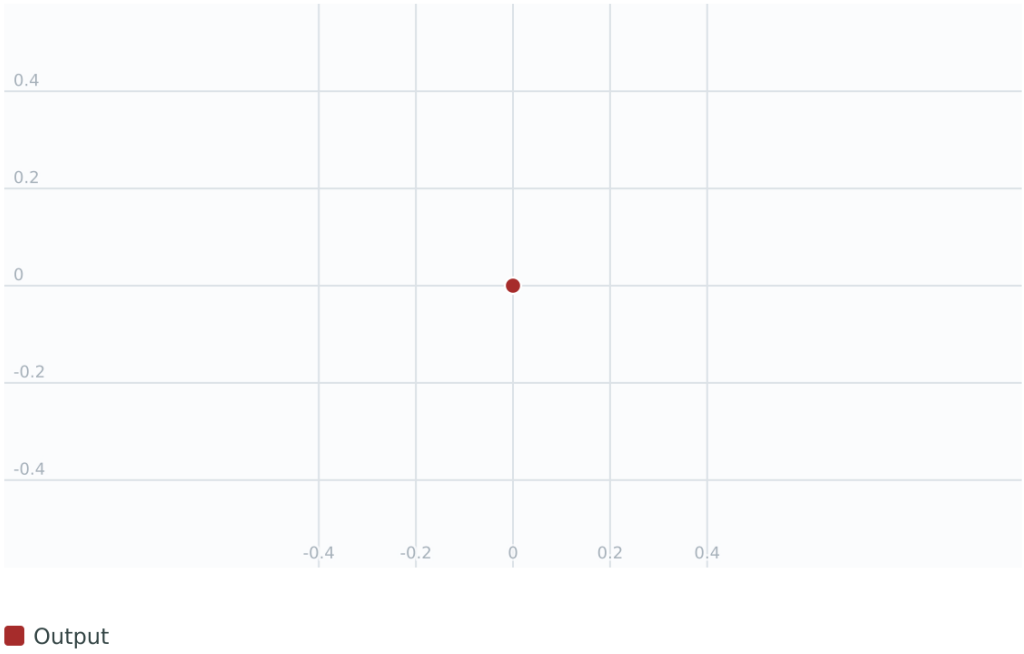
Code

```
SELECT ST_PixelAsCentroid(rast, 1, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
POINT(1.5 2)
```

Figure



Se även

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_Pixel

11.6.6 ST_PixelAsCentroids

ST_PixelAsCentroids — Returnerar centroiden (punktgeometri) för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrin är centroiden för det område som representeras av en pixel.

Synopsis

setof record **ST_PixelAsCentroids**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

Description

Returnerar centroiden (punktgeometri) för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrin är centroiden för det område som representeras av en pixel.

Format för returpost: *geom* geometry, *val* dubbel precision, *x* heltal, *y* heltal.

Notera
När *exclude_nodata_value* = TRUE returneras endast de pixlar vars värden inte är NODATA som punkter.

Förbättrad: 3.2.0 Snabbare nu implementerad i C.

Ändrad: 2.1.1 Ändrat beteende för exclude_nodata_value.

Tillgänglighet: 2.1.0

Exempel

This example uses LATERAL syntax.

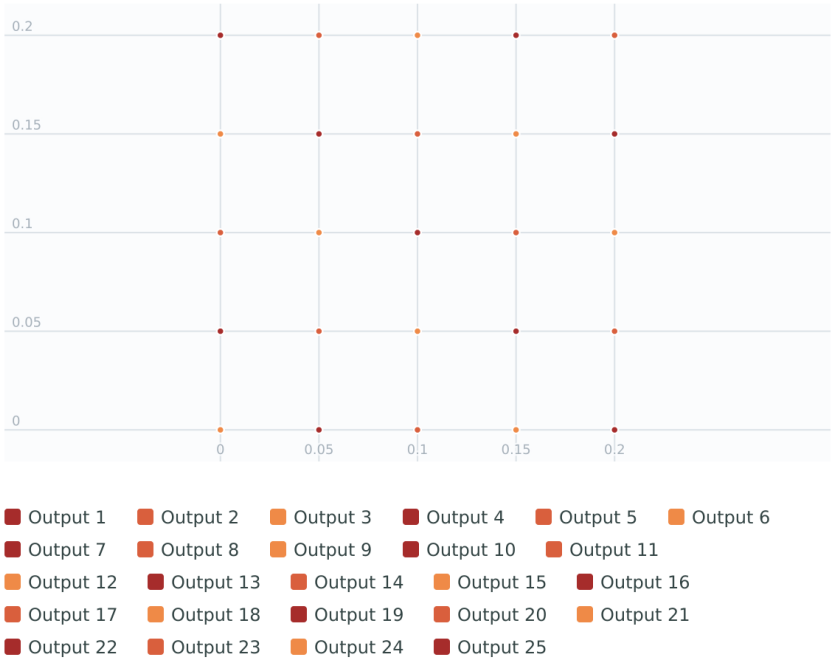
Code

```
SELECT x, y, val,geom
    FROM (SELECT dp.* FROM dummy_rast, LATERAL ST_PixelAsCentroids(rast, 1) AS dp WHERE rid ↩
     = 2) foo;
```

Output

```
x | y | val |           st_astext
---+---+-----+-------------------------------
 1 | 1 | 253 | POINT(3427927.775 5793243.975)
 2 | 1 | 254 | POINT(3427927.825 5793243.975)
 3 | 1 | 253 | POINT(3427927.875 5793243.975)
 4 | 1 | 254 | POINT(3427927.925 5793243.975)
 5 | 1 | 254 | POINT(3427927.975 5793243.975)
 1 | 2 | 253 | POINT(3427927.775 5793243.925)
 2 | 2 | 254 | POINT(3427927.825 5793243.925)
 3 | 2 | 254 | POINT(3427927.875 5793243.925)
 4 | 2 | 253 | POINT(3427927.925 5793243.925)
 5 | 2 | 249 | POINT(3427927.975 5793243.925)
 1 | 3 | 250 | POINT(3427927.775 5793243.875)
 2 | 3 | 254 | POINT(3427927.825 5793243.875)
 3 | 3 | 254 | POINT(3427927.875 5793243.875)
 4 | 3 | 252 | POINT(3427927.925 5793243.875)
 5 | 3 | 249 | POINT(3427927.975 5793243.875)
 1 | 4 | 251 | POINT(3427927.775 5793243.825)
 2 | 4 | 253 | POINT(3427927.825 5793243.825)
 3 | 4 | 254 | POINT(3427927.875 5793243.825)
 4 | 4 | 254 | POINT(3427927.925 5793243.825)
 5 | 4 | 253 | POINT(3427927.975 5793243.825)
 1 | 5 | 252 | POINT(3427927.775 5793243.775)
 2 | 5 | 250 | POINT(3427927.825 5793243.775)
 3 | 5 | 254 | POINT(3427927.875 5793243.775)
 4 | 5 | 254 | POINT(3427927.925 5793243.775)
 5 | 5 | 254 | POINT(3427927.975 5793243.775)
```

Figure



Se även

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_Pixel

11.6.7 ST_Value

ST_Value — Returnerar värdet för ett visst band i en viss kolumnx, radpixel eller vid en viss geometrisk punkt. Bandnummer börjar på 1 och antas vara 1 om det inte anges. Om `exclude_nodata_value` är satt till false, anses alla pixlar inklusive `nodata-pixlar` korsa varandra och returnerar värdet. Om `värdet exclude_nodata_value` inte anges läses det från rastrets metadata.

Synopsis

double precision **ST_Value**(raster rast, geometry pt, boolean exclude_nodata_value=true);
double precision **ST_Value**(raster rast, integer band, geometry pt, boolean exclude_nodata_value=true, text resample='nearest');
double precision **ST_Value**(raster rast, integer x, integer y, boolean exclude_nodata_value=true);
double precision **ST_Value**(raster rast, integer band, integer x, integer y, boolean exclude_nodata_value=tru

Description

Returnerar värdet för ett givet band i en given kolumnx, radpixel eller vid en given geometripunkt. Bandnummer börjar på 1 och band antas vara 1 om det inte specificeras.

Om `värdet exclude_nodata_value` är satt till true beaktas endast pixlar som inte är `nodata`. Om `värdet exclude_nodata_value` är satt till false beaktas alla pixlar.

The allowed values of the `resample` parameter are neareståhich performs the default nearest-neighbor resampling, bilinearåhich performs a bilinear interpolation to estimate the value between pixel centers, and the nearest-neighbor boundary options nearest-UL", nearest-UR", nearest-LL", and nearest-LR". Boundary options choose which pixel is returned when the point lies on a horizontal or vertical pixel boundary, including corner intersections.

Enhanced: 3.7.0 resample accepts nearest-neighbor boundary options nearest-UL", nearest-UR", nnearest-LL", and nearest-LR".

Förbättrad: 3.2.0 resample valfritt argument lades till.

Förbättrad: 2.0.0 exclude_nodata_value valfritt argument lades till.

Exempel

Get raster values at particular postgis geometry points. The srid of your geometry should be same as for your raster.

Code

```
SELECT rid, ST_Value(rast, foo.pt_geom) As b1pval, ST_Value(rast, 2, foo.pt_geom) As b2pval
FROM dummy_rast CROSS JOIN (SELECT ST_SetSRID(ST_Point(3427927.77, 5793243.76), 0) As  ↩
    pt_geom) As foo
WHERE rid=2;
```

Output

```
rid | b1pval | b2pval
-----+--------+--------
   2 |    252 |     79
```

General fictitious example using a real table.

Code

```
SELECT rid, ST_Value(rast, 3, sometable.geom) As b3pval
FROM sometable
WHERE ST_Intersects(rast, sometable.geom);
```

Code

```
SELECT rid, ST_Value(rast, 1, 1, 1) As b1pval,
    ST_Value(rast, 2, 1, 1) As b2pval, ST_Value(rast, 3, 1, 1) As b3pval
FROM dummy_rast
WHERE rid=2;
```

Output

```
rid | b1pval | b2pval | b3pval
-----+--------+--------+--------
   2 |    253 |     78 |     70
```

Get all values in bands 1,2,3 of each pixel.

Code

```
SELECT x, y, ST_Value(rast, 1, x, y) As b1val,
    ST_Value(rast, 2, x, y) As b2val, ST_Value(rast, 3, x, y) As b3val
FROM dummy_rast CROSS JOIN
generate_series(1, 1000) As x CROSS JOIN generate_series(1, 1000) As y
WHERE rid =  2 AND x <= ST_Width(rast) AND y <= ST_Height(rast);
```

Output

```
 x | y | b1val | b2val | b3val
---+---+-------+-------+-------
 1 | 1 |   253 |    78 |    70
 1 | 2 |   253 |    96 |    80
 1 | 3 |   250 |    99 |    90
 1 | 4 |   251 |    89 |    77
 1 | 5 |   252 |    79 |    62
 2 | 1 |   254 |    98 |    86
 2 | 2 |   254 |   118 |   108
 :
 :
```

Get all values in bands 1,2,3 of each pixel same as above but returning the upper left point point of each pixel.

Code

```
SELECT ST_SetSRID(ST_Point(ST_UpperLeftX(rast) + ST_ScaleX(rast)*x,
        ST_UpperLeftY(rast) + ST_ScaleY(rast)*y),
        ST_SRID(rast)) As uplpt,
    ST_Value(rast, 1, x, y) As b1val,
    ST_Value(rast, 2, x, y) As b2val, ST_Value(rast, 3, x, y) As b3val
FROM dummy_rast CROSS JOIN
generate_series(1,1000) As x CROSS JOIN generate_series(1,1000) As y
WHERE rid =  2 AND x <= ST_Width(rast) AND y <= ST_Height(rast);
```

Output

```
            uplpt            | b1val | b2val | b3val
-----------------------------+-------+-------+-------
 POINT(3427929.25 5793245.5) |   253 |    78 |    70
 POINT(3427929.25 5793247)   |   253 |    96 |    80
 POINT(3427929.25 5793248.5) |   250 |    99 |    90
:
```

Get a polygon formed by union of all pixels that fall in a particular value range and intersect a particular polygon.

Code

```
SELECT ST_Union(pixpolyg) As shadow
FROM (SELECT ST_Translate(ST_MakeEnvelope(ST_UpperLeftX(rast), ST_UpperLeftY(rast),
            ST_UpperLeftX(rast) + ST_ScaleX(rast),
            ST_UpperLeftY(rast) + ST_ScaleY(rast), 0
            ), ST_ScaleX(rast)*x, ST_ScaleY(rast)*y
        ) As pixpolyg, ST_Value(rast, 2, x, y) As b2val
    FROM dummy_rast CROSS JOIN
generate_series(1,1000) As x CROSS JOIN generate_series(1,1000) As y
WHERE rid =  2
    AND x <= ST_Width(rast) AND y <= ST_Height(rast)) As foo
WHERE
    ST_Intersects(
        pixpolyg,
        ST_GeomFromText('POLYGON((3427928 5793244,3427927.75 5793243.75,3427928 ↩
    5793243.75,3427928 5793244))',0)
        ) AND b2val != 254;
```

Output

```
       shadow
-------------------------------------------------------------------------------
 MULTIPOLYGON(((3427928 5793243.9,3427928 5793243.85,3427927.95 5793243.85,3427927.95 5793243.9,
 3427927.95 5793243.95,3427928 5793243.95,3427928.05 5793243.95,3427928.05 5793243.9,3427928 5793243.9)),((3427927.95 ↩
    5793243.9,3427927.95 579324
```

```
3.85,3427927.9 5793243.85,3427927.85 5793243.85,3427927.85 5793243.9,3427927.9 5793243.9,3427927.9 5793243.95,
3427927.95 5793243.95,3427927.95 5793243.9)),((3427927.85 5793243.75,3427927.85 5793243.7,3427927.8 ↩
    5793243.7,3427927.8 5793243.75,
3427927.8 5793243.8,3427927.8 5793243.85,3427927.85 5793243.85,3427927.85 5793243.8,3427927.85 5793243.75)),
((3427928.05 5793243.75,3427928.05 5793243.7,3427928 5793243.7,3427927.95 5793243.7,3427927.95 5793243.75,3427927.95 ↩
    5793243.8,3427
927.95 5793243.85,3427928 5793243.85,3427928 5793243.8,3427928.05 5793243.8,
3427928.05 5793243.75)),((3427927.95 5793243.75,3427927.95 5793243.7,3427927.9 5793243.7,3427927.85 5793243.7,
3427927.85 5793243.75,3427927.85 5793243.8,3427927.85 5793243.85,3427927.9 5793243.85,
3427927.95 5793243.85,3427927.95 5793243.8,3427927.95 5793243.75)))
```

Checking all the pixels of a large raster tile can take a long time. You can dramatically improve speed at some lose of precision by orders of magnitude. By sampling pixels using the step optional parameter of generate_series. This next example does the same as previous but by checking 1 for every 4 (2x2) pixels and putting in the last checked. Putting in the checked pixel as the value for subsequent 4.

Code

```
SELECT ST_Union(pixpolyg) As shadow
FROM (SELECT ST_Translate(ST_MakeEnvelope(ST_UpperLeftX(rast), ST_UpperLeftY(rast),
            ST_UpperLeftX(rast) + ST_ScaleX(rast)*2,
            ST_UpperLeftY(rast) + ST_ScaleY(rast)*2, 0
            ), ST_ScaleX(rast)*x, ST_ScaleY(rast)*y
        ) As pixpolyg, ST_Value(rast, 2, x, y) As b2val
    FROM dummy_rast CROSS JOIN
generate_series(1,1000,2) As x CROSS JOIN generate_series(1,1000,2) As y
WHERE rid =  2
    AND x <= ST_Width(rast)  AND y <= ST_Height(rast)  ) As foo
WHERE
    ST_Intersects(
        pixpolyg,
        ST_GeomFromText('POLYGON((3427928 5793244,3427927.75 5793243.75,3427928 ↩
    5793243.75,3427928 5793244))',0)
        ) AND b2val != 254;
```

Output

```
        shadow
-----------------------------------------------------------------------------------
 MULTIPOLYGON(((3427927.9 5793243.85,3427927.8 5793243.85,3427927.8 5793243.95,
 3427927.9 5793243.95,3427928 5793243.95,3427928.1 5793243.95,3427928.1 5793243.85,3427928 ↩
    5793243.85,3427927.9 5793243.85)),
 ((3427927.9 5793243.65,3427927.8 5793243.65,3427927.8 5793243.75,3427927.8 ↩
    5793243.85,3427927.9 5793243.85,
 3427928 5793243.85,3427928 5793243.75,3427928.1 5793243.75,3427928.1 5793243.65,3427928 ↩
    5793243.65,3427927.9 5793243.65)))
```

Se även

ST_DumpValues, ST_SetValue, ST_DumpAsPolygons, ST_NumBands, ST_PixelAsPolygon, ST_ScaleX, ST_ScaleY, ST_UpperLeftX, ST_UpperLeftY, ST_SRID, ST_AsText, ST_Point, ST_MakeEnvelope, ST_Intersec
ST_Intersection

11.6.8 ST_NearestValue

ST_NearestValue — Returnerar det närmaste `icke-NODATA-värdet` för ett givet bands pixel som anges av en kolumnx och rowy eller en geometrisk punkt uttryckt i samma spatiala referenskoordinatsystem som rastret.

Synopsis

double precision **ST_NearestValue**(raster rast, integer bandnum, geometry pt, boolean exclude_nodata_val
double precision **ST_NearestValue**(raster rast, geometry pt, boolean exclude_nodata_value=true);
double precision **ST_NearestValue**(raster rast, integer bandnum, integer columnx, integer rowy, boolean exclude_nodata_value=true);
double precision **ST_NearestValue**(raster rast, integer columnx, integer rowy, boolean exclude_nodata_valu

Description

Returnerar det närmaste `icke-NODATA-värdet` för ett givet band i en given kolumnx, rowy-pixel eller i en specifik geometrisk punkt. Om pixeln i kolumnx, rowy eller pixeln i den angivna geometriska punkten är `NODATA`, kommer funktionen att hitta den närmaste pixeln till pixeln i kolumnx, rowy eller den geometriska punkten vars värde inte är `NODATA`.

Bandnumren börjar på 1 och `bandnum` antas vara 1 om det inte anges. Om `exclude_nodata_value` är satt till false anses alla pixlar inklusive `nodata-pixlar` skära varandra och returnerar värdet. Om `värdet exclude_nodata_value` inte anges läses det från rastrets metadata.

Tillgänglighet: 2.1.0

Notera
ST_NearestValue är en drop-in-ersättning för ST_Value.

Exempel

This example reads a value from pixel 2x2.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
            1, '8BUI', 1, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 1, 1, 0],
            [1, 1, 0, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_Value(rast, 2, 2) AS value,
    ST_NearestValue(rast, 2, 2) AS nearestvalue
FROM source
```

Output

```
value | nearestvalue
------+--------------
    1 |            1
```

This example reads a NODATA value from pixel 2x3.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
            1, '8BUI', 1, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 1, 1, 0],
            [1, 1, 0, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_Value(rast, 2, 3) AS value,
    ST_NearestValue(rast, 2, 3) AS nearestvalue
FROM source
```

Output

```
value | nearestvalue
------+--------------
      |            1
```

Se även

ST_Neighborhood, ST_Value

11.6.9 ST_SetZ

ST_SetZ — Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopierade till Z-dimensionen med hjälp av den begärda resample-algoritmen.

Synopsis

geometry **ST_SetZ**(raster rast, geometry geom, text resample=nearest, integer band=1);

Description

Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopieras till Z-dimensionerna med hjälp av den begärda resample-algoritmen.

Parametern `resample` kan ställas in på nearestför att kopiera värdena från den cell som varje vertex faller inom, eller bilinearför att använda bilineär interpolation för att beräkna ett värde som även tar hänsyn till angränsande celler.

Tillgänglighet: 3.2.0

Exempel

2x2 test raster with values. 10 50. 40 20.

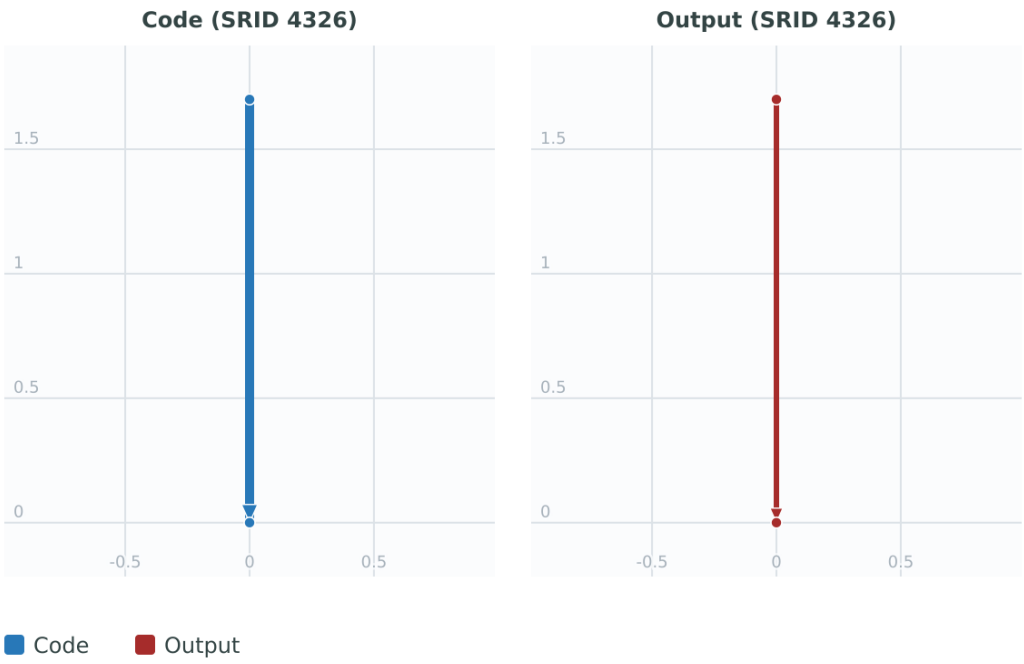
Code

```
WITH test_raster AS (
SELECT
ST_SetValues(
ST_AddBand(
ST_MakeEmptyRaster(
      width => 2,
      height => 2,
      upperleftx => 0,
      upperlefty => 2,
      scalex => 1.0,
      scaley => -1.0,
      skewx => 0,
      skewy => 0,
      srid => 4326),
    index => 1,
    pixeltype => '16BSI',
    initialvalue => 0,
    nodataval => -999
),
  1,
  1,
  1,
  newvalueset => ARRAY[ARRAY[10.0::float8, 50.0::float8], ARRAY[40.0::float8, 20.0::float8 ↩
    ]]) AS rast
)
SELECT
ST_SetZ(
    rast,
    band => 1,
    geom => 'SRID=4326;LINESTRING(1.0 1.9,1.0 0.2)'::geometry,
    resample => 'bilinear'
)
FROM test_raster
```

Output

```
LINESTRING Z (1 1.9 38,1 0.2 27)
```

Figure



Se även

ST_Value, ST_SetM

11.6.10 ST_SetM

ST_SetM — Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopierade till M-dimensionen med hjälp av den begärda resample-algoritmen.

Synopsis

geometry **ST_SetM**(raster rast, geometry geom, text resample=nearest, integer band=1);

Description

Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopierade till M-dimensionerna med hjälp av den begärda resample-algoritmen.

Parametern `resample` kan ställas in på nearestför att kopiera värdena från den cell som varje vertex faller inom, eller bilinearför att använda bilineär interpolation för att beräkna ett värde som även tar hänsyn till angränsande celler.

Tillgänglighet: 3.2.0

Exempel

2x2 test raster with values. 10 50. 40 20.

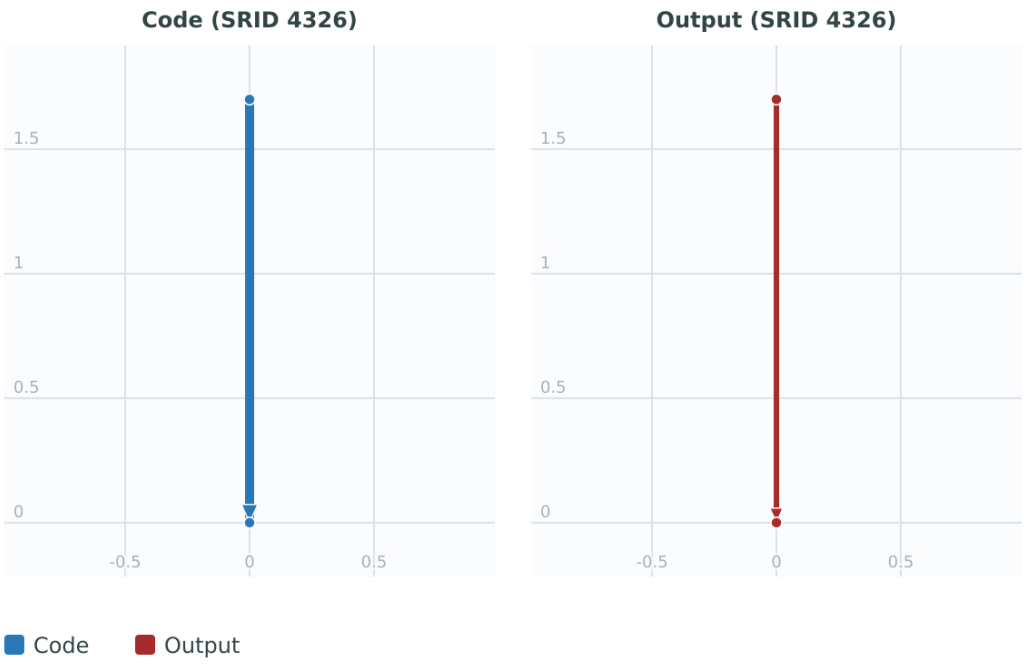
Code

```
WITH test_raster AS (
SELECT
ST_SetValues(
ST_AddBand(
ST_MakeEmptyRaster(
      width => 2,
      height => 2,
      upperleftx => 0,
      upperlefty => 2,
      scalex => 1.0,
      scaley => -1.0,
      skewx => 0,
      skewy => 0,
      srid => 4326),
    index => 1,
    pixeltype => '16BSI',
    initialvalue => 0,
    nodataval => -999
),
  1,
  1,
  1,
  newvalueset => ARRAY[ARRAY[10.0::float8, 50.0::float8], ARRAY[40.0::float8, 20.0::float8 ↩
    ]]) AS rast
)
SELECT
ST_SetM(
    rast,
    band => 1,
    geom => 'SRID=4326;LINESTRING(1.0 1.9,1.0 0.2)'::geometry,
    resample => 'bilinear'
)
FROM test_raster
```

Output

```
LINESTRING M (1 1.9 38,1 0.2 27)
```

Figure



Se även

ST_Value, ST_SetZ

11.6.11 ST_Neighborhood

ST_Neighborhood — Returnerar en 2D-array med dubbel precision av `icke-NODATA-värden` runt ett visst bands pixel som anges av antingen en kolumnX och radY eller en geometrisk punkt uttryckt i samma spatiala referenskoordinatsystem som rastret.

Synopsis

double precision[][] **ST_Neighborhood**(raster rast, integer bandnum, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
double precision[][] **ST_Neighborhood**(raster rast, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
double precision[][] **ST_Neighborhood**(raster rast, integer bandnum, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
double precision[][] **ST_Neighborhood**(raster rast, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

Description

Returnerar en 2D-array med dubbel precision av `icke-NODATA-värdena` runt ett visst bands pixel som anges av antingen en kolumnX och radY eller en geometrisk punkt uttryckt i samma spatiala referenskoordinatsystem som rastret. Parametrarna `distanceX` och `distanceY` definierar antalet pixlar runt den angivna pixeln i X- och Y-axlarna, t.ex. vill jag ha alla värden inom 3 pixlars avstånd längs

X-axeln och 2 pixlars avstånd längs Y-axeln runt min intressanta pixel. Centrumvärdet för 2D-arrayen kommer att vara värdet på den pixel som anges av kolumnX och radY eller den geometriska punkten.

Bandnumren börjar på 1 och `bandnum` antas vara 1 om det inte anges. Om `exclude_nodata_value` är satt till false anses alla pixlar inklusive `nodata-pixlar` skära varandra och returnerar värdet. Om `värdet exclude_nodata_value` inte anges läses det från rastrets metadata.

Notera
Antalet element längs varje axel i den returnerande 2D-arrayen är 2 *(`avståndX|avståndY`) + 1. Så för ett `avståndX` och ett `avståndY` på 1 blir den returnerade matrisen 3x3.

Notera
Utdata från 2D-arrayen kan skickas till någon av de inbyggda funktionerna för rasterbearbetning, t.ex. ST_Min4ma, ST_Sum4ma, ST_Mean4ma.

Tillgänglighet: 2.1.0

Exempel

This example reads a value from pixel 2x2.

Code

```
SELECT
    ST_Neighborhood(rast, 2, 2, 1, 1)
FROM (
    SELECT
        ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                '8BUI'::text, 1, 0
            ),
            1, 1, 1, ARRAY[
                [0, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 0],
                [1, 1, 0, 1, 1]
            ]::double precision[],
            1
        ) AS rast
) AS foo
```

Output

```
{{NULL,1,1},{1,1,1},{1,NULL,1}}
```

This example reads a NODATA value from pixel 2x3.

Code

```
SELECT
    ST_Neighborhood(rast, 2, 3, 1, 1)
FROM (
    SELECT
        ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                '8BUI'::text, 1, 0
```

```
            ),
            1, 1, 1, ARRAY[
                [0, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 0],
                [1, 1, 0, 1, 1]
            ]::double precision[],
            1
        ) AS rast
) AS foo
```

Output

```
{{1,1,1},{1,NULL,1},{1,1,1}}
```

Pixel 3x3 has value. Exclude_nodata_value = FALSE.

Code

```
SELECT
    ST_Neighborhood(rast, 3, 3, 1, 1, false)
FROM ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                '8BUI'::text, 1, 0
            ),
            1, 1, 1, ARRAY[
                [0, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 0],
                [1, 1, 0, 1, 1]
            ]::double precision[],
            1
        ) AS rast
```

Output

```
{{1,1,0},{0,1,1},{1,1,1}}
```

Se även

ST_NearestValue, ST_Min4ma, ST_Max4ma, ST_Sum4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.6.12 ST_SetValue

ST_SetValue — Returnerar modifierad raster som är resultatet av att värdet för ett givet band har ställts in i en given kolumnx, radpixel eller de pixlar som skär en viss geometri. Bandnummer börjar på 1 och antas vara 1 om de inte specificeras.

Synopsis

raster **ST_SetValue**(raster rast, integer bandnum, geometry geom, double precision newvalue);
raster **ST_SetValue**(raster rast, geometry geom, double precision newvalue);
raster **ST_SetValue**(raster rast, integer bandnum, integer columnx, integer rowy, double precision newvalue);
raster **ST_SetValue**(raster rast, integer columnx, integer rowy, double precision newvalue);

Description

Returnerar modifierat raster som är resultatet av att de angivna pixlarnas värden sätts till nytt värde för det angivna bandet givet rastrets rad och kolumn eller en geometri. Om inget band anges antas band 1.

Förbättrad: 2.1.0 Geometrivarianten av ST_SetValue() stöder nu alla geometrityper, inte bara punkt. Geometrivarianten är ett omslag runt geomval[]-varianten av ST_SetValues()

Exempel

This example uses geometry input.

Code

```
               SELECT (foo.geomval).val, ST_Union((foo.geomval).geom)
FROM (SELECT ST_DumpAsPolygons(ST_SetValue(
            rast,
            1,
            ST_Point(3427927.75, 5793243.95),
            50)
            ) As geomval
FROM dummy_rast
where rid = 2) As foo
WHERE (foo.geomval).val < 250
GROUP BY (foo.geomval).val;
```

Output

```
 val |                                                        st_astext
-----+---------------------------------------------------------------------
  50 | POLYGON((3427927.75 5793244,3427927.75 5793243.95,3427927.8 579324 ...
 249 | POLYGON((3427927.95 5793243.95,3427927.95 5793243.85,3427928 57932 ...
```

This example stores the changed raster.

Code

```
UPDATE dummy_rast SET rast = ST_SetValue(rast, 1, ST_Point(3427927.75, 5793243.95), 100)
    WHERE rid = 2   ;
```

Se även

ST_Value, ST_DumpAsPolygons

11.6.13 ST_SetValues

ST_SetValues — Returnerar modifierat raster som är resultatet av att värdena för ett givet band har ställts in.

Synopsis

raster **ST_SetValues**(raster rast, integer nband, integer columnx, integer rowy, double precision[][] newvalueset, boolean[][] noset=NULL, boolean keepnodata=FALSE);
raster **ST_SetValues**(raster rast, integer nband, integer columnx, integer rowy, double precision[][] newvalueset, double precision nosetvalue, boolean keepnodata=FALSE);
raster **ST_SetValues**(raster rast, integer nband, integer columnx, integer rowy, integer width, integer height, double precision newvalue, boolean keepnodata=FALSE);
raster **ST_SetValues**(raster rast, integer columnx, integer rowy, integer width, integer height, double precision newvalue, boolean keepnodata=FALSE);
raster **ST_SetValues**(raster rast, integer nband, geomval[] geomvalset, boolean keepnodata=FALSE);

Description

Returnerar modifierat raster som är resultatet av att angivna pixlar har satts till nya värden för det angivna bandet. `columnx` och `rowy` är 1-indexerade.

Om `keepnodata` är TRUE kommer de pixlar vars värden är NODATA inte att förses med motsvarande värde i `newvalueset`.

För variant 1 bestäms de specifika pixlarna som ska ställas in av `kolumnx`, `rowy-pixelkoordinaterna` och dimensionerna för `newvalueset-arrayen`. `noset` kan användas för att förhindra att pixlar med värden som finns i `newvalueset` ställs in (på grund av att PostgreSQL inte tillåter ragged / jagged arrays). Se exempel Variant 1.

Variant 2 är som variant 1 men med ett enkelt `nosetvärde` med dubbel precision i stället för en boolesk `noset-array`. Element i `newvalueset` med värdet `nosetvalue` hoppas över. Se exempel Variant 2.

För variant 3 bestäms de specifika pixlar som ska ställas in av pixelkoordinaterna för `columnx`, `rowy`, `width` och `height`. Se exempel Variant 3.

Variant 4 är densamma som variant 3 med undantaget att den förutsätter att det första bandets pixlar av `rast` kommer att ställas in.

För variant 5 används en matris med geomval för att bestämma vilka specifika pixlar som ska ställas in. Om alla geometrier i matrisen är av typen POINT eller MULTIPOINT använder funktionen en genväg där longituden och latituden för varje punkt används för att ställa in en pixel direkt. I annat fall konverteras geometrierna till raster och itereras sedan igenom i ett pass. Se exempel Variant 5.

Tillgänglighet: 2.1.0

Exempel

Variant 1.

These examples pass FALSE for *`exclude_nodata_value`* so that every raster cell remains visible. NODATA cells display the band's NODATA value, which is `0` here.

Set the lower-right 2 by 2 block of a 3 by 3 raster to 9.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
```

```
            ),
            1, 2, 2, ARRAY[[9, 9], [9, 9]]::double precision[][]
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
 x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

Replace the complete raster from an array; the central `NULL` becomes NODATA.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision[][]
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
 x | y | val
---+---+-----
 1 | 1 |   9
 1 | 2 |   9
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

Use the *noset* mask to retain the left-middle source pixel while replacing the other cells.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
```

```
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1,
                ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision[][],
                ARRAY[[false], [true]]::boolean[][]
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   9
 1 | 2 |   1
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

With *keepnodata* enabled, an existing NODATA pixel is not overwritten.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0,  ↩
    1, -1, 0, 0, 0),
                    1, '8BUI', 1, 0
                ),
                1, 1, 1, NULL
            ),
            1, 1, 1,
                ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision[][],
                ARRAY[[false], [true]]::boolean[][],
                TRUE
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   0
 1 | 2 |   1
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

Variant 2.

Use `-1` as the value that should not be written; only the lower-right 2 by 2 block changes.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[-1, -1, -1], [-1, 9, 9], [-1, 9, 9]]::double precision[][], -1
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

The same update can use `NULL` as the value that should not be written.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[NULL, NULL, NULL], [NULL, 9, 9], [NULL, 9, 9]]::double  ↩
    precision[][], NULL::double precision
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
```

```
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

Variant 3.

Set a 2 by 2 rectangle starting at column 2, row 2.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 2, 2, 2, 2, 9
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

When *keepnodata* is true, the NODATA cell inside the rectangle remains unchanged.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0,  ↩
    1, -1, 0, 0, 0),
                    1, '8BUI', 1, 0
                ),
                1, 2, 2, NULL
            ),
            1, 2, 2, 2, 2, 9, TRUE
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

Variant 5.

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI',  ↩
    0, 0) AS rast
), bar AS (
    SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)'::geometry geom UNION ALL
    SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1,4 -1,4 -4,1 -4,1 -1))'::geometry geom UNION ALL
    SELECT 3 AS gid, 'SRID=0;POLYGON((0 0,5 0,5 -1,1 -1,1 -4,0 -4,0 0))'::geometry geom  ↩
    UNION ALL
    SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0,4 4,4 -4)'::geometry
), changed AS (
    SELECT rid, gid, ST_SetValue(rast, 1, geom, gid) AS rast
    FROM foo
    CROSS JOIN bar
)
SELECT
    rid,
    gid,
    y,
    array_agg(ST_Value(rast, 1, x, y) ORDER BY x) AS values
FROM changed
CROSS JOIN generate_series(1, 5) AS x
CROSS JOIN generate_series(1, 5) AS y
GROUP BY rid, gid, y
ORDER BY rid, gid, y;
```

Output

```
rid | gid | y |           values
-----+-----+---+----------------------------
   1 |   1 | 1 | {NULL,NULL,NULL,NULL,NULL}
   1 |   1 | 2 | {NULL,NULL,NULL,NULL,NULL}
   1 |   1 | 3 | {NULL,NULL,1,NULL,NULL}
   1 |   1 | 4 | {NULL,NULL,NULL,NULL,NULL}
   1 |   1 | 5 | {NULL,NULL,NULL,NULL,NULL}
   1 |   2 | 1 | {NULL,NULL,NULL,NULL,NULL}
   1 |   2 | 2 | {NULL,2,2,2,NULL}
   1 |   2 | 3 | {NULL,2,2,2,NULL}
   1 |   2 | 4 | {NULL,2,2,2,NULL}
   1 |   2 | 5 | {NULL,NULL,NULL,NULL,NULL}
   1 |   3 | 1 | {3,3,3,3,3}
   1 |   3 | 2 | {3,NULL,NULL,NULL,NULL}
   1 |   3 | 3 | {3,NULL,NULL,NULL,NULL}
   1 |   3 | 4 | {3,NULL,NULL,NULL,NULL}
   1 |   3 | 5 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 1 | {4,NULL,NULL,NULL,NULL}
   1 |   4 | 2 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 3 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 4 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 5 | {NULL,NULL,NULL,NULL,4}
(20 rows)
```

Följande exempel visar att geomval senare i matrisen kan skriva över tidigare geomval

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI',  ↩
    0, 0) AS rast
), bar AS (
    SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)'::geometry geom UNION ALL
    SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1,4 -1,4 -4,1 -4,1 -1))'::geometry geom UNION ALL
    SELECT 3 AS gid, 'SRID=0;POLYGON((0 0,5 0,5 -1,1 -1,1 -4,0 -4,0 0))'::geometry geom  ↩
    UNION ALL
    SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0,4 4,4 -4)'::geometry
), changed AS (
    SELECT ST_SetValues(
        rast,
        1,
        ARRAY[
            ROW((SELECT geom FROM bar WHERE gid = 1), 1),
            ROW((SELECT geom FROM bar WHERE gid = 2), 2)
        ]::geomval[]
    ) AS rast
    FROM foo
)
SELECT
    y,
    array_agg(ST_Value(rast, 1, x, y) ORDER BY x) AS values
FROM changed
CROSS JOIN generate_series(1, 5) AS x
CROSS JOIN generate_series(1, 5) AS y
GROUP BY y
ORDER BY y;
```

Output

```
y |           values
---+---------------------------
 1 | {NULL,NULL,NULL,NULL,NULL}
 2 | {NULL,2,2,2,NULL}
 3 | {NULL,2,2,2,NULL}
 4 | {NULL,2,2,2,NULL}
 5 | {NULL,NULL,NULL,NULL,NULL}
(5 rows)
```

Detta exempel är motsatsen till det föregående exemplet

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI',  ↩
    0, 0) AS rast
), bar AS (
    SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)'::geometry geom UNION ALL
    SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1,4 -1,4 -4,1 -4,1 -1))'::geometry geom UNION ALL
    SELECT 3 AS gid, 'SRID=0;POLYGON((0 0,5 0,5 -1,1 -1,1 -4,0 -4,0 0))'::geometry geom  ↩
    UNION ALL
    SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0,4 4,4 -4)'::geometry
), changed AS (
    SELECT ST_SetValues(
        rast,
        1,
        ARRAY[
            ROW((SELECT geom FROM bar WHERE gid = 2), 2),
            ROW((SELECT geom FROM bar WHERE gid = 1), 1)
        ]::geomval[]
    ) AS rast
```

```
    FROM foo
)
SELECT
    y,
    array_agg(ST_Value(rast, 1, x, y) ORDER BY x) AS values
FROM changed
CROSS JOIN generate_series(1, 5) AS x
CROSS JOIN generate_series(1, 5) AS y
GROUP BY y
ORDER BY y;
```

Output

```
y |           values
---+---------------------------
 1 | {NULL,NULL,NULL,NULL,NULL}
 2 | {NULL,2,2,2,NULL}
 3 | {NULL,2,1,2,NULL}
 4 | {NULL,2,2,2,NULL}
 5 | {NULL,NULL,NULL,NULL,NULL}
(5 rows)
```

Se även

ST_Value, ST_SetValue, ST_PixelAsPolygons

11.6.14 ST_DumpValues

ST_DumpValues — Hämta värdena för det angivna bandet som en 2-dimensionell array.

Synopsis

setof record **ST_DumpValues**(raster rast , integer[] nband=NULL , boolean exclude_nodata_value=true);
double precision[][] **ST_DumpValues**(raster rast , integer nband , boolean exclude_nodata_value=true);

Description

Hämta värdena för det angivna bandet som en 2-dimensionell array (första indexet är rad, andra är kolumn). Om `nband` är NULL eller inte anges bearbetas alla rasterband.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), ↩
     1, '8BUI'::text, 1, 0), 2, '32BF'::text, 3, -9999), 3, '16BSI', 0, 0) AS rast
)
SELECT
    (ST_DumpValues(rast)).*
FROM foo;
```

Output

```
nband |                      valarray
-------+-----------------------------------------------------
     1 | {{1,1,1},{1,1,1},{1,1,1}}
     2 | {{3,3,3},{3,3,3},{3,3,3}}
     3 | {{NULL,NULL,NULL},{NULL,NULL,NULL},{NULL,NULL,NULL}}
(3 rows)
```

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), ↩
     1, '8BUI'::text, 1, 0), 2, '32BF'::text, 3, -9999), 3, '16BSI', 0, 0) AS rast
)
SELECT
    (ST_DumpValues(rast, ARRAY[3, 1])).*
FROM foo;
```

Output

```
nband |                      valarray
-------+-----------------------------------------------------
     3 | {{NULL,NULL,NULL},{NULL,NULL,NULL},{NULL,NULL,NULL}}
     1 | {{1,1,1},{1,1,1},{1,1,1}}
(2 rows)
```

Code

```
WITH foo AS (
    SELECT ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 1, 0), 1, 2, 5) AS rast
)
SELECT
    (ST_DumpValues(rast, 1))[2][1]
FROM foo;
```

Output

```
5
```

Se även

ST_Value, ST_SetValue, ST_SetValues

11.6.15 ST_PixelOfValue

ST_PixelOfValue — Hämta koordinaterna för kolumnx, rowy för den pixel vars värde är lika med sökvärdet.

Synopsis

setof record **ST_PixelOfValue**(raster rast , integer nband , double precision[] search , boolean exclude_nodata_value=true);
setof record **ST_PixelOfValue**(raster rast , double precision[] search , boolean exclude_nodata_value=true);
setof record **ST_PixelOfValue**(raster rast , integer nband , double precision search , boolean exclude_nodata_value=true);
setof record **ST_PixelOfValue**(raster rast , double precision search , boolean exclude_nodata_value=true);

Description

Hämta koordinaterna för kolumnx och radwy för den pixel vars värde är lika med sökvärdet. Om inget band anges antas band 1.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
            1, '8BUI', 1, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 1, 1, 255],
            [1, 1, 0, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT (ST_PixelOfValue(rast, 1, ARRAY[1, 255])).*
FROM source
```

Output

```
val | x | y
-----+---+---
   1 | 1 | 2
   1 | 1 | 3
   1 | 1 | 4
   1 | 1 | 5
   1 | 2 | 1
   1 | 2 | 2
   1 | 2 | 4
   1 | 2 | 5
   1 | 3 | 1
   1 | 3 | 2
   1 | 3 | 3
   1 | 3 | 4
   1 | 4 | 1
   1 | 4 | 3
   1 | 4 | 4
   1 | 4 | 5
   1 | 5 | 1
   1 | 5 | 2
   1 | 5 | 3
 255 | 5 | 4
   1 | 5 | 5
```

11.7 Raster-redigerare

11.7.1 ST_SetGeoReference

ST_SetGeoReference — Set Georeference 6 georeferensparametrar i ett enda anrop. Siffrorna ska separeras med vitt utrymme. Accepterar inmatningar i GDAL- eller ESRI-format. Standard är GDAL.

Synopsis

raster **ST_SetGeoReference**(raster rast, text georefcoords, text format=GDAL);
raster **ST_SetGeoReference**(raster rast, double precision upperleftx, double precision upperlefty, double precision scalex, double precision scaley, double precision skewx, double precision skewy);

Description

Set Georeference 6 georeferensparametrar i ett enda anrop. Accepterar indata i GDAL- eller ”ESRI-format. Standard är GDAL. Om 6 koordinater inte anges kommer null att returneras.

Skillnaden mellan formatrepresentationer är följande:

GDAL uses the value order shown below.

Code

```
scalex skewy skewx scaley upperleftx upperlefty
```

ESRI uses the value order shown below.

Code

```
scalex skewy skewx scaley upperleftx + scalex*0.5 upperlefty + scaley*0.5
```

Notera
Om rastret har out-db-band kan en ändring av georeferensen leda till felaktig åtkomst till bandets externt lagrade data.

Förbättrad: 2.1.0 Tillägg av varianten ST_SetGeoReference(raster, dubbel precision, ...)

Exempel

Code

```
WITH source AS (
    SELECT ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0) AS rast
), variants AS (
    SELECT 0 AS rid, rast FROM source
    UNION ALL
    SELECT 1, ST_SetGeoReference(rast, '10 0 0 -10 0.1 0.1', 'GDAL')
    FROM source
    UNION ALL
    SELECT 2, ST_SetGeoReference(rast, '10 0 0 -10 5.1 -4.9', 'ESRI')
    FROM source
    UNION ALL
    SELECT 3, ST_SetGeoReference(rast, 1, 1, 10, -10, 0.001, 0.001)
    FROM source
)
SELECT
    rid,
    round(ST_UpperLeftX(rast)::numeric, 3) AS upperleftx,
    round(ST_UpperLeftY(rast)::numeric, 3) AS upperlefty,
    ST_ScaleX(rast) AS scalex,
    ST_ScaleY(rast) AS scaley,
    ST_SkewX(rast) AS skewx,
    ST_SkewY(rast) AS skewy
FROM variants
ORDER BY rid;
```

Output

```
rid | upperleftx | upperlefty | scalex | scaley | skewx | skewy
-----+------------+------------+--------+--------+-------+-------
   0 |      0.000 |      0.000 |      1 |     -1 |     0 |     0
   1 |      0.100 |      0.100 |     10 |    -10 |     0 |     0
   2 |      0.100 |      0.100 |     10 |    -10 |     0 |     0
   3 |      1.000 |      1.000 |     10 |    -10 | 0.001 | 0.001
(4 rows)
```

Se även

ST_GeoReference, ST_ScaleX, ST_ScaleY, ST_UpperLeftX, ST_UpperLeftY

11.7.2 ST_SetRotation

ST_SetRotation — Ställ in rastrets rotation i radian.

Synopsis

raster **ST_SetRotation**(raster rast, float8 rotation);

Description

Rotera rastret på ett enhetligt sätt. Rotationen anges i radianer. Se World File för mer information.

Exempel

Code

```
WITH variants AS (
    SELECT rid, 'rotated' AS kind, ST_SetRotation(rast, 15) AS rast
    FROM dummy_rast
    UNION ALL
    SELECT rid, 'original', rast
    FROM dummy_rast
)
SELECT
    rid,
    kind,
    round(ST_ScaleX(rast)::numeric, 4) AS scale_x,
    round(ST_ScaleY(rast)::numeric, 4) AS scale_y,
    round(ST_SkewX(rast)::numeric, 4) AS skew_x,
    round(ST_SkewY(rast)::numeric, 4) AS skew_y
FROM variants
ORDER BY rid, kind;
```

Output

```
rid |   kind   | scale_x | scale_y | skew_x | skew_y
-----+----------+---------+---------+--------+--------
   1 | original |  2.0000 |  3.0000 | 0.0000 | 0.0000
   1 | rotated  | -1.5194 | -2.2791 | 1.9509 | 1.3006
   2 | original |  0.0500 | -0.0500 | 0.0000 | 0.0000
   2 | rotated  | -0.0380 | -0.0380 | 0.0325 | 0.0325
(4 rows)
```

Se även

ST_Rotation, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.7.3 ST_SetScale

ST_SetScale — Ställer in X- och Y-storleken för pixlar i enheter i koordinatreferenssystemet. Antal enheter/pixelbredd/höjd.

Synopsis

raster **ST_SetScale**(raster rast, float8 xy);
raster **ST_SetScale**(raster rast, float8 x, float8 y);

Description

Ställer in X- och Y-storleken för pixlar i enheter i koordinatreferenssystemet. Antal enheter/pixelbredd/höjd. Om endast en enhet skickas in antas X och Y vara samma antal.

Notera
ST_SetScale skiljer sig från ST_Rescale genom att ST_SetScale inte gör om rastret för att matcha rastrets utsträckning. Den ändrar bara rastrets metadata (eller georeferens) för att korrigera en ursprungligen felaktigt specificerad skalning. ST_Rescale resulterar i ett raster med olika bredd och höjd som beräknats för att passa den geografiska omfattningen av indatarastret. ST_SetScale ändrar inte bredden eller höjden på rastret.

Ändrad: 2.0.0 I WKTRaster-versioner kallades detta ST_SetPixelSize. Detta ändrades i 2.0.0.

Exempel

Code

```
UPDATE dummy_rast
    SET rast = ST_SetScale(rast, 1.5)
WHERE rid = 2;

SELECT ST_ScaleX(rast) As pixx, ST_ScaleY(rast) As pixy, Box3D(rast) As newbox
FROM dummy_rast
WHERE rid = 2;
```

Output

```
pixx | pixy |                    newbox
------+------+-----------------------------------------------
  1.5 |  1.5 | BOX(3427927.75 5793244 0, 3427935.25 5793251.5 0)
```

Code

```
UPDATE dummy_rast
    SET rast = ST_SetScale(rast, 1.5, 0.55)
WHERE rid = 2;

SELECT ST_ScaleX(rast) As pixx, ST_ScaleY(rast) As pixy, Box3D(rast) As newbox
FROM dummy_rast
WHERE rid = 2;
```

Output

```
pixx | pixy |                    newbox
------+------+-------------------------------------------
  1.5 | 0.55 | BOX(3427927.75 5793244 0,3427935.25 5793247 0)
```

Se även

ST_ScaleX, ST_ScaleY, Box3D

11.7.4 ST_SetSkew

ST_SetSkew — Ställer in georeferensens X- och Y-skevhet (eller rotationsparameter). Om endast en parameter anges, sätts X och Y till samma värde.

Synopsis

raster **ST_SetSkew**(raster rast, float8 skewxy);
raster **ST_SetSkew**(raster rast, float8 skewx, float8 skewy);

Description

Ställer in georeferensens X- och Y-skevhet (eller rotationsparameter). Om endast en parameter skickas in, sätts X och Y till samma värde. Se World File för mer information.

Exempel

This is the first example.

Code

```
UPDATE dummy_rast SET rast = ST_SetSkew(rast, 1, 2) WHERE rid = 1;
SELECT
    rid,
    format('(%s, %s)', skewx, skewy) AS skew,
    term,
    georef[idx] AS value
FROM (
    SELECT
        rid,
        ST_SkewX(rast) AS skewx,
        ST_SkewY(rast) AS skewy,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
    WHERE rid = 1
) AS refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY idx;
```

Output

```
rid |  skew  |  term   |    value
-----+--------+---------+-------------
   1 | (1, 2) | scaleX  | 2.0000000000
   1 | (1, 2) | skewY   | 2.0000000000
   1 | (1, 2) | skewX   | 1.0000000000
   1 | (1, 2) | scaleY  | 3.0000000000
   1 | (1, 2) | originX | 0.5000000000
   1 | (1, 2) | originY | 0.5000000000
(6 rows)
```

This second example sets both values to the same number.

Code

```
UPDATE dummy_rast SET rast = ST_SetSkew(rast, 0) WHERE rid = 1;
SELECT
    rid,
    format('(%s, %s)', skewx, skewy) AS skew,
    term,
    georef[idx] AS value
FROM (
    SELECT
        rid,
        ST_SkewX(rast) AS skewx,
        ST_SkewY(rast) AS skewy,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
    WHERE rid = 1
) AS refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY idx;
```

Output

```
rid |  skew  |  term   |    value
-----+--------+---------+-------------
   1 | (0, 0) | scaleX  | 2.0000000000
   1 | (0, 0) | skewY   | 0.0000000000
   1 | (0, 0) | skewX   | 0.0000000000
   1 | (0, 0) | scaleY  | 3.0000000000
   1 | (0, 0) | originX | 0.5000000000
   1 | (0, 0) | originY | 0.5000000000
(6 rows)
```

Se även

ST_GeoReference, ST_SetGeoReference, ST_SkewX, ST_SkewY

11.7.5 ST_SetSRID

ST_SetSRID — Ställer in SRID för ett raster till ett visst heltal srid som definieras i tabellen spatial_ref_sys.

Synopsis

raster **ST_SetSRID**(raster rast, integer srid);

Description

Ställer in SRID på ett raster till ett visst heltalsvärde.

Notera
Den här funktionen transformerar inte rastret på något sätt - den anger bara metadata som definierar den spatiala ref för det koordinatreferenssystem som det för närvarande befinner sig i. Användbart för transformationer senare.

Se även

Avsnitt 4.5, ST_SRID

11.7.6 ST_SetUpperLeft

ST_SetUpperLeft — Ställer in värdet för det övre vänstra hörnet av pixeln i rastret till projicerade X- och Y-koordinater.

Synopsis

raster **ST_SetUpperLeft**(raster rast, double precision x, double precision y);

Description

Ställ in värdet för det övre vänstra hörnet av rastret till de projicerade X- och Y-koordinaterna

Exempel

Code

```
SELECT ST_SetUpperLeft(rast, -71.01, 42.37)
FROM dummy_rast
WHERE rid = 2;
```

Se även

ST_UpperLeftX, ST_UpperLeftY

11.7.7 ST_Resample

ST_Resample — Resampla ett raster med hjälp av en specificerad resamplingsalgoritm, nya dimensioner, ett godtyckligt rutnätshörn och en uppsättning rastergeoreferensattribut som definierats eller lånats från ett annat raster.

Synopsis

raster **ST_Resample**(raster rast, integer width, integer height, double precision gridx=NULL, double precision gridy=NULL, double precision skewx=0, double precision skewy=0, text algorithm=NearestNeigh double precision maxerr=0.125);
raster **ST_Resample**(raster rast, double precision scalex=0, double precision scaley=0, double precision gridx=NULL, double precision gridy=NULL, double precision skewx=0, double precision skewy=0, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Resample**(raster rast, raster ref, text algorithm=NearestNeighbor, double precision maxerr=0.125, boolean usescale=true);
raster **ST_Resample**(raster rast, raster ref, boolean usescale, text algorithm=NearestNeighbor, double precision maxerr=0.125);

Description

Resampla ett raster med en specificerad resamplingsalgoritm, nya dimensioner (bredd & höjd), ett rasterhörn (gridx & gridy) och en uppsättning rastergeoreferensattribut (scalex, scaley, skewx & skewy) definierade eller lånade från ett annat raster. Om man använder ett referensraster måste de två rastren ha samma SRID.

Nya pixelvärden beräknas med hjälp av någon av följande omsamplingsalgoritmer:

- NearestNeighbor (engelsk eller amerikansk stavning)
- Bilinjär
- Kubisk
- CubicSpline
- Lanczos
- Max
- Min

Standardinställningen är NearestNeighbor, vilket är snabbast men ger den sämsta interpoleringen.

En maxerrorprocent på 0,125 används om ingen `maxerr` har angetts.

Notera
Hänvisa till: GDAL Warp resampling methods för mer information.

Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+

Förbättrad: 3.4.0 max- och min-alternativ för omsampling har lagts till

Exempel

Code

```
SELECT
    ST_Width(orig) AS orig_width,
    ST_Width(reduce_100) AS new_width
FROM (
    SELECT
        rast AS orig,
        ST_Resample(rast, 100, 100) AS reduce_100
    FROM aerials.boston
    WHERE ST_Intersects(
        rast,
        ST_Transform(ST_MakeEnvelope(-71.128, 42.2392, -71.1277, 42.2397, 4326),
            26986)
    )
    LIMIT 1
) AS foo;
```

Output

```
orig_width | new_width
-----------+-------------
       200 |         100
```

Se även

ST_Rescale, ST_Resize, ST_Transform

11.7.8 ST_Rescale

ST_Rescale — Resampla ett raster genom att endast justera dess skala (eller pixelstorlek). Nya pixelvärden beräknas med hjälp av omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline, Lanczos, Max eller Min. Standard är NearestNeighbor.

Synopsis

raster **ST_Rescale**(raster rast, double precision scalexy, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Rescale**(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeig
double precision maxerr=0.125);

Description

Omsampla ett raster genom att endast justera dess skala (eller pixelstorlek). Nya pixelvärden beräknas med hjälp av någon av följande omsamplingsalgoritmer:

- NearestNeighbor (engelsk eller amerikansk stavning)
- Bilinjär
- Kubisk
- CubicSpline
- Lanczos
- Max

- Min

Standardinställningen är NearestNeighbor, vilket är snabbast men ger den sämsta interpoleringen.

`scalex` och `scaley` definierar den nya pixelstorleken. scaley måste ofta vara negativ för att få ett välorienterat raster.

När den nya scalex eller scaley inte är en divisor av rastrets bredd eller höjd, utökas det resulterande rastrets utsträckning så att det omfattar det tillhandahållna rastrets utsträckning. Om du vill vara säker på att behålla den exakta omfattningen av inmatningen, se ST_Resize

`maxerr` är tröskelvärdet för omsamplingsalgoritmens approximation av transformationen (i pixelenheter). Ett standardvärde på 0,125 används om `maxerr` inte anges, vilket är samma värde som används i GDAL:s verktyg gdalwarp. Om värdet sätts till noll sker ingen approximering.

Notera
Hänvisa till: GDAL Warp resampling methods för mer information.

Notera
ST_Rescale skiljer sig från ST_SetScale genom att ST_SetScale inte omsamplar rastret för att matcha rastrets utsträckning. ST_SetScale ändrar endast rastrets metadata (eller georeferens) för att korrigera en ursprungligen felaktigt specificerad skalning. ST_Rescale resulterar i ett raster med olika bredd och höjd som beräknats för att passa den geografiska omfattningen av indatarastret. ST_SetScale ändrar inte bredden eller höjden på rastret.

Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+

Förbättrad: 3.4.0 max- och min-alternativ för omsampling har lagts till

Ändrad: 2.1.0 Fungerar på raster utan SRID

Exempel

Ett enkelt exempel är omskalning av ett raster från en pixelstorlek på 0,001 grader till en pixelstorlek på 0,0015 grader.

This example returns the original raster pixel size.

Code

```
SELECT ST_PixelWidth(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0,  ↩
    4269), '8BUI'::text, 1, 0)) width
```

Output

```
0.001
```

This example returns the pixel size of the rescaled raster.

Code

```
SELECT ST_PixelWidth(ST_Rescale(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001,  ↩
    -0.001, 0, 0, 4269), '8BUI'::text, 1, 0), 0.0015)) width
```

Output

```
0.0015
```

Se även

ST_Resize, ST_Resample, ST_SetScale, ST_ScaleX, ST_ScaleY, ST_Transform

11.7.9 ST_Reskew

ST_Reskew — Resampla ett raster genom att endast justera dess skevhet (eller rotationsparametrar). Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är NearestNeighbor.

Synopsis

raster **ST_Reskew**(raster rast, double precision skewxy, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Reskew**(raster rast, double precision skewx, double precision skewy, text algorithm=NearestNeig
double precision maxerr=0.125);

Description

Resampla ett raster genom att endast justera dess skevhet (eller rotationsparametrar). Nya pixelvärden beräknas med omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos. Standardinställningen är NearestNeighbor, som är snabbast men ger den sämsta interpoleringen.

`skewx` och `skewy` definierar den nya skew.

Utbredningen av det nya rastret kommer att omfatta utbredningen av det tillhandahållna rastret.

En maxerror-procent på 0,125 om ingen `maxerr` har angetts.

Notera
Hänvisa till: GDAL Warp resampling methods för mer information.

Notera
ST_Reskew skiljer sig från ST_SetSkew på så sätt att ST_SetSkew inte omsamplar rastret för att matcha rastrets utsträckning. ST_SetSkew ändrar bara rastrets metadata (eller georeferens) för att korrigera en ursprungligen felaktigt angiven skevhet. ST_Reskew resulterar i ett raster med olika bredd och höjd som beräknats för att passa den geografiska utbredningen av indatarastret. ST_SetSkew ändrar inte bredden eller höjden på rastret.

Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+

Ändrad: 2.1.0 Fungerar på raster utan SRID

Exempel

Ett enkelt exempel på hur ett raster kan ändras från en skevhet på 0,0 till en skevhet på 0,0015.

This example uses the original non-rotated raster.

Code

```
SELECT ST_Rotation(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0, 4269) ↩
    , '8BUI'::text, 1, 0));
```

Output

```
st_rotation
------------
          0
(1 row)
```

The reskewed raster rotation.

Code

```
SELECT ST_Rotation(ST_Reskew(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001,  ↩
    0, 0, 4269), '8BUI'::text, 1, 0), 0.0015));
```

Output

```
st_rotation
------------------
 -0.982793723247329
(1 row)
```

Se även

ST_Resample, ST_Rescale, ST_SetSkew, ST_SetRotation, ST_SkewX, ST_SkewY, ST_Transform

11.7.10 ST_SnapToGrid

ST_SnapToGrid — Sampla om ett raster genom att fästa det i ett rutnät. Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är NearestNeighbor.

Synopsis

raster **ST_SnapToGrid**(raster rast, double precision gridx, double precision gridy, text algorithm=NearestN
double precision maxerr=0.125, double precision scalex=DEFAULT 0, double precision scaley=DEFAULT
0);
raster **ST_SnapToGrid**(raster rast, double precision gridx, double precision gridy, double precision
scalex, double precision scaley, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_SnapToGrid**(raster rast, double precision gridx, double precision gridy, double precision
scalexy, text algorithm=NearestNeighbor, double precision maxerr=0.125);

Description

Resample en raster genom att snappa den till ett rutnät som definieras av ett godtyckligt pixelhörn (gridx & gridy) och eventuellt en pixelstorlek (scalex & scaley). Nya pixelvärden beräknas med hjälp av omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos. Standardinställningen är NearestNeighbor, som är snabbast men ger den sämsta interpoleringen.

`gridx` och `gridy` definierar ett godtyckligt pixelhörn i det nya rastret. Detta är inte nödvändigtvis det övre vänstra hörnet av det nya rastret och det behöver inte ligga inom eller på kanten av det nya rastrets utsträckning.

Du kan eventuellt definiera pixelstorleken för det nya rutnätet med `scalex` och `scaley`.
Utbredningen av det nya rastret kommer att omfatta utbredningen av det tillhandahållna rastret.
En maxerror-procent på 0,125 om ingen `maxerr` har angetts.

Notera
Hänvisa till: GDAL Warp resampling methods för mer information.

Notera
Använd ST_Resample om du behöver mer kontroll över gridparametrarna.

Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+
Ändrad: 2.1.0 Fungerar på raster utan SRID

Exempel

Ett enkelt exempel på hur man snappar ett raster till ett något annorlunda rutnät.
This example returns the upper-left X coordinate of the original raster.

Code

```
SELECT ST_UpperLeftX(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 0.001, -0.001, 0, 0, 4269) ↩
    , '8BUI'::text, 1, 0));
```

Output

```
st_upperleftx
--------------
             0
(1 row)
```

The upper left of raster after snapping.

Code

```
SELECT ST_UpperLeftX(ST_SnapToGrid(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 0.001,  ↩
    -0.001, 0, 0, 4269), '8BUI'::text, 1, 0), 0.0002, 0.0002));
```

Output

```
st_upperleftx
---------------
       -0.0008
(1 row)
```

Se även

ST_Resample, ST_Rescale, ST_UpperLeftX, ST_UpperLeftY

11.7.11 ST_Resize

ST_Resize — Ändra storlek på ett raster till en ny bredd/höjd

Synopsis

raster **ST_Resize**(raster rast, integer width, integer height, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Resize**(raster rast, double precision percentwidth, double precision percentheight, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Resize**(raster rast, text width, text height, text algorithm=NearestNeighbor, double precision maxerr=0.125);

Description

Ändra storlek på ett raster till en ny bredd/höjd. Den nya bredden/höjden kan anges i exakt antal pixlar eller som en procentandel av rastrets bredd/höjd. Omfattningen av det nya rastret kommer att vara densamma som omfattningen av det tillhandahållna rastret.

Nya pixelvärden beräknas med hjälp av omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos. Standardinställningen är NearestNeighbor, som är snabbast men ger den sämsta interpoleringen.

Variant 1 förväntar sig den faktiska bredden/höjden på utdatarastret.

Variant 2 förväntar sig decimalvärden mellan noll (0) och ett (1) som anger procentandelen av inmatningsrastrets bredd/höjd.

Variant 3 tar antingen den faktiska bredden/höjden på utdatarastret eller en procentandel i textform (”20%”) som anger procentandelen av inmatningsrastrets bredd/höjd.

Tillgänglighet: 2.1.0 Kräver GDAL 1.6.1+

Exempel

Code

```
WITH foo AS(
SELECT
    1 AS rid,
    ST_Resize(ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0),
            1, '8BUI', 255, 0
        ),
    '50%', '500') AS rast
UNION ALL
SELECT
    2 AS rid,
    ST_Resize(ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0),
            1, '8BUI', 255, 0
        ),
    500, 100) AS rast
UNION ALL
SELECT
    3 AS rid,
    ST_Resize(ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0),
            1, '8BUI', 255, 0
        ),
    0.25, 0.9) AS rast
)
SELECT
    rid,
    ST_Width(rast) AS width,
    ST_Height(rast) AS height,
    ST_ScaleX(rast) AS scale_x,
```

```
    round(ST_ScaleY(rast)::numeric, 4) AS scale_y
FROM foo
ORDER BY rid
```

Output

```
rid | width | height | scale_x | scale_y
-----+-------+--------+---------+---------
   1 |   500 |    500 |       2 | -2.0000
   2 |   500 |    100 |       2 | -10.0000
   3 |   250 |    900 |       4 | -1.1111
(3 rows)
```

Se även

ST_Resample, ST_Rescale, ST_Reskew, ST_SnapToGrid

11.7.12 ST_Transform

ST_Transform — Återprojicerar ett raster i ett känt spatialt referenssystem till ett annat känt spatialt referenssystem med hjälp av en angiven omsamplingsalgoritm. Alternativen är NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos med NearestNeighbor som standard.

Synopsis

raster **ST_Transform**(raster rast, integer srid, text algorithm=NearestNeighbor, double precision maxerr=0.125, double precision scalex, double precision scaley);
raster **ST_Transform**(raster rast, integer srid, double precision scalex, double precision scaley, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Transform**(raster rast, raster alignto, text algorithm=NearestNeighbor, double precision maxerr=0.125);

Description

Återprojicerar ett raster i ett känt spatialt referenssystem till ett annat känt spatialt referenssystem med hjälp av en angiven pixelwarpingalgoritm. Använder NearestNeighboröm ingen algoritm har angetts och maxerror procent på 0,125 om ingen maxerr har angetts.

Algoritmalternativen är: NearestNeighbor", Bilinear", "Cubic", "CubicSplineöch Lanczos". Hänvisa till: GDAL Warp resampling methods för mer information.

ST_Transform förväxlas ofta med ST_SetSRID(). ST_Transform ändrar faktiskt koordinaterna för ett raster (och omsamplar pixelvärdena) från ett spatialt referenssystem till ett annat, medan ST_SetSRID() helt enkelt ändrar SRID-identifieraren för rastret.

Till skillnad från de andra varianterna kräver variant 3 ett referensraster som `alignto`. Det transformerade rastret kommer att transformeras till referensrastrets spatiala referenssystem (SRID) och anpassas (ST_SameAlignment = TRUE) till referensrastret.

Notera
If you find your transformation support is not working right, you may need to set the environment variable PROJSO to the projection library your PostGIS build is using. This just needs to have the name of the file. So for example on windows, you would in Control Panel -> System -> Environment Variables add a system variable called PROJSO and set it to `libproj.dll`. You'll have to restart your PostgreSQL service/daemon after this change.

Varning
När du omvandlar en täckning av plattor vill du nästan alltid använda ett referensraster för att säkerställa samma inriktning och inga luckor i dina plattor, vilket visas i exemplet: Variant 3.

Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+

Förbättrad: 2.1.0 Tillägg av ST_Transform(rast, alignto)-variant

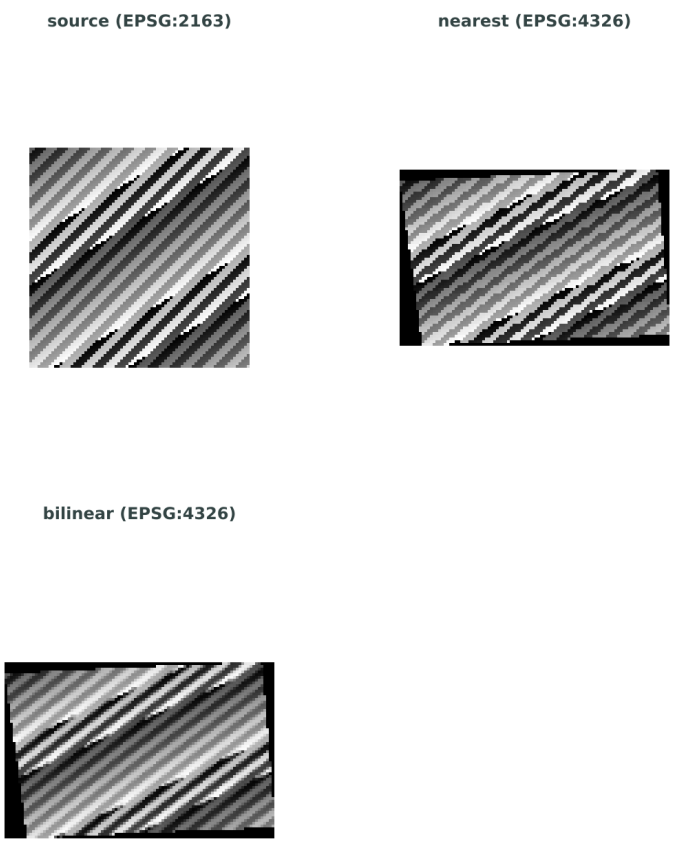
Exempel

Compare a source raster with nearest-neighbor and bilinear reprojections. The diagonal bands make the resampling behavior visible: nearest-neighbor keeps hard pixel steps, while bilinear smooths values between pixels. All three figures are generated from this query at manual build time.

Code

```
WITH rows AS (
    SELECT
        y,
        array_agg(
            ((
                    x * 3 + y * 5
                    + CASE WHEN (x + y) % 8 < 4 THEN 80 ELSE 0 END
                ) % 255)::double precision
            ORDER BY x
        ) AS row_values
    FROM generate_series(1, 80) AS y
    CROSS JOIN generate_series(1, 80) AS x
    GROUP BY y
), source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(
                80, 80,
                -500000, 600000,
                2500, -2500,
                0, 0,
                2163
            ),
            1, '8BUI', 0, NULL
        ),
        1, 1, 1,
        array_agg(row_values ORDER BY y)::double precision[][]
    ) AS rast
    FROM rows
), transformed AS (
    SELECT
        rast,
        ST_Transform(rast, 4326, 'NearestNeighbor') AS nearest,
        ST_Transform(rast, 4326, 'Bilinear') AS bilinear
    FROM source
)
SELECT
    ST_AsPNG(rast) AS "source (EPSG:2163)",
    ST_AsPNG(nearest) AS "nearest (EPSG:4326)",
    ST_AsPNG(bilinear) AS "bilinear (EPSG:4326)"
FROM transformed;
```

Figure



Raster reprojection and resampling.

Massachusetts State Plane source

WGS 84 nearest-neighbor

WGS 84 bilinear

Variant 3.

Följande visar skillnaden mellan att använda ST_Transform(raster, srid) och ST_Transform(raster, alignto)

Code

```
WITH foo AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 600000, 100, -100, 0, 0,
    2163), 1, '16BUI', 1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 600000, 100, -100, 0, 0, 2163),
    1, '16BUI', 2, 0) AS rast UNION ALL
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 600000, 100, -100, 0, 0, 2163),
    1, '16BUI', 3, 0) AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 599800, 100, -100, 0, 0, 2163),
    1, '16BUI', 10, 0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 599800, 100, -100, 0, 0, 2163),
    1, '16BUI', 20, 0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 599800, 100, -100, 0, 0, 2163),
    1, '16BUI', 30, 0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 599600, 100, -100, 0, 0, 2163),
    1, '16BUI', 100, 0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 599600, 100, -100, 0, 0, 2163),
    1, '16BUI', 200, 0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 599600, 100, -100, 0, 0, 2163),
    1, '16BUI', 300, 0) AS rast
), bar AS (
    SELECT
        ST_Transform(rast, 4269) AS alignto
    FROM foo
    LIMIT 1
), baz AS (
    SELECT
        rid,
        rast,
        ST_Transform(rast, 4269) AS not_aligned,
        ST_Transform(rast, alignto) AS aligned
    FROM foo
    CROSS JOIN bar
)
SELECT
    ST_Collect(ST_ConvexHull(not_aligned) ORDER BY rid) AS not_aligned,
    ST_Collect(ST_ConvexHull(aligned) ORDER BY rid) AS aligned
FROM baz
```

Output

```
MULTIPOLYGON(((-107.007 50.2,-107.005 50.2,-107.005 50.198,-107.007 50.198,-107.007 50.2)),((-107.004 50.201,-107.002
    50.201,-107.002 50.198,-107.004 50.198,-107.004 50.201)),((-107.001 50.201,-106.999 50.201,-106.999
    50.198,-107.001 50.198,-107.001 50.201)),((-107.007 50.199,-107.004 50.199,-107.004 50.196,-107.007
    50.196,-107.007 50.199)),((-107.004 50.199,-107.001 50.199,-107.001 50.196,-107.004 50.196,-107.004 50.199))
    ,((-107.001 50.199,-106.999 50.199,-106.999 50.196,-107.001 50.196,-107.001 50.199)),((-107.006 50.197,-107.004
    50.197,-107.004 50.194,-107.006 50.194,-107.006 50.197)),((-107.004 50.197,-107.001 50.197,-107.001
    50.194,-107.004 50.194,-107.004 50.197)),((-107.001 50.197,-106.998 50.197,-106.998 50.195,-107.001
    50.195,-107.001 50.197))) | MULTIPOLYGON(((-107.007 50.2,-107.005 50.2,-107.005 50.198,-107.007 50.198,-107.007
    50.2)),((-107.005 50.202,-107.001 50.202,-107.001 50.198,-107.005 50.198,-107.005 50.202)),((-107.002
    50.202,-106.998 50.202,-106.998 50.198,-107.002 50.198,-107.002 50.202)),((-107.007 50.199,-107.003
    50.199,-107.003 50.195,-107.007 50.195,-107.007 50.199)),((-107.005 50.199,-107.001 50.199,-107.001
    50.195,-107.005 50.195,-107.005 50.199)),((-107.002 50.199,-106.998 50.199,-106.998 50.195,-107.002
    50.195,-107.002 50.199)),((-107.007 50.198,-107.003 50.198,-107.003 50.194,-107.007 50.194,-107.007 50.198))
    ,((-107.005 50.198,-107.001 50.198,-107.001 50.194,-107.005 50.194,-107.005 50.198)),((-107.002 50.198,-106.998
    50.198,-106.998 50.194,-107.002 50.194,-107.002 50.198)))
```

Figure

Se även

ST_Transform, ST_SetSRID

11.8 Rasterbandredigerare

11.8.1 ST_SetBandNoDataValue

ST_SetBandNoDataValue — Ställer in värdet för det angivna bandet som inte representerar några data. Band 1 förutsätts om inget band anges. För att markera ett band som att det inte har något nodata-värde, ställ in nodata-värdet = NULL.

Synopsis

raster **ST_SetBandNoDataValue**(raster rast, double precision nodatavalue);
raster **ST_SetBandNoDataValue**(raster rast, integer band, double precision nodatavalue, boolean forcechecking=false);

Description

Ställer in det värde som representerar inga data för bandet. Band 1 antas om det inte specificeras. Detta påverkar resultaten från ST_Polygon, ST_DumpAsPolygons, och funktionerna ST_PixelAs...().

Exempel

This example changes only the first band NODATA value.

Code

```
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast, 1, 254)
WHERE rid = 2;
```

This example changes the NODATA value for bands 1, 2, and 3.

Code

```
UPDATE dummy_rast
    SET rast =
        ST_SetBandNoDataValue(ST_SetBandNoDataValue(ST_SetBandNoDataValue(rast, 1, 254),
                2, 99),
                3, 108)
        WHERE rid = 2;
```

This example removes the NODATA value so that all pixels are considered by processing functions.

Code

```
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast, 1, NULL)
WHERE rid = 2;
```

Se även

ST_BandNoDataValue, ST_NumBands

11.8.2 ST_SetBandIsNoData

ST_SetBandIsNoData — Ställer in bandets isnodata-flagga till TRUE.

Synopsis

raster **ST_SetBandIsNoData**(raster rast, integer band=1);

Description

Ställer in isnodata-flaggan för bandet till true. Band 1 antas om det inte anges. Denna funktion bör endast anropas när flaggan anses vara smutsig. Det vill säga, när resultatet av anropet ST_BandIsNoData är annorlunda med TRUE som sista argument och utan att använda det

Tillgänglighet: 2.0.0

Exempel

This example creates a dummy table with one raster column. The raster has two bands and one pixel per band. In the first band, the NODATA value and pixel value are both 3. In the second band, the NODATA value is 13 and the pixel value is 4.

Code

```
create table dummy_rast (rid integer, rast raster);

insert into dummy_rast values(
1,
(
'01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0200' -- nBands (uint16 0)
||
'17263529ED684A3F' -- scaleX (float64 0.000805965234044584)
||
'F9253529ED684ABF' -- scaleY (float64 -0.00080596523404458)
||
'1C9F33CE69E352C0' -- ipX (float64 -75.5533328537098)
||
'718F0E9A27A44840' -- ipY (float64 49.2824585505576)
||
'ED50EB853EC32B3F' -- skewX (float64 0.000211812383858707)
||
'7550EB853EC32B3F' -- skewY (float64 0.000211812383858704)
||
'E6100000' -- SRID (int32 4326)
||
'0100' -- width (uint16 1)
||
'0100' -- height (uint16 1)
||
'4' -- hasnodatavalue set to true, isnodata value set to false (when it should be true)
||
'2' -- first band type (4BUI)
||
'03' -- novalue==3
||
'03' -- pixel(0,0)==3 (same that nodata)
||
'0' -- hasnodatavalue set to false
||
'5' -- second band type (16BSI)
||
'0D00' -- novalue==13
||
'0400' -- pixel(0,0)==4
)::raster
);

select ST_BandIsNoData(rast, 1) from dummy_rast where rid = 1; -- Expected false
select ST_BandIsNoData(rast, 1, TRUE) from dummy_rast where rid = 1; -- Expected true
```

The `isnodata` flag is dirty, so set it to true.

Code

```
update dummy_rast set rast = ST_SetBandIsNoData(rast, 1) where rid = 1;


select ST_BandIsNoData(rast, 1) from dummy_rast where rid = 1; -- Expected true
```

Se även

ST_BandNoDataValue, ST_NumBands, ST_SetBandNoDataValue, ST_BandIsNoData

11.8.3 ST_SetBandPath

ST_SetBandPath — Uppdatera den externa sökvägen och bandnumret för ett out-db-band

Synopsis

raster **ST_SetBandPath**(raster rast, integer band, text outdbpath, integer outdbindex, boolean force=false);

Description

Uppdaterar ett out-db-bands externa rasterfilsökväg och externa bandnummer.

Notera
Om `force` är satt till true görs inga tester för att säkerställa kompatibilitet (t.ex. justering, pixelstöd) mellan den externa rasterfilen och PostGIS-rastern. Detta läge är avsett för filsystemändringar där det externa rastret finns.

Tillgänglighet: 2.5.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_AddBand(
        NULL::raster,
        '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif',
        NULL::int[]
    ) AS rast
)
SELECT
    1 AS query,
    bandnum,
    regexp_replace(path, '^.*/', '') AS path,
    outdbbandnum
FROM ST_BandMetaData(
    (SELECT rast FROM foo),
    ARRAY[1, 3, 2]::int[]
) AS metadata
UNION ALL
SELECT
    2, bandnum,
    regexp_replace(path, '^.*/', ''),
    outdbbandnum
FROM ST_BandMetaData((
        SELECT
            ST_SetBandPath(rast,
                2,
```

```
                '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected2.tif ↩
    ',
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1, 3, 2]::int[]
) AS metadata
ORDER BY 1, 2;
```

In the second result set, band 2 now points at `Projected2.tif` and uses out-db band 1.

Output

```
query | bandnum | path           | outdbbandnum
------+---------+----------------+-------------
    1 |       1 | Projected.tif  |            1
    1 |       2 | Projected.tif  |            2
    1 |       3 | Projected.tif  |            3
    2 |       1 | Projected.tif  |            1
    2 |       2 | Projected2.tif |            1
    2 |       3 | Projected.tif  |            3
```

Se även

ST_BandMetaData, ST_SetBandIndex

11.8.4 ST_SetBandIndex

ST_SetBandIndex — Uppdatera det externa bandnumret för ett out-db-band

Synopsis

raster **ST_SetBandIndex**(raster rast, integer band, integer outdbindex, boolean force=false);

Description

Uppdaterar ett out-db-bands externa bandnummer. Detta påverkar inte den externa rasterfil som är associerad med out-db-bandet

Notera
Om `force` är satt till true görs inga tester för att säkerställa kompatibilitet (t.ex. justering, pixelstöd) mellan den externa rasterfilen och PostGIS-rastern. Detta läge är avsett för fall där band flyttas runt i den externa rasterfilen.

Notera
Internt ersätter denna metod PostGIS-rastrets band vid indexbandet med ett nytt band istället för att uppdatera den befintliga sökvägsinformationen.

Tillgänglighet: 2.5.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_AddBand(
        NULL::raster,
        '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif',
        NULL::int[]
    ) AS rast
)
SELECT
    1 AS query,
    bandnum,
    regexp_replace(path, '^.*/', '') AS path,
    outdbbandnum
FROM ST_BandMetaData(
    (SELECT rast FROM foo),
    ARRAY[1, 3, 2]::int[]
) AS metadata
UNION ALL
SELECT
    2, bandnum,
    regexp_replace(path, '^.*/', ''),
    outdbbandnum
FROM ST_BandMetaData((
        SELECT
            ST_SetBandIndex(rast,
                2,
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1, 3, 2]::int[]
) AS metadata
ORDER BY 1, 2;
```

In the second result set, band 2 still points at the same file but now reads external band 1.

Output

```
query | bandnum | path          | outdbbandnum
------+---------+---------------+-------------
    1 |       1 | Projected.tif |            1
    1 |       2 | Projected.tif |            2
    1 |       3 | Projected.tif |            3
    2 |       1 | Projected.tif |            1
    2 |       2 | Projected.tif |            1
    2 |       3 | Projected.tif |            3
```

Se även

ST_BandMetaData, ST_SetBandPath

11.9 Statistik och analys av rasterband

11.9.1 ST_Count

ST_Count — Returnerar antalet pixlar i ett givet band i ett raster eller en rastertäckning. Om inget band anges är standardvärdet band 1. Om exclude_nodata_value är satt till true räknas endast pixlar som inte är lika med nodatavärdet.

Synopsis

bigint **ST_Count**(raster rast, integer nband=1, boolean exclude_nodata_value=true);
bigint **ST_Count**(raster rast, boolean exclude_nodata_value);

Description

Returnerar antalet pixlar i ett givet band i ett raster eller en rastertäckning. Om inget band anges är nband standardvärdet 1.

Notera
Om exclude_nodata_value är satt till true räknas endast pixlar med ett värde som inte är lika med rastrets nodatavärde. Ställ in exclude_nodata_value till false för att räkna alla pixlar

Ändrad: 3.1.0 - ST_Count(rastertable, rastercolumn, ...) varianterna borttagna. Använd ST_CountAgg istället.

Tillgänglighet: 2.0.0

Exempel

Example will count all pixels not 249 and one will count all pixels.

Code

```
SELECT rid, ST_Count(ST_SetBandNoDataValue(rast, 249)) As exclude_nodata,
        ST_Count(ST_SetBandNoDataValue(rast, 249), false) As include_nodata
    FROM dummy_rast WHERE rid=2;
```

Output

```
rid | exclude_nodata | include_nodata
-----+----------------+----------------
   2 |             23 |             25
```

Se även

ST_CountAgg, ST_SummaryStats, ST_SetBandNoDataValue

11.9.2 ST_CountAgg

ST_CountAgg — Aggregera. Returnerar antalet pixlar i ett givet band i en uppsättning raster. Om inget band anges är standardvärdet band 1. Om exclude_nodata_value är satt till true räknas endast pixlar som inte är lika med NODATA-värdet.

Synopsis

bigint **ST_CountAgg**(raster rast, integer nband, boolean exclude_nodata_value, double precision sample_percent);
bigint **ST_CountAgg**(raster rast, integer nband, boolean exclude_nodata_value);
bigint **ST_CountAgg**(raster rast, boolean exclude_nodata_value);

Description

Returnerar antalet pixlar i ett givet band i en uppsättning raster. Om inget band anges är `nband` standardvärdet 1.

Om `exclude_nodata_value` är satt till true räknas endast pixlar med ett värde som inte är lika med `NODATA-värdet` i rastret. Ställ in `exclude_nodata_value` till false för att räkna alla pixlar

Som standard samplas alla pixlar. För att få snabbare svar, ställ in `sample_percent` till ett värde mellan noll (0) och ett (1)

Tillgänglighet: 2.2.0

Exempel

Code

```
WITH foo AS (
    SELECT
        rast.rast
    FROM (
        SELECT ST_SetValue(ST_SetValue(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(10, 10,  ↩
    10, 10, 2, 2, 0, 0, 0),
                          1, '64BF', 0, 0
                      ),
                      1, 1, 1, -10
                  ),
                  1, 5, 4, 0
              ),
              1, 5, 5, 3.14159
        ) AS rast
    ) AS rast
    FULL JOIN (
        SELECT generate_series(1, 10) AS id
    ) AS id
        ON 1 = 1
)
SELECT
    ST_CountAgg(rast, 1, TRUE) AS non_nodata_pixels
FROM foo;
```

Output

```
non_nodata_pixels
------------------
                20
(1 row)
```

Se även

ST_Count, ST_SummaryStats, ST_SetBandNoDataValue

11.9.3 ST_Histogram

ST_Histogram — Returnerar en uppsättning poster som sammanfattar en raster- eller rastertäckningsdatadistribution i separata bin-områden. Antalet bin beräknas automatiskt om det inte anges.

Synopsis

SETOF record **ST_Histogram**(raster rast, integer nband=1, boolean exclude_nodata_value=true, integer bins=autocomputed, double precision[] width=NULL, boolean right=false);
SETOF record **ST_Histogram**(raster rast, integer nband, integer bins, double precision[] width=NULL, boolean right=false);
SETOF record **ST_Histogram**(raster rast, integer nband, boolean exclude_nodata_value, integer bins, boolean right);
SETOF record **ST_Histogram**(raster rast, integer nband, integer bins, boolean right);

Description

Returnerar en uppsättning poster bestående av min, max, count, procent för ett givet rasterband för varje bin. Om inget band anges är `nband` standardvärdet 1.

Notera
Som standard beaktas endast pixelvärden som inte är lika med `nodatavärdet`. Ställ in `exclude_nodata_value` på false för att räkna alla pixlar.

`width` width: en matris som anger bredden på varje kategori/behållare. Om antalet fack är större än antalet bredder upprepas bredderna.

Exempel: 9 fack med bredden [a, b, c] ger utdata som [a, b, c, a, b, c, a, b, c]

`bins` Number of breakouts -- detta är antalet poster som du får tillbaka från funktionen om det anges. Om det inte anges beräknas antalet breakouts automatiskt.

`right` beräkna histogrammet från höger i stället för från vänster (standard). Detta ändrar kriterierna för att utvärdera ett värde x från [a, b] till (a, b]

Ändrad: 3.1.0 Borttagen ST_Histogram(table_name, column_name)-variant.

Tillgänglighet: 2.0.0

Exempel

Single raster tile - compute histograms for bands 1, 2, 3 and autocompute bins.

Code

```
SELECT band, (stats).*
FROM (SELECT rid, band, ST_Histogram(rast, band) As stats
    FROM dummy_rast CROSS JOIN generate_series(1, 3) As band
     WHERE rid=2) As foo;
```

Output

```
band |  min  |  max  | count | percent
------+-------+-------+-------+---------
   1 |   249 |   250 |     2 |    0.08
   1 |   250 |   251 |     2 |    0.08
   1 |   251 |   252 |     1 |    0.04
   1 |   252 |   253 |     2 |    0.08
   1 |   253 |   254 |    18 |    0.72
   2 |    78 | 113.2 |    11 |    0.44
   2 | 113.2 | 148.4 |     4 |    0.16
   2 | 148.4 | 183.6 |     4 |    0.16
   2 | 183.6 | 218.8 |     1 |    0.04
```

```
    2 | 218.8 |   254 |     5 |     0.2
    3 |    62 | 100.4 |    11 |    0.44
    3 | 100.4 | 138.8 |     5 |     0.2
    3 | 138.8 | 177.2 |     4 |    0.16
    3 | 177.2 | 215.6 |     1 |    0.04
    3 | 215.6 |   254 |     4 |    0.16
```

Just band 2 but for 6 bins.

Code

```
SELECT (stats).*
FROM (SELECT rid, ST_Histogram(rast, 2, 6) As stats
    FROM dummy_rast
     WHERE rid=2) As foo;
```

Output

```
min     |    max     | count | percent
------------+------------+-------+---------
         78 | 107.333333 |     9 |    0.36
 107.333333 | 136.666667 |     6 |    0.24
 136.666667 |        166 |     0 |       0
        166 | 195.333333 |     4 |    0.16
 195.333333 | 224.666667 |     1 |    0.04
 224.666667 |        254 |     5 |     0.2
(6 rows)
```

Same as previous but we explicitly control the pixel value range of each bin.

Code

```
SELECT (stats).*
FROM (SELECT rid, ST_Histogram(rast, 2, 6, ARRAY[0.5, 1, 4, 100, 5]) As stats
    FROM dummy_rast
     WHERE rid=2) As foo;
```

Output

```
min  |  max  | count | percent
-------+-------+-------+----------
    78 |  78.5 |     1 |     0.08
  78.5 |  79.5 |     1 |     0.04
  79.5 |  83.5 |     0 |        0
  83.5 | 183.5 |    17 |   0.0068
 183.5 | 188.5 |     0 |        0
 188.5 |   254 |     6 | 0.003664
(6 rows)
```

Se även

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg

11.9.4 ST_Quantile

ST_Quantile — Beräkna kvantiler för ett raster eller en rastertabells täckning i samband med urvalet eller populationen. Ett värde kan alltså undersökas för att ligga vid rastrets 25 %, 50 %, 75% percentil.

Synopsis

SETOF record **ST_Quantile**(raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] quantiles=NULL);
SETOF record **ST_Quantile**(raster rast, double precision[] quantiles);
SETOF record **ST_Quantile**(raster rast, integer nband, double precision[] quantiles);
double precision **ST_Quantile**(raster rast, double precision quantile);
double precision **ST_Quantile**(raster rast, boolean exclude_nodata_value, double precision quantile=NULL);
double precision **ST_Quantile**(raster rast, integer nband, double precision quantile);
double precision **ST_Quantile**(raster rast, integer nband, boolean exclude_nodata_value, double precision quantile);
double precision **ST_Quantile**(raster rast, integer nband, double precision quantile);

Description

Beräkna kvantiler för ett raster eller en rastertabells täckning i samband med urvalet eller populationen. Ett värde kan alltså undersökas för att ligga vid rastrets 25 %, 50 %, 75% percentil.

Notera
Om värdet för `exclude_nodata_value` är satt till false räknas även pixlar utan data.

Ändrad: 3.1.0 Borttagen ST_Quantile(table_name, column_name)-variant.

Tillgänglighet: 2.0.0

Exempel

Example will consider only pixels of band 1 that are not 249 and in named quantiles.

Code

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast, 249) WHERE rid=2;

SELECT (pvq).*
FROM (SELECT ST_Quantile(rast, ARRAY[0.25, 0.75]) As pvq
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvq).quantile;
```

Output

```
quantile | value
----------+-------
    0.25 |   253
    0.75 |   254
```

Code

```
SELECT ST_Quantile(rast, 0.75) As value
FROM dummy_rast WHERE rid=2;
```

Output

```
254
```

Real live example. Quantile of all pixels in band 2 intersecting a geometry.

Code

```
SELECT rid, (ST_Quantile(rast, 2)).* As pvc
    FROM o_4_boston
        WHERE ST_Intersects(
            rast,
            ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706 ↩
    892151,224486 892151))', 26986)
            )
ORDER BY value, quantile, rid
;
```

Output

```
rid | quantile | value
-----+----------+-------
   1 |        0 |     0
   2 |        0 |     0
  14 |        0 |     1
  15 |        0 |     2
  14 |     0.25 |    37
   1 |     0.25 |    42
  15 |     0.25 |    47
   2 |     0.25 |    50
  14 |      0.5 |    56
   1 |      0.5 |    64
  15 |      0.5 |    66
   2 |      0.5 |    77
  14 |     0.75 |    81
  15 |     0.75 |    87
   1 |     0.75 |    94
   2 |     0.75 |   106
  14 |        1 |   199
   1 |        1 |   244
   2 |        1 |   255
  15 |        1 |   255
```

Se även

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg, ST_SetBandNoDataValue

11.9.5 ST_SummaryStats

ST_SummaryStats — Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i ett raster eller en rastertäckning. Band 1 antas om inget band anges.

Synopsis

summarystats **ST_SummaryStats**(raster rast, boolean exclude_nodata_value);
summarystats **ST_SummaryStats**(raster rast, integer nband, boolean exclude_nodata_value);

Description

Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i ett raster eller en rastertäckning. Om inget band anges är `nband` standardvärdet 1.

Notera
Som standard beaktas endast pixelvärden som inte är lika med nodatavärdet. Ställ in exclude_nodata_value på false för att få en räkning av alla pixlar.

Notera
Som standard samplas alla pixlar. För att få snabbare svar, ställ in sample_percent till lägre än 1

Ändrad: 3.1.0 ST_SummaryStats(rastertable, rastercolumn, ...) varianterna har tagits bort. Använd ST_SummaryStatsAgg istället.

Tillgänglighet: 2.0.0

Exempel

Single raster tile.

Code

```
SELECT rid, band, (stats).*
FROM (SELECT rid, band, ST_SummaryStats(rast, band) As stats
    FROM dummy_rast CROSS JOIN generate_series(1, 3) As band
     WHERE rid=2) As foo;
```

Output

```
rid | band | count | sum  |    mean    |  stddev   | min | max
----+------+-------+------+------------+-----------+-----+-----
  2 |    1 |    23 | 5821 | 253.086957 |  1.248061 | 250 | 254
  2 |    2 |    25 | 3682 |     147.28 | 59.862188 |  78 | 254
  2 |    3 |    25 | 3290 |      131.6 | 61.647384 |  62 | 254
```

Summarize pixels that intersect buildings of interest.

This example selects the target buildings, clips band 2 of the raster tiles to each building boundary, computes statistics for each clipped region, and then summarizes those statistics by building. It took 574ms on PostGIS Windows 64-bit with all of Boston Buildings and aerial Tiles (tiles each 150x150 pixels ~ 134,000 tiles), ~102,000 building records.

Code

```
WITH
-- our features of interest
feat AS (
    SELECT gid AS building_id, geom_26986 AS geom
    FROM buildings AS b
    WHERE gid IN (100, 103, 150)
),
-- clip band 2 of raster tiles to boundaries of builds
-- then get stats for these clipped regions
b_stats AS (
    SELECT building_id, (stats).*
    FROM (
        SELECT
            building_id,
            ST_SummaryStats(ST_Clip(rast, 2, geom)) AS stats
        FROM aerials.boston
```

```
        INNER JOIN feat
            ON ST_Intersects(feat.geom, rast)
    ) AS foo
)
-- finally summarize stats
SELECT
    building_id,
    SUM(count) AS num_pixels,
    MIN(min) AS min_pval,
    MAX(max) AS max_pval,
    SUM(mean * count) / SUM(count) AS avg_pval
FROM b_stats
WHERE count
> 0
GROUP BY building_id
ORDER BY building_id;
```

Output

```
building_id | num_pixels | min_pval | max_pval |     avg_pval
------------+------------+----------+----------+------------------
        100 |       1090 |        1 |      255 | 61.0697247706422
        103 |        655 |        7 |      182 | 70.5038167938931
        150 |        895 |        2 |      252 | 185.642458100559
```

Raster coverage.

This example returns statistics for each band.

Code

```
SELECT band, (stats).*
FROM (SELECT band, ST_SummaryStats('o_4_boston', 'rast', band) As stats
    FROM generate_series(1, 3) As band) As foo;
```

Output

```
band |  count  |  sum   |       mean       |      stddev      | min | max
-----+---------+--------+------------------+------------------+-----+-----
   1 | 8450000 | 725799 | 82.7064349112426 | 45.6800222638537 |   0 | 255
   2 | 8450000 | 700487 | 81.4197705325444 | 44.2161184161765 |   0 | 255
   3 | 8450000 | 575943 |  74.682739408284 | 44.2143885481407 |   0 | 255
```

For a table, sampling less than 100% can improve speed. Here the sampling is set to 25%.

Code

```
SELECT band, (stats).*
FROM (SELECT band, ST_SummaryStats('o_4_boston', 'rast', band, true, 0.25) As stats
    FROM generate_series(1, 3) As band) As foo;
```

Output

```
band |  count  |  sum   |       mean       |      stddev      | min | max
-----+---------+--------+------------------+------------------+-----+-----
   1 | 2112500 | 180686 | 82.6890480473373 | 45.6961043857248 |   0 | 255
   2 | 2112500 | 174571 |  81.448503668639 | 44.2252623171821 |   0 | 255
   3 | 2112500 | 144364 | 74.6765884023669 | 44.2014869384578 |   0 | 255
```

Se även

summarystats, ST_SummaryStatsAgg, ST_Count, ST_Clip

11.9.6 ST_SummaryStatsAgg

ST_SummaryStatsAgg — Aggregera. Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i en uppsättning raster. Band 1 antas om inget band anges.

Synopsis

summarystats **ST_SummaryStatsAgg**(raster rast, integer nband, boolean exclude_nodata_value, double precision sample_percent);
summarystats **ST_SummaryStatsAgg**(raster rast, boolean exclude_nodata_value, double precision sample_percent);
summarystats **ST_SummaryStatsAgg**(raster rast, integer nband, boolean exclude_nodata_value);

Description

Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i ett raster eller en rastertäckning. Om inget band anges är `nband` standardvärdet 1.

Notera
Som standard beaktas endast pixelvärden som inte är lika med `NODATA-värdet`. Ställ in `exclude_nodata_value` på False för att få en räkning av alla pixlar.

Notera
Som standard samplas alla pixlar. För att få snabbare svar, ställ in `sample_percent` till ett värde mellan 0 och 1

Tillgänglighet: 2.2.0

Exempel

Code

```
WITH foo AS (
    SELECT
        rast.rast
    FROM (
        SELECT ST_SetValue(ST_SetValue(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(10, 10,  ↩
    10, 10, 2, 2, 0, 0, 0),
                        1, '64BF', 0, 0
                    ),
                    1, 1, 1, -10
                ),
                1, 5, 4, 0
            ),
            1, 5, 5, 3.14159
        ) AS rast
    ) AS rast
    FULL JOIN (
        SELECT generate_series(1, 10) AS id
    ) AS id
        ON 1 = 1
```

```
)
SELECT
    (stats).count AS count,
    round((stats).sum::numeric, 3) AS sum,
    round((stats).mean::numeric, 3) AS mean,
    round((stats).stddev::numeric, 3) AS stddev,
    round((stats).min::numeric, 3) AS min,
    round((stats).max::numeric, 3) AS max
FROM (
    SELECT
        ST_SummaryStatsAgg(rast, 1, TRUE, 1) AS stats
    FROM foo
) bar;
```

Output

```
count |   sum   |  mean  | stddev |   min   |  max
-------+---------+--------+--------+---------+-------
    20 | -68.584 | -3.429 |  6.571 | -10.000 | 3.142
(1 row)
```

Se även

summarystats, ST_SummaryStats, ST_Count, ST_Clip

11.9.7 ST_ValueCount

ST_ValueCount — Returnerar en uppsättning poster som innehåller ett pixelbandvärde och en räkning av antalet pixlar i ett givet band i ett raster (eller en rastertäckning) som har en given uppsättning värden. Om inget band anges är standardvärdet band 1. Som standard räknas inte pixlar med nodata-värden. och alla andra värden i pixeln matas ut och pixelbandvärden avrundas till närmaste heltal.

Synopsis

SETOF record **ST_ValueCount**(raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] searchvalues=NULL, double precision roundto=0, double precision OUT value, integer OUT count);
SETOF record **ST_ValueCount**(raster rast, integer nband, double precision[] searchvalues, double precision roundto=0, double precision OUT value, integer OUT count);
SETOF record **ST_ValueCount**(raster rast, double precision[] searchvalues, double precision roundto=0, double precision OUT value, integer OUT count);
bigint **ST_ValueCount**(raster rast, double precision searchvalue, double precision roundto=0);
bigint **ST_ValueCount**(raster rast, integer nband, boolean exclude_nodata_value, double precision searchvalue, double precision roundto=0);
bigint **ST_ValueCount**(raster rast, integer nband, double precision searchvalue, double precision roundto=0);
SETOF record **ST_ValueCount**(text rastertable, text rastercolumn, integer nband=1, boolean exclude_nodata_value=true, double precision[] searchvalues=NULL, double precision roundto=0, double precision OUT value, integer OUT count);
SETOF record **ST_ValueCount**(text rastertable, text rastercolumn, double precision[] searchvalues, double precision roundto=0, double precision OUT value, integer OUT count);
SETOF record **ST_ValueCount**(text rastertable, text rastercolumn, integer nband, double precision[] searchvalues, double precision roundto=0, double precision OUT value, integer OUT count);
bigint**ST_ValueCount**(text rastertable, text rastercolumn, integer nband, boolean exclude_nodata_value, double precision searchvalue, double precision roundto=0);

bigint **ST_ValueCount**(text rastertable, text rastercolumn, double precision searchvalue, double precision roundto=0);
bigint **ST_ValueCount**(text rastertable, text rastercolumn, integer nband, double precision searchvalue, double precision roundto=0);

Description

Returnerar en uppsättning poster med kolumnerna `value count` som innehåller pixelbandvärdet och antalet pixlar i rasterplattan eller rastertäckningen för det valda bandet.

Om inget band anges är `nband` standardvärdet 1. Om inga `searchvalues` anges returneras alla pixelvärden som finns i rastret eller rastertäckningen. Om ett sökvärde anges, returneras ett heltal i stället för poster som anger antalet pixlar som har det pixelbandvärdet

Notera
Om värdet för `exclude_nodata_value` är satt till false räknas även pixlar utan data.

Tillgänglighet: 2.0.0

Exempel

Example will count only pixels of band 1 that are not 249.

Code

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast, 249) WHERE rid=2;

SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;
```

Output

```
value | count
-------+-------
   250 |     2
   251 |     1
   252 |     2
   253 |     6
   254 |    12
```

Example will coount all pixels of band 1 including 249.

Code

```
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast, 1, false) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;
```

Output

```
value | count
-------+-------
   249 |     2
   250 |     2
   251 |     1
   252 |     2
   253 |     6
   254 |    12
```

Example will count only non-nodata value pixels of band 2.

Code

```
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast, 2) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;
```

Output

```
value | count
-------+-------
    78 |     1
    79 |     1
    88 |     1
    89 |     1
    96 |     1
    97 |     1
    98 |     1
    99 |     2
   112 |     2
:
```

Real live example. Count all the pixels in an aerial raster tile band 2 intersecting a geometry. And return only the pixel band values that have a count > 500.

Code

```
SELECT (pvc).value, SUM((pvc).count) As total
FROM (SELECT ST_ValueCount(rast, 2) As pvc
    FROM o_4_boston
        WHERE ST_Intersects(
            rast,
            ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706 ↩
    892151,224486 892151))', 26986)
             )
        ) As foo
    GROUP BY (pvc).value
    HAVING SUM((pvc).count) > 500
    ORDER BY (pvc).value;
```

Output

```
value | total
-------+-----
    51 | 502
    54 | 521
```

Just return count of pixels in each raster tile that have value of 100 of tiles that intersect a specific geometry.

Code

```
SELECT rid, ST_ValueCount(rast, 2, 100) As count
    FROM o_4_boston
        WHERE ST_Intersects(
            rast,
            ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706 ↩
    892151,224486 892151))', 26986)
             ) ;
```

Output

```
rid | count
-----+-------
   1 |    56
   2 |    95
  14 |    37
  15 |    64
```

Se även

ST_Count, ST_SetBandNoDataValue

11.10 Raster-indata

11.10.1 ST_RastFromWKB

ST_RastFromWKB — Returnera ett rastervärde från ett Well-Known Binary (WKB)-raster.

Synopsis

raster **ST_RastFromWKB**(bytea wkb);

Description

Givet ett Well-Known Binary (WKB) raster, returnera ett raster.

Tillgänglighet: 2.5.0

Exempel

Code

```
SELECT (ST_MetaData(ST_RastFromWKB('\001\000\000\000\000\000\000\000\000\000\000@ ↩
    \000\000\000\000\000\000\010@ ↩
    \000\000\000\000\000\000\340?\000\000\000\000\000\000\340?\000\000\000\000\000\000\000\000\0
    '::bytea
    )
)).* AS metadata;
```

Output

```
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands
-----------+------------+-------+--------+--------+--------+-------+-------+------+----------
        0.5 |        0.5 |    10 |     20 |      2 |      3 |     0 |     0 |   10 |        0
```

Se även

ST_MetaData, ST_RastFromHexWKB, ST_AsBinary/ST_AsWKB, ST_AsHexWKB

11.10.2 ST_RastFromHexWKB

ST_RastFromHexWKB — Returnera ett rastervärde från en Hex-representation av Well-Known Binary (WKB)-raster.

Synopsis

raster **ST_RastFromHexWKB**(text wkb);

Description

Givet ett Well-Known Binary (WKB)-raster i Hex-representation, returnera ett raster.

Tillgänglighet: 2.5.0

Exempel

Code

```
SELECT (ST_MetaData(ST_RastFromHexWKB(' ↩
    010000000000000000000000400000000000000840000000000000 ↩
    E03F000000000000E03F0000000000000000000000000000000A0000000A001400'
    )
)).* AS metadata;
```

Output

```
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands
-----------+------------+-------+--------+--------+--------+-------+-------+------+----------
       0.5 |        0.5 |    10 |     20 |      2 |      3 |     0 |     0 |   10 |        0
```

Se även

ST_MetaData, ST_RastFromWKB, ST_AsBinary/ST_AsWKB, ST_AsHexWKB

11.11 Rasterutdata

11.11.1 ST_AsBinary/ST_AsWKB

ST_AsBinary/ST_AsWKB — Returnerar WKB-representationen (Well-Known Binary) av rastret.

Synopsis

bytea **ST_AsBinary**(raster rast, boolean outasin=FALSE);
bytea **ST_AsWKB**(raster rast, boolean outasin=FALSE);

Description

Returnerar den binära representationen av rastret. Om `outasin` är TRUE behandlas out-db-band som in-db. Se raster/doc/RFC2-WellKnownBinaryFormat som finns i PostGIS källmapp för detaljer om representationen.

Detta är användbart i binära markörer för att hämta data från databasen utan att konvertera dem till en strängrepresentation.

Notera
Som standard innehåller WKB-utdata den externa filsökvägen för out-db-band. Om klienten inte har tillgång till rasterfilen som ligger till grund för ett out-db-band, ska `outasin` sättas till TRUE.

Förbättrad: 2.1.0 Tillägg av `outasin`

Förbättrad: 2.5.0 Tillägg av `ST_AsWKB`

Exempel

Code

```
SELECT ST_AsBinary(rast) AS rastbin FROM dummy_rast WHERE rid=1;
```

Output

```
\001\000\000\000\000\000\000\000\000\000\000@\000\000\000\000\000\000\010@ ↩
     \000\000\000\000\000\000\340?\000\000\000\000\000\000\340?\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
```

Se även

ST_RastFromWKB, ST_AsHexWKB

11.11.2 ST_AsHexWKB

ST_AsHexWKB — Returnerar Well-Known Binary (WKB) i Hex-representation av rastret.

Synopsis

bytea **ST_AsHexWKB**(raster rast, boolean outasin=FALSE);

Description

Returnerar den binära representationen i Hex-representationen av rastret. Om `outasin` är TRUE behandlas out-db-band som in-db. Se raster/doc/RFC2-WellKnownBinaryFormat som finns i PostGIS källmapp för detaljer om representationen.

Notera
Som standard innehåller Hex WKB-utdata den externa filsökvägen för out-db-band. Om klienten inte har tillgång till den rasterfil som ligger till grund för ett out-db-band, ska `outasin` sättas till TRUE.

Tillgänglighet: 2.5.0

Exempel

Code

```
SELECT ST_AsHexWKB(rast) AS rastbin FROM dummy_rast WHERE rid=1;
```

Output

```
01000000000000000000000040000000000000008400000000000000 ↩
    E03F000000000000E03F000000000000000000000000000000000A0000000A001400
```

Se även

ST_RastFromHexWKB, ST_AsBinary/ST_AsWKB

11.11.3 ST_AsGDALRaster

ST_AsGDALRaster — Returnerar rasterplattan i det angivna GDAL Raster-formatet. Rasterformat är ett av de format som stöds av ditt kompilerade bibliotek. Använd ST_GDALDrivers() för att få en lista över de format som stöds av ditt bibliotek.

Synopsis

bytea **ST_AsGDALRaster**(raster rast, text format, text[] options=NULL, integer srid=sameassource);

Description

Returnerar rasterplattan i det angivna formatet. Argumenten specificeras nedan:

- `format` som ska matas ut. Detta är beroende av de drivrutiner som kompileras i ditt libgdal-bibliotek. Generellt tillgängliga är 'JPEG', 'GTIff', 'PNG'. Använd ST_GDALDrivers för att få en lista över format som stöds av ditt bibliotek.
- `options` textuppsättning av GDAL-alternativ. Giltiga alternativ är beroende av formatet. Se alternativ för GDAL Rasterformat för mer information.
- `srid` the spatial reference identifier to embed in the output raster

Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0.

Exempel

Export multiple raster tiles as a single JPEG image with 50 percent quality.

Code

```
SELECT ST_AsGDALRaster(ST_Union(rast), 'JPEG', ARRAY['QUALITY=50']) As rastjpg
FROM dummy_rast
WHERE rast && ST_MakeEnvelope(10, 10, 11, 11);
```

One way to export raster into another format is using PostgreSQL large object export functions. The following repeats the JPEG example and exports the result. Note that this requires superuser access because it uses server-side large object functions. It will also export to path on server network. If you need export locally, use the psql equivalent lo_ functions which export to the local file system instead of the server file system.

Code

```
DROP TABLE IF EXISTS tmp_out ;

CREATE TABLE tmp_out AS
SELECT lo_from_bytea(
       0,
       ST_AsGDALRaster(ST_Union(rast), 'JPEG', ARRAY['QUALITY=50'])
        ) AS loid
  FROM dummy_rast
WHERE rast && ST_MakeEnvelope(10, 10, 11, 11);

SELECT lo_export(loid, '/tmp/dummy.jpg')
   FROM tmp_out;

SELECT lo_unlink(loid)
  FROM tmp_out;
```

Export a raster as a GeoTIFF using the default creation options.

Code

```
SELECT ST_AsGDALRaster(rast, 'GTiff') AS rasttiff
FROM dummy_rast WHERE rid = 2;
```

Export a GeoTIFF using JPEG compression at 90 percent quality and embed SRID 4269.

Code

```
SELECT ST_AsGDALRaster(
  rast,
  'GTiff',
  ARRAY['COMPRESS=JPEG', 'JPEG_QUALITY=90'],
  4269) AS rasttiff
FROM dummy_rast WHERE rid = 2;
```

Se även

Avsnitt 10.4, ST_GDALDrivers, ST_SRID

11.11.4 ST_AsJPEG

ST_AsJPEG — Returnerar de valda banden i rasterplattan som en enda JPEG-bild (byte-array) (Joint Photographic Exports Group). Om inget band anges och 1 eller fler än 3 band, används endast det första bandet. Om endast 3 band anges används alla 3 banden och mappas till RGB.

Synopsis

bytea **ST_AsJPEG**(raster rast, text[] options=NULL);
bytea **ST_AsJPEG**(raster rast, integer nband, integer quality);
bytea **ST_AsJPEG**(raster rast, integer nband, text[] options=NULL);
bytea **ST_AsJPEG**(raster rast, integer[] nbands, text[] options=NULL);
bytea **ST_AsJPEG**(raster rast, integer[] nbands, integer quality);

Description

Returnerar de valda banden i rastret som en enda JPEG-bild (Joint Photographic Exports Group Image). Använd ST_AsGDALRaster om du behöver exportera som mindre vanliga rastertyper. Om inget band anges och 1 eller fler än 3 band, används endast det första bandet. Om 3 band används alla 3 banden. Det finns många varianter av funktionen med många alternativ. Dessa specificeras nedan:

- `nband` är för export av enstaka band.
- `nbands` är en array av band som ska exporteras (notera att max är 3 för JPEG) och ordningen på banden är RGB. t.ex. ARRAY[3,2,1] innebär att band 3 mappas till rött, band 2 till grönt och band 1 till blått
- `kvalitetssiffra` från 0 till 100. Ju högre siffra desto skarpare blir bilden.
- `options` text Mängd GDAL-alternativ enligt definitionen för JPEG (se create_options för JPEG ST_GDALDri För JPEG är giltiga alternativ `PROGRESSIVE` ON eller OFF och `QUALITY` ett intervall från 0 till 100 och standardvärdet är 75. Se GDAL Rasterformatalternativ för mer information.

Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0.

Exempel: Utdata

Output the first three bands at 75 percent quality.

Code

```
SELECT ST_AsJPEG(rast) As rastjpg
FROM dummy_rast WHERE rid=2;
```

Output only the first band at 90 percent quality.

Code

```
SELECT ST_AsJPEG(rast, 1, 90) As rastjpg
FROM dummy_rast WHERE rid=2;
```

Output the first three bands, mapping band 2 to red, band 1 to green, and band 3 to blue, using progressive encoding at 90 percent quality.

Code

```
SELECT ST_AsJPEG(rast, ARRAY[2, 1, 3], ARRAY['QUALITY=90', 'PROGRESSIVE=ON']) As rastjpg
FROM dummy_rast WHERE rid=2;
```

Se även

Avsnitt 10.4, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsTIFF

11.11.5 ST_AsPNG

ST_AsPNG — Returnerar de valda banden i rasterkaklet som en enda PNG-bild (portable network graphics) (byte-array). Om 1, 3 eller 4 band i rastret och inga band anges, används alla band. Om fler än 2 eller fler än 4 band och inga band anges, används endast band 1. Banden mappas till RGB- eller RGBA-rymd.

Synopsis

bytea **ST_AsPNG**(raster rast, text[] options=NULL);
bytea **ST_AsPNG**(raster rast, integer nband, integer compression);
bytea **ST_AsPNG**(raster rast, integer nband, text[] options=NULL);
bytea **ST_AsPNG**(raster rast, integer[] nbands, integer compression);
bytea **ST_AsPNG**(raster rast, integer[] nbands, text[] options=NULL);

Description

Returnerar de valda banden i rastret som en enda PNG-bild (Portable Network Graphics Image). Använd ST_AsGDALRaster om du behöver exportera som mindre vanliga rastertyper. Om inget band anges exporteras de 3 första banden. Det finns många varianter av funktionen med många alternativ. Om ingen `srid` anges används rastrets srid. Dessa specificeras nedan:

- `nband` är för export av enstaka band.
- `nbands` är en array av band som ska exporteras (notera att max är 4 för PNG) och ordningen på banden är RGBA. t.ex. ARRAY[3,2,1] betyder att band 3 mappas till rött, band 2 till grönt och band 1 till blått
- `kompressionsnummer` från 1 till 9. Ju högre siffra desto större kompression.
- `options` text Array av GDAL-alternativ enligt definitionen för PNG (se create_options for PNG på ST_GDALDrivers). För PNG är endast ZLEVEL giltigt (den tid som ska läggas på komprimering - standard 6), t.ex. ARRAY['ZLEVEL=9']. WORLDFILE är inte tillåtet eftersom funktionen skulle behöva mata ut två utdata. Se GDAL Rasterformatalternativ för mer information.

Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0.

Exempel

Export the first 3 bands and map band 3 to red, band 1 to green, and band 2 to blue.

Code

```
SELECT ST_AsPNG(rast) As rastpng
FROM dummy_rast WHERE rid=2;
SELECT ST_AsPNG(rast, ARRAY[3, 1, 2]) As rastpng
FROM dummy_rast WHERE rid=2;
```

Se även

ST_AsGDALRaster, ST_ColorMap, ST_GDALDrivers, Avsnitt 10.4

11.11.6 ST_AsTIFF

ST_AsTIFF — Returnerar de band som valts i rastret som en enda TIFF-bild (byte-array). Om inget band anges eller om något av de angivna banden inte finns i rastret, försöker alla band användas.

Synopsis

bytea **ST_AsTIFF**(raster rast, text[] options=”, integer srid=sameassource);
bytea **ST_AsTIFF**(raster rast, text compression=”, integer srid=sameassource);
bytea **ST_AsTIFF**(raster rast, integer[] nbands, text compression=”, integer srid=sameassource);
bytea **ST_AsTIFF**(raster rast, integer[] nbands, text[] options, integer srid=sameassource);

Description

Returnerar de valda banden i rastret som ett enda TIFF-format (Tagged Image File Format). Om inget band anges försöker alla band användas. Detta är ett omslag runt ST_AsGDALRaster. Använd ST_AsGDALRaster om du behöver exportera som mindre vanliga rastertyper. Det finns många varianter av funktionen med många alternativ. Om det inte finns någon SRS-text för spatial referens används rastrets spatiala referens. Dessa specificeras nedan:

- `nbands` är en array av band som ska exporteras (notera att max är 3 för PNG) och ordningen på banden är RGB. t.ex. ARRAY[3,2,1] innebär att band 3 mappas till rött, band 2 till grönt och band 1 till blått
- `komprimering` Kompressionsuttryck -- JPEG90 (eller någon annan procent), LZW, JPEG, DEFLATE9.
- `options` text Array av GDAL-skaparalternativ enligt definitionen för GTiff (se create_options for GTiff på ST_GDALDrivers). eller se GDAL Raster format options för mer information.
- `srid` srid av spatial_ref_sys för rastret. Detta används för att fylla i georeferensinformationen

Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0.

Exempel: Använd jpeg-komprimering 90%

Code

```
SELECT ST_AsTIFF(rast, 'JPEG90') As rasttiff
FROM dummy_rast WHERE rid=2;
```

Se även

ST_GDALDrivers, ST_AsGDALRaster, ST_SRID

11.12 Rasterbearbetning: Kartalgebra

11.12.1 ST_Clip

ST_Clip — Returnerar det raster som klippts av indatageometrin. Om bandnummer inte anges bearbetas alla band. Om `crop` inte anges eller om TRUE anges, beskärs utdatarastret. Om `touched` är inställt på TRUE inkluderas pixlar som berörs, annars inkluderas endast pixlar vars mittpunkt ligger i geometrin.

Synopsis

raster **ST_Clip**(raster rast, integer[] nband, geometry geom, double precision[] nodataval=NULL, boolean crop=TRUE, boolean touched=FALSE);
raster **ST_Clip**(raster rast, integer nband, geometry geom, double precision nodataval, boolean crop=TRUE, boolean touched=FALSE);
raster **ST_Clip**(raster rast, integer nband, geometry geom, boolean crop, boolean touched=FALSE);
raster **ST_Clip**(raster rast, geometry geom, double precision[] nodataval=NULL, boolean crop=TRUE, boolean touched=FALSE);
raster **ST_Clip**(raster rast, geometry geom, double precision nodataval, boolean crop=TRUE, boolean touched=FALSE);
raster **ST_Clip**(raster rast, geometry geom, boolean crop, boolean touched=FALSE);

Description

Returnerar ett raster som är klippt av indatageometrin `geom`. Om bandindex inte anges bearbetas alla band.

Raster som blir resultatet av ST_Clip måste ha ett nodatavärde tilldelat för områden som klippts, ett för varje band. Om inget anges och indatarastret inte har något nodatavärde definierat, sätts nodatavärdena för det resulterande rastret till ST_MinPossibleValue(ST_BandPixelType(rast, band)). När antalet nodatavärden i matrisen är mindre än antalet band används det sista i matrisen för de återstående banden. Om antalet nodatavärden är större än antalet band ignoreras de extra nodatavärdena. Alla varianter som accepterar en matris med noddatavärden accepterar också ett enda värde som tilldelas varje band.

Om `crop` inte anges antas true, vilket innebär att utdatarastret beskärs till skärningspunkten mellan `geom- och rast-utbredningarna`. Om `crop` är satt till false får det nya rastret samma utsträckning som `rast`. Om `touched` är satt till true väljs alla pixlar i `rast` som skär geometrin.

Notera
Standardinställningen är touched=false, vilket innebär att endast pixlar vars mittpunkt täcks av geometrin väljs.

Förbättrad: 3.5.0 - rörda argument har lagts till.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Omskriven i C

I exemplen här används flygdata från Massachusetts som finns tillgängliga på MassGIS webbplats MassGIS Aerial Orthos.

Exempel

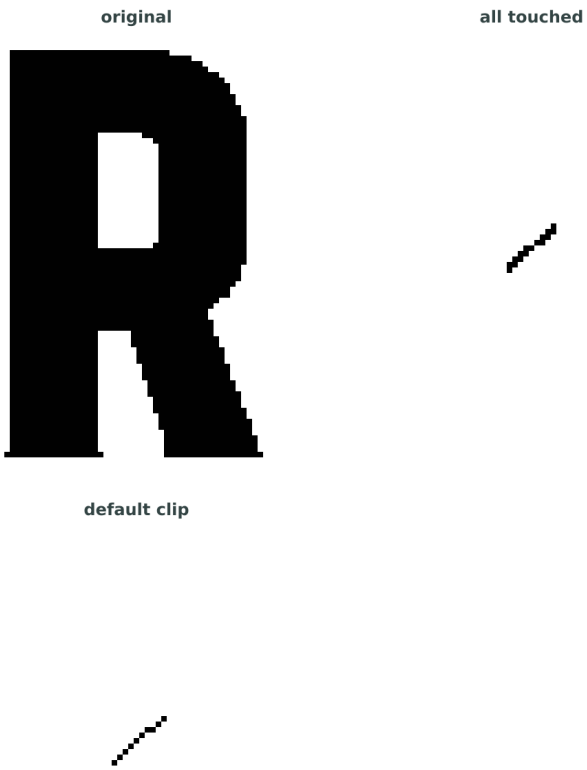
Comparing selecting all touched vs. not all touched.

Code

```
WITH source AS (
    SELECT rast, geom
    FROM ST_AsRaster(ST_Letters('R'), scalex =
> 1.0, scaley =
> -1.0) AS r(rast)
    INNER JOIN ST_GeomFromText('LINESTRING(0 1,5 6,10 10)') AS g(geom)
        ON ST_Intersects(r.rast, g.geom)
```

```
), clipped AS (
    SELECT rast,
        ST_Clip(rast, geom, touched =
> true) AS rast_touched,
        ST_Clip(rast, geom, touched =
> false) AS rast_not_touched
    FROM source
), variants AS (
    SELECT 'original' AS title, ST_Count(rast) AS pixel_count, rast AS rendered
    FROM clipped
    UNION ALL
    SELECT 'all touched', ST_Count(rast_touched), rast_touched
    FROM clipped
    UNION ALL
    SELECT 'default clip', ST_Count(rast_not_touched), rast_not_touched
    FROM clipped
)
SELECT title, pixel_count, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY CASE title
    WHEN 'original' THEN 1
    WHEN 'all touched' THEN 2
    ELSE 3
END;
```

Figure



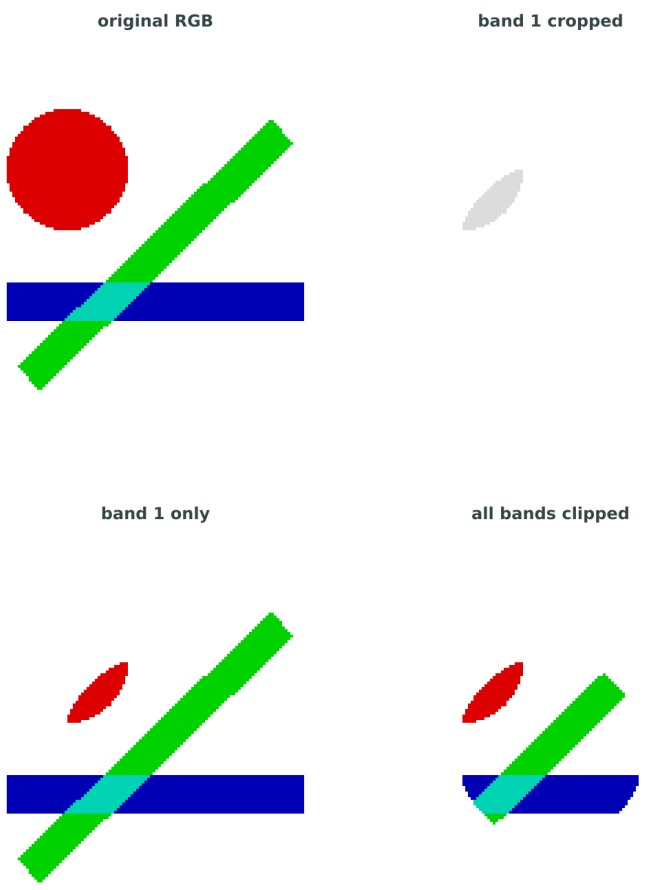
The following self-contained example uses three simple shapes as RGB bands. It compares cropping band 1, clipping band 1 without cropping and then restoring bands 2 and 3, and clipping every band without cropping.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_AddBand(
            ST_AddBand(
                ST_MakeEmptyRaster(128, 128, 0, 128, 1, -1, 0, 0, 0),
                '8BUI'::text, 0::double precision, 0::double precision
            ),
            '8BUI'::text, 0::double precision, 0::double precision
        ),
        '8BUI'::text, 0::double precision, 0::double precision
    ) AS rast
), source AS (
    SELECT ST_SetValue(
        ST_SetValue(
            ST_SetValue(rast, 1, ST_Buffer('POINT(32 96)'::geometry, 22), 220),
            2, ST_Buffer('LINESTRING(18 20,110 110)'::geometry, 6,
                         'endcap=flat join=bevel'), 210
        ),
```

```
        3, ST_Buffer('LINESTRING(10 48,118 48)'::geometry, 7,
                     'endcap=flat'), 180
    ) AS rast,
    ST_Buffer('POINT(64 64)'::geometry, 34) AS clipper
    FROM canvas
), variants AS (
    SELECT 'original RGB' AS title, rast AS rendered FROM source
    UNION ALL
    SELECT 'band 1 cropped', ST_Clip(rast, 1, clipper, true) FROM source
    UNION ALL
    SELECT
        'band 1 only',
        ST_AddBand(
            ST_Clip(rast, 1, clipper, false),
            ARRAY[ST_Band(rast, 2), ST_Band(rast, 3)]
        )
    FROM source
    UNION ALL
    SELECT 'all bands clipped', ST_Clip(rast, clipper, false) FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY CASE title
    WHEN 'original RGB' THEN 1
    WHEN 'band 1 cropped' THEN 2
    WHEN 'band 1 only' THEN 3
    ELSE 4
END;
```

Figure



Crop, single-band clipping, and all-band clipping.

Source tile

Cropped

One-band source

One-band clip

All-band source

All-band clip

Se även

ST_AddBand, ST_Count, ST_MapAlgebra (callback function version), ST_Intersection

11.12.2 ST_ColorMap

ST_ColorMap — Skapar ett nytt raster med upp till fyra 8BUI-band (gråskala, RGB, RGBA) från källrastret och ett angivet band. Band 1 antas om det inte specificeras.

Synopsis

raster **ST_ColorMap**(raster rast, integer nband=1, text colormap=grayscale, text method=INTERPOLATE);
raster **ST_ColorMap**(raster rast, text colormap, text method=INTERPOLATE);

Description

Tillämpa en `färgkarta` på bandet vid `nband` i `rast`, vilket resulterar i ett nytt raster som består av upp till fyra 8BUI-band. Antalet 8BUI-band i det nya rastret bestäms av antalet färgkomponenter som definieras i `colormap`.

Om `nband` inte anges antas band 1.

`colormap` kan vara ett nyckelord för en fördefinierad colormap eller en uppsättning linjer som definierar värdet och färgkomponenterna.

Giltigt fördefinierat nyckelord för `färgkartor`:

- `gråskala` eller `gråskala` för ett 8BUI-bandraster av nyanser av grått.
- `pseudofärg` för ett raster med fyra 8BUI (RGBA)-band med färger från blått till grönt till rött.
- `brand` för ett raster med fyra 8BUI (RGBA)-band med färger från svart till rött till ljusgult.
- `bluered` för ett raster med fyra 8BUI (RGBA)-band med färger från blått till ljusvitt till rött.

Användare kan skicka en uppsättning poster (en per rad) till `colormap` för att ange anpassade färgkartor. Varje post består i allmänhet av fem värden: pixelvärdet och motsvarande röd-, grön-, blå- och alfakomponenter (färgkomponenter mellan 0 och 255). Procentvärden kan användas i stället för pixelvärden, där 0% aoch 100% aär de lägsta och högsta värdena i rasterbandet. Värdena kan separeras med kommatecken (’,’), tabbar, kolon (’:’) och/eller mellanslag. Pixelvärdet kan sättas till *nv*, *null* eller *nodata* för NODATA-värdet. Ett exempel ges nedan.

Code

```
5 0 0 0 255
4 100:50 55 255
1 150, 100 150 255
0% 255 255 255 255
nv 0 0 0 0
```

Syntaxen för `colormap` liknar den för färgreliefläget i GDAL gdaldem.

Giltiga nyckelord för `metod`:

- `INTERPOLATE` för att använda linjär interpolation för att mjukt blanda färgerna mellan de angivna pixelvärdena
- `EXACT` för att strikt matcha endast de pixelvärden som finns i färgkartan. Pixlar vars värde inte matchar en post i färgkartan kommer att sättas till 0 0 0 0 0 (RGBA)
- `NEAREST` för att använda den färgkartspost vars värde ligger närmast pixelvärdet

Notera
En bra referens för färgkartor är ColorBrewer.

Varning
De resulterande banden av nytt raster kommer inte att ha något NODATA-värde inställt. Använd ST_SetBandNoDataValue för att ställa in ett NODATA-värde om ett sådant behövs.

Tillgänglighet: 2.1.0

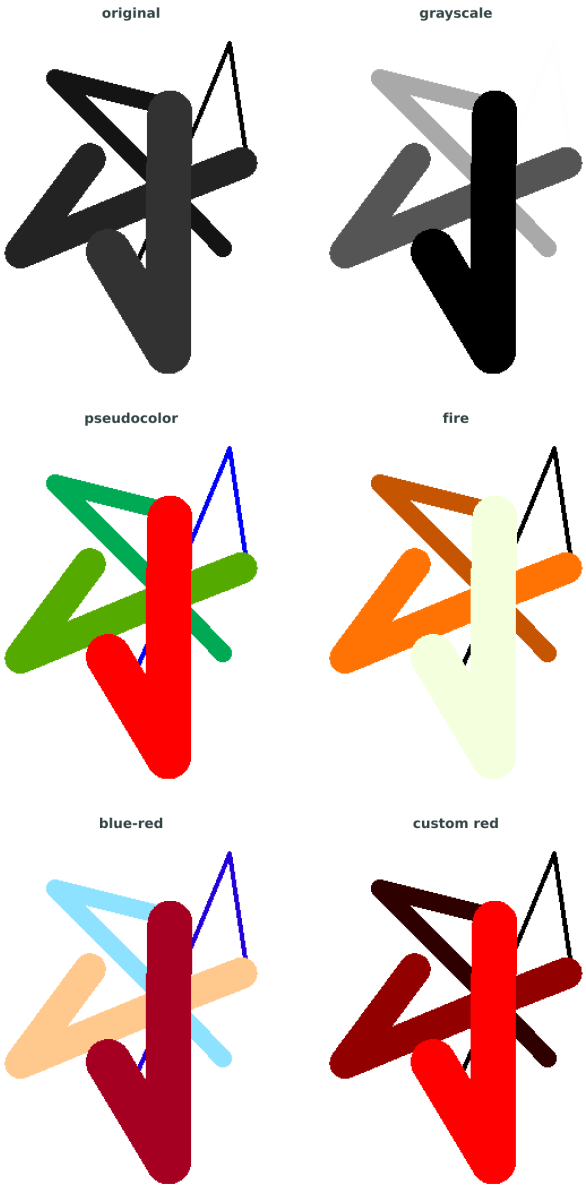
Exempel

This example builds a raster from rotated buffered lines and compares the original values with predefined and custom color maps.

Code

```
WITH ref AS (
    SELECT ST_MakeEmptyRaster(200, 200, 0, 200, 1, -1, 0, 0) AS rast
), source AS (
    SELECT ST_Union(rast) AS rast
    FROM (
    SELECT
        ST_AsRaster(ST_Rotate(ST_Buffer('LINESTRING(0 2,50 50,150 150,125 50)'::geometry,
                    i*2
                ),
                pi() * i * 0.125, ST_Point(50, 50)
            ),
            ref.rast, '8BUI'::text, i * 5
        ) AS rast
    FROM ref
    CROSS JOIN generate_series(1, 10, 3) AS i
) AS shapes
)
SELECT title, ST_AsPNG(rendered) AS image
FROM source
CROSS JOIN LATERAL (VALUES
    ('original', rast),
    ('grayscale', ST_ColorMap(rast, 1, 'greyscale')),
    ('pseudocolor', ST_ColorMap(rast, 1, 'pseudocolor')),
    ('fire', ST_ColorMap(rast, 1, 'fire')),
    ('blue-red', ST_ColorMap(rast, 1, 'bluered')),
    ('custom red', ST_ColorMap(
        rast,
        1,
        E'100% 255 0 0 255\n80% 160 0 0 255\n50% 130 0 0 255\n' ||
        E'30% 30 0 0 255\n20% 60 0 0 255\n0% 0 0 0 255\n' ||
        E'nv 255 255 255 0'
    ))
) AS variants(title, rendered);
```

Figure



Se även

ST_AsPNG, ST_AsRaster ST_MapAlgebra (callback function version), ST_Grayscale ST_NumBands, ST_Reclass, ST_SetBandNoDataValue, ST_Union

11.12.3 ST_Grayscale

ST_Grayscale — Skapar ett nytt bandraster med ett 8BUI-band från källrastret och angivna band som representerar rött, grönt och blått

Synopsis

(1) raster **ST_Grayscale**(raster rast, integer redband=1, integer greenband=2, integer blueband=3, text extenttype=INTERSECTION);

(2) raster **ST_Grayscale**(rastbandarg[] rastbandargset, text extenttype=INTERSECTION);

Description

Skapa ett raster med ett 8BUI-band givet tre indataband (från ett eller flera raster). Alla indataband vars pixeltyp inte är 8BUI kommer att omklassificeras med hjälp av ST_Reclass.

Notera
Den här funktionen är inte som ST_ColorMap med nyckelordet `grayscale` eftersom ST_ColorMap bara använder ett band medan den här funktionen förväntar sig tre band för RGB. Denna funktion tillämpar följande ekvation för att konvertera RGB till gråskala: 0.2989 * RÖD + 0,5870 * GRÖN + 0,1140 * BLÅ

Tillgänglighet: 2.5.0

Exempel

This self-contained example paints an apple with gradients and highlights into three raster bands. It compares the original RGB raster with the equivalent grayscale results produced by both function variants.

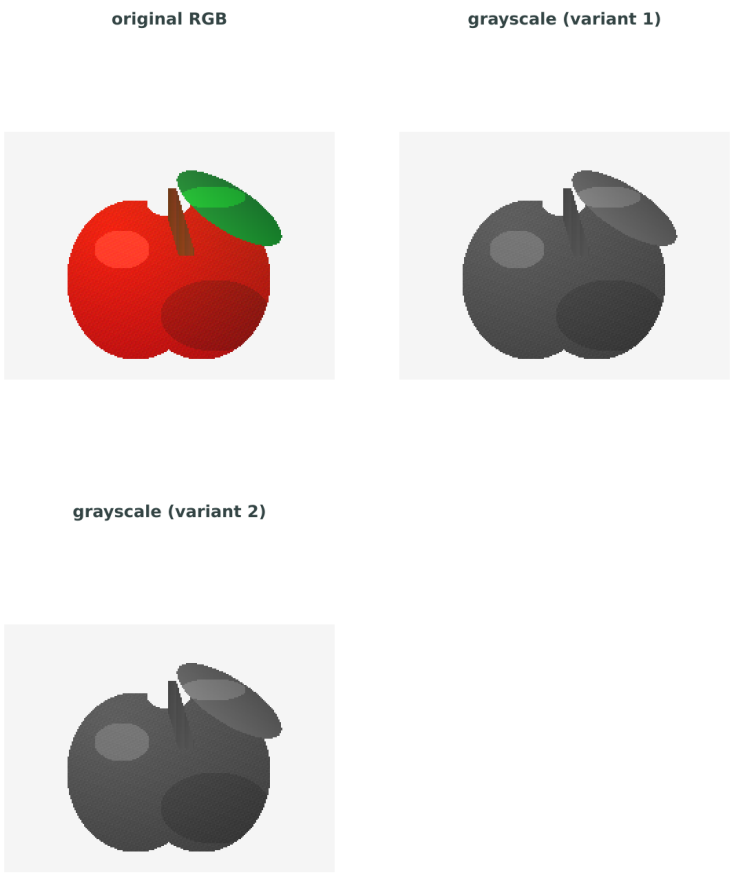
Code

```
WITH pixels AS (
    SELECT
        col,
        row,
        x,
        y,
        (((x - 48.0)^2 / (25.0^2) + (y - 54.0)^2 / (29.0^2) < 1)
         OR ((x - 72.0)^2 / (25.0^2) + (y - 54.0)^2 / (29.0^2) < 1))
         AND NOT ((x - 60.0)^2 / (8.0^2) + (y - 26.0)^2 / (5.0^2) < 1) AS fruit,
        ((x - 82.0) * 0.84 + (y - 28.0) * 0.54)^2 / (22.0^2)
          + (-(x - 82.0) * 0.54 + (y - 28.0) * 0.84)^2 / (8.0^2) < 1 AS leaf,
        abs((x - 62.0) * 0.96 - (y - 29.0) * 0.28) < 3
          AND x BETWEEN 60 AND 69 AND y BETWEEN 21 AND 45 AS stem
    FROM (
        SELECT
            col,
            row,
            col / 2.0 AS x,
            row / 2.0 AS y
        FROM generate_series(1, 180) AS row
        CROSS JOIN generate_series(1, 240) AS col
    ) AS source_pixels
), shaded AS (
    SELECT
        col,
        row,
        x,
        y,
        CASE
```

```
            WHEN stem THEN 95 + y * 0.7 + (x % 3) * 8
            WHEN leaf THEN 28 + (90 - x) * 0.5
            WHEN fruit THEN LEAST(255, GREATEST(0,
                150
                + (85 - sqrt((x - 39.0)^2 + (y - 37.0)^2)) * 1.1
                + CASE
                    WHEN (x - 43.0)^2 / (10.0^2) + (y - 43.0)^2 / (7.0^2) < 1
                    THEN 35
                    ELSE 0
                END
                - CASE
                    WHEN (x - 77.0)^2 / (20.0^2) + (y - 67.0)^2 / (13.0^2) < 1
                    THEN 35
                    ELSE 0
                END
                + ((x * 13 + y * 7) % 19 - 9)
            ))
            ELSE 245
        END AS red,
        CASE
            WHEN stem THEN 50 + y * 0.35
            WHEN leaf THEN LEAST(190, GREATEST(40,
                105 + (95 - x) * 1.2 + (y - 20) * 1.5
                + CASE
                    WHEN (x - 76.0)^2 / (12.0^2) + (y - 24.0)^2 / (4.0^2) < 1
                    THEN 45
                    ELSE 0
                END
            ))
            WHEN fruit THEN LEAST(95, GREATEST(15,
                24 + (68 - y) * 0.35
                + CASE
                    WHEN (x - 43.0)^2 / (10.0^2) + (y - 43.0)^2 / (7.0^2) < 1
                    THEN 30
                    ELSE 0
                END
                + ((x * 5 + y * 11) % 13 - 6)
            ))
            ELSE 245
        END AS green,
        CASE
            WHEN stem THEN 22 + (x % 4) * 3
            WHEN leaf THEN 42 + (100 - x) * 0.4
            WHEN fruit THEN LEAST(75, GREATEST(10,
                18
                + CASE
                    WHEN (x - 43.0)^2 / (10.0^2) + (y - 43.0)^2 / (7.0^2) < 1
                    THEN 25
                    ELSE 0
                END
                + ((x * 3 + y * 5) % 11 - 5)
            ))
            ELSE 245
        END AS blue
    FROM pixels
), rows AS (
    SELECT
        row,
        array_agg(red::double precision ORDER BY col) AS red_values,
        array_agg(green::double precision ORDER BY col) AS green_values,
        array_agg(blue::double precision ORDER BY col) AS blue_values
    FROM shaded
```

```
    GROUP BY row
), source AS (
    SELECT ST_SetValues(ST_SetValues(ST_SetValues(
        ST_AddBand(ST_AddBand(ST_AddBand(
            ST_MakeEmptyRaster(240, 180, 0, 180, 1, -1, 0, 0, 0),
            '8BUI'::text, 245::double precision, 0::double precision),
            '8BUI'::text, 245::double precision, 0::double precision),
            '8BUI'::text, 245::double precision, 0::double precision),
        1, 1, 1, array_agg(red_values ORDER BY row)::double precision[][]),
        2, 1, 1, array_agg(green_values ORDER BY row)::double precision[][]),
        3, 1, 1, array_agg(blue_values ORDER BY row)::double precision[][]) AS rast
    FROM rows
)
SELECT ST_AsPNG(rast) AS "original RGB",
    ST_AsPNG(ST_Grayscale(rast)) AS "grayscale (variant 1)",
    ST_AsPNG(ST_Grayscale(ARRAY[
        ROW(rast, 1)::rastbandarg,
        ROW(rast, 2)::rastbandarg,
        ROW(rast, 3)::rastbandarg
    ]::rastbandarg[])) AS "grayscale (variant 2)"
FROM source;
```

Figure



Original photographic illustration:

original apple

grayscale apple

Se även

ST_AsPNG, ST_Reclass, ST_ColorMap

11.12.4 ST_Intersection

ST_Intersection — Returnerar ett raster eller en uppsättning geometri-pixelvärdespar som representerar den delade delen av två raster eller den geometriska skärningspunkten mellan en vektorisering av rastret och en geometri.

Synopsis

setof geomval **ST_Intersection**(geometry geom, raster rast, integer band_num=1);
setof geomval **ST_Intersection**(raster rast, geometry geom);
setof geomval **ST_Intersection**(raster rast, integer band, geometry geomin);
raster **ST_Intersection**(raster rast1, raster rast2, double precision[] nodataval);
raster **ST_Intersection**(raster rast1, raster rast2, text returnband, double precision[] nodataval);
raster **ST_Intersection**(raster rast1, integer band1, raster rast2, integer band2, double precision[] nodataval);
raster **ST_Intersection**(raster rast1, integer band1, raster rast2, integer band2, text returnband, double precision[] nodataval);

Description

Returnerar ett raster eller en uppsättning geometri-pixelvärdespar som representerar den delade delen av två raster eller den geometriska skärningspunkten mellan en vektorisering av rastret och en geometri.

De tre första varianterna, som returnerar en uppsättning geomval, fungerar i vektorrymd. Rastret vektoriseras först (med hjälp av ST_DumpAsPolygons) till en uppsättning geomval-rader och dessa rader korsas sedan med geometrin med hjälp av PostGIS-funktionen ST_Intersection (geometry, geometry). Geometrier som endast skär ett noddatavärdesområde i ett raster returnerar en tom geometri. De utesluts normalt från resultaten genom korrekt användning av ST_Intersects i WHERE-satsen.

Du kan komma åt geometri- och värdedelarna i den resulterande uppsättningen geomval genom att omge dem med parenteser och lägga till ’.geom’ eller ’.val’ i slutet av uttrycket. t.ex. (ST_Intersection(rast, geom)).geom

De andra varianterna, som returnerar ett raster, fungerar i rasterutrymme. De använder två raster-versionen av ST_MapAlgebraExpr för att utföra korsningen.

Det resulterande rastrets utsträckning motsvarar den geometriska skärningspunkten mellan de två rasterutsträckningarna. Det resulterande rastret innehåller BAND1", BAND2"eller BOTHband, enligt vad som skickas som `returbandparameter`. Nodatavärdesområden som finns i något band resulterar i nodatavärdesområden i alla band i resultatet. Med andra ord blir varje pixel som korsas av en nodatavärdespixel en nodatavärdespixel i resultatet.

Raster som blir resultatet av ST_Intersection måste ha ett nodatavärde tilldelat för områden som inte korsar varandra. Du kan definiera eller ersätta nodata-värdet för alla resulterande band genom att tillhandahålla en `nodataval[]` -array med ett eller två nodata-värden beroende på om du begär BAND1-, BAND2- eller BÅDA-band. Det första värdet i matrisen ersätter nodatavärdet i det första bandet och det andra värdet ersätter nodatavärdet i det andra bandet. Om ett inmatningsband inte har något nodatavärde definierat och inga sådana tillhandahålls som en matris, väljs ett med hjälp av funktionen ST_MinPossibleValue. Alla varianter som accepterar en matris av nodatavärden kan också acceptera ett enda värde som kommer att tilldelas varje efterfrågat band.

I alla varianter, om inget bandnummer anges, antas band 1. Om du behöver en skärningspunkt mellan ett raster och en geometri som returnerar ett raster, se ST_Clip.

Notera
For more control over the result extent or NODATA handling, use the two-raster variant of ST_MapAlgebraExpr. To compute a raster-space intersection with a geometry, use ST_Clip. Use this function together with ST_Intersects and an index on the raster or geometry column when filtering table data.

Förbättrad: 2.0.0 - Intersektion i rasterrymden infördes. I tidigare versioner före 2.0.0 stöddes endast intersektion som utfördes i vektorrymd.

Exempel på detta: Geometri, Raster -- resulterar i geometri vals

Code

```
SELECT
    foo.rid,
    foo.gid,
    GeometryType((foo.geomval).geom) AS geom_type,
    (foo.geomval).val AS value
FROM (
    SELECT
        A.rid,
        g.gid,
        ST_Intersection(A.rast, g.geom) As geomval
    FROM dummy_rast AS A
    CROSS JOIN (
        VALUES
            (1, ST_Point(3427928, 5793243.85) ),
            (2, 'LINESTRING(3427927.85 5793243.75,3427927.8 5793243.75,3427927.8 5793243.8) ↩
    '::geometry),
            (3, 'LINESTRING(1 2,3 4)'::geometry)
    ) As g(gid, geom)
    WHERE A.rid = 2
) AS foo;
```

Output

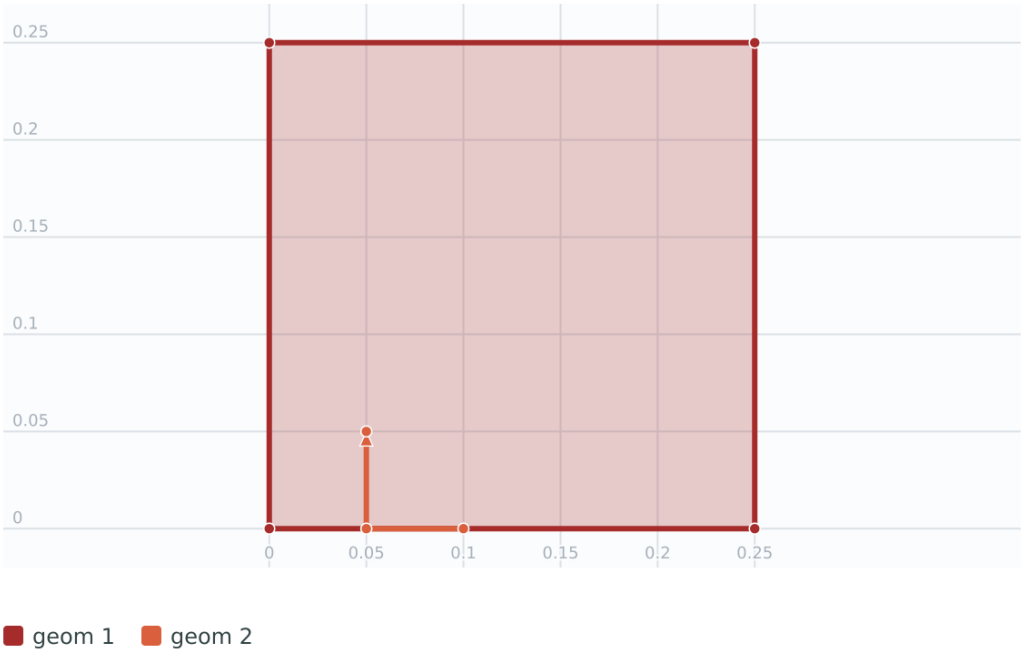
```
rid | gid |      geom_type       | value
----+-----+----------------------+-------
  2 |   1 | POINT                |   249
  2 |   1 | POINT                |   253
  2 |   2 | POINT                |   254
  2 |   2 | POINT                |   251
  2 |   2 | POINT                |   253
  2 |   2 | LINESTRING           |   252
  2 |   2 | MULTILINESTRING      |   250
  2 |   3 | GEOMETRYCOLLECTION   |
```

The same line example rendered in a common coordinate frame. The raster footprint and complete input line remain visible behind the returned intersection pieces.

Code

```
WITH source AS (
    SELECT
        rast,
        ST_GeomFromText(
            'LINESTRING(3427927.85 5793243.75,'
            '3427927.8 5793243.75,3427927.8 5793243.8)'
        ) AS input_geometry
    FROM dummy_rast
    WHERE rid = 2
), intersection AS (
SELECT
    rast,
    input_geometry,
    ST_Collect(gv.geom) AS result_geometry
FROM source
CROSS JOIN LATERAL ST_Intersection(rast, input_geometry) AS gv
GROUP BY rast, input_geometry
), layers AS (
    SELECT
        'raster' AS title,
        ST_Translate(ST_Envelope(rast), -3427927.75, -5793243.75) AS geom
    FROM intersection
    UNION ALL
    SELECT
        'input',
        ST_Translate(input_geometry, -3427927.75, -5793243.75)
    FROM intersection
    UNION ALL
    SELECT
        'intersection',
        ST_Translate(result_geometry, -3427927.75, -5793243.75)
    FROM intersection
)
SELECT title, geom AS geom
FROM layers
ORDER BY CASE title
    WHEN 'raster' THEN 1
    WHEN 'input' THEN 2
    ELSE 3
END;
```

Figure



Se även

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText

11.12.5 ST_MapAlgebra (callback function version)

ST_MapAlgebra (callback function version) — Callback function version - Returnerar ett enbandsraster med ett eller flera indataraster, bandindex och en användarspecificerad callback-funktion.

Synopsis

raster **ST_MapAlgebra**(rastbandarg[] rastbandargset, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=INTERSECTION, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast, integer[] nband, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=FIRST, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast, integer nband, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=FIRST, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast1, integer nband1, raster rast2, integer nband2, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=INTERSECTION, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast, integer nband, regprocedure callbackfunc, float8[] mask, boolean weighted, text pixeltype=NULL, text extenttype=INTERSECTION, raster customextent=NULL, text[] VARIADIC userargs=NULL);

Description

Returnerar ett enbandsraster med ett eller flera indataraster, bandindex och en användarspecificerad återkallningsfunktion.

rast,rast1,rast2, rastbandargset Raster på vilka kartalgebraprocessen utvärderas.

`rastbandargset` gör det möjligt att använda en map algebra-operation på många raster och/eller många band. Se exempel Variant 1.

nband, nband1, nband2 Bandnummer för det raster som ska utvärderas. nband kan vara ett heltal eller ett heltal[] som anger banden. nband1 är band på rast1 och nband2 är band på rast2 för fallet 2 raster/2 band.

callbackfunc Parametern `callbackfunc` måste vara namnet och signaturen för en SQL- eller PL/pgSQL-funktion, som kastas till en regprocedure. Ett exempel på PL/pgSQL-funktionsexempel är:

Code

```
CREATE OR REPLACE FUNCTION sample_callbackfunc(value double precision[][][], pos ↩
    integer[][], VARIADIC userargs text[])
    RETURNS double precision
    AS $$
    BEGIN
        RETURN 0;
    END;
    $$ LANGUAGE 'plpgsql' IMMUTABLE;
```

The `callbackfunc` must have two or three arguments: a 3-dimension double precision array, a 2-dimension integer array and, optionally, a variadic 1-dimension text array. The first argument `value` is the set of values (as double precision) from all input rasters. The three dimensions (where indexes are 1-based) are: raster #, row y, column x. The second argument `position` is the set of pixel positions from the output raster and input rasters. The outer dimension (where indexes are 0-based) is the raster #. The position at outer dimension index 0 is the output raster's pixel position. For each outer dimension, there are two elements in the inner dimension for X and Y. The optional third argument `userargs` is for passing through any user-specified arguments.

The callback return value is used as the value of the output pixel. If the callback returns `NULL`, the output pixel is set to NODATA when the output band has a NODATA value. If the output band has no NODATA value, a `NULL` callback result raises an error.

För att skicka ett regprocedure-argument till en SQL-funktion krävs att hela funktionssignaturen skickas och sedan kastas till en regprocedure-typ. För att skicka ovanstående exempel på PL / pgSQL-funktion som ett argument är SQL för argumentet:

Code

```
'sample_callbackfunc(double precision[], integer[], text[])'::regprocedure
```

Note that PostgreSQL array types do not encode dimensions in function signatures, so multidimensional callback arguments are written as double precision[] and integer[] in the regprocedure signature.

mask En n-dimensionell array (matris) med tal som används för att filtrera vilka celler som skickas till map algebra call-back-funktionen. 0 betyder att ett granncellsvärde ska behandlas som ingen data och 1 betyder att värdet ska behandlas som data. Om weight är satt till true används värdena som multiplikatorer för att multiplicera pixelvärdet för det värdet i grannskapspositionen.

weighted boolean (true/false) för att ange om ett maskvärde ska viktas (multipliceras med originalvärdet) eller inte (gäller endast proto som tar en mask).

pixeltype Om `pixeltype` skickas in kommer det ena bandet i det nya rastret att ha den pixeltypen. Om pixeltype anges som NULL eller utelämnas kommer det nya rasterbandet att ha samma pixeltype som det angivna bandet i det första rastret (för omfattningstyper: INTERSECTION, UNION, FIRST, CUSTOM) eller det angivna bandet i det lämpliga rastret (för omfattningstyper: SECOND, LAST). Om du är osäker, ange alltid `pixeltype`.

Den resulterande pixeltypen för utdatarastret måste vara en som listas i ST_BandPixelType eller utelämnas eller sättas till NULL.

extenttype Möjliga värden är INTERSECTION (standard), UNION, FIRST (standard för en rastervariant), SECOND (andra), LAST (sista), CUSTOM (anpassad).

customextent Om `extentype` är CUSTOM måste ett raster tillhandahållas för `customextent`. Se exempel 4 i Variant 1.

distancex Avståndet i pixlar från referenscellen i x-riktningen. Bredden på den resulterande matrisen skulle alltså vara `2*avståndx + 1`. Om inget anges beaktas endast referenscellen (grannskap 0).

distancey Avståndet i pixlar från referenscellen i y-riktningen. Höjden på den resulterande matrisen blir `2*avståndety + 1`. Om inget anges beaktas endast referenscellen (grannskapet 0).

userargs If the `callbackfunc` declares a third argument, that argument is a variadic text array. All trailing text arguments are passed through to the specified `callbackfunc`, and are contained in the `userargs` argument.

Notera
För mer information om VARIADIC-nyckelordet, se PostgreSQL-dokumentationen och avsnittet SQL-funktioner med variabelt antal argument”i Query Language (SQL) -funktioner.

Variant 1 accepterar en array `av rastbandarg` som gör det möjligt att använda en map algebra-operation på många raster och/eller många band. Se exempel Variant 1.

Varianterna 2 och 3 arbetar med ett eller flera band i ett raster. Se exempel Variant 2 och 3.

Variant 4 arbetar med två raster med ett band per raster. Se exempel Variant 4.

Tillgänglighet: 2.2.0: Möjlighet att lägga till en mask

Tillgänglighet: 2.1.0

Exempel

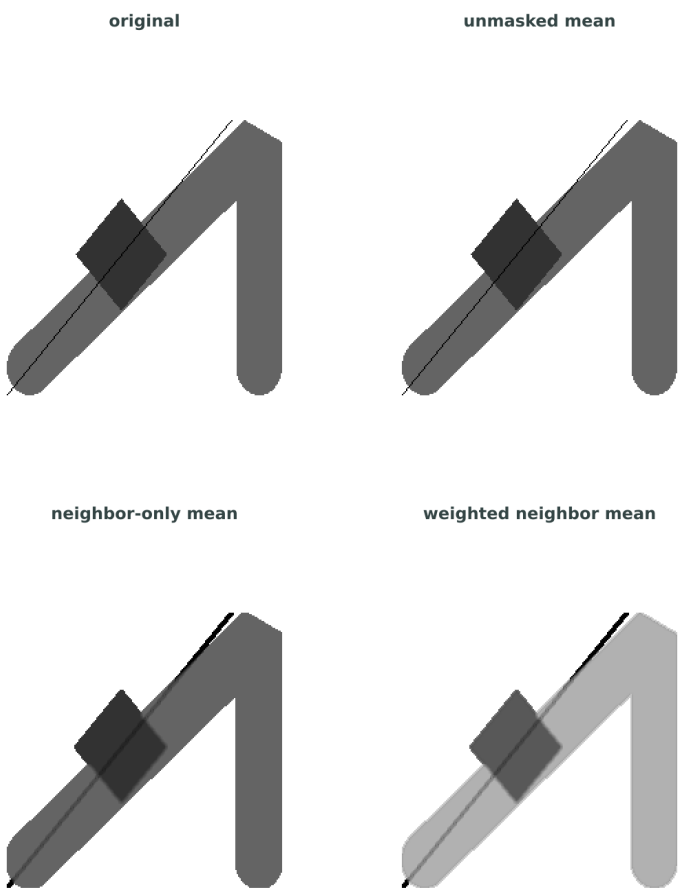
Compare an unmasked mean with neighbor-only and weighted-neighbor masks. The weighted mask multiplies neighboring values by 2 before computing the result.

Code

```
WITH foo AS (
    SELECT ST_SetBandNoDataValue(
        ST_SetValue(
            ST_SetValue(
                ST_AsRaster(
                    ST_Buffer('LINESTRING(50 50,100 90,100 50)'::geometry, 5, 'join=bevel') ↩
    ,
                    200, 200, ARRAY['8BUI'], ARRAY[100], ARRAY[0]
                ),
                ST_Buffer('POINT(70 70)'::geometry, 10, 'quad_segs=1'),
                50
            ),
            'LINESTRING(20 20,100 100,150 98)'::geometry,
```

```
            1
        ),
        0
    ) AS rast
), variants AS (
    SELECT 'original' AS title, rast
    FROM foo
    UNION ALL
    SELECT 'unmasked mean' AS title,
        ST_MapAlgebra(rast, 1, 'ST_Mean4ma(double precision[], int[], text[])':: ↩
    regprocedure) AS rast
    FROM foo
    UNION ALL
    SELECT 'neighbor-only mean' AS title,
        ST_MapAlgebra(
            rast,
            1,
            'ST_Mean4ma(double precision[], int[], text[])'::regprocedure,
            '{{1,1,1}, {1,0,1}, {1,1,1}}'::double precision[],
            false
        ) AS rast
    FROM foo
    UNION ALL
    SELECT 'weighted neighbor mean' AS title,
        ST_MapAlgebra(
            rast,
            1,
            'ST_Mean4ma(double precision[], int[], text[])'::regprocedure,
            '{{2,2,2}, {2,0,2}, {2,2,2}}'::double precision[],
            true
        ) AS rast
    FROM foo
)
SELECT title, ST_AsPNG(rast) AS image
FROM variants;
```

Figure



Variant 1.

Ett raster, ett band

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16BUI', ↩
     1, 0) AS rast
)
SELECT
    ST_MapAlgebra(ARRAY[ROW(rast, 1)]::rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

Ett raster, flera band

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast
)
```

```
SELECT
    ST_MapAlgebra(ARRAY[ROW(rast, 3), ROW(rast, 1), ROW(rast, 3), ROW(rast, 2)]:: ↩
    rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

Flera raster, flera band

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast UNION ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 2, 0), 2, '8BUI', 20, 0), 3, '32BUI', 300, 0) AS rast
)
SELECT
    ST_MapAlgebra(ARRAY[ROW(t1.rast, 3), ROW(t2.rast, 1), ROW(t2.rast, 3), ROW(t1.rast, 2) ↩
    ]::rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 1
    AND t2.rid = 2
```

Complete example of tiles of a coverage with neighborhood.

Code

```
WITH foo AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16BUI', ↩
     1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, 0, 1, -1, 0, 0, 0), 1, '16BUI', 2, 0)  ↩
    AS rast UNION ALL
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, 0, 1, -1, 0, 0, 0), 1, '16BUI', 3, 0)  ↩
    AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -2, 1, -1, 0, 0, 0), 1, '16BUI', 10,  ↩
    0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -2, 1, -1, 0, 0, 0), 1, '16BUI', 20,  ↩
    0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -2, 1, -1, 0, 0, 0), 1, '16BUI', 30,  ↩
    0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -4, 1, -1, 0, 0, 0), 1, '16BUI', 100,  ↩
    0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -4, 1, -1, 0, 0, 0), 1, '16BUI', 200,  ↩
    0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -4, 1, -1, 0, 0, 0), 1, '16BUI', 300,  ↩
    0) AS rast
)
SELECT
    t1.rid,
    ST_MapAlgebra(ARRAY[ROW(ST_Union(t2.rast), 1)]::rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure,
        '32BUI',
        'CUSTOM', t1.rast,
        1, 1
```

```
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 4
    AND t2.rid BETWEEN 0 AND 8
    AND ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rid, t1.rast
```

Example like the prior one for tiles of a coverage with neighborhood but using only basic SQL features.

Code

```
WITH src AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16BUI', ↩
     1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, 0, 1, -1, 0, 0, 0), 1, '16BUI', 2, 0)  ↩
    AS rast UNION ALL
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, 0, 1, -1, 0, 0, 0), 1, '16BUI', 3, 0)  ↩
    AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -2, 1, -1, 0, 0, 0), 1, '16BUI', 10,  ↩
    0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -2, 1, -1, 0, 0, 0), 1, '16BUI', 20,  ↩
    0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -2, 1, -1, 0, 0, 0), 1, '16BUI', 30,  ↩
    0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -4, 1, -1, 0, 0, 0), 1, '16BUI', 100,  ↩
    0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -4, 1, -1, 0, 0, 0), 1, '16BUI', 200,  ↩
    0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -4, 1, -1, 0, 0, 0), 1, '16BUI', 300,  ↩
    0) AS rast
), foo AS (
    SELECT
        t1.rid,
        ST_Union(t2.rast) AS rast
    FROM src t1
    JOIN src t2
        ON ST_Intersects(t1.rast, t2.rast)
        AND t2.rid BETWEEN 0 AND 8
    WHERE t1.rid = 4
    GROUP BY t1.rid
), bar AS (
    SELECT
        t1.rid,
        ST_MapAlgebra(
            ARRAY[ROW(t2.rast, 1)]::rastbandarg[],
            'sample_callbackfunc(double precision[], int[], text[])'::regprocedure,
            '32BUI',
            'CUSTOM', t1.rast,
            1, 1
        ) AS rast
    FROM src t1
    JOIN foo t2
        ON t1.rid = t2.rid
)
SELECT
    rid,
    (ST_MetaData(rast)),
    (ST_BandMetaData(rast, 1)),
```

```
    ST_Value(rast, 1, 1, 1)
FROM bar;
```

Variants 2 and 3.

Ett raster, flera band

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        rast,
        ARRAY[3, 1, 3, 2]::integer[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

Ett raster, ett band

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        rast,
        2,
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

Variant 4.

Två rasters, två band

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast UNION ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 2, 0), 2, '8BUI', 20, 0), 3, '32BUI', 300, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        t1.rast,
        2,
        t2.rast,
        1,
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 1
```

```
    AND t2.rid = 2
```

Se även

rastbandarg, ST_Union, ST_MapAlgebra (expression version)

11.12.6 ST_MapAlgebra (expression version)

ST_MapAlgebra (expression version) — Expression version - Returnerar ett enbandsraster med ett eller två indataraster, bandindex och ett eller flera användarspecifika SQL-uttryck.

Synopsis

raster **ST_MapAlgebra**(raster rast, integer nband, text pixeltype, text expression, double precision nodataval=NULL);
raster **ST_MapAlgebra**(raster rast, text pixeltype, text expression, double precision nodataval=NULL);
raster **ST_MapAlgebra**(raster rast1, integer nband1, raster rast2, integer nband2, text expression, text pixeltype=NULL, text extenttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);
raster **ST_MapAlgebra**(raster rast1, raster rast2, text expression, text pixeltype=NULL, text extenttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);

Description

Expression version - Returnerar ett enbandsraster med ett eller två indataraster, bandindex och ett eller flera användarspecifika SQL-uttryck.

Tillgänglighet: 2.1.0

Beskrivning: Varianter 1 och 2 (ett raster)

Skapar ett nytt raster med ett band som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation som definieras av `uttrycket` på inmatningsrastret(`rast`). Om `nband` inte tillhandahålls antas band 1. Den nya rastern kommer att ha samma georeferens, bredd och höjd som den ursprungliga rastern men kommer bara att ha ett band.

Om `pixeltyp` skickas in kommer det nya rastret att ha ett band av den pixeltypen. Om pixeltyp skickas till NULL kommer det nya rasterbandet att ha samma pixeltyp som det ingående `rasterbandet`.

- Nyckelord som är tillåtna för `uttryck`
  1. `[rast]` - Pixelvärdet för den intressanta pixeln
  2. `[rast.val]` - Pixelvärdet för den intressanta pixeln
  3. `[rast.x]` - 1-baserad pixelkolumn för den intressanta pixeln
  4. `[rast.y]` - 1-baserad pixelrad för den intressanta pixeln

Beskrivning: Varianter 3 och 4 (två raster)

Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på de två banden som definieras av `uttrycket` på de två inmatningsrasterbanden `rast1,(rast2)`. Om inget `band1`, `band2` anges antas band 1. Det resulterande rastret kommer att vara justerat (skala, skevhet och pixelhörn) på det rutnät som definieras av det första rastret. Det resulterande rastret kommer att ha den utsträckning som definieras av parametern `extenttype`.

`expression` Ett PostgreSQL-algebraiskt uttryck som involverar de två rasterna och PostgreSQL-definierade funktioner / operatorer som definierar pixelvärdet när pixlar korsar varandra. t.ex. (([rast1] + [rast2])/2.0):: heltal

`pixeltype` Den resulterande pixeltypen för utdatarastern. Måste vara en som listas i ST_BandPixelType, utelämnas eller sätts till NULL. Om den inte skickas in eller sätts till NULL, kommer den första rastrets pixeltyp att användas som standard.

`extenttype` Styr omfattningen av det resulterande rastret

1. `INTERSECTION` - Det nya rastrets utsträckning är skärningspunkten mellan de två rastren. Detta är standardinställningen.
2. `UNION` - Utbredningen av det nya rastret är en sammanslagning av de två rastren.
3. `FIRST` - Utbredningen av det nya rastret är densamma som för det första rastret.
4. `SECOND` - Utbredningen av det nya rastret är densamma som för det andra rastret.

`nodata1expr` Ett algebraiskt uttryck som endast omfattar `rast2` eller en konstant som definierar vad som ska returneras när pixlar i `rast1` har nodatavärden och spatialt motsvarande pixlar i rast2 har värden.

`nodata2expr` Ett algebraiskt uttryck som endast omfattar `rast1` eller en konstant som definierar vad som ska returneras när pixlar i `rast2` har nodatavärden och spatialt motsvarande pixlar i rast1 har värden.

`nodatanodataval` En numerisk konstant som returneras när spatialt motsvarande rast1- och rast2-pixlar båda är nodatavärden.

- Nyckelord tillåtna i `uttryck`, `nodata1expr` och `nodata2expr`

1. `[rast1]` - Pixelvärdet för den intressanta pixeln från `rast1`
2. `[rast1.val]` - Pixelvärdet för den intressanta pixeln från `rast1`
3. `[rast1.x]` - 1-baserad pixelkolumn för den intressanta pixeln från `rast1`
4. `[rast1.y]` - 1-baserad pixelrad för den intressanta pixeln från `rast1`
5. `[rast2]` - Pixelvärdet för den intressanta pixeln från `rast2`
6. `[rast2.val]` - Pixelvärdet för den intressanta pixeln från `rast2`
7. `[rast2.x]` - 1-baserad pixelkolumn för den intressanta pixeln från `rast2`
8. `[rast2.y]` - 1-baserad pixelrad för den intressanta pixeln från `rast2`

Exempel

Variants 1 and 2.

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, 1, 0, 0, 0), '32BF'::text, 1, -1) ↩
     AS rast
)
SELECT
    ST_MapAlgebra(rast, 1, NULL, 'ceil([rast]*[rast.x]/[rast.y]+[rast.val])')
FROM foo;
```

Variant 3 and 4.

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI'::text, 100, 0) AS rast UNION ↩
     ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 2, 0), 2, '8BUI', 20, 0), 3, '32BUI'::text, 300, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        t1.rast,
        2,
        t2.rast,
        1,
        '([rast2] + [rast1.val]) / 2'
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 1
    AND t2.rid = 2;
```

Se även

rastbandarg, ST_Union, ST_MapAlgebra (callback function version)

11.12.7 ST_MapAlgebraExpr

ST_MapAlgebraExpr — 1 raster band version: Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på inmatningsrasterbandet och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges.

Synopsis

raster **ST_MapAlgebraExpr**(raster rast, integer band, text pixeltype, text expression, double precision nodataval=NULL);
raster **ST_MapAlgebraExpr**(raster rast, text pixeltype, text expression, double precision nodataval=NULL);

Description

Varning
ST_MapAlgebraExpr är föråldrad från och med 2.1.0. Använd ST_MapAlgebra (expression version) istället.

Skapar en ny raster med ett band som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation definierad av `uttrycket` på inmatningsrastern(`rast`). Om inget `band` anges antas band 1. Den nya rastern kommer att ha samma georeferens, bredd och höjd som den ursprungliga rastern men kommer bara att ha ett band.

Om `pixeltyp` skickas in kommer det nya rastret att ha ett band av den pixeltypen. Om pixeltyp skickas till NULL kommer det nya rasterbandet att ha samma pixeltyp som det ingående `rasterbandet`.

I uttrycket kan du använda termen `[rast]` för att hänvisa till pixelvärdet för originalbandet, `[rast.x]` för att hänvisa till det 1-baserade pixelkolumnindexet, `[rast.y]` för att hänvisa till det 1-baserade pixelradindexet.

Tillgänglighet: 2.0.0

Exempel

Skapa ett nytt 1-bandsraster från vårt original som är en funktion av modulo 2 av det ursprungliga rasterbandet.

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
UPDATE dummy_rast SET map_rast = ST_MapAlgebraExpr(rast, NULL, 'mod([rast]::numeric,2)')  ↩
    WHERE rid = 2;

SELECT
    ST_Value(rast, 1, i, j) As origval,
    ST_Value(map_rast, 1, i, j) As mapval
FROM dummy_rast
CROSS JOIN generate_series(1, 3) AS i
CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
origval | mapval
---------+--------
     253 |      1
     254 |      0
     253 |      1
     253 |      1
     254 |      0
     254 |      0
     250 |      0
     254 |      0
     254 |      0
```

Skapa ett nytt 1-bandsraster av pixeltyp 2BUI från vårt original som är omklassificerat och ställ in nodatavärdet till 0.

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast2 raster;
UPDATE dummy_rast SET
    map_rast2 = ST_MapAlgebraExpr(rast, '2BUI'::text, 'CASE WHEN [rast] BETWEEN 100 and 250 ↩
     THEN 1 WHEN [rast] = 252 THEN 2 WHEN [rast] BETWEEN 253 and 254 THEN 3 ELSE 0 END':: ↩
    text, '0')
WHERE rid = 2;

SELECT DISTINCT
    ST_Value(rast, 1, i, j) As origval,
    ST_Value(map_rast2, 1, i, j) As mapval
FROM dummy_rast
CROSS JOIN generate_series(1, 5) AS i
CROSS JOIN generate_series(1, 5) AS j
WHERE rid = 2;
```

Output

```
origval | mapval
---------+--------
     249 |      1
     250 |      1
     251 |
     252 |      2
     253 |      3
     254 |      3
```

Code

```
SELECT
    ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast
WHERE rid = 2;
```

Output

```
2BUI
```

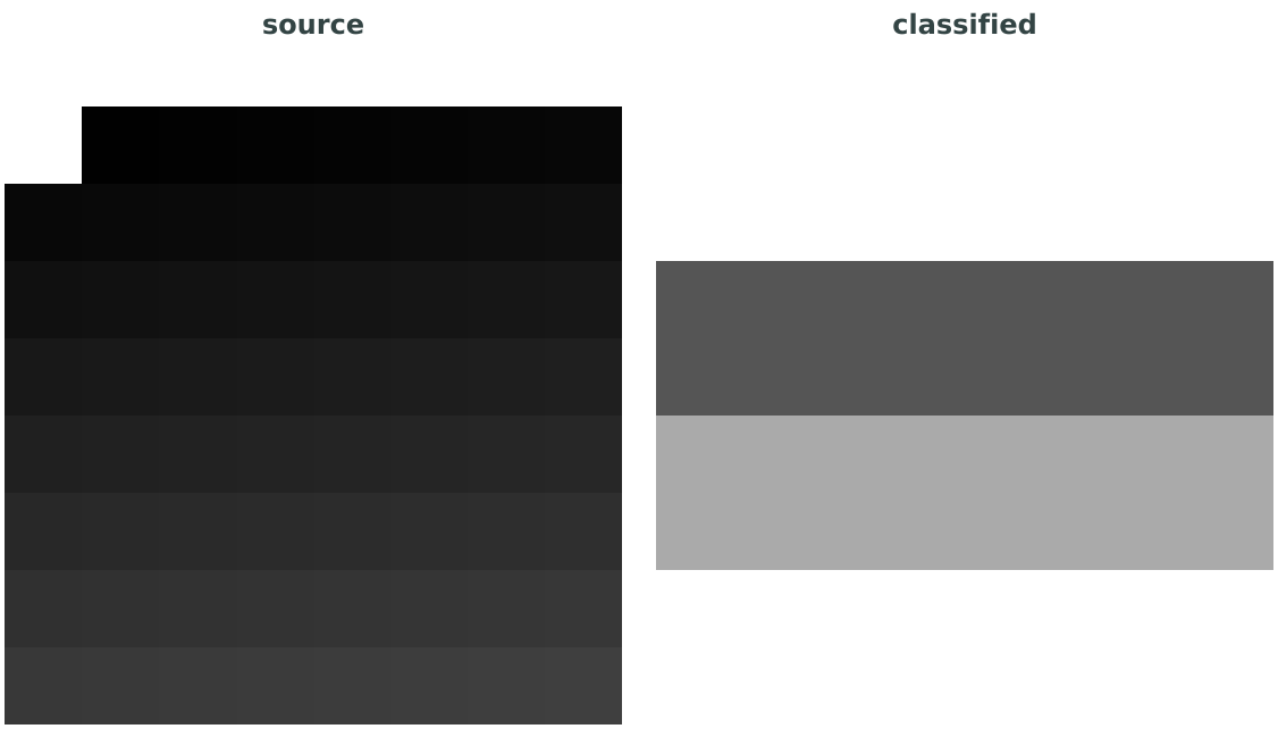
A self-contained source raster and the raster produced by classifying its values into four display levels.

Code

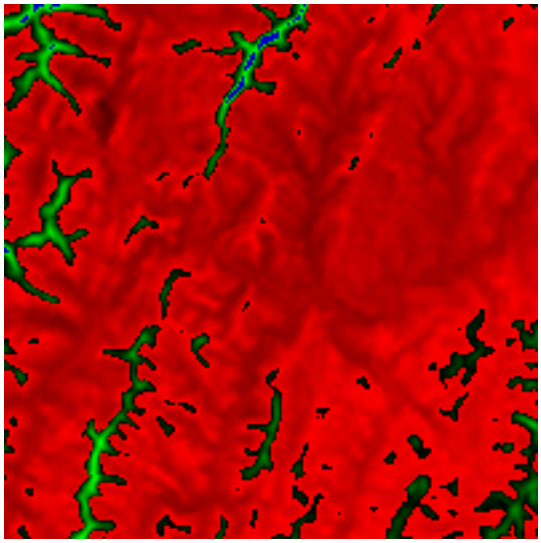
```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(8, 8, 0, 8, 1, -1, 0, 0, 0),
            1, '8BUI', 0, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 2, 3, 4, 5, 6, 7],
            [8, 9, 10, 11, 12, 13, 14, 15],
            [16, 17, 18, 19, 20, 21, 22, 23],
            [24, 25, 26, 27, 28, 29, 30, 31],
            [32, 33, 34, 35, 36, 37, 38, 39],
            [40, 41, 42, 43, 44, 45, 46, 47],
            [48, 49, 50, 51, 52, 53, 54, 55],
            [56, 57, 58, 59, 60, 61, 62, 63]
        ]::double precision[][]
    ) AS rast
), variants AS (
    SELECT 'source' AS title, rast AS rendered
    FROM source
```

```
    UNION ALL
    SELECT
        'classified',
        ST_MapAlgebraExpr(rast, '8BUI', 'floor([rast] / 16) * 85')
    FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

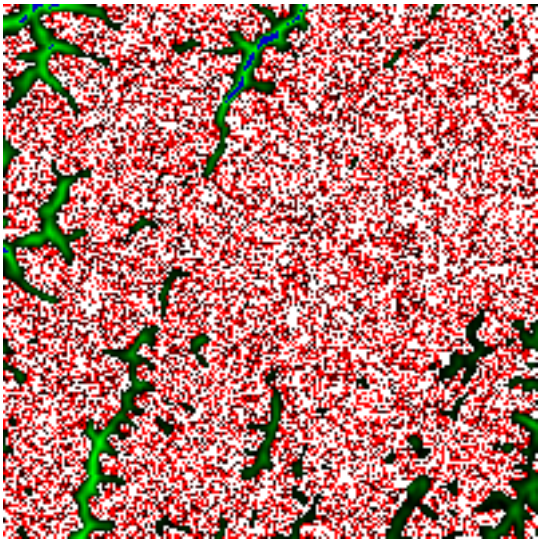
Figure



Map algebra altering the first band of a three-band raster.



original (kolumn rast_view)



rast_view_ma

Skapa en ny 3-bandsraster samma pixeltyp från vår ursprungliga 3-bandsraster med första bandet ändrat av kartalgebra och återstående 2 band oförändrade.

Code

```
SELECT
    ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(rast_view),
                ST_MapAlgebraExpr(rast_view, 1, NULL, 'tan([rast])*[rast]')
            ),
            ST_Band(rast_view, 2)
        ),
        ST_Band(rast_view, 3)
    )  As rast_view_ma
FROM wind
WHERE rid=167;
```

Se även

ST_MapAlgebraExpr, ST_MapAlgebraFct, ST_BandPixelType, ST_GeoReference, ST_Value

11.12.8 ST_MapAlgebraExpr

ST_MapAlgebraExpr — 2 rasterbandversion: Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på de två inmatningsrasterbanden och av pixeltyp som tillhandahålls. band 1 för varje raster antas om inga bandnummer anges. Den resulterande rastern kommer att justeras (skala, skevhet och pixelhörn) på det rutnät som definieras av den första rastern och ha sin utsträckning definierad av parametern ”extenttype”. Värden för ”extenttype”kan vara: INTERSECTION, UNION, FIRST, SECOND.

Synopsis

raster **ST_MapAlgebraExpr**(raster rast1, raster rast2, text expression, text pixeltype=same_as_rast1_band text extenttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);

raster **ST_MapAlgebraExpr**(raster rast1, integer band1, raster rast2, integer band2, text expression, text pixeltype=same_as_rast1_band, text extenttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);

Description

Varning
ST_MapAlgebraExpr är föråldrad från och med 2.1.0. Använd ST_MapAlgebra (expression version) istället.

Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på de två banden som definieras av `uttrycket` på de två inmatningsrasterbanden `rast1,(rast2)`. Om inget `band1`, `band2` anges antas band 1. Det resulterande rastret kommer att vara justerat (skala, skevhet och pixelhörn) på det rutnät som definieras av det första rastret. Det resulterande rastret kommer att ha den utsträckning som definieras av parametern `extenttype`.

`expression` Ett PostgreSQL-algebraiskt uttryck som involverar de två rasterna och PostgreSQL-definierade funktioner / operatorer som definierar pixelvärdet när pixlar korsar varandra. t.ex. (([rast1] + [rast2])/2.0):: heltal

`pixeltype` Den resulterande pixeltypen för utdatarastern. Måste vara en som listas i ST_BandPixelType, utelämnas eller sätts till NULL. Om den inte skickas in eller sätts till NULL, kommer den första rastrets pixeltyp att användas som standard.

`extenttype` Styr omfattningen av det resulterande rastret

1. `INTERSECTION` - Det nya rastrets utsträckning är skärningspunkten mellan de två rastren. Detta är standardinställningen.
2. `UNION` - Utbredningen av det nya rastret är en sammanslagning av de två rastren.
3. `FIRST` - Utbredningen av det nya rastret är densamma som för det första rastret.
4. `SECOND` - Utbredningen av det nya rastret är densamma som för det andra rastret.

`nodata1expr` Ett algebraiskt uttryck som endast omfattar `rast2` eller en konstant som definierar vad som ska returneras när pixlar i `rast1` har nodatavärden och spatialt motsvarande pixlar i rast2 har värden.

`nodata2expr` Ett algebraiskt uttryck som endast omfattar `rast1` eller en konstant som definierar vad som ska returneras när pixlar i `rast2` har nodatavärden och spatialt motsvarande pixlar i rast1 har värden.

`nodatanodataval` En numerisk konstant som returneras när spatialt motsvarande rast1- och rast2-pixlar båda är nodatavärden.

Om `pixeltype` skickas in kommer det nya rastret att ha ett band av den pixeltypen. Om pixeltype skickas till NULL eller ingen pixeltyp anges, kommer det nya rasterbandet att ha samma pixeltyp som det inmatade `rast1-bandet`.

Använd termen `[rast1. val] [rast2.val]` för att hänvisa till pixelvärdet för de ursprungliga rasterbanden och [rast1. `x`], [rast1. `y`] etc. för att hänvisa till pixlarnas kolumn-/radpositioner.

Tillgänglighet: 2.0.0

Exempel

2 Band Intersection and Union.

Skapa ett nytt 1-bandsraster från vårt original som är en funktion av modulo 2 av det ursprungliga rasterbandet.

Create a cool set of rasters.

Insert some cool shapes around Boston in Massachusetts state plane meters, then map their intersection and union.

Code

```
DROP TABLE IF EXISTS fun_shapes;
CREATE TABLE fun_shapes(rid serial PRIMARY KEY, fun_name text, rast raster);

INSERT INTO fun_shapes(fun_name, rast)
VALUES ('ref', ST_AsRaster(ST_MakeEnvelope(235229, 899970, 237229, 901930, 26986), 200,  ↩
    200, '8BUI', 0, 0));

INSERT INTO fun_shapes(fun_name, rast)
WITH ref(rast) AS (SELECT rast FROM fun_shapes WHERE fun_name = 'ref' )
SELECT 'area' AS fun_name, ST_AsRaster(ST_Buffer(ST_SetSRID(ST_Point(236229, 900930),  ↩
    26986), 1000),
            ref.rast, '8BUI', 10, 0) As rast
FROM ref
UNION ALL
SELECT 'rand bubbles',
       ST_AsRaster(
           (
               SELECT ST_Collect(geom)
               FROM (
                   SELECT ST_Buffer(
                       ST_SetSRID(ST_Point(236229 + i*random()*100, 900930 + j*random() ↩
    *100), 26986),
                       random()*20
                   ) AS geom
                   FROM generate_series(1, 10) AS i,
                        generate_series(1, 10) AS j
               ) AS foo
           ),
           ref.rast, '8BUI', 200, 0
       )
FROM ref;

SELECT  ST_MapAlgebraExpr(area.rast, bub.rast, '[rast2.val]', '8BUI', 'INTERSECTION', '[ ↩
    rast2.val]', '[rast1.val]') As interrast,
        ST_MapAlgebraExpr(area.rast, bub.rast, '[rast2.val]', '8BUI', 'UNION', '[rast2.val] ↩
    ', '[rast1.val]') As unionrast
FROM
  (SELECT rast FROM fun_shapes WHERE fun_name = 'area') As area
CROSS JOIN
  (SELECT rast FROM fun_shapes WHERE fun_name = 'rand bubbles') As bub
```

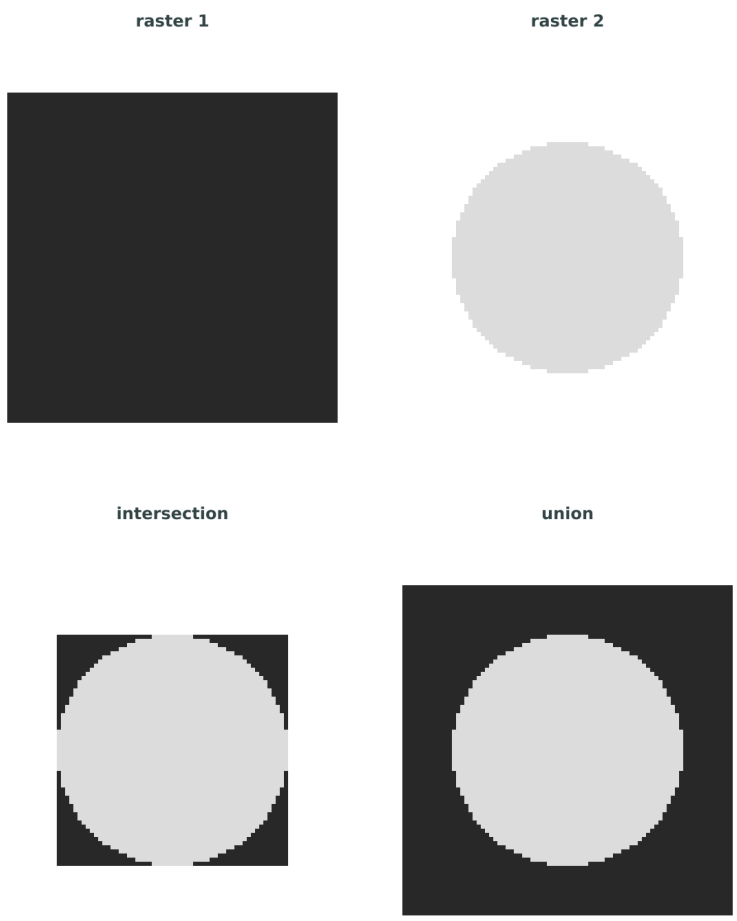
The same intersection and union operations using two deterministic, self-contained rasters.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(80, 80, 0, 80, 1, -1, 0, 0, 0),
```

```
        1, '8BUI', 40, 0
    ) AS rast
), inputs AS (
    SELECT
        rast AS raster_1,
        ST_AsRaster(
            ST_Buffer(ST_Point(40, 40), 28),
            rast,
            '8BUI', 220, 0
        ) AS raster_2
    FROM canvas
), variants AS (
    SELECT 'raster 1' AS title, raster_1 AS rendered FROM inputs
    UNION ALL
    SELECT 'raster 2', raster_2 FROM inputs
    UNION ALL
    SELECT 'intersection', ST_MapAlgebraExpr(
        raster_1, raster_2,
        '[rast2.val]', '8BUI', 'INTERSECTION',
        '[rast2.val]', '[rast1.val]'
    ) FROM inputs
    UNION ALL
    SELECT 'union', ST_MapAlgebraExpr(
        raster_1, raster_2,
        '[rast2.val]', '8BUI', 'UNION',
        '[rast2.val]', '[rast1.val]'
    ) FROM inputs
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY CASE title
    WHEN 'raster 1' THEN 1
    WHEN 'raster 2' THEN 2
    WHEN 'intersection' THEN 3
    ELSE 4
END;
```

Figure



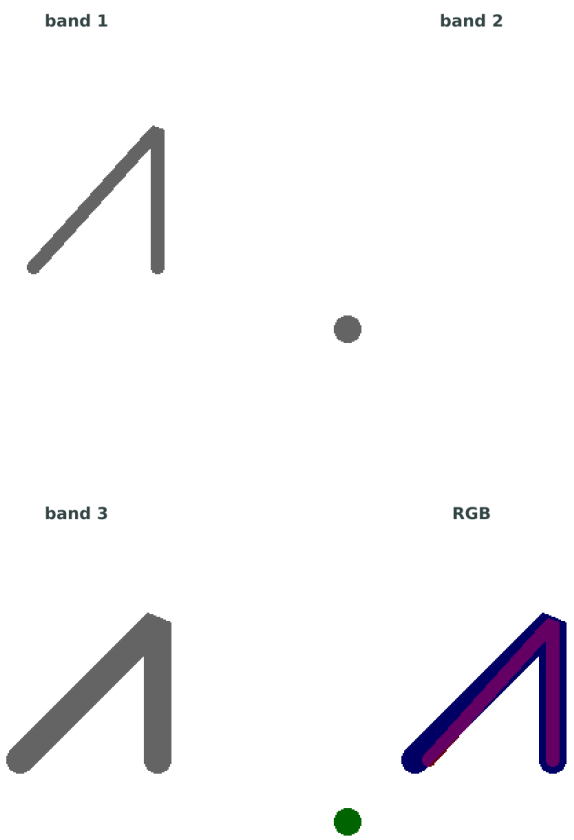
Overlaying rasters on a one-unit-per-pixel canvas as separate bands. We use ST_AsPNG to render the image so all single band ones look grey.

Code

```
WITH mygeoms AS (
    SELECT 2 AS bnum, ST_Buffer(ST_Point(1, 5), 10) AS geom
    UNION ALL
    SELECT 3,
        ST_Buffer(
            ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'),
            10,
            'join=bevel'
        )
    UNION ALL
    SELECT 1,
        ST_Buffer(
            ST_GeomFromText('LINESTRING(60 50,150 150,150 50)'),
            5,
            'join=bevel'
        )
), canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(
```

```
            200,
            200,
            ST_XMin(e)::integer,
            ST_YMax(e)::integer,
            1, -1, 0, 0),
        '8BUI'::text,
        0
    ) AS rast
    FROM (
      SELECT ST_Extent(geom) AS e
      FROM mygeoms
    ) AS bounds
), rbands AS (
    SELECT ARRAY(
        SELECT ST_MapAlgebraExpr(
            canvas.rast,
            ST_AsRaster(m.geom, canvas.rast, '8BUI', 100),
            '[rast2.val]', '8BUI', 'FIRST',
            '[rast2.val]', '[rast1.val]'
        )
        FROM mygeoms AS m
        CROSS JOIN canvas
        ORDER BY m.bnum
    ) AS rasts
)
SELECT title, ST_AsPNG(rendered) AS image
FROM rbands
CROSS JOIN LATERAL (VALUES
    ('band 1', rasts[1]),
    ('band 2', rasts[2]),
    ('band 3', rasts[3]),
    ('RGB', ST_AddBand(ST_AddBand(rasts[1], rasts[2]), rasts[3]))
) AS variants(title, rendered);
```

Figure



Overlay a two-meter boundary of selected parcels over aerial imagery. The query first unions the parcel geometries, clips the source raster shards to the region, unions those smaller shards, and finally adds the first two bands plus a third-band map algebra overlay.

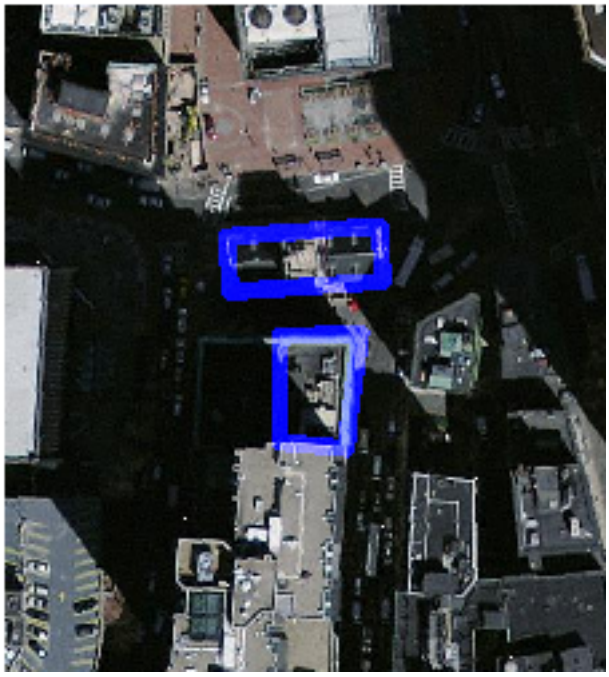
Create a new three-band raster composed of the first two clipped bands and the third band overlaid with the geometry. This query took 3.6 seconds on a 64-bit PostGIS Windows installation.

Code

```
WITH pr AS (
    SELECT ST_Clip(rast, ST_Expand(geom, 50)) AS rast, g.geom
    FROM aerials.o_2_boston AS r
    INNER JOIN (
        SELECT ST_Union(ST_Transform(geom, 26986)) AS geom
        FROM landparcels
        WHERE pid IN ('0303890000', '0303900000')
    ) AS g
        ON ST_Intersects(rast::geometry, ST_Expand(g.geom, 50))
), prunion AS (
    SELECT ST_AddBand(
        NULL,
        ARRAY[
            ST_Union(rast, 1),
            ST_Union(rast, 2),
```

```
            ST_Union(rast, 3)
        ]
    ) AS clipped,
    geom
    FROM pr
    GROUP BY geom
)
SELECT ST_AddBand(
    ST_Band(clipped, ARRAY[1, 2]),
    ST_MapAlgebraExpr(
     ST_Band(clipped, 3),
     ST_AsRaster(ST_Buffer(ST_Boundary(geom), 2), clipped, '8BUI', 250),
     '[rast2.val]',
     '8BUI',
     'FIRST',
     '[rast2.val]',
     '[rast1.val]') ) As rast
FROM prunion;
```

Parcel-boundary overlay.



The blue lines are the boundaries of selected parcels

The generated four-panel canvas example above shows the same band-overlay technique without requiring an external aerial-imagery or parcel table.

Se även

ST_MapAlgebraExpr, ST_AddBand, ST_AsPNG, ST_AsRaster, ST_MapAlgebraFct, ST_BandPixelType, ST_GeoReference, ST_Value, ST_Union, ST_Union

11.12.9 ST_MapAlgebraFct

ST_MapAlgebraFct — 1 bandversion - Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion på inmatningsrasterbandet och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges.

Synopsis

raster **ST_MapAlgebraFct**(raster rast, regprocedure onerasteruserfunc);
raster **ST_MapAlgebraFct**(raster rast, regprocedure onerasteruserfunc, text[] VARIADIC args);
raster **ST_MapAlgebraFct**(raster rast, text pixeltype, regprocedure onerasteruserfunc);
raster **ST_MapAlgebraFct**(raster rast, text pixeltype, regprocedure onerasteruserfunc, text[] VARIADIC args);
raster **ST_MapAlgebraFct**(raster rast, integer band, regprocedure onerasteruserfunc);
raster **ST_MapAlgebraFct**(raster rast, integer band, regprocedure onerasteruserfunc, text[] VARIADIC args);
raster **ST_MapAlgebraFct**(raster rast, integer band, text pixeltype, regprocedure onerasteruserfunc);
raster **ST_MapAlgebraFct**(raster rast, integer band, text pixeltype, regprocedure onerasteruserfunc, text[] VARIADIC args);

Description

Varning
ST_MapAlgebraFct är föråldrad från och med 2.1.0. Använd ST_MapAlgebra (callback function version) istället.

Skapar ett nytt ettbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion som anges av `onerasteruserfunc` på inmatningsrastret`(rast)`. Om inget `band` anges antas band 1. Det nya rastret kommer att ha samma georeferens, bredd och höjd som det ursprungliga rastret men kommer bara att ha ett band.

Om `pixeltyp` skickas in kommer det nya rastret att ha ett band av den pixeltypen. Om pixeltyp skickas till NULL kommer det nya rasterbandet att ha samma pixeltyp som det ingående `rasterbandet`.

Parametern `onerasteruserfunc` måste vara namnet och signaturen för en SQL- eller PL/pgSQL-funktion, som kastas till en regprocedure. Ett mycket enkelt och ganska värdelöst PL / pgSQL-funktionsexem är:

Code

```
CREATE OR REPLACE FUNCTION simple_function(pixel FLOAT, pos INTEGER[], VARIADIC args TEXT ↩
    [])
RETURNS FLOAT
AS $$ BEGIN
    RETURN 0.0;
END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

`Användarfunktionen` kan ta emot två eller tre argument: ett floatvärde, en valfri heltalsarray och en variadisk textarray. Det första argumentet är värdet för en enskild rastercell (oberoende av rasterdatatyp). Det andra argumentet är positionen för den aktuella bearbetningscellen i formen '{x,y}'. Det tredje argumentet anger att alla återstående parametrar till ST_MapAlgebraFct ska skickas vidare till `användarfunktionen`.

För att skicka ett regprodedure-argument till en SQL-funktion krävs att hela funktionssignaturen skickas och sedan kastas till en regprocedure-typ. För att skicka ovanstående exempel på PL / pgSQL-funktion som ett argument är SQL för argumentet:

Code

```
'simple_function(float,integer[],text[])'::regprocedure
```

Observera att argumentet innehåller namnet på funktionen, typerna av funktionsargumenten, citattecken runt namnet och argumenttyperna och en cast till en regprocedure.

Det tredje argumentet till `användarfunktionen` är en variadisk textmatris. Alla efterföljande textargument i ett anrop till ST_MapAlgebraFct skickas vidare till den angivna `användarfunktionen` och ingår i argumentet `args`.

Notera
För mer information om VARIADIC-nyckelordet, se PostgreSQL-dokumentationen och avsnittet SQL-funktioner med variabelt antal argument”i Query Language (SQL) -funktioner.

Notera
Argumentet text[] till `användarfunktionen` är obligatoriskt, oavsett om du väljer att skicka några argument till din användarfunktion för bearbetning eller inte.

Tillgänglighet: 2.0.0

Exempel

Skapa ett nytt 1-bandsraster från vårt original som är en funktion av modulo 2 av det ursprungliga rasterbandet.

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
CREATE FUNCTION mod_fct(pixel float, pos integer[], variadic args text[])
RETURNS float
AS $$
BEGIN
    RETURN pixel::integer % 2;
END;
$$
LANGUAGE 'plpgsql' IMMUTABLE;

UPDATE dummy_rast SET map_rast = ST_MapAlgebraFct(rast, NULL, 'mod_fct(float,integer[],text ↩
    [])'::regprocedure) WHERE rid = 2;

SELECT ST_Value(rast, 1, i, j) As origval, ST_Value(map_rast, 1, i, j) As mapval
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
origval | mapval
---------+--------
     253 |      1
     254 |      0
     253 |      1
```

```
     253 |      1
     254 |      0
     254 |      0
     250 |      0
     254 |      0
     254 |      0
```

Skapa ett nytt 1-bandsraster av pixeltyp 2BUI från vårt original som omklassificeras och ställ in no-datavärdet till en passerad parameter till användarfunktionen (0).

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast2 raster;
CREATE FUNCTION classify_fct(pixel float, pos integer[], variadic args text[])
RETURNS float
AS
$$
DECLARE
    nodata float := 0;
BEGIN
    IF NOT args[1] IS NULL THEN
        nodata := args[1];
    END IF;
    IF pixel < 251 THEN
        RETURN 1;
    ELSIF pixel = 252 THEN
        RETURN 2;
    ELSIF pixel
> 252 THEN
        RETURN 3;
    ELSE
        RETURN nodata;
    END IF;
END;
$$
LANGUAGE 'plpgsql';
UPDATE dummy_rast SET map_rast2 = ST_MapAlgebraFct(rast,'2BUI','classify_fct(float,integer ↩
    [],text[])'::regprocedure, '0') WHERE rid = 2;

SELECT DISTINCT ST_Value(rast,1,i,j) As origval, ST_Value(map_rast2, 1, i, j) As mapval
FROM dummy_rast CROSS JOIN generate_series(1, 5) AS i CROSS JOIN generate_series(1,5) AS j
WHERE rid = 2;
```

Output

```
 origval | mapval
---------+--------
     249 |      1
     250 |      1
     251 |
     252 |      2
     253 |      3
     254 |      3
```

Code

```
SELECT ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast WHERE rid = 2;
```

Output

```
 b1pixtyp
----------
 2BUI
```

Skapa en ny 3-bandsraster samma pixeltyp från vår ursprungliga 3-bandsraster med första bandet ändrat av kartalgebra och återstående 2 band oförändrade.

Code

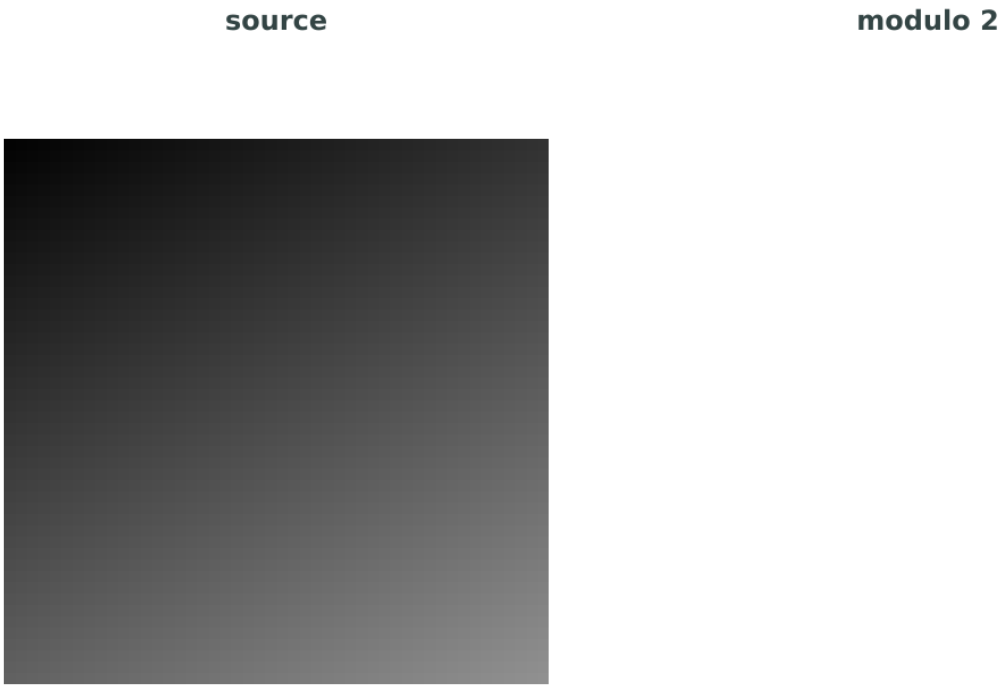
```
CREATE FUNCTION rast_plus_tan(pixel float, pos integer[], variadic args text[])
RETURNS float
AS
$$
BEGIN
    RETURN tan(pixel) * pixel;
END;
$$
LANGUAGE 'plpgsql';

SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(rast_view),
            ST_MapAlgebraFct(rast_view, 1, NULL, 'rast_plus_tan(float,integer[],text[])':: ↩
    regprocedure)
        ),
        ST_Band(rast_view, 2)
    ),
    ST_Band(rast_view, 3) As rast_view_ma
)
FROM wind
WHERE rid=167;
```

The deprecated callback form above is equivalent to a modern ST_MapAlgebra (callback function version) expression for simple per-pixel transforms. This build-time example renders the original raster and a modulo-2 classification.

Code

```
WITH pixels AS (
    SELECT x, y, (x + y * 2)::double precision AS value
    FROM generate_series(1, 48) AS x
    CROSS JOIN generate_series(1, 48) AS y
), rows AS (
    SELECT y, array_agg(value ORDER BY x) AS values
    FROM pixels
    GROUP BY y
), source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(48, 48, 0, 48, 1, -1, 0, 0, 0),
            1, '8BUI', 0, 0
        ),
        1, 1, 1, array_agg(values ORDER BY y)::double precision[][]
    ) AS rast
    FROM rows
), variants AS (
    SELECT 'source' AS title, rast AS rendered FROM source
    UNION ALL
    SELECT 'modulo 2', ST_MapAlgebra(
        rast, 1,
        '8BUI',
        'mod([rast]::integer,2) * 255'
    )
    FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Figure



Se även

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

11.12.10 ST_MapAlgebraFct

ST_MapAlgebraFct — 2 band version - Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion på de 2 inmatningsrasterbanden och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges. Utsträckningstyp är som standard INTERSECTION om den inte anges.

Synopsis

raster **ST_MapAlgebraFct**(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);
raster **ST_MapAlgebraFct**(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

Description

Varning
ST_MapAlgebraFct är föråldrad från och med 2.1.0. Använd ST_MapAlgebra (callback function version) istället.

Skapar ett nytt ettbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion som anges av `tworastuserfunc` på inmatningsrastret `rast1`, `rast2`. Om inget `band1` eller `band2` anges antas band 1. Det nya rastret kommer att ha samma georeferens, bredd och höjd som de ursprungliga rastren men kommer bara att ha ett band.

Om `pixeltype` skickas in kommer det nya rastret att ha ett band av den pixeltypen. Om pixeltype skickas som NULL eller utelämnas kommer det nya rasterbandet att ha samma pixeltype som det ingående `rast1-bandet`.

Parametern `tworastuserfunc` måste vara namnet och signaturen för en SQL- eller PL/pgSQL-funktion, som kastas till en regprocedure. Ett exempel på PL/pgSQL-funktionsexempel är:

Code

```
CREATE OR REPLACE FUNCTION simple_function_for_two_rasters(pixel1 FLOAT, pixel2 FLOAT, pos ↩
    INTEGER[], VARIADIC args TEXT[])
RETURNS FLOAT
AS $$ BEGIN
    RETURN 0.0;
END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

`Tworastuserfunc` kan acceptera tre eller fyra argument: ett värde med dubbel precision, ett värde med dubbel precision, en valfri heltalsarray och en variadisk textarray. Det första argumentet är värdet för en enskild rastercell i `rast1` (oavsett rasterdatatyp). Det andra argumentet är ett individuellt rastercellsvärde i `rast2`. Det tredje argumentet är positionen för den aktuella bearbetningscellen i formen '{x,y}'. Det fjärde argumentet anger att alla återstående parametrar till ST_MapAlgebraFct ska skickas vidare till `tworastuserfunc`.

För att skicka ett regprodedure-argument till en SQL-funktion krävs att hela funktionssignaturen skickas och sedan kastas till en regprocedure-typ. För att skicka ovanstående exempel på PL / pgSQL-funktion som ett argument är SQL för argumentet:

Code

```
'simple_function(double precision, double precision, integer[], text[])'::regprocedure
```

Observera att argumentet innehåller namnet på funktionen, typerna av funktionsargumenten, citattecken runt namnet och argumenttyperna och en cast till en regprocedure.

Det fjärde argumentet till `tworastuserfunc` är en variadisk textmatris. Alla efterföljande textargument till ett ST_MapAlgebraFct -anrop skickas vidare till den angivna `tworastuserfunc` och ingår i argumentet `userargs`.

Notera
För mer information om VARIADIC-nyckelordet, se PostgreSQL-dokumentationen och avsnittet SQL-funktioner med variabelt antal argument"i Query Language (SQL) -funktioner.

Notera
Argumentet text[] till `tworastuserfunc` är obligatoriskt, oavsett om du väljer att skicka några argument till din användarfunktion för bearbetning eller inte.

Tillgänglighet: 2.0.0

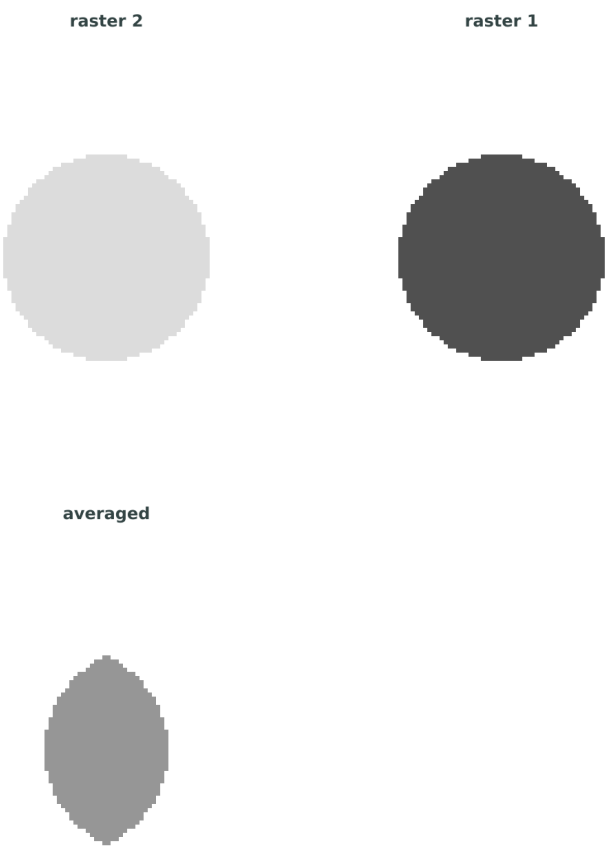
Exempel: Överlagring av raster på en duk som separata band

The deprecated two-raster callback form can often be expressed with the modern two-raster ST_MapAlgebra (callback function version) expression variant. This build-time example renders two input rasters and their averaged overlap.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(80, 80, 0, 80, 1, -1, 0, 0, 0),
        1, '8BUI', 0, 0
    ) AS rast
), inputs AS (
    SELECT
        ST_AsRaster(ST_Buffer(ST_Point(30, 40), 25), rast, '8BUI', 80, 0)
            AS raster_1,
        ST_AsRaster(ST_Buffer(ST_Point(50, 40), 25), rast, '8BUI', 220, 0)
            AS raster_2
    FROM canvas
), variants AS (
    SELECT 'raster 1' AS title, raster_1 AS rendered FROM inputs
    UNION ALL
    SELECT 'raster 2', raster_2 FROM inputs
    UNION ALL
    SELECT 'averaged', ST_MapAlgebra(
        raster_1, 1,
        raster_2, 1,
        '([rast1] + [rast2]) / 2',
        '8BUI',
        'UNION'
    ) FROM inputs
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Figure



Define our user defined function.

Code

```
CREATE OR REPLACE FUNCTION raster_mapalgebra_union(
    rast1 double precision,
    rast2 double precision,
    pos integer[],
    VARIADIC userargs text[]
)
    RETURNS double precision
    AS $$
    DECLARE
    BEGIN
        CASE
            WHEN rast1 IS NOT NULL AND rast2 IS NOT NULL THEN
                RETURN ((rast1 + rast2)/2.);
            WHEN rast1 IS NULL AND rast2 IS NULL THEN
                RETURN NULL;
            WHEN rast1 IS NULL THEN
                RETURN rast2;
            ELSE
                RETURN rast1;
```

```
        END CASE;

        RETURN NULL;
    END;
    $$ LANGUAGE 'plpgsql' IMMUTABLE COST 1000;
```

Prepare a test table of rasters.

Code

```
DROP TABLE IF EXISTS map_shapes;
CREATE TABLE map_shapes(rid serial PRIMARY KEY, rast raster, bnum integer, descrip text);
INSERT INTO map_shapes(rast,bnum, descrip)
WITH mygeoms
    AS ( SELECT 2 As bnum, ST_Buffer(ST_Point(90,90),30) As geom, 'circle' As descrip
            UNION ALL
            SELECT 3 AS bnum,
                ST_Buffer(ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'), 15) As geom, ↩
     'big road' As descrip
            UNION ALL
            SELECT 1 As bnum,
                ST_Translate(ST_Buffer(ST_GeomFromText('LINESTRING(60 50,150 150,150 50)'), ↩
     8,'join=bevel'), 10,-6) As geom, 'small road' As descrip
            ),
   -- define our canvas to be 1 to 1 pixel to geometry
   canvas
    AS ( SELECT ST_AddBand(
        ST_MakeEmptyRaster(
            250,
            250,
            ST_XMin(e)::integer,
            ST_YMax(e)::integer,
            1, -1, 0, 0),
        '8BUI'::text,
        0) As rast
        FROM (SELECT ST_Extent(geom) As e,
                    Max(ST_SRID(geom)) As srid
                    from mygeoms
                    ) As foo
            )
-- return our rasters aligned with our canvas
SELECT ST_AsRaster(m.geom, canvas.rast, '8BUI', 240) As rast, bnum, descrip
                FROM mygeoms AS m CROSS JOIN canvas
UNION ALL
SELECT canvas.rast, 4, 'canvas'
FROM canvas;
```

Map algebra on single-band rasters and then collect the result with ST_AddBand.

Code

```
INSERT INTO map_shapes(rast,bnum,descrip)
SELECT ST_AddBand(ST_AddBand(rasts[1], rasts[2]),rasts[3]), 4, 'map bands overlay fct union ↩
     (canvas)'
    FROM (SELECT ARRAY(SELECT ST_MapAlgebraFct(
            m1.rast,
            m2.rast,
            'raster_mapalgebra_union(double precision, double precision, integer[], text[]) ↩
    '::regprocedure,
            '8BUI',
            'FIRST')
                FROM map_shapes As m1 CROSS JOIN map_shapes As m2
```

```
    WHERE m1.descrip = 'canvas' AND m2.descrip <
> 'canvas' ORDER BY m2.bnum) As rasts) As foo;
```

A compact, executable callback example. The source rasters and their averaged overlap are rendered at build time.

Code

```
CREATE OR REPLACE FUNCTION pg_temp.raster_average(
    value_1 double precision,
    value_2 double precision,
    position integer[],
    VARIADIC userargs text[]
) RETURNS double precision
LANGUAGE SQL IMMUTABLE
AS 'SELECT COALESCE(($1 + $2) / 2.0, $1, $2)';

WITH canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(80, 80, 0, 80, 1, -1, 0, 0, 0),
        1, '8BUI', 0, 0
    ) AS rast
), inputs AS (
    SELECT
        ST_AsRaster(ST_Buffer(ST_Point(30, 40), 25), rast, '8BUI', 80, 0)
            AS raster_1,
        ST_AsRaster(ST_Buffer(ST_Point(50, 40), 25), rast, '8BUI', 220, 0)
            AS raster_2
    FROM canvas
), variants AS (
    SELECT 'raster 1' AS title, raster_1 AS rendered FROM inputs
    UNION ALL
    SELECT 'raster 2', raster_2 FROM inputs
    UNION ALL
    SELECT 'averaged', ST_MapAlgebraFct(
        raster_1,
        raster_2,
        'pg_temp.raster_average(double precision,double precision,integer[],text[])':: ↩
    regprocedure,
        '8BUI',
        'UNION'
    ) FROM inputs
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Användardefinierad funktion som tar extra args

Code

```
CREATE OR REPLACE FUNCTION raster_mapalgebra_userargs(
    rast1 double precision,
    rast2 double precision,
    pos integer[],
    VARIADIC userargs text[]
)
    RETURNS double precision
    AS $$
```

```
    DECLARE
    BEGIN
        CASE
            WHEN rast1 IS NOT NULL AND rast2 IS NOT NULL THEN
                RETURN least(userargs[1]::integer,(rast1 + rast2)/2.);
            WHEN rast1 IS NULL AND rast2 IS NULL THEN
                RETURN userargs[2]::integer;
            WHEN rast1 IS NULL THEN
                RETURN greatest(rast2, random()*userargs[3]::integer)::integer;
            ELSE
                RETURN greatest(rast1, random()*userargs[4]::integer)::integer;
        END CASE;

        RETURN NULL;
    END;
    $$ LANGUAGE 'plpgsql' VOLATILE COST 1000;

SELECT ST_MapAlgebraFct(
    m1.rast,
    1,
    m1.rast,
    3,
    'raster_mapalgebra_userargs(double precision, double precision, integer[], text[])':: ↩
    regprocedure,
    '8BUI',
    'INTERSECT',
    '100', '200', '200', '0')
                FROM map_shapes As m1
    WHERE m1.descrip = 'map bands overlay fct union (canvas)';
```

The generated callback figure above demonstrates the raster alignment and overlap used by this variant; the exact callback result remains available as text.

Se även

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

11.12.11 ST_MapAlgebraFctNgb

ST_MapAlgebraFctNgb — 1-bandsversion: Kartlägg Algebra närmaste granne med hjälp av användardefinierad PostgreSQL-funktion. Returnera en raster vars värden är resultatet av en PLPGSQL-användarfunktion som involverar ett grannskap av värden från inmatningsrasterbandet.

Synopsis

raster **ST_MapAlgebraFctNgb**(raster rast, integer band, text pixeltype, integer ngbwidth, integer ngbheight, regprocedure onerastngbuserfunc, text nodatamode, text[] VARIADIC args);

Description

Varning
ST_MapAlgebraFctNgb är föråldrad från och med 2.1.0. Använd ST_MapAlgebra (callback function version) istället.

(en rasterversion) Returnera en raster vars värden är resultatet av en PLPGSQL-användarfunktion som involverar ett grannskap av värden från inmatningsrasterbandet. Användarfunktionen tar grannskapet av pixelvärden som en matris med siffror, för varje pixel, returnerar resultatet från användarfunktionen och ersätter pixelvärdet för den för närvarande inspekterade pixeln med funktionsresultatet.

rast Raster på vilket användarfunktionen utvärderas.

band Bandnummer för det raster som ska utvärderas. Standardvärdet är 1.

pixeltype Den resulterande pixeltypen för utdatarastern. Måste vara en som listas i ST_BandPixelType eller utelämnas eller sätts till NULL. Om den inte skickas in eller sätts till NULL, kommer pixeltypen för `rastret` att användas som standard. Resultaten trunkeras om de är större än vad som är tillåtet för pixeltypen.

ngbwidth Bredden på grannskapet, i celler.

ngbheight Höjden på kvarteret, i celler.

onerastngbuserfunc PLPGSQL / psql-användarfunktion för att applicera på grannskapspixlar i ett enda band av en raster. Det första elementet är en 2-dimensionell matris med siffror som representerar den rektangulära pixelgrannskapet

nodatamode Definierar vilket värde som ska skickas till funktionen för en grannskapspixel som är nodata eller NULL

'ignore': alla NODATA-värden som påträffas i grannskapet ignoreras av beräkningen -- denna flagga måste skickas till användarens återuppringningsfunktion, och användarfunktionen avgör hur den ska ignoreras.

NULL": alla NODATA-värden som påträffas i grannskapet gör att den resulterande pixeln blir NULL - användarens återuppringningsfunktion hoppas över i detta fall.

"value": alla NODATA-värden som påträffas i grannskapet ersätts av referenspixeln (den i mitten av grannskapet). Observera att om detta värde är NODATA är beteendet detsamma som 'NULL' (för den berörda grannskapet)

args Argument som ska skickas till användarfunktionen.

Tillgänglighet: 2.0.0

Exempel

Exemplen använder katrina-raster som lästs in som en enda bricka som beskrivs i https://gdal.org/-user/drivers/raster/postgisraster.html och sedan förberetts i exemplen ST_Rescale

A simple 'callback' user function that averages up all the values in a neighborhood.

Code

```
CREATE OR REPLACE FUNCTION rast_avg(matrix float[][], nodatamode text, variadic args text ↩
    [])
    RETURNS float AS
    $$
    DECLARE
        _matrix float[][];
        x1 integer;
        x2 integer;
        y1 integer;
        y2 integer;
        sum float;
    BEGIN
```

```
        _matrix := matrix;
        sum := 0;
        FOR x in array_lower(matrix, 1)..array_upper(matrix, 1) LOOP
            FOR y in array_lower(matrix, 2)..array_upper(matrix, 2) LOOP
                sum := sum + _matrix[x][y];
            END LOOP;
        END LOOP;
        RETURN (sum*1.0/(array_upper(matrix,1)*array_upper(matrix,2) ))::integer ;
    END;
    $$
LANGUAGE 'plpgsql' IMMUTABLE COST 1000;
```

Apply the raster averaging callback within two pixels of each input pixel in the X and Y directions.

Code

```
SELECT ST_MapAlgebraFctNgb(
        rast,
        1,
        '8BUI',
        4,
        4,
        'rast_avg(float[][], text, text[])'::regprocedure,
        'NULL',
        NULL) As nn_with_border
    FROM katrinas_rescaled
    limit 1;
```

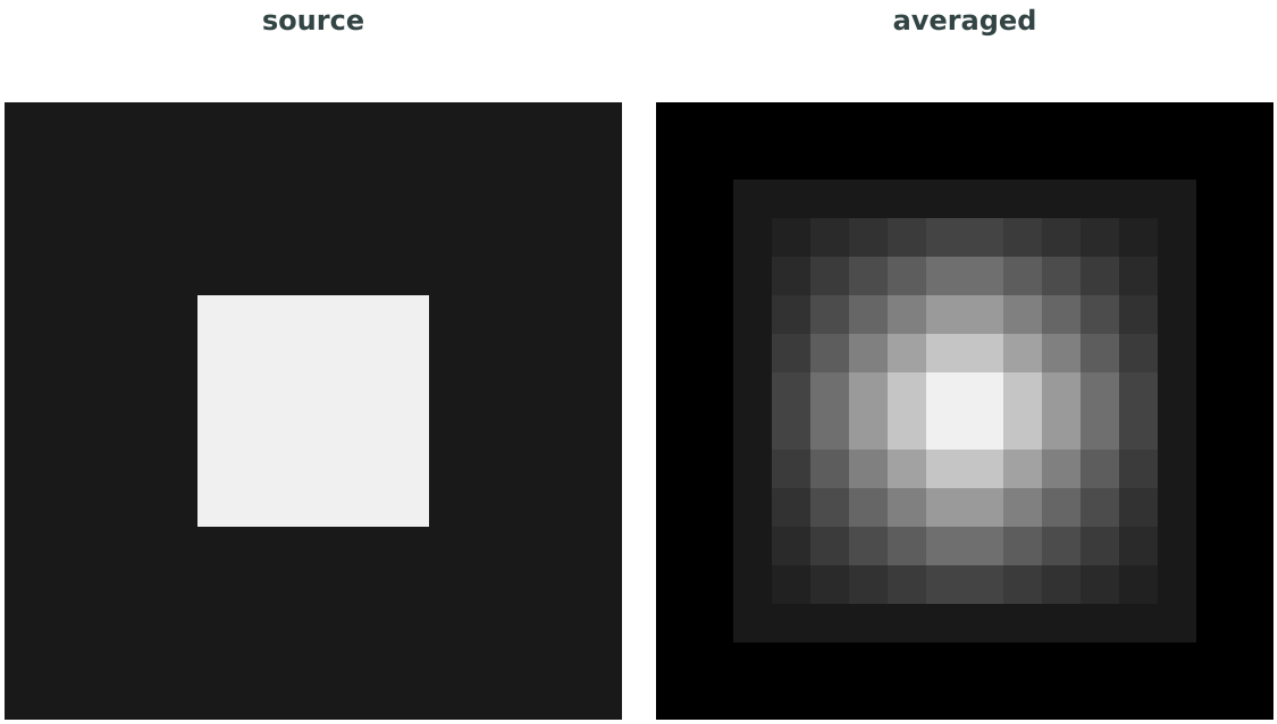
A self-contained neighborhood average. The source raster has a sharp central square, while the result spreads those values across neighboring pixels. The source uses a non-NODATA background so `nodatamode ignore` averages background values with the square values, instead of treating the background as absent.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(16, 16, 0, 16, 1, -1, 0, 0, 0),
            1, '8BUI', 25, NULL
        ),
        1, 6, 6,
        ARRAY[
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240]
        ]::double precision[][]
    ) AS rast
), variants AS (
    SELECT 'source' AS title, rast AS rendered FROM source
    UNION ALL
    SELECT 'averaged', ST_SetBandNoDataValue(ST_MapAlgebraFctNgb(
        rast,
        1,
        '8BUI',
        2,
        2,
```

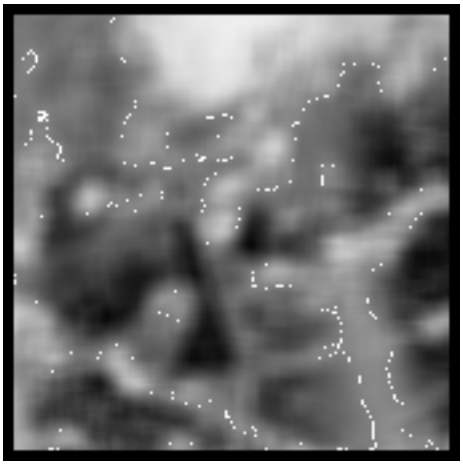
```
        'ST_Mean4ma(float[][],text,text[])'::regprocedure,
        'ignore',
        NULL
    ), NULL) FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Figure



Neighborhood averaging.

Första bandet av vår raster



Averaged result

Se även

ST_MapAlgebraFct, ST_MapAlgebraExpr, ST_Rescale

11.12.12 ST_Reclass

ST_Reclass — Skapar ett nytt raster som består av bandtyper som omklassificerats från originalet. Nband är det band som ska ändras. Om nband inte anges antas värdet vara 1. Alla andra band returneras oförändrade. Användningsfall: konvertera ett 16BUI-band till ett 8BUI och så vidare för enklare rendering som visningsbara format.

Synopsis

raster **ST_Reclass**(raster rast, integer nband, text reclassexpr, text pixeltype, double precision nodataval=NULL);
raster **ST_Reclass**(raster rast, reclassarg[] VARIADIC reclassargset);
raster **ST_Reclass**(raster rast, text reclassexpr, text pixeltype);

Description

Skapar ett nytt raster som bildas genom att tillämpa en omklassificeringsoperation som definieras av `reclassexpr` på indatarastret(`rast`). Se reclassarg för en beskrivning av omklassificeringsuttryck. Om inget `band` anges antas band 1.

Det nya rastret kommer att ha samma georeferens, bredd och höjd som det ursprungliga rastret. Banden i det nya rastret har pixeltyp av `pixeltype`. Om `reclassargset` specificeras definierar varje `reclassarg` typen av målband. Band som inte har angetts returneras oförändrade.

Tillgänglighet: 2.0.0

Exempel

Basic.

Skapa ett nytt raster från originalet där band 2 konverteras från 8BUI till 4BUI och alla värden från 101-254 sätts till nodatavärde.

Code

```
ALTER TABLE dummy_rast ADD COLUMN reclass_rast raster;
UPDATE dummy_rast SET reclass_rast = ST_Reclass(rast, 2, '0-87:1-10, 88-100:11-15,  ↩
    101-254:0-0', '4BUI', 0) WHERE rid = 2;

SELECT i as col, j as row, ST_Value(rast, 2, i, j) As origval,
    ST_Value(reclass_rast, 2, i, j) As reclassval,
    ST_Value(reclass_rast, 2, i, j, false) As reclassval_include_nodata
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
col | row | origval | reclassval | reclassval_include_nodata
-----+-----+---------+------------+--------------------------
   1 |   1 |      78 |          9 |                         9
   2 |   1 |      98 |         14 |                        14
   3 |   1 |     122 |            |                         0
   1 |   2 |      96 |         14 |                        14
   2 |   2 |     118 |            |                         0
   3 |   2 |     180 |            |                         0
   1 |   3 |      99 |         15 |                        15
   2 |   3 |     112 |            |                         0
   3 |   3 |     169 |            |                         0
```

Advanced using multiple reclassargs.

Skapa ett nytt raster från originalet där band 1,2,3 konverteras till 1BB,4BUI, 4BUI respektive och omklassificeras. Observera att detta använder det variadiska `reclassarg`-argumentet som kan ta ett obegränsat antal reclassargs som indata (teoretiskt sett så många band som du har)

Code

```
UPDATE dummy_rast SET reclass_rast =
    ST_Reclass(
        rast,
        ROW(2, '0-87]:1-10, (87-100]:11-15, (101-254]:0-0', '4BUI', NULL)::reclassarg,
        ROW(1, '0-253]:1, 254:0', '1BB', NULL)::reclassarg,
        ROW(3, '0-70]:1, (70-86:2, [86-150):3, [150-255:4', '4BUI', NULL)::reclassarg
        ) WHERE rid = 2;

SELECT i as col, j as row, ST_Value(rast, 1, i, j) As ov1, ST_Value(reclass_rast, 1, i, j)  ↩
    As rv1,
    ST_Value(rast, 2, i, j) As ov2, ST_Value(reclass_rast, 2, i, j) As rv2,
    ST_Value(rast, 3, i, j) As ov3, ST_Value(reclass_rast, 3, i, j) As rv3
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
col | row | ov1 | rv1 | ov2 | rv2 | ov3 | rv3
----+-----+-----+-----+-----+-----+-----+-----
  1 |   1 | 253 |   1 |  78 |   9 |  70 |   1
  2 |   1 | 254 |   0 |  98 |  14 |  86 |   3
  3 |   1 | 253 |   1 | 122 |   0 | 100 |   3
  1 |   2 | 253 |   1 |  96 |  14 |  80 |   2
  2 |   2 | 254 |   0 | 118 |   0 | 108 |   3
  3 |   2 | 254 |   0 | 180 |   0 | 162 |   4
  1 |   3 | 250 |   1 |  99 |  15 |  90 |   3
  2 |   3 | 254 |   0 | 112 |   0 | 108 |   3
  3 |   3 | 254 |   0 | 169 |   0 | 175 |   4
```

Advanced Map a single band 32BF raster to multiple viewable bands.

Skapa ett nytt 3-bands (8BUI,8BUI,8BUI synligt raster) från ett raster som bara har ett 32bf-band

Code

```
ALTER TABLE wind ADD COLUMN rast_view raster;
UPDATE wind
    set rast_view = ST_AddBand(
        NULL,
        ARRAY[
    ST_Reclass(rast, 1, '0.1-10]:1-10,9-10]:11,(11-33:0'::text, '8BUI'::text, 0),
    ST_Reclass(rast, 1, '11-33):0-255,[0-32:0,(34-1000:0'::text, '8BUI'::text, 0),
    ST_Reclass(rast, 1, '0-32]:0,(32-100:100-255'::text, '8BUI'::text, 0)
    ]
    );
```

Se även

ST_AddBand, ST_Band, ST_BandPixelType, ST_MakeEmptyRaster, reclassarg, ST_Value

11.12.13 ST_ReclassExact

ST_ReclassExact — Skapar ett nytt raster som består av band som omklassificerats från originalbandet med hjälp av en 1:1-mappning från värden i originalbandet till nya värden i destinationsbandet.

Synopsis

raster **ST_ReclassExact**(raster rast, double precision[] inputvalues, double precision[] outputvalues, integer bandnumber=1, text pixeltype=32BF, double precision nodatavalue=NULL);

Description

Skapar ett nytt raster som bildas genom att tillämpa en omklassificeringsoperation som definieras av matriserna `inputvalues` och `outputvalues`. Pixelvärden som finns i input-arrayen mappas till motsvarande värde i output-arrayen. Alla andra pixelvärden mappas till `nodatavärdet`.

Pixeltypen för utdata är som standard float, men kan specificeras med hjälp av parametern `pixeltype`. Om inget `bandnumber` anges antas band 1.

Det nya rastret kommer att ha samma georeferens, bredd och höjd som det ursprungliga rastret. Band som inte anges returneras oförändrade.

Tillgänglighet: 3.6.0

Exempel: Grundläggande

Skapa ett litet raster och mappa dess pixlar till nya värden.

Create a raster with four pixels: first row 1 2, second row 3 4.

Code

```
CREATE TABLE reclassexact (
    id integer,
    rast raster
);
INSERT INTO reclassexact (id, rast)
```

```
SELECT 1, ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(2, -- width in pixels
        2, -- height in pixels
        0, -- upper-left x-coordinate
        0, -- upper-left y-coordinate
        1, -- pixel size in x-direction
        -1, -- pixel size in y-direction (negative for north-up)
        0, -- skew in x-direction
        0, -- skew in y-direction
        4326  -- SRID (e.g., WGS 84)
      ),
      '32BUI'::text, -- pixel type (e.g., '32BF' for float, '8BUI' for unsigned 8-bit int)
      0.0, -- initial value for the band (e.g., 0.0 or a no-data value)
      -99            -- nodatavalue
    ),
    1, -- band number (usually 1 for single-band rasters)
    1, -- x origin for setting values (usually 1)
    1, -- y origin for setting values (usually 1)
    ARRAY[
      ARRAY[1, 2],
      ARRAY[3, 4]
    ]::double precision[][] -- 2D array of values
  );
```

Reclass the values to new values and dump the values of the new raster for display.

Code

```
WITH rc AS (
  SELECT ST_ReclassExact(rast, -- input raster
    ARRAY[4, 3, 2, 1], -- input map
    ARRAY[14, 13, 12, 11], -- output map
    1, -- band number to remap
    '32BUI'               -- output raster pixel type
    ) AS rast
  FROM reclassexact
  WHERE id = 1
  )
SELECT 'rce-1', (ST_DumpValues(rc.rast)).*
FROM rc;
```

Se även

ST_Reclass, ST_AddBand, ST_Band, ST_MakeEmptyRaster

11.12.14 ST_Union

ST_Union — Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.

Synopsis

raster **ST_Union**(setof raster rast);
raster **ST_Union**(setof raster rast, unionarg[] unionargset);
raster **ST_Union**(setof raster rast, integer nband);
raster **ST_Union**(setof raster rast, text uniontype);
raster **ST_Union**(setof raster rast, integer nband, text uniontype);

Description

Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av minst ett band. Det resulterande rastrets utsträckning är hela uppsättningens utsträckning. Vid intersektion definieras det resulterande värdet av `uniontype`, som är något av följande: LAST (standard), FIRST, MIN, MAX, COUNT, SUM, MEAN, RANGE.

Notera
För att rasters ska kunna sammanfogas måste de alla ha samma inriktning. Använd ST_SameAlignment och ST_NotSameAlignmentReason för mer information och hjälp. Ett sätt att åtgärda problem med alignment är att använda ST_Resample och använda samma referensraster för alignment.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Förbättrad hastighet (helt C-baserad).

Tillgänglighet: 2.1.0 Varianten ST_Union(rast, unionarg) introducerades.

Förbättrad: 2.1.0 ST_Union(rast) (variant 1) förenar alla band i alla ingående raster. Tidigare versioner av PostGIS antog det första bandet.

Förbättrad: 2.1.0 ST_Union(rast, uniontype) (variant 4) förenar alla band i alla ingående raster.

Exempel

Reconstitute a single band chunked raster tile.

This creates a single band from first band of raster tiles. That form the original file system tile.

Code

```
SELECT filename, ST_Union(rast, 1) As file_rast
FROM sometable WHERE filename IN('dem01', 'dem02') GROUP BY filename;
```

Return a multi-band raster that is the union of tiles intersecting geometry.

This creates a multi band raster collecting all the tiles that intersect a line. Note that In 2.0, this would have just returned a single band raster. , new union works on all bands by default. This is equivalent to unionarg: ARRAY[ROW(1, 'LAST'), ROW(2, 'LAST'), ROW(3, 'LAST')]::unionarg[].

Code

```
SELECT ST_Union(rast)
FROM aerials.boston
WHERE ST_Intersects(rast, ST_GeomFromText('LINESTRING(230486 887771,230500 88772)', 26986) ↩
    );
```

Select and reorder bands while unioning tiles that intersect a geometry.

Här använder vi den längre syntaxen om vi bara vill ha en delmängd av banden eller om vi vill ändra ordningen på banden

This creates a multi band raster collecting all the tiles that intersect a line.

Code

```
SELECT ST_Union(rast, ARRAY[ROW(2, 'LAST'), ROW(1, 'LAST'), ROW(3, 'LAST')]::unionarg[])
FROM aerials.boston
WHERE ST_Intersects(rast, ST_GeomFromText('LINESTRING(230486 887771,230500 88772)', 26986) ↩
    );
```

Se även

unionarg, ST_Envelope, ST_ConvexHull, ST_Clip, ST_Union

11.13 Inbyggda Map Algebra återuppringningsfunktioner

11.13.1 ST_Distinct4ma

ST_Distinct4ma — Rasterbearbetningsfunktion som beräknar antalet unika pixelvärden i ett grannskap.

Synopsis

float8 **ST_Distinct4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Distinct4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna antalet unika pixelvärden i ett grannskap av pixlar.

Notera
Variant 1 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebraFctNgb.

Notera
Variant 2 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebra (callback function version).

Varning
Användning av variant 1 avråds eftersom ST_MapAlgebraFctNgb har utgått från och med 2.1.0.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Tillägg av variant 2

Exempel

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Distinct4ma(float[][],text,text ↩
    [])'::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |        3
(1 row)
```

Se även

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Sum4ma ST_Mean4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.2 ST_InvDistWeight4ma

ST_InvDistWeight4ma — Rasterbearbetningsfunktion som interpolerar en pixels värde från pixelns närområde.

Synopsis

double precision **ST_InvDistWeight4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna ett interpolerat värde för en pixel med hjälp av den omvända avståndsviktade metoden.

Det finns två valfria parametrar som kan skickas via `userargs`. Den första parametern är den effektfaktor (variabel k i ekvationen nedan) mellan 0 och 1 som används i den inversa avståndsviktade ekvationen. Om den inte anges är standardvärdet 1. Den andra parametern är den viktprocent som endast tillämpas när värdet för den intressanta pixeln ingår i det interpolerade värdet från grannskapet. Om inget anges och den intressanta pixeln har ett värde, returneras detta värde.

The basic inverse-distance weighted estimate is: z = (Σ_i z_i / d_i^k) / (Σ_i 1 / d_i^k) Here z_i is a neighboring pixel value, d_i is its distance from the pixel being estimated, and k is a real-valued power factor between 0 and 1.

Notera
Denna funktion är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebra (callback function version).

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', NULL::double precision, -9999),
        1, 1, 1, ARRAY[
            [10, 10, 10],
            [10, NULL, 20],
            [20, 20, 20]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(
    ST_MapAlgebra(
        rast,
        1,
        'ST_InvDistWeight4ma(double precision[][][],integer[][],text[])'::regprocedure,
        '32BF',
        'FIRST',
        NULL,
        1,
        1
    ),
    2, 2
) AS interpolated_value
FROM foo;
```

Output

```
15
```

Se även

ST_MapAlgebra (callback function version), ST_MinDist4ma

11.13.3 ST_Max4ma

ST_Max4ma — Rasterbearbetningsfunktion som beräknar det maximala pixelvärdet i ett grannskap.

Synopsis

float8 **ST_Max4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Max4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna det maximala pixelvärdet i ett grannskap av pixlar.

För variant 2 kan ett ersättningsvärde för NODATA-pixlar anges genom att skicka det värdet till userargs.

Notera
Variant 1 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebraFctNgb.

Notera
Variant 2 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebra (callback function version).

Varning
Användning av variant 1 avråds eftersom ST_MapAlgebraFctNgb har utgått från och med 2.1.0.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Tillägg av variant 2

Exempel

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Max4ma(float[][],text,text[])':: ↩
    regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |      254
(1 row)
```

Se även

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Sum4ma, ST_Mean4m
ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.4 ST_Mean4ma

ST_Mean4ma — Rasterbearbetningsfunktion som beräknar det genomsnittliga pixelvärdet i ett grannskap.

Synopsis

float8 **ST_Mean4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Mean4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna det genomsnittliga pixelvärdet i ett grannskap av pixlar.

För variant 2 kan ett ersättningsvärde för NODATA-pixlar anges genom att skicka det värdet till userargs.

Notera
Variant 1 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebraFctNgb.

Notera
Variant 2 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebra (callback function version).

Varning
Användning av variant 1 avråds eftersom ST_MapAlgebraFctNgb har utgått från och med 2.1.0.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Tillägg av variant 2

Exempel

Variant 1.

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, '32BF', 1, 1, 'ST_Mean4ma(float[][],text,text[])' ↩
    ::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid |     st_value
----+------------------
   2 | 253.222229003906
(1 row)
```

Variant 2.

Code

```
SELECT
  rid,
  ST_Value(
      ST_MapAlgebra(rast, 1, 'ST_Mean4ma(double precision[][][], integer[][], text[])':: ↩
    regprocedure, '32BF', 'FIRST', NULL, 1, 1),
      2,
      2)
FROM dummy_rast
 WHERE rid = 2;
```

Output

```
rid |     st_value
-----+------------------
   2 | 253.222229003906
(1 row)
```

Se även

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Sum4ma ST_Range4ma, ST_StdDev4ma

11.13.5 ST_Min4ma

ST_Min4ma — Rasterbearbetningsfunktion som beräknar det lägsta pixelvärdet i ett grannskap.

Synopsis

float8 **ST_Min4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Min4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna det lägsta pixelvärdet i ett grannskap av pixlar.

För variant 2 kan ett ersättningsvärde för NODATA-pixlar anges genom att skicka det värdet till userargs.

Notera
Variant 1 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebraFctNgb.

Notera
Variant 2 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebra (callback function version).

Varning
Användning av variant 1 avråds eftersom ST_MapAlgebraFctNgb har utgått från och med 2.1.0.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Tillägg av variant 2

Exempel

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Min4ma(float[][],text,text[])':: ↩
    regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |      250
(1 row)
```

Se även

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Max4ma, ST_Sum4ma, ST_Mean4m
ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.6 ST_MinDist4ma

ST_MinDist4ma — Rasterbehandlingsfunktion som returnerar det minsta avståndet (i antal pixlar) mellan den intressanta pixeln och en angränsande pixel med värde.

Synopsis

double precision **ST_MinDist4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Returnerar det kortaste avståndet (i antal pixlar) mellan den intressanta pixeln och den närmaste pixeln med värde i grannskapet.

Notera
Syftet med den här funktionen är att tillhandahålla en informativ datapunkt som hjälper till att dra slutsatser om användbarheten av den intressanta pixelns interpolerade värde från ST_InvDistWeight4ma. Den här funktionen är särskilt användbar när grannskapet är glest befolkat.

Notera
Denna funktion är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebra (callback function version).

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', NULL::double precision, -9999),
        1, 1, 1, ARRAY[
            [10, 10, 10],
            [10, NULL, 20],
            [20, 20, 20]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(
    ST_MapAlgebra(
        rast,
        1,
        'ST_MinDist4ma(double precision[][][],integer[][],text[])'::regprocedure,
        '32BF',
        'FIRST',
        NULL,
        1,
        1
    ),
    2, 2
) AS min_distance
FROM foo;
```

Output

```
1
```

Se även

ST_MapAlgebra (callback function version), ST_InvDistWeight4ma

11.13.7 ST_Range4ma

ST_Range4ma — Rasterbearbetningsfunktion som beräknar intervallet för pixelvärden i ett område.

Synopsis

float8 **ST_Range4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Range4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna intervallet av pixelvärden i ett grannskap av pixlar.

För variant 2 kan ett ersättningsvärde för NODATA-pixlar anges genom att skicka det värdet till userargs.

Notera
Variant 1 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebraFctNgb.

Notera
Variant 2 är en specialiserad återuppringningsfunktion för användning som återuppringnings-parameter till ST_MapAlgebra (callback function version).

Varning
Användning av variant 1 avråds eftersom ST_MapAlgebraFctNgb har utgått från och med 2.1.0.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Tillägg av variant 2

Exempel

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Range4ma(float[][],text,text[])' ↩
    ::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |        4
(1 row)
```

Se även

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Sum4ma ST_Mean4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.8 ST_StdDev4ma

ST_StdDev4ma — Rasterbearbetningsfunktion som beräknar standardavvikelsen för pixelvärden i ett grannskap.

Synopsis

float8 **ST_StdDev4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_StdDev4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna standardavvikelsen för pixelvärden i ett grannskap av pixlar.

Notera
Variant 1 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebraFctNgb.

Notera
Variant 2 är en specialiserad återuppringningsfunktion för användning som återuppringningsparameter till ST_MapAlgebra (callback function version).

Varning
Användning av variant 1 avråds eftersom ST_MapAlgebraFctNgb har utgått från och med 2.1.0.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Tillägg av variant 2

Exempel

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, '32BF', 1, 1, 'ST_StdDev4ma(float[][],text,text ↩
    [])'::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid |     st_value
-----+------------------
   2 | 1.30170822143555
(1 row)
```

Se även

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Sum4ma ST_Mean4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.9 ST_Sum4ma

ST_Sum4ma — Rasterbearbetningsfunktion som beräknar summan av alla pixelvärden i ett grannskap.

Synopsis

float8 **ST_Sum4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Sum4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Beräkna summan av alla pixelvärden i ett grannskap av pixlar.

För variant 2 kan ett ersättningsvärde för NODATA-pixlar anges genom att skicka det värdet till userargs.

Notera
Variant 1 är en specialiserad återuppringningsfunktion för användning som återuppringnings-parameter till ST_MapAlgebraFctNgb.

Notera
Variant 2 är en specialiserad återuppringningsfunktion för användning som återuppringnings-parameter till ST_MapAlgebra (callback function version).

Varning
Användning av variant 1 avråds eftersom ST_MapAlgebraFctNgb har utgått från och med 2.1.0.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Tillägg av variant 2

Exempel

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, '32BF', 1, 1, 'ST_Sum4ma(float[][],text,text[])' ↩
    ::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |     2279
(1 row)
```

Se även

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Mean4m
ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.14 Rasterbearbetning: DEM (höjdsystem)

11.14.1 ST_Aspect

ST_Aspect — Returnerar aspekten (i grader som standard) för ett höjdrasterband. Användbart för analys av terräng.

Synopsis

raster **ST_Aspect**(raster rast, integer band=1, text pixeltype=32BF, text units=DEGREES, boolean interpolate_nodata=FALSE);
raster **ST_Aspect**(raster rast, integer band, raster customextent, text pixeltype=32BF, text units=DEGREES, boolean interpolate_nodata=FALSE);

Description

Returnerar aspekten (i grader som standard) för ett höjdrasterband. Utnyttjar kartalgebra och tillämpar aspektekvationen på närliggande pixlar.

`units` anger enheten för aspekten. Möjliga värden är: RADIANS, DEGREES (standard).

När `units` = RADIANS är värdena mellan 0 och 2 * pi radianer mätt medurs från norr.

När `units` = DEGREES är värdena mellan 0 och 360 grader mätt medurs från norr.

Om pixelns lutning är noll är pixelns aspekt -1.

Notera
Mer information om Slope, Aspect och Hillshade finns i ESRI - How hillshade works och ERDAS Field Guide - Aspect Images.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Använder ST_MapAlgebra() och har lagt till en valfri funktionsparameter för `interpolate_`

Ändrad: 2.1.0 I tidigare versioner var returvärdena i radianer. Nu är returvärdena som standard grader

Exempel

Variant 1.

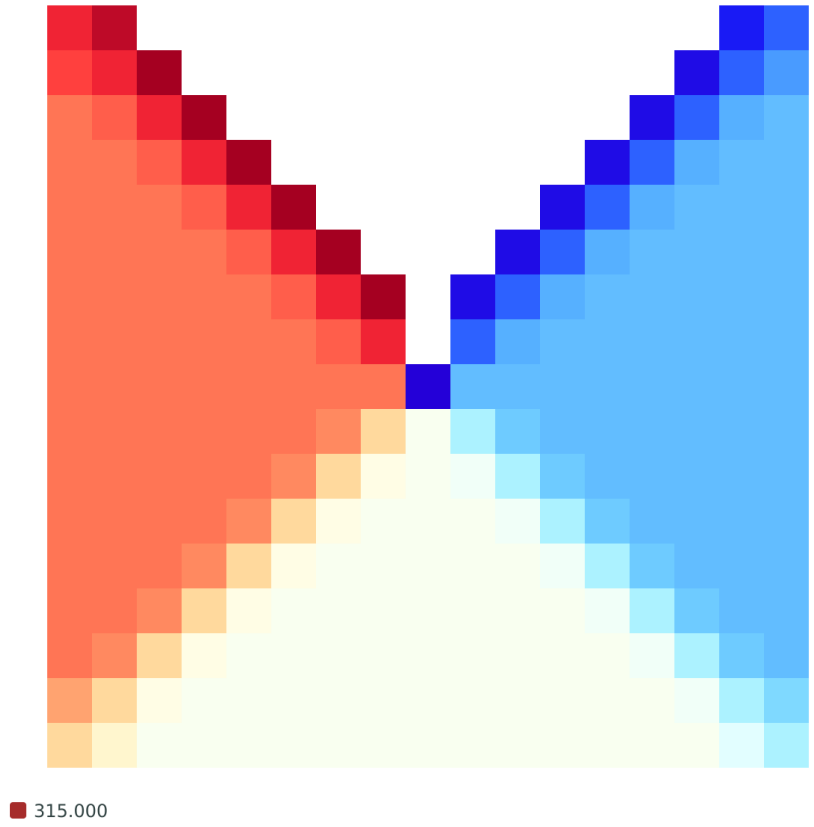
This symmetric peak highlights the expected compass directions at the corners and the `-1` aspect at the summit. The generated PNG shows the full aspect surface without relying on a manual illustration. The generated source raster is large enough for the PNG to show the directional field around the summit.

Code

```
WITH pixels AS (
    SELECT
        x,
        y,
        (20 - greatest(abs(x - 9), abs(y - 9)))::double precision AS value
    FROM generate_series(1, 17) AS y
```

```
    CROSS JOIN generate_series(1, 17) AS x
), rows AS (
    SELECT
        y,
        array_agg(value ORDER BY x) AS row_values
    FROM pixels
    GROUP BY y
), source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(17, 17, 0, 0, 1, -1, 0, 0, 0),
            1, '32BF', 0, -9999
        ),
        1, 1, 1,
        array_agg(row_values ORDER BY y)::double precision[][]
    ) AS rast
    FROM rows
), aspect AS (
    SELECT ST_Aspect(rast, 1, '32BF') AS rast
    FROM source
)
SELECT
    round(ST_Value(rast, 1, 1, 1)::numeric, 3) AS northwest_degrees,
    round(ST_Value(rast, 1, 9, 9)::numeric, 3) AS summit_degrees,
    round(ST_Value(rast, 1, 17, 17)::numeric, 3) AS southeast_degrees,
    ST_AsPNG(ST_ColorMap(rast, 1, 'bluered')) AS image
FROM aspect;
```

Figure



Variant 2.

Complete example of tiles of a coverage.

Code

```
WITH foo AS (
    SELECT ST_Tile(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(6, 6, 0, 0, 1, -1, 0, 0, 0),
                1, '32BF', 0, -9999
            ),
            1, 1, 1, ARRAY[
                [1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 2, 1],
                [1, 2, 2, 3, 3, 1],
                [1, 1, 3, 2, 1, 1],
                [1, 2, 2, 1, 2, 1],
                [1, 1, 1, 1, 1, 1]
            ]::double precision[]
        ),
        2, 2
    ) AS rast
)
SELECT
    t1.rast,
    ST_Aspect(ST_Union(t2.rast), 1, t1.rast)
FROM foo t1
CROSS JOIN foo t2
WHERE ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rast;
```

Se även

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_HillShade, ST_Slope

11.14.2 ST_HillShade

ST_HillShade — Returnerar den hypotetiska belysningen för ett höjdrasterband med hjälp av angivna indata för azimut, höjd, ljusstyrka och skala.

Synopsis

raster **ST_HillShade**(raster rast, integer band=1, text pixeltype=32BF, double precision azimuth=315, double precision altitude=45, double precision max_bright=255, double precision scale=1.0, boolean interpolate_nodata=FALSE);
raster **ST_HillShade**(raster rast, integer band, raster customextent, text pixeltype=32BF, double precision azimuth=315, double precision altitude=45, double precision max_bright=255, double precision scale=1.0, boolean interpolate_nodata=FALSE);

Description

Returnerar den hypotetiska belysningen av ett höjdrasterband med hjälp av indatavärdena azimut, höjd, ljusstyrka och skala. Använder kartalgebra och tillämpar ekvationen för kullskugga på närliggande pixlar. Returpixelvärdena ligger mellan 0 och 255.

`azimut` är ett värde mellan 0 och 360 grader mätt medurs från norr.

`höjden` är ett värde mellan 0 och 90 grader där 0 grader är vid horisonten och 90 grader är rakt ovanför.

`max_bright` är ett värde mellan 0 och 255 med 0 som ingen ljusstyrka och 255 som maximal ljusstyrka.

`skala` är förhållandet mellan vertikala enheter och horisontella. För Feet:LatLon används scale=370400, för Meter:LatLon används scale=111120.

Om `interpolate_nodata` är TRUE interpoleras värdena för NODATA-pixlar från indatarastret med ST_InvDistWeight4ma innan belysningen för hillshade beräknas.

Notera
Mer information om Hillshade finns i avsnittet Så fungerar Hillshade.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Använder ST_MapAlgebra() och har lagt till en valfri funktionsparameter för `interpolate_`

Ändrad: 2.1.0 I tidigare versioner uttrycktes azimut och höjd i radianer. Nu uttrycks azimut och höjd i grader

Exempel

Variant 1.

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 1, 1, 1],
            [1, 2, 2, 2, 1],
            [1, 2, 3, 2, 1],
            [1, 2, 2, 2, 1],
            [1, 1, 1, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_DumpValues(ST_HillShade(rast, 1, '32BF'))
FROM foo

                                                
```

Code

```
WITH foo AS (
    SELECT ST_Tile(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(6, 6, 0, 0, 1, -1, 0, 0, 0),
                1, '32BF', 0, -9999
            ),
            1, 1, 1, ARRAY[
                [1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 2, 1],
                [1, 2, 2, 3, 3, 1],
                [1, 1, 3, 2, 1, 1],
                [1, 2, 2, 1, 2, 1],
                [1, 1, 1, 1, 1, 1]
            ]::double precision[]
        ),
        2, 2
    ) AS rast
)
SELECT
    t1.rast,
    ST_HillShade(ST_Union(t2.rast), 1, t1.rast)
FROM foo t1
CROSS JOIN foo t2
WHERE ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rast;
```

Se även

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_Aspect, ST_Slope

11.14.3 ST_Roughness

ST_Roughness — Returnerar ett raster med den beräknade öjämnhetenför en DEM.

Synopsis

raster **ST_Roughness**(raster rast, integer nband, raster customextent, text pixeltype=”32BF”, boolean interpolate_nodata=FALSE);

Description

Beräknar grovheten”i en DEM genom att subtrahera maximum från minimum för ett visst område.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
```

```
            [1, 1, 2, 3, 3],
            [1, 2, 4, 15, 3],
            [2, 4, 10, 7, 5],
            [3, 5, 7, 6, 4],
            [3, 3, 5, 4, 4]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(ST_Roughness(rast, 1, '32BF'), 3, 3) AS roughness
FROM foo;
```

Output

```
13
```

Se även

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Slope, ST_HillShade, ST_Aspect

11.14.4 ST_Slope

ST_Slope — Returnerar lutningen (i grader som standard) för ett höjdrasterband. Användbart för att analysera terräng.

Synopsis

raster **ST_Slope**(raster rast, integer nband=1, text pixeltype=32BF, text units=DEGREES, double precision scale=1.0, boolean interpolate_nodata=FALSE);
raster **ST_Slope**(raster rast, integer nband, raster customextent, text pixeltype=32BF, text units=DEGREES, double precision scale=1.0, boolean interpolate_nodata=FALSE);

Description

Returnerar lutningen (i grader som standard) för ett höjdrasterband. Utnyttjar kartalgebra och tillämpar lutningsekvationen på angränsande pixlar.

`units` anger enheten för lutningen. Möjliga värden är: RADIANS, DEGREES (standard), PERCENT.

`skala` är förhållandet mellan vertikala enheter och horisontella. För Feet:LatLon används scale=370400, för Meter:LatLon används scale=111120.

Om `interpolate_nodata` är TRUE interpoleras värden för NODATA-pixlar från indatarastret med ST_InvDistWeight4ma innan ytans lutning beräknas.

Notera
Mer information om Slope, Aspect och Hillshade finns i ESRI - How hillshade works och ERDAS Field Guide - Slope Images.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 Använder ST_MapAlgebra() och har lagt till valfria funktionsparametrar för `units`, `scale`, `interpolate_nodata`

Ändrad: 2.1.0 I tidigare versioner var returvärdena i radianer. Nu är returvärdena som standard grader

Exempel

Variant 1.

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 1, 1, 1],
            [1, 2, 2, 2, 1],
            [1, 2, 3, 2, 1],
            [1, 2, 2, 2, 1],
            [1, 1, 1, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_DumpValues(ST_Slope(rast, 1, '32BF'))
FROM foo

                              st_dumpvalues
```

Output

```
----------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------
  ↩
    (1,"{{10.0249881744385,21.5681285858154,26.5650520324707,21.5681285858154,10.0249881744385},{21.5681285858154,35.2643890380808

{26.5650520324707,36.8698959350586,0,36.8698959350586,26.5650520324707},{21.5681285858154,35.2643890380859,36.8698959350586,35.26

5681285858154,26.5650520324707,21.5681285858154,10.0249881744385}}")
(1 row)
```

Variant 2.

Complete example of tiles of a coverage.

Code

```
WITH foo AS (
    SELECT ST_Tile(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(6, 6, 0, 0, 1, -1, 0, 0, 0),
                1, '32BF', 0, -9999
            ),
            1, 1, 1, ARRAY[
                [1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 2, 1],
                [1, 2, 2, 3, 3, 1],
                [1, 1, 3, 2, 1, 1],
                [1, 2, 2, 1, 2, 1],
                [1, 1, 1, 1, 1, 1]
            ]::double precision[]
        ),
        2, 2
    ) AS rast
)
SELECT
    t1.rast,
    ST_Slope(ST_Union(t2.rast), 1, t1.rast)
FROM foo t1
CROSS JOIN foo t2
```

```
WHERE ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rast;
```

Se även

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_HillShade, ST_Aspect

11.14.5 ST_TPI

ST_TPI — Returnerar ett raster med det beräknade topografiska positionsindexet.

Synopsis

raster **ST_TPI**(raster rast, integer nband, raster customextent, text pixeltype=”32BF”, boolean interpolate_nodata=FALSE);

Description

Beräknar det topografiska positionsindexet, som definieras som det fokala medelvärdet med radien ett minus den centrala cellen.

Notera
Denna funktion stöder endast en fokal medelradie på ett.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 2, 3, 3],
            [1, 2, 4, 15, 3],
            [2, 4, 10, 7, 5],
            [3, 5, 7, 6, 4],
            [3, 3, 5, 4, 4]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(ST_TPI(rast, 1, '32BF'), 3, 3) AS tpi
FROM foo;
```

Output

```
3.75
```

Se även

ST_MapAlgebra (callback function version), ST_TRI, ST_Roughness, ST_Slope, ST_HillShade, ST_Aspect

11.14.6 ST_TRI

ST_TRI — Returnerar ett raster med det beräknade Terrain Ruggedness Index.

Synopsis

raster **ST_TRI**(raster rast, integer nband, raster customextent, text pixeltype=”32BF”, boolean interpolate_nodata=FALSE);

Description

Terrain Ruggedness Index beräknas genom att jämföra en central pixel med dess grannar, ta de absoluta värdena av skillnaderna och beräkna medelvärdet av resultatet.

Notera
Denna funktion stöder endast en fokal medelradie på ett.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 2, 3, 3],
            [1, 2, 4, 15, 3],
            [2, 4, 10, 7, 5],
            [3, 5, 7, 6, 4],
            [3, 3, 5, 4, 4]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(ST_TRI(rast, 1, '32BF'), 3, 3) AS tri
FROM foo;
```

Output

```
5
```

Se även

ST_MapAlgebra (callback function version), ST_Roughness, ST_TPI, ST_Slope, ST_HillShade, ST_Aspect

11.14.7 ST_InterpolateRaster

ST_InterpolateRaster — Interpolerar en rutnätsyta baserat på en indatauppsättning av 3D-punkter, med hjälp av X- och Y-värdena för att positionera punkterna i rutnätet och punkternas Z-värde som ytans höjd.

Synopsis

raster **ST_InterpolateRaster**(geometry input_points, text algorithm_options, raster template, integer template_band_num=1);

Description

Interpolerar en yta i rutnät baserat på en indatauppsättning med 3D-punkter, med hjälp av X- och Y-värdena för att positionera punkterna i rutnätet och punkternas Z-värde som ytans höjd. Det finns fem interpolationsalgoritmer tillgängliga: invers avstånd, invers avstånd närmaste granne, glidande medelvärde, närmaste granne och linjär interpolation. Se dokumentationen för gdal_grid för mer information om algoritmerna och deras parametrar. Mer information om hur interpoleringar beräknas finns i GDAL Grid Tutorial.

Inmatningsparametrar är:

input_points De punkter som ska driva interpoleringen. Alla geometrier med Z-värden är acceptabla, alla punkter i indata kommer att användas.

algorithm_options En sträng som definierar algoritmen och algoritmalternativen, i det format som används av gdal_grid. Om du t.ex. vill göra en interpolering med omvänt avstånd och en utjämning på 2 använder du ”invdist:smoothing=2.0”

template En rastermall för att styra geometrin i utdatarastret. Bredd, höjd, pixelstorlek, spatial utsträckning och pixeltyp kommer att läsas från denna mall.

template_band_num Som standard används det första bandet i mallrastret för att driva utdatarastret, men det kan justeras med den här parametern.

Tillgänglighet: 3.2.0

Exempel

Code

```
SELECT ST_InterpolateRaster('MULTIPOINT(10.5 9.5 1000,11.5 8.5 1000,10.5 8.5 500,11.5 9.5 ↩
    500)'::geometry,
    'invdist:smoothing:2.0',
    ST_AddBand(ST_MakeEmptyRaster(200, 400, 10, 10, 0.01, -0.005, 0, 0), '16BSI')
)
```

Se även

ST_Contour

11.14.8 ST_Contour

ST_Contour — Skapar en uppsättning vektorkonturer från det tillhandahållna rasterbandet med hjälp av GDAL-konturalgoritmen.

Synopsis

setof record **ST_Contour**(raster rast, integer bandnumber=1, double precision level_interval=100.0, double precision level_base=0.0, double precision[] fixed_levels=ARRAY[], boolean polygonize=false);

Description

Skapar en uppsättning vektorkonturer från det tillhandahållna rasterbandet med hjälp av GDAL-konturalgoritmen.

När parametern `fixed_levels` är en icke-tom array ignoreras parametrarna `level_interval` och `level_base`.

Inmatningsparametrar är:

`rast` Rastret för att generera konturen av

`bandnumber` Bandet för att generera konturen av

`level_interval` Höjdintervallet mellan genererade konturer

`level_base` Den bas”i förhållande till vilken konturintervallen tillämpas, normalt noll, men kan vara annorlunda. För att generera konturer på 10 m vid 5, 15, 25, ... skulle LEVEL_BASE vara 5.

`fixed_levels` Höjdintervallet mellan genererade konturer

`polygonize` Om `true`, kommer konturpolygoner att skapas i stället för polygonlinjer.

Returvärdena är en uppsättning poster med följande attribut:

`geom` Konturlinjens geometri.

`id` En unik identifierare som GDAL ger till konturlinjen.

`value` Det rastervärde som linjen representerar. För en höjd-DEM-indata skulle detta vara höjden på den utgående konturen.

Tillgänglighet: 3.2.0

Exempel

Code

```
WITH c AS (
SELECT (ST_Contour(rast, 1, fixed_levels => ARRAY[100.0, 200.0, 300.0])).*
FROM dem_grid WHERE rid = 1
)
SELECT geom, id, value
FROM c;
```

Se även

ST_InterpolateRaster

11.15 Rasterbearbetning: Raster till geometri

11.15.1 Box3D

Box3D — Returnerar box 3d-representationen av den omslutande boxen i rastret.

Synopsis

box3d **Box3D**(raster rast);

Description

Returnerar den ruta som representerar rastrets utsträckning.

Polygonen definieras av hörnpunkterna i den avgränsande boxen ((MINX, MINY),(MAXX, MAXY))

Ändrad: 2.0.0 I versioner före 2.0 brukade det finnas en box2d istället för box3d. Eftersom box2d är en föråldrad typ ändrades detta till box3d.

Exempel

Code

```
SELECT
    rid,
    Box3D(rast) AS rastbox
FROM dummy_rast;
```

Output

```
rid |        rastbox
----+------------------------------------------------
1   | BOX3D(0.5 0.5 0,20.5 60.5 0)
2   | BOX3D(3427927.75 5793243.5 0,3427928 5793244 0)
```

Se även

ST_Envelope

11.15.2 ST_ConvexHull

ST_ConvexHull — Returnerar rastrets konvexa skrovgeometri inklusive pixelvärden som är lika med BandNoDataValue. För regelbundet formade och icke snedställda raster ger detta samma resultat som ST_Envelope, så det är endast användbart för oregelbundet formade eller snedställda raster.

Synopsis

geometry **ST_ConvexHull**(raster rast);

Description

Returnerar rastrets konvexa skrovgeometri inklusive NoDataBandValue-bandpixlarna. För regelbundet formade och icke-skeva raster ger detta mer eller mindre samma resultat som ST_Envelope, så det är endast användbart för oregelbundet formade eller skeva raster.

Notera
ST_Envelope golvar koordinaterna och lägger därför till en liten buffert runt rastret så svaret är subtilt annorlunda än ST_ConvevexHull som inte golvar.

Exempel

See `doc/development/internals/raster-affine.md` for the scale, skew, and rotation terms that make the convex hull and envelope differ for skewed rasters.

For regular non-skewed rasters, the envelope and convex hull are more or less the same.

Code

```
SELECT ST_ConvexHull(rast) As convhull, ST_Envelope(rast) As env
FROM dummy_rast WHERE rid=1;
```

Output

```
convhull                        |               env
--------------------------------------------------------+-----------------------------------
 POLYGON((0.5 0.5,20.5 0.5,20.5 60.5,0.5 60.5,0.5 0.5)) | POLYGON((0 0,20 0,20 60,0 60,0 0))
```

Now we skew the raster. Note how the convex hull and envelope are now different.

Code

```
SELECT ST_ConvexHull(rast) As convhull, ST_Envelope(rast) As env
FROM (SELECT ST_SetRotation(rast, 0.1, 0.1) As rast
    FROM dummy_rast WHERE rid=1) As foo;
```

Output

```
convhull                        |               env
--------------------------------------------------------+-----------------------------------
 POLYGON((0.5 0.5,20.5 1.5,22.5 61.5,2.5 60.5,0.5 0.5)) | POLYGON((0 0,22 0,22 61,0 61,0 0))
```

Se även

ST_Envelope, ST_MinConvexHull, ST_ConvexHull, ST_AsText

11.15.3 ST_DumpAsPolygons

ST_DumpAsPolygons — Returnerar en uppsättning geomval (geom,val)-rader från ett givet rasterband. Om inget bandnummer anges är standardvärdet för bandnum 1.

Synopsis

setof geomval **ST_DumpAsPolygons**(raster rast, integer band_num=1, boolean exclude_nodata_value=TRU

Description

Detta är en SRF-funktion (set-returning function). Den returnerar en uppsättning geomval-rader, som bildas av en geometri (geom) och ett pixelbandvärde (val). Varje polygon är en sammanslagning av alla pixlar för det bandet som har samma pixelvärde som betecknas med val.

ST_DumpAsPolygon är användbar för polygonisering av raster. Det är motsatsen till en GROUP BY eftersom det skapar nya rader. Det kan t.ex. användas för att expandera ett enda raster till flera POLYGONER/MULTIPOLYGONER.

Ändrad 3.3.0, validering och fixering är inaktiverad för att förbättra prestanda. Kan resultera i ogiltiga geometrier.

Changed 3.7.0, the polygonization honours PostgreSQL interrupts so cancellations and statement timeouts halt processing promptly.

Tillgänglighet: Kräver GDAL 1.7 eller högre.

Notera
Om ett no data-värde har ställts in för ett band kommer pixlar med det värdet inte att returneras, utom i fallet med exclude_nodata_value=false.

Notera
Om du bara bryr dig om antalet pixlar med ett visst värde i ett raster går det snabbare att använda ST_ValueCount.

Notera
Detta skiljer sig från ST_PixelAsPolygons där en geometri returneras för varje pixel oavsett pixelvärde.

Exempel

This syntax uses a LATERAL join.

Code

```
SELECT val,geom As geomwkt
FROM (
SELECT dp.*
FROM dummy_rast, LATERAL ST_DumpAsPolygons(rast) AS dp
WHERE rid = 2
) As foo
WHERE val BETWEEN 249 and 251
ORDER BY val;
```

Output

```
val |                                                                    geomwkt
-----+-------------------------------------------------------------------------
 249 | POLYGON((3427927.95 5793243.95,3427927.95 5793243.85,3427928 5793243.85,
        3427928 5793243.95,3427927.95 5793243.95))
 250 | POLYGON((3427927.75 5793243.9,3427927.75 5793243.85,3427927.8 5793243.85,
        3427927.8 5793243.9,3427927.75 5793243.9))
 250 | POLYGON((3427927.8 5793243.8,3427927.8 5793243.75,3427927.85 5793243.75,
        3427927.85 5793243.8, 3427927.8 5793243.8))
 251 | POLYGON((3427927.75 5793243.85,3427927.75 5793243.8,3427927.8 5793243.8,
        3427927.8 5793243.85,3427927.75 5793243.85))
```

Se även

geomval, ST_AsRasterAgg, ST_Value, ST_Polygon, ST_ValueCount

11.15.4 ST_Envelope

ST_Envelope — Returnerar en polygonrepresentation av rastrets utbredning.

Synopsis

geometry **ST_Envelope**(raster rast);

Description

Returnerar polygonrepresentationen av rastrets utbredning i spatiala koordinatenheter definierade av srid. Det är en float8 minsta avgränsande box som representeras som en polygon.

Polygonen definieras av hörnpunkterna i den begränsande boxen ((`MINX`, `MINY`), (`MINX`, `MAXY`), (`MAXX`, `MAXY`), (`MAXX`, `MINY`), (`MINX`, `MINY`))

Exempel

Code

```
SELECT rid, ST_Envelope(rast) As envgeomwkt
FROM dummy_rast;
```

Output

```
rid |                                      envgeomwkt
-----+-------------------------------------------------------------------
   1 | POLYGON((0 0,20 0,20 60,0 60,0 0))
   2 | POLYGON((3427927 5793243,3427928 5793243,
        3427928 5793244,3427927 5793244, 3427927 5793243))
```

Se även

ST_Envelope, ST_AsText, ST_SRID

11.15.5 ST_MinConvexHull

ST_MinConvexHull — Returnerar rastrets konvexa skrovgeometri exklusive NODATA-pixlar.

Synopsis

geometry **ST_MinConvexHull**(raster rast, integer nband=NULL);

Description

Returnerar rastrets konvexa skrovgeometri exklusive NODATA-pixlar. Om `nband` är NULL beaktas alla band i rastret.

Tillgänglighet: 2.1.0

Exempel

Code

```
WITH foo AS (
    SELECT
        ST_SetValues(ST_SetValues(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(9, 9, 0, 0, 1,  ↩
    -1, 0, 0, 0), 1, '8BUI', 0, 0), 2, '8BUI', 1, 0),
                1, 1, 1,
                ARRAY[
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 1, 0, 0, 0, 0, 1],
                    [0, 0, 0, 1, 1, 0, 0, 0, 0],
                    [0, 0, 0, 1, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0]
                ]::double precision[][]
            ),
            2, 1, 1,
            ARRAY[
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [1, 0, 0, 0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 1, 1, 0, 0, 0],
                [0, 0, 0, 0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 1, 0, 0, 0, 0, 0, 0]
            ]::double precision[][]
        ) AS rast
)
SELECT title, geom AS geom
FROM foo
CROSS JOIN LATERAL (VALUES
    ('raster hull', ST_ConvexHull(rast)),
    ('all bands', ST_MinConvexHull(rast)),
    ('band 1', ST_MinConvexHull(rast, 1)),
    ('band 2', ST_MinConvexHull(rast, 2))
) AS variants(title, geom);
```

Output

```
title    |                 geom
---------+--------------------------------------
 raster hull | POLYGON((0 0,9 0,9 -9,0 -9,0 0))
 all bands   | POLYGON((0 -3,9 -3,9 -9,0 -9,0 -3))
 band 1      | POLYGON((3 -3,9 -3,9 -6,3 -6,3 -3))
 band 2      | POLYGON((0 -3,6 -3,6 -9,0 -9,0 -3))
```

Figure

Se även

ST_Envelope, ST_ConvexHull, ST_ConvexHull, ST_AsText

11.15.6 ST_Polygon

ST_Polygon — Returnerar en multipolygongeometri som bildas av sammanslagningen av pixlar som har ett pixelvärde som inte är något datavärde. Om inget bandnummer anges är standardvärdet för bandnum 1.

Synopsis

geometry **ST_Polygon**(raster rast, integer band_num=1);

Description

Ändrad 3.3.0, validering och fixering är inaktiverad för att förbättra prestanda. Kan resultera i ogiltiga geometrier.

Tillgänglighet: 0.1.6 Kräver GDAL 1.7 eller högre.

Förbättrad: 2.1.0 Förbättrad hastighet (helt C-baserad) och den returnerande multipolygonen är säkerställd att vara giltig.

Ändrad: 2.1.0 I tidigare versioner returnerades ibland en polygon, ändrat till att alltid returnera multipolygon.

Exempel

Compare polygonization of every pixel with polygonization after zero is assigned as the NODATA value. The latter preserves the gap formed by zero-valued cells.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(4, 4, 0, 4, 1, -1, 0, 0, 0),
            1, '8BUI', 1, NULL
        ),
        1, 1, 1,
        ARRAY[
            [1, 1, 1, 1],
            [1, 0, 0, 1],
            [1, 1, 0, 1],
            [1, 1, 1, 1]
        ]::double precision[][]
    ) AS rast
), variants AS (
    SELECT 'all pixels' AS title, ST_Polygon(rast) AS geom
    FROM source
    UNION ALL
    SELECT
        'zero excluded',
        ST_Polygon(ST_SetBandNoDataValue(rast, 1, 0))
    FROM source
)
SELECT title, geom AS geom
FROM variants
ORDER BY CASE title
    WHEN 'all pixels' THEN 1
    ELSE 2
END;
```

Figure

Se även

ST_Value, ST_DumpAsPolygons

11.15.7 ST_IntersectionFractions

ST_IntersectionFractions — Beräknar den del av varje rastercell som täcks av en given geometri.

Synopsis

raster **ST_IntersectionFractions**(raster rast, geometry geom);

Description

Beräknar den del av varje rastercell som täcks av en given geometri. Det första argumentet är en raster som definierar rutnätets geometri som ska användas för beräkningen. Omfattningen och cellstorleken läses från rasterparametern. Det andra argumentet är en geometri som läggs över rutnätet, och varje rutnät fylls i baserat på överlagringen av geometrin på rutnätet. För polygoner är det värde som returneras för varje cell andelen av dess area som täcks av geometrin. För linjesträngar är det värde som returneras för varje cell längden som ingår i cellen.

Tillgänglighet: 3.6.0 Kräver GEOS 3.14 eller högre.

Exempel

This rotated square polygon covers half of each cell in the raster.

Code

```
CREATE TABLE raster_proportions_rast (
    name text,
    rast raster
);

INSERT INTO raster_proportions_rast (name, rast) VALUES (
  '2x2 raster covering 0,0 to 10,10',
  ST_MakeEmptyRaster(2, 2, -- raster width/height in pixels
    0, 10, -- upper-left corner x/y coordinates
    5, -5, -- pixel width/height in ground units
    0, 0, -- skew x/y
    0        -- SRID
  ));

SELECT name, ST_DumpValues(
    ST_IntersectionFractions(
        rast,
        'POLYGON((5 0,0 5,5 10,10 5,5 0))'::geometry),
    1)
FROM raster_proportions_rast;
```

Output

```
2x2 raster covering 0,0 to 10,10
--------------------------------
 {{0.5,0.5},{0.5,0.5}}
```

Se även

ST_MakeEmptyRaster

11.16 Rasteroperatorer

11.16.1 &&

&& — Returnerar TRUE om A:s avgränsande box skär B:s avgränsande box.

Synopsis

boolean **&&**(raster A , raster B);
boolean **&&**(raster A , geometry B);
boolean **&&**(geometry B , raster A);

Description

Operatorn && returnerar TRUE om begränsningsrutan för raster/geometr A skär begränsningsrutan för raster/geometr B.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på rastren.

Tillgänglighet: 2.0.0

Exempel

Code

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast && B.rast As intersect
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B LIMIT 3;
```

Output

```
a_rid | b_rid | intersect
-------+-------+---------
     2 |     2 | t
     2 |     3 | f
     2 |     1 | f
```

11.16.2 &<

&< — Returnerar TRUE om A:s avgränsande box är till vänster om B:s.

Synopsis

boolean **&<**(raster A , raster B);

Description

Operatorn &< returnerar TRUE om begränsningsrutan för raster A överlappar eller ligger till vänster om begränsningsrutan för raster B, eller mer exakt, överlappar eller INTE ligger till höger om begränsningsrutan för raster B.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på rastren.

Exempel

Code

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast &< B.rast As overleft
 FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

Output

```
a_rid | b_rid | overleft
------+-------+----------
    2 |     2 | t
    2 |     3 | f
    2 |     1 | f
    3 |     2 | t
    3 |     3 | t
    3 |     1 | f
    1 |     2 | t
    1 |     3 | t
    1 |     1 | t
```

11.16.3 &>

&> — Returnerar TRUE om A:s avgränsande box ligger till höger om B:s.

Synopsis

boolean **&>**(raster A , raster B);

Description

Operatorn &> returnerar TRUE om raster A:s avgränsande box överlappar eller ligger till höger om raster B:s avgränsande box, eller mer exakt, överlappar eller INTE ligger till vänster om raster B:s avgränsande box.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på geometrierna.

Exempel

Code

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast &
> B.rast As overright
 FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

Output

```
 a_rid | b_rid | overright
-------+-------+-----------
     2 |     2 | t
     2 |     3 | t
     2 |     1 | t
     3 |     2 | f
     3 |     3 | t
     3 |     1 | f
     1 |     2 | f
     1 |     3 | t
     1 |     1 | t
```

11.16.4 =

= — Returnerar TRUE om A:s avgränsande box är densamma som B:s. Använder avgränsningsbox med dubbel precision.

Synopsis

boolean **=**(raster A , raster B);

Description

Operatorn = returnerar SANT om gränsboxen för raster A är densamma som gränsboxen för raster B. PostgreSQL använder operatorerna =, < och > som definieras för raster för att utföra interna ordningar och jämförelse av raster (dvs. i en GROUP BY- eller ORDER BY-klausul).

Observera
Denna operand kommer INTE att använda några index som kan finnas tillgängliga på rastren. Använd ~= istället. Den här operatorn finns främst för att man ska kunna gruppera efter rasterkolumnen.

Tillgänglighet: 2.1.0

Se även

~=

11.16.5 @

@ — Returnerar TRUE om A:s avgränsande box är innesluten i B:s. Använder avgränsande box med dubbel precision.

Synopsis

boolean @(raster A , raster B);
boolean @(geometry A , raster B);
boolean @(raster B , geometry A);

Description

Operatorn @ returnerar TRUE om begränsningsrutan för raster/geometri A ingår i begränsningsrutan för raster/geometri B.

Notera
Detta operand kommer att använda spatiala index på rastrerna.

Tillgänglighet: 2.0.0 raster @ raster, raster @ geometri infördes

Tillgänglighet: 2.0.5 geometry @ raster infördes

Se även

~

11.16.6 ~=

~= — Returnerar TRUE om A:s avgränsande box är densamma som B:s.

Synopsis

boolean ~=(raster A , raster B);

Description

Operatorn ~= returnerar TRUE om begränsningsrutan för raster A är densamma som begränsningsrutan för raster B.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på rastren.

Tillgänglighet: 2.0.0

Exempel

Ett mycket användbart användarfall är att ta två uppsättningar av enkelbandsraster som är av samma stycke men representerar olika teman och skapa ett flerbandsraster

Code

```
SELECT ST_AddBand(prec.rast, alt.rast) As new_rast
FROM prec INNER JOIN alt ON (prec.rast ~= alt.rast);
```

Se även

ST_AddBand, =

11.16.7 ~

~ — Returnerar TRUE om A:s avgränsande box innehåller B:s. Använder avgränsande box med dubbel precision.

Synopsis

boolean ~(raster A , raster B);
boolean ~(geometry A , raster B);
boolean ~(raster B , geometry A);

Description

Operatorn ~ returnerar TRUE om begränsningsrutan för raster/geometri A innehåller begränsningsrutan för raster/geometri B.

Notera
Detta operand kommer att använda spatiala index på rastrerna.

Tillgänglighet: 2.0.0

Se även

@

11.17 Raster och Rasterband - spatiala relationer

11.17.1 ST_Contains

ST_Contains — Returnerar true om inga punkter i raster rastB ligger i raster rastA:s exteriör och minst en punkt i rastB:s interiör ligger i rastA:s interiör.

Synopsis

boolean **ST_Contains**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Contains**(raster rastA , raster rastB);

Description

Raster rastA innehåller rastB om och endast om inga punkter i rastB ligger i rastA:s utsida och minst en punkt i rastB:s insida ligger i rastA:s insida. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Notera
Denna funktion kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_Contains(ST_Polygon(raster), geometry) eller ST_Contains(geometry, ST_Polygon(raster)).

Notera
ST_Contains() är inversen av ST_Within(). Så ST_Contains(rastA, rastB) innebär ST_Within(rastB, rastA).

Tillgänglighet: 2.1.0

Exempel

Specified band numbers.

Code

```
SELECT r1.rid, r2.rid, ST_Contains(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 1;
```

Output

```
NOTICE:  The first raster provided has no bands
 rid | rid | st_contains
-----+-----+-------------
   1 |   1 |
   1 |   2 | f
```

No band numbers specified.

Code

```
SELECT r1.rid, r2.rid, ST_Contains(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 1;
```

Output

```
rid | rid | st_contains
-----+-----+-------------
   1 |   1 | t
   1 |   2 | f
```

Se även

ST_Intersects, ST_Within

11.17.2 ST_ContainsProperly

ST_ContainsProperly — Returnerar true om rastB skär rastA:s insida men inte rastA:s gräns eller utsida.

Synopsis

boolean **ST_ContainsProperly**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_ContainsProperly**(raster rastA , raster rastB);

Description

Raster rastA innehåller korrekt rastB om rastB skär rastA:s inre men inte rastA:s gräns eller yttre. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Raster rastA innehåller inte riktigt sig själv men innehåller sig själv.

Notera
Denna funktion kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_ContainsProperly(ST_Polygon(raster), geometry) eller ST_ContainsProperly(geometry, ST_Polygon(raster)).

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT r1.rid, r2.rid, ST_ContainsProperly(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS ↩
     JOIN dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_containsproperly
-----+-----+---------------------
   2 |   1 | f
   2 |   2 | f
```

Se även

ST_Intersects, ST_Contains

11.17.3 ST_Covers

ST_Covers — Returnerar true om inga punkter i raster rastB ligger utanför raster rastA.

Synopsis

boolean **ST_Covers**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Covers**(raster rastA , raster rastB);

Description

Raster rastA täcker rastB om och endast om inga punkter i rastB ligger i rastA:s utsida. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Notera
Denna funktion kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_Covers(ST_Polygon(raster), geometry) eller ST_Covers(geometry, ST_Polygon(raster)).

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT r1.rid, r2.rid, ST_Covers(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_covers
-----+-----+-----------
   2 |   1 | f
   2 |   2 | t
```

Se även

ST_Intersects, ST_CoveredBy

11.17.4 ST_CoveredBy

ST_CoveredBy — Returnerar true om inga punkter i raster rastA ligger utanför raster rastB.

Synopsis

boolean **ST_CoveredBy**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_CoveredBy**(raster rastA , raster rastB);

Description

Raster rastA täcks av rastB om och endast om inga punkter i rastA ligger i rastB:s utsida. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Notera
Denna funktion kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
För att testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_CoveredBy(ST_Polygon(raster), geometry) eller ST_CoveredBy(geometry, ST_Polygon(raster)).

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT r1.rid, r2.rid, ST_CoveredBy(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_coveredby
-----+-----+--------------
   2 |   1 | f
   2 |   2 | t
```

Se även

ST_Intersects, ST_Covers

11.17.5 ST_Disjoint

ST_Disjoint — Returnerar true om raster rastA inte spatialt korsar rastB.

Synopsis

boolean **ST_Disjoint**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Disjoint**(raster rastA , raster rastB);

Description

Raster rastA och rastB är disjointed om de inte delar något utrymme tillsammans. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Notera
Denna funktion använder INTE några index.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_Disjoint(ST_Polygon(raster), geometry).

Tillgänglighet: 2.1.0

Exempel

Rid = 1 has no bands, hence the NOTICE and the NULL value for st_disjoint.

Code

```
SELECT r1.rid, r2.rid, ST_Disjoint(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
NOTICE:  The second raster provided has no bands
 rid | rid | st_disjoint
-----+-----+-------------
   2 |   1 |
   2 |   2 | f
```

This time, without specifying band numbers.

Code

```
SELECT r1.rid, r2.rid, ST_Disjoint(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_disjoint
-----+-----+-------------
   2 |   1 | t
   2 |   2 | f
```

Se även

ST_Intersects

11.17.6 ST_Intersects

ST_Intersects — Returnerar true om raster rastA spatialt korsar raster rastB.

Synopsis

boolean **ST_Intersects**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Intersects**(raster rastA , raster rastB);
boolean **ST_Intersects**(raster rast , integer nband , geometry geommin);
boolean **ST_Intersects**(raster rast , geometry geommin , integer nband=NULL);
boolean **ST_Intersects**(geometry geommin , raster rast , integer nband=NULL);

Description

Returnerar true om raster rastA spatialt korsar raster rastB. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges, beaktas endast de pixlar som har ett värde (inte NODATA) i testet.

Notera
Denna funktion kommer att använda alla index som kan finnas tillgängliga på rastren.

Förbättrad: 2.0.0 stöd för raster/raster intersects introducerades.

Varning
Ändrad: 2.1.0 Beteendet för ST_Intersects(raster, geometri)-varianterna ändrades så att det överensstämmer med beteendet för ST_Intersects(geometri, raster).

Exempel

Different bands of same raster.

Code

```
SELECT ST_Intersects(rast, 2, rast, 3) FROM dummy_rast WHERE rid = 2;
```

Output

```
t
```

Se även

ST_Intersection, ST_Disjoint

11.17.7 ST_Overlaps

ST_Overlaps — Returnerar true om raster rastA och rastB korsar varandra men det ena inte helt innehåller det andra.

Synopsis

boolean **ST_Overlaps**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Overlaps**(raster rastA , raster rastB);

Description

Returnerar true om raster rastA spatialt överlappar raster rastB. Detta innebär att rastA och rastB korsar varandra men att den ena inte helt innehåller den andra. Om bandnumret inte anges (eller om det sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Notera
Denna funktion kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_Overlaps(ST_Polygon(raster), geometry).

Tillgänglighet: 2.1.0

Exempel

Comparing different bands of same raster.

Code

```
SELECT ST_Overlaps(rast, 1, rast, 2) FROM dummy_rast WHERE rid = 2;
```

Output

```
f
```

Se även

ST_Intersects

11.17.8 ST_Touches

ST_Touches — Returnerar true om raster rastA och rastB har minst en gemensam punkt men deras inre delar inte skär varandra.

Synopsis

boolean **ST_Touches**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Touches**(raster rastA , raster rastB);

Description

Returnerar true om raster rastA spatialt berör raster rastB. Detta innebär att rastA och rastB har minst en gemensam punkt men att deras inre delar inte skär varandra. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Notera
Denna funktion kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
För att testa det spatiala förhållandet mellan ett raster och en geometri, använd ST_Polygon på rastret, t.ex. ST_Touches(ST_Polygon(raster), geometry).

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT r1.rid, r2.rid, ST_Touches(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_touches
-----+-----+------------
   2 |   1 | f
   2 |   2 | f
```

Se även

ST_Intersects

11.17.9 ST_SameAlignment

ST_SameAlignment — Returnerar true om raster har samma skevhet, skala, spatiala ref och offset (pixlar kan placeras i samma rutnät utan att skära i pixlar) och false om de inte har det med ett meddelande om detaljproblem.

Synopsis

boolean **ST_SameAlignment**(raster rastA , raster rastB);
boolean **ST_SameAlignment**(double precision ulx1 , double precision uly1 , double precision scalex1 , double precision scaley1 , double precision skewx1 , double precision skewy1 , double precision ulx2 , double precision uly2 , double precision scalex2 , double precision scaley2 , double precision skewx2 , double precision skewy2);
boolean **ST_SameAlignment**(raster set rastfield);

Description

Icke-aggregerad version (variant 1 och 2): Returnerar true om de två rastren (antingen tillhandahållna direkt eller skapade med hjälp av värdena för upperleft, scale, skew och srid) har samma scale, skew, srid och om minst ett av de fyra hörnen av en pixel i ett raster faller på ett hörn av rutnätet i det andra rastret. Returnerar false om så inte är fallet och ett NOTICE som beskriver justeringsproblemet.

Aggregerad version (Variant 3): Från en uppsättning raster returneras sant om alla raster i uppsättningen är inriktade. ST_SameAlignment () -funktionen är en äggregeradfunktion i terminologin för PostgreSQL. Det betyder att den fungerar på rader med data, på samma sätt som SUM () och AVG () -funktionerna gör.

Tillgänglighet: 2.0.0

Förbättrad: 2.1.0 tillägg av Aggregate-variant

Exempel: Raster

Code

```
SELECT ST_SameAlignment(ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
    ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0)
) as sm;
```

Output

```
t
```

Code

```
SELECT ST_SameAlignment(A.rast, b.rast)
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

Output

```
NOTICE:  The two rasters provided have different SRIDs
NOTICE:  The two rasters provided have different SRIDs
 t
 f
 f
 f
```

Se även

Avsnitt 10.1, ST_NotSameAlignmentReason, ST_MakeEmptyRaster

11.17.10 ST_NotSameAlignmentReason

ST_NotSameAlignmentReason — Returnerar text som anger om rastrerna är inriktade och om de inte är inriktade, en anledning till varför.

Synopsis

text **ST_NotSameAlignmentReason**(raster rastA, raster rastB);

Description

Returnerar text som anger om rastrerna är inriktade och om de inte är inriktade, en anledning till varför.

Notera
Om det finns flera anledningar till att rastrerna inte är i linje kommer endast en anledning (det första testet som misslyckas) att returneras.

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT
    ST_SameAlignment(ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
        ST_MakeEmptyRaster(1, 1, 0, 0, 1.1, 1.1, 0, 0)
    ),
    ST_NotSameAlignmentReason(ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
        ST_MakeEmptyRaster(1, 1, 0, 0, 1.1, 1.1, 0, 0)
    )
;
```

Output

```
st_samealignment |            st_notsamealignmentreason
-----------------+------------------------------------------------
 f               | The rasters have different scales on the X axis
(1 row)
```

Se även

Avsnitt 10.1, ST_SameAlignment

11.17.11 ST_Within

ST_Within — Returnerar true om inga punkter i raster rastA ligger i raster rastB:s exteriör och minst en punkt i rastA:s interiör ligger i rastB:s interiör.

Synopsis

boolean **ST_Within**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Within**(raster rastA , raster rastB);

Description

Raster rastA ligger inom rastB om och endast om inga punkter i rastA ligger i rastB:s utsida och minst en punkt i rastA:s insida ligger i rastB:s insida. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_Within(ST_Polygon(raster), geometry) eller ST_Within(geometry, ST_Polygon(raster)).

Notera
ST_Within() är inversen av ST_Contains(). Så ST_Within(rastA, rastB) innebär ST_Contains(rastB, rastA).

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT r1.rid, r2.rid, ST_Within(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_within
-----+-----+-----------
   2 |   1 | f
   2 |   2 | t
```

Se även

ST_Intersects, ST_Contains, ST_DWithin, ST_DFullyWithin

11.17.12 ST_DWithin

ST_DWithin — Returnerar true om rasterna rastA och rastB ligger inom det angivna avståndet från varandra.

Synopsis

boolean **ST_DWithin**(raster rastA , integer nbandA , raster rastB , integer nbandB , double precision distance_of_srid);
boolean **ST_DWithin**(raster rastA , raster rastB , double precision distance_of_srid);

Description

Returnerar true om rastren rastA och rastB ligger inom det angivna avståndet från varandra. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Avståndet anges i enheter som definieras av rastrernas spatiala referenssystem. För att denna funktion ska vara meningsfull måste källrastren ha samma koordinatprojektion och samma SRID.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_DWithin(ST_Polygon(raster), geometry).

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT r1.rid, r2.rid, ST_DWithin(r1.rast, 1, r2.rast, 1, 3.14) FROM dummy_rast r1 CROSS ↩
    JOIN dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_dwithin
-----+-----+------------
   2 |   1 | f
   2 |   2 | t
```

Se även

ST_Within, ST_DFullyWithin

11.17.13 ST_DFullyWithin

ST_DFullyWithin — Returnerar true om rasterna rastA och rastB är helt inom det angivna avståndet från varandra.

Synopsis

boolean **ST_DFullyWithin**(raster rastA , integer nbandA , raster rastB , integer nbandB , double precision distance_of_srid);
boolean **ST_DFullyWithin**(raster rastA , raster rastB , double precision distance_of_srid);

Description

Returnerar true om rastren rastA och rastB är helt inom det angivna avståndet från varandra. Om bandnumret inte anges (eller sätts till NULL), beaktas endast rastrets konvexa skrov i testet. Om bandnumret anges är det endast de pixlar som har ett värde (inte NODATA) som beaktas i testet.

Avståndet anges i enheter som definieras av rastrernas spatiala referenssystem. För att denna funktion ska vara meningsfull måste källrastren ha samma koordinatprojektion och samma SRID.

Notera
Detta operand kommer att använda alla index som kan finnas tillgängliga på rastren.

Notera
Om du vill testa det spatiala förhållandet mellan ett raster och en geometri använder du ST_Polygon på rastret, t.ex. ST_DFullyWithin(ST_Polygon(raster), geometry).

Tillgänglighet: 2.1.0

Exempel

Code

```
SELECT r1.rid, r2.rid, ST_DFullyWithin(r1.rast, 1, r2.rast, 1, 3.14) FROM dummy_rast r1 ↩
    CROSS JOIN dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_dfullywithin
-----+-----+-----------------
   2 |   1 | f
   2 |   2 | t
```

Se även

ST_Within, ST_DWithin

11.18 Tips om raster

11.18.1 Ut-DB Rasters

11.18.1.1 Katalog som innehåller många filer

När GDAL öppnar en fil söker GDAL ivrigt igenom filens katalog för att bygga upp en katalog över andra filer. Om den här katalogen innehåller många filer (t.ex. tusentals, miljontals) blir det extremt långsamt att öppna filen (särskilt om filen råkar finnas på en nätverksenhet som t.ex. NFS).

För att kontrollera detta beteende tillhandahåller GDAL följande miljövariabel: GDAL_DISABLE_READDIR_O
Ställ in `GDAL_DISABLE_READDIR_ON_OPEN` till `TRUE` för att inaktivera katalogsökning.

I Ubuntu (och förutsatt att du använder PostgreSQL: s paket för Ubuntu) kan `GDAL_DISABLE_READDIR_ON_OPE` ställas in i */etc/postgresql/POSTGRESQL_VERSION/CLUSTER_NAME/environment* (där POSTGRESQL_VER är versionen av PostgreSQL, t.ex. 9.6 och CLUSTER_NAME är namnet på klustret, t.ex. maindb). Du kan också ställa in PostGIS-miljövariabler här också.

Code

```
# environment variables for postmaster process
# This file has the same syntax as postgresql.conf:
#  VARIABLE = simple_value
#  VARIABLE2 = 'any value!'
# I. e. you need to enclose any value which does not only consist of letters,
# numbers, and '-', '_', '.' in single quotes. Shell commands are not
# evaluated.
POSTGIS_GDAL_ENABLED_DRIVERS = 'ENABLE_ALL'

POSTGIS_ENABLE_OUTDB_RASTERS = 1

GDAL_DISABLE_READDIR_ON_OPEN = 'TRUE'
```

11.18.1.2 Maximalt antal öppna filer

Det maximala antalet öppna filer som tillåts av Linux och PostgreSQL är vanligtvis konservativa (vanligtvis 1024 öppna filer per process) med tanke på antagandet att systemet konsumeras av mänskliga användare. För Out-DB Rasters kan en enda giltig fråga lätt överskrida denna gräns (t.ex. en dataset

med 10 års värde av raster med en raster för varje dag som innehåller minimi- och maximitemperaturer och vi vill veta det absoluta min- och maxvärdet för en pixel i den datasetet).

Den enklaste ändringen att göra är följande PostgreSQL-inställning: max_files_per_process. Standardvärdet är inställt på 1000, vilket är alldeles för lågt för Out-DB Rasters. Ett säkert startvärde kan vara 65536 men detta beror verkligen på dina datamängder och de frågor som körs mot dessa datamängder. Denna inställning kan endast göras vid serverstart och förmodligen endast i PostgreSQL-konfigurationsfilen (t.ex. */etc/postgresql/POSTGRESQL_VERSION/CLUSTER_NAME/postgresql.conf* i Ubuntu-miljöer).

Code

```
...
# - Kernel Resource Usage -

max_files_per_process = 65536           # min 25
                                        # (change requires restart)
...
```

Den största förändringen som måste göras är Linux-kärnans gränser för öppna filer. Det finns två delar i detta:

- Maximalt antal öppna filer för hela systemet
- Maximalt antal öppna filer per process

11.18.1.2.1 Maximalt antal öppna filer för hela systemet

Du kan kontrollera det aktuella maximala antalet öppna filer för hela systemet med hjälp av följande exempel:

Code

```
$ sysctl -a | grep fs.file-max
fs.file-max = 131072
```

Om det värde som returneras inte är tillräckligt stort lägger du till en fil i */etc/sysctl.d* / enligt följande exempel:

Code

```
$ echo "fs.file-max = 6145324" >> /etc/sysctl.d/fs.conf

$ cat /etc/sysctl.d/fs.conf
fs.file-max = 6145324

$ sysctl -p --system
* Applying /etc/sysctl.d/fs.conf ...
fs.file-max = 2097152
* Applying /etc/sysctl.conf ...

$ sysctl -a | grep fs.file-max
fs.file-max = 6145324
```

11.18.1.2.2 Maximalt antal öppna filer per process

Vi måste öka det maximala antalet öppna filer per process för PostgreSQL-serverprocesserna.

För att se vad de aktuella PostgreSQL-serviceprocesserna använder för maximalt antal öppna filer, gör enligt följande exempel (se till att PostgreSQL körs):

Code

```
$ ps aux | grep postgres
```

Output

```
postgres 31713  0.0  0.4 179012 17564 pts/0    S    Dec26   0:03 /home/dustymugs/devel/postgresql/sandbox/10/usr/local ↩
    /bin/postgres -D /home/dustymugs/devel/postgresql/sandbox/10/pgdata
postgres 31716  0.0  0.8 179776 33632 ?        Ss   Dec26   0:01 postgres: checkpointer process
postgres 31717  0.0  0.2 179144  9416 ?        Ss   Dec26   0:05 postgres: writer process
postgres 31718  0.0  0.2 179012  8708 ?        Ss   Dec26   0:06 postgres: wal writer process
postgres 31719  0.0  0.1 179568  7252 ?        Ss   Dec26   0:03 postgres: autovacuum launcher process
postgres 31720  0.0  0.1  34228  4124 ?        Ss   Dec26   0:09 postgres: stats collector process
postgres 31721  0.0  0.1 179308  6052 ?        Ss   Dec26   0:00 postgres: bgworker: logical replication launcher
```

Code

```
$ cat /proc/31718/limits
```

Output

```
Limit                     Soft Limit           Hard Limit           Units
Max cpu time              unlimited            unlimited            seconds
Max file size             unlimited            unlimited            bytes
Max data size             unlimited            unlimited            bytes
Max stack size            8388608              unlimited            bytes
Max core file size        0                    unlimited            bytes
Max resident set          unlimited            unlimited            bytes
Max processes             15738                15738                processes
Max open files            1024                 4096                 files
Max locked memory         65536                65536                bytes
Max address space         unlimited            unlimited            bytes
Max file locks            unlimited            unlimited            locks
Max pending signals       15738                15738                signals
Max msgqueue size         819200               819200               bytes
Max nice priority         0                    0
Max realtime priority     0                    0
Max realtime timeout      unlimited            unlimited            us
```

I exemplet ovan inspekterade vi gränsen för öppna filer för process 31718. Det spelar ingen roll vilken PostgreSQL-process, någon av dem kommer att göra. Svaret vi är intresserade av är *Max öppna filer*.

Vi vill öka *Soft Limit* och *Hard Limit* för *Max öppna filer* för att vara större än det värde vi angav för PostgreSQL-inställningen `max_files_per_process`. I vårt exempel ställer vi in `max_files_per_process` till 65536.

I Ubuntu (och förutsatt att du använder PostgreSQL: s paket för Ubuntu) är det enklaste sättet att ändra *Soft Limit* och *Hard Limit* att redigera **/etc/init.d/postgresql** (SysV) eller **/lib/systemd/system/postgresql * .service** (systemd).

Låt oss först ta itu med SysV Ubuntu-fallet där vi lägger till **ulimit -H -n 262144** och **ulimit -n 131072** i **/etc/init.d/postgresql**.

Code

```
...
case "$1" in
    start|stop|restart|reload)
        if [ "$1" = "start" ]; then
            create_socket_directory
        fi
    if [ -z "`pg_lsclusters -h`" ]; then
        log_warning_msg 'No PostgreSQL clusters exist; see "man pg_createcluster"'
        exit 0
    fi

    ulimit -H -n 262144
    ulimit -n 131072
```

```
    for v in $versions; do
        $1 $v || EXIT=$?
    done
    exit ${EXIT:-0}
        ;;
    status)
...
```

Nu ska vi ta itu med systemd Ubuntu-fallet. Vi kommer att lägga till **LimitNOFILE=131072** i varje **/lib/systemd/system/postgresql*.service-fil** i avsnittet **[Service]**.

Code

```
...
[Service]

LimitNOFILE=131072

...

[Install]
WantedBy=multi-user.target
...
```

När du har gjort de nödvändiga ändringarna i systemd ska du se till att ladda om daemon

Code

```
systemctl daemon-reload
```

Kapitel 12

PostGIS Extrafunktioner

Detta kapitel dokumenterar funktioner som finns i mappen extras i PostGIS-källans tarballs och källförvar. Dessa är inte alltid paketerade med PostGIS binära utgåvor, men är vanligtvis PL/pgSQL-baserade eller standardskalskript som kan köras som de är.

Kapitel 13

Index över specialfunktioner i PostGIS

13.1 PostGIS aggregerade funktioner

Funktionerna nedan är spatiala aggregatfunktioner som används på samma sätt som SQL-aggregatfunktione
`som summa` och `genomsnitt`.

- CG_3DUnion - Utför 3D-union med hjälp av postgis_sfcgal.
- ST_3DExtent - Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.
- ST_AsFlatGeobuf - Returnerar en FlatGeobuf-representation av en uppsättning rader.
- ST_AsGeobuf - Returnerar en Geobuf-representation av en uppsättning rader.
- ST_AsMVT - Aggregatfunktion som returnerar en MVT-representation av en uppsättning rader.
- ST_ClusterDBSCAN - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av DBSCAN-algoritmen.
- ST_ClusterIntersecting - Aggregerad funktion som klustrar inmatade geometrier till sammanhängande mängder.
- ST_ClusterIntersectingWin - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri och klustrar indatageometrier i sammanhängande uppsättningar.
- ST_ClusterKMeans - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av K-means-algoritmen.
- ST_ClusterRelateWin - Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.
- ST_ClusterWithin - Aggregatfunktion som klustrar geometrier efter separationsavstånd.
- ST_ClusterWithinWin - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri, klustring med hjälp av separationsavstånd.
- ST_Collect - Skapar en GeometryCollection eller Multi* geometri från en uppsättning geometrier.
- ST_CoverageClean - Beräknar en ren (kantmatchad, icke-överlappande, gap-cleared) polygontäckning, givet en icke ren indata.

- ST_CoverageInvalidEdges - Fönsterfunktion som hittar platser där polygonerna inte bildar en giltig täckning.
- ST_CoverageSimplify - Fönsterfunktion som förenklar kanterna på en polygonal täckning.
- ST_CoverageUnion - Beräknar unionen av en uppsättning polygoner som bildar en täckning genom att ta bort gemensamma kanter.
- ST_Extent - Aggregerad funktion som returnerar geometriernas avgränsande box.
- ST_MakeLine - Skapar en LineString från Point-, MultiPoint- eller LineString-geometrier.
- ST_MemUnion - Aggregatfunktion som kombinerar geometrier på ett minneseffektivt men långsammare sätt
- ST_MinimumSpanningTree - Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.
- ST_Polygonize - Beräknar en samling polygoner som bildas av linjerna i en uppsättning geometrier.
- ST_SameAlignment - Returnerar true om raster har samma skevhet, skala, spatiala ref och offset (pixlar kan placeras i samma rutnät utan att skära i pixlar) och false om de inte har det med ett meddelande om detaljproblem.
- ST_Union - Beräknar en geometri som representerar punktuppsättningssammanslagningen av indatageometrierna.
- ST_Union - Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.
- TopoElementArray_Agg - Returnerar en topoelementarray för en uppsättning element_id, typ arrayer (topoelements).

13.2 Funktioner i PostGIS-fönstret

Funktionerna nedan är spatiala fönsterfunktioner som används på samma sätt som SQL-fönsterfunktioner som `row_number()`, `lead()` och `lag()`. De måste följas av en `OVER()` -klausul.

- ST_ClusterDBSCAN - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av DBSCAN-algoritmen.
- ST_ClusterIntersectingWin - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri och klustrar indatageometrier i sammanhängande uppsättningar.
- ST_ClusterKMeans - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av K-means-algoritmen.
- ST_ClusterRelateWin - Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.
- ST_ClusterWithinWin - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri, klustring med hjälp av separationsavstånd.
- ST_CoverageClean - Beräknar en ren (kantmatchad, icke-överlappande, gap-cleared) polygontäckning, givet en icke ren indata.
- ST_CoverageInvalidEdges - Fönsterfunktion som hittar platser där polygonerna inte bildar en giltig täckning.
- ST_CoverageSimplify - Fönsterfunktion som förenklar kanterna på en polygonal täckning.
- ST_MinimumSpanningTree - Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.

13.3 PostGIS SQL-MM-kompatibla funktioner

De funktioner som anges nedan är PostGIS-funktioner som överensstämmer med SQL/MM 3-standarden

- CG_3DArea - Beräknar area för 3D-ytgeometrier. Returnerar 0 för solider.
- CG_3DDifference - Utföra 3D-differens
- CG_3DIntersection - Utför 3D-intersektion
- CG_3DUnion - Utför 3D-union med hjälp av postgis_sfcgal.
- CG_Volume - Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.
- ST_3DDWithin - Testar om två 3D-geometrier befinner sig inom ett givet 3D-avstånd
- ST_3DDistance - Returnerar det kartesiska 3D-minsta avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DIntersects - Testar om två geometrier korsar varandra spatialt i 3D - endast för punkter, linestrings, polygoner, polyedrisk yta (area)
- ST_3DLength - Returnerar 3D-längden för en linjär geometri.
- ST_3DPerimeter - Returnerar 3D-perimetern för en polygonal geometri.
- ST_AddEdgeModFace - Lägg till en ny kant och, om den delar en yta, modifiera den ursprungliga ytan och lägg till en ny yta.
- ST_AddEdgeNewFaces - Lägg till en ny kant och, om den delar en yta, ta bort den ursprungliga ytan och ersätt den med två nya ytor.
- ST_AddIsoEdge - Lägger till en isolerad kant definierad av geometrin alinestring till en topologi som förbinder två befintliga isolerade noder anode och anothernode och returnerar kant-ID för den nya kanten.
- ST_AddIsoNode - Lägger till en isolerad nod till en face i en topologi och returnerar nodeid för den nya noden. Om face är null skapas noden ändå.
- ST_Area - Returnerar arean för en polygonal geometri.
- ST_AsBinary - Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsGML - Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsText - Returnera WKT-representationen (Well-Known Text) av geometrin/geografin utan SRID-metadata.
- ST_Boundary - Returnerar gränsen för en geometri.
- ST_Buffer - Beräknar en geometri som täcker alla punkter inom ett givet avstånd från en geometri.
- ST_Centroid - Returnerar den geometriska mittpunkten för en geometri.
- ST_ChangeEdgeGeom - Ändrar formen på en kant utan att påverka topologins struktur.
- ST_Contains - Testar om varje punkt i B ligger i A, och deras interiörer har en gemensam punkt
- ST_ConvexHull - Beräknar det konvexa skrovet av en geometri.
- ST_CoordDim - Returnerar koordinatdimensionen för en geometri.
- ST_CreateTopoGeo - Lägger till en samling geometrier till en given tom topologi och returnerar ett meddelande om det lyckas.

- ST_Crosses - Testar om två geometrier har vissa, men inte alla, inre punkter gemensamt
- ST_CurveN - Returnerar den N:te komponentkurvgeometrin för en CompoundCurve.
- ST_CurveToLine - Konverterar en geometri som innehåller kurvor till en linjär geometri.
- ST_Difference - Beräknar en geometri som representerar den del av geometri A som inte skär geometri B.
- ST_Dimension - Returnerar den topologiska dimensionen för en geometri.
- ST_Disjoint - Testar om två geometrier inte har några gemensamma punkter
- ST_Distance - Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_EndPoint - Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_Envelope - Returnerar en geometri som representerar en geometris avgränsande box.
- ST_Equals - Testar om två geometrier innehåller samma uppsättning punkter
- ST_ExteriorRing - Returnerar en LineString som representerar den yttre ringen av en Polygon.
- ST_GMLToSQL - Returns a geometry from GML. Alias for ST_GeomFromGML.
- ST_GeomCollFromText - Skapar en geometrisk samling från samlingen WKT med angiven SRID. Om SRID inte anges är standardvärdet 0.
- ST_GeomFromText - Returnera ett specificerat ST_Geometry-värde från Well-Known Text representation (WKT).
- ST_GeomFromWKB - Skapar en geometriinstans från en Well-Known Binary geometrirepresentation (WKB) och valfri SRID.
- ST_GeometryFromText - Returns a geometry from Well-Known Text (WKT). Alias for ST_GeomFromText.
- ST_GeometryN - Returnerar ett element i en geometrisamling.
- ST_GeometryType - Returnerar SQL-MM-typen för en geometri som text.
- ST_GetFaceEdges - Returnerar en uppsättning ordnade kanter som avgränsar en yta..
- ST_GetFaceGeometry - Returnerar polygonen i den angivna topologin med det angivna ytans id.
- ST_InitTopoGeo - Skapar ett nytt topologischema och registrerar det i tabellen topology.topology.
- ST_InteriorRingN - Returnerar den N:te inre ringen (hålet) i en polygon.
- ST_Intersection - Beräknar en geometri som representerar den delade delen av geometrierna A och B.
- ST_Intersects - Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- ST_IsClosed - Testar om en LineStrings start- och slutpunkter är sammanfallande. För en PolyhedralSurface testas om den är sluten (volymetrisk).
- ST_IsEmpty - Testar om en geometri är tom.
- ST_IsRing - Testar om en LineString är sluten och enkel.
- ST_IsSimple - Testar om en geometri inte har några punkter med självskärning eller självtangentiering.
- ST_IsValid - Testar om en geometri är välformad i 2D.
- ST_Length - Returnerar 2D-längden för en linjär geometri.

- ST_LineFromText - Skapar en geometri från en WKT-representation med angiven SRID. Om SRID inte anges är standardvärdet 0.
- ST_LineFromWKB - Creates a LINESTRING from a WKB representation, optionally using a given SRID.
- ST_LinestringFromWKB - Creates a LINESTRING from a WKB representation, optionally using a given SRID.
- ST_LocateAlong - Returnerar den eller de punkter på en geometri som matchar ett mätvärde.
- ST_LocateBetween - Returnerar de delar av en geometri som matchar ett mätintervall.
- ST_M - Returnerar M-koordinaten för en punkt.
- ST_MLineFromText - Returnera ett specificerat ST_MultiLineString-värde från WKT-representation.
- ST_MPointFromText - Skapar en geometri från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.
- ST_MPolyFromText - Skapar en MultiPolygon Geometry från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.
- ST_ModEdgeHeal - Läker två kanter genom att ta bort den nod som förbinder dem, modifiera den första kanten och ta bort den andra kanten. Returnerar id för den borttagna noden.
- ST_ModEdgeSplit - Dela en kant genom att skapa en ny nod längs en befintlig kant, modifiera den ursprungliga kanten och lägga till en ny kant.
- ST_MoveIsoNode - Flyttar en isolerad nod i en topologi från en punkt till en annan. Om den nya apoint-geometrin existerar som en nod kastas ett fel. Returnerar beskrivning av förflyttning.
- ST_NewEdgeHeal - Läker två kanter genom att ta bort noden som förbinder dem, ta bort båda kanterna och ersätta dem med en kant vars riktning är densamma som den första tillhandahållna kanten.
- ST_NewEdgesSplit - Dela en kant genom att skapa en ny nod längs en befintlig kant, ta bort den ursprungliga kanten och ersätta den med två nya kanter. Returnerar id för den nya nod som skapats och som sammanfogar de nya kanterna.
- ST_NumCurves - Returnerar antalet komponentkurvor i en CompoundCurve.
- ST_NumGeometries - Returnerar antalet element i en geometrisamling.
- ST_NumInteriorRings - Returnerar antalet inre ringar (hål) i en polygon.
- ST_NumPatches - Returns the number of faces in a PolyhedralSurface or TIN.
- ST_NumPoints - Returnerar antalet punkter i en LineString eller CircularString.
- ST_OrderingEquals - Testar om två geometrier representerar samma geometri och har punkter i samma riktningsordning
- ST_Overlaps - Testar om två geometrier har samma dimension och skär varandra, men var och en har minst en punkt som inte finns i den andra
- ST_PatchN - Returns the Nth face of a PolyhedralSurface or TIN.
- ST_Perimeter - Returnerar längden på gränsen för en polygonal geometri eller geografi.
- ST_Point - Skapar en punkt med X-, Y- och SRID-värden.
- ST_PointFromText - Skapar en punktgeometri från WKT med angiven SRID. Om SRID inte anges är standardvärdet okänt.
- ST_PointFromWKB - Skapar en geometri från WKB med den angivna SRID

- ST_PointN - Returnerar den N:te punkten i den första LineString eller cirkulära LineString i en geometri.
- ST_PointOnSurface - Beräknar en punkt som garanterat ligger i en polygon eller på en geometri.
- ST_Polygon - Skapar en polygon från en LineString med en angiven SRID.
- ST_PolygonFromText - Skapar en geometri från WKT med den angivna SRID. Om SRID inte anges är standardvärdet 0.
- ST_Relate - Testar om två geometrier har en topologisk relation som matchar ett Intersection Matrix-mönster, eller beräknar deras Intersection Matrix
- ST_RemEdgeModFace - Tar bort en kant, och om kanten separerar två ytor tas den ena ytan bort och den andra ytan modifieras så att den täcker utrymmet för båda ytorna.
- ST_RemEdgeNewFace - Tar bort en kant och, om den borttagna kanten separerade två ytor, tar bort de ursprungliga ytorna och ersätter dem med en ny yta.
- ST_RemoveIsoEdge - Tar bort en isolerad kant och returnerar en beskrivning av åtgärden. Om kanten inte är isolerad kastas ett undantag.
- ST_RemoveIsoNode - Tar bort en isolerad nod och returnerar en beskrivning av åtgärden. Om noden inte är isolerad (är början eller slutet på en kant), kastas ett undantag.
- ST_SRID - Returnerar den spatiala referensidentifieraren för en geometri.
- ST_StartPoint - Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_SymDifference - Beräknar en geometri som representerar de delar av geometrierna A och B som inte korsar varandra.
- ST_Touches - Testar om två geometrier har minst en gemensam punkt, men deras inre delar inte skär varandra
- ST_Transform - Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem.
- ST_Union - Beräknar en geometri som representerar punktuppsättningssammanslagningen av indatageometrierna.
- ST_WKBToSQL - Returns a geometry from Well-Known Binary (WKB). SQL/MM alias for ST_GeomFromWK without an SRID argument.
- ST_WKTToSQL - Returns a geometry from Well-Known Text (WKT). SQL/MM alias for ST_GeomFromText.
- ST_Within - Testar om varje punkt i A ligger i B, och deras interiörer har en gemensam punkt
- ST_X - Returnerar X-koordinaten för en Point.
- ST_Y - Returnerar Y-koordinaten för en Point.
- ST_Z - Returnerar Z-koordinaten för en Point.

13.4 Stödfunktioner för geografi i PostGIS

De funktioner och operatorer som anges nedan är PostGIS-funktioner/operatorer som tar som indata eller returnerar som utdata ett geografiskt datatypobjekt.

Notera
Funktioner med ett (T) är inte inbyggda geodetiska funktioner och använder ett ST_Transform-anrop till och från geometri för att utföra operationen. Därför kanske de inte beter sig som förväntat när de går över datalinjer, poler och för stora geometrier eller geometripar som täcker mer än en UTM-zon. Grundtransform - (föredrar UTM, Lambert Azimuthal (North/South), och faller tillbaka på Mercator i värsta fall)

- ST_Area - Returnerar arean för en polygonal geometri.
- ST_AsBinary - Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsEWKT - Returnera WKT-representationen (Well-Known Text) av geometrin med SRID-metadata.
- ST_AsGML - Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsGeoJSON - Returnerar en geometri eller funktion i GeoJSON-format.
- ST_AsKML - Returnera geometrin som ett KML-element.
- ST_AsSVG - Returnerar SVG-banedata för en geometri.
- ST_AsText - Returnera WKT-representationen (Well-Known Text) av geometrin/geografin utan SRID-metadata.
- ST_Azimuth - Returnerar den nordbaserade azimuten för en linje mellan två punkter.
- ST_Buffer - Beräknar en geometri som täcker alla punkter inom ett givet avstånd från en geometri.
- ST_Centroid - Returnerar den geometriska mittpunkten för en geometri.
- ST_ClosestPoint - Returnerar den 2D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste linjen från den ena geometrin till den andra.
- ST_CoveredBy - Testar om varje punkt i A ligger i B
- ST_Covers - Testar om varje punkt i B ligger i A
- ST_DWithin - Testar om två geometrier ligger inom ett givet avstånd
- ST_Distance - Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_GeogFromText - Returnera ett angivet geografiskt värde från Well-Known Text representation or extended (WKT).
- ST_GeogFromWKB - Skapar en geografisk instans från en geometrisk representation av Well-Known Binary (WKB) eller utökad Well Known Binary (EWKB).
- ST_GeographyFromText - Returnera ett angivet geografiskt värde från Well-Known Text representation or extended (WKT).
- = - Returnerar TRUE om koordinaterna och koordinatordningen för geometri/geografi A är samma som koordinaterna och koordinatordningen för geometri/geografi B.
- ST_Intersection - Beräknar en geometri som representerar den delade delen av geometrierna A och B.
- ST_Intersects - Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- ST_Length - Returnerar 2D-längden för en linjär geometri.
- ST_LineInterpolatePoint - Returnerar en punkt som interpolerats längs en linje på en fraktionerad plats.

- ST_LineInterpolatePoints - Returnerar punkter interpolerade längs en linje med ett fraktionerat intervall.
- ST_LineLocatePoint - Returnerar den fraktionerade positionen för den punkt på en linje som ligger närmast en punkt.
- ST_LineSubstring - Returnerar delen av en linje mellan två fraktionerade platser.
- ST_Perimeter - Returnerar längden på gränsen för en polygonal geometri eller geografi.
- ST_Project - Returnerar en punkt som projiceras från en startpunkt med ett avstånd och en bäring (azimut).
- ST_Segmentize - Returnerar en modifierad geometri/geografi som inte har något segment som är längre än ett givet avstånd.
- ST_ShortestLine - Returnerar den kortaste 2D-linjen mellan två geometrier
- ST_Summary - Returnerar en textsammanfattning av innehållet i en geometri.
- <-> - Returnerar 2D-avståndet mellan A och B.
- && - Returnerar TRUE om A:s 2D-begränsningsbox skär B:s 2D-begränsningsbox.

13.5 Stödfunktioner för PostGIS Raster

Nedanstående funktioner och operatorer är PostGIS funktioner/operatorer som tar som indata eller returnerar som utdata ett raster datatypobjekt. Listade i alfabetisk ordning.

- Box3D - Returnerar box 3d-representationen av den omslutande boxen i rastret.
- @ - Returnerar TRUE om A:s avgränsande box är innesluten i B:s. Använder avgränsande box med dubbel precision.
- ~ - Returnerar TRUE om A:s avgränsande box innehåller B:s. Använder avgränsande box med dubbel precision.
- = - Returnerar TRUE om A:s avgränsande box är densamma som B:s. Använder avgränsningsbox med dubbel precision.
- && - Returnerar TRUE om A:s avgränsande box skär B:s avgränsande box.
- &< - Returnerar TRUE om A:s avgränsande box är till vänster om B:s.
- &> - Returnerar TRUE om A:s avgränsande box ligger till höger om B:s.
- ~= - Returnerar TRUE om A:s avgränsande box är densamma som B:s.
- ST_Retile - Returnerar en uppsättning konfigurerade plattor från en godtyckligt uppdelad rastertäckning.
- ST_AddBand - Returnerar ett raster med det eller de nya banden av given typ som lagts till med givet initialvärde på den givna indexplatsen. Om inget index anges läggs bandet till i slutet.
- ST_AsBinary/ST_AsWKB - Returnerar WKB-representationen (Well-Known Binary) av rastret.
- ST_AsGDALRaster - Returnerar rasterplattan i det angivna GDAL Raster-formatet. Rasterformat är ett av de format som stöds av ditt kompilerade bibliotek. Använd ST_GDALDrivers() för att få en lista över de format som stöds av ditt bibliotek.
- ST_AsHexWKB - Returnerar Well-Known Binary (WKB) i Hex-representation av rastret.

- ST_AsJPEG - Returnerar de valda banden i rasterplattan som en enda JPEG-bild (byte-array) (Joint Photographic Exports Group). Om inget band anges och 1 eller fler än 3 band, används endast det första bandet. Om endast 3 band anges används alla 3 banden och mappas till RGB.
- ST_AsPNG - Returnerar de valda banden i rasterkaklet som en enda PNG-bild (portable network graphics) (byte-array). Om 1, 3 eller 4 band i rastret och inga band anges, används alla band. Om fler än 2 eller fler än 4 band och inga band anges, används endast band 1. Banden mappas till RGB- eller RGBA-rymd.
- ST_AsRaster - Konverterar en PostGIS-geometri till ett PostGIS-raster.
- ST_AsRasterAgg - Aggregera. Renderar PostGIS-geometrier till ett nytt raster.
- ST_AsTIFF - Returnerar de band som valts i rastret som en enda TIFF-bild (byte-array). Om inget band anges eller om något av de angivna banden inte finns i rastret, försöker alla band användas.
- ST_Aspect - Returnerar aspekten (i grader som standard) för ett höjdrasterband. Användbart för analys av terräng.
- ST_Band - Returnerar ett eller flera band från ett befintligt raster som ett nytt raster. Användbart för att bygga nya raster från befintliga raster.
- ST_BandFileSize - Returnerar filstorleken för ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.
- ST_BandFileTimestamp - Returnerar filens tidsstämpel för ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.
- ST_BandIsNoData - Returnerar true om bandet är fyllt med endast nodata-värden.
- ST_BandMetaData - Returnerar grundläggande metadata för ett specifikt rasterband. bandnummer 1 antas om det inte specificeras.
- ST_BandNoDataValue - Returnerar värdet i ett givet band som inte representerar några data. Om inget band finns antas siffran 1.
- ST_BandPath - Returnerar systemfilens sökväg till ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.
- ST_BandPixelType - Returnerar pixeltyp för givet band. Om inget bandnummer anges antas 1.
- ST_Clip - Returnerar det raster som klippts av indatageometrin. Om bandnummer inte anges bearbetas alla band. Om crop inte anges eller om TRUE anges, beskärs utdatarastret. Om touched är inställt på TRUE inkluderas pixlar som berörs, annars inkluderas endast pixlar vars mittpunkt ligger i geometrin.
- ST_ColorMap - Skapar ett nytt raster med upp till fyra 8BUI-band (gråskala, RGB, RGBA) från källrastret och ett angivet band. Band 1 antas om det inte specificeras.
- ST_Contains - Returnerar true om inga punkter i raster rastB ligger i raster rastA:s exteriör och minst en punkt i rastB:s interiör ligger i rastA:s interiör.
- ST_ContainsProperly - Returnerar true om rastB skär rastA:s insida men inte rastA:s gräns eller utsida.
- ST_Contour - Skapar en uppsättning vektorkonturer från det tillhandahållna rasterbandet med hjälp av GDAL-konturalgoritmen.
- ST_ConvexHull - Returnerar rastrets konvexa skrovgeometri inklusive pixelvärden som är lika med BandNoDataValue. För regelbundet formade och icke snedställda raster ger detta samma resultat som ST_Envelope, så det är endast användbart för oregelbundet formade eller snedställda raster.
- ST_Count - Returnerar antalet pixlar i ett givet band i ett raster eller en rastertäckning. Om inget band anges är standardvärdet band 1. Om exclude_nodata_value är satt till true räknas endast pixlar som inte är lika med nodatavärdet.

- ST_CountAgg - Aggregera. Returnerar antalet pixlar i ett givet band i en uppsättning raster. Om inget band anges är standardvärdet band 1. Om exclude_nodata_value är satt till true räknas endast pixlar som inte är lika med NODATA-värdet.
- ST_CoveredBy - Returnerar true om inga punkter i raster rastA ligger utanför raster rastB.
- ST_Covers - Returnerar true om inga punkter i raster rastB ligger utanför raster rastA.
- ST_DFullyWithin - Returnerar true om rasterna rastA och rastB är helt inom det angivna avståndet från varandra.
- ST_DWithin - Returnerar true om rasterna rastA och rastB ligger inom det angivna avståndet från varandra.
- ST_Disjoint - Returnerar true om raster rastA inte spatialt korsar rastB.
- ST_DumpAsPolygons - Returnerar en uppsättning geomval (geom,val)-rader från ett givet rasterband. Om inget bandnummer anges är standardvärdet för bandnum 1.
- ST_DumpValues - Hämta värdena för det angivna bandet som en 2-dimensionell array.
- ST_Envelope - Returnerar en polygonrepresentation av rastrets utbredning.
- ST_FromGDALRaster - Returnerar ett raster från en GDAL-rasterfil som stöds.
- ST_GeoReference - Returnerar metadata för georeferenser i GDAL- eller ESRI-format, vilket är vanligt förekommande i en world-fil. Standard är GDAL.
- ST_Grayscale - Skapar ett nytt bandraster med ett 8BUI-band från källrastret och angivna band som representerar rött, grönt och blått
- ST_HasNoBand - Returnerar true om det inte finns något band med angivet bandnummer. Om inget bandnummer anges antas bandnummer 1.
- ST_Height - Returnerar höjden på rastret i pixlar.
- ST_HillShade - Returnerar den hypotetiska belysningen för ett höjdrasterband med hjälp av angivna indata för azimut, höjd, ljusstyrka och skala.
- ST_Histogram - Returnerar en uppsättning poster som sammanfattar en raster- eller rastertäckningsdatadistribution i separata bin-områden. Antalet bin beräknas automatiskt om det inte anges.
- ST_InterpolateRaster - Interpolerar en rutnätsyta baserat på en indatauppsättning av 3D-punkter, med hjälp av X- och Y-värdena för att positionera punkterna i rutnätet och punkternas Z-värde som ytans höjd.
- ST_Intersection - Returnerar ett raster eller en uppsättning geometri-pixelvärdespar som representerar den delade delen av två raster eller den geometriska skärningspunkten mellan en vektorisering av rastret och en geometri.
- ST_IntersectionFractions - Beräknar den del av varje rastercell som täcks av en given geometri.
- ST_Intersects - Returnerar true om raster rastA spatialt korsar raster rastB.
- ST_IsEmpty - Returnerar true om rastret är tomt (bredd = 0 och höjd = 0). I annat fall returneras false.
- ST_MakeEmptyCoverage - Täck georefererat område med ett rutnät av tomma rasterplattor.
- ST_MakeEmptyRaster - Returnerar ett tomt raster (utan band) med givna dimensioner (bredd & höjd), övre vänstra X och Y, pixelstorlek och rotation (scalex, scaley, skewx & skewy) och referenssystem (srid). Om ett raster skickas in returneras ett nytt raster med samma storlek, inriktning och SRID. Om srid utelämnas sätts den spatiala ref till okänd (0).

- ST_MapAlgebra (callback function version) - Callback function version - Returnerar ett enbandsraster med ett eller flera indataraster, bandindex och en användarspecificerad callback-funktion.
- ST_MapAlgebraExpr - 1 raster band version: Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på inmatningsrasterbandet och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges.
- ST_MapAlgebraExpr - 2 rasterbandversion: Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på de två inmatningsrasterbanden och av pixeltyp som tillhandahålls. band 1 för varje raster antas om inga bandnummer anges. Den resulterande rastern kommer att justeras (skala, skevhet och pixelhörn) på det rutnät som definieras av den första rastern och ha sin utsträckning definierad av parametern ”extenttype”. Värden för ”extenttype”kan vara: INTERSECTION, UNION, FIRST, SECOND.
- ST_MapAlgebraFct - 1 bandversion - Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion på inmatningsrasterbandet och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges.
- ST_MapAlgebraFct - 2 band version - Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion på de 2 inmatningsrasterbanden och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges. Utsträckningstyp är som standard INTERSECTION om den inte anges.
- ST_MapAlgebraFctNgb - 1-bandsversion: Kartlägg Algebra närmaste granne med hjälp av användardefinierad PostgreSQL-funktion. Returnera en raster vars värden är resultatet av en PLPGSQL-användarfunktion som involverar ett grannskap av värden från inmatningsrasterbandet.
- ST_MapAlgebra (expression version) - Expression version - Returnerar ett enbandsraster med ett eller två indataraster, bandindex och ett eller flera användarspecifika SQL-uttryck.
- ST_MemSize - Returnerar den mängd utrymme (i byte) som rastret tar upp.
- ST_MetaData - Returnerar grundläggande metadata om ett rasterobjekt, t.ex. pixelstorlek, rotation (skew), övre, nedre vänster etc.
- ST_MinConvexHull - Returnerar rastrets konvexa skrovgeometri exklusive NODATA-pixlar.
- ST_NearestValue - Returnerar det närmaste icke-NODATA-värdet för ett givet bands pixel som anges av en kolumnx och rowy eller en geometrisk punkt uttryckt i samma spatiala referenskoordinatsystem som rastret.
- ST_Neighborhood - Returnerar en 2D-array med dubbel precision av icke-NODATA-värden runt ett visst bands pixel som anges av antingen en kolumnX och radY eller en geometrisk punkt uttryckt i samma spatiala referenskoordinatsystem som rastret.
- ST_NotSameAlignmentReason - Returnerar text som anger om rastrerna är inriktade och om de inte är inriktade, en anledning till varför.
- ST_NumBands - Returnerar antalet band i rasterobjektet.
- ST_Overlaps - Returnerar true om raster rastA och rastB korsar varandra men det ena inte helt innehåller det andra.
- ST_PixelAsCentroid - Returnerar centroiden (punktgeometri) för det område som representeras av en pixel.
- ST_PixelAsCentroids - Returnerar centroiden (punktgeometri) för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrin är centroiden för det område som representeras av en pixel.
- ST_PixelAsPoint - Returnerar en punktgeometri för pixelns övre vänstra hörn.

- ST_PixelAsPoints - Returnerar en punktgeometri för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrins koordinater är pixelns övre vänstra hörn.
- ST_PixelAsPolygon - Returnerar den polygongeometri som avgränsar pixeln för en viss rad och kolumn.
- ST_PixelAsPolygons - Returnerar den polygongeometri som avgränsar varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel.
- ST_PixelHeight - Returnerar pixelhöjden i geometriska enheter i det spatiala referenssystemet.
- ST_PixelOfValue - Hämta koordinaterna för kolumnx, rowy för den pixel vars värde är lika med sökvärdet.
- ST_PixelWidth - Returnerar pixelbredden i geometriska enheter i det spatiala referenssystemet.
- ST_Polygon - Returnerar en multipolygongeometri som bildas av sammanslagningen av pixlar som har ett pixelvärde som inte är något datavärde. Om inget bandnummer anges är standardvärdet för bandnum 1.
- ST_Quantile - Beräkna kvantiler för ett raster eller en rastertabells täckning i samband med urvalet eller populationen. Ett värde kan alltså undersökas för att ligga vid rastrets 25 %, 50 %, 75% percentil.
- ST_RastFromHexWKB - Returnera ett rastervärde från en Hex-representation av Well-Known Binary (WKB)-raster.
- ST_RastFromWKB - Returnera ett rastervärde från ett Well-Known Binary (WKB)-raster.
- ST_RasterToWorldCoord - Returnerar rastrets övre vänstra hörn som geometriska X och Y (longitud och latitud) givet en kolumn och rad. Kolumn och rad börjar på 1.
- ST_RasterToWorldCoordX - Returnerar den geometriska X-koordinaten uppe till vänster i ett raster, en kolumn och en rad. Numreringen av kolumner och rader börjar på 1.
- ST_RasterToWorldCoordY - Returnerar den geometriska Y-koordinaten i övre vänstra hörnet av ett raster, en kolumn och en rad. Numreringen av kolumner och rader börjar på 1.
- ST_Reclass - Skapar ett nytt raster som består av bandtyper som omklassificerats från originalet. Nband är det band som ska ändras. Om nband inte anges antas värdet vara 1. Alla andra band returneras oförändrade. Användningsfall: konvertera ett 16BUI-band till ett 8BUI och så vidare för enklare rendering som visningsbara format.
- ST_ReclassExact - Skapar ett nytt raster som består av band som omklassificerats från originalbandet med hjälp av en 1:1-mappning från värden i originalbandet till nya värden i destinationsbandet.
- ST_Resample - Resampla ett raster med hjälp av en specificerad resamplingsalgoritm, nya dimensioner, ett godtyckligt rutnätshörn och en uppsättning rastergeoreferensattribut som definierats eller lånats från ett annat raster.
- ST_Rescale - Resampla ett raster genom att endast justera dess skala (eller pixelstorlek). Nya pixelvärden beräknas med hjälp av omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline, Lanczos, Max eller Min. Standard är NearestNeighbor.
- ST_Resize - Ändra storlek på ett raster till en ny bredd/höjd
- ST_Reskew - Resampla ett raster genom att endast justera dess skevhet (eller rotationsparametrar). Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är NearestNeighbor.
- ST_Rotation - Returnerar rastrets rotation i radianer.
- ST_Roughness - Returnerar ett raster med den beräknade öjämnhetenför en DEM.

- ST_SRID - Returnerar den spatiala referensidentifieraren för rastret enligt definitionen i tabellen spatial_ref_sys.
- ST_SameAlignment - Returnerar true om raster har samma skevhet, skala, spatiala ref och offset (pixlar kan placeras i samma rutnät utan att skära i pixlar) och false om de inte har det med ett meddelande om detaljproblem.
- ST_ScaleX - Returnerar X-komponenten av pixelbredden i enheter av koordinatreferenssystemet.
- ST_ScaleY - Returnerar Y-komponenten av pixelhöjden i enheter av koordinatreferenssystemet.
- ST_SetBandIndex - Uppdatera det externa bandnumret för ett out-db-band
- ST_SetBandIsNoData - Ställer in bandets isnodata-flagga till TRUE.
- ST_SetBandNoDataValue - Ställer in värdet för det angivna bandet som inte representerar några data. Band 1 förutsätts om inget band anges. För att markera ett band som att det inte har något nodata-värde, ställ in nodata-värdet = NULL.
- ST_SetBandPath - Uppdatera den externa sökvägen och bandnumret för ett out-db-band
- ST_SetGeoReference - Set Georeference 6 georeferensparametrar i ett enda anrop. Siffrorna ska separeras med vitt utrymme. Accepterar inmatningar i GDAL- eller ESRI-format. Standard är GDAL.
- ST_SetM - Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopierade till M-dimensionen med hjälp av den begärda resample-algoritmen.
- ST_SetRotation - Ställ in rastrets rotation i radian.
- ST_SetSRID - Ställer in SRID för ett raster till ett visst heltal srid som definieras i tabellen spatial_ref_sys.
- ST_SetScale - Ställer in X- och Y-storleken för pixlar i enheter i koordinatreferenssystemet. Antal enheter/pixelbredd/höjd.
- ST_SetSkew - Ställer in georeferensens X- och Y-skevhet (eller rotationsparameter). Om endast en parameter anges, sätts X och Y till samma värde.
- ST_SetUpperLeft - Ställer in värdet för det övre vänstra hörnet av pixeln i rastret till projicerade X- och Y-koordinater.
- ST_SetValue - Returnerar modifierad raster som är resultatet av att värdet för ett givet band har ställts in i en given kolumnx, radpixel eller de pixlar som skär en viss geometri. Bandnummer börjar på 1 och antas vara 1 om de inte specificeras.
- ST_SetValues - Returnerar modifierat raster som är resultatet av att värdena för ett givet band har ställts in.
- ST_SetZ - Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopierade till Z-dimensionen med hjälp av den begärda resample-algoritmen.
- ST_SkewX - Returnerar georeferensens X skew (eller rotationsparameter).
- ST_SkewY - Returnerar georeferensens Y skew (eller rotationsparameter).
- ST_Slope - Returnerar lutningen (i grader som standard) för ett höjdrasterband. Användbart för att analysera terräng.
- ST_SnapToGrid - Sampla om ett raster genom att fästa det i ett rutnät. Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är NearestNeighbor.
- ST_Summary - Returnerar en textsammanfattning av innehållet i rastret.

- ST_SummaryStats - Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i ett raster eller en rastertäckning. Band 1 antas om inget band anges.
- ST_SummaryStatsAgg - Aggregera. Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i en uppsättning raster. Band 1 antas om inget band anges.
- ST_TPI - Returnerar ett raster med det beräknade topografiska positionsindexet.
- ST_TRI - Returnerar ett raster med det beräknade Terrain Ruggedness Index.
- ST_Tile - Returnerar en uppsättning raster som är resultatet av uppdelningen av inmatningsrastret baserat på de önskade dimensionerna för utdatarastren.
- ST_Touches - Returnerar true om raster rastA och rastB har minst en gemensam punkt men deras inre delar inte skär varandra.
- ST_Transform - Återprojicerar ett raster i ett känt spatialt referenssystem till ett annat känt spatialt referenssystem med hjälp av en angiven omsamplingsalgoritm. Alternativen är NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos med NearestNeighbor som standard.
- ST_Union - Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.
- ST_UpperLeftX - Returnerar den övre vänstra X-koordinaten för rastret i projicerad spatial ref.
- ST_UpperLeftY - Returnerar den övre vänstra Y-koordinaten för rastret i projicerad spatial ref.
- ST_Value - Returnerar värdet för ett visst band i en viss kolumnx, radpixel eller vid en viss geometrisk punkt. Bandnummer börjar på 1 och antas vara 1 om det inte anges. Om exclude_nodata_value är satt till false, anses alla pixlar inklusive nodata-pixlar korsa varandra och returnerar värdet. Om värdet exclude_nodata_value inte anges läses det från rastrets metadata.
- ST_ValueCount - Returnerar en uppsättning poster som innehåller ett pixelbandvärde och en räkning av antalet pixlar i ett givet band i ett raster (eller en rastertäckning) som har en given uppsättning värden. Om inget band anges är standardvärdet band 1. Som standard räknas inte pixlar med nodatavärden. och alla andra värden i pixeln matas ut och pixelbandvärden avrundas till närmaste heltal.
- ST_Width - Returnerar rastrets bredd i pixlar.
- ST_Within - Returnerar true om inga punkter i raster rastA ligger i raster rastB:s exteriör och minst en punkt i rastA:s interiör ligger i rastB:s interiör.
- ST_WorldToRasterCoord - Returnerar det övre vänstra hörnet som kolumn och rad givet geometriskt X och Y (longitud och latitud) eller en punktgeometri uttryckt i rastrets spatiala referenskoordinatsystem.
- ST_WorldToRasterCoordX - Returnerar kolumnen i rastret för punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw) som representeras i rastrets världsspatiala referenssystem.
- ST_WorldToRasterCoordY - Returnerar raden i rastret för punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw) som representeras i rastrets världsspatiala referenssystem.
- UpdateRasterSRID - Ändra SRID för alla raster i den användarspecifika kolumnen och tabellen.

13.6 PostGIS Geometri / Geografi / Raster Dumpfunktioner

Funktionerna som anges nedan är PostGIS-funktioner som tar som indata eller returnerar som utdata en uppsättning eller ett enda geometry_dump eller geomval datatypobjekt.

- ST_DumpAsPolygons - Returnerar en uppsättning geomval (geom,val)-rader från ett givet rasterband. Om inget bandnummer anges är standardvärdet för bandnum 1.
- ST_Intersection - Returnerar ett raster eller en uppsättning geometri-pixelvärdespar som representerar den delade delen av två raster eller den geometriska skärningspunkten mellan en vektorisering av rastret och en geometri.
- ST_Dump - Returnerar en uppsättning geometry_dump-rader för komponenterna i en geometri.
- ST_DumpPoints - Returnerar en uppsättning geometry_dump-rader för koordinaterna i en geometri.
- ST_DumpRings - Returnerar en uppsättning geometry_dump-rader för de yttre och inre ringarna i en polygon.
- ST_DumpSegments - Returnerar en uppsättning geometry_dump-rader för segmenten i en geometri.

13.7 PostGIS Box-funktioner

De funktioner som anges nedan är PostGIS-funktioner som tar som indata eller returnerar som utdata box*-familjen av PostGIS spatiala typer. Box-familjen av typer består av box2d och box3d

- Box2D - Returnerar en BOX2D som representerar 2D-utbredningen av en geometri.
- Box3D - Returnerar en BOX3D som representerar 3D-utbredningen av en geometri.
- MakeTopologyPrecise - Fäst topologivinklar till precisionsrutnätet.
- Box3D - Returnerar box 3d-representationen av den omslutande boxen i rastret.
- ST_3DExtent - Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.
- ST_3DMakeBox - Skapar en BOX3D som definieras av två 3D-punktgeometrier.
- ST_AsMVTGeom - Transformerar en geometri till koordinatrymden för en MVT-platta.
- ST_AsTWKB - Returnerar geometrin som TWKB, aka Tiny Well-Known Binary”
- ST_Box2dFromGeoHash - Returnerar en BOX2D från en GeoHash-sträng.
- ST_ClipByBox2D - Beräknar den del av en geometri som faller inom en rektangel.
- ST_EstimatedExtent - Returnerar den uppskattade omfattningen av en spatial tabell.
- ST_Expand - Returnerar en bounding box som expanderats från en annan bounding box eller en geometri.
- ST_Extent - Aggregerad funktion som returnerar geometriernas avgränsande box.
- ST_MakeBox2D - Skapar en BOX2D som definieras av två 2D-punktgeometrier.
- ST_RemoveIrrelevantPointsForView - Tar bort punkter som är irrelevanta för rendering av en specifik rektangulär vy av en geometri.
- ST_XMax - Returnerar X-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_XMin - Returnerar X-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_XSize - Returns the X size of a 2D or 3D bounding box or a geometry.
- ST_YMax - Returnerar Y-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_YMin - Returnerar Y-minima för en 2D- eller 3D-begränsningsbox eller en geometri.

- ST_YSize - Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_ZMax - Returnerar Z-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_ZMin - Returnerar Z-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_ZSize - Returns the Z size of a 2D or 3D bounding box or a geometry.
- RemoveUnusedPrimitives - Tar bort topologiprimitiver som inte behövs för att definiera befintliga TopoGeometry-objekt.
- ValidateTopology - Returnerar en uppsättning validatetopology_returntype-objekt som beskriver problem med topologin.
- ValidateTopologyPrecision - Returnerar icke-precisa toppunkter i topologin.
- ~(box2df,box2df) - Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) innehåller en annan avgränsande 2D-box med flytande precision (BOX2DF).
- ~(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) innehåller en geometris 2D-bondingbox.
- ~(geometry,box2df) - Returnerar TRUE om en geometris 2D-bindningsbox innehåller en 2D-bindningsbox med floatprecision (GIDX).
- @(box2df,box2df) - Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) ingår i en annan avgränsande 2D-box med flytande precision.
- @(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) ingår i en geometris 2D-gränsbox.
- @(geometry,box2df) - Returnerar TRUE om en geometris 2D-begränsningsbox ingår i en 2D-begränsningsl med flytande precision (BOX2DF).
- &&(box2df,box2df) - Returnerar TRUE om två 2D-begränsningsboxar med flytande precision (BOX2DF) skär varandra.
- &&(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) skär en geometris (cachelagrade) 2D-gränsbox.
- &&(geometry,box2df) - Returnerar TRUE om en geometris (cachelagrade) 2D-begränsningsbox skär en 2D-begränsningsbox med flytande precision (BOX2DF).

13.8 PostGIS-funktioner med stöd för 3D

De funktioner som anges nedan är PostGIS-funktioner som inte kastar bort Z-index.

- AddGeometryColumn - Lägger till en geometrikolumn i en befintlig tabell.
- Box3D - Returnerar en BOX3D som representerar 3D-utbredningen av en geometri.
- CG_3DAlphaWrapping - Beräknar en 3D Alpha-wrapping som strikt omsluter en geometri.
- CG_3DArea - Beräknar area för 3D-ytgeometrier. Returnerar 0 för solider.
- CG_3DConvexHull - Beräknar den konvexa 3D-skålen för en geometri.
- CG_3DDifference - Utföra 3D-differens
- CG_3DIntersection - Utför 3D-intersektion
- CG_3DRotate - Roterar en geometri i 3D-rymden runt en axelvektor.

- CG_3DScale - Skalar en geometri med separata faktorer längs X-, Y- och Z-axlarna.
- CG_3DScaleAroundCenter - Skalar en geometri i 3D-rymden runt en angiven mittpunkt.
- CG_3DTranslate - Translaterar (flyttar) en geometri med hjälp av givna offsets i 3D-rymden.
- CG_3DUnion - Utför 3D-union med hjälp av postgis_sfcgal.
- CG_ConstrainedDelaunayTriangles - Returnerar en begränsad Delaunay-triangulering runt den givna indatageometrin.
- CG_Extrude - Extrudera en yta till en relaterad volym
- CG_ForceLHR - Tvinga fram LHR-orientering
- CG_IsPlanar - Kontrollera om en yta är plan eller inte
- CG_IsSolid - Testar om geometrin är en solid. Ingen validitetskontroll utförs.
- CG_MakeSolid - Casta geometrin till en solid. Ingen kontroll utförs. För att få en giltig solid måste indatageometrin vara en sluten polyedrisk yta eller en sluten TIN.
- CG_NurbsCurveApproximate - Creates an approximating NURBS curve fitting data points within a tolerance
- CG_NurbsCurveDerivative - Computes the derivative (tangent, curvature) of a NURBS curve at a given parameter
- CG_NurbsCurveInterpolate - Creates an interpolating NURBS curve passing through all given data points
- CG_Orientation - Bestäm ytans orientering
- CG_RotateX - Roterar en geometri runt X-axeln med en given vinkel.
- CG_RotateY - Roterar en geometri runt Y-axeln med en given vinkel.
- CG_RotateZ - Roterar en geometri runt Z-axeln med en given vinkel.
- CG_Simplify - Minskar komplexiteten i en geometri samtidigt som viktiga egenskaper och Z/M-värden bevaras.
- CG_Tesselate - Tessellerar ytan på en polygon eller polyederyta och returnerar som en TIN eller en samling av TINS
- CG_Visibility - Beräkna en synlighetspolygon från en punkt eller ett segment i en polygongeometri
- CG_Volume - Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.
- DropGeometryColumn - Tar bort en geometrikolumn från en spatial tabell.
- GeometryType - Returnerar typen av geometri som text.
- ST_3DClosestPoint - Returnerar den 3D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste 3D-linjen.
- ST_3DDFullyWithin - Testar om två 3D-geometrier är helt inom ett givet 3D-avstånd
- ST_3DDWithin - Testar om två 3D-geometrier befinner sig inom ett givet 3D-avstånd
- ST_3DDistance - Returnerar det kartesiska 3D-minsta avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DExtent - Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.
- ST_3DInterpolatePoint - Returns the interpolated measure of a geometry closest to a point in 3D.

- ST_3DIntersects - Testar om två geometrier korsar varandra spatialt i 3D - endast för punkter, linestrings, polygoner, polyedrisk yta (area)
- ST_3DLength - Returnerar 3D-längden för en linjär geometri.
- ST_3DLineInterpolatePoint - Returnerar en punkt som interpolerats längs en 3D-linje på en fraktionerad plats.
- ST_3DLongestLine - Returnerar den längsta 3D-linjen mellan två geometrier
- ST_3DMaxDistance - Returnerar det maximala kartesiska 3D-avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DPerimeter - Returnerar 3D-perimetern för en polygonal geometri.
- ST_3DShortestLine - Returnerar den kortaste 3D-linjen mellan två geometrier
- ST_AddMeasure - Interpolerar mått längs en linjär geometri.
- ST_AddPoint - Lägg till en punkt i en LineString.
- ST_Affine - Tillämpa en 3D-affin transformation på en geometri.
- ST_AsBinary - Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsEWKB - Returnerar EWKB-representationen (Extended Well-Known Binary) av geometrin med SRID-metadata.
- ST_AsEWKT - Returnera WKT-representationen (Well-Known Text) av geometrin med SRID-metadata.
- ST_AsGML - Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsGeoJSON - Returnerar en geometri eller funktion i GeoJSON-format.
- ST_AsHEXEWKB - Returnerar en geometri i HEXEWKB-format (som text) med antingen little-endian (NDR) eller big-endian (XDR) kodning.
- ST_AsKML - Returnera geometrin som ett KML-element.
- ST_AsX3D - Returnerar en geometri i X3D xml-nodelementformat: ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Boundary - Returnerar gränsen för en geometri.
- ST_BoundingDiagonal - Returnerar diagonalen i en geometris avgränsande box.
- ST_CPAWithin - Testar om den närmaste punkten för två banor ligger inom det angivna avståndet.
- ST_CatmullRomSmoothing - Returns a smoothed version of a geometry, using the Catmull-Rom spline algorithm
- ST_ChaikinSmoothing - Returnerar en utjämnad version av en geometri med hjälp av Chaikin-algoritmen
- ST_ClosestPointOfApproach - Returnerar ett mått på den närmaste närmandepunkten för två banor.
- ST_Collect - Skapar en GeometryCollection eller Multi* geometri från en uppsättning geometrier.
- ST_ConvexHull - Beräknar det konvexa skrovet av en geometri.
- ST_CoordDim - Returnerar koordinatdimensionen för en geometri.
- ST_CurveN - Returnerar den N:te komponentkurvgeometrin för en CompoundCurve.
- ST_CurveToLine - Konverterar en geometri som innehåller kurvor till en linjär geometri.

- ST_DelaunayTriangles - Returnerar Delaunay-trianguleringen av hörnen i en geometri.
- ST_Difference - Beräknar en geometri som representerar den del av geometri A som inte skär geometri B.
- ST_DistanceCPA - Returnerar avståndet mellan de två banornas närmaste närmandepunkter.
- ST_Dump - Returnerar en uppsättning geometry_dump-rader för komponenterna i en geometri.
- ST_DumpPoints - Returnerar en uppsättning geometry_dump-rader för koordinaterna i en geometri.
- ST_DumpRings - Returnerar en uppsättning geometry_dump-rader för de yttre och inre ringarna i en polygon.
- ST_DumpSegments - Returnerar en uppsättning geometry_dump-rader för segmenten i en geometri.
- ST_EndPoint - Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_ExteriorRing - Returnerar en LineString som representerar den yttre ringen av en Polygon.
- ST_FlipCoordinates - Returnerar en version av en geometri med X- och Y-axlarna vända.
- ST_Force2D - Tvinga geometrierna till ett ”2-dimensionellt läge”.
- ST_ForceCurve - Upcasta en geometri till dess kurvade typ, om tillämpligt.
- ST_ForcePolygonCCW - Orienterar alla yttre ringar moturs och alla inre ringar medurs.
- ST_ForcePolygonCW - Orienterar alla yttre ringar medurs och alla inre ringar moturs.
- ST_ForceRHR - Tvinga orienteringen av hörnen i en polygon att följa högerhands-regeln.
- ST_ForceSFS - Force geometries to use SFS 1.1 or 1.2 geometry types.
- ST_Force3D - Tvingar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ.
- ST_Force3DZ - Tvinga geometrierna till XYZ-läge.
- ST_Force4D - Tvinga geometrierna till XYZM-läge.
- ST_ForceCollection - Konvertera geometrin till en GEOMETRYCOLLECTION.
- ST_GeomFromEWKB - Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Binary representation (EWKB).
- ST_GeomFromEWKT - Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Text representation (EWKT).
- ST_GeomFromGML - Tar som indata GML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeomFromGeoJSON - Tar som indata en geojson-representation av en geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeomFromKML - Tar som indata KML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeometricMedian - Returnerar den geometriska medianen för en MultiPoint.
- ST_GeometryN - Returnerar ett element i en geometrisamling.
- ST_GeometryType - Returnerar SQL-MM-typen för en geometri som text.
- ST_HasArc - Testar om en geometri innehåller en cirkelbåge
- ST_HasM - Kontrollerar om en geometri har en M (mått)-dimension.

- ST_HasZ - Kontrollerar om en geometri har en Z-dimension.
- ST_InteriorRingN - Returnerar den N:te inre ringen (hålet) i en polygon.
- ST_InterpolatePoint - Returnerar det interpolerade måttet för en geometri som ligger närmast en punkt.
- ST_Intersection - Beräknar en geometri som representerar den delade delen av geometrierna A och B.
- ST_IsClosed - Testar om en LineStrings start- och slutpunkter är sammanfallande. För en PolyhedralSurface testas om den är sluten (volymetrisk).
- ST_IsCollection - Testar om en geometri är en geometrisamlingstyp.
- ST_IsPolygonCCW - Testar om polygoner har yttre ringar som är orienterade moturs och inre ringar som är orienterade medurs.
- ST_IsPolygonCW - Testar om polygoner har yttre ringar som är orienterade medurs och inre ringar som är orienterade moturs.
- ST_IsSimple - Testar om en geometri inte har några punkter med självskärning eller självtangentiering.
- ST_IsValidTrajectory - Testar om geometrin är en giltig bana.
- ST_LengthSpheroid - Returnerar 2D- eller 3D-längd/perimeter för en lon/lat-geometri på en sfäroid.
- ST_LineFromMultiPoint - Skapar en LineString från en MultiPoint-geometri.
- ST_LineInterpolatePoint - Returnerar en punkt som interpolerats längs en linje på en fraktionerad plats.
- ST_LineInterpolatePoints - Returnerar punkter interpolerade längs en linje med ett fraktionerat intervall.
- ST_LineSubstring - Returnerar delen av en linje mellan två fraktionerade platser.
- ST_LineToCurve - Konverterar en linjär geometri till en krökt geometri.
- ST_LocateBetweenElevations - Returnerar de delar av en geometri som ligger inom ett höjdintervall (Z).
- ST_M - Returnerar M-koordinaten för en punkt.
- ST_MMax - Returns the M maxima of a geometry.
- ST_MMin - Returns the M minima of a geometry.
- ST_MSize - Returns the M size of a geometry.
- ST_MakeLine - Skapar en LineString från Point-, MultiPoint- eller LineString-geometrier.
- ST_MakePoint - Skapar en 2D-, 3DZ- eller 4D-punkt.
- ST_MakePolygon - Creates a Polygon or CurvePolygon from a shell and optional list of holes.
- ST_MakeValid - Försöker göra en ogiltig geometri giltig utan att förlora toppar.
- ST_MemSize - Returnerar hur mycket minnesutrymme en geometri tar upp.
- ST_MemUnion - Aggregatfunktion som kombinerar geometrier på ett minneseffektivt men långsammare sätt
- ST_NDims - Returnerar koordinatdimensionen för en geometri.
- ST_NPoints - Returnerar antalet punkter (vertices) i en geometri.

- ST_NRings - Returnerar antalet ringar i en polygonal geometri.
- ST_Node - Noder en samling av linjer.
- ST_NumCurves - Returnerar antalet komponentkurvor i en CompoundCurve.
- ST_NumGeometries - Returnerar antalet element i en geometrisamling.
- ST_NumPatches - Returns the number of faces in a PolyhedralSurface or TIN.
- ST_PatchN - Returns the Nth face of a PolyhedralSurface or TIN.
- ST_PointFromWKB - Skapar en geometri från WKB med den angivna SRID
- ST_PointN - Returnerar den N:te punkten i den första LineString eller cirkulära LineString i en geometri.
- ST_PointOnSurface - Beräknar en punkt som garanterat ligger i en polygon eller på en geometri.
- ST_Points - Returnerar en MultiPoint som innehåller koordinaterna för en geometri.
- ST_Polygon - Skapar en polygon från en LineString med en angiven SRID.
- ST_RemovePoint - Ta bort en punkt från en linestrings.
- ST_RemoveRepeatedPoints - Returnerar en version av en geometri där dubbletter av punkter har tagits bort.
- ST_Reverse - Returnera geometrin med omvänd ordning på topparna.
- ST_Rotate - Roterar en geometri runt en ursprungspunkt.
- ST_RotateX - Roterar en geometri runt X-axeln.
- ST_RotateY - Roterar en geometri runt Y-axeln.
- ST_RotateZ - Roterar en geometri runt Z-axeln.
- ST_Scale - Skalar en geometri med givna faktorer.
- ST_Scroll - Ändra startpunkt för en sluten LineString.
- ST_SetPoint - Ersätt punkten i en linestrings med en given punkt.
- ST_ShiftLongitude - Flyttar longitudkoordinaterna för en geometri mellan -180..180 och 0..360.
- ST_SnapToGrid - Fäst alla punkter i indatageometrin i ett regelbundet rutnät.
- ST_StartPoint - Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_SwapOrdinates - Returnerar en version av den givna geometrin med givna ordinatvärden ombytta.
- ST_SymDifference - Beräknar en geometri som representerar de delar av geometrierna A och B som inte korsar varandra.
- ST_TransScale - Translaterar och skalar en geometri med hjälp av givna offsets och faktorer.
- ST_Translate - Translaterar en geometri med givna offsets.
- ST_UnaryUnion - Beräknar sammanslagningen av komponenterna i en enda geometri.
- ST_Union - Beräknar en geometri som representerar punktuppsättningssammanslagningen av indatageometrierna.
- ST_WrapX - Omsluta en geometri runt ett X-värde.
- ST_X - Returnerar X-koordinaten för en Point.

- ST_XMax - Returnerar X-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_XMin - Returnerar X-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_XSize - Returns the X size of a 2D or 3D bounding box or a geometry.
- ST_Y - Returnerar Y-koordinaten för en Point.
- ST_YMax - Returnerar Y-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_YMin - Returnerar Y-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_YSize - Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_Z - Returnerar Z-koordinaten för en Point.
- ST_ZMFlag - Returnerar en kod som anger ZM-koordinatdimensionen för en geometri.
- ST_ZMax - Returnerar Z-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_ZMin - Returnerar Z-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_ZSize - Returns the Z size of a 2D or 3D bounding box or a geometry.
- Equals - Returnerar true om två topogeometrier består av samma topologiprimitiver.
- Intersects - Returnerar true om något par av primitiver från de två topogeometrierna korsar varandra.
- UpdateGeometrySRID - Uppdaterar SRID för alla objekt i en geometrikolumn och metadata för tabellen.
- &&& - Returnerar TRUE om A:s n-D-gränsbox skär B:s n-D-gränsbox.
- &&&(geometry,gidx) - Returnerar TRUE om en geometris (cachade) n-D-begränsningsbox skär en n-D-begränsningsbox med flytande precision (GIDX).
- &&&(gidx,geometry) - Returnerar TRUE om en n-D avgränsningsbox med flytande precision (GIDX) skär en geometris (cachelagrade) n-D avgränsningsbox.
- &&&(gidx,gidx) - Returnerar TRUE om två gränsboxar (GIDX) med n-D floatprecision skär varandra.

13.9 Stödfunktioner för krökt geometri i PostGIS

Funktionerna nedan är PostGIS-funktioner som kan använda CIRCULARSTRING, CURVEPOLYGON och andra krökta geometrityper

- AddGeometryColumn - Lägger till en geometrikolumn i en befintlig tabell.
- Box2D - Returnerar en BOX2D som representerar 2D-utbredningen av en geometri.
- Box3D - Returnerar en BOX3D som representerar 3D-utbredningen av en geometri.
- DropGeometryColumn - Tar bort en geometrikolumn från en spatial tabell.
- GeometryType - Returnerar typen av geometri som text.
- PostGIS_AddBBox - Lägg till en begränsningsbox till geometrin.
- PostGIS_DropBBox - Ta bort begränsningsboxens cache från geometrin.
- PostGIS_HasBBox - Returnerar TRUE om bboxen för denna geometri är cachad, FALSE annars.
- ST_3DExtent - Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.

- ST_Affine - Tillämpa en 3D-affin transformation på en geometri.
- ST_AsBinary - Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsEWKB - Returnerar EWKB-representationen (Extended Well-Known Binary) av geometrin med SRID-metadata.
- ST_AsEWKT - Returnera WKT-representationen (Well-Known Text) av geometrin med SRID-metadata.
- ST_AsHEXEWKB - Returnerar en geometri i HEXEWKB-format (som text) med antingen little-endian (NDR) eller big-endian (XDR) kodning.
- ST_AsSVG - Returnerar SVG-banedata för en geometri.
- ST_AsText - Returnera WKT-representationen (Well-Known Text) av geometrin/geografin utan SRID-metadata.
- ST_ClusterDBSCAN - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av DBSCAN-algoritmen.
- ST_ClusterWithin - Aggregatfunktion som klustrar geometrier efter separationsavstånd.
- ST_ClusterWithinWin - Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri, klustring med hjälp av separationsavstånd.
- ST_Collect - Skapar en GeometryCollection eller Multi* geometri från en uppsättning geometrier.
- ST_CoordDim - Returnerar koordinatdimensionen för en geometri.
- ST_CurveToLine - Konverterar en geometri som innehåller kurvor till en linjär geometri.
- ST_Distance - Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_Dump - Returnerar en uppsättning geometry_dump-rader för komponenterna i en geometri.
- ST_DumpPoints - Returnerar en uppsättning geometry_dump-rader för koordinaterna i en geometri.
- ST_EndM - Returns the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_EndPoint - Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_EstimatedExtent - Returnerar den uppskattade omfattningen av en spatial tabell.
- ST_FlipCoordinates - Returnerar en version av en geometri med X- och Y-axlarna vända.
- ST_Force2D - Tvinga geometrierna till ett ”2-dimensionellt läge”.
- ST_ForceCurve - Upcasta en geometri till dess kurvade typ, om tillämpligt.
- ST_ForceSFS - Force geometries to use SFS 1.1 or 1.2 geometry types.
- ST_Force3D - Tvingar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ.
- ST_Force3DM - Tvinga geometrierna till XYM-läge.
- ST_Force3DZ - Tvinga geometrierna till XYZ-läge.
- ST_Force4D - Tvinga geometrierna till XYZM-läge.
- ST_ForceCollection - Konvertera geometrin till en GEOMETRYCOLLECTION.
- ST_GeoHash - Returnerar en GeoHash-representation av geometrin.
- ST_GeogFromWKB - Skapar en geografisk instans från en geometrisk representation av Well-Known Binary (WKB) eller utökad Well Known Binary (EWKB).

- ST_GeomFromEWKB - Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Binary representation (EWKB).
- ST_GeomFromEWKT - Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Text representation (EWKT).
- ST_GeomFromText - Returnera ett specificerat ST_Geometry-värde från Well-Known Text representation (WKT).
- ST_GeomFromWKB - Skapar en geometriinstans från en Well-Known Binary geometrirepresentation (WKB) och valfri SRID.
- ST_GeometryN - Returnerar ett element i en geometrisamling.
- = - Returnerar TRUE om koordinaterna och koordinatordningen för geometri/geografi A är samma som koordinaterna och koordinatordningen för geometri/geografi B.
- &<| - Returnerar TRUE om A:s avgränsande box överlappar eller ligger under B:s.
- ST_HasArc - Testar om en geometri innehåller en cirkelbåge
- ST_Intersects - Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- ST_IsClosed - Testar om en LineStrings start- och slutpunkter är sammanfallande. För en PolyhedralSurface testas om den är sluten (volymetrisk).
- ST_IsCollection - Testar om en geometri är en geometrisamlingstyp.
- ST_IsEmpty - Testar om en geometri är tom.
- ST_LineToCurve - Konverterar en linjär geometri till en krökt geometri.
- ST_MMax - Returns the M maxima of a geometry.
- ST_MMin - Returns the M minima of a geometry.
- ST_MSize - Returns the M size of a geometry.
- ST_MemSize - Returnerar hur mycket minnesutrymme en geometri tar upp.
- ST_NPoints - Returnerar antalet punkter (vertices) i en geometri.
- ST_NRings - Returnerar antalet ringar i en polygonal geometri.
- ST_PointFromWKB - Skapar en geometri från WKB med den angivna SRID
- ST_PointN - Returnerar den N:te punkten i den första LineString eller cirkulära LineString i en geometri.
- ST_Points - Returnerar en MultiPoint som innehåller koordinaterna för en geometri.
- ST_Rotate - Roterar en geometri runt en ursprungspunkt.
- ST_RotateZ - Roterar en geometri runt Z-axeln.
- ST_SRID - Returnerar den spatiala referensidentifieraren för en geometri.
- ST_Scale - Skalar en geometri med givna faktorer.
- ST_SetEndM - Sets the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_SetSRID - Ställ in SRID på en geometri.
- ST_SetStartM - Sets the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.

- ST_StartM - Returns the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_StartPoint - Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_Summary - Returnerar en textsammanfattning av innehållet i en geometri.
- ST_SwapOrdinates - Returnerar en version av den givna geometrin med givna ordinatvärden ombytta.
- ST_TransScale - Translaterar och skalar en geometri med hjälp av givna offsets och faktorer.
- ST_Transform - Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem.
- ST_Translate - Translaterar en geometri med givna offsets.
- ST_XMax - Returnerar X-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_XMin - Returnerar X-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_XSize - Returns the X size of a 2D or 3D bounding box or a geometry.
- ST_YMax - Returnerar Y-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_YMin - Returnerar Y-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_YSize - Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_ZMFlag - Returnerar en kod som anger ZM-koordinatdimensionen för en geometri.
- ST_ZMax - Returnerar Z-maximum för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_ZMin - Returnerar Z-minima för en 2D- eller 3D-begränsningsbox eller en geometri.
- ST_ZSize - Returns the Z size of a 2D or 3D bounding box or a geometry.
- UpdateGeometrySRID - Uppdaterar SRID för alla objekt i en geometrikolumn och metadata för tabellen.
- ~(box2df,box2df) - Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) innehåller en annan avgränsande 2D-box med flytande precision (BOX2DF).
- ~(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) innehåller en geometris 2D-bondingbox.
- ~(geometry,box2df) - Returnerar TRUE om en geometris 2D-bindningsbox innehåller en 2D-bindningsbox med floatprecision (GIDX).
- && - Returnerar TRUE om A:s 2D-begränsningsbox skär B:s 2D-begränsningsbox.
- &&& - Returnerar TRUE om A:s n-D-gränsbox skär B:s n-D-gränsbox.
- @(box2df,box2df) - Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) ingår i en annan avgränsande 2D-box med flytande precision.
- @(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) ingår i en geometris 2D-gränsbox.
- @(geometry,box2df) - Returnerar TRUE om en geometris 2D-begränsningsbox ingår i en 2D-begränsningsl med flytande precision (BOX2DF).
- &&(box2df,box2df) - Returnerar TRUE om två 2D-begränsningsboxar med flytande precision (BOX2DF) skär varandra.
- &&(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) skär en geometris (cachelagrade) 2D-gränsbox.

- &&(geometry,box2df) - Returnerar TRUE om en geometris (cachelagrade) 2D-begränsningsbox skär en 2D-begränsningsbox med flytande precision (BOX2DF).
- &&&(geometry,gidx) - Returnerar TRUE om en geometris (cachade) n-D-begränsningsbox skär en n-D-begränsningsbox med flytande precision (GIDX).
- &&&(gidx,geometry) - Returnerar TRUE om en n-D avgränsningsbox med flytande precision (GIDX) skär en geometris (cachelagrade) n-D avgränsningsbox.
- &&&(gidx,gidx) - Returnerar TRUE om två gränsboxar (GIDX) med n-D floatprecision skär varandra.

13.10 PostGIS stödfunktioner för polyedriska ytor

Funktionerna nedan är PostGIS-funktioner som kan använda geometrierna POLYHEDRALSURFACE, POLYHEDRALSURFACEM

- Box2D - Returnerar en BOX2D som representerar 2D-utbredningen av en geometri.
- Box3D - Returnerar en BOX3D som representerar 3D-utbredningen av en geometri.
- CG_3DArea - Beräknar area för 3D-ytgeometrier. Returnerar 0 för solider.
- CG_3DConvexHull - Beräknar den konvexa 3D-skålen för en geometri.
- CG_3DDifference - Utföra 3D-differens
- CG_3DIntersection - Utför 3D-intersektion
- CG_3DUnion - Utför 3D-union med hjälp av postgis_sfcgal.
- CG_ApproximateMedialAxis - Beräkna den ungefärliga mediala axeln för en arealgeometri.
- CG_Extrude - Extrudera en yta till en relaterad volym
- CG_ForceLHR - Tvinga fram LHR-orientering
- CG_GenerateFlatRoof - Generate a flat (box) roof from a footprint polygon.
- CG_GenerateGableRoof - Generate a gable roof from a footprint polygon.
- CG_GenerateHippedRoof - Generate a hipped roof from a footprint polygon.
- CG_GenerateRoof - Generate a roof of the requested type from a footprint polygon.
- CG_GenerateSkillionRoof - Generate a skillion (single-slope) roof from a footprint polygon.
- CG_IsPlanar - Kontrollera om en yta är plan eller inte
- CG_IsSolid - Testar om geometrin är en solid. Ingen validitetskontroll utförs.
- CG_MakeSolid - Casta geometrin till en solid. Ingen kontroll utförs. För att få en giltig solid måste indatageometrin vara en sluten polyedrisk yta eller en sluten TIN.
- CG_PolygonRepair - Repair an invalid polygon or multipolygon.
- CG_StraightSkeleton - Beräkna ett rakt skelett från en geometri
- CG_Tesselate - Tessellerar ytan på en polygon eller polyederyta och returnerar som en TIN eller en samling av TINS
- CG_Visibility - Beräkna en synlighetspolygon från en punkt eller ett segment i en polygongeometri
- CG_Volume - Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.

- GeometryType - Returnerar typen av geometri som text.
- ST_3DClosestPoint - Returnerar den 3D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste 3D-linjen.
- ST_3DDFullyWithin - Testar om två 3D-geometrier är helt inom ett givet 3D-avstånd
- ST_3DDWithin - Testar om två 3D-geometrier befinner sig inom ett givet 3D-avstånd
- ST_3DDistance - Returnerar det kartesiska 3D-minsta avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DExtent - Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.
- ST_3DIntersects - Testar om två geometrier korsar varandra spatialt i 3D - endast för punkter, linestrings, polygoner, polyedrisk yta (area)
- ST_3DLongestLine - Returnerar den längsta 3D-linjen mellan två geometrier
- ST_3DMaxDistance - Returnerar det maximala kartesiska 3D-avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DShortestLine - Returnerar den kortaste 3D-linjen mellan två geometrier
- ST_Affine - Tillämpa en 3D-affin transformation på en geometri.
- ST_Area - Returnerar arean för en polygonal geometri.
- ST_AsBinary - Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsEWKB - Returnerar EWKB-representationen (Extended Well-Known Binary) av geometrin med SRID-metadata.
- ST_AsEWKT - Returnera WKT-representationen (Well-Known Text) av geometrin med SRID-metadata.
- ST_AsGML - Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsX3D - Returnerar en geometri i X3D xml-nodelementformat: ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_CoordDim - Returnerar koordinatdimensionen för en geometri.
- ST_Dimension - Returnerar den topologiska dimensionen för en geometri.
- ST_Dump - Returnerar en uppsättning geometry_dump-rader för komponenterna i en geometri.
- ST_DumpPoints - Returnerar en uppsättning geometry_dump-rader för koordinaterna i en geometri.
- ST_Expand - Returnerar en bounding box som expanderats från en annan bounding box eller en geometri.
- ST_Extent - Aggregerad funktion som returnerar geometriernas avgränsande box.
- ST_FlipCoordinates - Returnerar en version av en geometri med X- och Y-axlarna vända.
- ST_Force2D - Tvinga geometrierna till ett ”2-dimensionellt läge”.
- ST_ForceRHR - Tvinga orienteringen av hörnen i en polygon att följa högerhands-regeln.
- ST_ForceSFS - Force geometries to use SFS 1.1 or 1.2 geometry types.
- ST_Force3D - Tvingar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ.
- ST_Force3DZ - Tvinga geometrierna till XYZ-läge.
- ST_ForceCollection - Konvertera geometrin till en GEOMETRYCOLLECTION.

- ST_GeomFromEWKB - Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Binary representation (EWKB).
- ST_GeomFromEWKT - Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Text representation (EWKT).
- ST_GeomFromGML - Tar som indata GML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeometryN - Returnerar ett element i en geometrisamling.
- ST_GeometryType - Returnerar SQL-MM-typen för en geometri som text.
- = - Returnerar TRUE om koordinaterna och koordinatordningen för geometri/geografi A är samma som koordinaterna och koordinatordningen för geometri/geografi B.
- &<| - Returnerar TRUE om A:s avgränsande box överlappar eller ligger under B:s.
- ~= - Returnerar TRUE om A:s avgränsande box är densamma som B:s.
- ST_IsClosed - Testar om en LineStrings start- och slutpunkter är sammanfallande. För en PolyhedralSurface testas om den är sluten (volymetrisk).
- ST_MemSize - Returnerar hur mycket minnesutrymme en geometri tar upp.
- ST_NPoints - Returnerar antalet punkter (vertices) i en geometri.
- ST_NumGeometries - Returnerar antalet element i en geometrisamling.
- ST_NumPatches - Returns the number of faces in a PolyhedralSurface or TIN.
- ST_PatchN - Returns the Nth face of a PolyhedralSurface or TIN.
- ST_RemoveRepeatedPoints - Returnerar en version av en geometri där dubbletter av punkter har tagits bort.
- ST_Reverse - Returnera geometrin med omvänd ordning på topparna.
- ST_Rotate - Roterar en geometri runt en ursprungspunkt.
- ST_RotateX - Roterar en geometri runt X-axeln.
- ST_RotateY - Roterar en geometri runt Y-axeln.
- ST_RotateZ - Roterar en geometri runt Z-axeln.
- ST_Scale - Skalar en geometri med givna faktorer.
- ST_ShiftLongitude - Flyttar longitudkoordinaterna för en geometri mellan -180..180 och 0..360.
- ST_Summary - Returnerar en textsammanfattning av innehållet i en geometri.
- ST_SwapOrdinates - Returnerar en version av den givna geometrin med givna ordinatvärden ombytta.
- ST_Transform - Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem.
- ~(box2df,box2df) - Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) innehåller en annan avgränsande 2D-box med flytande precision (BOX2DF).
- ~(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) innehåller en geometris 2D-bondingbox.
- ~(geometry,box2df) - Returnerar TRUE om en geometris 2D-bindningsbox innehåller en 2D-bindningsbox med floatprecision (GIDX).

- && - Returnerar TRUE om A:s 2D-begränsningsbox skär B:s 2D-begränsningsbox.
- &&& - Returnerar TRUE om A:s n-D-gränsbox skär B:s n-D-gränsbox.
- @(box2df,box2df) - Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) ingår i en annan avgränsande 2D-box med flytande precision.
- @(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) ingår i en geometris 2D-gränsbox.
- @(geometry,box2df) - Returnerar TRUE om en geometris 2D-begränsningsbox ingår i en 2D-begränsningsl med flytande precision (BOX2DF).
- &&(box2df,box2df) - Returnerar TRUE om två 2D-begränsningsboxar med flytande precision (BOX2DF) skär varandra.
- &&(box2df,geometry) - Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) skär en geometris (cachelagrade) 2D-gränsbox.
- &&(geometry,box2df) - Returnerar TRUE om en geometris (cachelagrade) 2D-begränsningsbox skär en 2D-begränsningsbox med flytande precision (BOX2DF).
- &&&(geometry,gidx) - Returnerar TRUE om en geometris (cachade) n-D-begränsningsbox skär en n-D-begränsningsbox med flytande precision (GIDX).
- &&&(gidx,geometry) - Returnerar TRUE om en n-D avgränsningsbox med flytande precision (GIDX) skär en geometris (cachelagrade) n-D avgränsningsbox.
- &&&(gidx,gidx) - Returnerar TRUE om två gränsboxar (GIDX) med n-D floatprecision skär varandra.

13.11 PostGIS matris för funktionsstöd

Nedan finns en alfabetisk lista över spatiala specifika funktioner i PostGIS och de spatiala typer de arbetar med eller OGC/SQL-överensstämmelse som de försöker följa.

- A betyder att funktionen fungerar med typen eller subtypen i original.
- A betyder att det fungerar men med en transform cast inbyggd med cast till geometri, transformera till en bästa sridspatial ref och sedan casta tillbaka. Resultaten kanske inte blir som förväntat för stora områden eller områden vid poler och kan ackumulera flyttalsskräp.
- A betyder att funktionen fungerar med typen på grund av en auto-cast till en annan, t.ex. till box3d, snarare än direkt typstöd.
- A betyder att funktionen endast är tillgänglig om PostGIS har kompilerats med SFCGAL-stöd.
- geom - Grundläggande stöd för 2D-geometri (x,y).
- geog - Grundläggande 2D geografiskt stöd (x,y).
- 2.5D - grundläggande 2D-geometrier i 3 D/4D-rymd (har Z- eller M-koordinat).
- PS - Polyedriska ytor
- T - Trianglar och triangulerade oregelbundna nätverksytor (TIN)

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_Collect	✓		✓	✓			
ST_LineFromMu	✓ Point		✓				
ST_MakeEnvelo	✓						
ST_MakeLine	✓		✓				
ST_MakePoint	✓		✓				
ST_MakePointM	✓						
ST_MakePolygo	✓		✓				
ST_Point	✓				✓		
ST_PointZ	✓						
ST_PointM	✓						
ST_PointZM	✓						
ST_Polygon	✓		✓		✓		
ST_TileEnvelop	✓						
ST_HexagonGri	✓						
ST_Hexagon	✓						
ST_SquareGrid	✓						
ST_Square	✓						
ST_Letters	✓						
ST_MakeNurbs	✓ ve						
GeometryType	✓		✓	✓		✓	✓
ST_Boundary	✓		✓		✓		
ST_BoundingDia	✓ nal		✓				
ST_CoordDim	✓		✓	✓	✓	✓	✓
ST_Dimension	✓				✓	✓	✓
ST_Dump	✓		✓	✓		✓	✓
ST_DumpPoints	✓		✓	✓		✓	✓
ST_DumpSegme	✓ s		✓				✓
ST_DumpRings	✓		✓				
ST_EndPoint	✓		✓	✓	✓		
ST_Envelope	✓				✓		
ST_ExteriorRing	✓		✓		✓		
ST_GeometryN	✓		✓	✓	✓	✓	✓
ST_GeometryTy	✓		✓		✓	✓	
ST_HasArc	✓		✓	✓			

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_InteriorRing	✓		✓		✓		
ST_NumCurves	✓		✓		✓		
ST_CurveN	✓		✓		✓		
ST_IsClosed	✓		✓	✓	✓	✓	
ST_IsCollection	✓		✓	✓			
ST_IsEmpty	✓			✓	✓		
ST_IsPolygonCC	✓		✓				
ST_IsPolygonCW	✓		✓				
ST_IsRing	✓				✓		
ST_IsSimple	✓		✓		✓		
ST_M	✓		✓		✓		
ST_MemSize	✓		✓	✓		✓	✓
ST_NDims	✓		✓				
ST_NPoints	✓		✓	✓		✓	
ST_NRings	✓		✓	✓			
ST_NumGeometries	✓		✓		✓	✓	✓
ST_NumInteriorRings	✓				✓		
ST_NumInteriorRing	✓						
ST_NumPatches	✓		✓		✓	✓	
ST_NumPoints	✓				✓		
ST_PatchN	✓		✓		✓	✓	✓
ST_PointN	✓		✓	✓	✓		
ST_Points	✓		✓	✓			
ST_StartPoint	✓		✓	✓	✓		
ST_StartM	✓			✓			
ST_EndM	✓			✓			
ST_SetStartM	✓			✓			
ST_SetEndM	✓			✓			
ST_Summary	✓	✓		✓		✓	✓
ST_X	✓		✓		✓		
ST_Y	✓		✓		✓		
ST_Z	✓		✓		✓		
ST_ZMFlag	✓		✓	✓			
ST_HasZ	✓		✓				

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_HasM	✓		✓				
ST_ControlPoint	✓						
ST_Degree	✓						
ST_Weights	✓						
ST_Knots	✓						
ST_NumControlPoints	✓						
ST_NurbsCurveIsRational	✓						
ST_IsNurbsCurve	✓						
ST_Evaluate	✓						
ST_NurbsToLineString	✓						
ST_AddPoint	✓		✓				
ST_CollectionExtract	✓						
ST_CollectionHomogenize	✓						
ST_CurveToLine	✓		✓	✓	✓		
ST_Scroll	✓		✓				
ST_FlipCoordinates	✓		✓	✓		✓	✓
ST_Force2D	✓		✓	✓		✓	
ST_Force3D	✓		✓	✓		✓	
ST_Force3DZ	✓		✓	✓		✓	
ST_Force3DM	✓			✓			
ST_Force4D	✓		✓	✓			
ST_ForceCollection	✓		✓	✓		✓	
ST_ForceCurve	✓		✓	✓			
ST_ForcePolygonCW	✓		✓				
ST_ForcePolygonCCW	✓		✓				
ST_ForceSFS	✓		✓	✓		✓	✓
ST_ForceRHR	✓		✓			✓	
ST_LineExtend	✓						
ST_LineToCurve	✓		✓	✓			
ST_Multi	✓						
ST_Normalize	✓						
ST_Project	✓	✓					
ST_QuantizeCoordinates	✓						
ST_RemovePoint	✓		✓				

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_RemoveRep	✓	edPoints	✓			✓	
ST_RemoveIrrel	✓	antPointsForView					
ST_RemoveSma	✓	arts					
ST_Reverse	✓		✓			✓	
ST_Segmentize	✓	✓					
ST_SetPoint	✓		✓				
ST_ShiftLongitu	✓		✓			✓	✓
ST_WrapX	✓		✓				
ST_SnapToGrid	✓		✓				
ST_Snap	✓						
ST_SwapOrdina	✓		✓	✓		✓	✓
ST_IsValid	✓				✓		
ST_IsValidDetai	✓						
ST_IsValidReas	✓						
ST_MakeValid	✓		✓				
ST_InverseTran	✓	rmPipeline					
ST_SetSRID	✓			✓			
ST_SRID	✓			✓	✓		
ST_Transform	✓			✓	✓	✓	
ST_TransformPi	✓	ine					
postgis_srs_codes							
postgis_srs							
postgis_srs_all							
postgis_srs_sea	✓						
ST_BdPolyFrom	✓	t					
ST_BdMPolyFro	✓	ext					
ST_GeogFromText		✓					
ST_GeographyFromText		✓					
ST_GeomCollFr	✓	Text			✓		
ST_GeomFromE	✓	CT	✓	✓		✓	✓
ST_GeomFromM	✓	RC21					
ST_GeometryFr	✓	Text			✓		
ST_GeomFromT	✓			✓	✓		
ST_LineFromTe	✓				✓		
ST_MLineFrom	✓	t			✓		

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_MPointFrom	✓				✓		
ST_MPolyFrom	✓				✓		
ST_PointFromT	✓				✓		
ST_PolygonFro	✓				✓		
ST_WKTToSQL	✓				✓		
ST_GeogFromWKB		✓		✓			
ST_GeomFromE	✓		✓	✓		✓	✓
ST_GeomFromW	✓			✓	✓		
ST_LineFromW	✓				✓		
ST_LinestringF	✓				✓		
ST_PointFromW	✓		✓	✓	✓		
ST_WKBToSQL	✓				✓		
ST_Box2dFromG	☑						
ST_GeomFromG	✓						
ST_GeomFromG	✓		✓			✓	✓
ST_GeomFromG	✓		✓				
ST_GeomFromK	✓		✓				
ST_GeomFromT	✓						
ST_GMLToSQL	✓				✓		
ST_LineFromEn	✓						
ST_PointFromGeoHash							
ST_FromFlatGeobufToTable							
ST_FromFlatGeobuf							
ST_AsEWKT	✓	✓	✓	✓		✓	✓
ST_AsText	✓	✓		✓	✓		
ST_AsBinary	✓	✓	✓	✓	✓	✓	✓
ST_AsEWKB	✓		✓	✓		✓	✓
ST_AsHEXEWK	✓		✓	✓			
ST_AsEncodedP	✓						
ST_AsFlatGeobu	☑						
ST_AsGeobuf	☑						
ST_AsGeoJSON	✓	✓	✓				
ST_AsGML	✓	✓	✓		✓	✓	✓
ST_AsKML	✓	✓	✓				
ST_AsLatLonTe	✓						

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_AsMARC21	✓						
ST_AsMVTGeom	✓						
ST_AsMVT	✓						
ST_AsSVG	✓	✓		✓			
ST_AsTWKB	✓						
ST_AsX3D	✓		✓			✓	✓
ST_GeoHash	✓			✓			
&&	✓	✓		✓		✓	
&&(geometry,box2df)	✓			✓		✓	
&&(box2df,geometry)	✓			✓		✓	
&&(box2df,box2df)	✓			✓		✓	
&&&	✓		✓	✓		✓	✓
&&&(geometry,gidx)	✓		✓	✓		✓	✓
&&&(gidx,geometry)	✓		✓	✓		✓	✓
&&&(gidx,gidx)			✓	✓		✓	✓
&<	✓						
&<|	✓			✓		✓	
&>	✓						
<<	✓						
<<|	✓						
=	✓	✓		✓		✓	
>>	✓						
@	✓						
@(geometry,box2df)	✓			✓		✓	
@(box2df,geometry)	✓			✓		✓	
@(box2df,box2df)	✓			✓		✓	
|&>	✓						
|>>	✓						
~	✓						
~(geometry,box2df)	✓			✓		✓	
~(box2df,geometry)	✓			✓		✓	
~(box2df,box2df)	✓			✓		✓	
~=	✓					✓	
<->	✓	✓					

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
|=|	✓						
<#>	✓						
<<->>	✓						
ST_3DIntersects	✓		✓		✓	✓	✓
ST_Contains	✓				✓		
ST_ContainsProperly	✓						
ST_CoveredBy	✓	✓					
ST_Covers	✓	✓					
ST_Crosses	✓				✓		
ST_Disjoint	✓				✓		
ST_Equals	✓				✓		
ST_Intersects	✓	✓		✓	✓		✓
ST_LineCrossingDirection	✓						
ST_OrderingEquals	✓				✓		
ST_Overlaps	✓				✓		
ST_Relate	✓				✓		
ST_RelateMatch							
ST_Touches	✓				✓		
ST_Within	✓				✓		
ST_3DDWithin	✓		✓		✓	✓	
ST_3DDFullyWithin	✓		✓			✓	
ST_DFullyWithin	✓						
ST_DWithin	✓	✓					
ST_PointInsideCircle	✓						
ST_Area	✓	✓			✓	✓	
ST_Azimuth	✓	✓					
ST_Angle	✓						
ST_ClosestPoint	✓	✓					
ST_3DClosestPoint	✓		✓			✓	
ST_Distance	✓	✓		✓	✓		
ST_3DDistance	✓		✓		✓	✓	
ST_DistanceSphere	✓						
ST_DistanceSpheroid	✓						
ST_FrechetDistance	✓						

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_HausdorffDi nce	✓						
ST_Length	✓	✓			✓		
ST_Length2D	✓						
ST_3DLength	✓		✓		✓		
ST_LengthSphe d	✓		✓				
ST_LongestLine	✓						
ST_3DLongestL	✓		✓			✓	
ST_MaxDistanc	✓						
ST_3DMaxDista e	✓		✓			✓	
ST_MinimumCl ance	✓						
ST_MinimumCl anceLine	✓						
ST_Perimeter	✓	✓			✓		
ST_Perimeter2D	✓						
ST_3DPerimeter	✓		✓		✓		
ST_ShortestLin	✓	✓					
ST_3DShortestL e	✓		✓			✓	
ST_ClipByBox2D	✓						
ST_Difference	✓		✓		✓		
ST_Intersection	✓	😁	✓		✓		
ST_MemUnion	✓		✓				
ST_Node	✓		✓				
ST_Split	✓						
ST_Subdivide	✓						
ST_SymDifferen	✓		✓		✓		
ST_UnaryUnion	✓		✓				
ST_Union	✓		✓		✓		
ST_Buffer	✓	😁			✓		
ST_BuildArea	✓						
ST_Centroid	✓	✓			✓		
ST_ChaikinSmo ing	✓		✓				
ST_CatmullRom oothing	✓		✓				
ST_ConcaveHul	✓						
ST_ConvexHull	✓		✓		✓		
ST_DelaunayTri gles	✓		✓				✓

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_FilterByM	✓						
ST_GeneratePoi	✓						
ST_GeometricM	✓	an		✓			
ST_LineMerge	✓						
ST_MaximumIn	✓	bedCircle					
ST_LargestEmp	✓	ircle					
ST_MinimumBo	✓	lingCircle					
ST_MinimumBo	✓	lingRadius					
ST_OrientedEnv	✓	pe					
ST_OffsetCurve	✓						
ST_PointOnSurf	✓		✓		✓		
ST_Polygonize	✓						
ST_ReducePrec	✓	n					
ST_SharedPaths	✓						
ST_Simplify	✓						
ST_SimplifyPres	✓	veTopology					
ST_SimplifyPoly	✓	nHull					
ST_SimplifyVW	✓						
ST_SetEffective	✓	a					
ST_Triangulatel	✓	gon					
ST_VoronoiLine	✓						
ST_VoronoiPoly	✓	s					
ST_CoverageInv	✓	dEdges					
ST_CoverageCl	✓						
ST_CoverageSi	✓	ify					
ST_CoverageEd	✓	s					
ST_CoverageUn	✓						
ST_Affine	✓		✓	✓		✓	✓
ST_Rotate	✓		✓	✓		✓	✓
ST_RotateX	✓		✓			✓	✓
ST_RotateY	✓		✓			✓	✓
ST_RotateZ	✓		✓	✓		✓	✓
ST_Scale	✓		✓	✓		✓	✓
ST_Translate	✓		✓	✓			

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_TransScale	✓		✓	✓			
ST_ClusterDBS	✓			✓			
ST_ClusterInter	✓						
ST_ClusterInter	✓						
ST_ClusterRela	✓						
ST_MinimumSp	✓						
ST_ClusterKMe	✓						
ST_ClusterWith	✓			✓			
ST_ClusterWith	✓			✓			
Box2D	✓			✓		✓	✓
Box3D	✓		✓	✓		✓	✓
ST_EstimatedEx	✓			✓			
ST_Expand	✓					✓	✓
ST_Extent	✓					✓	✓
ST_3DExtent	✓		✓	✓		✓	✓
ST_MakeBox2D	✓						
ST_3DMakeBox	✓						
ST_XMax	✓		✓	✓			
ST_XMin	✓		✓	✓			
ST_YMax	✓		✓	✓			
ST_YMin	✓		✓	✓			
ST_ZMax	✓		✓	✓			
ST_ZMin	✓		✓	✓			
ST_XSize	✓		✓	✓			
ST_YSize	✓		✓	✓			
ST_ZSize	✓		✓	✓			
ST_MMin	✓		✓	✓			
ST_MMax	✓		✓	✓			
ST_MSize	✓		✓	✓			
ST_LineInterpol	✓	✓	✓				
ST_3DLineInter	✓		✓				
ST_LineInterpol	✓	✓	✓				
ST_LineLocateP	✓	✓					
ST_LineSubstri	✓	✓	✓				

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_LocateAlong	✓				✓		
ST_LocateBetw	✓				✓		
ST_LocateBetw	✓ Elevations		✓				
ST_InterpolateP	✓ t		✓				
ST_3DInterpola	✓ oint		✓				
ST_AddMeasure	✓		✓				
ST_IsValidTraje	✓ ry		✓				
ST_ClosestPoint	✓ Approach		✓				
ST_DistanceCPA	✓		✓				
ST_CPAWithin	✓		✓				
postgis.gdal_datapath							
postgis.gdal_enabled_drivers							
postgis.enable_outdb_rasters							
postgis.gdal_vsi_options							
postgis.gdal_cpl_debug							
PostGIS_AddBB	✓			✓			
PostGIS_DropBB	✓			✓			
PostGIS_HasBB	✓			✓			
postgis_sfcgal_version							
postgis_sfcgal_full_version							
CG_ForceLHR							
CG_IsPlanar							
CG_IsSolid							
CG_MakeSolid							
CG_Orientatior							
CG_Area							
CG_3DArea							
CG_Volume							
CG_Intersectio							
CG_Intersects							
CG_3DIntersec							
CG_Difference							

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
CG_3DDifferen							
CG_Distance							
CG_3DDistance							
CG_3DConvexH							
CG_3DIntersec							
CG_Union							
CG_3DUnion							
CG_AlphaShap							
CG_ApproxCon		artition					
CG_Approxima		dialAxis					
CG_GenerateFl		oof					
CG_GenerateH		dRoof					
CG_GenerateG		Roof					
CG_GenerateSl		nRoof					
CG_GenerateR							
CG_PolygonRe							
CG_Constraine		aunayTriangles					
CG_Extrude							
CG_ExtrudeStr		tSkeleton					
CG_GreeneApp		onvexPartition					
CG_Minkowski							
CG_OptimalAlp		ıape					
CG_OptimalCo		Partition					
CG_StraightSk		n					

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
CG_Tesselate							
CG_Triangulate							
CG_Visibility							
CG_YMonotone	ition						
CG_StraightSk	nPartition						
CG_3DBuffer							
CG_Rotate							
CG_2DRotate							
CG_3DRotate							
CG_RotateX							
CG_RotateY							
CG_RotateZ							
CG_Scale							
CG_3DScale							
CG_3DScaleAr	Center						
CG_Translate							
CG_3DTranslat							
CG_Simplify							
CG_3DAlphaWr	ng						
CG_NurbsCurv	erpolate						
CG_NurbsCurv	proximate						
CG_NurbsCurv	ivative						
getfaceedges_returntype							
TopoGeometry							
validatetopology_returntype							
TopoElement							
TopoElementArray							

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
AddTopoGeometryColumn							
RenameTopoGeometryColumn							
DropTopology							
RenameTopology							
DropTopoGeometryColumn							
FixCorruptTopoGeometryColumn							
Populate_Topology_Layer							
TopologySummary							
ValidateTopolog	✓						
ValidateTopologyRelation							
ValidateTopolog	✓ ecision						
MakeTopologyP	✓ ise						
FindTopology	✓						
FindLayer	✓						
TotalTopologySize							
UpgradeTopology							
CreateTopology							
CopyTopology							
ST_InitTopoGeo					✓		
ST_CreateTopo	✓				✓		
TopoGeo_AddPo	✓						
TopoGeo_AddLi	✓ tring						
TopoGeo_AddPo	✓ on						
TopoGeo_LoadG	✓ metry						
ST_AddIsoNode	✓				✓		
ST_AddIsoEdge	✓				✓		
ST_AddEdgeNe	✓ aces				✓		
ST_AddEdgeMo	✓ ace				✓		
ST_RemEdgeNewFace					✓		
ST_RemEdgeModFace					✓		
ST_ChangeEdge	✓ om				✓		
ST_ModEdgeSp	✓				✓		
ST_ModEdgeHeal					✓		
ST_NewEdgeHeal					✓		
ST_MoveIsoNod	✓				✓		
ST_NewEdgesS	✓				✓		
ST_RemoveIsoNode					✓		
ST_RemoveIsoEdge					✓		
GetEdgeByPoin	✓						

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
GetFaceByPoint	✓						
GetFaceContainPoint	✓						
GetNodeByPoint	✓						
GetTopologyID							
GetTopologySRID							
GetTopologyName							
ST_GetFaceEdges					✓		
ST_GetFaceGeometry	✓				✓		
GetRingEdges							
GetNodeEdges							
FindVertexSegmentPairsBelowDistance							
Polygonize							
AddNode	✓						
AddEdge	✓						
AddFace	✓						
ST_Simplify	✓						
RemoveUnusedPrimitives	✓						
CreateTopoGeom	✓						
toTopoGeom	✓						
TopoElementArray_Agg							
TopoElement	✓						
clearTopoGeom	✓						
TopoGeom_addElement	✓						
TopoGeom_remElement	✓						
TopoGeom_addToGeom	✓						
toTopoGeom							
GetTopoGeomElementArray							
GetTopoGeomElements							
ST_SRID	✓						
AsGML	✓						
AsTopoJSON	✓						
Equals	✓		✓				
Intersects	✓		✓				
geomval							
addbandarg							
rastbandarg							
raster							
reclassarg							
summarystats							
unionarg							
AddRasterConstraints							
DropRasterConstraints							

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
AddOverviewConstraints							
DropOverviewConstraints							
PostGIS_GDAL_Version							
PostGIS_Raster_Lib_Build_Date							
PostGIS_Raster_Lib_Version							
ST_GDALDrivers							
UpdateRasterSRID							
ST_CreateOverview							
ST_AddBand							
ST_AsRaster	✓						
ST_AsRasterAgg	✓						
ST_Band							
ST_MakeEmptyCoverage							
ST_MakeEmptyRaster							
ST_Tile							
ST_Retile	✓						
ST_FromGDALRaster							
ST_GeoReference							
ST_Height							
ST_IsEmpty							
ST_MemSize							
ST_MetaData							
ST_NumBands							
ST_PixelHeight							
ST_PixelWidth							
ST_ScaleX							
ST_ScaleY							
ST_RasterToWorldCoord							
ST_RasterToWorldCoordX							
ST_RasterToWorldCoordY							
ST_Rotation							
ST_SkewX							
ST_SkewY							
ST_SRID							
ST_Summary							
ST_UpperLeftX							
ST_UpperLeftY							
ST_Width							
ST_WorldToRasCoord	✓						
ST_WorldToRasCoordX	✓						
ST_WorldToRasCoordY	✓						
ST_BandMetaData							
ST_BandNoDataValue							
ST_BandIsNoData							
ST_BandPath							
ST_BandFileSize							
ST_BandFileTimestamp							
ST_BandPixelType							
ST_MinPossibleValue							
ST_HasNoBand							
ST_PixelAsPolyg	✓						

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_PixelAsPolygons							
ST_PixelAsPoint	✓						
ST_PixelAsPoints							
ST_PixelAsCentroid	✓						
ST_PixelAsCentroids							
ST_Value	✓						
ST_NearestValue	✓						
ST_SetZ	✓						
ST_SetM	✓						
ST_Neighborhood	✓						
ST_SetValue	✓						
ST_SetValues							
ST_DumpValues							
ST_PixelOfValue							
ST_SetGeoReference							
ST_SetRotation							
ST_SetScale							
ST_SetSkew							
ST_SetSRID							
ST_SetUpperLeft							
ST_Resample							
ST_Rescale							
ST_Reskew							
ST_SnapToGrid							
ST_Resize							
ST_Transform							
ST_SetBandNoDataValue							
ST_SetBandIsNoData							
ST_SetBandPath							
ST_SetBandIndex							
ST_Count							
ST_CountAgg							
ST_Histogram							
ST_Quantile							
ST_SummaryStats							
ST_SummaryStatsAgg							
ST_ValueCount							
ST_RastFromWKB							
ST_RastFromHexWKB							
ST_AsBinary/ST_AsWKB							
ST_AsHexWKB							
ST_AsGDALRaster							
ST_AsJPEG							
ST_AsPNG							
ST_AsTIFF							
ST_Clip	✓						
ST_ColorMap							
ST_Grayscale							
ST_Intersection	✓						

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_MapAlgebra (callback function version)							
ST_MapAlgebra (expres-sion version)							
ST_MapAlgebraExpr							
ST_MapAlgebraExpr							
ST_MapAlgebraFct							
ST_MapAlgebraFct							
ST_MapAlgebraFctNgb							
ST_Reclass							
ST_ReclassExact							
ST_Union							
ST_Distinct4ma							
ST_InvDistWeight4ma							
ST_Max4ma							
ST_Mean4ma							
ST_Min4ma							
ST_MinDist4ma							
ST_Range4ma							
ST_StdDev4ma							
ST_Sum4ma							
ST_Aspect							
ST_HillShade							
ST_Roughness							
ST_Slope							
ST_TPI							
ST_TRI							
ST_InterpolateR✓er							
ST_Contour							
Box3D	☑						
ST_ConvexHull	✓						
ST_DumpAsPolygons							
ST_Envelope	✓						
ST_MinConvexH✓	✓						
ST_Polygon	✓						
ST_Intersection✓ctions	✓						
&&	✓						
&<							
&>							
=							
@	✓						
~=							
~	✓						
ST_Contains							
ST_ContainsProperly							

Funktion	geom	geog	2.5D	Kurvor	SQL MM	PS	T
ST_Covers							
ST_CoveredBy							
ST_Disjoint							
ST_Intersects	✓						
ST_Overlaps							
ST_Touches							
ST_SameAlignment							
ST_NotSameAlignmentReason							
ST_Within							
ST_DWithin							
ST_DFullyWithin							

13.12 Nya, förbättrade eller ändrade PostGIS-funktioner

13.12.1 PostGIS Funktioner nya eller förbättrade i 3.7

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.7

- CG_ApproximateMedialAxis - Availability: 3.7.0 - projected parameter. Requires SFCGAL >= 2.3.0 for projected result; falls back to non-projected with a notice on older versions. Beräkna den ungefärliga mediala axeln för en arealgeometri.
- CG_GenerateFlatRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a flat (box) roof from a footprint polygon.
- CG_GenerateGableRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a gable roof from a footprint polygon.
- CG_GenerateHippedRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a hipped roof from a footprint polygon.
- CG_GenerateRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a roof of the requested type from a footprint polygon.
- CG_GenerateSkillionRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a skillion (single-slope) roof from a footprint polygon.
- CG_NurbsCurveApproximate - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Creates an approximating NURBS curve fitting data points within a tolerance
- CG_NurbsCurveDerivative - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Computes the derivative (tangent, curvature) of a NURBS curve at a given parameter
- CG_NurbsCurveInterpolate - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Creates an interpolating NURBS curve passing through all given data points
- CG_PolygonRepair - Availability: 3.7.0 - requires SFCGAL >= 2.3.0 and CGAL >= 6.0. Repair an invalid polygon or multipolygon.
- FindVertexSegmentPairsBelowDistance - Availability: 3.7 Find pairs of topology vertex/segment that are closer than tolerated distance
- ST_3DInterpolatePoint - Availability: 3.7.0 Returns the interpolated measure of a geometry closest to a point in 3D.

- ST_ClusterRelateWin - Availability: 3.7.0 Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.
- ST_ControlPoints - Availability: 3.7.0 Returns the control points of a NURBS curve as a MULTIPOINT geometry.
- ST_CoverageEdges - Availability: 3.7.0 Computes the unique edges of a polygonal coverage.
- ST_Degree - Availability: 3.7.0 Returns the polynomial degree of a NURBS curve.
- ST_Evaluate - Availability: 3.7.0 Evaluates a NURBS curve at a specific parameter value and returns the resulting point.
- ST_IsNurbsCurve - Availability: 3.7.0 Checks if a geometry is a NURBS curve.
- ST_Knots - Availability: 3.7.0 Returns the knot vector of a NURBS curve.
- ST_MSize - Availability: 3.7.0 Returns the M size of a geometry.
- ST_MakeLine - Availability: 3.7.0 - Support for MultiLineString input elements was introduced Skapar en LineString från Point-, MultiPoint- eller LineString-geometrier.
- ST_MakeNurbsCurve - Availability: 3.7.0 Creates a NURBS (Non-Uniform Rational B-Spline) curve from control points, optional weights, and optional knot vector.
- ST_MinimumSpanningTree - Availability: 3.7.0 Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.
- ST_NumControlPoints - Availability: 3.7.0 Returns the number of control points in a NURBS curve.
- ST_NurbsCurveIsRational - Availability: 3.7.0 Checks if a NURBS curve is rational (has weights).
- ST_NurbsToLineString - Availability: 3.7.0 Converts a NURBS curve to a LINESTRING by uniform sampling.
- ST_Weights - Availability: 3.7.0 Returns the weight array of a rational NURBS curve.
- ST_XSize - Availability: 3.7.0 Returns the X size of a 2D or 3D bounding box or a geometry.
- ST_YSize - Availability: 3.7.0 Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_ZSize - Availability: 3.7.0 Returns the Z size of a 2D or 3D bounding box or a geometry.

Funktioner förbättrade i PostGIS 3.7

- ST_AsX3D - Enhanced: 3.7.0: Support for curve geometries. NURBSCURVE is emitted as X3D NurbsCurve; other curve types are linearized before output. Returnerar en geometri i X3D xml-nodelementformat: ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_GMLToSQL - Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced. Returns a geometry from GML. Alias for ST_GeomFromGML.
- ST_GeomFromGML - Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced. Tar som indata GML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_LongestLine - Enhanced: 3.7.0 - support for a single geometry input. Returnerar den längsta 2D-linjen mellan två geometrier.
- ST_MakePolygon - Enhanced: 3.7.0 - Support for curved input rings was introduced. Creates a Polygon or CurvePolygon from a shell and optional list of holes.

- ST_MaxDistance - Enhanced: 3.7.0 - support for a single geometry input. Returnerar det största 2D-avståndet mellan två geometrier i projicerade enheter.
- ST_Value - Enhanced: 3.7.0 resample accepts nearest-neighbor boundary options nearest-UL", nnearest-UR", nearest-LL", and nearest-LR". Returnerar värdet för ett visst band i en viss kolumnx, radpixel eller vid en viss geometrisk punkt. Bandnummer börjar på 1 och antas vara 1 om det inte anges. Om exclude_nodata_value är satt till false, anses alla pixlar inklusive nodata-pixlar korsa varandra och returnerar värdet. Om värdet exclude_nodata_value inte anges läses det från rastrets metadata.
- TopoGeo_AddLineString - Enhanced: 3.7.0 added support for limiting the number of new edges created in the topology. Lägger till en linestrings till en befintlig topologi med hjälp av en tolerans och eventuellt delning av befintliga kanter/ytor.

Funktioner ändrade i PostGIS 3.7

- ST_AsGeoJSON - Changed: 3.7.0 added warning about duplicate keys Returnerar en geometri eller funktion i GeoJSON-format.
- toTopoGeom - Changed: 3.7.0 changes meaning of tolerance 0 to really mean 0 and -1 (new default) to request using the topology precision. Konverterar en enkel geometri till en topogeometri.

13.12.2 PostGIS Funktioner nya eller förbättrade i 3.6

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.6

- CG_2DRotate - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Roterar en geometri med en given vinkel runt en angiven punkt i 2D.
- CG_3DAlphaWrapping - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.1.0 Beräknar en 3D Alpha-wrapping som strikt omsluter en geometri.
- CG_3DBuffer - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Beräknar en 3D-buffert runt en geometri.
- CG_3DRotate - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Roterar en geometri i 3D-rymden runt en axelvektor.
- CG_3DScale - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Skalar en geometri med separata faktorer längs X-, Y- och Z-axlarna.
- CG_3DScaleAroundCenter - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Skalar en geometri i 3D-rymden runt en angiven mittpunkt.
- CG_3DTranslate - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Translaterar (flyttar) en geometri med hjälp av givna offsets i 3D-rymden.
- CG_Rotate - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Roterar en geometri med en given vinkel runt origo (0,0).
- CG_RotateX - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Roterar en geometri runt X-axeln med en given vinkel.
- CG_RotateY - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Roterar en geometri runt Y-axeln med en given vinkel.
- CG_RotateZ - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Roterar en geometri runt Z-axeln med en given vinkel.

- CG_Scale - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Skalar en geometri enhetligt i alla dimensioner med en given faktor.
- CG_Simplify - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.1.0 Minskar komplexiteten i en geometri samtidigt som viktiga egenskaper och Z/M-värden bevaras.
- CG_StraightSkeletonPartition - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0. Beräknar den raka skelettpartitionen av en polygon.
- CG_Translate - Tillgänglighet: 3.6.0 - kräver SFCGAL >= 2.0.0 Translaterar (flyttar) en geometri med hjälp av givna offsets i 2D-rymden.
- FixCorruptTopoGeometryColumn - Availability: 3.6.1 Fixes topogeometry corruption caused by upgrade to postgis_topology 3.6.0 and higher
- MakeTopologyPrecise - Tillgänglighet: 3.6.0 Fäst topologivinklar till precisionsrutnätet.
- ST_AsRasterAgg - Tillgänglighet: 3.6.0 Aggregera. Renderar PostGIS-geometrier till ett nytt raster.
- ST_CatmullRomSmoothing - Tillgänglighet: 3.6.0 Returns a smoothed version of a geometry, using the Catmull-Rom spline algorithm
- ST_CoverageClean - Tillgänglighet: 3.6.0 - kräver GEOS >= 3.14.0 Beräknar en ren (kantmatchad, icke-överlappande, gap-cleared) polygontäckning, givet en icke ren indata.
- ST_IntersectionFractions - Tillgänglighet: 3.6.0 Kräver GEOS 3.14 eller högre. Beräknar den del av varje rastercell som täcks av en given geometri.
- ST_ReclassExact - Tillgänglighet: 3.6.0 Skapar ett nytt raster som består av band som omklassificerats från originalbandet med hjälp av en 1:1-mappning från värden i originalbandet till nya värden i destinationsbandet.
- TotalTopologySize - Tillgänglighet: 3.6.0 Totalt diskutrymme som används av den angivna topologin, inklusive alla index och TOAST-data.
- UpgradeTopology - Tillgänglighet: 3.6.0 Uppgraderar den angivna topologin till att stödja stora ids (int8) för topologi och primitiva ids.
- ValidateTopologyPrecision - Tillgänglighet: 3.6.0 Returnerar icke-precisa toppunkter i topologin.
- postgis.gdal_cpl_debug - Tillgänglighet: 3.6.0 En boolesk konfiguration för att aktivera eller inaktivera loggning av GDAL-felsökningsmeddelanden.

Funktioner förbättrade i PostGIS 3.6

- CreateTopology - Enhanced: 3.6.0 added topoid and usesLargeIDs parameters. Skapar ett nytt topologischema och registrerar det i tabellen topology.topology.

Funktioner ändrade i PostGIS 3.6

- ST_GeometryN - Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and are not decomposed into patches; use ST_PatchN to access faces. Returnerar ett element i en geometrisamling.
- ST_NumGeometries - Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and return 1; use ST_NumPatches to count faces. Returnerar antalet element i en geometrisamling.

13.12.3 PostGIS Funktioner nya eller förbättrade i 3.5

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.5

- CG_3DArea - Tillgänglighet: 3.5.0 Beräknar area för 3D-ytgeometrier. Returnerar 0 för solider.
- CG_3DConvexHull - Tillgänglighet: 3.5.0 Beräknar den konvexa 3D-skålen för en geometri.
- CG_3DDifference - Tillgänglighet: 3.5.0 Utföra 3D-differens
- CG_3DDistance - Tillgänglighet: 3.5.0 Beräknar det minsta 3D-avståndet mellan två geometrier
- CG_3DIntersection - Tillgänglighet: 3.5.0 Utför 3D-intersektion
- CG_3DIntersects - Tillgänglighet: 3.5.0 Testar om två 3D-geometrier korsar varandra
- CG_3DUnion - Tillgänglighet: 3.5.0 Utför 3D-union med hjälp av postgis_sfcgal.
- CG_AlphaShape - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.4.1. Beräknar en Alpha-form som omsluter en geometri
- CG_ApproxConvexPartition - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0. Beräknar approximal konvex partition av polygongeometrin
- CG_ApproximateMedialAxis - Tillgänglighet: 3.5.0 Beräkna den ungefärliga mediala axeln för en arealgeometri.
- CG_Area - Tillgänglighet: 3.5.0 Beräknar arean av en geometri
- CG_Difference - Tillgänglighet: 3.5.0 Beräknar den geometriska skillnaden mellan två geometrier
- CG_Distance - Tillgänglighet: 3.5.0 Beräknar det minsta avståndet mellan två geometrier
- CG_Extrude - Tillgänglighet: 3.5.0 Extrudera en yta till en relaterad volym
- CG_ExtrudeStraightSkeleton - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0. Extrudering av raka skelett
- CG_ForceLHR - Tillgänglighet: 3.5.0 Tvinga fram LHR-orientering
- CG_GreeneApproxConvexPartition - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0. Beräknar approximal konvex partition av polygongeometrin
- CG_Intersection - Tillgänglighet: 3.5.0 Beräknar skärningspunkten mellan två geometrier
- CG_Intersects - Tillgänglighet: 3.5.0 Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- CG_IsPlanar - Tillgänglighet: 3.5.0 Kontrollera om en yta är plan eller inte
- CG_IsSolid - Tillgänglighet: 3.5.0 Testar om geometrin är en solid. Ingen validitetskontroll utförs.
- CG_MakeSolid - Tillgänglighet: 3.5.0 Casta geometrin till en solid. Ingen kontroll utförs. För att få en giltig solid måste indatageometrin vara en sluten polyedrisk yta eller en sluten TIN.
- CG_MinkowskiSum - Tillgänglighet: 3.5.0 Utför Minkowski-summa
- CG_OptimalAlphaShape - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.4.1. Beräknar en alpha-form som omsluter en geometri med ett öptimaltälpha-värde.
- CG_OptimalConvexPartition - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0. Beräknar en optimal konvex partition av polygongeometrin
- CG_Orientation - Tillgänglighet: 3.5.0 Bestäm ytans orientering

- CG_StraightSkeleton - Tillgänglighet: 3.5.0 Beräkna ett rakt skelett från en geometri
- CG_Tesselate - Tillgänglighet: 3.5.0 Tessellerar ytan på en polygon eller polyederyta och returnerar som en TIN eller en samling av TINS
- CG_Triangulate - Tillgänglighet: 3.5.0 Triangulerar en polygonal geometri
- CG_Union - Tillgänglighet: 3.5.0 Beräknar föreningen av två geometrier
- CG_Visibility - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0. Beräkna en synlighetspolygon från en punkt eller ett segment i en polygongeometri
- CG_Volume - Tillgänglighet: 3.5.0 Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.
- CG_YMonotonePartition - Tillgänglighet: 3.5.0 - kräver SFCGAL >= 1.5.0. Beräknar y-monoton partition av polygongeometrin
- ST_HasM - Tillgänglighet: 3.5.0 Kontrollerar om en geometri har en M (mått)-dimension.
- ST_HasZ - Tillgänglighet: 3.5.0 Kontrollerar om en geometri har en Z-dimension.
- ST_RemoveIrrelevantPointsForView - Tillgänglighet: 3.5.0 Tar bort punkter som är irrelevanta för rendering av en specifik rektangulär vy av en geometri.
- ST_RemoveSmallParts - Tillgänglighet: 3.5.0 Tar bort små delar (polygonringar eller linestrings) av en geometri.
- TopoGeo_LoadGeometry - Tillgänglighet: 3.5.0 Läs in en geometri i en befintlig topologi, snappa och dela efter behov.

Funktioner förbättrade i PostGIS 3.5

- ST_Clip - Förbättrad: 3.5.0 - rörda argument har lagts till. Returnerar det raster som klippts av indatageometrin. Om bandnummer inte anges bearbetas alla band. Om crop inte anges eller om TRUE anges, beskärs utdatarastret. Om touched är inställt på TRUE inkluderas pixlar som berörs, annars inkluderas endast pixlar vars mittpunkt ligger i geometrin.

Funktioner ändrade i PostGIS 3.5

- ST_AsGeoJSON - Ändrad: 3.5.0 gör det möjligt att ange kolumnen som innehåller funktionens id Returnerar en geometri eller funktion i GeoJSON-format.
- ST_DFullyWithin - Ändrad: 3.5.0 : logiken bakom funktionen använder nu ett test av inneslutning inom en buffert, snarare än ST_MaxDistance-algoritmen. Resultaten kommer att skilja sig från tidigare versioner, men bör ligga närmare användarnas förväntningar. Testar om en geometri är helt inom ett avstånd från en annan

13.12.4 PostGIS Funktioner nya eller förbättrade i 3.4

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.4

- PostGIS_GEOS_Compiled_Version - Tillgänglighet: 3.4.0 Returnerar versionsnumret för GEOS-biblioteket som PostGIS byggdes mot.
- PostGIS_PROJ_Compiled_Version - Tillgänglighet: 3.5.0 Returnerar versionsnumret för PROJ-biblioteket som PostGIS byggdes mot.
- RenameTopoGeometryColumn - Tillgänglighet: 3.4.0 Byter namn på en topogeometri-kolumn

- RenameTopology - Tillgänglighet: 3.4.0 Byter namn på en topologi
- ST_ClusterIntersectingWin - Tillgänglighet: 3.4.0 Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri och klustrar indatageometrier i sammanhängande uppsättningar.
- ST_ClusterWithinWin - Tillgänglighet: 3.4.0 Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri, klustring med hjälp av separationsavstånd.
- ST_CoverageInvalidEdges - Tillgänglighet: 3.4.0 Fönsterfunktion som hittar platser där polygonerna inte bildar en giltig täckning.
- ST_CoverageSimplify - Tillgänglighet: 3.4.0 Fönsterfunktion som förenklar kanterna på en polygonal täckning.
- ST_CoverageUnion - Tillgänglighet: 3.4.0 - kräver GEOS >= 3.8.0 Beräknar unionen av en uppsättning polygoner som bildar en täckning genom att ta bort gemensamma kanter.
- ST_InverseTransformPipeline - Tillgänglighet: 3.4.0 Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem med hjälp av inversen av en definierad pipeline för koordinattransformation.
- ST_LargestEmptyCircle - Tillgänglighet: 3.4.0. Beräknar den största cirkeln som inte överlappar en geometri.
- ST_LineExtend - Tillgänglighet: 3.4.0 Returnerar en linje som sträcker sig framåt och bakåt med angivna avstånd.
- ST_TransformPipeline - Tillgänglighet: 3.4.0 Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem med hjälp av en definierad pipeline för koordinattransformation.
- TopoElement - Tillgänglighet: 3.4.0 Konverterar en topogeometri till ett topoelement.
- postgis_srs - Tillgänglighet: 3.4.0 Returnerar en metadatapost för den begärda myndigheten och srid.
- postgis_srs_all - Tillgänglighet: 3.4.0 Returnera metadataposter för varje spatialt referenssystem i den underliggande Proj-databasen.
- postgis_srs_codes - Tillgänglighet: 3.4.0 Returnerar listan över SRS-koder som är associerade med den angivna myndigheten.
- postgis_srs_search - Tillgänglighet: 3.4.0 Returnera metadataposter för projicerade koordinatsystem som har användningsområden som helt innehåller parametern bounds.

Funktioner förbättrade i PostGIS 3.4

- PostGIS_Full_Version - Förbättrad: 3.4.0 innehåller nu extra PROJ-konfigurationer NETWORK_ENABLED, URL_ENDPOINT och DATABASE_PATH för proj.db-platsen Rapporterar fullständig information om PostGIS-version och byggkonfiguration.
- PostGIS_PROJ_Version - Förbättrad: 3.4.0 inkluderar nu NETWORK_ENABLED, URL_ENDPOINT och DATABASE_PATH för proj.db-platsen Returnerar versionsnumret för PROJ4-biblioteket.
- ST_AsSVG - Förbättrad: 3.4.0 för stöd för alla kurvtyper Returnerar SVG-banedata för en geometri.
- ST_ClosestPoint - Förbättrad: 3.4.0 - Stöd för geografi. Returnerar den 2D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste linjen från den ena geometrin till den andra.
- ST_LineSubstring - Förbättrad: 3.4.0 - Stöd för geografi infördes. Returnerar delen av en linje mellan två fraktionerade platser.

- ST_Project - Förbättrad: 3.4.0 Tillåt geometriargument och tvåpunktsform som utelämnar azimut. Returnerar en punkt som projiceras från en startpunkt med ett avstånd och en bäring (azimut).
- ST_Resample - Förbättrad: 3.4.0 max- och min-alternativ för omsampling har lagts till Resampla ett raster med hjälp av en specificerad resamplingsalgoritm, nya dimensioner, ett godtyckligt rutnätshörn och en uppsättning rastergeoreferensattribut som definierats eller lånats från ett annat raster.
- ST_Rescale - Förbättrad: 3.4.0 max- och min-alternativ för omsampling har lagts till Resampla ett raster genom att endast justera dess skala (eller pixelstorlek). Nya pixelvärden beräknas med hjälp av omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline, Lanczos, Max eller Min. Standard är NearestNeighbor.
- ST_ShortestLine - Förbättrad: 3.4.0 - stöd för geografi. Returnerar den kortaste 2D-linjen mellan två geometrier

Funktioner ändrade i PostGIS 3.4

- PostGIS_Extensions_Upgrade - Ändrad: 3.4.0 för att lägga till argumentet target_version. Paketerar och uppgraderar PostGIS-tillägg (t.ex. postgis_raster, postgis_topology, postgis_sfcgal) till given eller senaste version.

13.12.5 PostGIS Funktioner nya eller förbättrade i 3.3

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.3

- RemoveUnusedPrimitives - Tillgänglighet: 3.3.0 Tar bort topologiprimitiver som inte behövs för att definiera befintliga TopoGeometry-objekt.
- ST_AsMARC21 - Tillgänglighet: 3.3.0 Returnerar geometri som en MARC21/XML-post med ett geografiskt datafält (034).
- ST_GeomFromMARC21 - Tillgänglighet: 3.3.0, kräver libxml2 2.6+ Tar MARC21/XML-geografiska data som indata och returnerar ett PostGIS-geometriobjekt.
- ST_Letters - Tillgänglighet: 3.3.0 Returnerar inmatade bokstäver som återges som geometri med en standardstartposition vid origo och en standardtexthöjd på 100.
- ST_SimplifyPolygonHull - Tillgänglighet: 3.3.0. Beräknar ett förenklat topologibevarande yttre eller inre skrov av en polygonal geometri.
- ST_TriangulatePolygon - Tillgänglighet: 3.3.0. Beräknar den begränsade Delaunay-trianguleringen av polygoner
- postgis_sfcgal_full_version - Tillgänglighet: 3.3.0 Returnerar den fullständiga versionen av SFCGAL som används, inklusive CGAL- och Boost-versioner

Funktioner förbättrade i PostGIS 3.3

- ST_ConcaveHull - Förbättrad: 3.3.0, inbyggd GEOS-implementering aktiverad för GEOS 3.11+ Beräknar en eventuellt konkav geometri som innehåller alla indatageometrins toppar
- ST_LineMerge - Förbättrad: 3.3.0 acceptera en riktad parameter. Returnerar de linjer som bildas genom att sy ihop en MultiLineString.

Funktioner ändrade i PostGIS 3.3

- PostGIS_Extensions_Upgrade - Ändrat: 3.3.0 stöd för uppgraderingar från alla PostGIS-versioner. Fungerar inte på alla system. Paketerar och uppgraderar PostGIS-tillägg (t.ex. postgis_raster, postgis_topology, postgis_sfcgal) till given eller senaste version.

13.12.6 PostGIS Funktioner nya eller förbättrade i 3.2

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.2

- FindLayer - Tillgänglighet: 3.2.0 Returnerar en topology.layer-post på olika sätt.
- FindTopology - Tillgänglighet: 3.2.0 Returnerar en topologipost på olika sätt.
- GetFaceContainingPoint - Tillgänglighet: 3.2.0 Hittar den yta som innehåller en punkt.
- ST_AsFlatGeobuf - Tillgänglighet: 3.2.0 Returnerar en FlatGeobuf-representation av en uppsättning rader.
- ST_Contour - Tillgänglighet: 3.2.0 Skapar en uppsättning vektorkonturer från det tillhandahållna rasterbandet med hjälp av GDAL-konturalgoritmen.
- ST_DumpSegments - Tillgänglighet: 3.2.0 Returnerar en uppsättning geometry_dump-rader för segmenten i en geometri.
- ST_FromFlatGeobuf - Tillgänglighet: 3.2.0 Läser FlatGeobuf-data.
- ST_FromFlatGeobufToTable - Tillgänglighet: 3.2.0 Skapar en tabell baserad på strukturen i FlatGeobuf-data.
- ST_InterpolateRaster - Tillgänglighet: 3.2.0 Interpolerar en rutnätsyta baserat på en indatauppsättning av 3D-punkter, med hjälp av X- och Y-värdena för att positionera punkterna i rutnätet och punkternas Z-värde som ytans höjd.
- ST_SRID - Tillgänglighet: 3.2.0 Returnerar den spatiala referensidentifieraren för en topogeometri.
- ST_Scroll - Tillgänglighet: 3.2.0 Ändra startpunkt för en sluten LineString.
- ST_SetM - Tillgänglighet: 3.2.0 Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopierade till M-dimensionen med hjälp av den begärda resamplealgoritmen.
- ST_SetZ - Tillgänglighet: 3.2.0 Returnerar en geometri med samma X/Y-koordinater som indatageometrin, och värden från rastret kopierade till Z-dimensionen med hjälp av den begärda resamplealgoritmen.
- TopoGeom_addTopoGeom - Tillgänglighet: 3.2 Lägger till element i en TopoGeometry till definitionen av en annan TopoGeometry.
- ValidateTopologyRelation - Tillgänglighet: 3.2.0 Returnerar information om ogiltiga topologirelationsposter
- postgis.gdal_vsi_options - Tillgänglighet: 3.2.0 En strängkonfiguration för att ställa in alternativ som används när du arbetar med ett out-db-raster.

Funktioner förbättrade i PostGIS 3.2

- GetFaceByPoint - Förbättrad: 3.2.0 effektivare implementering och tydligare kontrakt, slutar fungera med ogiltiga topologier. Hitta en yta som skär en given punkt.
- ST_ClusterKMeans - Förbättrad: 3.2.0 Stöd för max_radius Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av K-means-algoritmen.
- ST_MakeValid - Förbättrad: 3.2.0, algoritmalternativen lineworköch structure”har lagts till, vilket kräver GEOS >= 3.10.0. Försöker göra en ogiltig geometri giltig utan att förlora toppar.
- ST_PixelAsCentroid - Förbättrad: 3.2.0 Snabbare nu implementerad i C. Returnerar centroiden (punktgeometri) för det område som representeras av en pixel.

- ST_PixelAsCentroids - Förbättrad: 3.2.0 Snabbare nu implementerad i C. Returnerar centroiden (punktgeometri) för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrin är centroiden för det område som representeras av en pixel.
- ST_Point - Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin. Skapar en punkt med X-, Y- och SRID-värden.
- ST_PointM - Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin. Skapar en punkt med X-, Y-, M- och SRID-värden.
- ST_PointZ - Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin. Skapar en punkt med X-, Y-, Z- och SRID-värden.
- ST_PointZM - Förbättrad: 3.2.0 srid som ett extra valfritt argument har lagts till. Äldre installationer kräver att man kombinerar med ST_SetSRID för att markera srid på geometrin. Skapar en punkt med X-, Y-, Z-, M- och SRID-värden.
- ST_RemovePoint - Förbättrad: 3.2.0 Ta bort en punkt från en linestrings.
- ST_RemoveRepeatedPoints - Förbättrad: 3.2.0 Returnerar en version av en geometri där dubbletter av punkter har tagits bort.
- ST_StartPoint - Förbättrad: 3.2.0 returnerar en punkt för alla geometrier. Tidigare beteende returnerade NULL om indata inte var en LineString. Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_Value - Förbättrad: 3.2.0 resample valfritt argument lades till. Returnerar värdet för ett visst band i en viss kolumnx, radpixel eller vid en viss geometrisk punkt. Bandnummer börjar på 1 och antas vara 1 om det inte anges. Om exclude_nodata_value är satt till false, anses alla pixlar inklusive nodata-pixlar korsa varandra och returnerar värdet. Om värdet exclude_nodata_value inte anges läses det från rastrets metadata.
- TopoGeo_AddLineString - Förbättrad: 3.2.0 lade till stöd för att returnera signerad identifierare. Lägger till en linestrings till en befintlig topologi med hjälp av en tolerans och eventuellt delning av befintliga kanter/ytor.

Funktioner ändrade i PostGIS 3.2

- ST_Boundary - Ändrat: 3.2.0 stöd för TIN, använder inte geos, lineariserar inte kurvor Returnerar gränsen för en geometri.
- ValidateTopology - Ändrad: 3.2.0 lade till valfri bbox-parameter, utför kontroller av ytmärkning och kantlänkning. Returnerar en uppsättning validatetopology_returntype-objekt som beskriver problem med topologin.

13.12.7 PostGIS Funktioner nya eller förbättrade i 3.1

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.1

- ST_Hexagon - Tillgänglighet: 3.1.0 Returnerar en enda hexagon med den angivna kantstorleken och cellkoordinaten inom hexagonens rutnätsutrymme.
- ST_HexagonGrid - Tillgänglighet: 3.1.0 Returnerar en uppsättning hexagoner och cellindex som helt täcker gränserna för geometriargumentet.

- ST_MaximumInscribedCircle - Tillgänglighet: 3.1.0. Beräknar den största cirkeln inom en geometri.
- ST_ReducePrecision - Tillgänglighet: 3.1.0. Returnerar en giltig geometri med punkter som avrundats till en rutnätstolerans.
- ST_Square - Tillgänglighet: 3.1.0 Returnerar en enda kvadrat med hjälp av den angivna kantstorleken och cellkoordinaten inom kvadratens rutnätsutrymme.
- ST_SquareGrid - Tillgänglighet: 3.1.0 Returnerar en uppsättning rutnätsrutor och cellindex som helt täcker gränserna för geometriargumentet.

Funktioner förbättrade i PostGIS 3.1

- ST_AsEWKT - Förbättrad: 3.1.0 stöd för valfri precisionsparameter. Returnera WKT-representationen (Well-Known Text) av geometrin med SRID-metadata.
- ST_ClusterKMeans - Förbättrad: 3.1.0 Stöd för 3D-geometrier och vikter Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av K-means-algoritmen.
- ST_Difference - Förbättrad: 3.1.0 accepterar en gridSize-parameter. Beräknar en geometri som representerar den del av geometri A som inte skär geometri B.
- ST_Intersection - Förbättrad: 3.1.0 accepterar en gridSize-parameter Beräknar en geometri som representerar den delade delen av geometrierna A och B.
- ST_MakeValid - Förbättrad: 3.1.0, har lagt till borttagning av koordinater med NaN-värden. Försöker göra en ogiltig geometri giltig utan att förlora toppar.
- ST_Subdivide - Förbättrad: 3.1.0 accepterar en gridSize-parameter. Beräknar en rätlinjig subdivision av en geometri.
- ST_SymDifference - Förbättrad: 3.1.0 accepterar en gridSize-parameter. Beräknar en geometri som representerar de delar av geometrierna A och B som inte korsar varandra.
- ST_TileEnvelope - Förbättrad: 3.1.0 Lagt till marginalparameter. Skapar en rektangulär polygon i Web Mercator (SRID:3857) med hjälp av XYZ-plattsystemet.
- ST_UnaryUnion - Förbättrad: 3.1.0 accepterar en gridSize-parameter. Beräknar sammanslagningen av komponenterna i en enda geometri.
- ST_Union - Förbättrad: 3.1.0 accepterar en gridSize-parameter. Beräknar en geometri som representerar punktuppsättningssammanslagningen av indatageometrierna.

Funktioner ändrade i PostGIS 3.1

- ST_Count - Ändrad: 3.1.0 - ST_Count(rastertable, rastercolumn, ...) varianterna borttagna. Använd istället. Returnerar antalet pixlar i ett givet band i ett raster eller en rastertäckning. Om inget band anges är standardvärdet band 1. Om exclude_nodata_value är satt till true räknas endast pixlar som inte är lika med nodatavärdet.
- ST_Force3D - Ändrad: 3.1.0. Lagt till stöd för att ange ett Z-värde som inte är noll. Tvingar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ.
- ST_Force3DM - Ändrad: 3.1.0. Stöd för att ange ett M-värde som inte är noll har lagts till. Tvinga geometrierna till XYM-läge.
- ST_Force3DZ - Ändrad: 3.1.0. Lagt till stöd för att ange ett Z-värde som inte är noll. Tvinga geometrierna till XYZ-läge.
- ST_Force4D - Ändrad: 3.1.0. Stöd för att ange Z- och M-värden som inte är noll har lagts till. Tvinga geometrierna till XYZM-läge.

- ST_Histogram - Ändrad: 3.1.0 Borttagen ST_Histogram(table_name, column_name)-variant. Returnerar en uppsättning poster som sammanfattar en raster- eller rastertäckningsdatadistribution i separata bin-områden. Antalet bin beräknas automatiskt om det inte anges.
- ST_Quantile - Ändrad: 3.1.0 Borttagen ST_Quantile(table_name, column_name)-variant. Beräkna kvantiler för ett raster eller en rastertabells täckning i samband med urvalet eller populationen. Ett värde kan alltså undersökas för att ligga vid rastrets 25 %, 50 %, 75% percentil.
- ST_SummaryStats - Ändrad: 3.1.0 ST_SummaryStats(rastertable, rastercolumn, ...) varianterna har tagits bort. Använd istället. Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i ett raster eller en rastertäckning. Band 1 antas om inget band anges.

13.12.8 PostGIS Funktioner nya eller förbättrade i 3.0

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 3.0

- CG_ConstrainedDelaunayTriangles - Tillgänglighet: 3.0.0 Returnerar en begränsad Delaunay-triangulerin runt den givna indatageometrin.
- ST_3DLineInterpolatePoint - Tillgänglighet: 3.0.0 Returnerar en punkt som interpolerats längs en 3D-linje på en fraktionerad plats.
- ST_TileEnvelope - Tillgänglighet: 3.0.0 Skapar en rektangulär polygon i Web Mercator (SRID:3857) med hjälp av XYZ-plattsystemet.

Funktioner förbättrade i PostGIS 3.0

- ST_AsMVT - Förbättrad: 3.0 - stöd för Feature ID har lagts till. Aggregatfunktion som returnerar en MVT-representation av en uppsättning rader.
- ST_Contains - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om varje punkt i B ligger i A, och deras interiörer har en gemensam punkt
- ST_ContainsProperly - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om varje punkt i B ligger i det inre av A
- ST_CoveredBy - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om varje punkt i A ligger i B
- ST_Covers - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om varje punkt i B ligger i A
- ST_Crosses - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om två geometrier har vissa, men inte alla, inre punkter gemensamt
- ST_CurveToLine - Förbättrad: 3.0.0 implementerade ett minsta antal segment per linjäriserad båge för att förhindra topologisk kollaps. Konverterar en geometri som innehåller kurvor till en linjär geometri.
- ST_Disjoint - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om två geometrier inte har några gemensamma punkter
- ST_Equals - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om två geometrier innehåller samma uppsättning punkter
- ST_GeneratePoints - Förbättrad: 3.0.0, tillagd parameter för utsäde Genererar en multipoint av slumpmässiga punkter som ingår i en polygon eller multipolygon.

- ST_GeomFromGeoJSON - Enhanced: 3.0.0 parsad geometri har SRID=4326 som standard om inget annat anges. Tar som indata en geojson-representation av en geometri och matar ut ett PostGIS-geometriobjekt
- ST_LocateBetween - Förbättrad: 3.0.0 - stöd för POLYGON, TIN, TRIANGLE har lagts till. Returnerar de delar av en geometri som matchar ett mätintervall.
- ST_LocateBetweenElevations - Förbättrad: 3.0.0 - stöd för POLYGON, TIN, TRIANGLE har lagts till. Returnerar de delar av en geometri som ligger inom ett höjdintervall (Z).
- ST_Overlaps - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om två geometrier har samma dimension och skär varandra, men var och en har minst en punkt som inte finns i den andra
- ST_Relate - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om två geometrier har en topologisk relation som matchar ett Intersection Matrix-mönster, eller beräknar deras Intersection Matrix
- ST_Segmentize - Förbättrad: 3.0.0 Segmentize-geometri producerar nu lika långa undersegment Returnerar en modifierad geometri/geografi som inte har något segment som är längre än ett givet avstånd.
- ST_Touches - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om två geometrier har minst en gemensam punkt, men deras inre delar inte skär varandra
- ST_Within - Förbättrad: 3.0.0 aktiverade stöd för GEOMETRYCOLLECTION Testar om varje punkt i A ligger i B, och deras interiörer har en gemensam punkt

Funktioner ändrade i PostGIS 3.0

- PostGIS_Extensions_Upgrade - Ändrad: 3.0.0 för att packa om lösa tillägg och stödja postgis_raster. Paketerar och uppgraderar PostGIS-tillägg (t.ex. postgis_raster, postgis_topology, postgis_sfcgal) till given eller senaste version.
- ST_3DDistance - Ändrad: 3.0.0 - SFCGAL-versionen borttagen Returnerar det kartesiska 3D-minsta avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DIntersects - Ändrad: 3.0.0 SFCGAL backend borttagen, GEOS backend stöder TINs. Testar om två geometrier korsar varandra spatialt i 3D - endast för punkter, linestrings, polygoner, polyedrisk yta (area)
- ST_Area - Ändrad: 3.0.0 - är inte längre beroende av SFCGAL. Returnerar arean för en polygonal geometri.
- ST_AsGeoJSON - Ändrat: 3.0.0 stödjer poster som indata Returnerar en geometri eller funktion i GeoJSON-format.
- ST_AsGeoJSON - Ändrat: 3.0.0 mata ut SRID om inte EPSG:4326. Returnerar en geometri eller funktion i GeoJSON-format.
- ST_AsKML - Ändrad: 3.0.0 - Variantsignaturen ”versioned”har tagits bort Returnera geometrin som ett KML-element.
- ST_Distance - Ändrad: 3.0.0 - är inte längre beroende av SFCGAL. Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_Intersection - Ändrat: 3.0.0 är inte beroende av SFCGAL. Beräknar en geometri som representerar den delade delen av geometrierna A och B.
- ST_Intersects - Ändrad: 3.0.0 SFCGAL-versionen har tagits bort och inbyggt stöd för 2D TINS har lagts till. Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- ST_Union - Ändrat: 3.0.0 är inte beroende av SFCGAL. Beräknar en geometri som representerar punktuppsättningssammanslagningen av indatageometrierna.

13.12.9 PostGIS Funktioner nya eller förbättrade i 2.5

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 2.5

- PostGIS_Extensions_Upgrade - Tillgänglighet: 2.5.0 Paketerar och uppgraderar PostGIS-tillägg (t.ex. postgis_raster, postgis_topology, postgis_sfcgal) till given eller senaste version.
- ST_Angle - Tillgänglighet: 2.5.0 Returnerar vinkeln mellan två vektorer som definieras av 3 eller 4 punkter, eller 2 linjer.
- ST_AsHexWKB - Tillgänglighet: 2.5.0 Returnerar Well-Known Binary (WKB) i Hex-representation av rastret.
- ST_BandFileSize - Tillgänglighet: 2.5.0 Returnerar filstorleken för ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.
- ST_BandFileTimestamp - Tillgänglighet: 2.5.0 Returnerar filens tidsstämpel för ett band som finns lagrat i filsystemet. Om inget bandnummer anges antas 1.
- ST_ChaikinSmoothing - Tillgänglighet: 2.5.0 Returnerar en utjämnad version av en geometri med hjälp av Chaikin-algoritmen
- ST_FilterByM - Tillgänglighet: 2.5.0 Tar bort hörn baserat på deras M-värde
- ST_Grayscale - Tillgänglighet: 2.5.0 Skapar ett nytt bandraster med ett 8BUI-band från källrastret och angivna band som representerar rött, grönt och blått
- ST_LineInterpolatePoints - Tillgänglighet: 2.5.0 Returnerar punkter interpolerade längs en linje med ett fraktionerat intervall.
- ST_OrientedEnvelope - Tillgänglighet: 2.5.0. Returnerar en rektangel med minsta yta som innehåller en geometri.
- ST_QuantizeCoordinates - Tillgänglighet: 2.5.0 Sätter koordinaternas minst signifikanta bitar till noll
- ST_RastFromHexWKB - Tillgänglighet: 2.5.0 Returnera ett rastervärde från en Hex-representation av Well-Known Binary (WKB)-raster.
- ST_RastFromWKB - Tillgänglighet: 2.5.0 Returnera ett rastervärde från ett Well-Known Binary (WKB)-raster.
- ST_SetBandIndex - Tillgänglighet: 2.5.0 Uppdatera det externa bandnumret för ett out-db-band
- ST_SetBandPath - Tillgänglighet: 2.5.0 Uppdatera den externa sökvägen och bandnumret för ett out-db-band

Funktioner förbättrade i PostGIS 2.5

- ST_AsBinary/ST_AsWKB - Förbättrad: 2.5.0 Tillägg av ST_AsWKB Returnerar WKB-representationen (Well-Known Binary) av rastret.
- ST_AsMVT - Förbättrad: 2.5.0 - stöd för parallella frågor har lagts till. Aggregatfunktion som returnerar en MVT-representation av en uppsättning rader.
- ST_AsText - Förbättrad: 2.5 - valfri parameterprecision infördes. Returnera WKT-representationen (Well-Known Text) av geometrin/geografin utan SRID-metadata.
- ST_BandMetaData - Förbättrad: 2.5.0 för att inkludera outdbbandnum, filstorlek och filetimestamp för outdb-raster. Returnerar grundläggande metadata för ett specifikt rasterband. bandnummer 1 antas om det inte specificeras.

- ST_Buffer - Förbättrad: 2.5.0 - ST_Buffer geometri stöd förbättrades för att möjliggöra sidobuffring specifikation side=both|left|right. Beräknar en geometri som täcker alla punkter inom ett givet avstånd från en geometri.
- ST_GeomFromGeoJSON - Förbättrad: 2.5.0 kan nu acceptera json och jsonb som indata. Tar som indata en geojson-representation av en geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeometricMedian - Förbättrad: 2.5.0 Lagt till stöd för M som vikt för punkter. Returnerar den geometriska medianen för en MultiPoint.
- ST_Intersects - Förbättrad: 2.5.0 Stöder GEOMETRYCOLLECTION. Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- ST_OffsetCurve - Förbättrad: 2.5 - stöd för GEOMETRYCOLLECTION och MULTILINESTRING har lagts till Returnerar en förskjuten linje på ett givet avstånd och sida från en inmatad linje.
- ST_Scale - Förbättrad: 2.5.0 stöd för skalning i förhållande till ett lokalt ursprung (origin-parametern) infördes. Skalar en geometri med givna faktorer.
- ST_Split - Förbättrad: 2.5.0 stöd för att dela en polygon med en multilinje infördes. Returnerar en samling geometrier som skapats genom att dela en geometri med en annan geometri.
- ST_Subdivide - Förbättrad: 2.5.0 återanvänder befintliga punkter vid polygondelning, antalet vertex sänks från 8 till 5. Beräknar en rätlinjig subdivision av en geometri.

Funktioner ändrade i PostGIS 2.5

- ST_GDALDrivers - Ändrad: 2.5.0 - lägg till kolumnerna can_read och can_write. Returnerar en lista över rasterformat som stöds av PostGIS via GDAL. Endast de format med can_write=True kan användas av ST_AsGDALRaster

13.12.10 PostGIS Funktioner nya eller förbättrade i 2.4

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 2.4

- ST_AsGeobuf - Tillgänglighet: 2.4.0 Returnerar en Geobuf-representation av en uppsättning rader.
- ST_AsMVT - Tillgänglighet: 2.4.0 Aggregatfunktion som returnerar en MVT-representation av en uppsättning rader.
- ST_AsMVTGeom - Tillgänglighet: 2.4.0 Transformerar en geometri till koordinatrymden för en MVT-platta.
- ST_Centroid - Tillgänglighet: 2.4.0 stöd för geografi infördes. Returnerar den geometriska mittpunkten för en geometri.
- ST_ForcePolygonCCW - Tillgänglighet: 2.4.0 Orienterar alla yttre ringar moturs och alla inre ringar medurs.
- ST_ForcePolygonCW - Tillgänglighet: 2.4.0 Orienterar alla yttre ringar medurs och alla inre ringar moturs.
- ST_FrechetDistance - Tillgänglighet: 2.4.0 - kräver GEOS >= 3.7.0 Returnerar Fréchet-avståndet mellan två geometrier.
- ST_IsPolygonCCW - Tillgänglighet: 2.4.0 Testar om polygoner har yttre ringar som är orienterade moturs och inre ringar som är orienterade medurs.

- ST_IsPolygonCW - Tillgänglighet: 2.4.0 Testar om polygoner har yttre ringar som är orienterade medurs och inre ringar som är orienterade moturs.
- ST_MakeEmptyCoverage - Tillgänglighet: 2.4.0 Täck georefererat område med ett rutnät av tomma rasterplattor.

Funktioner förbättrade i PostGIS 2.4

- ST_AsTWKB - Förbättrad: 2.4.0 minnes- och hastighetsförbättringar. Returnerar geometrin som TWKB, aka Tiny Well-Known Binary”
- ST_Covers - Förbättrad: 2.4.0 Stöd för polygon i polygon och linje i polygon har lagts till för geografityp Testar om varje punkt i B ligger i A
- ST_CurveToLine - Förbättrad: I 2.4.0 tillkom stöd för toleranserna max-avvikelse och max-vinkel samt för symmetrisk utdata. Konverterar en geometri som innehåller kurvor till en linjär geometri.
- ST_Project - Förbättrad: 2.4.0 Tillåt negativt avstånd och icke-normaliserad azimut. Returnerar en punkt som projiceras från en startpunkt med ett avstånd och en bäring (azimut).
- ST_Reverse - Förbättrad: 2.4.0 stöd för kurvor infördes. Returnera geometrin med omvänd ordning på topparna.

Funktioner ändrade i PostGIS 2.4

- = - Ändrad: 2.4.0, i tidigare versioner var detta bounding box-likhet inte en geometrisk likhet. Om du behöver bounding box-likhet, använd istället. Returnerar TRUE om koordinaterna och koordinatordningen för geometri/geografi A är samma som koordinaterna och koordinatordningen för geometri/geografi B.
- ST_Node - Ändrad: 2.4.0 denna funktion använder GEOSNode internt istället för GEOSUnaryUnion. Detta kan leda till att de resulterande linestrings har en annan ordning och riktning jämfört med PostGIS < 2.4. Noder en samling av linjer.

13.12.11 PostGIS Funktioner nya eller förbättrade i 2.3

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 2.3

- &&&(geometry,gidx) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en geometris (cachade) n-D-begränsningsbox skär en n-D-begränsningsbox med flytande precision (GIDX).
- &&&(gidx,geometry) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en n-D avgränsningsbox med flytande precision (GIDX) skär en geometris (cachelagrade) n-D avgränsningsbox.
- &&&(gidx,gidx) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om två gränsboxar (GIDX) med n-D floatprecision skär varandra.
- &&(box2df,box2df) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om två 2D-begränsningsboxar med flytande precision (BOX2DF) skär varandra.
- &&(box2df,geometry) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) skär en geometris (cachelagrade) 2D-gränsbox.

- &&(geometry,box2df) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en geometris (cachelagrade) 2D-begränsningsbox skär en 2D-begränsningsbox med flytande precision (BOX2DF).
- @(box2df,box2df) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) ingår i en annan avgränsande 2D-box med flytande precision.
- @(box2df,geometry) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) ingår i en geometris 2D-gränsbox.
- @(geometry,box2df) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en geometris 2D-begränsningsbox ingår i en 2D-begränsningsbox med flytande precision (BOX2DF).
- Populate_Topology_Layer - Tillgänglighet: 2.3.0 Lägger till saknade poster i tabellen topology.layer genom att läsa metadata från topotabeller.
- ST_ClusterDBSCAN - Tillgänglighet: 2.3.0 Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av DBSCAN-algoritmen.
- ST_ClusterKMeans - Tillgänglighet: 2.3.0 Fönsterfunktion som returnerar ett kluster-ID för varje indatageometri med hjälp av K-means-algoritmen.
- ST_GeneratePoints - Tillgänglighet: 2.3.0 Genererar en multipoint av slumpmässiga punkter som ingår i en polygon eller multipolygon.
- ST_GeometricMedian - Tillgänglighet: 2.3.0 Returnerar den geometriska medianen för en MultiPoint.
- ST_MakeLine - Tillgänglighet: 2.3.0 - Stöd för MultiPoint-indataelement infördes Skapar en LineString från Point-, MultiPoint- eller LineString-geometrier.
- ST_MinimumBoundingRadius - Tillgänglighet - 2.3.0 Returnerar mittpunkten och radien för den minsta cirkeln som innehåller en geometri.
- ST_MinimumClearance - Tillgänglighet: 2.3.0 Returnerar den minsta frigången för en geometri, ett mått på geometrins robusthet.
- ST_MinimumClearanceLine - Tillgänglighet: 2.3.0 - kräver GEOS >= 3.6.0 Returnerar LineString med två punkter som sträcker sig över en geometris minsta spelrum.
- ST_Normalize - Tillgänglighet: 2.3.0 Returnera geometrin i dess kanoniska form.
- ST_Points - Tillgänglighet: 2.3.0 Returnerar en MultiPoint som innehåller koordinaterna för en geometri.
- ST_VoronoiLines - Tillgänglighet: 2.3.0 Returnerar gränserna för Voronoi-diagrammet för geometrins hörnpunkter.
- ST_VoronoiPolygons - Tillgänglighet: 2.3.0 Returnerar cellerna i Voronoi-diagrammet för hörnen i en geometri.
- ST_WrapX - Tillgänglighet: 2.3.0 kräver GEOS Omsluta en geometri runt ett X-värde.
- TopoGeom_addElement - Tillgänglighet: 2.3 Lägger till ett element till definitionen av en TopoGeometry.
- TopoGeom_remElement - Tillgänglighet: 2.3 Tar bort ett element från definitionen av en TopoGeometry.
- ~(box2df,box2df) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en avgränsande 2D-box med flytande precision (BOX2DF) innehåller en annan avgränsande 2D-box med flytande precision (BOX2DF).

- ~(box2df,geometry) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en 2D-gränsbox med flytande precision (BOX2DF) innehåller en geometris 2D-bondingbox.
- ~(geometry,box2df) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Returnerar TRUE om en geometris 2D-bindningsbox innehåller en 2D-bindningsbox med floatprecision (GIDX).

Funktioner förbättrade i PostGIS 2.3

- ST_Contains - Förbättrad: 2.3.0 Förbättring av PIP-kortslutning utökad till att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon. Testar om varje punkt i B ligger i A, och deras interiörer har en gemensam punkt
- ST_Covers - Förbättrad: 2.3.0 Förbättring av PIP-kortslutning för geometri utökad för att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon. Testar om varje punkt i B ligger i A
- ST_Expand - Förbättrad: 2.3.0 stöd har lagts till för att expandera en ruta med olika belopp i olika dimensioner. Returnerar en bounding box som expanderats från en annan bounding box eller en geometri.
- ST_Intersects - Förbättrad: 2.3.0 Förbättring av PIP-kortslutning utökad till att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon. Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- ST_Segmentize - Förbättrad: 2.3.0 Segmentize-geografin producerar nu lika långa undersegment Returnerar en modifierad geometri/geografi som inte har något segment som är längre än ett givet avstånd.
- ST_Transform - Förbättrad: 2.3.0 stöd för direkt PROJ.4-text infördes. Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem.
- ST_Within - Förbättrad: 2.3.0 Förbättring av PIP-kortslutning för geometri utökad för att stödja MultiPoints med få punkter. Tidigare versioner stödde endast punkt i polygon. Testar om varje punkt i A ligger i B, och deras interiörer har en gemensam punkt

Funktioner ändrade i PostGIS 2.3

- ST_PointN - Ändrad: 2.3.0 : negativ indexering tillgänglig (-1 är sista punkten) Returnerar den N:te punkten i den första LineString eller cirkulära LineString i en geometri.

13.12.12 PostGIS Funktioner nya eller förbättrade i 2.2

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 2.2

- <<->> - Availability: 2.2.0. Returnerar n-D-avståndet mellan geometrierna eller begränsningsrutorna A och B
- ST_AsEncodedPolyline - Tillgänglighet: 2.2.0 Returnerar en kodad polylinje från en LineString-geometri.
- ST_AsTWKB - Tillgänglighet: 2.2.0 Returnerar geometrin som TWKB, aka Tiny Well-Known Binary”
- ST_BoundingDiagonal - Tillgänglighet: 2.2.0 Returnerar diagonalen i en geometris avgränsande box.

- ST_CPAWithin - Tillgänglighet: 2.2.0 Testar om den närmaste punkten för två banor ligger inom det angivna avståndet.
- ST_ClipByBox2D - Tillgänglighet: 2.2.0 Beräknar den del av en geometri som faller inom en rektangel.
- ST_ClosestPointOfApproach - Tillgänglighet: 2.2.0 Returnerar ett mått på den närmaste närmandepunkten för två banor.
- ST_ClusterIntersecting - Tillgänglighet: 2.2.0 Aggregerad funktion som klustrar inmatade geometrier till sammanhängande mängder.
- ST_ClusterWithin - Tillgänglighet: 2.2.0 Aggregatfunktion som klustrar geometrier efter separationsavstånd.
- ST_CountAgg - Tillgänglighet: 2.2.0 Aggregera. Returnerar antalet pixlar i ett givet band i en uppsättning raster. Om inget band anges är standardvärdet band 1. Om exclude_nodata_value är satt till true räknas endast pixlar som inte är lika med NODATA-värdet.
- ST_CreateOverview - Tillgänglighet: 2.2.0 Skapa en version med reducerad upplösning av en given rastertäckning.
- ST_DistanceCPA - Tillgänglighet: 2.2.0 Returnerar avståndet mellan de två banornas närmaste närmandepunkter.
- ST_ForceCurve - Tillgänglighet: 2.2.0 Upcasta en geometri till dess kurvade typ, om tillämpligt.
- ST_IsValidTrajectory - Tillgänglighet: 2.2.0 Testar om geometrin är en giltig bana.
- ST_LineFromEncodedPolyline - Tillgänglighet: 2.2.0 Skapar en LineString från en kodad polylinje.
- ST_MapAlgebra (callback function version) - Tillgänglighet: 2.2.0: Möjlighet att lägga till en mask Callback function version - Returnerar ett enbandsraster med ett eller flera indataraster, bandindex och en användarspecificerad callback-funktion.
- ST_MemSize - Tillgänglighet: 2.2.0 Returnerar den mängd utrymme (i byte) som rastret tar upp.
- ST_RemoveRepeatedPoints - Tillgänglighet: 2.2.0 Returnerar en version av en geometri där dubbletter av punkter har tagits bort.
- ST_Retile - Tillgänglighet: 2.2.0 Returnerar en uppsättning konfigurerade plattor från en godtyckligt uppdelad rastertäckning.
- ST_SetEffectiveArea - Tillgänglighet: 2.2.0 Ställer in den effektiva ytan för varje toppunkt med hjälp av Visvalingam-Whyatt-algoritmen.
- ST_SimplifyVW - Tillgänglighet: 2.2.0 Returnerar en förenklad representation av en geometri med hjälp av Visvalingam-Whyatt-algoritmen
- ST_Subdivide - Tillgänglighet: 2.2.0 Beräknar en rätlinjig subdivision av en geometri.
- ST_SummaryStatsAgg - Tillgänglighet: 2.2.0 Aggregera. Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i en uppsättning raster. Band 1 antas om inget band anges.
- ST_SwapOrdinates - Tillgänglighet: 2.2.0 Returnerar en version av den givna geometrin med givna ordinatvärden ombytta.
- postgis.enable_outdb_rasters - Tillgänglighet: 2.2.0 Ett booleskt konfigurationsalternativ för att aktivera åtkomst till out-db-rasterband.
- postgis.gdal_datapath - Tillgänglighet: 2.2.0 Ett konfigurationsalternativ för att tilldela värdet för GDAL:s GDAL_DATA-alternativ. Om den inte är inställd används den miljöinställda variabeln GDAL_DATA.

- postgis.gdal_enabled_drivers - Tillgänglighet: 2.2.0 Ett konfigurationsalternativ för att ställa in de aktiverade GDAL-drivrutinerna i PostGIS-miljön. Påverkar GDAL-konfigurationsvariabeln GDAL_SKIP.
- |=| - Availability: 2.2.0. Returnerar avståndet mellan A- och B-banorna vid deras närmaste inflygningspunkt.

Funktioner förbättrade i PostGIS 2.2

- <-> - Enhanced: 2.2.0 -- True KNN (”K nearest neighbor”) behavior for geometry and geography. Note for geography KNN is based on sphere rather than spheroid. Returnerar 2D-avståndet mellan A och B.
- AsTopoJSON - Förbättrad: 2.2.1 har lagt till stöd för puntal-indata Returnerar TopoJSON-representationen av en topogeometri.
- ST_Area - Förbättrad: 2.2.0 - mätning på sfäroid utförd med GeographicLib för förbättrad noggrannhet och robusthet. Kräver PROJ >= 4.9.0 för att dra nytta av den nya funktionen. Returnerar arean för en polygonal geometri.
- ST_AsX3D - Förbättrad: 2.2.0: Stöd för GeoCoordinates och vändning av axlar (x/y, long/lat). Titta på alternativ för detaljer. Returnerar en geometri i X3D xml-nodelementformat: ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Azimuth - Förbättrad: 2.2.0 mätning på sfäroid utförd med GeographicLib för förbättrad noggrannhet och robusthet. Kräver PROJ >= 4.9.0 för att dra nytta av den nya funktionen. Returnerar den nordbaserade azimuten för en linje mellan två punkter.
- ST_Distance - Förbättrad: 2.2.0 - mätning på sfäroid utförd med GeographicLib för förbättrad noggrannhet och robusthet. Kräver PROJ >= 4.9.0 för att dra nytta av den nya funktionen. Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_Scale - Förbättrad: 2.2.0 stöd för skalning av alla dimensioner(faktorparameter) infördes. Skalar en geometri med givna faktorer.
- ST_Split - Förbättrad: 2.2.0 stöd för att dela en linje med en multilinje, en multipunkt eller (multi)polygongräns infördes. Returnerar en samling geometrier som skapats genom att dela en geometri med en annan geometri.
- ST_Summary - Förbättrad: 2.2.0 Lagt till stöd för TIN och kurvor Returnerar en textsammanfattning av innehållet i en geometri.

Funktioner ändrade i PostGIS 2.2

- <-> - Changed: 2.2.0 -- Old hybrid-syntax workarounds may be slower once true KNN is available. See examples below. Returnerar 2D-avståndet mellan A och B.
- ST_3DClosestPoint - Ändrad: 2.2.0 - om 2 2D-geometrier matas in returneras en 2D-punkt (istället för det gamla beteendet som antog 0 för saknad Z). Vid 2D och 3D antas Z inte längre vara 0 för saknad Z. Returnerar den 3D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste 3D-linjen.
- ST_3DDistance - Ändrad: 2.2.0 - När det gäller 2D och 3D antas Z inte längre vara 0 om Z saknas. Returnerar det kartesiska 3D-minsta avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DLongestLine - Ändrad: 2.2.0 - om 2 2D-geometrier matas in returneras en 2D-punkt (istället för det gamla beteendet som antog 0 för saknad Z). Vid 2D och 3D antas Z inte längre vara 0 för saknad Z. Returnerar den längsta 3D-linjen mellan två geometrier
- ST_3DMaxDistance - Ändrad: 2.2.0 - När det gäller 2D och 3D antas Z inte längre vara 0 om Z saknas. Returnerar det maximala kartesiska 3D-avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.

- ST_3DShortestLine - Ändrad: 2.2.0 - om 2 2D-geometrier matas in returneras en 2D-punkt (istället för det gamla beteendet som antog 0 för saknad Z). Vid 2D och 3D antas Z inte längre vara 0 för saknad Z. Returnerar den kortaste 3D-linjen mellan två geometrier
- ST_DistanceSphere - Ändrad: 2.2.0 I tidigare versioner brukade detta kallas ST_Distance_Sphere Returnerar minsta avstånd i meter mellan två lon/lat-geometrier med hjälp av en sfärisk jordmodell.
- ST_DistanceSpheroid - Ändrad: 2.2.0 I tidigare versioner kallades detta ST_Distance_Spheroid Returnerar det minsta avståndet mellan två lon/lat-geometrier med hjälp av en sfäroid jordmodell.
- ST_Equals - Ändrad: 2.2.0 Returnerar sant även för ogiltiga geometrier om de är binärt lika Testar om två geometrier innehåller samma uppsättning punkter
- ST_LengthSpheroid - Ändrad: 2.2.0 I tidigare versioner kallades detta ST_Length_Spheroid och hade aliaset ST_3DLength_Spheroid Returnerar 2D- eller 3D-längd/perimeter för en lon/lat-geometri på en sfäroid.
- ST_MemSize - Ändrad: 2.2.0 namnet ändrat till ST_MemSize för att följa namngivningskonventionen. Returnerar hur mycket minnesutrymme en geometri tar upp.
- ST_PointInsideCircle - Ändrad: 2.2.0 I tidigare versioner kallades detta ST_Point_Inside_Circle Testar om en punktgeometri ligger inom en cirkel definierad av centrum och radie
- ValidateTopology - Ändrad: 2.2.0 värdena för id1 och id2 byttes ut mot ”edge crosses nodeför att stämma överens med felbeskrivningen. Returnerar en uppsättning validatetopology_returntype-objekt som beskriver problem med topologin.

13.12.13 PostGIS Funktioner nya eller förbättrade i 2.1

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 2.1

- = - Tillgänglighet: 2.1.0 Returnerar TRUE om A:s avgränsande box är densamma som B:s. Använder avgränsningsbox med dubbel precision.
- AsTopoJSON - Tillgänglighet: 2.1.0 Returnerar TopoJSON-representationen av en topogeometri.
- ST_Box2dFromGeoHash - Tillgänglighet: 2.1.0 Returnerar en BOX2D från en GeoHash-sträng.
- ST_ColorMap - Tillgänglighet: 2.1.0 Skapar ett nytt raster med upp till fyra 8BUI-band (gråskala, RGB, RGBA) från källrastret och ett angivet band. Band 1 antas om det inte specificeras.
- ST_Contains - Tillgänglighet: 2.1.0 Returnerar true om inga punkter i raster rastB ligger i raster rastA:s exteriör och minst en punkt i rastB:s interiör ligger i rastA:s interiör.
- ST_ContainsProperly - Tillgänglighet: 2.1.0 Returnerar true om rastB skär rastA:s insida men inte rastA:s gräns eller utsida.
- ST_CoveredBy - Tillgänglighet: 2.1.0 Returnerar true om inga punkter i raster rastA ligger utanför raster rastB.
- ST_Covers - Tillgänglighet: 2.1.0 Returnerar true om inga punkter i raster rastB ligger utanför raster rastA.
- ST_DFullyWithin - Tillgänglighet: 2.1.0 Returnerar true om rasterna rastA och rastB är helt inom det angivna avståndet från varandra.
- ST_DWithin - Tillgänglighet: 2.1.0 Returnerar true om rasterna rastA och rastB ligger inom det angivna avståndet från varandra.

- ST_DelaunayTriangles - Tillgänglighet: 2.1.0 Returnerar Delaunay-trianguleringen av hörnen i en geometri.
- ST_Disjoint - Tillgänglighet: 2.1.0 Returnerar true om raster rastA inte spatialt korsar rastB.
- ST_DumpValues - Tillgänglighet: 2.1.0 Hämta värdena för det angivna bandet som en 2-dimensionell array.
- ST_ForceSFS - Tillgänglighet: 2.1.0 Force geometries to use SFS 1.1 or 1.2 geometry types.
- ST_FromGDALRaster - Tillgänglighet: 2.1.0 Returnerar ett raster från en GDAL-rasterfil som stöds.
- ST_GeomFromGeoHash - Tillgänglighet: 2.1.0 Returnerar en geometri från en GeoHash-sträng.
- ST_InvDistWeight4ma - Tillgänglighet: 2.1.0 Rasterbearbetningsfunktion som interpolerar en pixels värde från pixelns närområde.
- ST_MapAlgebra (callback function version) - Tillgänglighet: 2.1.0 Callback function version - Returnerar ett enbandsraster med ett eller flera indataraster, bandindex och en användarspecificerad callback-funktion.
- ST_MapAlgebra (expression version) - Tillgänglighet: 2.1.0 Expression version - Returnerar ett enbandsraster med ett eller två indataraster, bandindex och ett eller flera användarspecifika SQL-uttryck.
- ST_MinConvexHull - Tillgänglighet: 2.1.0 Returnerar rastrets konvexa skrovgeometri exklusive NODATA-pixlar.
- ST_MinDist4ma - Tillgänglighet: 2.1.0 Rasterbehandlingsfunktion som returnerar det minsta avståndet (i antal pixlar) mellan den intressanta pixeln och en angränsande pixel med värde.
- ST_NearestValue - Tillgänglighet: 2.1.0 Returnerar det närmaste icke-NODATA-värdet för ett givet bands pixel som anges av en kolumnx och rowy eller en geometrisk punkt uttryckt i samma spatiala referenskoordinatsystem som rastret.
- ST_Neighborhood - Tillgänglighet: 2.1.0 Returnerar en 2D-array med dubbel precision av icke-NODATA-värden runt ett visst bands pixel som anges av antingen en kolumnX och radY eller en geometrisk punkt uttryckt i samma spatiala referenskoordinatsystem som rastret.
- ST_NotSameAlignmentReason - Tillgänglighet: 2.1.0 Returnerar text som anger om rastrerna är inriktade och om de inte är inriktade, en anledning till varför.
- ST_Overlaps - Tillgänglighet: 2.1.0 Returnerar true om raster rastA och rastB korsar varandra men det ena inte helt innehåller det andra.
- ST_PixelAsCentroid - Tillgänglighet: 2.1.0 Returnerar centroiden (punktgeometri) för det område som representeras av en pixel.
- ST_PixelAsCentroids - Tillgänglighet: 2.1.0 Returnerar centroiden (punktgeometri) för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrin är centroiden för det område som representeras av en pixel.
- ST_PixelAsPoint - Tillgänglighet: 2.1.0 Returnerar en punktgeometri för pixelns övre vänstra hörn.
- ST_PixelAsPoints - Tillgänglighet: 2.1.0 Returnerar en punktgeometri för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrins koordinater är pixelns övre vänstra hörn.
- ST_PixelOfValue - Tillgänglighet: 2.1.0 Hämta koordinaterna för kolumnx, rowy för den pixel vars värde är lika med sökvärdet.
- ST_PointFromGeoHash - Tillgänglighet: 2.1.0 Returnerar en punkt från en GeoHash-sträng.
- ST_RasterToWorldCoord - Tillgänglighet: 2.1.0 Returnerar rastrets övre vänstra hörn som geometriska X och Y (longitud och latitud) givet en kolumn och rad. Kolumn och rad börjar på 1.

- ST_Resize - Tillgänglighet: 2.1.0 Kräver GDAL 1.6.1+ Ändra storlek på ett raster till en ny bredd/höjd
- ST_Roughness - Tillgänglighet: 2.1.0 Returnerar ett raster med den beräknade ojämnhetenför en DEM.
- ST_SetValues - Tillgänglighet: 2.1.0 Returnerar modifierat raster som är resultatet av att värdena för ett givet band har ställts in.
- ST_Simplify - Tillgänglighet: 2.1.0 Returnerar en förenkladgeometriversion av den angivna TopoGeometry med hjälp av Douglas-Peucker-algoritmen.
- ST_Summary - Tillgänglighet: 2.1.0 Returnerar en textsammanfattning av innehållet i rastret.
- ST_TPI - Tillgänglighet: 2.1.0 Returnerar ett raster med det beräknade topografiska positionsindexet.
- ST_TRI - Tillgänglighet: 2.1.0 Returnerar ett raster med det beräknade Terrain Ruggedness Index.
- ST_Tile - Tillgänglighet: 2.1.0 Returnerar en uppsättning raster som är resultatet av uppdelningen av inmatningsrastret baserat på de önskade dimensionerna för utdatarastren.
- ST_Touches - Tillgänglighet: 2.1.0 Returnerar true om raster rastA och rastB har minst en gemensam punkt men deras inre delar inte skär varandra.
- ST_Union - Tillgänglighet: 2.1.0 Varianten ST_Union(rast, unionarg) introducerades. Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.
- ST_Within - Tillgänglighet: 2.1.0 Returnerar true om inga punkter i raster rastA ligger i raster rastB:s exteriör och minst en punkt i rastA:s interiör ligger i rastB:s interiör.
- ST_WorldToRasterCoord - Tillgänglighet: 2.1.0 Returnerar det övre vänstra hörnet som kolumn och rad givet geometriskt X och Y (longitud och latitud) eller en punktgeometri uttryckt i rastrets spatiala referenskoordinatsystem.
- UpdateRasterSRID - Tillgänglighet: 2.1.0 Ändra SRID för alla raster i den användarspecifika kolumnen och tabellen.
- clearTopoGeom - Tillgänglighet: 2.1 Rensar innehållet i en topogeometri.
- postgis_sfcgal_version - Tillgänglighet: 2.1.0 Returnerar den version av SFCGAL som används

Funktioner förbättrade i PostGIS 2.1

- ST_AddBand - Förbättrad: 2.1.0-stöd för addbandarg har lagts till. Returnerar ett raster med det eller de nya banden av given typ som lagts till med givet initialvärde på den givna indexplatsen. Om inget index anges läggs bandet till i slutet.
- ST_AddBand - Förbättrad: 2.1.0 stöd för nya out-db band tillagt. Returnerar ett raster med det eller de nya banden av given typ som lagts till med givet initialvärde på den givna indexplatsen. Om inget index anges läggs bandet till i slutet.
- ST_AsBinary/ST_AsWKB - Förbättrad: 2.1.0 Tillägg av outasin Returnerar WKB-representationen (Well-Known Binary) av rastret.
- ST_AsGML - Förbättrad: 2.1.0 id-stöd infördes för GML 3. Returnera geometrin som ett GML-element version 2 eller 3.
- ST_Aspect - Förbättrad: 2.1.0 Använder ST_MapAlgebra() och har lagt till en valfri funktionsparameter för interpolate_nodata Returnerar aspekten (i grader som standard) för ett höjdrasterband. Användbart för analys av terräng.
- ST_Boundary - Förbättrad: 2.1.0-stöd för Triangle infördes Returnerar gränsen för en geometri.

- ST_Clip - Förbättrad: 2.1.0 Omskriven i C Returnerar det raster som klippts av indatageometrin. Om bandnummer inte anges bearbetas alla band. Om crop inte anges eller om TRUE anges, beskärs utdatarastret. Om touched är inställt på TRUE inkluderas pixlar som berörs, annars inkluderas endast pixlar vars mittpunkt ligger i geometrin.
- ST_DWithin - Förbättrad: 2.1.0 förbättrad hastighet för geografi. Se Göra geografi snabbare för mer information. Testar om två geometrier ligger inom ett givet avstånd
- ST_DWithin - Förbättrad: 2.1.0 stöd för krökta geometrier infördes. Testar om två geometrier ligger inom ett givet avstånd
- ST_Distance - Förbättrad: 2.1.0 förbättrad hastighet för geografi. Se Göra geografi snabbare för mer information. Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_Distance - Förbättrad: 2.1.0 - stöd för krökta geometrier infördes. Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_Distinct4ma - Förbättrad: 2.1.0 Tillägg av variant 2 Rasterbearbetningsfunktion som beräknar antalet unika pixelvärden i ett grannskap.
- ST_DumpPoints - Förbättrad: 2.1.0 Snabbare hastighet. Återimplementerad som native-C. Returnerar en uppsättning geometry_dump-rader för koordinaterna i en geometri.
- ST_HillShade - Förbättrad: 2.1.0 Använder ST_MapAlgebra() och har lagt till en valfri funktionsparameter för interpolate_nodata Returnerar den hypotetiska belysningen för ett höjdrasterband med hjälp av angivna indata för azimut, höjd, ljusstyrka och skala.
- ST_MakeValid - Förbättrad: 2.1.0, stöd för GEOMETRYCOLLECTION och MULTIPOINT har lagts till. Försöker göra en ogiltig geometri giltig utan att förlora toppar.
- ST_Max4ma - Förbättrad: 2.1.0 Tillägg av variant 2 Rasterbearbetningsfunktion som beräknar det maximala pixelvärdet i ett grannskap.
- ST_Mean4ma - Förbättrad: 2.1.0 Tillägg av variant 2 Rasterbearbetningsfunktion som beräknar det genomsnittliga pixelvärdet i ett grannskap.
- ST_Min4ma - Förbättrad: 2.1.0 Tillägg av variant 2 Rasterbearbetningsfunktion som beräknar det lägsta pixelvärdet i ett grannskap.
- ST_PixelAsPolygons - Förbättrad: 2.1.0 exclude_nodata_value valfritt argument lades till. Returnerar den polygongeometri som avgränsar varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel.
- ST_Polygon - Förbättrad: 2.1.0 Förbättrad hastighet (helt C-baserad) och den returnerande multipolygonen är säkerställd att vara giltig. Returnerar en multipolygongeometri som bildas av sammanslagningen av pixlar som har ett pixelvärde som inte är något datavärde. Om inget bandnummer anges är standardvärdet för bandnum 1.
- ST_Range4ma - Förbättrad: 2.1.0 Tillägg av variant 2 Rasterbearbetningsfunktion som beräknar intervallet för pixelvärden i ett område.
- ST_SameAlignment - Förbättrad: 2.1.0 tillägg av Aggregate-variant Returnerar true om raster har samma skevhet, skala, spatiala ref och offset (pixlar kan placeras i samma rutnät utan att skära i pixlar) och false om de inte har det med ett meddelande om detaljproblem.
- ST_Segmentize - Förbättrad: 2.1.0 stöd för geografi infördes. Returnerar en modifierad geometri/geografi som inte har något segment som är längre än ett givet avstånd.
- ST_SetGeoReference - Förbättrad: 2.1.0 Tillägg av varianten ST_SetGeoReference(raster, dubbel precision, ...) Set Georeference 6 georeferensparametrar i ett enda anrop. Siffrorna ska separeras med vitt utrymme. Accepterar inmatningar i GDAL- eller ESRI-format. Standard är GDAL.

- ST_SetValue - Förbättrad: 2.1.0 Geometrivarianten av ST_SetValue() stöder nu alla geometrityper, inte bara punkt. Geometrivarianten är ett omslag runt geomval[]-varianten av ST_SetValues() Returnerar modifierad raster som är resultatet av att värdet för ett givet band har ställts in i en given kolumnx, radpixel eller de pixlar som skär en viss geometri. Bandnummer börjar på 1 och antas vara 1 om de inte specificeras.
- ST_Slope - Förbättrad: 2.1.0 Använder ST_MapAlgebra() och har lagt till valfria funktionsparametrar för units, scale, interpolate_nodata Returnerar lutningen (i grader som standard) för ett höjdrasterband. Användbart för att analysera terräng.
- ST_StdDev4ma - Förbättrad: 2.1.0 Tillägg av variant 2 Rasterbearbetningsfunktion som beräknar standardavvikelsen för pixelvärden i ett grannskap.
- ST_Sum4ma - Förbättrad: 2.1.0 Tillägg av variant 2 Rasterbearbetningsfunktion som beräknar summan av alla pixelvärden i ett grannskap.
- ST_Summary - Utökad: 2.1.0 S-flagga för att ange om det finns ett känt spatialt referenssystem Returnerar en textsammanfattning av innehållet i en geometri.
- ST_Transform - Förbättrad: 2.1.0 Tillägg av ST_Transform(rast, alignto)-variant Återprojicerar ett raster i ett känt spatialt referenssystem till ett annat känt spatialt referenssystem med hjälp av en angiven omsamplingsalgoritm. Alternativen är NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos med NearestNeighbor som standard.
- ST_Union - Förbättrad: 2.1.0 Förbättrad hastighet (helt C-baserad). Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.
- ST_Union - Förbättrad: 2.1.0 ST_Union(rast) (variant 1) förenar alla band i alla ingående raster. Tidigare versioner av PostGIS antog det första bandet. Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.
- ST_Union - Förbättrad: 2.1.0 ST_Union(rast, uniontype) (variant 4) förenar alla band i alla ingående raster. Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.
- toTopoGeom - Förbättrad: 2.1.0 lägger till versionen som tar en befintlig TopoGeometry. Konverterar en enkel geometri till en topogeometri.

Funktioner ändrade i PostGIS 2.1

- ST_Aspect - Ändrad: 2.1.0 I tidigare versioner var returvärdena i radianer. Nu är returvärdena som standard grader Returnerar aspekten (i grader som standard) för ett höjdrasterband. Användbart för analys av terräng.
- ST_EstimatedExtent - Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Estimated_Extent. Returnerar den uppskattade omfattningen av en spatial tabell.
- ST_Force2D - Ändrad: 2.1.0. Fram till 2.0.x kallades detta för ST_Force_2D. Tvinga geometrierna till ett ”2-dimensionellt läge”.
- ST_Force3D - Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Force_3D. Tvingar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ.
- ST_Force3DM - Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Force_3DM. Tvinga geometrierna till XYM-läge.
- ST_Force3DZ - Ändrad: 2.1.0. Fram till 2.0.x kallades detta för ST_Force_3DZ. Tvinga geometrierna till XYZ-läge.
- ST_Force4D - Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Force_4D. Tvinga geometrierna till XYZM-läge.

- ST_ForceCollection - Ändrad: 2.1.0. Fram till 2.0.x hette detta ST_Force_Collection. Konvertera geometrin till en GEOMETRYCOLLECTION.
- ST_HillShade - Ändrad: 2.1.0 I tidigare versioner uttrycktes azimut och höjd i radianer. Nu uttrycks azimut och höjd i grader Returnerar den hypotetiska belysningen för ett höjdrasterband med hjälp av angivna indata för azimut, höjd, ljusstyrka och skala.
- ST_LineInterpolatePoint - Ändrad: 2.1.0. Fram till 2.0.x hette detta ST_Line_Interpolate_Point. Returnerar en punkt som interpolerats längs en linje på en fraktionerad plats.
- ST_LineLocatePoint - Ändrad: 2.1.0. Fram till 2.0.x hette detta ST_Line_Locate_Point. Returnerar den fraktionerade positionen för den punkt på en linje som ligger närmast en punkt.
- ST_LineSubstring - Ändrad: 2.1.0. Fram till 2.0.x kallades detta ST_Line_Substring. Returnerar delen av en linje mellan två fraktionerade platser.
- ST_PixelAsCentroids - Ändrad: 2.1.1 Ändrat beteende för exclude_nodata_value. Returnerar centroiden (punktgeometri) för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrin är centroiden för det område som representeras av en pixel.
- ST_PixelAsPoints - Ändrad: 2.1.1 Ändrat beteende för exclude_nodata_value. Returnerar en punktgeometri för varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel. Punktgeometrins koordinater är pixelns övre vänstra hörn.
- ST_PixelAsPolygons - Ändrad: 2.1.1 Ändrat beteende för exclude_nodata_value. Returnerar den polygongeometri som avgränsar varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel.
- ST_Polygon - Ändrad: 2.1.0 I tidigare versioner returnerades ibland en polygon, ändrat till att alltid returnera multipolygon. Returnerar en multipolygongeometri som bildas av sammanslagningen av pixlar som har ett pixelvärde som inte är något datavärde. Om inget bandnummer anges är standardvärdet för bandnum 1.
- ST_RasterToWorldCoordX - Ändrad: 2.1.0 I tidigare versioner kallades detta ST_Raster2WorldCoordX Returnerar den geometriska X-koordinaten uppe till vänster i ett raster, en kolumn och en rad. Numreringen av kolumner och rader börjar på 1.
- ST_RasterToWorldCoordY - Ändrad: 2.1.0 I tidigare versioner kallades detta ST_Raster2WorldCoordY Returnerar den geometriska Y-koordinaten i övre vänstra hörnet av ett raster, en kolumn och en rad. Numreringen av kolumner och rader börjar på 1.
- ST_Rescale - Ändrad: 2.1.0 Fungerar på raster utan SRID Resampla ett raster genom att endast justera dess skala (eller pixelstorlek). Nya pixelvärden beräknas med hjälp av omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline, Lanczos, Max eller Min. Standard är NearestNeighbor.
- ST_Reskew - Ändrad: 2.1.0 Fungerar på raster utan SRID Resampla ett raster genom att endast justera dess skevhet (eller rotationsparametrar). Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är NearestNeighbor.
- ST_Segmentize - Ändrad: 2.1.0 Som ett resultat av införandet av geografistöd orsakar användningen ST_Segmentize('LINESTRING(1 2, 3 4)', 0.5) ett tvetydigt funktionsfel. Indata måste vara korrekt typad som en geometri eller geografi. Använd ST_GeomFromText, ST_GeogFromText eller en cast till den nödvändiga typen (t.ex. ST_Segmentize('LINESTRING(1 2, 3 4)'::geometry, 0.5)) Returnerar en modifierad geometri/geografi som inte har något segment som är längre än ett givet avstånd.
- ST_Slope - Ändrad: 2.1.0 I tidigare versioner var returvärdena i radianer. Nu är returvärdena som standard grader Returnerar lutningen (i grader som standard) för ett höjdrasterband. Användbart för att analysera terräng.

- ST_SnapToGrid - Ändrad: 2.1.0 Fungerar på raster utan SRID Sampla om ett raster genom att fästa det i ett rutnät. Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är Nearest-Neighbor.
- ST_WorldToRasterCoordX - Ändrad: 2.1.0 I tidigare versioner kallades detta ST_World2RasterCoordX Returnerar kolumnen i rastret för punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw) som representeras i rastrets världsspatiala referenssystem.
- ST_WorldToRasterCoordY - Ändrad: 2.1.0 I tidigare versioner kallades detta ST_World2RasterCoordY Returnerar raden i rastret för punktgeometrin (pt) eller en X- och Y-världskoordinat (xw, yw) som representeras i rastrets världsspatiala referenssystem.

13.12.14 PostGIS Funktioner nya eller förbättrade i 2.0

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 2.0

- && - Tillgänglighet: 2.0.0 Returnerar TRUE om A:s avgränsande box skär B:s avgränsande box.
- &&& - Tillgänglighet: 2.0.0 Returnerar TRUE om A:s n-D-gränsbox skär B:s n-D-gränsbox.
- <#> - Availability: 2.0.0. Returnerar 2D-avståndet mellan A- och B-begränsningsrutorna.
- <-> - Availability: 2.0.0 -- Weak KNN provides nearest neighbors based on geometry centroid distances instead of true distances. Exact results for points, inexact for all other types. Returnerar 2D-avståndet mellan A och B.
- @ - Tillgänglighet: 2.0.0 raster @ raster, raster @ geometri infördes Returnerar TRUE om A:s avgränsande box är innesluten i B:s. Använder avgränsande box med dubbel precision.
- @ - Tillgänglighet: 2.0.5 geometry @ raster infördes Returnerar TRUE om A:s avgränsande box är innesluten i B:s. Använder avgränsande box med dubbel precision.
- AddEdge - Tillgänglighet: 2.0.0 Lägger till en linestringskant i tabellen edge och associerade start- och slutpunkter i tabellen point nodes i det angivna topologischemat med hjälp av den angivna linestringsgeometrin och returnerar edgeid för den nya (eller befintliga) kanten.
- AddFace - Tillgänglighet: 2.0.0 Registrerar en ytprimitiv till en topologi och hämtar dess identifierare.
- AddNode - Tillgänglighet: 2.0.0 Lägger till en punktnod i nodtabellen i det angivna topologischemat och returnerar den nya nodens nodid. Om punkten redan finns som nod returneras det befintliga nodid.
- AddOverviewConstraints - Tillgänglighet: 2.0.0 Märk en rasterkolumn som en översikt av en annan.
- AddRasterConstraints - Tillgänglighet: 2.0.0 Lägger till rasterbegränsningar till en laddad rastertabell för en specifik kolumn som begränsar spatial ref, skalning, blockstorlek, inriktning, band, bandtyp och en flagga för att ange om rasterkolumnen blockeras regelbundet. Tabellen måste vara laddad med data för att begränsningarna ska kunna härledas. Returnerar true om begränsningsinställningen har utförts och utfärdar ett meddelande i annat fall.
- AsGML - Tillgänglighet: 2.0.0 Returnerar GML-representationen av en topogeometri.
- CopyTopology - Tillgänglighet: 2.0.0 Gör en kopia av en topologi (noder, kanter, ytor, lager och TopoGeometries) till ett nytt schema
- DropOverviewConstraints - Tillgänglighet: 2.0.0 Avmarkera en rasterkolumn från att vara en översikt av en annan.

- DropRasterConstraints - Tillgänglighet: 2.0.0 Tar bort PostGIS-rasterbegränsningar som hänvisar till en rastertabellkolumn. Användbart om du behöver ladda om data eller uppdatera dina rasterkolumndata.
- GetEdgeByPoint - Tillgänglighet: 2.0.0 Hittar kant-id för en kant som skär en given punkt.
- GetFaceByPoint - Tillgänglighet: 2.0.0 Hitta en yta som skär en given punkt.
- GetNodeByPoint - Tillgänglighet: 2.0.0 Hittar nod-id för en nod vid en punktposition.
- GetNodeEdges - Tillgänglighet: 2.0 Returnerar en ordnad uppsättning kanter som är kopplade till den angivna noden.
- GetRingEdges - Tillgänglighet: 2.0.0 Returnerar den ordnade uppsättningen av signerade kantidentifierare som man möter när man går på en given kantsida.
- GetTopoGeomElements - Tillgänglighet: 2.0.0 Returnerar en uppsättning topoelement-objekt som innehåller topologiska element_id,element_type för den givna TopoGeometry (primitiva element).
- GetTopologySRID - Tillgänglighet: 2.0.0 Returnerar SRID för en topologi i tabellen topology.topology med topologins namn.
- Polygonize - Tillgänglighet: 2.0.0 Hittar och registrerar alla ytor som definieras av topologikanter.
- ST_3DClosestPoint - Tillgänglighet: 2.0.0 Returnerar den 3D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste 3D-linjen.
- ST_3DDFullyWithin - Tillgänglighet: 2.0.0 Testar om två 3D-geometrier är helt inom ett givet 3D-avstånd
- ST_3DDWithin - Tillgänglighet: 2.0.0 Testar om två 3D-geometrier befinner sig inom ett givet 3D-avstånd
- ST_3DDistance - Tillgänglighet: 2.0.0 Returnerar det kartesiska 3D-minsta avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DIntersects - Tillgänglighet: 2.0.0 Testar om två geometrier korsar varandra spatialt i 3D - endast för punkter, linestrings, polygoner, polyedrisk yta (area)
- ST_3DLongestLine - Tillgänglighet: 2.0.0 Returnerar den längsta 3D-linjen mellan två geometrier
- ST_3DMaxDistance - Tillgänglighet: 2.0.0 Returnerar det maximala kartesiska 3D-avståndet (baserat på spatial ref) mellan två geometrier i projicerade enheter.
- ST_3DShortestLine - Tillgänglighet: 2.0.0 Returnerar den kortaste 3D-linjen mellan två geometrier
- ST_AddEdgeModFace - Tillgänglighet: 2.0 Lägg till en ny kant och, om den delar en yta, modifiera den ursprungliga ytan och lägg till en ny yta.
- ST_AddEdgeNewFaces - Tillgänglighet: 2.0 Lägg till en ny kant och, om den delar en yta, ta bort den ursprungliga ytan och ersätt den med två nya ytor.
- ST_AsGDALRaster - Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0. Returnerar rasterplattan i det angivna GDAL Raster-formatet. Rasterformat är ett av de format som stöds av ditt kompilerade bibliotek. Använd ST_GDALDrivers() för att få en lista över de format som stöds av ditt bibliotek.
- ST_AsJPEG - Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0. Returnerar de valda banden i rasterplattan som en enda JPEG-bild (byte-array) (Joint Photographic Exports Group). Om inget band anges och 1 eller fler än 3 band, används endast det första bandet. Om endast 3 band anges används alla 3 banden och mappas till RGB.
- ST_AsLatLonText - Tillgänglighet: 2.0 Returnerar grader, minuter och sekunder för den angivna punkten.

- ST_AsPNG - Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0. Returnerar de valda banden i rasterkaklet som en enda PNG-bild (portable network graphics) (byte-array). Om 1, 3 eller 4 band i rastret och inga band anges, används alla band. Om fler än 2 eller fler än 4 band och inga band anges, används endast band 1. Banden mappas till RGB- eller RGBA-rymd.
- ST_AsRaster - Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0. Konverterar en PostGIS-geometri till ett PostGIS-raster.
- ST_AsTIFF - Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0. Returnerar de band som valts i rastret som en enda TIFF-bild (byte-array). Om inget band anges eller om något av de angivna banden inte finns i rastret, försöker alla band användas.
- ST_AsX3D - Tillgänglighet: 2.0.0: ISO-IEC-19776-1.2-X3DEncodings-XML Returnerar en geometri i X3D xml-nodelementformat: ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Aspect - Tillgänglighet: 2.0.0 Returnerar aspekten (i grader som standard) för ett höjdrasterband. Användbart för analys av terräng.
- ST_Band - Tillgänglighet: 2.0.0 Returnerar ett eller flera band från ett befintligt raster som ett nytt raster. Användbart för att bygga nya raster från befintliga raster.
- ST_BandIsNoData - Tillgänglighet: 2.0.0 Returnerar true om bandet är fyllt med endast nodatavärden.
- ST_Clip - Tillgänglighet: 2.0.0 Returnerar det raster som klippts av indatageometrin. Om bandnummer inte anges bearbetas alla band. Om crop inte anges eller om TRUE anges, beskärs utdatarastret. Om touched är inställt på TRUE inkluderas pixlar som berörs, annars inkluderas endast pixlar vars mittpunkt ligger i geometrin.
- ST_CollectionHomogenize - Tillgänglighet: 2.0.0 Returnerar den enklaste representationen av en geometrisamling.
- ST_ConcaveHull - Tillgänglighet: 2.0.0 Beräknar en eventuellt konkav geometri som innehåller alla indatageometrins toppar
- ST_Count - Tillgänglighet: 2.0.0 Returnerar antalet pixlar i ett givet band i ett raster eller en rastertäckning. Om inget band anges är standardvärdet band 1. Om exclude_nodata_value är satt till true räknas endast pixlar som inte är lika med nodatavärdet.
- ST_CreateTopoGeo - Tillgänglighet: 2.0 Lägger till en samling geometrier till en given tom topologi och returnerar ett meddelande om det lyckas.
- ST_Distinct4ma - Tillgänglighet: 2.0.0 Rasterbearbetningsfunktion som beräknar antalet unika pixelvärden i ett grannskap.
- ST_FlipCoordinates - Tillgänglighet: 2.0.0 Returnerar en version av en geometri med X- och Y-axlarna vända.
- ST_GDALDrivers - Tillgänglighet: 2.0.0 - kräver GDAL >= 1.6.0. Returnerar en lista över rasterformat som stöds av PostGIS via GDAL. Endast de format med can_write=True kan användas av ST_AsGDALRaster
- ST_GeomFromGeoJSON - Tillgänglighet: 2.0.0 kräver - JSON-C >= 0.9 Tar som indata en geojson-representation av en geometri och matar ut ett PostGIS-geometriobjekt
- ST_GetFaceEdges - Tillgänglighet: 2.0 Returnerar en uppsättning ordnade kanter som avgränsar en yta..
- ST_HasNoBand - Tillgänglighet: 2.0.0 Returnerar true om det inte finns något band med angivet bandnummer. Om inget bandnummer anges antas bandnummer 1.
- ST_HillShade - Tillgänglighet: 2.0.0 Returnerar den hypotetiska belysningen för ett höjdrasterband med hjälp av angivna indata för azimut, höjd, ljusstyrka och skala.

- ST_Histogram - Tillgänglighet: 2.0.0 Returnerar en uppsättning poster som sammanfattar en raster- eller rastertäckningsdatadistribution i separata bin-områden. Antalet bin beräknas automatiskt om det inte anges.
- ST_InterpolatePoint - Tillgänglighet: 2.0.0 Returnerar det interpolerade måttet för en geometri som ligger närmast en punkt.
- ST_IsEmpty - Tillgänglighet: 2.0.0 Returnerar true om rastret är tomt (bredd = 0 och höjd = 0). I annat fall returneras false.
- ST_IsValidDetail - Tillgänglighet: 2.0.0 Returnerar en valid_detail-rad som anger om en geometri är giltig eller om den inte är det, en orsak och en plats.
- ST_IsValidReason - Tillgänglighet: 2.0-versionen tar flaggor. Returnerar text som anger om en geometri är giltig, eller en orsak till ogiltigheten.
- ST_MakeLine - Tillgänglighet: 2.0.0 - Stöd för LineString-indataelement infördes Skapar en LineString från Point-, MultiPoint- eller LineString-geometrier.
- ST_MakeValid - Tillgänglighet: 2.0.0 Försöker göra en ogiltig geometri giltig utan att förlora toppar.
- ST_MapAlgebraExpr - Tillgänglighet: 2.0.0 1 raster band version: Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på inmatningsrasterbandet och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges.
- ST_MapAlgebraExpr - Tillgänglighet: 2.0.0 2 rasterbandversion: Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-algebraisk operation på de två inmatningsrasterbanden och av pixeltyp som tillhandahålls. band 1 för varje raster antas om inga bandnummer anges. Den resulterande rastern kommer att justeras (skala, skevhet och pixelhörn) på det rutnät som definieras av den första rastern och ha sin utsträckning definierad av parametern ”extenttype”. Värden för ”extenttype”kan vara: INTERSECTION, UNION, FIRST, SECOND.
- ST_MapAlgebraFct - Tillgänglighet: 2.0.0 1 bandversion - Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion på inmatningsrasterbandet och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges.
- ST_MapAlgebraFct - Tillgänglighet: 2.0.0 2 band version - Skapar en ny enbandsraster som bildas genom att tillämpa en giltig PostgreSQL-funktion på de 2 inmatningsrasterbanden och av pixeltyp som tillhandahålls. Band 1 antas om inget band anges. Utsträckningstyp är som standard INTERSECTION om den inte anges.
- ST_MapAlgebraFctNgb - Tillgänglighet: 2.0.0 1-bandsversion: Kartlägg Algebra närmaste granne med hjälp av användardefinierad PostgreSQL-funktion. Returnera en raster vars värden är resultatet av en PLPGSQL-användarfunktion som involverar ett grannskap av värden från inmatningsrasterbandet.
- ST_Max4ma - Tillgänglighet: 2.0.0 Rasterbearbetningsfunktion som beräknar det maximala pixelvärdet i ett grannskap.
- ST_Mean4ma - Tillgänglighet: 2.0.0 Rasterbearbetningsfunktion som beräknar det genomsnittliga pixelvärdet i ett grannskap.
- ST_Min4ma - Tillgänglighet: 2.0.0 Rasterbearbetningsfunktion som beräknar det lägsta pixelvärdet i ett grannskap.
- ST_ModEdgeHeal - Tillgänglighet: 2.0 Läker två kanter genom att ta bort den nod som förbinder dem, modifiera den första kanten och ta bort den andra kanten. Returnerar id för den borttagna noden.
- ST_MoveIsoNode - Tillgänglighet: 2.0.0 Flyttar en isolerad nod i en topologi från en punkt till en annan. Om den nya apoint-geometrin existerar som en nod kastas ett fel. Returnerar beskrivning av förflyttning.

- ST_NewEdgeHeal - Tillgänglighet: 2.0 Läker två kanter genom att ta bort noden som förbinder dem, ta bort båda kanterna och ersätta dem med en kant vars riktning är densamma som den första tillhandahållna kanten.
- ST_Node - Tillgänglighet: 2.0.0 Noder en samling av linjer.
- ST_NumPatches - Tillgänglighet: 2.0.0 Returns the number of faces in a PolyhedralSurface or TIN.
- ST_OffsetCurve - Tillgänglighet: 2.0 Returnerar en förskjuten linje på ett givet avstånd och sida från en inmatad linje.
- ST_PatchN - Tillgänglighet: 2.0.0 Returns the Nth face of a PolyhedralSurface or TIN.
- ST_Perimeter - Tillgänglighet 2.0.0: Stöd för geografi infördes Returnerar längden på gränsen för en polygonal geometri eller geografi.
- ST_PixelAsPolygon - Tillgänglighet: 2.0.0 Returnerar den polygongeometri som avgränsar pixeln för en viss rad och kolumn.
- ST_PixelAsPolygons - Tillgänglighet: 2.0.0 Returnerar den polygongeometri som avgränsar varje pixel i ett rasterband tillsammans med värdet, X- och Y-rasterkoordinaterna för varje pixel.
- ST_Project - Tillgänglighet: 2.0.0 Returnerar en punkt som projiceras från en startpunkt med ett avstånd och en bäring (azimut).
- ST_Quantile - Tillgänglighet: 2.0.0 Beräkna kvantiler för ett raster eller en rastertabells täckning i samband med urvalet eller populationen. Ett värde kan alltså undersökas för att ligga vid rastrets 25 %, 50 %, 75% percentil.
- ST_Range4ma - Tillgänglighet: 2.0.0 Rasterbearbetningsfunktion som beräknar intervallet för pixelvärden i ett område.
- ST_Reclass - Tillgänglighet: 2.0.0 Skapar ett nytt raster som består av bandtyper som omklassificerats från originalet. Nband är det band som ska ändras. Om nband inte anges antas värdet vara 1. Alla andra band returneras oförändrade. Användningsfall: konvertera ett 16BUI-band till ett 8BUI och så vidare för enklare rendering som visningsbara format.
- ST_RelateMatch - Tillgänglighet: 2.0.0 Testar om en DE-9IM Intersection Matrix matchar ett Intersection Matrix-mönster
- ST_RemEdgeModFace - Tillgänglighet: 2.0 Tar bort en kant, och om kanten separerar två ytor tas den ena ytan bort och den andra ytan modifieras så att den täcker utrymmet för båda ytorna.
- ST_RemEdgeNewFace - Tillgänglighet: 2.0 Tar bort en kant och, om den borttagna kanten separerade två ytor, tar bort de ursprungliga ytorna och ersätter dem med en ny yta.
- ST_Resample - Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+ Resampla ett raster med hjälp av en specificerad resamplingsalgoritm, nya dimensioner, ett godtyckligt rutnätshörn och en uppsättning rastergeoreferensattribut som definierats eller lånats från ett annat raster.
- ST_Rescale - Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+ Resampla ett raster genom att endast justera dess skala (eller pixelstorlek). Nya pixelvärden beräknas med hjälp av omsamplingsalgoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline, Lanczos, Max eller Min. Standard är NearestNeighbor.
- ST_Reskew - Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+ Resampla ett raster genom att endast justera dess skevhet (eller rotationsparametrar). Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är NearestNeighbor.
- ST_SameAlignment - Tillgänglighet: 2.0.0 Returnerar true om raster har samma skevhet, skala, spatiala ref och offset (pixlar kan placeras i samma rutnät utan att skära i pixlar) och false om de inte har det med ett meddelande om detaljproblem.

- ST_SetBandIsNoData - Tillgänglighet: 2.0.0 Ställer in bandets isnodata-flagga till TRUE.
- ST_SharedPaths - Tillgänglighet: 2.0.0 Returnerar en samling som innehåller sökvägar som delas av de två inmatade linestrings/multilinestrings.
- ST_Slope - Tillgänglighet: 2.0.0 Returnerar lutningen (i grader som standard) för ett höjdrasterband. Användbart för att analysera terräng.
- ST_Snap - Tillgänglighet: 2.0.0 Fäst segment och vertikaler i indatageometrin till vertikaler i en referensgeometri.
- ST_SnapToGrid - Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+ Sampla om ett raster genom att fästa det i ett rutnät. Nya pixelvärden beräknas med hjälp av algoritmen NearestNeighbor (engelsk eller amerikansk stavning), Bilinear, Cubic, CubicSpline eller Lanczos resampling. Standard är Nearest-Neighbor.
- ST_Split - Tillgänglighet: 2.0.0 kräver GEOS Returnerar en samling geometrier som skapats genom att dela en geometri med en annan geometri.
- ST_StdDev4ma - Tillgänglighet: 2.0.0 Rasterbearbetningsfunktion som beräknar standardavvikelsen för pixelvärden i ett grannskap.
- ST_Sum4ma - Tillgänglighet: 2.0.0 Rasterbearbetningsfunktion som beräknar summan av alla pixelvärden i ett grannskap.
- ST_SummaryStats - Tillgänglighet: 2.0.0 Returnerar summarystats bestående av count, sum, mean, stddev, min, max för ett givet rasterband i ett raster eller en rastertäckning. Band 1 antas om inget band anges.
- ST_Transform - Tillgänglighet: 2.0.0 Kräver GDAL 1.6.1+ Återprojicerar ett raster i ett känt spatialt referenssystem till ett annat känt spatialt referenssystem med hjälp av en angiven omsamplingsalgoritm. Alternativen är NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos med NearestNeighbor som standard.
- ST_UnaryUnion - Tillgänglighet: 2.0.0 Beräknar sammanslagningen av komponenterna i en enda geometri.
- ST_Union - Tillgänglighet: 2.0.0 Returnerar sammanslagningen av en uppsättning rasterplattor till ett enda raster som består av 1 eller flera band.
- ST_ValueCount - Tillgänglighet: 2.0.0 Returnerar en uppsättning poster som innehåller ett pixelbandvärde och en räkning av antalet pixlar i ett givet band i ett raster (eller en rastertäckning) som har en given uppsättning värden. Om inget band anges är standardvärdet band 1. Som standard räknas inte pixlar med nodatavärden. och alla andra värden i pixeln matas ut och pixelbandvärden avrundas till närmaste heltal.
- TopoElementArray_Agg - Tillgänglighet: 2.0.0 Returnerar en topoelementarray för en uppsättning element_id, typ arrayer (topoelements).
- TopoGeo_AddLineString - Tillgänglighet: 2.0.0 Lägger till en linestrings till en befintlig topologi med hjälp av en tolerans och eventuellt delning av befintliga kanter/ytor.
- TopoGeo_AddPoint - Tillgänglighet: 2.0.0 Lägger till en punkt i en befintlig topologi med hjälp av en tolerans och eventuellt genom att dela en befintlig kant.
- TopoGeo_AddPolygon - Tillgänglighet: 2.0.0 Lägger till en polygon till en befintlig topologi med hjälp av en tolerans och eventuellt delning av befintliga kanter/ytor. Returnerar ytidentifierare.
- TopologySummary - Tillgänglighet: 2.0.0 Tar ett topologinamn och ger sammanfattande totalsummor för olika typer av objekt i topologin.
- toTopoGeom - Tillgänglighet: 2.0 Konverterar en enkel geometri till en topogeometri.

- ~ - Tillgänglighet: 2.0.0 Returnerar TRUE om A:s avgränsande box innehåller B:s. Använder avgränsande box med dubbel precision.
- ~= - Tillgänglighet: 2.0.0 Returnerar TRUE om A:s avgränsande box är densamma som B:s.

Funktioner förbättrade i PostGIS 2.0

- && - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Returnerar TRUE om A:s 2D-begränsningsbox skär B:s 2D-begränsningsbox.
- AddGeometryColumn - Förbättrad: 2.0.0 use_typmod-argumentet infördes. Standard är att skapa typmod-geometrikolumn istället för begränsningsbaserad. Lägger till en geometrikolumn i en befintlig tabell.
- Box2D - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar en BOX2D som representerar 2D-utbredningen av en geometri.
- Box3D - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar en BOX3D som representerar 3D-utbredningen av en geometri.
- CreateTopology - Enhanced: 2.0 added support for hasZ. Skapar ett nytt topologischema och registrerar det i tabellen topology.topology.
- GeometryType - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar typen av geometri som text.
- Populate_Geometry_Columns - Förbättrad: 2.0.0 use_typmod valfritt argument introducerades som gör det möjligt att kontrollera om kolumner skapas med typmodifierare eller med kontrollbegränsningar. Säkerställer att geometrikolumner definieras med typmodifierare eller har lämpliga spatiala begränsningar.
- ST_3DExtent - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.
- ST_Affine - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Tillämpa en 3D-affin transformation på en geometri.
- ST_Area - Förbättrad: 2.0.0 - stöd för 2D polyhedrala ytor infördes. Returnerar arean för en polygonal geometri.
- ST_AsBinary - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsBinary - Förbättrad: 2.0.0 stöd för högre koordinatdimensioner infördes. Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsBinary - Förbättrad: 2.0.0 stöd för att ange endian med geografi infördes. Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsEWKB - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar EWKB-representationen (Extended Well-Known Binary) av geometrin med SRID-metadata.
- ST_AsEWKT - Förbättrad: 2.0.0 stöd för Geography, Polyhedral surfaces, Triangles och TIN infördes. Returnera WKT-representationen (Well-Known Text) av geometrin med SRID-metadata.
- ST_AsGML - Förbättrad: Stöd för prefix 2.0.0 har införts. Alternativ 4 för GML3 infördes för att tillåta användning av LineString istället för Curve-tagg för linjer. GML3-stöd för polyedriska ytor och TINS infördes. Alternativ 32 infördes för att mata ut boxen. Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsKML - Förbättrad: 2.0.0 - Lägg till prefixnamnrymd, använd standard- och namngivna args Returnera geometrin som ett KML-element.

- ST_Azimuth - Förbättrad: 2.0.0 stöd för geografi infördes. Returnerar den nordbaserade azimuten för en linje mellan två punkter.
- ST_Dimension - Förbättrad: 2.0.0 stöd för polyhedrala ytor och TINs infördes. Kastar inte längre ett undantag om en tom geometri ges. Returnerar den topologiska dimensionen för en geometri.
- ST_Dump - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar en uppsättning geometry_dump-rader för komponenterna i en geometri.
- ST_DumpPoints - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar en uppsättning geometry_dump-rader för koordinaterna i en geometri.
- ST_Expand - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar en bounding box som expanderats från en annan bounding box eller en geometri.
- ST_Extent - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Aggregerad funktion som returnerar geometriernas avgränsande box.
- ST_Force2D - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Tvinga geometrierna till ett ”2-dimensionellt läge”.
- ST_Force3D - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Tvingar geometrierna till XYZ-läge. Detta är ett alias för ST_Force3DZ.
- ST_Force3DZ - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Tvinga geometrierna till XYZ-läge.
- ST_ForceCollection - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Konvertera geometrin till en GEOMETRYCOLLECTION.
- ST_ForceRHR - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Tvinga orienteringen av hörnen i en polygon att följa högerhands-regeln.
- ST_GMLToSQL - Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes. Returns a geometry from GML. Alias for ST_GeomFromGML.
- ST_GMLToSQL - Förbättrad: 2.0.0 standard srid valfri parameter tillagd. Returns a geometry from GML. Alias for ST_GeomFromGML.
- ST_GeomFromEWKB - Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes. Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Binary representation (EWKB).
- ST_GeomFromEWKT - Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes. Returnera ett specificerat ST_Geometry-värde från Extended Well-Known Text representation (EWKT).
- ST_GeomFromGML - Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes. Tar som indata GML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeomFromGML - Förbättrad: 2.0.0 standard srid valfri parameter tillagd. Tar som indata GML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeometryN - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar ett element i en geometrisamling.
- ST_GeometryType - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Returnerar SQL-MM-typen för en geometri som text.
- ST_IsClosed - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Testar om en LineStrings start- och slutpunkter är sammanfallande. För en PolyhedralSurface testas om den är sluten (volymetrisk).
- ST_MakeEnvelope - Förbättrad: 2.0: Möjlighet att ange ett kuvert utan att ange en SRID infördes. Skapar en rektangulär polygon från minimi- och maximikoordinater.
- ST_MakeValid - Förbättrad: 2.0.1, hastighetsförbättringar Försöker göra en ogiltig geometri giltig utan att förlora toppar.

- ST_NPoints - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Returnerar antalet punkter (vertices) i en geometri.
- ST_NumGeometries - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Returnerar antalet element i en geometrisamling.
- ST_Relate - Förbättrad: 2.0.0 - stöd för att ange en regel för gränsnoder har lagts till. Testar om två geometrier har en topologisk relation som matchar ett Intersection Matrix-mönster, eller beräknar deras Intersection Matrix
- ST_Rotate - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Roterar en geometri runt en ursprungspunkt.
- ST_Rotate - Förbättrad: 2.0.0 ytterligare parametrar för att ange rotationens ursprung lades till. Roterar en geometri runt en ursprungspunkt.
- ST_RotateX - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Roterar en geometri runt X-axeln.
- ST_RotateY - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Roterar en geometri runt Y-axeln.
- ST_RotateZ - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Roterar en geometri runt Z-axeln.
- ST_Scale - Förbättrad: 2.0.0 stöd för polyhedrala ytor, trianglar och TIN infördes. Skalar en geometri med givna faktorer.
- ST_ShiftLongitude - Förbättrad: 2.0.0 stöd för polyhedrala ytor och TIN infördes. Flyttar longitudkoordinaterna för en geometri mellan -180..180 och 0..360.
- ST_Summary - Förbättrad: 2.0.0 har lagt till stöd för geografi Returnerar en textsammanfattning av innehållet i en geometri.
- ST_Transform - Förbättrad: 2.0.0 stöd för polyhedrala ytor infördes. Returnerar en ny geometri med koordinater som transformerats till ett annat spatialt referenssystem.
- ST_Value - Förbättrad: 2.0.0 exclude_nodata_value valfritt argument lades till. Returnerar värdet för ett visst band i en viss kolumnx, radpixel eller vid en viss geometrisk punkt. Bandnummer börjar på 1 och antas vara 1 om det inte anges. Om exclude_nodata_value är satt till false, anses alla pixlar inklusive nodata-pixlar korsa varandra och returnerar värdet. Om värdet exclude_nodata_value inte anges läses det från rastrets metadata.
- ValidateTopology - Förbättrad: 2.0.0 effektivare detektering av kantkorsningar och korrigeringar för falska positiva som fanns i tidigare versioner. Returnerar en uppsättning validatetopology_returntype-objekt som beskriver problem med topologin.

Funktioner ändrade i PostGIS 2.0

- AddGeometryColumn - Ändrad: 2.0.0 Den här funktionen uppdaterar inte längre geometry_columns eftersom geometry_columns är en vy som läser från systemkataloger. Som standard skapar det inte heller begränsningar, utan använder istället det inbyggda typmodifieringsbeteendet för PostgreSQL. Så till exempel att bygga en wgs84 POINT-kolumn med den här funktionen motsvarar nu: ALTER TABLE some_table ADD COLUMN geom geometry (Point,4326); Lägger till en geometrikolumn i en befintlig tabell.
- AddGeometryColumn - Ändrad: 2.0.0 Om du vill ha det gamla beteendet för begränsningar, använd standardvärdet use_typmod, men sätt det till false. Lägger till en geometrikolumn i en befintlig tabell.

- AddGeometryColumn - Ändrad: 2.0.0 Vyer kan inte längre registreras manuellt i geometry_columns, men vyer som är byggda mot geometri typmod-tabellgeometrier och används utan omslagsfunktioner kommer att registrera sig korrekt eftersom de ärver typmod-beteendet för sin överordnade tabellkolumn. Vyer som använder geometrifunktioner som matar ut andra geometrier måste castas till typmod-geometrier för att dessa vygeometrikolumner ska registreras korrekt i geometry_columns. Se . Lägger till en geometrikolumn i en befintlig tabell.
- Box3D - Ändrad: 2.0.0 I versioner före 2.0 brukade det finnas en box2d istället för box3d. Eftersom box2d är en föråldrad typ ändrades detta till box3d. Returnerar box 3d-representationen av den omslutande boxen i rastret.
- DropGeometryColumn - Changed: 2.0.0. Retained for backward compatibility after geometry_columns became a view of the system catalogs. Tar bort en geometrikolumn från en spatial tabell.
- DropGeometryTable - Changed: 2.0.0. Retained for backward compatibility after geometry_columns became a view of the system catalogs. Tar bort en tabell och alla dess referenser i geometry_columns.
- Populate_Geometry_Columns - Ändrad: 2.0.0 Som standard används nu typmodifierare i stället för kontrollbegränsningar för att begränsa geometrityper. Du kan fortfarande använda check constraint-beteende istället genom att använda den nya use_typmod och ställa in den på false. Säkerställer att geometrikolumner definieras med typmodifierare eller har lämpliga spatiala begränsningar.
- ST_3DExtent - Ändrad: 2.0.0 I tidigare versioner hette detta ST_Extent3D Aggregerad funktion som returnerar geometriernas 3D-begränsningsbox.
- ST_3DLength - Ändrad: 2.0.0 I tidigare versioner brukade detta kallas ST_Length3D Returnerar 3D-längden för en linjär geometri.
- ST_3DMakeBox - Ändrad: 2.0.0 I tidigare versioner brukade detta kallas ST_MakeBox3D Skapar en BOX3D som definieras av två 3D-punktgeometrier.
- ST_3DPerimeter - Ändrad: 2.0.0 I tidigare versioner brukade detta kallas ST_Perimeter3D Returnerar 3D-perimetern för en polygonal geometri.
- ST_AsBinary - Changed: 2.0.0 Inputs to this function can not be unknown -- must be geometry. Constructs such as ST_AsBinary('POINT(1 2)') are no longer valid and you will get an n ST_AsBinary(unknown) is not unique error. Code like that needs to be changed to ST_AsBinary('POINT(1 2)'::geometry);. If that is not possible, then install legacy.sql. Returnera OGC/ISO Well-Known Binary (WKB)-representationer av geometrin/geografin utan SRID-metadata.
- ST_AsGML - Ändrad: 2.0.0 använder standardnamn för args Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsGeoJSON - Ändrad: 2.0.0 stödjer standard args och namngivna args. Returnerar en geometri eller funktion i GeoJSON-format.
- ST_AsSVG - Ändrad: 2.0.0 för att använda standardargs och stödja namngivna args Returnerar SVG-banedata för en geometri.
- ST_EndPoint - Ändrad: 2.0.0 fungerar inte längre med MultiLineStrings med en geometri. I äldre versioner av PostGIS skulle en MultiLineString med en enda rad fungera med den här funktionen och returnera slutpunkten. I 2.0.0 returnerar den NULL som vilken annan MultiLineString som helst. Det gamla beteendet var en odokumenterad funktion, men personer som antog att de hade sina data lagrade som LINESTRING kan uppleva att dessa returnerar NULL i 2.0.0. Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_GDALDrivers - Ändrad: 2.0.6, 2.1.3 - som standard är inga drivrutiner aktiverade, såvida inte GUC eller miljövariabeln gdal_enabled_drivers är inställd. Returnerar en lista över rasterformat som stöds av PostGIS via GDAL. Endast de format med can_write=True kan användas av ST_AsGDALRaster

- ST_GeomFromText - Ändrad: 2.0.0 I tidigare versioner av PostGIS var ST_GeomFromText('GEOMETRYCO tillåtet. Detta är nu olagligt i PostGIS 2.0.0 för att bättre överensstämma med SQL/MM-standarder. Detta ska nu skrivas som ST_GeomFromText('GEOMETRYCOLLECTION EMPTY') Returnera ett specificerat ST_Geometry-värde från Well-Known Text representation (WKT).
- ST_GeometryN - Ändrad: 2.0.0 Tidigare versioner skulle returnera NULL för singulära geometrier. Detta ändrades till att returnera geometrin för ST_GeometryN(..,1) fallet. Returnerar ett element i en geometrisamling.
- ST_IsEmpty - Ändrad: 2.0.0 I tidigare versioner av PostGIS var ST_GeomFromText('GEOMETRYCOLLECTI tillåtet. Detta är nu olagligt i PostGIS 2.0.0 för att bättre överensstämma med SQL/MM-standarder Testar om en geometri är tom.
- ST_Length - Ändrad: 2.0.0 Genomgripande ändring - i tidigare versioner gav en tillämpning av detta på en MULTI/POLYGON av typen geografi omkretsen av POLYGONEN/MULTIPOLYGONEN. I 2.0.0 ändrades detta till att returnera 0 för att vara i linje med geometrins beteende. Använd ST_Perimeter om du vill ha omkretsen av en polygon Returnerar 2D-längden för en linjär geometri.
- ST_LocateAlong - Ändrad: 2.0.0 i tidigare versioner kallades detta för ST_Locate_Along_Measure. Returnerar den eller de punkter på en geometri som matchar ett mätvärde.
- ST_LocateBetween - Ändrad: 2.0.0 - i tidigare versioner hette detta ST_Locate_Between_Measures. Returnerar de delar av en geometri som matchar ett mätintervall.
- ST_ModEdgeSplit - Ändrad: 2.0 - I tidigare versioner var detta felaktigt benämnt ST_ModEdgesSplit Dela en kant genom att skapa en ny nod längs en befintlig kant, modifiera den ursprungliga kanten och lägga till en ny kant.
- ST_NumGeometries - Ändrad: 2.0.0 I tidigare versioner skulle detta returnera NULL om geometrin inte var en samling/MULTI-typ. 2.0.0+ returnerar nu 1 för enskilda geometrier, t.ex. POLYGON, LINESTRING, POINT. Returnerar antalet element i en geometrisamling.
- ST_NumInteriorRings - Ändrad: 2.0.0 - i tidigare versioner kunde man skicka en MULTIPOLYGON och få tillbaka antalet inre ringar i den första POLYGONEN. Returnerar antalet inre ringar (hål) i en polygon.
- ST_PointN - Ändrad: 2.0.0 fungerar inte längre med multilinestrings med en geometri. I äldre versioner av PostGIS -- skulle en multilinestring med en enda linje fungera bra med den här funktionen och returnera startpunkten. I 2.0.0 returnerar den bara NULL som vilken annan multilinjesträng som helst. Returnerar den N:te punkten i den första LineString eller cirkulära LineString i en geometri.
- ST_ScaleX - Ändrad: 2.0.0. I WKTRaster-versioner kallades detta ST_PixelSizeX. Returnerar X-komponente av pixelbredden i enheter av koordinatreferenssystemet.
- ST_ScaleY - Ändrad: 2.0.0. I WKTRaster-versioner kallades detta ST_PixelSizeY. Returnerar Y-komponente av pixelhöjden i enheter av koordinatreferenssystemet.
- ST_SetScale - Ändrad: 2.0.0 I WKTRaster-versioner kallades detta ST_SetPixelSize. Detta ändrades i 2.0.0. Ställer in X- och Y-storleken för pixlar i enheter i koordinatreferenssystemet. Antal enheter/pixelbredd/höjd.
- ST_StartPoint - Ändrad: 2.0.0 fungerar inte längre med MultiLineStrings med en geometri. I äldre versioner av PostGIS skulle en MultiLineString med en enda rad fungera bra med den här funktionen och returnera startpunkten. I 2.0.0 returnerar den bara NULL som vilken annan MultiLineString som helst. Det gamla beteendet var en odokumenterad funktion, men personer som antog att de hade sina data lagrade som LINESTRING kan uppleva att dessa returnerar NULL i 2.0.0. Returns the first point of a LineString, CircularLineString, or NURBSCurve.

13.12.15 PostGIS Funktioner nya eller förbättrade i 1.5

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 1.5

- && - Tillgänglighet: 1.5.0 stöd för geografi infördes. Returnerar TRUE om A:s 2D-begränsningsbox skär B:s 2D-begränsningsbox.
- PostGIS_LibXML_Version - Tillgänglighet: 1,5 Returnerar versionsnumret för libxml2-biblioteket.
- ST_AddMeasure - Tillgänglighet: 1.5.0 Interpolerar mått längs en linjär geometri.
- ST_AsBinary - Tillgänglighet: 1.5.0 stöd för geografi infördes. Returnera OGC/ISO Well-Known Binary (WKB)-representationen av geometrin/geografin utan SRID-metadata.
- ST_AsGML - Tillgänglighet: 1.5.0 stöd för geografi infördes. Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsGeoJSON - Tillgänglighet: 1.5.0 stöd för geografi infördes. Returnerar en geometri eller funktion i GeoJSON-format.
- ST_AsText - Tillgänglighet: 1.5 - stöd för geografi infördes. Returnera WKT-representationen (Well-Known Text) av geometrin/geografin utan SRID-metadata.
- ST_Buffer - Tillgänglighet: 1.5 - ST_Buffer har förbättrats för att stödja olika ändkapslar och jointyper. Dessa är användbara för att t.ex. konvertera väglinjer till polygonvägar med platta eller fyrkantiga kanter istället för rundade kanter. Tunt omslag för geografi har lagts till. Beräknar en geometri som täcker alla punkter inom ett givet avstånd från en geometri.
- ST_ClosestPoint - Tillgänglighet: 1.5.0 Returnerar den 2D-punkt på g1 som ligger närmast g2. Detta är den första punkten på den kortaste linjen från den ena geometrin till den andra.
- ST_CollectionExtract - Tillgänglighet: 1.5.0 Ger en geometrisamling och returnerar en multigeometri som endast innehåller element av en angiven typ.
- ST_Covers - Tillgänglighet: 1.5 - stöd för geografi infördes. Testar om varje punkt i B ligger i A
- ST_DFullyWithin - Tillgänglighet: 1.5.0 Testar om en geometri är helt inom ett avstånd från en annan
- ST_DWithin - Tillgänglighet: 1.5.0 stöd för geografi infördes Testar om två geometrier ligger inom ett givet avstånd
- ST_Distance - Tillgänglighet: 1.5.0 Stöd för geografi infördes i 1.5. Hastighetsförbättringar för planar för att bättre hantera stora eller många vertexgeometrier Returnerar avståndet mellan två geometri- eller geografivärden.
- ST_DistanceSphere - Tillgänglighet: 1.5 - stöd för andra geometrityper än punkter infördes. Tidigare versioner fungerar bara med punkter. Returnerar minsta avstånd i meter mellan två lon/lat-geometrier med hjälp av en sfärisk jordmodell.
- ST_DistanceSpheroid - Tillgänglighet: 1.5 - stöd för andra geometrityper än punkter infördes. Tidigare versioner fungerar bara med punkter. Returnerar det minsta avståndet mellan två lon/lat-geometrier med hjälp av en sfäroid jordmodell.
- ST_DumpPoints - Tillgänglighet: 1.5.0 Returnerar en uppsättning geometry_dump-rader för koordinaterna i en geometri.
- ST_Envelope - Tillgänglighet: 1.5.0 ändrat beteende för att mata ut dubbel precision istället för float4 Returnerar en geometri som representerar en geometris avgränsande box.
- ST_Expand - Tillgänglighet: 1.5.0 ändrat beteende för att mata ut dubbel precision istället för float4-koordinater. Returnerar en bounding box som expanderats från en annan bounding box eller en geometri.

- ST_GMLToSQL - Tillgänglighet: 1.5, kräver libxml2 1.6+ Returns a geometry from GML. Alias for ST_GeomFromGML.
- ST_GeomFromGML - Tillgänglighet: 1.5, kräver libxml2 1.6+ Tar som indata GML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_GeomFromKML - Tillgänglighet: 1.5, kräver libxml2 2.6+ Tar som indata KML-representation av geometri och matar ut ett PostGIS-geometriobjekt
- ST_HausdorffDistance - Tillgänglighet: 1.5.0 Returnerar Hausdorff-avståndet mellan två geometrier.
- ST_Intersection - Tillgänglighet: 1.5 stöd för datatypen Geography infördes. Beräknar en geometri som representerar den delade delen av geometrierna A och B.
- ST_Intersects - Tillgänglighet: 1.5 stöd för geografi infördes. Testar om två geometrier skär varandra (de har minst en gemensam punkt)
- ST_Length - Tillgänglighet: 1.5.0 Stöd för geografi infördes i 1.5. Returnerar 2D-längden för en linjär geometri.
- ST_LongestLine - Tillgänglighet: 1.5.0 Returnerar den längsta 2D-linjen mellan två geometrier.
- ST_MakeEnvelope - Tillgänglighet: 1,5 Skapar en rektangulär polygon från minimi- och maximikoordinater.
- ST_MaxDistance - Tillgänglighet: 1.5.0 Returnerar det största 2D-avståndet mellan två geometrier i projicerade enheter.
- ST_ShortestLine - Tillgänglighet: 1.5.0 Returnerar den kortaste 2D-linjen mellan två geometrier
- ~= - Tillgänglighet: 1.5.0 ändrat beteende Returnerar TRUE om A:s avgränsande box är densamma som B:s.

13.12.16 PostGIS Funktioner nya eller förbättrade i 1.4

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 1.4

- Populate_Geometry_Columns - Tillgänglighet: 1.4.0 Säkerställer att geometrikolumner definieras med typmodifierare eller har lämpliga spatiala begränsningar.
- ST_Collect - Tillgänglighet: 1.4.0 - ST_Collect(geomarray) introducerades. ST_Collect förbättrades för att hantera fler geometrier snabbare. Skapar en GeometryCollection eller Multi* geometri från en uppsättning geometrier.
- ST_ContainsProperly - Tillgänglighet: 1.4.0 Testar om varje punkt i B ligger i det inre av A
- ST_GeoHash - Tillgänglighet: 1.4.0 Returnerar en GeoHash-representation av geometrin.
- ST_IsValidReason - Tillgänglighet: 1.4 Returnerar text som anger om en geometri är giltig, eller en orsak till ogiltigheten.
- ST_LineCrossingDirection - Tillgänglighet: 1.4 Returnerar ett tal som anger korsningsbeteendet för två LineStrings
- ST_LocateBetweenElevations - Tillgänglighet: 1.4.0 Returnerar de delar av en geometri som ligger inom ett höjdintervall (Z).
- ST_MakeLine - Tillgänglighet: 1.4.0 - ST_MakeLine(geomarray) introducerades. ST_MakeLine aggregatfunktioner förbättrades för att hantera fler punkter snabbare. Skapar en LineString från Point-, MultiPoint- eller LineString-geometrier.

- ST_MinimumBoundingCircle - Tillgänglighet: 1.4.0 Returnerar den minsta cirkelpolygonen som innehåller en geometri.
- ST_Union - Tillgänglighet: 1.4.0 - ST_Union förbättrades. ST_Union (geomarray) introducerades och även snabbare aggregerad samling i PostgreSQL. Beräknar en geometri som representerar punktuppsättningssammanslagningen av indatageometrierna.

13.12.17 PostGIS Funktioner nya eller förbättrade i 1.3

De funktioner som anges nedan är PostGIS-funktioner som har lagts till eller förbättrats.

Funktioner som är nya i PostGIS 1.3

- ST_AsGML - Tillgänglighet: 1.3.2 Returnera geometrin som ett GML-element version 2 eller 3.
- ST_AsGeoJSON - Tillgänglighet: 1.3.4 Returnerar en geometri eller funktion i GeoJSON-format.
- ST_CurveToLine - Tillgänglighet: 1.3.0 Konverterar en geometri som innehåller kurvor till en linjär geometri.
- ST_LineToCurve - Tillgänglighet: 1.3.0 Konverterar en linjär geometri till en krökt geometri.
- ST_SimplifyPreserveTopology - Tillgänglighet: 1.3.3 Returnerar en förenklad och giltig representation av en geometri med hjälp av Douglas-Peucker-algoritmen.

Kapitel 14

Rapportering av problem

14.1 Rapportering av programvarubuggar

Att rapportera buggar på ett effektivt sätt är ett grundläggande sätt att hjälpa PostGIS-utvecklingen. Den mest effektiva buggrapporten är den som gör det möjligt för PostGIS-utvecklare att reproducera den, så den skulle helst innehålla ett skript som utlöser den och all information om miljön där den upptäcktes. Tillräckligt bra information kan extraheras genom att köra `SELECT postgis_full_version()` [för PostGIS] och `SELECT version()` [för postgresql].

Om du inte använder den senaste versionen är det värt att ta en titt på versionens changelog först för att ta reda på om din bugg redan har åtgärdats.

Genom att använda PostGIS buggspårare säkerställer du att dina rapporter inte kasseras och håller dig informerad om hanteringsprocessen. Innan du rapporterar en ny bugg, kontrollera i databasen om den är känd och om den är det, lägg till eventuell ny information du har om den.

Du kanske vill läsa Simon Tathams artikel om hur du rapporterar buggar på ett effektivt sätt innan du skickar in en ny rapport.

14.2 Rapportering av dokumentationsproblem

Dokumentationen ska på ett korrekt sätt återspegla programvarans funktioner och beteende. Om den inte gör det kan det bero på en programvarufel eller på att dokumentationen är felaktig eller bristfällig.

Dokumentationsproblem kan också rapporteras till PostGIS bug tracker.

Om din revidering är trivial kan du bara beskriva den i ett nytt buggtrackerärende och ange dess plats i dokumentationen.

Om dina ändringar är mer omfattande är en patch definitivt att föredra. Detta är en process i fyra steg på Unix (förutsatt att du redan har git installerat):

Before editing or building the manual, read the repository's documentation development guide. It covers the executable example, generated figure, validation, HTML, and PDF workflows used by the manual.

1. Klona PostGIS git-repository. På Unix, skriv:

   git clone https://git.osgeo.org/gitea/postgis/postgis.git

   Detta kommer att lagras i katalogen postgis

2. Gör dina ändringar i dokumentationen med din favorittextredigerare. På Unix skriver du (till exempel):

   vim doc/postgis.xml

   Observera att dokumentationen är skriven i DocBook XML snarare än HTML, så om du inte är bekant med det, följ exemplet i resten av dokumentationen.

3. Skapa en patch-fil som innehåller skillnaderna från huvudkopian av dokumentationen. På Unix skriver du:

   git diff doc/postgis.xml > doc.patch

4. Bifoga korrigeringen till ett nytt problem i buggtracker.

Kapitel 15

Credits and Acknowledgments

PostGIS is built and sustained by a worldwide community. This chapter records the people, organizations, funders, and upstream projects that have contributed code, documentation, infrastructure, testing, support, and funding.

15.1 Project Steering Committee

The PostGIS Project Steering Committee (PSC) coordinates the general direction, release cycles, documentation, and outreach efforts for the PostGIS project. In addition the PSC provides general user support, accepts and approves patches from the general PostGIS community and votes on miscellaneous issues involving PostGIS such as developer commit access, new PSC members or significant API changes.

Regina Obe CI and website maintenance, Windows production and experimental builds, documentation, alignment of PostGIS with PostgreSQL releases, X3D support, management functions.

Darafei Praliaskouski Index improvements, bug fixing and geometry/geography function improvements, SFCGAL, raster, GitHub curation, and ci maintenance.

Paul Ramsey (Chair) Co-founder of PostGIS project. General bug fixing, geography support, geography and geometry index support (2D, 3D, nD index and anything spatial index), underlying geometry internal structures, GEOS functionality integration and alignment with GEOS releases, alignment of PostGIS with PostgreSQL releases, loader/dumper, and Shapefile GUI loader.

Sandro Santilli Bug fixes and maintenance, ci maintenance, git mirror management, management functions, integration of new GEOS functionality and alignment with GEOS releases, topology support, and raster framework and low level API functions.

15.2 Core Contributors Present

Nicklas Avén Distance function enhancements (including 3D distance and relationship functions) and additions, Tiny WKB (TWKB) output format and general user support

Loïc Bartoletti SFCGAL enhancements and maintenance and ci support

Dan Baston Geometry clustering function additions, other geometry algorithm enhancements, GEOS enhancements and general user support

Martin Davis GEOS enhancements and documentation

Björn Harrtell MapBox Vector Tile, GeoBuf, and Flatgeobuf functions. Gitea testing and GitLab experimentation.

Aliaksandr Kalenik Geometry Processing, PostgreSQL gist, general bug fixing

Laurenţiu Nicola Bug fixes, maintenance, stability, and performance

15.3 Core Contributors Past

Raúl Marín Rodríguez Prior PSC Member, MVT support, Bug fixing, Performance and stability improvements, GitHub curation, alignment of PostGIS with PostgreSQL releases

Bborie Park Prior PSC Member. Raster development, integration with GDAL, raster loader, user support, general bug fixing, testing on various OS (Slackware, Mac, Windows, and more)

Mark Cave-Ayland Prior PSC Member. Coordinated bug fixing and maintenance effort, spatial index selectivity and binding, loader/dumper, and Shapefile GUI Loader, integration of new and new function enhancements.

Jorge Arévalo Raster development, GDAL driver support, loader

Olivier Courtin (Emeritus) Input/output XML (KML,GML)/GeoJSON functions, 3D support and bug fixes.

Chris Hodgson Prior PSC Member. General development, site and buildbot maintenance, OSGeo incubation management

Mateusz Loskot CMake support for PostGIS, built original raster loader in python and low level raster API functions

Kevin Neufeld Prior PSC Member. Documentation and documentation support tools, buildbot maintenance, advanced user support on PostGIS newsgroup, and PostGIS maintenance function enhancements.

Dave Blasby The original developer/Co-founder of PostGIS. Dave wrote the server side objects, index bindings, and many of the server side analytical functions.

Jeff Lounsbury Original development of the Shapefile loader/dumper.

Mark Leslie Ongoing maintenance and development of core functions. Enhanced curve support. Shapefile GUI loader.

Pierre Racine Architect of PostGIS raster implementation. Raster overall architecture, prototyping, programming support

David Zwarg Raster development (mostly map algebra analytic functions)

15.4 PostGIS Raster Origins

PostGIS raster discussions date to 2002 and included early PGCHIP experiments. Xing Lin developed the **geotiff2pgraster** PGRaster prototype during Google Summer of Code 2007. The WKT Raster project began in 2008, Jorge Arévalo developed the first GDAL driver prototype during Google Summer of Code 2009, and the 2010 FOSS4G code sprint agreed to make WKT Raster the PostGIS raster type. Raster support shipped as part of PostGIS 2.0 in 2012.

The early implementation team included Pierre Racine, Mateusz Loskot, Sandro Santilli, Jorge Arévalo, Bborie Park, David Zwarg, and Regina Obe. Their work established the raster type, loader, GDAL integration, raster/vector operations, map algebra, documentation, and testing that became the current raster subsystem.

Other recorded contributions include Jeff Adams's early sprint work, Qing Liu's raster distance analysis, Maurício Carvalho Mathias de Paulo's QGIS raster plugin and related paper with Lubia Vinhas, and Ignacio Brodin's gvSIG raster plugin and presentation with Jorge Sanz. Bryce L. Nordgren created early raster coverage and seamless-architecture diagrams. Pierre Racine wrote the original raster/-vector tutorial and created its walkthrough and early API diagrams.

The historical project planning record credits Steve Cumming and Université Laval with funding Pierre Racine's work from June 2008 through July 2012; Martin Daly and Cadcorp with funding about six months of Mateusz Loskot's work through June 2009; Tyler Erickson and Michigan Tech Research Institute with a $1,200 USD contribution; Miguel Lizondo and Deimos Space with support for about two years of Jorge Arévalo's work through June 2011; and the Center for Vectorborne Diseases at UC Davis with funding Bborie Park's work from March 2010 through July 2012.

15.5 Other Contributors

Individual Contributors

@amwps290 (GitHub user)
@kannes (GitHub user: Hannes)
@sunhm89 (GitHub user)
@wewindy (GitHub user: Chen)
Adrien Berchet
Alex Bodnaru
Alex Mayrhofer
Andrea Peri
Andreas Forø Tollefsen
Andreas Neumann
Andreas Schild
Andrew Gierth
Andrew Harvey
Anne Ghisla
Antoine Bajolet
Ariel Mashraki
Arno Renevier
Arthur Bazin
Arthur Lesuisse
Artur Zakirov
Ayo Adesugba
Barbara Phillipot
Baris Ergun
Bas Couwenberg
Ben Jubb
Benjamin Morel
Bernd Reiß
Bernhard Reiter
Bill Mill
Björn Esser
Brian Hamlin
Bruce Rindahl
Bruno Wolff III
Bryce L. Nordgren
Carl Anderson
Carlos López Quintanilla
Charles Karney
Charlie Savage
Chris Mayo
Christian Schroeder
Christoph Berg
Christoph Moench-Tegeder
Dan Faudemer
Dan Levy
Dane Springmeyer
Daniel Nylander
Dapeng Wang
Daryl Herzmann
Dave Fuhry
David G. Johnston
David Garnier
David Skea
David Techer
Dennis Tighe
Denys Kovshun
Devrim Gündüz
Dian M Fay
Dmitry Vasilyev
Edouard Choinière
Eduin Carrillo
Egor Ivkov
Eric Ridge
Esteban Zimanyi
Eugene Antimirov
Florian Weimer
François Bonzon
Frank Warmerdam
George Silva
Gerald Fenoy
Gino Lucrezi
Giovanni Zezza
Grant Hutchins
Greg Troxel
Guillaume Lelarge
Giuseppe Broccolo
Han Wang
Hannes Janetzek
Hans Lemuet
Haribabu Kommi
Harrissou Sant-anna
Havard Tveite
Ignacio Brodin
IIDA Tetsushi
Ingvild Nystuen
Jackie Leng
James Addison
James Marca
Jan Katins
Jan Tojnar
Jason Smith
Jean Felder
Jeff Adams
Jelte Fennema
Jim Jones
Joe Conway
Jonne Savolainen
Jorge Gustavo Rocha
Jorge Sanz
Jose Carlos Martinez Llari
Jostein Leira
Jörg Habenicht
Juan Sebastian Jimenez Aldana
Juergen E. Fischer
Julien Monticolo
Julien Rouhaud
Kashif Rasul
Klaus Foerster
Kris Jurka
Kristian Thy
Lars Aksel Opsahl
Laurenţiu Nicola
Laurenz Albe
Lars Roessiger
Lauri Kajan
Leo Hsu
Loic Dachary
Louis Descoteaux
Luca Delucchi
Luca S. Percich
Lucas C. Villa Real
Ludovic Hirlimann
Lubia Vinhas
Maksim Korotkov
Manuel Morillo
Maria Arias de Reyna
Marc Ducobu
Mark Sondheim
Markus Schaber
Maurício
Maxim Ko
Maxime G
Maxime va
Maxime S
Megan Ma
Mehmet I
Michael F
Michal Pá
Mike Toev
Mitchell H
Nathan W
Nathaniel
Nicolas Ro
Nikita Shu
Nikolai Be
Norman B
Norman V
Patricia To
Paul Norm
Paulo Cesa
Pavel Sele
Pawel Ost
Peter Ngu
Pierre Car
Qing Liu
Rafal Mag
Ralph Mas
Rémi Cura
Richard G
Robert Co
Rodrigo S
Roger Cre
Ron Maye
Sam Peter
Sarath Ku
Sergei Sho
Sergey Fe
Shinichi S
Shoaib Bu
Silvio Gros
Simon Gre
Stefan Cor
Steffen Ma
Stepan Ku
Stephen F
Stephen W
Steven Ott
Svyatoslav
Talha Rizv
Temuri Do
Teramoto
Thomas M
Thomas Pe
Tobias Bus
Tom Glanc
Tom van T
Victor Col
Vincent Br
Vincent M
Vincent Pi
Volf Tomáš
Xing Lin
Yoichi Kay

Corporate Sponsors These are corporate entities that have contributed developer time, hosting, or direct monetary funding to the PostGIS project. In alphabetical order:

- Aiven
- Arrival 3D
- Associazione Italiana per l'Informazione Geografica Libera (GFOSS.it)
- AusVet
- Avencia
- Azavea
- Boundless
- Cadcorp
- Camptocamp
- Carto
- Crunchy Data
- City of Boston (DND)
- City of Helsinki
- Clever Elephant Solutions
- Cooperativa Alveo
- Deimos Space
- Faunalia
- Geographic Data BC
- Google
- HighGo
- Hunter Systems Group
- IITM Pravartak Security Team
- INIA-CSIC
- ISciences, LLC
- Jirotech (formerly LISAsoft)
- Koordinates
- Kontur
- Lidwala Consulting Engineers
- Logical Tracking & Tracing International AG
- Maponics
- Maumap
- Michigan Tech Research Institute
- Natural Resources Canada
- Norwegian Forest and Landscape Institute
- Norwegian Institute of Bioeconomy Research (NIBIO)
- OSGeo
- Oslandia
- Palantir Technologies
- Paragon Corporation
- PlanetScale
- Postgres Pro

- R3 GIS
- Refractions Research
- Regione Toscana - SITA
- Safe Software
- Sirius Corporation plc
- Stadt Uster
- UC Davis Center for Vectorborne Diseases
- Université Laval
- U.S. Census Bureau
- U.S. Department of State (HIU)
- Zonar Systems

Crowd Funding Campaigns Crowd funding campaigns are campaigns we run to get badly wanted features funded that can service a large number of people. Each campaign is specifically focused on a particular feature or set of features. Each sponsor chips in a small fraction of the needed funding and with enough people/organizations contributing, we have the funds to pay for the work that will help many. If you have an idea for a feature you think many others would be willing to co-fund, please post to the PostGIS newsgroup your thoughts and together we can make it happen.

PostGIS has experimented with community fundraising for features in the past. We used PledgeBank and we got two successful campaigns out of it.

postgistopology - 10 plus sponsors each contributed $250 USD to build toTopoGeometry function and beef up topology support in 2.0.0. It happened.

postgis64windows - 20 someodd sponsors each contributed $100 USD to pay for the work needed to work out PostGIS 64-bit issues on windows. It happened.

External Libraries and Platforms PostgreSQL, the extensible database system on which PostGIS is built. PostgreSQL provides the SQL engine, transactions, query planner, extension framework, and index access methods used by PostGIS.

The GEOS geometry operations library.

The PROJ coordinate transformation and geodesic calculation library.

The GDAL Geospatial Data Abstraction Library used to power much of the raster functionality introduced in PostGIS 2. In kind, improvements needed in GDAL to support PostGIS are contributed back to the GDAL project.

LibXML2 for XML-based spatial formats, and JSON-C for GeoJSON input.

The GNU Multiple Precision Arithmetic Library (GMP) for precision-sensitive topology operations.

protobuf-c for Mapbox Vector Tile and Geobuf output.

The iconv API, supplied by the C library or GNU libiconv, for character encoding conversion in the shapefile loader.

SFCGAL for advanced 2D and 3D functions, together with the CGAL algorithms that SFCGAL exposes.

Bundled Libraries and Source Code PostGIS ships selected source from FlatGeobuf and FlatBuffers to support FlatGeobuf input and output.

Wagyu and Mapbox Geometry provide polygon clipping and validation for Mapbox Vector Tile output.

Ryu provides round-trip-safe floating-point formatting, and uthash provides in-process hash tables.

The shapefile loader and dumper include code derived from ShapeLib, originally by Frank Warmerdam and Even Rouault. They also include public-domain `getopt` code from AT&T.

`liblwgeom/lookup3.c` contains the public-domain `lookup3` hash function by Bob Jenkins.

Bilaga A

Anteckningar för release

A.1 PostGIS 3.7.0

This version requires GEOS 3.10+, PostgreSQL 14-19beta1, Proj 6.1+, libgmp. To take advantage of all features postgis extension features, GEOS 3.15+ is needed. To take advantage of all postgis_sfcgal extension features SFCGAL 2.3+ is needed.

A.1.1 Förändringar

#5688, [topology] topology building functions now interpret tolerance 0 (old default) to really mean 0 and want -1 (new default) to mean ”use topology precision”(Sandro Santilli)

#6024, Remove support for PostgreSQL 12 and 13 (Regina Obe)

#6040, ST_GeoHash round trip through ST_GeomFromGeoHash (Paul Ramsey)

#6053, [address_standardizer] Extension removed and moved to https://github.com/postgis/address_standa (Paul Ramsey)

#6052, [tiger_geocoder] Extension removed and moved to https://git.osgeo.org/postgis/postgis_tiger_geoco (Regina Obe)

#6079, Remove support for GEOS < 3.10 (Sandro Santilli)

#1793, Remove ST_MemCollect aggregate (Darafei Praliaskouski)

GH-848, CG_AlphaShape now returns a MultiPolygon (Jean Felder)

A.1.2 Nya funktioner

#1124, #2935, #3033, #4658, #4659, [loader] Rework loader arguments: shp2pgsql can create UNLOGGED tables and choose the feature id column name, shp2pgsql --drop-table can emit DROP TABLE before prepare output, raster2pgsql/shp2pgsql expose loader actions as long options, both loaders accept long aliases for existing options, and

-if-not-exists makes creation actions idempotent (Darafei Praliaskouski)

#4208, Add single-geometry variants of ST_MaxDistance and ST_LongestLine (Darafei Praliaskouski)

[topology] FindVertexSegmentPairsBelowDistance function (Sandro Santilli)

ST_CoverageEdges, returns MultiLinestring of distinct shared edges in polygonal coverage (Paul Ramsey)

ST_MinimumSpanningTree, window function to calculate MST (Paul Ramsey)

#5993, [topology] Add max_edges parameter to TopoGeo_AddLinestring (Sandro Santilli)

#6001, support MultiLineString in ST_MakeLine (Paul Ramsey)

#2858, ST_MMin and ST_MMax (Paul Ramsey)

#5941, [raster] Support GDT_Float16 pixel type (Darafei Praliaskouski)

#5992, Optimize GiST index for repeated edge subdivision in topology (Darafei Praliaskouski)

#5702, Allow the compiler to detect the parallelism -flto=auto (Darafei Praliaskouski)

#4798, ST_AsGeoJSON warns about duplicate property keys (Darafei Praliaskouski)

#5950, Document POSTGIS_REGRESS_DB_OWNER for sandboxed regression roles (Darafei Praliaskouski)

#4332, Clarify the scope of several Function Reference categories (Darafei Praliaskouski)

#4222, [raster] ST_DumpAsPolygons honours PostgreSQL interrupts (Darafei Praliaskouski)

#5109, Document the meaning of topology.next_left_edge and topology.next_right_edge links (Darafei Praliaskouski)

#5889, [topology] Include representative locations in topology build errors (Darafei Praliaskouski)

#2614, Use GEOSPreparedDistance and caching to accelerate ST_DWithin (Paul Ramsey)

#6022, Document ST_AsMVTGeom tile-coordinate simplification (Darafei Praliaskouski)

GH-839, ST_Multi support for TIN and surfaces (Loïc Bartoletti)

#4398, Add ST_CatmullSmoothing smoothes but retains original vertices (Paul Ramsey)

#4560, ST_3DInterpolatePoint for M interpolation from XYZ inputs (Paul Ramsey)

#1291, ST_MakePolygon support for curved rings (Darafei Praliaskouski)

GT-270, Add NURBSCurve (Loïc Bartoletti)

#2863, Add ST_XSize, ST_YSize, ST_ZSize, and ST_MSize dimension helpers (Darafei Praliaskouski)

[fuzzers] Centralize OSS-Fuzz build/dependency logic in PostGIS scripts, prefer requested CXX over pg_config --cc for internal C++ checks, and make fuzzer linking/runtime dependency packaging portable (Darafei Praliaskouski)

A.1.3 Enhancements

#2045, pgsql2shp query dumps no longer require temporary table privileges (Darafei Praliaskouski)

#1577, [loader] Report unsupported .zip archive input before trying to open shapefile sidecar files (Darafei Praliaskouski)

#3158, Move geometry and geography typmod bit helpers out of the public liblwgeom header (Darafei Praliaskouski)

#4678, [raster] Move st_sum4ma neighborhood callback to C (Darafei Praliaskouski)

#2116, [raster] Add ST_Value nearest-neighbor boundary options (Darafei Praliaskouski)

#2804, #4315, [raster] Support two-argument ST_MapAlgebra callbacks and pass callback call data as actual arguments (Darafei Praliaskouski)

#2898, Document SQL function cost tiers for contributors

#6062, [topology] Stop using recursive snapping, for improved robustness (Sandro Santilli)

#6065, Improved winding order computation robustness for rings having collapsed elements (Sandro Santilli)

#3743, #4385, [raster loader] Document raster2pgsql -s FROM_SRID:SRID reprojection support and clarify index creation timing for append loads (Darafei Praliaskouski)

#2137, [raster] Reuse matching source raster overviews in raster2pgsql -l (Darafei Praliaskouski)

#4749, Use point-in-polygon predicate fast paths for point-only GeometryCollections (Darafei Praliaskouski)

#5532, Validate the manual against DocBook XMLSchema in check-xml (Darafei Praliaskouski)

#6087, Add a repository-owned CI status checker and static dashboard generator (Darafei Praliaskouski)

#2807, [raster] Preserve missing NODATA values in ST_MapAlgebra outputs (Darafei Praliaskouski)

#2832, [raster] Avoid padding single-tile raster2pgsql overviews when padding is not requested (Darafei Praliaskouski)

#2623, Generate a pgsql2shp GID field when exporting geometry-only shapefiles (Darafei Praliaskouski)

#2850, Preserve pgsql2shp numeric precision and scale in DBF metadata (Darafei Praliaskouski)

#2386, Add raster2pgsql man page (Darafei Praliaskouski)

Build PostgreSQL extension modules with `-fno-semantic-interposition` when available so LTO can optimize same-DSO calls to exported PostGIS entry points directly instead of routing through the module PLT; in a local PostgreSQL 18 `pgbench` microbenchmark over 100k four-point groups this reduced `ST_Collect` aggregate latency by about 4% (Darafei Praliaskouski)

#5973, Install PostGIS utilities under `pg_config --bindir` by default and document `pg_config`-derived install paths more accurately (Darafei Praliaskouski)

#2808, Clarify postgis_full_version() output when installed component scripts are newer than core scripts (Darafei Praliaskouski)

#5593, Docs: render manual images with GraphicsMagick without temporary files, enabling parallel builds, keeping ImageMagick fallbacks working, and documenting the generator workflow for contributors (Darafei Praliaskouski)

#5645, Docs: keep code operators (->”) intact in translated manuals by enforcing verbatim CSS (Darafei Praliaskouski)

#1408, Docs: flatten transparent manual images onto white backgrounds for PDF builds while keeping HTML images transparent (Darafei Praliaskouski)

#2838, Emit valid X3D Coordinate nodes for 2D polygon output (Darafei Praliaskouski)

GH-899, [raster] Honor PostgreSQL interrupts in long-running GDAL progress callbacks (Darafei Praliaskouski)

#2583, Preserve Z/ZM dimensional qualifiers in ST_AsEWKT output for TIN (Darafei Praliaskouski)