Manuel PostGIS 3.5.4dev

DEV (Fri 29 Aug 2025 09:16:00 AM UTC rev. eaddd48)

Contents

1 Introduction **1**

1.1 Comité de direction du projet (Project Steering Committee) . . . 1

1.2 Contributeurs cœur actuels . . . 2

1.3 Anciens contributeurs cœur . . . 2

1.4 Autres contributeurs . . . 3

2 Installation de PostGIS **6**

2.1 Version courte . . . 6

2.2 Compilation et installation depuis les sources . . . 6

2.2.1 Obtenir les Sources . . . 7

2.2.2 Pré requis à l'installation . . . 7

2.2.3 Configuration de la compilation . . . 8

2.2.4 Compiler . . . 10

2.2.5 Compiler les Extensions PostGIS et les déployer . . . 10

2.2.6 Tests . . . 12

2.2.7 Installation . . . 15

2.3 Installation et utilisation de l'extension address standardize . . . 16

2.4 Installation, mise à jour et chargement de données pour le géocodeur Tiger . . . 16

2.4.1 Tiger Geocoder Activation de votre base de données PostGIS . . . 16

2.4.2 Utilisation de l'Extension Address Standardizer avec le Geocodeur Tiger . . . 19

2.4.3 Outils nécessaires pour charger des données tiger . . . 19

2.4.4 Mise à jour du géocoder Tiger et de ses données . . . 20

2.5 Problèmes courants pendant l'installation . . . 21

3 PostGIS Administration **22**

3.1 Optimisation des performances . . . 22

3.1.1 Démarrage . . . 22

3.1.2 Temps d'exécution . . . 23

3.2 Configuring raster support . . . 23

3.3 Creating spatial databases . . . 24

3.3.1 Spatially enable database using EXTENSION . . . 24

3.3.2 Spatially enable database without using EXTENSION (discouraged) . . . 24
3.4 Upgrading spatial databases . . . 25
3.4.1 Soft upgrade . . . 25
3.4.1.1 Soft Upgrade 9.1+ using extensions . . . 25
3.4.1.2 Soft Upgrade Pre 9.1+ or without extensions . . . 26
3.4.2 Hard upgrade . . . 27

4 Gestion des données **29**
4.1 Modèle de données spatiales . . . 29
4.1.1 Géométrie OGC . . . 29
4.1.1.1 Point . . . 30
4.1.1.2 LineString . . . 30
4.1.1.3 LinearRing . . . 30
4.1.1.4 Polygon . . . 30
4.1.1.5 MultiPoint . . . 30
4.1.1.6 MultiLineString . . . 30
4.1.1.7 MultiPolygon . . . 31
4.1.1.8 GeometryCollection . . . 31
4.1.1.9 PolyhedralSurface . . . 31
4.1.1.10 Triangle . . . 31
4.1.1.11 TIN . . . 31
4.1.2 SQL/MM Part 3 - Courbes . . . 31
4.1.2.1 CircularString . . . 32
4.1.2.2 CompoundCurve . . . 32
4.1.2.3 CurvePolygon . . . 32
4.1.2.4 MultiCurve . . . 32
4.1.2.5 MultiSurface . . . 32
4.1.3 WKT et WKB . . . 33
4.2 Type de données Geometry . . . 34
4.2.1 PostGIS EWKB et EWKT . . . 34
4.3 Type de données Geography . . . 36
4.3.1 Création de tables géographiques . . . 36
4.3.2 Utilisation des tables géographiques . . . 37
4.3.3 Quand utiliser le type de données Geography . . . 38
4.3.4 FAQ Geography avancée . . . 39
4.4 Validation de la géométrie . . . 39
4.4.1 Géométrie simple . . . 39
4.4.2 Géométrie valide . . . 41
4.4.3 Gestion de la validité . . . 43

4.5 Systèmes de référence spatiale . . . 44
4.5.1 Table SPATIAL_REF_SYS . . . 45
4.5.2 Systèmes de référence spatiale définis par l'utilisateur . . . 46
4.6 Tables spatiales . . . 46
4.6.1 Créer une table spatiale . . . 46
4.6.2 Vue GEOMETRY_COLUMNS . . . 47
4.6.3 Enregistrement manuel des colonnes de géométrie . . . 48
4.7 Chargement des données spatiales . . . 50
4.7.1 Utilisation de SQL pour charger des données . . . 50
4.7.2 Utilisation de l'utilitaire qui permet de charger des fichiers Shapefile . . . 50
4.8 Extraction de données spatiales . . . 52
4.8.1 Utiliser SQL pour extraire des données . . . 52
4.8.2 Utilisation de Shapefile Dumper . . . 53
4.9 Index spatiaux . . . 54
4.9.1 Index GiST . . . 54
4.9.2 Index BRIN . . . 55
4.9.3 Index SP-GiST . . . 57
4.9.4 Optimisation de l'utilisation de l'index . . . 57

5 Requêtes spatiales **59**
5.1 Déterminer les relations spatiales . . . 59
5.1.1 Modèle à 9 intersections dimensionnellement étendu . . . 59
5.1.2 Relations spatiales nommées . . . 61
5.1.3 Relations spatiales générales . . . 62
5.2 Utilisation des index spatiaux . . . 64
5.3 Exemples de SQL spatial . . . 64

6 Conseils sur les performances **68**
6.1 Petites tables de grandes géométries . . . 68
6.1.1 Description du problème . . . 68
6.1.2 Solutions de contournement . . . 68
6.2 CLUSTER d'index géométriques . . . 69
6.3 Eviter les conversions de dimension . . . 69

7 Référence PostGIS **71**
7.1 Types de données PostGIS Geometry/Geography/Box . . . 71
7.1.1 box2d . . . 71
7.1.2 box3d . . . 72
7.1.3 geometry . . . 72
7.1.4 geometry_dump . . . 73

7.1.5 geography . . . 73
7.2 Fonctions de gestion des tables . . . 74
7.2.1 AddGeometryColumn . . . 74
7.2.2 DropGeometryColumn . . . 76
7.2.3 DropGeometryTable . . . 76
7.2.4 Find_SRID . . . 77
7.2.5 Populate_Geometry_Columns . . . 78
7.2.6 UpdateGeometrySRID . . . 79
7.3 Constructeurs de géométries . . . 80
7.3.1 ST_Collect . . . 80
7.3.2 ST_LineFromMultiPoint . . . 82
7.3.3 ST_MakeEnvelope . . . 83
7.3.4 ST_MakeLine . . . 83
7.3.5 ST_MakePoint . . . 85
7.3.6 ST_MakePointM . . . 86
7.3.7 ST_MakePolygon . . . 87
7.3.8 ST_Point . . . 89
7.3.9 ST_PointZ . . . 90
7.3.10 ST_PointM . . . 91
7.3.11 ST_PointZM . . . 91
7.3.12 ST_Polygon . . . 92
7.3.13 ST_TileEnvelope . . . 93
7.3.14 ST_HexagonGrid . . . 94
7.3.15 ST_Hexagon . . . 97
7.3.16 ST_SquareGrid . . . 98
7.3.17 ST_Square . . . 99
7.3.18 ST_Letters . . . 100
7.4 Fonctions d'accès aux géométries . . . 101
7.4.1 GeometryType . . . 101
7.4.2 ST_Boundary . . . 102
7.4.3 ST_BoundingDiagonal . . . 104
7.4.4 ST_CoordDim . . . 105
7.4.5 ST_Dimension . . . 106
7.4.6 ST_Dump . . . 106
7.4.7 ST_DumpPoints . . . 108
7.4.8 ST_DumpSegments . . . 112
7.4.9 ST_DumpRings . . . 114
7.4.10 ST_EndPoint . . . 115
7.4.11 ST_Envelope . . . 116

7.4.12 ST_ExteriorRing . . . 118
7.4.13 ST_GeometryN . . . 119
7.4.14 ST_GeometryType . . . 121
7.4.15 ST_HasArc . . . 122
7.4.16 ST_InteriorRingN . . . 123
7.4.17 ST_NumCurves . . . 123
7.4.18 ST_CurveN . . . 124
7.4.19 ST_IsClosed . . . 125
7.4.20 ST_IsCollection . . . 126
7.4.21 ST_IsEmpty . . . 128
7.4.22 ST_IsPolygonCCW . . . 129
7.4.23 ST_IsPolygonCW . . . 129
7.4.24 ST_IsRing . . . 130
7.4.25 ST_IsSimple . . . 131
7.4.26 ST_M . . . 132
7.4.27 ST_MemSize . . . 132
7.4.28 ST_NDims . . . 134
7.4.29 ST_NPoints . . . 134
7.4.30 ST_NRings . . . 135
7.4.31 ST_NumGeometries . . . 135
7.4.32 ST_NumInteriorRings . . . 136
7.4.33 ST_NumInteriorRing . . . 137
7.4.34 ST_NumPatches . . . 137
7.4.35 ST_NumPoints . . . 138
7.4.36 ST_PatchN . . . 138
7.4.37 ST_PointN . . . 139
7.4.38 ST_Points . . . 141
7.4.39 ST_StartPoint . . . 142
7.4.40 ST_Summary . . . 143
7.4.41 ST_X . . . 144
7.4.42 ST_Y . . . 145
7.4.43 ST_Z . . . 145
7.4.44 ST_Zmflag . . . 146
7.4.45 ST_HasZ . . . 147
7.4.46 ST_HasM . . . 148
7.5 Éditeurs de géométrie . . . 148
7.5.1 ST_AddPoint . . . 148
7.5.2 ST_CollectionExtract . . . 149
7.5.3 ST_CollectionHomogenize . . . 151

7.5.4 ST_CurveToLine . . . 152
7.5.5 ST_Scroll . . . 155
7.5.6 ST_FlipCoordinates . . . 155
7.5.7 ST_Force2D . . . 156
7.5.8 ST_Force3D . . . 157
7.5.9 ST_Force3DZ . . . 158
7.5.10 ST_Force3DM . . . 159
7.5.11 ST_Force4D . . . 159
7.5.12 ST_ForceCollection . . . 160
7.5.13 ST_ForceCurve . . . 161
7.5.14 ST_ForcePolygonCCW . . . 162
7.5.15 ST_ForcePolygonCW . . . 162
7.5.16 ST_ForceSFS . . . 163
7.5.17 ST_ForceRHR . . . 163
7.5.18 ST_LineExtend . . . 164
7.5.19 ST_LineToCurve . . . 165
7.5.20 ST_Multi . . . 166
7.5.21 ST_Normalize . . . 167
7.5.22 ST_Project . . . 168
7.5.23 ST_QuantizeCoordinates . . . 168
7.5.24 ST_RemovePoint . . . 170
7.5.25 ST_RemoveRepeatedPoints . . . 171
7.5.26 ST_RemoveIrrelevantPointsForView . . . 172
7.5.27 ST_RemoveSmallParts . . . 174
7.5.28 ST_Reverse . . . 176
7.5.29 ST_Segmentize . . . 177
7.5.30 ST_SetPoint . . . 178
7.5.31 ST_ShiftLongitude . . . 179
7.5.32 ST_WrapX . . . 180
7.5.33 ST_SnapToGrid . . . 181
7.5.34 ST_Snap . . . 183
7.5.35 ST_SwapOrdinates . . . 185
7.6 Validation de la géométrie . . . 186
7.6.1 ST_IsValid . . . 186
7.6.2 ST_IsValidDetail . . . 187
7.6.3 ST_IsValidReason . . . 189
7.6.4 ST_MakeValid . . . 190
7.7 Fonctions des systèmes de référence spatiale . . . 195
7.7.1 ST_InverseTransformPipeline . . . 195

7.7.2 ST_SetSRID . . . 196
7.7.3 ST_SRID . . . 197
7.7.4 ST_Transform . . . 198
7.7.5 ST_TransformPipeline . . . 200
7.7.6 postgis_srs_codes . . . 202
7.7.7 postgis_srs . . . 202
7.7.8 postgis_srs_all . . . 203
7.7.9 postgis_srs_search . . . 204
7.8 Import de géométrie . . . 205
7.8.1 Well-Known Text (WKT) . . . 205
7.8.1.1 ST_BdPolyFromText . . . 205
7.8.1.2 ST_BdMPolyFromText . . . 205
7.8.1.3 ST_GeogFromText . . . 206
7.8.1.4 ST_GeographyFromText . . . 206
7.8.1.5 ST_GeomCollFromText . . . 207
7.8.1.6 ST_GeomFromEWKT . . . 207
7.8.1.7 ST_GeomFromMARC21 . . . 209
7.8.1.8 ST_GeometryFromText . . . 211
7.8.1.9 ST_GeomFromText . . . 212
7.8.1.10 ST_LineFromText . . . 213
7.8.1.11 ST_MLineFromText . . . 214
7.8.1.12 ST_MPointFromText . . . 215
7.8.1.13 ST_MPolyFromText . . . 215
7.8.1.14 ST_PointFromText . . . 216
7.8.1.15 ST_PolygonFromText . . . 217
7.8.1.16 ST_WKTToSQL . . . 218
7.8.2 Well-Known Binary (WKB) . . . 218
7.8.2.1 ST_GeogFromWKB . . . 218
7.8.2.2 ST_GeomFromEWKB . . . 219
7.8.2.3 ST_GeomFromWKB . . . 220
7.8.2.4 ST_LineFromWKB . . . 221
7.8.2.5 ST_LinestringFromWKB . . . 222
7.8.2.6 ST_PointFromWKB . . . 223
7.8.2.7 ST_WKBToSQL . . . 224
7.8.3 Autres formats . . . 224
7.8.3.1 ST_Box2dFromGeoHash . . . 224
7.8.3.2 ST_GeomFromGeoHash . . . 225
7.8.3.3 ST_GeomFromGML . . . 226
7.8.3.4 ST_GeomFromGeoJSON . . . 229

7.8.3.5 ST_GeomFromKML . . . 230
7.8.3.6 ST_GeomFromTWKB . . . 230
7.8.3.7 ST_GMLToSQL . . . 231
7.8.3.8 ST_LineFromEncodedPolyline . . . 232
7.8.3.9 ST_PointFromGeoHash . . . 232
7.8.3.10 ST_FromFlatGeobufToTable . . . 233
7.8.3.11 ST_FromFlatGeobuf . . . 233
7.9 Export de géométrie . . . 234
7.9.1 Well-Known Text (WKT) . . . 234
7.9.1.1 ST_AsEWKT . . . 234
7.9.1.2 ST_AsText . . . 235
7.9.2 Well-Known Binary (WKB) . . . 236
7.9.2.1 ST_AsBinary . . . 236
7.9.2.2 ST_AsEWKB . . . 238
7.9.2.3 ST_AsHEXEWKB . . . 239
7.9.3 Autres formats . . . 240
7.9.3.1 ST_AsEncodedPolyline . . . 240
7.9.3.2 ST_AsFlatGeobuf . . . 241
7.9.3.3 ST_AsGeobuf . . . 241
7.9.3.4 ST_AsGeoJSON . . . 242
7.9.3.5 ST_AsGML . . . 244
7.9.3.6 ST_AsKML . . . 247
7.9.3.7 ST_AsLatLonText . . . 248
7.9.3.8 ST_AsMARC21 . . . 250
7.9.3.9 ST_AsMVTGeom . . . 252
7.9.3.10 ST_AsMVT . . . 253
7.9.3.11 ST_AsSVG . . . 255
7.9.3.12 ST_AsTWKB . . . 256
7.9.3.13 ST_AsX3D . . . 257
7.9.3.14 ST_GeoHash . . . 261
7.10 Opérateurs . . . 263
7.10.1 Opérateurs de Bounding Box . . . 263
7.10.1.1 && . . . 263
7.10.1.2 &&(geometry,box2df) . . . 263
7.10.1.3 &&(box2df,geometry) . . . 264
7.10.1.4 &&(box2df,box2df) . . . 265
7.10.1.5 &&& . . . 266
7.10.1.6 &&&(geometry,gidx) . . . 267
7.10.1.7 &&&(gidx,geometry) . . . 268

7.10.1.8 &&&(gidx,gidx) . . . 269
7.10.1.9 &< . . . 269
7.10.1.10 &<| . . . 270
7.10.1.11 &> . . . 271
7.10.1.12 << . . . 272
7.10.1.13 <<| . . . 273
7.10.1.14 = . . . 273
7.10.1.15 >> . . . 275
7.10.1.16 @ . . . 276
7.10.1.17 @(geometry,box2df) . . . 276
7.10.1.18 @(box2df,geometry) . . . 277
7.10.1.19 @(box2df,box2df) . . . 278
7.10.1.20 |&> . . . 279
7.10.1.21 |>> . . . 280
7.10.1.22 ~ . . . 280
7.10.1.23 ~(geometry,box2df) . . . 281
7.10.1.24 ~(box2df,geometry) . . . 282
7.10.1.25 ~(box2df,box2df) . . . 283
7.10.1.26 ~= . . . 284
7.10.2 Opérateurs de distance . . . 284
7.10.2.1 <-> . . . 284
7.10.2.2 |=| . . . 286
7.10.2.3 <#> . . . 287
7.10.2.4 <<->> . . . 289
7.11 Relations spatiales . . . 289
7.11.1 Relations topologiques . . . 289
7.11.1.1 ST_3DIntersects . . . 289
7.11.1.2 ST_Contains . . . 290
7.11.1.3 ST_ContainsProperly . . . 294
7.11.1.4 ST_CoveredBy . . . 296
7.11.1.5 ST_Covers . . . 297
7.11.1.6 ST_Crosses . . . 298
7.11.1.7 ST_Disjoint . . . 300
7.11.1.8 ST_Equals . . . 301
7.11.1.9 ST_Intersects . . . 302
7.11.1.10 ST_LineCrossingDirection . . . 304
7.11.1.11 ST_OrderingEquals . . . 307
7.11.1.12 ST_Overlaps . . . 308
7.11.1.13 ST_Relate . . . 311

7.11.1.14 ST_RelateMatch . . . 313
7.11.1.15 ST_Touches . . . 314
7.11.1.16 ST_Within . . . 316
7.11.2 Relations de distance . . . 318
7.11.2.1 ST_3DDWithin . . . 318
7.11.2.2 ST_3DDFullyWithin . . . 319
7.11.2.3 ST_DFullyWithin . . . 319
7.11.2.4 ST_DWithin . . . 320
7.11.2.5 ST_PointInsideCircle . . . 322
7.12 Fonctions de mesure . . . 322
7.12.1 ST_Area . . . 322
7.12.2 ST_Azimuth . . . 324
7.12.3 ST_Angle . . . 325
7.12.4 ST_ClosestPoint . . . 326
7.12.5 ST_3DClosestPoint . . . 328
7.12.6 ST_Distance . . . 329
7.12.7 ST_3DDistance . . . 331
7.12.8 ST_DistanceSphere . . . 332
7.12.9 ST_DistanceSpheroid . . . 333
7.12.10 ST_FrechetDistance . . . 334
7.12.11 ST_HausdorffDistance . . . 335
7.12.12 ST_Length . . . 336
7.12.13 ST_Length2D . . . 338
7.12.14 ST_3DLength . . . 338
7.12.15 ST_LengthSpheroid . . . 339
7.12.16 ST_LongestLine . . . 340
7.12.17 ST_3DLongestLine . . . 342
7.12.18 ST_MaxDistance . . . 343
7.12.19 ST_3DMaxDistance . . . 344
7.12.20 ST_MinimumClearance . . . 345
7.12.21 ST_MinimumClearanceLine . . . 346
7.12.22 ST_Perimeter . . . 346
7.12.23 ST_Perimeter2D . . . 348
7.12.24 ST_3DPerimeter . . . 348
7.12.25 ST_ShortestLine . . . 349
7.12.26 ST_3DShortestLine . . . 351
7.13 Fonctions de superposition . . . 352
7.13.1 ST_ClipByBox2D . . . 352
7.13.2 ST_Difference . . . 353

7.13.3 ST_Intersection . . . 354
7.13.4 ST_MemUnion . . . 357
7.13.5 ST_Node . . . 357
7.13.6 ST_Split . . . 358
7.13.7 ST_Subdivide . . . 361
7.13.8 ST_SymDifference . . . 363
7.13.9 ST_UnaryUnion . . . 365
7.13.10 ST_Union . . . 365
7.14 Traitement des géométries . . . 368
7.14.1 ST_Buffer . . . 368
7.14.2 ST_BuildArea . . . 373
7.14.3 ST_Centroid . . . 374
7.14.4 ST_ChaikinSmoothing . . . 376
7.14.5 ST_ConcaveHull . . . 378
7.14.6 ST_ConvexHull . . . 381
7.14.7 ST_DelaunayTriangles . . . 383
7.14.8 ST_FilterByM . . . 388
7.14.9 ST_GeneratePoints . . . 389
7.14.10 ST_GeometricMedian . . . 390
7.14.11 ST_LineMerge . . . 391
7.14.12 ST_MaximumInscribedCircle . . . 393
7.14.13 ST_LargestEmptyCircle . . . 395
7.14.14 ST_MinimumBoundingCircle . . . 397
7.14.15 ST_MinimumBoundingRadius . . . 399
7.14.16 ST_OrientedEnvelope . . . 399
7.14.17 ST_OffsetCurve . . . 400
7.14.18 ST_PointOnSurface . . . 404
7.14.19 ST_Polygonize . . . 406
7.14.20 ST_ReducePrecision . . . 408
7.14.21 ST_SharedPaths . . . 410
7.14.22 ST_Simplify . . . 412
7.14.23 ST_SimplifyPreserveTopology . . . 413
7.14.24 ST_SimplifyPolygonHull . . . 415
7.14.25 ST_SimplifyVW . . . 418
7.14.26 ST_SetEffectiveArea . . . 419
7.14.27 ST_TriangulatePolygon . . . 421
7.14.28 ST_VoronoiLines . . . 423
7.14.29 ST_VoronoiPolygons . . . 424
7.15 Couvertures . . . 426

7.15.1 ST_CoverageInvalidEdges . . . 426
7.15.2 ST_CoverageSimplify . . . 427
7.15.3 ST_CoverageUnion . . . 429
7.16 Transformations affines . . . 430
7.16.1 ST_Affine . . . 430
7.16.2 ST_Rotate . . . 432
7.16.3 ST_RotateX . . . 433
7.16.4 ST_RotateY . . . 434
7.16.5 ST_RotateZ . . . 435
7.16.6 ST_Scale . . . 436
7.16.7 ST_Translate . . . 437
7.16.8 ST_TransScale . . . 438
7.17 Fonctions de clustering . . . 439
7.17.1 ST_ClusterDBSCAN . . . 439
7.17.2 ST_ClusterIntersecting . . . 442
7.17.3 ST_ClusterIntersectingWin . . . 442
7.17.4 ST_ClusterKMeans . . . 443
7.17.5 ST_ClusterWithin . . . 445
7.17.6 ST_ClusterWithinWin . . . 446
7.18 Fonctions des boîtes de délimitation . . . 447
7.18.1 Box2D . . . 447
7.18.2 Box3D . . . 448
7.18.3 ST_EstimatedExtent . . . 448
7.18.4 ST_Expand . . . 449
7.18.5 ST_Extent . . . 451
7.18.6 ST_3DExtent . . . 452
7.18.7 ST_MakeBox2D . . . 453
7.18.8 ST_3DMakeBox . . . 454
7.18.9 ST_XMax . . . 454
7.18.10 ST_XMin . . . 455
7.18.11 ST_YMax . . . 456
7.18.12 ST_YMin . . . 457
7.18.13 ST_ZMax . . . 458
7.18.14 ST_ZMin . . . 459
7.19 Référencement linéaire . . . 460
7.19.1 ST_LineInterpolatePoint . . . 460
7.19.2 ST_3DLineInterpolatePoint . . . 462
7.19.3 ST_LineInterpolatePoints . . . 462
7.19.4 ST_LineLocatePoint . . . 463

7.19.5 ST_LineSubstring . . . 464
7.19.6 ST_LocateAlong . . . 466
7.19.7 ST_LocateBetween . . . 467
7.19.8 ST_LocateBetweenElevations . . . 469
7.19.9 ST_InterpolatePoint . . . 470
7.19.10 ST_AddMeasure . . . 470
7.20 Fonctions de trajectoire . . . 471
7.20.1 ST_IsValidTrajectory . . . 471
7.20.2 ST_ClosestPointOfApproach . . . 472
7.20.3 ST_DistanceCPA . . . 473
7.20.4 ST_CPAWithin . . . 474
7.21 Fonctions de version . . . 474
7.21.1 PostGIS_Extensions_Upgrade . . . 474
7.21.2 PostGIS_Full_Version . . . 475
7.21.3 PostGIS_GEOS_Version . . . 476
7.21.4 PostGIS_GEOS_Compiled_Version . . . 476
7.21.5 PostGIS_Liblwgeom_Version . . . 477
7.21.6 PostGIS_LibXML_Version . . . 477
7.21.7 PostGIS_LibJSON_Version . . . 478
7.21.8 PostGIS_Lib_Build_Date . . . 478
7.21.9 PostGIS_Lib_Version . . . 479
7.21.10 PostGIS_PROJ_Version . . . 479
7.21.11 PostGIS_PROJ_Compiled_Version . . . 480
7.21.12 PostGIS_Wagyu_Version . . . 480
7.21.13 PostGIS_Scripts_Build_Date . . . 481
7.21.14 PostGIS_Scripts_Installed . . . 481
7.21.15 PostGIS_Scripts_Released . . . 482
7.21.16 PostGIS_Version . . . 483
7.22 Variables PostGIS GUC (Grand Unified Custom Variables) . . . 483
7.22.1 postgis.gdal_datapath . . . 483
7.22.2 postgis.gdal_enabled_drivers . . . 484
7.22.3 postgis.enable_outdb_rasters . . . 485
7.22.4 postgis.gdal_vsi_options . . . 486
7.23 Fonctions de dépannage . . . 487
7.23.1 PostGIS_AddBBox . . . 487
7.23.2 PostGIS_DropBBox . . . 487
7.23.3 PostGIS_HasBBox . . . 488

8 Référence des fonctions SFCGAL **490**

8.1 Fonctions de gestion de SFCGAL . . . 490

8.1.1 postgis_sfcgal_version . . . 490

8.1.2 postgis_sfcgal_full_version . . . 490

8.2 Fonctions d'accès et de modifications SFCGAL . . . 491

8.2.1 CG_ForceLHR . . . 491

8.2.2 CG_IsPlanar . . . 491

8.2.3 CG_IsSolid . . . 492

8.2.4 CG_MakeSolid . . . 492

8.2.5 CG_Orientation . . . 492

8.2.6 CG_Area . . . 493

8.2.7 CG_3DArea . . . 493

8.2.8 CG_Volume . . . 494

8.2.9 ST_ForceLHR . . . 495

8.2.10 ST_IsPlanar . . . 496

8.2.11 ST_IsSolid . . . 496

8.2.12 ST_MakeSolid . . . 497

8.2.13 ST_Orientation . . . 497

8.2.14 ST_3DArea . . . 498

8.2.15 ST_Volume . . . 499

8.3 Fonctions de traitement et de relation SFCGAL . . . 500

8.3.1 CG_Intersection . . . 500

8.3.2 CG_Intersects . . . 500

8.3.3 CG_3DIntersects . . . 501

8.3.4 CG_Difference . . . 502

8.3.5 ST_3DDifference . . . 503

8.3.6 CG_3DDifference . . . 503

8.3.7 CG_Distance . . . 504

8.3.8 CG_3DDistance . . . 505

8.3.9 ST_3DConvexHull . . . 506

8.3.10 CG_3DConvexHull . . . 506

8.3.11 ST_3DIntersection . . . 507

8.3.12 CG_3DIntersection . . . 508

8.3.13 CG_Union . . . 510

8.3.14 ST_3DUnion . . . 511

8.3.15 CG_3DUnion . . . 511

8.3.16 ST_AlphaShape . . . 512

8.3.17 CG_AlphaShape . . . 513

8.3.18 CG_ApproxConvexPartition . . . 516

8.3.19 ST_ApproximateMedialAxis . . . 517
8.3.20 CG_ApproximateMedialAxis . . . 518
8.3.21 ST_ConstrainedDelaunayTriangles . . . 519
8.3.22 CG_ConstrainedDelaunayTriangles . . . 519
8.3.23 ST_Extrude . . . 521
8.3.24 CG_Extrude . . . 521
8.3.25 CG_ExtrudeStraightSkeleton . . . 523
8.3.26 CG_GreeneApproxConvexPartition . . . 524
8.3.27 ST_MinkowskiSum . . . 525
8.3.28 CG_MinkowskiSum . . . 525
8.3.29 ST_OptimalAlphaShape . . . 527
8.3.30 CG_OptimalAlphaShape . . . 528
8.3.31 CG_OptimalConvexPartition . . . 530
8.3.32 CG_StraightSkeleton . . . 531
8.3.33 ST_StraightSkeleton . . . 532
8.3.34 ST_Tesselate . . . 534
8.3.35 CG_Tesselate . . . 534
8.3.36 CG_Triangulate . . . 536
8.3.37 CG_Visibility . . . 537
8.3.38 CG_YMonotonePartition . . . 538

9 Topologie **540**

9.1 Les types associés à "Topology" . . . 540
9.1.1 getfaceedges_returntype . . . 540
9.1.2 TopoGeometry . . . 541
9.1.3 validatetopology_returntype . . . 541
9.2 Domaines de topologie . . . 542
9.2.1 TopoElement . . . 542
9.2.2 TopoElementArray . . . 542
9.3 Gestion de la topologie et de TopoGeometry . . . 543
9.3.1 AddTopoGeometryColumn . . . 543
9.3.2 RenameTopoGeometryColumn . . . 544
9.3.3 DropTopology . . . 545
9.3.4 RenameTopology . . . 545
9.3.5 DropTopoGeometryColumn . . . 546
9.3.6 Populate_Topology_Layer . . . 546
9.3.7 TopologySummary . . . 547
9.3.8 ValidateTopology . . . 548
9.3.9 ValidateTopologyRelation . . . 550

9.3.10 FindTopology . . . 550
9.3.11 FindLayer . . . 551
9.4 Gestion des statistiques de topologie . . . 551
9.5 Constructeurs de topologie . . . 552
9.5.1 CreateTopology . . . 552
9.5.2 CopyTopology . . . 553
9.5.3 ST_InitTopoGeo . . . 553
9.5.4 ST_CreateTopoGeo . . . 554
9.5.5 TopoGeo_AddPoint . . . 555
9.5.6 TopoGeo_AddLineString . . . 555
9.5.7 TopoGeo_AddPolygon . . . 556
9.5.8 TopoGeo_LoadGeometry . . . 556
9.6 Éditeurs de topologie . . . 557
9.6.1 ST_AddIsoNode . . . 557
9.6.2 ST_AddIsoEdge . . . 557
9.6.3 ST_AddEdgeNewFaces . . . 558
9.6.4 ST_AddEdgeModFace . . . 558
9.6.5 ST_RemEdgeNewFace . . . 559
9.6.6 ST_RemEdgeModFace . . . 560
9.6.7 ST_ChangeEdgeGeom . . . 560
9.6.8 ST_ModEdgeSplit . . . 561
9.6.9 ST_ModEdgeHeal . . . 562
9.6.10 ST_NewEdgeHeal . . . 562
9.6.11 ST_MoveIsoNode . . . 563
9.6.12 ST_NewEdgesSplit . . . 564
9.6.13 ST_RemoveIsoNode . . . 564
9.6.14 ST_RemoveIsoEdge . . . 565
9.7 Accès à la topologie . . . 566
9.7.1 GetEdgeByPoint . . . 566
9.7.2 GetFaceByPoint . . . 567
9.7.3 GetFaceContainingPoint . . . 567
9.7.4 GetNodeByPoint . . . 568
9.7.5 GetTopologyID . . . 569
9.7.6 GetTopologySRID . . . 569
9.7.7 GetTopologyName . . . 570
9.7.8 ST_GetFaceEdges . . . 570
9.7.9 ST_GetFaceGeometry . . . 571
9.7.10 GetRingEdges . . . 572
9.7.11 GetNodeEdges . . . 572

9.8 Traitement de la topologie . . . 573
9.8.1 Polygonize . . . 573
9.8.2 AddNode . . . 573
9.8.3 AddEdge . . . 574
9.8.4 AddFace . . . 575
9.8.5 ST_Simplify . . . 577
9.8.6 RemoveUnusedPrimitives . . . 577
9.9 Constructeurs de TopoGeometry . . . 578
9.9.1 CreateTopoGeom . . . 578
9.9.2 toTopoGeom . . . 580
9.9.3 TopoElementArray_Agg . . . 581
9.9.4 TopoElement . . . 581
9.10 Editeurs de TopoGeometry . . . 582
9.10.1 clearTopoGeom . . . 582
9.10.2 TopoGeom_addElement . . . 583
9.10.3 TopoGeom_remElement . . . 583
9.10.4 TopoGeom_addTopoGeom . . . 584
9.10.5 toTopoGeom . . . 584
9.11 Accès aux TopoGeometry . . . 584
9.11.1 GetTopoGeomElementArray . . . 584
9.11.2 GetTopoGeomElements . . . 585
9.11.3 ST_SRID . . . 585
9.12 Sorties TopoGeometry . . . 586
9.12.1 AsGML . . . 586
9.12.2 AsTopoJSON . . . 588
9.13 Relations spatiales de topologie . . . 590
9.13.1 Equals . . . 590
9.13.2 Intersects . . . 591
9.14 Importer et exporter des topologies . . . 591
9.14.1 Utiliser l'exportateur de topologie . . . 592
9.14.2 Utiliser l'importateur de topologie . . . 592

10 Gestion des données raster, requêtes et applications 593
10.1 Chargement et création de rasters . . . 593
10.1.1 Utilisation de raster2pgsql pour charger des rasters . . . 593
10.1.1.1 Exemple d'utilisation . . . 593
10.1.1.2 options raster2pgsql . . . 594
10.1.2 Création de rasters à l'aide des fonctions raster de PostGIS . . . 596
10.1.3 Utilisation de rasters "out db" stockés sur le cloud . . . 596

10.2 Catalogues Raster . . . 597
10.2.1 Catalogue des colonnes raster . . . 597
10.2.2 Aperçu des données raster . . . 598
10.3 Créer des applications personnalisées avec PostGIS Raster . . . 599
10.3.1 Exemple de sortie PHP utilisant ST_AsPNG avec d'autres fonctions raster . . . 599
10.3.2 Exemple ASP.NET C# Sortie utilisant ST_AsPNG en conjonction avec d'autres fonctions raster . . . 600
10.3.3 Application console Java qui produit une requête raster sous forme de fichier image . . . 602
10.3.4 Utiliser PLPython pour extraire des images via SQL . . . 603
10.3.5 Sortie de données raster avec PSQL . . . 603

11 Référence Raster **605**

11.1 Types de données pour la prise en charge raster . . . 606
11.1.1 geomval . . . 606
11.1.2 addbandarg . . . 606
11.1.3 rastbandarg . . . 606
11.1.4 raster . . . 607
11.1.5 reclassarg . . . 607
11.1.6 summarystats . . . 608
11.1.7 unionarg . . . 608
11.2 Gestion raster . . . 609
11.2.1 AddRasterConstraints . . . 609
11.2.2 DropRasterConstraints . . . 611
11.2.3 AddOverviewConstraints . . . 612
11.2.4 DropOverviewConstraints . . . 613
11.2.5 PostGIS_GDAL_Version . . . 613
11.2.6 PostGIS_Raster_Lib_Build_Date . . . 613
11.2.7 PostGIS_Raster_Lib_Version . . . 614
11.2.8 ST_GDALDrivers . . . 614
11.2.9 ST_Contour . . . 619
11.2.10 ST_InterpolateRaster . . . 620
11.2.11 UpdateRasterSRID . . . 621
11.2.12 ST_CreateOverview . . . 621
11.3 Constructeurs de raster . . . 622
11.3.1 ST_AddBand . . . 622
11.3.2 ST_AsRaster . . . 625
11.3.3 ST_Band . . . 627
11.3.4 ST_MakeEmptyCoverage . . . 628
11.3.5 ST_MakeEmptyRaster . . . 630
11.3.6 ST_Tile . . . 631

11.3.7 ST_Retile . . . 633
11.3.8 ST_FromGDALRaster . . . 633
11.4 Fonctions d'accès aux rasters . . . 634
11.4.1 ST_GeoReference . . . 634
11.4.2 ST_Height . . . 635
11.4.3 ST_IsEmpty . . . 636
11.4.4 ST_MemSize . . . 636
11.4.5 ST_MetaData . . . 637
11.4.6 ST_NumBands . . . 637
11.4.7 ST_PixelHeight . . . 638
11.4.8 ST_PixelWidth . . . 639
11.4.9 ST_ScaleX . . . 640
11.4.10 ST_ScaleY . . . 641
11.4.11 ST_RasterToWorldCoord . . . 641
11.4.12 ST_RasterToWorldCoordX . . . 642
11.4.13 ST_RasterToWorldCoordY . . . 643
11.4.14 ST_Rotation . . . 644
11.4.15 ST_SkewX . . . 645
11.4.16 ST_SkewY . . . 645
11.4.17 ST_SRID . . . 646
11.4.18 ST_Summary . . . 647
11.4.19 ST_UpperLeftX . . . 647
11.4.20 ST_UpperLeftY . . . 648
11.4.21 ST_Width . . . 648
11.4.22 ST_WorldToRasterCoord . . . 649
11.4.23 ST_WorldToRasterCoordX . . . 650
11.4.24 ST_WorldToRasterCoordY . . . 650
11.5 Fonctions d'accès aux bandes raster . . . 651
11.5.1 ST_BandMetaData . . . 651
11.5.2 ST_BandNoDataValue . . . 652
11.5.3 ST_BandIsNoData . . . 653
11.5.4 ST_BandPath . . . 654
11.5.5 ST_BandFileSize . . . 655
11.5.6 ST_BandFileTimestamp . . . 655
11.5.7 ST_BandPixelType . . . 656
11.5.8 ST_MinPossibleValue . . . 657
11.5.9 ST_HasNoBand . . . 657
11.6 Fonctions d'accès et de modifications des pixels raster . . . 658
11.6.1 ST_PixelAsPolygon . . . 658

11.6.2 ST_PixelAsPolygons . . . 658
11.6.3 ST_PixelAsPoint . . . 659
11.6.4 ST_PixelAsPoints . . . 660
11.6.5 ST_PixelAsCentroid . . . 661
11.6.6 ST_PixelAsCentroids . . . 662
11.6.7 ST_Value . . . 663
11.6.8 ST_NearestValue . . . 666
11.6.9 ST_SetZ . . . 668
11.6.10 ST_SetM . . . 669
11.6.11 ST_Neighborhood . . . 670
11.6.12 ST_SetValue . . . 672
11.6.13 ST_SetValues . . . 673
11.6.14 ST_DumpValues . . . 681
11.6.15 ST_PixelOfValue . . . 682
11.7 Éditeurs de raster . . . 683
11.7.1 ST_SetGeoReference . . . 683
11.7.2 ST_SetRotation . . . 684
11.7.3 ST_SetScale . . . 685
11.7.4 ST_SetSkew . . . 686
11.7.5 ST_SetSRID . . . 687
11.7.6 ST_SetUpperLeft . . . 687
11.7.7 ST_Resample . . . 688
11.7.8 ST_Rescale . . . 689
11.7.9 ST_Reskew . . . 691
11.7.10 ST_SnapToGrid . . . 692
11.7.11 ST_Resize . . . 693
11.7.12 ST_Transform . . . 694
11.8 Éditeurs de bandes raster . . . 697
11.8.1 ST_SetBandNoDataValue . . . 697
11.8.2 ST_SetBandIsNoData . . . 698
11.8.3 ST_SetBandPath . . . 699
11.8.4 ST_SetBandIndex . . . 701
11.9 Statistiques et analyses des bandes raster . . . 702
11.9.1 ST_Count . . . 702
11.9.2 ST_CountAgg . . . 703
11.9.3 ST_Histogram . . . 704
11.9.4 ST_Quantile . . . 706
11.9.5 ST_SummaryStats . . . 708
11.9.6 ST_SummaryStatsAgg . . . 710

11.9.7 ST_ValueCount . . . 711
11.10 Import de raster . . . 713
11.10.1 ST_RastFromWKB . . . 713
11.10.2 ST_RastFromHexWKB . . . 714
11.11 Export de raster . . . 715
11.11.1 ST_AsBinary/ST_AsWKB . . . 715
11.11.2 ST_AsHexWKB . . . 715
11.11.3 ST_AsGDALRaster . . . 716
11.11.4 ST_AsJPEG . . . 717
11.11.5 ST_AsPNG . . . 718
11.11.6 ST_AsTIFF . . . 719
11.12 Traitement des données raster : algèbre cartographique . . . 720
11.12.1 ST_Clip . . . 720
11.12.2 ST_ColorMap . . . 724
11.12.3 ST_Grayscale . . . 727
11.12.4 ST_Intersection . . . 729
11.12.5 ST_MapAlgebra (callback function version) . . . 730
11.12.6 ST_MapAlgebra (expression version) . . . 737
11.12.7 ST_MapAlgebraExpr . . . 739
11.12.8 ST_MapAlgebraExpr . . . 741
11.12.9 ST_MapAlgebraFct . . . 746
11.12.10 ST_MapAlgebraFct . . . 750
11.12.11 ST_MapAlgebraFctNgb . . . 754
11.12.12 ST_Reclass . . . 756
11.12.13 ST_Union . . . 758
11.13 Fonctions de rappel intégrées d'algèbre cartographique . . . 759
11.13.1 ST_Distinct4ma . . . 759
11.13.2 ST_InvDistWeight4ma . . . 760
11.13.3 ST_Max4ma . . . 761
11.13.4 ST_Mean4ma . . . 762
11.13.5 ST_Min4ma . . . 763
11.13.6 ST_MinDist4ma . . . 764
11.13.7 ST_Range4ma . . . 765
11.13.8 ST_StdDev4ma . . . 766
11.13.9 ST_Sum4ma . . . 767
11.14 Traitement des données raster : MNT (élévation) . . . 768
11.14.1 ST_Aspect . . . 768
11.14.2 ST_HillShade . . . 770
11.14.3 ST_Roughness . . . 772

11.14.4 ST_Slope . . . 772
11.14.5 ST_TPI . . . 774
11.14.6 ST_TRI . . . 775
11.15 Traitement des données raster : raster vers géométrie . . . 775
11.15.1 Box3D . . . 775
11.15.2 ST_ConvexHull . . . 776
11.15.3 ST_DumpAsPolygons . . . 777
11.15.4 ST_Envelope . . . 778
11.15.5 ST_MinConvexHull . . . 779
11.15.6 ST_Polygon . . . 780
11.16 Opérateurs raster . . . 782
11.16.1 && . . . 782
11.16.2 &< . . . 782
11.16.3 &> . . . 783
11.16.4 = . . . 784
11.16.5 @ . . . 784
11.16.6 ~= . . . 785
11.16.7 ~ . . . 785
11.17 Relations spatiales entre raster et entre bandes raster . . . 786
11.17.1 ST_Contains . . . 786
11.17.2 ST_ContainsProperly . . . 787
11.17.3 ST_Covers . . . 788
11.17.4 ST_CoveredBy . . . 788
11.17.5 ST_Disjoint . . . 789
11.17.6 ST_Intersects . . . 790
11.17.7 ST_Overlaps . . . 791
11.17.8 ST_Touches . . . 792
11.17.9 ST_SameAlignment . . . 793
11.17.10 ST_NotSameAlignmentReason . . . 794
11.17.11 ST_Within . . . 795
11.17.12 ST_DWithin . . . 796
11.17.13 ST_DFullyWithin . . . 796
11.18 Astuces raster . . . 797
11.18.1 Rasters out-DB . . . 797
11.18.1.1 Répertoire contenant de nombreux fichiers . . . 797
11.18.1.2 Nombre maximum de fichiers ouverts . . . 798
11.18.1.2.1 Nombre maximal de fichiers ouverts pour l'ensemble du système . . . 798
11.18.1.2.2 Nombre maximal de fichiers ouverts par processus . . . 799

12 PostGIS Extras **801**

12.1 Address Standardizer . . . 801
12.1.1 Fonctionnement de l'analyseur . . . 801
12.1.2 Types de normalisateurs d'adresses . . . 802
12.1.2.1 stdaddr . . . 802
12.1.3 Tables Address Standardizer . . . 802
12.1.3.1 rules table . . . 802
12.1.3.2 lex table . . . 805
12.1.3.3 gaz table . . . 805
12.1.4 Fonctions Address Standardizer . . . 806
12.1.4.1 debug_standardize_address . . . 806
12.1.4.2 parse_address . . . 807
12.1.4.3 standardize_address . . . 808
12.2 Géocodeur Tiger . . . 810
12.2.1 Drop_Indexes_Generate_Script . . . 810
12.2.2 Drop_Nation_Tables_Generate_Script . . . 811
12.2.3 Drop_State_Tables_Generate_Script . . . 812
12.2.4 Geocode . . . 813
12.2.5 Geocode_Intersection . . . 815
12.2.6 Get_Geocode_Setting . . . 816
12.2.7 Get_Tract . . . 817
12.2.8 Install_Missing_Indexes . . . 818
12.2.9 Loader_Generate_Census_Script . . . 818
12.2.10 Loader_Generate_Script . . . 820
12.2.11 Loader_Generate_Nation_Script . . . 822
12.2.12 Missing_Indexes_Generate_Script . . . 823
12.2.13 Normalize_Address . . . 824
12.2.14 Pagc_Normalize_Address . . . 826
12.2.15 Pprint_Addy . . . 827
12.2.16 Reverse_Geocode . . . 828
12.2.17 Topology_Load_Tiger . . . 830
12.2.18 Set_Geocode_Setting . . . 832

13 Index des fonctions spéciales de PostGIS **834**

13.1 Fonctions d'agrégation de PostGIS . . . 834
13.2 Fonctions Window PostGIS . . . 835
13.3 Fonctions de PostGIS compatibles avec SQL-MM . . . 835
13.4 Fonctions d'aide au type geography de PostGIS . . . 839
13.5 Fonctions de support des données raster de PostGIS . . . 840

13.6 Fonctions PostGIS de dump Geometry / Geography / Raster . . . 846
13.7 Fonctions Box de PostGIS . . . 846
13.8 Fonctions PostGIS supportant la 3D . . . 848
13.9 Fonctions d'aide aux géométries courbes de PostGIS . . . 853
13.10 Fonctions de support des surfaces polyédriques de PostGIS . . . 856
13.11 Matrice d'aide aux fonctions de PostGIS . . . 860
13.12 Fonctions PostGIS nouvelles, améliorées ou modifiées . . . 878
13.12.1 Fonctions PostGIS nouvelles ou améliorées en 3.5 . . . 878
13.12.2 Fonctions PostGIS nouvelles ou améliorées en 3.4 . . . 879
13.12.3 Fonctions PostGIS nouvelles ou améliorées en 3.3 . . . 881
13.12.4 Fonctions PostGIS nouvelles ou améliorées en 3.2 . . . 882
13.12.5 Fonctions PostGIS nouvelles ou améliorées en 3.1 . . . 883
13.12.6 Fonctions PostGIS nouvelles ou améliorées en 3.0 . . . 885
13.12.7 Fonctions PostGIS nouvelles ou améliorées en 2.5 . . . 886
13.12.8 Fonctions PostGIS nouvelles ou améliorées en 2.4 . . . 888
13.12.9 Fonctions PostGIS nouvelles ou améliorées en 2.3 . . . 889
13.12.10 Fonctions PostGIS nouvelles ou améliorées en 2.2 . . . 891
13.12.11 Fonctions PostGIS nouvelles ou améliorées en 2.1 . . . 894
13.12.12 Fonctions PostGIS nouvelles ou améliorées en 2.0 . . . 900
13.12.13 Fonctions PostGIS nouvelles ou améliorées en 1.5 . . . 911
13.12.14 Fonctions PostGIS nouvelles ou améliorées en 1.4 . . . 912
13.12.15 Fonctions PostGIS nouvelles ou améliorées en 1.3 . . . 913

14 Rapporter un problème **914**
14.1 Rapporter un problème logiciel . . . 914
14.2 Signaler les problèmes de documentation . . . 914

A Annexes **915**
A.1 PostGIS 3.5.3 . . . 915
A.1.1 Bug Fixes . . . 915
A.2 PostGIS 3.5.2 . . . 915
A.2.1 Bug Fixes . . . 915
A.3 PostGIS 3.5.1 . . . 916
A.3.1 Changements avec rupture . . . 916
A.3.2 Améliorations . . . 916
A.3.3 Changements avec rupture . . . 916
A.4 PostGIS 3.5.0 . . . 916
A.4.1 Changements avec rupture . . . 917
A.4.2 Deprecated signatures . . . 917
A.4.3 Nouvelles fonctionnalités . . . 917
A.4.4 Améliorations . . . 918

Abstract

PostGIS est une extension du système de base de PostgreSQL données relationnel-objet qui permet de stocker des objets SIG (Système d'Information Géographique) dans la base. PostGIS comporte un support des index spatiaux R-Tree basé sur GiST et des fonctions d'analyse et de traitement des objets SIG.

Manuel de la version 3.5.4dev

Ce travail est soumis à une licence Creative Commons Attribution-Share Alike 3.0 License. Vous pouvez utiliser ce matériel comme bon vous semble, mais nous vous demandons de mentionner le projet PostGIS et, dans la mesure du possible, d'ajouter un lien vers https://postgis.net.

Chapter 1

Introduction

PostGIS est une extension spatiale pour la base de données relationnelle PostgreSQL. Elle a été créée par Refractions Research Inc, sous la forme d'un projet de recherche sur les technologies de base de données spatiales. Refractions est une société de conseil en SIG et base de données basée à Victoria, British Columbia, Canada, et spécialisée en intégration de données et développement logiciel.

PostGIS est désormais un projet de la OSGeo Foundation. PostGIS est développé et financé par de nombreux développeurs et organismes FOSS4G dans le monde entier, qui bénéficient de manière significative de ses fonctionnalités et de sa polyvalence.

Le groupe de développement du projet PostGIS a en charge la maintenance et l'amélioration de PostGIS pour un meilleur support de fonctionnalités importantes en SIG, dans les domaines des standards OGC et SQL/MM, des constructions topologiques avancées (couvertures, surfaces, réseaux), des sources de données pour les outils graphiques de bureautique pour la visualisation et l'édition des donnée SIG, ainsi que dans le domaine des outils web.

1.1 Comité de direction du projet (Project Steering Committee)

Le comité de direction du projet (PSC) coordonne la direction générale, les cycles de publication, la documentation et les efforts spécifiques pour le projet PostGIS. De plus, le PSC fournit un support général aux utilisateurs, accepte et approuve les patches de la communauté PostGIS et vote sur divers points concernant PostGIS, tels que les accès commit pour les développeurs, les nouveaux membres du PSC ou les changements majeurs d'API.

Raúl Marín Rodríguez Support MVT, Correction de bugs, Améliorations des performances et de la stabilité, Nettoyage du GitHub, Alignement de PostGIS avec les publications PostgreSQL

Regina Obe Maintenance de l'IC (intégration continue) et du site web, production Windows et builds expérimentaux, documentation, alignement de PostGIS sur les versions de PostgreSQL, support X3D, support du géocodeur TIGER, fonctions de gestion.

Darafei Praliaskouski Amélioration de l'index, correction de bugs et amélioration des fonctions de géométrie/géographie, SFCGAL, raster, curation GitHub, et maintenance ci (intégration continue).

Paul Ramsey (Directeur) Co-fondateur de PostGIS. Maintenance générale, maintenance des Geography, maintenance des index Geography et Geometry (index 2D, 3D, nD et tout index spatial), structures internes des Geometry, intégration des fonctionnalités GEOS et alignement avec les publications GEOS, alignement de PostGIS avec les publications PostgreSQL, loader/dumper, IHM de chargement Shapefile.

Sandro Santilli Corrections de bugs et maintenance, maintenance des CI (intégration continue), gestion des miroirs git, fonctions de gestion, intégration de nouvelles fonctionnalités GEOS et alignement sur les versions GEOS, support topologique, framework raster et fonctions API de bas niveau.

1.2 Contributeurs cœur actuels

Nicklas Avén Améliorations et ajouts de fonctions de distance (notamment fonctions de distance 3D et relations), format de sortie Tiny WKB (TWKB) et support utilisateur

Loïc Bartoletti Améliorations et maintenance de SFCGAL et maintien de l'intégration continue

Dan Baston Ajout de fonctions sur le clustering de géométries, améliorations sur les algorithmes sur les géométries, améliorations GEOS et support utilisateur

Martin Davis Améliorations GEOS et documentation

Björn Harrtell Fonctions MapBox Vector Tile, GeoBuf et Flatgeobuf. Tests Gitea et expérimentation GitLab.

Aliaksandr Kalenik Traitement des géométries, PostgreSQL gist, correction de bugs

1.3 Anciens contributeurs cœur

Bborie Park Ancien membre du PSC. Développement raster, intégration avec GDAL, loader raster, support utilisateur, correction de bugs, tests sur divers systèmes d'exploitation (Slackware, Mac, Windows, et autres)

Mark Cave-Ayland Ancien membre du PSC. Coordination de l'effort de correction de bugs et de maintenance, sélectivité et liaisons des index spatiaux, loader/dumper, IHM de chargement Shapefile, intégration et amélioration de nouvelles fonctions.

Jorge Arévalo Développement Raster, support du driver GDAL, outil de chargement

Olivier Courtin (Émérite) Fonctions d'entrée/sortie XML (KML,GML) et fonctions GeoJSON, support 3D et correction de bugs.

Chris Hodgson Ancien membre du PSC. Développement général, maintenance du site et du robot de build, gestion de l'incubation OSGeo

Mateusz Loskot Support de CMake pour PostGIS, a développé le loader raster original en python et les fonctions d'API bas-niveau raster

Kevin Neufeld Ancien membre du PSC. Documentation et outils pour la documentation, maintenance du robot de build, support utilisateur avancé sur le newsgroup PostGIS, et maintenance et améliorations des fonctions PostGIS.

Dave Blasby Développeur originel et co-fondateur de PostGIS. Dave a écrit les objets côté serveur, les liaisons des index, et de nombreuses fonctions d'analyse côté serveur.

Jeff Lounsbury Développement originel de l'outil de chargement/export de shapefiles.

Mark Leslie Maintenance générale et développement de fonctions du noyau PostGIS. Amélioration du support des courbes. Interface graphique de chargement des shapefiles.

Pierre Racine Architecte de l'implémentation raster de PostGIS. Architecture globale raster, prototypage, support au développement

David Zwarg Développement raster (principalement fonctions analytiques d'algèbre cartographique)

1.4 Autres contributeurs

Contributeurs individuels

Alex Bodnaru
Alex Mayrhofer
Andrea Peri
Andreas Forø Tollefsen
Andreas Neumann
Andrew Gierth
Anne Ghisla
Antoine Bajolet
Arthur Lesuisse
Artur Zakirov
Barbara Phillipot
Ben Jubb
Bernhard Reiter
Björn Esser
Brian Hamlin
Bruce Rindahl
Bruno Wolff III
Bryce L. Nordgren
Carl Anderson
Charlie Savage
Chris Mayo
Christian Schroeder
Christoph Berg
Christoph Moench-Tegeder
Dane Springmeyer
Dapeng Wang
Daryl Herzmann
Dave Fuhry
David Garnier
David Skea
David Techer
Dian M Fay
Dmitry Vasilyev
Eduin Carrillo
Esteban Zimanyi
Eugene Antimirov
Even Rouault
Florian Weimer
Frank Warmerdam
George Silva
Gerald Fenoy
Gino Lucrezi
Greg Troxel
Guillaume Lelarge
Giuseppe Broccolo
Han Wang
Hans Lemuet
Haribabu Kommi
Havard Tveite
IIDA Tetsushi
Ingvild Nystuen
Jackie Leng
James Addison
James Marca
Jan Katins
Jan Tojnar
Jason Smith
Jeff Adams
Jelte Fennema
Jim Jones
Joe Conway
Jonne Savolainen
Jose Carlos Martinez Llari
Jörg Habenicht
Julien Rouhaud
Kashif Rasul
Klaus Foerster
Kris Jurka
Laurențiu Nicola
Laurenz Albe
Lars Roessiger
Leo Hsu
Loic Dachary
Luca S. Percich
Lucas C. Villa Real
Maria Arias de Reyna
Marc Ducobu
Mark Sondheim
Markus Schaber
Markus Wanner
Matt Amos
Matt Bretl
Matthias Bay
Maxime Guillaud
Maxime van Noppen
Maxime Schoemans
Megan Ma
Michael Fuhr
Mike Toews
Nathan Wagner
Nathaniel Clay
Nikita Shulga
Norman Vine
Patricia Tozer
Rafal Magda
Ralph Mason
Rémi Cura
Richard Greenwood
Robert Coup
Roger Crew
Ron Mayer
Sam Peters
Sebastiaan Couwenberg
Sergei Shoulbakov
Sergey Fedoseev
Shinichi Sugiyama
Shoaib Burq
Silvio Grosso
Stefan Corneliu Petrea
Steffen Macke
Stepan Kuzmin
Stephen Frost
Steven Ottens
Talha Rizwan
Teramoto Ikuhiro
Tom Glancy
Tom van Tilburg
Victor Collod
Vincent Bre
Vincent Mora
Vincent Picavet
Volf Tomáš
Zuo Chenwei

Organisations sponsors Certaines organisations ont contribué du temps de développeur, de l'hébergement, ou du financement direct pour le projet PostGIS. Par ordre alphabétique :

- 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
- HighGo
- Hunter Systems Group
- >The National Institute for Agricultural and Food Research and Technology (INIA-CSIC)
- ISciences, LLC
- Kontur
- Lidwala Consulting Engineers
- LISAsoft
- Logical Tracking & Tracing International AG
- Maponics
- Michigan Tech Research Institute
- Natural Resources Canada
- Norwegian Forest and Landscape Institute
- Norwegian Institute of Bioeconomy Research (NIBIO)
- OSGeo
- Oslandia
- Palantir Technologies
- Paragon Corporation
- R3 GIS
- Refractions Research
- Regione Toscana - SITA
- Safe Software
- Sirius Corporation plc
- Stadt Uster
- UC Davis Center for Vectorborne Diseases
- Université Laval
- U.S. Department of State (HIU)
- Zonar Systems

Campagnes de financement participatif Les campagnes de financement participatif sont des campagnes que nous organisons pour financer des fonctionnalités très demandées qui peuvent servir à une large communauté. Chaque campagne est centrée sur une fonctionnalité, ou un lot de fonctionnalités. Chaque sponsor apporte une petite fraction du financement nécessaire, et avec assez de personnes et d'organisations contribuant, nous obtenons les fonds nécessaires pour payer le travail qui aidera la communauté. Si vous pensez qu'une fonctionnalité pourrait être co-financée par de nombreux acteurs, vous pouvez poster votre idée sur le newsgroup PostGIS et ensemble nous pourrons la réaliser.

PostGIS 2.0.0 a été la première version où cette stratégie a été testée. Nous avons utilisé PledgeBank et avons eu deux campagnes de financement réussies.

postgistopology - Plus de 10 sponsors ont contribué chacun 250USD pour créer la fonction toTopoGeometry et sortir le support de la topologie dans la version 2.0.0. Ce fut une réussite.

postgis64windows - Plus de 20 sponsors ont contribué chacun 100USD pour le support PostGIS 64-bit sur Windows. Ce fut une réussite.

Bibliothèques de base importantes GEOS, la bibliothèque pour les opérations sur les géometries

GDAL (Geospatial Data Abstraction Library), la bibliothèque utilisée pour propulser la plupart des fonctionnalités raster introduites dans PostGIS 2. En retour, les améliorations requises dans GDAL pour supporter PostGIS sont reversées dans le projet GDAL.

PROJ, la bibliothèque gérant les projections cartographiques

Enfin, mais non des moindres, le projet de SGBD PostgreSQL, géant sur les épaules duquel PostGIS s'appuie. La rapidité et flexibilité de PostGIS serait impossible sans l'extensibilité, le planificateur de requêtes, les index GiST et les nombreuses fonctionnalités SQL que fournit PostgreSQL.

Chapter 2

Installation de PostGIS

Ce chapitre décrit les étapes nécessaires pour installer PostGIS.

2.1 Version courte

Pour compiler, en supposant que toutes les dépendances soient dans votre chemin de recherche :

```
tar -xvzf postgis-3.5.4dev.tar.gz
cd postgis-3.5.4dev
./configure
make
make install
```

Une fois PostGIS installé, il est nécessaire de l'activer (Section 3.3) ou de le mettre à jour (Section 3.4) dans chaque base de données où vous voulez l'utiliser.

2.2 Compilation et installation depuis les sources

Note

La plupart des systèmes d'exploitation dispose de paquets pré-compilés de PostgreSQL/PostGIS. La compilation est réellement nécessaire uniquement pour disposer des toutes dernières fonctionnalités ou pour les responsables de paquets PostGIS.

Cette section décrit le processus générique de compilation. Si vous compilez pour Windows ou d'autres systèmes d'exploitation, vous trouverez des instructions plus détaillées dans les wiki de la communauté guides de compilation et PostGIS Dev Wiki.

Les paquets pré-compilés pour de multiples systèmes d'exploitation sont listés dans PostGIS Pre-built Packages

Pour les utilisateurs Windows, des versions stables sont disponibles sur Stackbuilder ou sur le site de téléchargement PostGIS Windows. Des compilations expérimentales incluant les dernières fonctionnalités sont disponibles. Ces compilations sont généralement mises à jour une ou deux fois par semaines, ou chaque fois qu'une nouvelle fonctionnalité intéressante est ajoutée. Vous pouvez les utiliser pour suivre l'avancée des versions de PostGIS

Le module PostGIS est une extensions pour le serveur backend PostgreSQL. Par conséquent, PostGIS 3.5.4dev *nécessite* l'ensemble des entêtes du serveur PostgreSQL pour pouvoir être compilé. PostGIS peut être compilé pour les versions PostgreSQL 12 à 17. Les versions précédentes de PostgreSQL ne sont *pas* supportées.

Référez-vous aux guides d'installation de PostgreSQL si vous ne l'avez pas encore installé : https://www.postgresql.org .

Note

Pour les fonctionnalités de GEOS, quand vous installez PostgreSQL, vous aurez peut-être besoin de lier explicitement PostgreSQL avec la bibliothèque standard C++ :

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

Ceci est une solution de contournement d'exceptions C++ d'interactions bugués dans des outils de développements plus ancien. Si vous tombez sur ce genre de problèmes (backend soudainement fermé ou des choses similaires) essayez cette astuce. Cela nécessite de recompiler votre PostgreSQL du début, bien sur.

Les étapes suivantes résument la configuration et la compilation des sources PostGIS. Elles ont été rédigées pour les utilisateurs sous Linux et ne fonctionneront pas pour Windows et Mac.

2.2.1 Obtenir les Sources

Les sources de PostGIS sont disponible depuis https://postgis.net/stuff/postgis-3.5.4dev.tar.gz

```
wget https://postgis.net/stuff/postgis-3.5.4dev.tar.gz
tar -xvzf postgis-3.5.4dev.tar.gz
cd postgis-3.5.4dev
```

Un répertoire appelé `postgis-3.5.4dev` sera créé dans le répertoire courant.

Les sources peuvent également être obtenues depuis le dépôt git https://git.osgeo.org/gitea/postgis/postgis/ .

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

Se placer dans le nouveau répertoire créé `postgis-3.5.4dev` pour poursuivre l'installation.

```
./configure
```

2.2.2 Pré requis à l'installation

La compilation et l'utilisation de PostGIS nécessitent les éléments suivants :

Obligatoire

- PostgreSQL 12 - 17. Une installation complète de PostgreSQL (y compris les en-têtes du serveur) est nécessaire. PostgreSQL est disponible à partir de https://www.postgresql.org .

  Pour une matrice complète de compatibilité PostgreSQL/PostGIS et PostGIS/GEOS, référez-vous à https://trac.osgeo.org/postgis/wiki/UsersWikiPostgreSQLPostGIS

- Un compilateur GNUC C (`gcc`). D'autres compilateurs ANSI C peuvent être utilisés, mais la compilation avec `gcc` est source de moins de problèmes.

- GNU Make (`gmake` ou `make`). Sur beaucoup de systemes, GNU `make` est la version par défaut de make. Vous pouvez vérifier la version de make avec la commande `make -v`. D'autres versions de `make` peuvent ne pas être compatibles avec le `Makefile` de PostGIS.

- Proj, bibliothèque de reprojection. Proj 6.1 ou supérieur est nécessaire. La bibliothèque Proj est utilisée pour fournir le support de reprojection de coordonnées dans PostGIS. Proj est disponible au téléchargement depuis https://proj.org/ .

- La bibliothèque de géométrie GEOS, version 3.8.0 ou supérieure, mais GEOS 3.12+ est nécessaire pour profiter pleinement de toutes les nouvelles fonctions et caractéristiques. GEOS peut être téléchargé à partir de https://libgeos.org .

- LibXML2, version 2.5.x ou supérieur. LibXML2 est actuellement utilisée dans certaines fonctions d'import (ST_GeomFromGML et ST_GeomFromKML). LibXML2 est disponible au téléchargement depuis https://gitlab.gnome.org/GNOME/libxml2/-/releases.
- JSON-C, version 0.9 ou supérieure. JSON-C est utilisé pour l'import GeoJSON avec la fonction ST_GeomFromGeoJson. JSON-C est disponible depuis https://github.com/json-c/json-c/releases/.
- GDAL, version 3+ is preferred. This is required for raster support. https://gdal.org/download.html.
- Si vous compilez avec PostgreSQL+JIT, LLVM version >=6 est nécessaire https://trac.osgeo.org/postgis/ticket/4125.

Optionnel

- GDAL (pseudo optionnel) vous pouvez l'omettre seulement si vous ne voulez pas du support raster. Pensez également à activer les pilotes que vous souhaitez utiliser, comme décrit dans Section 3.2.
- GTK (GTK+2.0, 2.8+) pour compiler le chargeur de shape file shp2pgsql-gui. http://www.gtk.org/ .
- SFCGAL, 1.4.1 or higher is required and 1.5.0+ 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 Chapter 8. And also allow to use SFCGAL rather than GEOS for some 2D functions provided by both backends (like ST_Intersection or ST_Area, for instance). A PostgreSQL configuration variable `postgis.backend` allow end user to control which backend he want to use if SFCGAL is installed (GEOS by default). Nota: SFCGAL 1.2 require at least CGAL 4.3 and Boost 1.54 (cf: https://sfcgal.org) https://gitlab.com/sfcgal/SFCGAL/.
- Pour compiler Section 12.1, vous aurez également besoin de PCRE http://www.pcre.org (qui est généralement déjà installé sur les systèmes nix). Section 12.1 sera automatiquement compilé s'il détecte une bibliothèque PCRE, ou si vous passez un `--with-pcre-dir=/path/to/pcre` valide pendant la configuration.
- Pour permettre d'utiliser ST_AsMVT, la bibliothèque protobuf-c 1.1.0 ou supérieur (pour l'utilisation) et le compilateur protoc-c (pour la compilation) sont nécessaires. Également, pkg-config est nécessaire pour vérifier la version minimale correcte de protobuf-c. Voir protobuf-c. Par défaut, PostGIS utilise Wagyu pour valider les polygones MVT plus rapidement, ce qui nécessite un compilateur c++11. Ceci utilisera les CXXFLAGS et le même compilateur que l'installation PostgreSQL. Pour désactiver cela et utiliser GEOS à la place, utilisez la variable `--without-wagyu` pendant l'étape de configuration.
- CUnit (`CUnit`). Nécessaire pour les tests de régression. http://cunit.sourceforge.net/
- DocBook (`xsltproc`) est nécessaire pour générer la documentation. Docbook est disponible depuis le site http://www.docbook.org/ .
- DBLatex (`dblatex`) est nécessaire pour générer la documentation au format PDF. DBLatex est disponible depuis http://dblatex.source .
- ImageMagick (`convert`) est nécessaire pour générer les images de la documentation. ImageMagick is available from http://www.imagemagick.org/ .

2.2.3 Configuration de la compilation

Comme pour la plupart des installations linux, la première étape est de générer le Makefile qui sera utilisé pour compiler le code source. Ceci est réalisée en lançant le script shell

./configure

Sans paramètre supplémentaire, cette commande tentera de localiser automatiquement les composants requis et les bibliothèques nécessaires à la compilation de PostGIS. Bien que cela soit l'utilisation la plus commune de la commande **./configure**, vous pouvez également ajouter différents paramètres à ce script. Par exemple, vous pouvez définir l'emplacement de bibliothèques ou de programmes si ceux-ci ne sont pas localisés à un emplacement standard.

La liste suivante présente les options les plus courantes. Pour consulter la liste complète utilisez l'option **--help** ou **--help=short**.

--with-library-minor-version À partir de PostGIS 3.0, les fichiers de bibliothèque générés par défaut n'auront plus la version mineur dans le nom de fichier. Ceci signifie que les libs PostGIS 3 se termineront par `postgis-3`. Ceci a été fait pour faciliter l'usage de pg_upgrade, avec la contrainte que vous ne pouvez installer qu'une seule version de PostGIS 3 sur votre serveur. Pour basculer sur l'ancien comportement et avoir des fichiers qui incluent la version mineure : e.g. `postgis-3.0`, ajoutez ce paramètre dans la commande de configuration.

--prefix=PREFIX Cela correspond à l'emplacement où les exécutables et librairies de PostGIS seront installés. Par défaut, cet emplacement est le même que celui de l'installation de PostgreSQL.

Caution
Ce paramètre est actuellement défectueux : le paquet s'installe uniquement dans le répertoire d'installation de PostgreSQL. Le suivi de ce bug est disponible depuis http://trac.osgeo.org/postgis/ticket/635 .

--with-pgconfig=FILE PostgreSQL fournit l'utilitaire **pg_config** permettant aux extensions comme PostGIS de localiser le répertoire d'installation de PostgreSQL. Utiliser ce paramètre (**--with-pgconfig=/path/to/pg_config**) pour spécifier une installation particulière de PostgreSQL pour laquelle PostGIS doit être compilée.

--with-gdalconfig=FILE GDAL, une des bibliothèques requises pour le support des rasters. **gdal-config** pour permettre au logiciel de localiser le répertoire d'installation de GDAL. Utiliser ce paramètre (**--with-gdalconfig=/path/to/gdal-config**) pour spécifier un répertoire d'installation particulier de GDAL qui sera utilisé pour compiler PostGIS.

--with-geosconfig=FILE GEOS, une des bibliothèques requises, fournit un utilitaire appelé **geos-config** permettant aux logiciels de localiser le répertoire d'installation de GEOS. Utiliser ce paramètre (**--with-geosconfig=/path/to/geos-config**) pour spécifier le repertoire de GEOS qui sera utilisé pour la compilation de PostGIS.

--with-xml2config=FILE LibXML est la bibliothèque requise pour les processus GeomFromKML/GML. Elle est normalement trouvée si libxml est installé, mais si ce n'est pas le cas ou si vous souhaitez qu'une version spécifique soit utilisée, vous devrez indiquer à PostGIS un fichier confi `xml2-config` spécifique pour permettre aux installations logicielles de localiser le répertoire d'installation de LibXML. Utilisez ce paramètre (**>--with-xml2config=/path/to/xml2-config**) pour spécifier manuellement une installation LibXML particulière avec laquelle PostGIS sera construit.

--with-projdir=DIR Proj4 est la bibliothèque de reprojection nécessaire à PostGIS. Utilisez ce paramètre (**--with-projdir=/path/t**
pour spécifier le répertoire de Proj qui sera utilisé pour la compilation de PostGIS.

--with-libiconv=DIR Répertoire d'installation d'iconv.

--with-jsondir=DIR JSON-C est une bibliothèque sous licence MIT utilisée par PostGIS pour les traitements JSON (ST_GeomFromJSON par exemple). Utiliser ce paramètre (**--with-jsondir=/path/to/jsondir**) pour spécifier le répertoire de JSON-C qui sera utilisé pour la compilation de PostGIS.

--with-pcredir=DIR PCRE (Perl Compatible Regular Expression) est une bibliothèque sous licence BSD requise par l'extension address_standardizer. Utilisez ce paramètre (**--with-pcredir=/chemin/vers/pcredir**) pour spécifier un répertoire contenant PCRE qui sera utilisé par PostGIS.

--with-gui Compile l'interface graphique d'import de données (nécessite GTK+2.0). Ceci créé l'interface graphique shp2pgsql-gui à shp2pgsql.

--without-raster Compilation sans support des raster.

--without-topology Désactive le support des topologies. Il n'y a pas de bibliothèque correspondante car toute la logique requise pour les topologies est incluse dans postgis-3.5.4dev.

--with-gettext=no Par défaut PostGIS tentera de détecter la gestion de gettext et de reposer dessus pour la compilation, cependant si vous tombez sur des problèmes d'incompatibilités qui cause la cassure du chargeur, vous pouvez le désactiver entièrement avec cette commande. Référez vous au ticket http://trac.osgeo.org/postgis/ticket/748 pour un exemple de problème résolu par cette configuration. NOTE : que vous perdez beaucoup de chose en le désactivant. Cela est utilisé pour la gestion de l'aide et des labels internationaux dans le chargeur graphique qui n'est pas documenté et encore expérimental.

--with-sfcgal=PATH Par défaut, PostGIS ne contiendra pas le support sfcgal sans cet argument. `PATH` est un argument optionnel permettant de préciser un chemin alternatif vers sfcgal-config.

--without-phony-revision Désactive la mise à jour de postgis_revision.h à partir du HEAD courant du dépôt git.

Note
Si vous avez téléchargé PostGIS depuis le dépôt du code , la première étape est d'exécuter le script **./autogen.sh**
Ce script générera le script **configure** qui est utilisé pour personnaliser votre installation de PostGIS.
Si vous avez obtenu PostGIS comme archive, lancer la commande **./autogen.sh** n'est pas nécessaire puisque **configure** a déjà été généré.

2.2.4 Compiler

Une fois le Makefile généré, compiler PostGIS est aussi simple que lancer

make

La dernière ligne de la sortie doit être "`PostGIS was built successfully. Ready to install.`"

À partir de PostGIS v1.4.0, toutes les fonctions ont leur commentaire généré à partir de la documentation. Si vous souhaitez installer ces commentaires dans votre base de données spatiale plus tard, exécutez la commande suivante, qui nécessite docbook. Les fichiers de commentaires pour PostGIS postgis_comments.sql et pour les autres paquets raster_comments.sql, topology_comments.sql sont aussi inclus dans le paquet tar.gz de la distribution, dans le répertoire doc, il est donc inutile d'utiliser cette commande si vous installez depuis l'archive tar. Les commentaires sont aussi inclus par l'installation via CREATE EXTENSION.

make comments

Introduit dans PostGIS 2.0. Cela génère un mémo en html disponible pour une référence rapide ou pour les étudiants. La compilation nécessite xsltproc et génèrera 4 fichiers dans le répertoire doc `topology_cheatsheet.html`, `tiger_geocoder_cheatsheet.html`, `raster_cheatsheet.html`, `postgis_cheatsheet.html`

Vous pouvez télécharger des pré-compilations disponibles en HTML et PDF à partir de PostGIS / PostgreSQL Study Guides

make cheatsheets

2.2.5 Compiler les Extensions PostGIS et les déployer

Les extensions PostGIS sont compilées et installées automatiquement si vous utilisez PostgreSQL 9.1+.

Si vous compilez à partir des dépôts des sources, vous devez compiler les descriptions de fonction d'abord. Ceci est compilé si vous avez docbook installé. Vous pouvez également compiler manuellement avec cette commande :

make comments

Compiler les commentaires n'est pas nécessaire si vous avez compilé à partir d'une release d'archive puisque ceux-ci sont des pré-compilations packagés avec le tar ball.

Les extensions devraient être automatiquement compilées lors du make install. Vous pouvez, si nécessaire, compiler à partir des répertoires d'extensions ou copier les fichiers si vous en avez besoin sur un serveur différent.

```
cd extensions
cd postgis
make clean
make
export PGUSER=postgres #overwrite psql variables
make check #to test before install
make install
# to test extensions
make check RUNTESTFLAGS=--extension
```

Note

`make check` utilise psql pour faire tourner les tests, et peut donc utiliser les variables d'environnement psql. Les variables classiques utiles à définir sont `PGUSER`,`PGPORT`, and `PGHOST`. Voir variables d'environnement psql

Les fichiers extensions seront toujours les mêmes pour les mêmes versions de PostGIS et PostgreSQL indépendamment de l'OS, par conséquent il n'y a pas de problème à copier les fichiers extensions d'un OS à un autre du moment que vous avez les binaires PostGIS déjà installés sur vos serveurs.

Si vous voulez installer les extensions manuellement sur un serveur différent séparé de votre développement, vous devez copier les fichiers suivants à partir du répertoire extension dans le répertoire `PostgreSQL / share / extension` de votre installation PostgreSQL ainsi que les binaires nécessaires pour une version correcte de PostGIS si vous ne les avez pas déjà sur le serveur.

- Ceux-ci sont les fichiers de contrôle qui renvoie les informations telles que la version de l'extension à installer si non spécifié. `postgis.control, postgis_topology.control`.
- Tous les fichiers dans le répertoire /sql de chaque extension. Notez que ceux-ci nécessitent d'être copiées à la racine du répertoire share/extension de PostgreSQL `extensions/postgis/sql/*.sql`, `extensions/postgis_topology/sql/*.sql`

Une fois que vous avez fait cela, vous devriez voir `postgis`, `postgis_topology` comme extensions disponibles dans PgAdmin -> extensions.

Si vous utilisez psql, vous pouvez vérifier que les extensions sont installées en lançant cette requête :

```
SELECT name, default_version,installed_version
FROM pg_available_extensions WHERE name LIKE 'postgis%' or name LIKE 'address%';

             name              | default_version | installed_version
-------------------------------+-----------------+-------------------
 address_standardizer          | 3.5.4dev        | 3.5.4dev
 address_standardizer_data_us  | 3.5.4dev        | 3.5.4dev
 postgis                       | 3.5.4dev        | 3.5.4dev
 postgis_raster                | 3.5.4dev        | 3.5.4dev
 postgis_sfcgal                | 3.5.4dev        |
 postgis_tiger_geocoder        | 3.5.4dev        | 3.5.4dev
 postgis_topology              | 3.5.4dev        |
(6 rows)
```

Si vous avez l'extension installée dans la base de données que vous interrogez, vous verrez la mention dans la colonne `installed_ver`
Si vous n'obtenez aucun enregistrement , cela signifie que vous n'avez pas d'extension postgis installés sur le serveur. PgAdmin III 1.14+ fournira aussi cette information dans la section `extensions` dans l'arbre de l'explorateur de la base de données et autorisera même la mise à jour ou la désinstallation par clic-droit.

Si vous avez les extensions disponibles, vous pouvez installer les extensions postgis dans votre base de données de votre choix soit en utilisant l'interface d'extension de PgAdmin ou lançant ces commandes SQL :

```
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION postgis_sfcgal;
CREATE EXTENSION fuzzystrmatch; --needed for postgis_tiger_geocoder
--optional used by postgis_tiger_geocoder, or can be used standalone
CREATE EXTENSION address_standardizer;
CREATE EXTENSION address_standardizer_data_us;
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION postgis_topology;
```

Avec psql, vous pouvez contrôler les versions installées ainsi que les schémas d'installation.

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

```
List of installed extensions
-[ RECORD 1 ]-------------------------------------------------
Name        | postgis
Version     | 3.5.4dev
Schema      | public
Description | PostGIS geometry, geography, and raster spat..
-[ RECORD 2 ]-------------------------------------------------
Name        | postgis_raster
Version     | 3.0.0dev
Schema      | public
Description | PostGIS raster types and functions
-[ RECORD 3 ]-------------------------------------------------
Name        | postgis_tiger_geocoder
Version     | 3.5.4dev
Schema      | tiger
Description | PostGIS tiger geocoder and reverse geocoder
-[ RECORD 4 ]-------------------------------------------------
Name        | postgis_topology
Version     | 3.5.4dev
Schema      | topology
Description | PostGIS topology spatial types and functions
```

Warning
Les tables d'extension `spatial_ref_sys`, `layer`, `topology` ne peuvent pas être explicitement sauvegardées. Elles peuvent être uniquement sauvegardées quand les extensions respectives `postgis` ou `postgis_topology` sont sauvegardées, ce qui semble seulement arriver quand vous sauvegardez l'ensemble de la base. À partir de PostGIS 2.0.1, seulement les enregistrements des srid non packagés avec PostGIS sont sauvegardés quand la base de données est sauvegardée. Par conséquent, ne vous attendez pas à ce vos modifications persistent si vous changez les srids que nous fournissons. Créez un ticket si vous trouvez un problème. Les structures des tables d'extensions ne sont jamais sauvegardées puisque créées avec `CREATE EXTENSION` et sont supposées être identiques à version égale d'une extension. Ce comportement fait partie du modèle actuel d'extensions de PostgreSQL, nous ne pouvons donc rien faire à ce sujet.

Si vous avez installé la version 3.5.4dev sans utiliser le système d'extensions, il est possible de l'activer via les commandes suivantes pour packager les fonctions dans leurs extensions respectives. L'installation via `unpackaged` a été supprimé dans PostgreSQL 13, nous vous invitons donc à basculer sur une installation utilisant le système d'extensions avant de migrer à PostgreSQL 13.

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

2.2.6 Tests

Si vous désirez tester la compilation de PostGIS, lancez

make check

La commande ci-dessus fonctionnera pour différentes tests de vérification et de régression en utilisant la bibliothèque générée selon la base de données actuelle.

Note
Si vous avez configuré PostGIS en utilisant un emplacement non standard de PostgreSQL, GEOS, ou Proj4, vous pourrez avoir besoin d'ajouter l'emplacement des bibliothèques à la variable d'environnement `LD_LIBRARY_PATH`.

Caution
Pour le moment, **make check** repose sur les variables d'environnement `PATH` et `PGPORT` lors de la réalisation des vérifications - il n'utilise *pas* la version de PostgreSQL qui a pu être définie en utilisant la paramètre de configuration **--with-pgconfig**. Assurez vous donc de modifier votre PATH pour correspondre l'installation de PostgreSQL détectée durant la configuration ou préparez vous à gérer des maux de tête à venir.

Si réussi, make check sortira les résultats de près de 500 tests. Les résultats seront similaires à la sortie ci-dessous (avec de nombreuses lignes omises) :

```
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
```

Les extensions `postgis_tiger_geocoder` et `address_standardizer` ne supportent actuellement que l'installcheck de PostgreSQL. Pour les tester, cf ci-dessous. Note : Il n'est pas nécessaire de lancer make install si cette commande a déjà été lancée dans le répertoire racine des sources PostGIS.

Pour l'extension address_standardize :

```
cd extensions/address_standardizer
make install
make installcheck
```

La sortie de la commande devrait ressembler à :

```
============== dropping database "contrib_regression" ==============
DROP DATABASE
============== creating database "contrib_regression" ==============
CREATE DATABASE
ALTER DATABASE
============== running regression test queries        ==============
test test-init-extensions     ... ok
test test-parseaddress        ... ok
test test-standardize_address_1 ... ok
test test-standardize_address_2 ... ok

====================
 All 4 tests passed.
====================
```

Le géocodeur tiger nécessite d'avoir les extensions postgis et fuzzystrmatch installée sur l'instance PostgreSQL. Les tests de l'extension address_standardizer seront lancés si PostGIS est compilé avec le support address_standardizer :

```
cd extensions/postgis_tiger_geocoder
make install
make installcheck
```

La sortie de la commande devrait ressembler à :

```
============== dropping database "contrib_regression" ==============
DROP DATABASE
============== creating database "contrib_regression" ==============
CREATE DATABASE
ALTER DATABASE
============== installing fuzzystrmatch               ==============
CREATE EXTENSION
============== installing postgis                     ==============
CREATE EXTENSION
============== installing postgis_tiger_geocoder      ==============
CREATE EXTENSION
============== installing address_standardizer        ==============
CREATE EXTENSION
============== running regression test queries        ==============
test test-normalize_address   ... ok
test test-pagc_normalize_address ... ok

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

2.2.7 Installation

Pour installer PostGIS, entrez

make install

Ceci copiera les fichiers d'installation de PostGIS dans leur sous-répertoires appropriés définis par le paramètre de configuration **--prefix**. En particulier :

- Les binaires du chargeur et du dumper sont installés dans `[prefix]/bin`.
- Les fichiers SQL, tel que `postgis.sql`, sont installé dans `[prefix]/share/contrib`.
- Les bibliothèques PostGIS sont installées dans `[prefix]/lib`.

Si vous avez déjà lancé la commande **make comments** pour générer les fichiers `postgis_comments.sql`, `raster_comments.sql`, installer le fichier sql en lançant

make comments-install

Note
`postgis_comments.sql`, `raster_comments.sql`, `topology_comments.sql` ont été séparés de la compilation initiale et des cibles de l'installation depuis qu'ils sont dépendant de **xsltproc**.

2.3 Installation et utilisation de l'extension address standardize

L'extension `address_standardizer` était précédemment livrée sous forme d'un paquet séparé nécessitant son propre téléchargement. Depuis la version 2.2 de PostGIS, cette extension est intégrée. Pour de plus amples informations sur cette extension, sa configuration, son utilisation, se référer à Section 12.1.

Ce normalisateur d'adresses peut être utilisé avec l'extension PostGIS tiger en remplacement de Normalize_Address. Se référer à la page Section 2.4.2 pour mettre en place ce remplacement. Il peut également être utilisé pour fabriquer son propre géocodeur ou pour normaliser des adresses pour les comparer plus facilement.

Le normalisateur d'adresses se base sur PCRE, généralement installé sur les systèmes Nix. Il peut également être téléchargé ici : http://www.pcre.org. Durant la phase Section 2.2.3, si PCRE est détecté, le normalisateur d'adresses sera automatiquement compilé. Pour utiliser une installation personnalisée de PCRE, passer le paramètre `--with-pcredir=/chemin/vers/pcre` dans la commande configure, où `/chemin/vers/pcre` est le répertoire contenant les sous-répertoires pcre include et lib.

Pour les utilisateurs de Windows®, les versions 2.1 et supérieures de PostGIS sont livrées avec l'extension address_standardizer. Il n'est donc pas besoin de compiler cette extension. La commande `CREATE EXTENSION` suffit.

Une fois installée, vous pouvez vous connecter à votre base de données et lancer le SQL :

```
CREATE EXTENSION address_standardizer;
```

Le test suivant ne nécessite pas de table rules, gas, ou lex

```
SELECT num, street, city, state, zip
 FROM parse_address('1 Devonshire Place PH301, Boston, MA 02109');
```

La sortie de la commande devrait ressembler à

```
num |         street         |  city  | state |  zip
-----+------------------------+--------+-------+-------
 1   | Devonshire Place PH301 | Boston | MA    | 02109
```

2.4 Installation, mise à jour et chargement de données pour le géocodeur Tiger

L'extension géocodeur Tiger peut ne pas être distribué avec votre installation de PostGIS. Si vous n'avez pas l'extension géocodeur Tiger, ou si vous souhaitez utiliser une version plus récente de celle de votre installation, vous pouvez utiliser les fichiers `share/extension/postgis_tiger_geocoder.*` depuis les paquets disponibles à Windows Unreleased Versions, dans la section pour votre version de PostgreSQL. Même si ces paquets sont pour Windows, les fichiers d'extension postgis_tiger_geocoder fonctionneront quel que soit votre système d'exploitation car c'est une extension en SQL/plpgsql.

2.4.1 Tiger Geocoder Activation de votre base de données PostGIS

1. Ces instructions supposent que l'extension postgis_tiger_geocoder soit déjà installée dans votre installation PostgreSQL.

2. Connectez-vous à la base de données avec psql ou PgAdmin (ou tout autre client) et lancez la commande SQL suivante. Note : Si l'installation se déroule sur une base de données contenant déjà PostGIS, la première étape n'est pas nécessaire. Si l'extension `fuzzystrmatch` est déjà installée, la seconde étape n'est pas nécessaire non plus.

```
CREATE EXTENSION postgis;
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION postgis_tiger_geocoder;
--this one is optional if you want to use the rules based standardizer ( ↩
    pagc_normalize_address)
CREATE EXTENSION address_standardizer;
```

   Si l'extension postgis_tiger_geocoder est déjà installée et que vous souhaitez la mettre à jour, lancez :

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION postgis_tiger_geocoder UPDATE;
```

   Si vous avez modifié `tiger.loader_platform` ou `tiger.loader_variables`, vous devrez peut être les mettre à jour.

3. Pour tester l'installation, lancez cette commande SQL sur la base de données :

```
SELECT na.address, na.streetname,na.streettypeabbrev, na.zip
        FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
```

   Qui devrait afficher

```
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
           1 | Devonshire | Pl               | 02109
```

4. Créez un nouvel enregistrement dans la table `tiger.loader_platform` contenant les chemins vers les exécutables et le serveur.

   Par exemple, pour créer un profil nommé debbie suivant la convention `sh`, vous feriez :

```
INSERT INTO tiger.loader_platform(os, declare_sect, pgbin, wget, unzip_command, psql,  ↩
    path_sep,
                    loader, environ_set_command, county_process_command)
SELECT 'debbie', declare_sect, pgbin, wget, unzip_command, psql, path_sep,
           loader, environ_set_command, county_process_command
  FROM tiger.loader_platform
  WHERE os = 'sh';
```

   Et modifiez les chemins dans la colonne *declare_sect* pour les faire correspondre aux chemins des programmes pg, unzip, shp2pgsql, psql, etc. sur le serveur Debbie.

   Si vous ne modifiez pas la table `loader_platform`, elle contiendra des chemins par défaut pour les programmes, et vous aurez à modifier les scripts après leur génération.

5. Depuis PostGIS 2.4.1, le chargement des données de "ZIP Code Tabulation Area" `zcta5` a été modifié pour charger les données actuelles zcta5, celui-ci est inclus dans Loader_Generate_Nation_Script si activé. Le chargement est désactivé par défaut car cela prend du temps (20 à 60 minutes), occupe de l'espace disque, et n'est pas souvent utile.

   Pour l'activer, exécutez :

```
UPDATE tiger.loader_lookuptables SET load = true WHERE table_name = 'zcta520';
```

   Si disponible, la fonction Geocode peut l'utiliser lorsqu'un filtre de frontière est utilisé avec des codes zips. La fonction Reverse_Geocode l'utilise si l'adresse retournée n'a pas de code zip, ce qui arrive souvent avec le géocodage inverse sur des autoroutes.

6. Créez un répertoire nommé `gisdata` à la racine du serveur ou de la machine locale si le réseau entre les deux est suffisamment rapide. Ce répertoire contiendra les fichiers tiger téléchargés et traités. Pour changer ce répertoire, modifier le champ `staging_fold` dans la table `tiger.loader_variables`.

7. Créez un répertoire nommé temp dans le répertoire `gisdata` (ou dans le répertoire que vous avez configuré dans le champ `staging_fold`). Ce répertoire contiendra les données tiger extraites.

8. Puis exécutez la fonction SQL Loader_Generate_Nation_Script, pour vous assurer que votre libellé de profil personnalisé est utilisé, et sauvegardez le script dans un fichier .sh ou .bat. Par exemple pour générer le script de chargement d'une nation :

```
psql -c "SELECT Loader_Generate_Nation_Script('debbie')" -d geocoder -tA > /gisdata/ ↩
    nation_script_load.sh
```

9. Exécutez en ligne de commande les scripts de chargement précédemment générés.

```
cd /gisdata
sh nation_script_load.sh
```

10. Lorsque vous avez terminé d'exécuter les scripts, vous devriez avoir trois tables dans votre schéma `tiger_data`, avec des données déjà remplies. Vous pouvez confirmer cela avec les requêtes suivantes, à exécuter dans psql ou pgAdmn

```
SELECT count(*) FROM tiger_data.county_all;
```

```
count
-------
  3235
(1 row)
```

```
SELECT count(*) FROM tiger_data.state_all;
```

```
count
-------
    56
(1 row)
```

Celle-ci n'aura des données que si vous avez spécifié le chargement de zcta5

```
SELECT count(*) FROM tiger_data.zcta5_all;
```

```
count
-------
  33933
(1 row)
```

11. Par défaut, les tables correspondant à `bg`, `tract`, `tabblock20` ne sont pas chargées. Ces tables ne sont pas utilisées par le géocodeur, mais sont typiquement utilisées pour les statistiques de population. Si vous souhaitez charger ces données lors du chargement des états, exécutez la requête suivante.

```
UPDATE tiger.loader_lookuptables SET load = true WHERE load = false AND lookup_name IN ↩
    ('tract', 'bg', 'tabblock20');
```

Autrement, il est possible de charger juste ces tables après le chargement des données sur les états en utilisant le Loader_Generate_

12. Pour chaque état dont vous voulez charger les données, générez un script d'état Loader_Generate_Script.

Warning
NE GÉNÉREZ PAS le script d'état avant d'avoir chargé les données de nation, car le script d'état utilise la liste des comtés chargés par le script de nation.

13.
```
psql -c "SELECT Loader_Generate_Script(ARRAY['MA'], 'debbie')" -d geocoder -tA > / ↩
    gisdata/ma_load.sh
```

14. Lancez alors les lignes de commande générées.

```
cd /gisdata
sh ma_load.sh
```

15. Après le chargement des données, ou lors d'une pause dans le chargement, il peut être utile de lancer analyze sur toutes les tables tiger pour mettre à jour les statistiques (y compris les statistiques héritées)

```
SELECT install_missing_indexes();
vacuum (analyze, verbose) tiger.addr;
vacuum (analyze, verbose) tiger.edges;
vacuum (analyze, verbose) tiger.faces;
vacuum (analyze, verbose) tiger.featnames;
vacuum (analyze, verbose) tiger.place;
vacuum (analyze, verbose) tiger.cousub;
vacuum (analyze, verbose) tiger.county;
vacuum (analyze, verbose) tiger.state;
vacuum (analyze, verbose) tiger.zcta5;
vacuum (analyze, verbose) tiger.zip_lookup_base;
vacuum (analyze, verbose) tiger.zip_state;
vacuum (analyze, verbose) tiger.zip_state_loc;
```

2.4.2 Utilisation de l'Extension Address Standardizer avec le Geocodeur Tiger

Une des plaintes les plus récurrentes concerne la fonction Normalize_Address, qui normalise l'adresse avant de la géocoder. La normalisation est loin d'être parfaite, et corriger ces imperfections consomme énormément de ressources. Par conséquent, nous avons intégré un autre projet qui a un moteur de standardisation d'adresses bien meilleur. Pour utiliser ce nouveau address_standardizer, compilez l'extension en suivant Section 2.3 et installez l'extension dans votre base de données.

Une fois que vous avez installé cette extension dans la même base de données où vous avez installé `postgis_tiger_geocoder`, vous pouvez utiliser Pagc_Normalize_Address à la place de Normalize_Address. Cette extension ne dépend pas de Tiger, et peut donc être utilisée avec d'autres sources de données telles que des adresses internationales. L'extension de géocodage Tiger est installée avec ses propres versions spécifiques de rules table (`tiger.pagc_rules`) , gaz table (`tiger.pagc_gaz`), et lex table (`tiger.pagc_lex`). Vous pouvez les améliorer et ajouter des règles pour avoir de meilleurs résultats de géocodage pour vos besoins spécifiques.

2.4.3 Outils nécessaires pour charger des données tiger

Le processus de chargement télécharge les données du site web census pour respectivement les fichiers nation, état demandé, extrait les fichiers, puis charge chaque état dans son ensemble de tables état. Chaque table state hérite de la table définie dans le schéma `tiger`, il est suffisant d'interroger ces tables pour accéder à toutes les données et supprimer un ensemble de table state n'importe quand en utilisant Drop_State_Tables_Generate_Script si vous devez recharger un état ou si vous en avez plus besoin.

Dans l'objectif de charger des données vous avez besoin des outils suivants :

- Un outils pour décompresser les fichiers zip du site web census.

  Pour les systèmes Unix-like : un exécutable `unzip` qui est habituellement installé sur la plupart des plateformes Unix-like.

  Pour Windows, 7-zip qui est un outils de compression/décompression libre, vous pouvez le récupérer à partir de http://www.7-zip.org/

- La commande `shp2pgsql` qui est installé par défaut quand vous installez PostGIS.

- `wget` qui est un outil de récupération de lien habituellement installé sur les systèmes Unix/Linux.

  Si vous êtes sous Windows, vous pouvez obtenir des binaires pré-compilés à partir de http://gnuwin32.sourceforge.net/-packages/wget.htm

Si vous mettez à jour depuis tiger_2010, vous aurez besoin d'abord de générer et exécuter le Drop_Nation_Tables_Generate_Script. Avant de charger les données d'états, vous devez charger les données de nation via Loader_Generate_Nation_Script. Celui-ci va générer un script de chargement pour vous. Loader_Generate_Nation_Script est une étape à exécuter une seule fois, pour la mises à jour (depuis des données tiger précédentes) ou pour de nouvelles installations.

Pour charger les données d'états, voir Loader_Generate_Script pour générer un script pour charger les données pour votre plate-forme et les états que vous souhaitez. Note : vous pouvez les charger peu à peu, vous n'avez pas besoin de tout charger d'un coup. Vous pouvez les charger lorsque vous en avez besoin.

Après que les états que vous désirez aient été chargé, assurez vous de lancer la commande :

```
SELECT install_missing_indexes();
```

comme décrit dans Install_Missing_Indexes.

Pour tester que les choses fonctionnent comme elles le devraient, essayez de lancer un géocodage sur une adresse de votre état en utilisant Geocode

2.4.4 Mise à jour du géocoder Tiger et de ses données

Tout d'abord, mettez à jour l'extension postgis_tiger_geocoder en exécutant :

```
ALTER EXTENSION postgis_tiger_geocoder UPDATE;
```

Puis supprimez toutes les tables nation et chargez les nouvelles. Générez un script de suppression avec la requête SQL comme détaillée dans Drop_Nation_Tables_Generate_Script

```
SELECT drop_nation_tables_generate_script();
```

Lancement des requêtes SQL de suppression générées.

Générez un script de chargement des pays avec la requête SELECT comme détaillé dans Loader_Generate_Nation_Script

Pour windows

```
SELECT loader_generate_nation_script('windows');
```

Pour unix/linux

```
SELECT loader_generate_nation_script('sh');
```

Reportez-vous à Section 2.4.1 pour savoir comment exécuter le script de génération. Cette opération ne doit être effectuée qu'une seule fois.

Note
Vous pouvez avoir un mix de différentes années dans vos tables d'états et vous pouvez les mettre à jour indépendamment. Avant de mettre à jour un état, vous devez d'abord supprimer l'année précédente dans les tables d'états pour cet état en utilisant Drop_State_Tables_Generate_Script.

2.5 Problèmes courants pendant l'installation

Il y a plusieurs choses à vérifier quand votre installation ou mise à jour ne va pas dans la direction souhaitée.

1. Vérifiez que vous avez installé PostgreSQL 12 ou plus récent et que vous êtes en train de compiler avec la même version du code source de PostgreSQL que la version qui fonctionne. Un mélange peut arriver lorsque votre distribution (Linux) a déjà une version de PostgreSQL installée ou que vous avez oublié que vous avez déjà installée une version. PostGIS fonctionnera uniquement avec PostgreSQL 12 ou plus récent, et des messages d'erreurs étranges et inhabituelles en résultera si vous utilisez une version plus ancienne. Pour vérifier la version de PostgreSQL qui fonctionne, connectez vous à la base en utilisant psql et lancez la requête :

   ```
   SELECT version();
   ```

   Si vous utilisez une distribution basé sur les RPM, vous pouvez vérifier l'existence de paquets pré-installés en utilisant la commande **rpm** comme suit : **rpm -qa | grep postgresql**

2. Si votre mise à jour plante, assurez vous de la présence de PostGIS dans la nouvelle base de données.

   ```
   SELECT postgis_full_version();
   ```

Vérifiez également que le script configure a correctement détecté les chemins et versions de PostgreSQL, de la bibliothèque Proj.4 et de la blibliothèque GEOS.

1. La sortie du configure est utilisée pour générer le fichier `postgis_config.h`. Vérifiez que les variables `POSTGIS_PGSQL_VE` `POSTGIS_PROJ_VERSION` et `POSTGIS_GEOS_VERSION` ont été définies correctement.

Chapter 3

PostGIS Administration

3.1 Optimisation des performances

Le réglage des performances de PostGIS est similaire à celui de n'importe quelle charge de travail PostgreSQL. La seule considération supplémentaire est que les géométries et les rasters sont généralement de grande taille, donc les optimisations liées à la mémoire ont généralement plus d'impact sur PostGIS que sur d'autres types de requêtes PostgreSQL.

Pour plus de détails sur l'optimisation de PostgreSQL, reportez-vous à Tuning your PostgreSQL Server.

Pour PostgreSQL 9.4+, la configuration peut être définie au niveau du serveur sans toucher à `postgresql.conf` ou `postgresql.a` en utilisant la commande `ALTER SYSTEM`.

```
ALTER SYSTEM SET work_mem = '256MB';
-- this forces non-startup configs to take effect for new connections
SELECT pg_reload_conf();
-- show current setting value
-- use SHOW ALL to see all settings
SHOW work_mem;
```

En plus des paramètres de Postgres, PostGIS a quelques paramètres personnalisés qui sont listés dans Section 7.22.

3.1.1 Démarrage

Ces paramètres sont configurés dans `postgresql.conf` :

constraint_exclusion

- Valeur par défaut : partition
- Ceci est généralement utilisé pour le partitionnement des tables. La valeur par défaut est "partition", ce qui est idéal pour PostgreSQL 8.4 et plus car cela force le planificateur à n'analyser les tables pour la prise en compte des contraintes que si elles sont dans une hiérarchie héritée et à ne pas payer de pénalité au planificateur dans le cas contraire.

shared_buffers

- Valeur par défaut : ~128MB dans PostgreSQL 9.6
- Réglez-le à environ 25 à 40 % de la mémoire vive disponible. Sous Windows, il se peut que vous ne puissiez pas définir une valeur aussi élevée.

max_worker_processes Ce paramètre n'est disponible que pour PostgreSQL 9.4+. Pour PostgreSQL 9.6+, ce paramètre a une importance supplémentaire car il contrôle le nombre maximum de processus que vous pouvez avoir pour les requêtes parallèles.

- Valeur par défaut : 8
- Définit le nombre maximum de processus d'arrière-plan que le système peut prendre en charge. Ce paramètre ne peut être défini qu'au démarrage du serveur.

3.1.2 Temps d'exécution

work_mem - définit la taille de la mémoire utilisée pour les opérations de tri et les requêtes complexes

- Valeur par défaut : 1-4MB
- Ajustement pour les grandes bases de données, les requêtes complexes, beaucoup de RAM
- Ajuster à la baisse pour de nombreux utilisateurs simultanés ou une faible mémoire vive.
- Si vous avez beaucoup de mémoire vive et peu de développeurs :

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

maintenance_work_mem - la taille de la mémoire utilisée pour VACUUM, CREATE INDEX, etc.

- Valeur par défaut : 16-64MB
- Généralement trop faible - immobilise les Entrées/Sorties, bloque les objets pendant l'échange de mémoire
- Nous recommandons 32MB à 1GB sur les serveurs de production avec beaucoup de RAM, mais cela dépend du nombre d'utilisateurs simultanés. Si vous avez beaucoup de RAM et peu de développeurs :

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

max_parallel_workers_per_gather

Ce paramètre n'est disponible que pour PostgreSQL 9.6+ et n'affecte que PostGIS 2.3+, puisque seul PostGIS 2.3+ supporte les requêtes parallèles. Si ce paramètre est supérieur à 0, certaines requêtes telles que celles impliquant des fonctions de relation comme `ST_Intersects` peuvent utiliser plusieurs processus et s'exécuter plus de deux fois plus rapidement. Si vous avez beaucoup de processeurs à disposition, vous devriez changer la valeur de ce paramètre pour autant de processeurs que vous avez. Assurez-vous également d'augmenter `max_worker_processes` à une valeur au moins égale à ce nombre.

- Valeur par défaut : 0
- Définit le nombre maximum de workers qui peuvent être démarrés par un seul nœud `Gather`. Les workers parallèles sont pris dans le pool de processus établi par `max_worker_processes`. Notez que le nombre de workers demandé peut ne pas être disponible au moment de l'exécution. Dans ce cas, le plan s'exécutera avec moins de workers que prévu, ce qui peut s'avérer inefficace. La définition de cette valeur à 0, qui est la valeur par défaut, désactive l'exécution parallèle des requêtes.

3.2 Configuring raster support

If you enabled raster support you may want to read below how to properly configure it.

As of PostGIS 2.1.3, 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_RASTERS` in the server environment. For PostGIS 2.2, you can use the more cross-platform approach of setting the corresponding Section 7.22.

If you want to enable offline raster:

```
POSTGIS_ENABLE_OUTDB_RASTERS=1
```

Any other setting or no setting at all will disable out of db rasters.

In order to enable all GDAL drivers available in your GDAL install, set this environment variable as follows

```
POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL
```

If you want to only enable specific drivers, set your environment variable as follows:

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

Note
If you are on windows, do not quote the driver list

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/10/main/environment` where 10 refers to version of PostgreSQL and main refers to the cluster.

On windows, if you are running as a service, you can set via System variables which for Windows 7 you can get to by right-clicking on Computer->Properties Advanced System Settings or in explorer navigating to `Control Panel\All Control Panel Items\System`. Then clicking *Advanced System Settings ->Advanced->Environment Variables* and adding new system variables.

After you set the environment variables, you'll need to restart your PostgreSQL service for the changes to take effect.

3.3 Creating spatial databases

3.3.1 Spatially enable database using EXTENSION

If you are using PostgreSQL 9.1+ and have compiled and installed the extensions/postgis modules, you can turn a database into a spatial one using the EXTENSION mechanism.

Core postgis extension includes geometry, geography, spatial_ref_sys and all the functions and comments. Raster and topology are packaged as a separate extension.

Run the following SQL snippet in the database you want to enable spatially:

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

3.3.2 Spatially enable database without using EXTENSION (discouraged)

Note
This is generally only needed if you cannot or don't want to get PostGIS installed in the PostgreSQL extension directory (for example during testing, development or in a restricted environment).

Adding PostGIS objects and function definitions into your database is done by loading the various sql files located in `[prefix]/share/contrib` as specified during the build phase.

The core PostGIS objects (geometry and geography types, and their support functions) are in the `postgis.sql` script. Raster objects are in the `rtpostgis.sql` script. Topology objects are in the `topology.sql` script.

For a complete set of EPSG coordinate system definition identifiers, you can also load the `spatial_ref_sys.sql` definitions file and populate the `spatial_ref_sys` table. This will permit you to perform ST_Transform() operations on geometries.

If you wish to add comments to the PostGIS functions, you can find them in the `postgis_comments.sql` script. Comments can be viewed by simply typing **\dd [function_name]** from a **psql** terminal window.

Run the following Shell commands in your terminal:

```
DB=[yourdatabase]
    SCRIPTSDIR=`pg_config --sharedir`/contrib/postgis-3.4/

    # 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 Upgrading spatial databases

Upgrading existing spatial databases can be tricky as it requires replacement or introduction of new PostGIS object definitions.

Unfortunately not all definitions can be easily replaced in a live database, so sometimes your best bet is a dump/reload process.

PostGIS provides a SOFT UPGRADE procedure for minor or bugfix releases, and a HARD UPGRADE procedure for major releases.

Before attempting to upgrade PostGIS, it is always worth to backup your data. If you use the -Fc flag to pg_dump you will always be able to restore the dump with a HARD UPGRADE.

3.4.1 Soft upgrade

If you installed your database using extensions, you'll need to upgrade using the extension model as well. If you installed using the old sql script way, you are advised to switch your install to extensions because the script way is no longer supported.

3.4.1.1 Soft Upgrade 9.1+ 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.

If you are running PostGIS 3 or above, then you should use the PostGIS_Extensions_Upgrade function to upgrade to the latest version you have installed.

```
SELECT postgis_extensions_upgrade();
```

If you are running PostGIS 2.5 or lower, then do the following:

```
ALTER EXTENSION postgis UPDATE;
SELECT postgis_extensions_upgrade();
-- This second call is needed to rebundle postgis_raster extension
SELECT postgis_extensions_upgrade();
```

If you have multiple versions of PostGIS installed, and you don't want to upgrade to the latest, you can explicitly specify the version as follows:

```
ALTER EXTENSION postgis UPDATE TO "3.5.4dev";
ALTER EXTENSION postgis_topology UPDATE TO "3.5.4dev";
```

If you get an error notice something like:

```
No migration path defined for ... to 3.5.4dev
```

Then you'll need to backup your database, create a fresh one as described in Section 3.3.1 and then restore your backup on top of this new database.

If you get a notice message like:

```
Version "3.5.4dev" of extension "postgis" is already installed
```

Then everything is already up to date and you can safely ignore it. **UNLESS** you're attempting to upgrade from an development version to the next (which doesn't get a new version number); in that case you can append "next" to the version string, and next time you'll need to drop the "next" suffix again:

```
ALTER EXTENSION postgis UPDATE TO "3.5.4devnext";
ALTER EXTENSION postgis_topology UPDATE TO "3.5.4devnext";
```

Note
If you installed PostGIS originally without a version specified, you can often skip the reinstallation of postgis extension before restoring since the backup just has `CREATE EXTENSION postgis` and thus picks up the newest latest version during restore.

Note
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. You can drop as follows:

```
DROP EXTENSION postgis_raster;
```

3.4.1.2 Soft Upgrade Pre 9.1+ or without extensions

This section applies only to those who installed PostGIS not using extensions. If you have extensions and try to upgrade with this approach you'll get messages like:

```
can't drop ... because postgis extension depends on it
```

NOTE: if you are moving from PostGIS 1.* to PostGIS 2.* or from PostGIS 2.* prior to r7409, you cannot use this procedure but would rather need to do a HARD UPGRADE.

After compiling and installing (make install) you should find a set of `*_upgrade.sql` files in the installation folders. You can list them all with:

```
ls `pg_config --sharedir`/contrib/postgis-3.5.4dev/*_upgrade.sql
```

Load them all in turn, starting from `postgis_upgrade.sql`.

```
psql -f postgis_upgrade.sql -d your_spatial_database
```

The same procedure applies to raster, topology and sfcgal extensions, with upgrade files named `rtpostgis_upgrade.sql`, `topology_upgrade.sql` and `sfcgal_upgrade.sql` respectively. If you need them:

```
psql -f rtpostgis_upgrade.sql -d your_spatial_database
```

```
psql -f topology_upgrade.sql -d your_spatial_database
```

```
psql -f sfcgal_upgrade.sql -d your_spatial_database
```

You are advised to switch to an extension based install by running

```
psql -c "SELECT postgis_extensions_upgrade();"
```

Note
If you can't find the `postgis_upgrade.sql` specific for upgrading your version you are using a version too early for a soft upgrade and need to do a HARD UPGRADE.

The PostGIS_Full_Version function should inform you about the need to run this kind of upgrade using a "procs need upgrade" message.

3.4.2 Hard upgrade

By HARD UPGRADE we mean full dump/reload of postgis-enabled databases. You need a HARD UPGRADE when PostGIS objects' internal storage changes or when SOFT UPGRADE is not possible. The Release Notes appendix reports for each version whether you need a dump/reload (HARD UPGRADE) to upgrade.

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 at Windows Hard upgrade.

The Procedure is as follows:

1. Create a "custom-format" dump of the database you want to upgrade (let's call it `olddb`) include binary blobs (-b) and verbose (-v) output. The user can be the owner of the db, need not be postgres super account.

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

2. Do a fresh install of PostGIS in a new database -- we'll refer to this database as `newdb`. Please refer to Section 3.3.2 and Section 3.3.1 for instructions on how to do this.

   The spatial_ref_sys entries found in your dump will be restored, but they will not override existing ones in spatial_ref_sys. This is to ensure that fixes in the official set will be properly propagated to restored databases. If for any reason you really want your own overrides of standard entries just don't load the spatial_ref_sys.sql file when creating the new db.

   If your database is really old or you know you've been using long deprecated functions in your views and functions, you might need to load `legacy.sql` for all your functions and views etc. to properly come back. Only do this if _really_ needed. Consider upgrading your views and functions before dumping instead, if possible. The deprecated functions can be later removed by loading `uninstall_legacy.sql`.

3. Restore your backup into your fresh `newdb` database using postgis_restore. Unexpected errors, if any, will be printed to the standard error stream by psql. Keep a log of those.

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

Errors may arise in the following cases:

1. Some of your views or functions make use of deprecated PostGIS objects. In order to fix this you may try loading `legacy.sql` script prior to restore or you'll have to restore to a version of PostGIS which still contains those objects and try a migration again after porting your code. If the `legacy.sql` way works for you, don't forget to fix your code to stop using deprecated functions and drop them loading `uninstall_legacy.sql`.

2. Some custom records of spatial_ref_sys in dump file have an invalid SRID value. Valid SRID values are bigger than 0 and smaller than 999000. Values in the 999000.999999 range are reserved for internal use while values > 999999 can't be used at all. All your custom records with invalid SRIDs will be retained, with those > 999999 moved into the reserved range, but the spatial_ref_sys table would lose a check constraint guarding for that invariant to hold and possibly also its primary key (when multiple invalid SRIDS get converted to the same reserved SRID value).

   In order to fix this you should copy your custom SRS to a SRID with a valid value (maybe in the 910000..910999 range), convert all your tables to the new srid (see UpdateGeometrySRID), delete the invalid entry from spatial_ref_sys and re-construct the check(s) with:

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

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

   If you are upgrading an old database containing french IGN cartography, you will have probably SRIDs out of range and you will see, when importing your database, issues like this :

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

   In this case, you can try following steps : first throw out completely the IGN from the sql which is resulting from postgis_restore. So, after having run :

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

   run this command :

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

   Create then your newdb, activate the required Postgis extensions, and insert properly the french system IGN with : this script After these operations, import your data :

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

Chapter 4

Gestion des données

4.1 Modèle de données spatiales

4.1.1 Géométrie OGC

L'Open Geospatial Consortium (OGC) a développé le standard *Simple Features Access* (SFA) pour fournir un modèle de données géospatiales. Ce standard définit le type spatial **Geometry**, ainsi que les opérations pour manipuler et transformer des géométries, et permettre des tâches d'analyses spatiales. PostGIS implémente le modèle OGC Geometry sous forme de types PostgreSQL geometry et geography.

Geometry est un type *abstrait*. Les valeurs géométriques utilisent les sous-types *concrets*, qui représentent les diverses formes géométriques. Ces types incluent les types **atomiques** Point, LineString, LinearRing et Polygon, ainsi que les types de **collections** MultiPoint, MultiLineString, MultiPolygon et GeometryCollection. Le standard *Simple Features Access - Part 1: Common architecture v1.2.1* ajoute les sous-types pour les structures PolyhedralSurface, Triangle et TIN.

Geometry représente des formes dans le plan cartésien en 2 dimensions. Les types PolyhedralSurface, Triangle, et TIN peuvent également représenter des formes en 3 dimensions. La taille et la localisation des formes sont spécifiées par leurs **coordonnées**. Chaque coordonnées a une **dimension** X et une Y, qui déterminent sa position sur le plan. Les formes sont construites à partir de points ou de segments. Les points sont spécifiés par une seul coordonnée ; les segments par deux coordonnées.

Les coordonnées peuvent inclure des valeurs pour les dimensions optionnelles Z et M. La dimension Z est souvent utilisée pour représenter l'élévation. La dimension M contient une mesure, qui représente par exemple le temps ou une distance. Si la dimension Z ou M est présente pour une valeur géométrique, elle doit être définie pour tous les points de la géométrie. Si une géométrie a une dimension Z ou M, la **dimension de la coordonnée** est 3D ; si elle a à la fois les dimension Z et M, la dimension de la coordonnée est 4D.

Les valeurs géométriques sont associées à un **système de coordonnées de référence** (SCR, ou en anglais spatial reference system, SRS), qui indique dans quel système de coordonnées les valeurs sont définies. Le SCR est identifié par un identifiant appelé SRID. Les unités sur les axes X et Y sont déterminées par ce système de coordonnées de référence. Dans un système **planaire**, les coordonnées X et Y représentent les distances respectivement selon l'Est et le Nord. Dans un système **géodésique** elles représentent la longitude et la latitude. L'identifiant SRID 0 représente un plan cartésien infini, sans unité sur ses axes. Voir Section 4.5.

La **dimension** d'une géométrie est une propriété des types géométriques. Les types Point ont une dimension 0, les types linéaires ont une dimension 1, et les types polygonaux ont une dimension 2. Les collections ont la dimension de leur élément de plus grande dimension.

Une valeur géométrique peut être **vide**. Une valeur vide ne contient aucun vertex (pour les types de géométriques atomiques) ou aucun élément (pour les collections).

Une propriété important des valeurs géométriques est leur **emprise** spatiale (extent en anglais) ou leur **boîte englobante** (bounding box en anglais), que le modèle OGC nomme **enveloppe** (envelope). La boîte englobante est la boîte 2D ou 3D qui contient les coordonnées d'une géométrie. C'est une façon efficace de représenter l'emprise d'une géométrie dans un espace et de tester comment deux géométries interagissent.

Le modèle de géométrie permet d'évaluer les relations topologiques, telles que décrites dans Section 5.1.1. Pour supporter cela, les concepts de **intérieur** (interior en anglais), **frontière** (boundary en anglais) et **extérieur** (exterior en anglais) sont définis pour tous les types de géométries. Les géométries sont topologiquement fermées, donc elle contiennent toujours leur frontière. La dimension de la géométrie de la frontière est la dimension de la géométrie moins un.

Le modèle de géométrie OGC définie des règles de validité pour chaque type géométrique. Ces règles permettent de s'assurer que les valeurs géométriques représentent des situations réalistes (e.g. il est possible de définir un polygone avec un trou à l'extérieur, mais cela n'a pas de sens au niveau géométrique, et est donc invalide). PostGIS permet de stocker et de manipuler des valeurs géométriques invalides, ceci permet de les détecter et de les corriger si besoin. Voir Section 4.4

4.1.1.1 Point

Un objet de type Point est une géométrie de dimension 0, qui représente un seul point dans l'espace.

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

4.1.1.2 LineString

Le type LineString est une ligne, de dimension 1, formée par une séquence de segments linéaires contigus. Chaque segment est défini par deux points, l'extrémité d'un segment étant le point de départ du segment suivant. Un LineString valide au sens OGC a soit zéro, soit deux points ou plus, mais PostGIS permet d'avoir des LineStrings avec un seul point. LineStrings peuvent se croiser (auto-intersection). Un LineString peut être **fermé** si les points de début et fin sont les mêmes. Un LineString est dit **simple** s'il ne s'auto-intersecte pas.

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

4.1.1.3 LinearRing

Le type LinearRing définit un LineString qui est à la fois fermé et simple. Autrement dit, le premier et dernier points doivent être égaux, et la ligne ne doit pas s'auto-intersecter.

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

4.1.1.4 Polygon

Un polygone, de type Polygon est une région d'un plan, de dimension 2, délimité par une frontière extérieure (la coquille, shell en anglais) et zéro, une ou plusieurs frontières intérieures (trous, holes en anglais). Chaque frontière est un LinearRing.

```
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

Le type MultiPoint est une collection de Points.

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

4.1.1.6 MultiLineString

Le type MultiLineString est une collection de LineStrings. Un MultiLineString est fermé si tous ses éléments sont fermés.

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

4.1.1.7 MultiPolygon

Un MultiPolygon est une collection de Polygons, non superposés et non adjacents. Les polygones de la collection peuvent se toucher, mais uniquement en un nombre fini de points.

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

4.1.1.8 GeometryCollection

Le type GeometryCollection représente une collection hétérogène de géométries (i.e. de types différents).

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

4.1.1.9 PolyhedralSurface

Le type PolyhedralSurface modélise une surface polyédrique, sous la forme d'une collection de faces qui partagent des arêtes. Chaque face est un Polygon plan. Si les coordonnées du Polygon ont une dimension Z, alors la surface est de dimension 3.

```
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 Triangle

Un Triangle est un polygone définit par trois sommets non colinéaires. Un Triangle étant un Polygon, et donc est fermé, il est définit par quatre coordonnées, la première et la dernière étant égales.

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

4.1.1.11 TIN

Un TIN est une collection de Triangles non superposés, représentant un réseau irrégulier triangulé (Triangulated Irregular Network).

```
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 Part 3 - Courbes

La norme *ISO/IEC 13249-3 SQL Multimedia - Spatial* (SQL/MM) étend l'OGC SFA pour définir des sous-types de géométrie contenant des courbes avec des arcs circulaires. Les types SQL/MM prennent en charge les coordonnées 3DM, 3DZ et 4D.

Note
Toutes les comparaisons en flottant dans l'implémentation de SQL-MM sont effectuées avec une tolérance spécifiée, actuellement 1E-8.

4.1.2.1 CircularString

CircularString est le type de courbe de base, similaire à LineString dans le monde linéaire. Un segment d'arc unique est spécifié par trois points : les points de départ et d'arrivée (premier et troisième) et un autre point de l'arc. Pour spécifier un cercle fermé, les points de départ et d'arrivée sont les mêmes et le point central est le point opposé sur le diamètre du cercle (qui est le centre de l'arc). Dans une séquence d'arcs, le point final de l'arc précédent est le point de départ de l'arc suivant, tout comme les segments d'une LineString. Cela signifie qu'une CircularString doit avoir un nombre impair de points supérieur à 1.

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

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

4.1.2.2 CompoundCurve

Une CompoundCurve est une courbe continue unique qui peut contenir à la fois des segments d'arc de cercle et des segments linéaires. Cela signifie qu'en plus d'avoir des composantes bien formées, le point final de chaque composante (sauf la dernière) doit coïncider avec le point de départ de la composante suivante.

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

4.1.2.3 CurvePolygon

Un CurvePolygon est semblable à un polygone, avec un anneau extérieur et zéro ou plusieurs anneaux intérieurs. La différence est qu'un anneau peut être une CircularString ou une CompoundCurve ainsi qu'une LineString.

Depuis PostGIS 1.4, PostGIS prend en charge les courbes composées dans un polygone de courbe.

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

Exemple : Un PolygoneCourbe dont l'enveloppe est définie par une CompoundCurve contenant une CircularString et une LineString, et dont le trou est défini par une CircularString

```
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

Une MultiCurve est un ensemble de courbes qui peuvent inclure des LineStrings, des CircularStrings ou des CompoundCurves.

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

4.1.2.5 MultiSurface

Une MultiSurface est un ensemble de surfaces, qui peuvent être des polygones (linéaires) ou des polygones courbes.

```
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 et WKB

La spécification OGC SFA définit deux formats pour représenter des valeurs géométriques : Well-Known Text (WKT) et Well-Known Binary (WKB). Ces deux formats incluent les informations sur le type d'objet et sur les coordonnées qui le définissent.

Well-Known Text (WKT) fournit un standard pour représenter de façon textuelle des données spatiales. Voici quelques exemples de représentations WKT :

- POINT(0 0)
- POINT Z (0 0 0)
- POINT ZM (0 0 0 0)
- POINT EMPTY
- LINESTRING(0 0,1 1,1 2)
- LINESTRING EMPTY
- 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 EMPTY
- 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)))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))
- GEOMETRYCOLLECTION EMPTY

Des méthodes d'entrée/sortie en WKT sont fournies via les fonctions ST_AsText et ST_GeomFromText :

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

Par exemple, une requête pour créer et insérer une objet spatial sous forme de WKT et avec un SRID :

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

Well-Known Binary (WKB) fournit un moyen portable et sans perte de précision pour représenter des données spatiales sous la forme de données binaires (tableau d'octets). Voici quelques exemples de représentations WKB :

- WKT : POINT(1 1)
  WKB : 01010000000000000000F03F000000000000F03
- WKT : LINESTRING (2 2, 9 9)
  WKB : 0102000000020000000000000000000040000000000000004000000000000022400000000000002240

Des méthodes d'entrée/sortie en WKB sont fournies via les fonctions ST_AsBinary et ST_GeomFromWKB :

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

Par exemple, une requête pour créer et insérer une objet spatial sous forme de WKB :

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

4.2 Type de données Geometry

PostGIS implémente le modèle Simple Features de l'OGC via un type PostgreSQL `geometry`. Ce type représente tous les sous-types de géométries en utilisant un type interne (voir GeometryType et ST_GeometryType). Cela permet de modéliser les entités spatiales comme lignes de tables qui contiennent une colonne de type `geometry`.

Le type `geometry` est *opaque*, ce qui veut dire que tout accès est fait en appelant des fonctions sur les valeurs géométriques. Des fonctions permettent la création d'objets géométriques, l'accès et la mise à jour des champs internes, ainsi que le calcul de nouvelles valeurs géométriques. PostGIS supporte toutes les fonctions spécifiées par la spécification OGC *Simple feature access - Part 2: SQL option* (SFS), ainsi que de nombreuses autres. voir Chapter 7 pour une liste complète des fonctions disponibles.

Note
PostGIS respecte le standard SFA en préfixant toutes les fonctions spatiales par "ST_". Initialement, cela voulait dire "Spatial and Temporal" (Spatial et Temporel), mais l'aspect temporel du standard n'a pas été développé. À la place, ceci peut être interprété comme "Spatial Type" (Type Spatial).

La standard SFA spécifie que les objets spatiaux doivent inclure un identifiant de système de coordonnées de référence (SRID). Ce SRID est obligatoire lors de la création d'objets spatiaux pour l'insertion dans la base de données (mais peut être défini par défaut à 0). Voir ST_SRID et Section 4.5

Pour optimiser les requêtes sur les géométries, PostGIS définit plusieurs types d'index spatiaux, ainsi que des opérateurs spatiaux pour les utiliser. Voir Section 4.9 et Section 5.2 pour plus d'informations.

4.2.1 PostGIS EWKB et EWKT

Les spécifications OGC SFA ne supportaient initialement que les géométries 2D, et le SRID de la géométrie n'est pas inclut dans les représentations d'entrée/sortie. La spécification OGC SFA 1.2.1 (qui est alignée avec le standard ISO 19125) ajoute le support pour la 3D (ZYZ) et les mesures (XYM et XYZM), mais n'inclut toujours pas la valeur du SRID.

À cause de ces limitations, PostGIS définit les formats étendus EWKB ((Extended Well-Known Binary) et EWKT (Extended Well-Known Text). Ils supportent la 3D (XYZ et XYM) et 4D (XYZM) et incluent l'information de SRID. Le format EWKB incluant toutes les informations de la géométrie, cela permet à PostGIS d'utiliser ce format pour les enregistrements (e.g. dans les fichiers DUMP).

EWKB et EWKT sont utilisés pour les "formes canoniques" des objets spatiaux de PostGIS. En tant qu'entrée, la forme canonique pour les données binaires est EWKB ; pour les données textuelles EWKB et EWKT sont tous deux acceptés. Cela permet de créer des valeurs géométriques en transtypant une valeur textuelle en HEXEWKB ou EWKT vers une valeur géométrique en utilisant `::geometry`. Pour la sortie, la forme canonique pour les données binaires est EWKB et HEXEWKB (hex-encoded EWKB) pour les données textuelles.

Par exemple, cette requête créé une géométrie en transtypant depuis une chaîne de caractères contenant du EKWT, et retourne sa valeur sous sa forme canonique en HEXEWKB :

```
SELECT 'SRID=4;POINT(0 0)'::geometry;
  geometry
  ----------------------------------------------------
  0101000020040000000000000000000000000000000000000000
```

La sortie PostGIS EWKT a quelques différences avec le OGC WKT :

- Pour les géométries 3DZ, le qualificatif Z est omis :
  OGC : POINT Z (1 2 3)
  EWKT : POINT (1 2 3)
- Pour les géométries 3DM, le qualificatif M est inclus :
  OGC : POINT M (1 2 3)
  EWKT : POINTM (1 2 3)

- Pour les géométries 4D, le qualificatif ZM est omis :
  OGC : POINT ZM (1 2 3 4)
  EWKT : POINT (1 2 3 4)

EWKT évite de sur-spécifier les dimensions et ainsi éviter les inconsistances possibles avec le format OGC/ISO, comme :

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

Caution
Les formats étendus PostGIS sont des sur-ensembles des formats OGC, donc les représentations valides WKB/WKT sont aussi des représentations EWKB/EWKT valides. Cependant, cela pourrait changer à l'avenir, si l'OGC définit un format qui rentrerait en conflit avec la définition de PosGIS. Vous ne devriez donc PAS vous appuyer sur cette compatibilité !

Voici quelques exemples de représentations d'objets spatiaux sous forme EWKT :

- POINT(0 0 0) -- XYZ
- SRID=32632;POINT(0 0) -- XY avec SRID
- POINTM(0 0 0) -- XYM
- POINT(0 0 0 0) -- XYZM
- SRID=4326;MULTIPOINTM(0 0 0,1 2 1) -- XYM avec 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(POINTM(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)))
- TRIANGLE ((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)))

Des méthodes d'entrée/sortie sont fournies via les fonctions suivantes :

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

Par exemple, une requête pour créer et insérer une objet spatial sous forme de EWKT :

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

4.3 Type de données Geography

Le type de données PostGIS `geography` permet le support des entités spatiales utilisant un système de coordonnées géographique (parfois appelé géodésique, ou "latitude/longitude" ou "longitude/latitude"). Les coordonnées géographiques sont des coordonnées sphériques, exprimées en unités d'angle (degrés).

Le type geometry de PostGIS est lié à un plan. Ainsi, le chemin le plus court entre deux points sur un plan est la ligne droite. Les fonctions sur les types geometry (aires, distances, intersections, etc.) sont donc calculées en utilisant des lignes droites et les mathématiques cartésiennes. Cela permet des implémentations plus faciles et plus rapides à exécuter, mais cela devient imprécis lorsque la rotondité de la Terre entre en jeu.

Le type PostGIS geography repose sur un modèle sphérique. Le chemin le plus court entre deux points sur une sphère est l'arc de cercle. Les fonctions sur les types geography (aires, distances, intersections, etc.) sont donc calculées en utilisant des arcs de cercle sur une sphère. En prenant en compte la rotondité de la Terre, ces fonctions permettent d'avoir une meilleure précision.

Les mathématiques utilisées pour les calculs étant plus compliquées, moins de fonctions sont définies pour le type geography que pour le type geometry. De nouveaux algorithmes sont ajoutés au fur et à mesure des versions de PostGIS, donc le support du type geography s'étend petit à petit. Si une fonction n'est pas disponible, il est toutefois possible de convertir un type geography en geometry puis vice-versa.

Comme le type geometry, les données géographiques sont liées à un identifiant de système de coordonnées de référence (SRID). Tout système géodésique (reposant sur longitude/latitude) peut être utilisé, tant qu'il défini dans la table `spatial_ref_sys`. (Avant PostGIS 2.2, le type geography ne supportait que le SCR WGS 84 (SRID:4326)). Vous pouvez ajouter votre propre SCR géodésique, comme décrit dans Section 4.5.2.

Pour tous les systèmes de coordonnées de référence, les unités des valeurs retournées par les fonctions de mesure (e.g. ST_Distance, ST_Length, ST_Perimeter, ST_Area) et le paramètre de distance de ST_DWithin sont en mètres.

4.3.1 Création de tables géographiques

Vous pouvez créer une table pour stocker des données géographiques en utilisant la requête SQL CREATE TABLE, avec une colonne de type `geography`. L'exemple suivant créé une table avec une colonne géographique pour stocker des lignes 2D dans un SCR géodésique WGS84 (SRID 4326) :

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

Le type geography supporte deux modificateurs de type :

- le modificateur de type spatial restreint le type de formes et dimensions de la colonne. Les valeurs permises pour le type spatial sont : POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION. Le type geography ne supporte pas les CURVEs, TINS, ni POLYHEDRALSURFACEs. Le modificateur permet de restreindre la dimension, en ajoutant les suffixes : Z, M ou ZM. Par exemple, un modificateur 'LINESTRINGM' permet uniquement de stocker des LineStrings en 3D, et traite la troisième dimension comme une mesure. De façon similaire, 'POINTZM' limite aux données 4D (XYZM).
- le modificateur SRID restreint à un SCR spécifique. Si omis, le SRID 4326 (WGS84 géodésique) est utilisé, et tous les calculs sont effectués en utilisant WGS84.

Exemples de création de tables utilisant des colonnes géographiques :

- Création d'une table avec une géographie POINT 2D avec le SRID par défaut 4326 (WGS84 longitude/latitude) :

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

- Création d'une table avec une géographie POINT 2D dans le CRS NAD83 longlat :

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

- Création d'une table avec une géographie POINT 3D (XYZ) avec le SCR explicite 4326 :

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

- Création d'une table avec une géographie LINESTRING 2D avec le SRID par défaut 4326 :

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

- Création d'une table avec une géographie POLYGON 2D avec le SRC 4267 (NAD 1927 long lat) :

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

Les colonnes géographiques sont enregistrées dans la vue système `geography_columns`. Vous pouvez effectuer un requête sur la vue `geography_columns` et vérifier que la table est bien listée :

```
SELECT * FROM geography_columns;
```

La création d'index spatiaux sur les colonnes de type geography fonctionne de la même manière qu'avec le type geometry. PostGIS va prendre en compte que le type de la colonne est GEOGRAPHY et créer un index sphérique au lieu d'un index planaire utilisé pour GEOMETRY.

```
-- Index the test table with a spherical index
CREATE INDEX global_points_gix ON global_points USING GIST ( location );
```

4.3.2 Utilisation des tables géographiques

Vous pouvez insérer des données dans des tables géographiques de la même façon qu'avec les géométries. Les données géométriques seront automatiquement transtypées en type geography si le SRID des données est 4326. Les formats EWKT et EWKB peuvent aussi être utilisés pour spécifier les valeurs géographiques.

```
-- Add some data into the test table
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)');
```

Tout CRS géodésique (longitude/latitude) disponible dans la table `spatial_ref_sys` peut être utilisé comme SRID d'une géographie. L'utilisation d'un CRS non géodésique provoquera une erreur.

```
-- NAD 83 lon/lat
SELECT 'SRID=4269;POINT(-123 34)'::geography;
                    geography
----------------------------------------------------
 0101000020AD10000000000000C05EC00000000000004140
```

```
-- NAD27 lon/lat
SELECT 'SRID=4267;POINT(-123 34)'::geography;
                    geography
----------------------------------------------------
 0101000020AB10000000000000C05EC00000000000004140
```

```
-- NAD83 UTM zone meters - gives an error since it is a meter-based planar projection
SELECT 'SRID=26910;POINT(-123 34)'::geography;

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

Les requêtes et les fonctions de mesure utilisent le mètre comme unité. Les paramètres de distance doivent être exprimés en mètres, et les valeurs de retours seront également en mètres (ou en mètres carré pour les surfaces).

```
-- A distance query using a 1000km tolerance
SELECT name FROM global_points WHERE ST_DWithin(location, 'SRID=4326;POINT(-110 29)':: ↩
    geography, 1000000);
```

Vous pouvez vérifier la puissance des géographies en calculant à quel point un avion s'approche de Reykjavik (POINT(-21.96 64.15)) lors d'un trajet en arc de cercle depuis Seattle à Londres (LINESTRING(-122.33 47.606, 0.0 51.5)) (voir la route).

Le type geography donne la plus petite distance réelle de 122,235 km sur la sphère entre Reykjavik et l'arc de cercle entre Seattle et Londres.

```
-- Distance calculation using GEOGRAPHY
SELECT ST_Distance('LINESTRING(-122.33 47.606, 0.0 51.5)'::geography, 'POINT(-21.96 64.15) ↩
    '::geography);
   st_distance
-----------------
 122235.23815667
```

Le type geometry quant à lui calcule la distance cartésienne entre Reykjavik et la ligne droite entre Seattle et Londres, telle qu'elle serait tracée sur un plan. Cette distance n'a pas de sens réel, d'autant que l'unité du résultat est techniquement en degrés, mais que le résultat ne correspond à aucune différence angulaire entre les points, donc même considérer le résultat comme des degrés serait faux.

```
-- Distance calculation using GEOMETRY
SELECT ST_Distance('LINESTRING(-122.33 47.606, 0.0 51.5)'::geometry, 'POINT(-21.96 64.15) ↩
    '::geometry);
      st_distance
--------------------
 13.342271221453624
```

4.3.3 Quand utiliser le type de données Geography

Le type de données geography permet de stocker des données en utilisant les coordonnées longitude/latitude, mais cela a un prix : il y a moins de fonctions disponibles sur le type GEOGRAPHY que sur le type GEOMETRY, et les fonctions disponibles prendront plus de temps CPU pour s'exécuter.

Le type de données à choisir devrait être déterminé par la zone de travail de l'application que vous construisez. Est-ce que vos données s'étendront sur l'ensemble du globe ou sur un continent, ou bien est-ce qu'elle seront locales, comme une région, un département ou une ville ?

- Si vous données sont limitées à une petite zone, utiliser un SCR adéquat et le type GEOMETRY est la meilleure solution, à la fois en terme de performances que de fonctionnalités disponibles.
- Si vos données s'étendent sur le monde entier ou couvrent un continent, le type GEOGRAPHY peut vous permettre de construire votre application sans trop vous soucier des projections. Vous pouvez stocker vos données en utilisant les coordonnées longitude/latitude, et utiliser les fonctions définies sur les GEOGRAPHY.
- Si vous n'êtes pas à l'aise avec les projections, que vous ne souhaitez pas approfondir le sujet, et que vous êtes prêts à accepter les limitations sur les fonctionnalités offertes par GEOGRAPHY, alors ce peut être plus facile d'utiliser GEOGRAPHY au lieu de GEOMETRY. Chargez vos données en longitude/latitude et partez de là.

Référez-vous à Section 13.11 pour comparer les supports Geography et Geometry. Pour avoir un résumé des fonctions géographiques disponibles, référez-vous à Section 13.4

4.3.4 FAQ Geography avancée

1. *Les calculs sont-ils faits sur la sphère ou sur la sphéroïde ?*

   Par défaut, tous les calculs sur les distances et les surfaces sont effectués sur la sphéroïde. Les résultats des calculs sur les petites zones devraient coïncider avec les résultats des calculs planaire en utilisant les projections locales adéquates. Sur de plus grandes zones, les calculs sphéroïdaux seront plus précis que ceux effectués sur un plan projeté. Toutes les fonctions géographiques ont une option pour calculer sur une sphère, en passant comme tout dernier paramètre booléen 'FALSE'. Ceci améliorera les performances des calculs, en particulier pour les géométries très simples.

2. *Qu'en est-il de la ligne de changement de date et des pôles ?*

   Tous les calculs omettent les concepts de ligne de changement de date et de pôles. Les coordonnées étant sphériques (longitude/latitude), une forme traversant la ligne de changement de date n'est, d'un point de vue calculs, pas différente de toute autre forme.

3. *Quel est l'arc le plus long que vous puissiez traiter ?*

   Nous utilisons les arcs de grand cercle comme "ligne d'interpolation" entre deux points. Cela signifie que deux points quelconques sont en fait reliés de deux manières différentes, selon la direction dans laquelle vous vous déplacez le long du grand cercle. Tout notre code suppose que les points sont reliés par le *plus court* des deux chemins le long du grand cercle. Par conséquent, les formes qui ont des arcs de plus de 180 degrés ne seront pas correctement modélisées.

4. *Pourquoi est-il si lent de calculer la superficie de l'Europe / de la Russie / insérer une grande région géographique ici ?*

   Parce que le polygone est vraiment énorme ! Les grandes zones sont néfastes pour deux raisons : leurs limites sont énormes, de sorte que l'index a tendance à tirer l'élément, quelle que soit la requête que vous exécutez ; le nombre de sommets est énorme, et les tests (distance, confinement) doivent parcourir la liste des sommets au moins une fois et parfois N fois (N étant le nombre de sommets dans l'autre élément candidat). Comme pour la GÉOMÉTRIE, nous vous recommandons, lorsque vous avez de très grands polygones, mais que vous effectuez des requêtes sur de petites zones, de "dénormaliser" vos données géométriques en morceaux plus petits, de sorte que l'index puisse effectivement interroger des parties de l'objet et que les requêtes n'aient pas à extraire l'objet entier à chaque fois. Veuillez consulter la documentation de la fonction ST_Subdivide. Ce n'est pas parce que vous *pouvez* stocker toute l'Europe dans un polygone que vous *devriez* le faire.

4.4 Validation de la géométrie

PostGIS est conforme à la spécification Simple Features de l'Open Geospatial Consortium (OGC). Cette norme définit les concepts de géométrie comme étant *simple* et *valide*. Ces définitions permettent au modèle géométrique Simple Features de représenter les objets spatiaux d'une manière cohérente et non ambiguë qui permet un calcul efficace. (Remarque : l'OGC SF et SQL/MM ont les mêmes définitions pour simple et valide.)

4.4.1 Géométrie simple

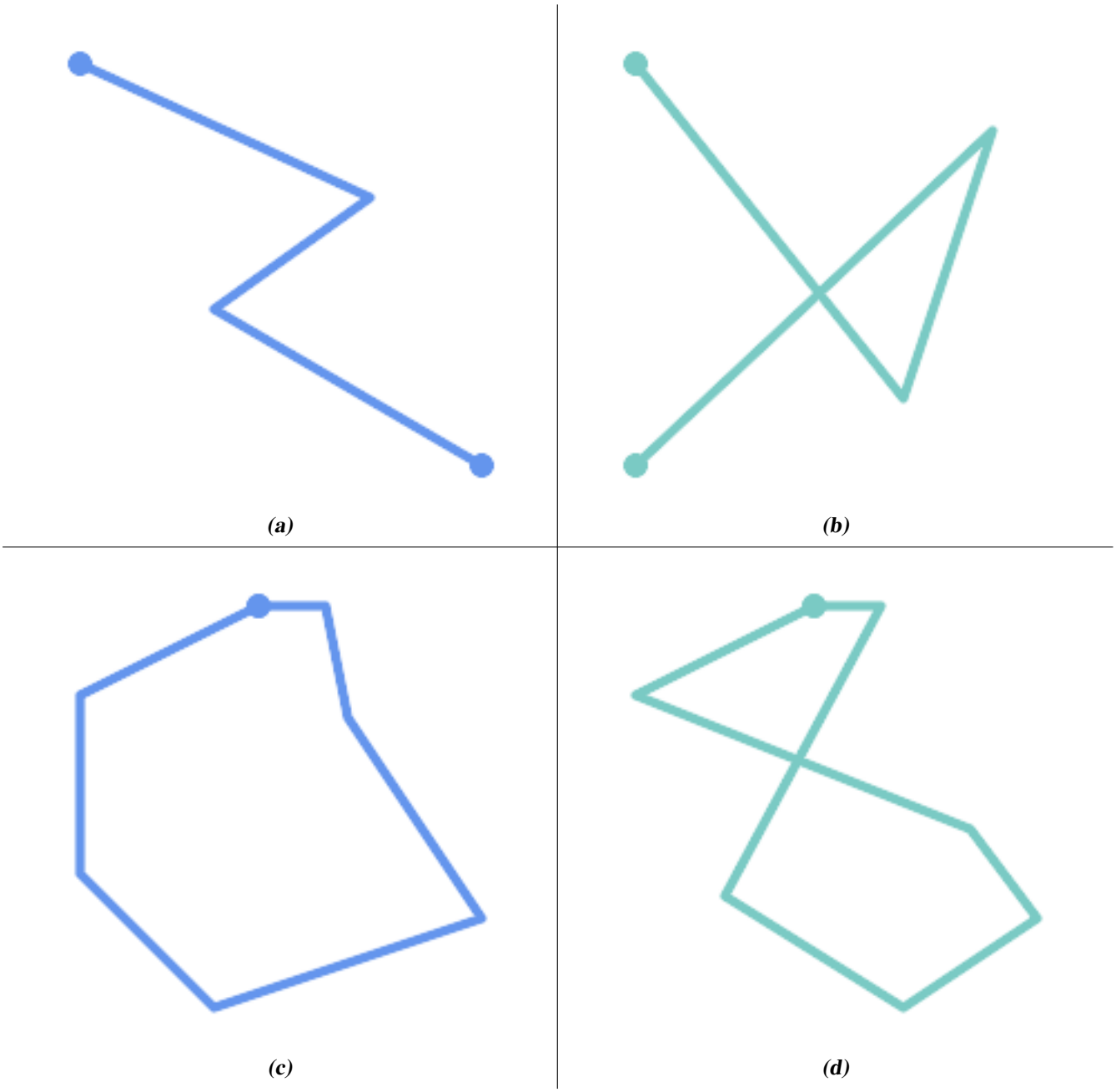
Une géométrie *simple* est une géométrie qui ne présente pas de points géométriques anormaux, tels qu'une intersection ou une tangence propre.

Un `POINT` est intrinsèquement *simple* en tant qu'objet géométrique à 0 dimension.

Les `MULTIPOINT` sont *simples* si deux coordonnées (`POINT`s) ne sont pas égales (ont des valeurs de coordonnées identiques).

Une `LINESTRING` est *simple* si elle ne passe pas deux fois par le même point, à l'exception des extrémités. Si les extrémités d'une simple LineString sont identiques, elle est dite *fermée* et appelée anneau linéaire.

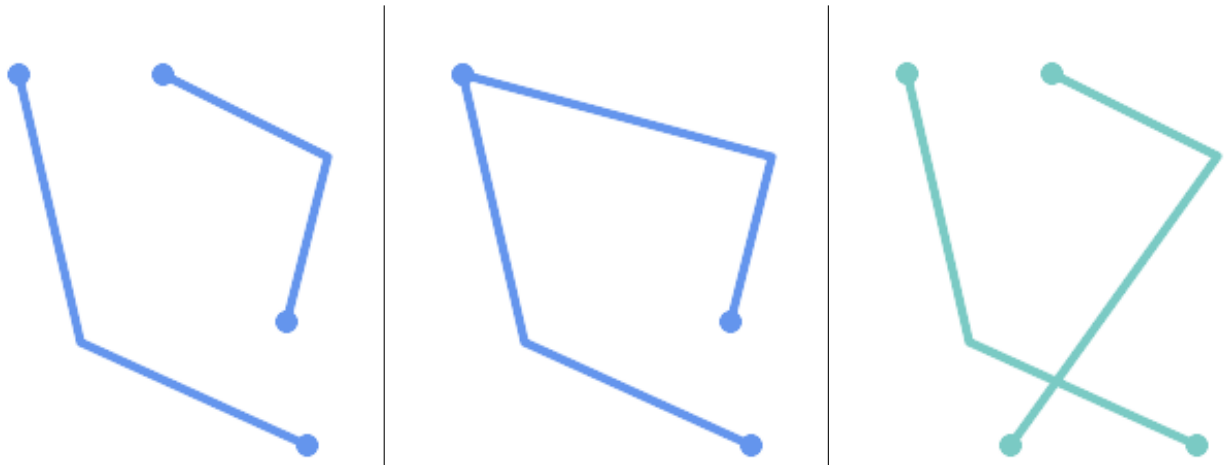
***(a)** et **(c)** sont des `LINESTRING` simples. **(b)** et **(d)** ne sont pas simples. **(c)** est un anneau linéaire fermé.*



(a) ***(b)*** ***(c)*** ***(d)***

Un `MULTILINESTRING` est *simple* uniquement si tous ses éléments sont simples et si la seule intersection entre deux éléments quelconques se produit en des points situés sur les limites des deux éléments.

(e) *et* ***(f)*** *sont de simples* `MULTILINESTRING`*s.* ***(g)*** *n'est pas simple.*



(e) *(f)* *(g)*

Les POLYGON sont formés à partir d'anneaux linéaires, de sorte que la géométrie polygonale valide est toujours *simple*.

Pour tester si une géométrie est simple, utilisez la fonction ST_IsSimple :

```
SELECT
   ST_IsSimple('LINESTRING(0 0, 100 100)') AS straight,
   ST_IsSimple('LINESTRING(0 0, 100 100, 100 0, 0 100)') AS crossing;

 straight | crossing
----------+----------
 t        | f
```

En général, les fonctions PostGIS n'exigent pas que les arguments géométriques soient simples. La simplicité est principalement utilisée comme base pour définir la validité géométrique. C'est également une exigence pour certains types de modèles de données spatiales (par exemple, les réseaux linéaires interdisent souvent les lignes qui se croisent). La géométrie multipoint et linéaire peut être rendue simple en utilisant ST_UnaryUnion.

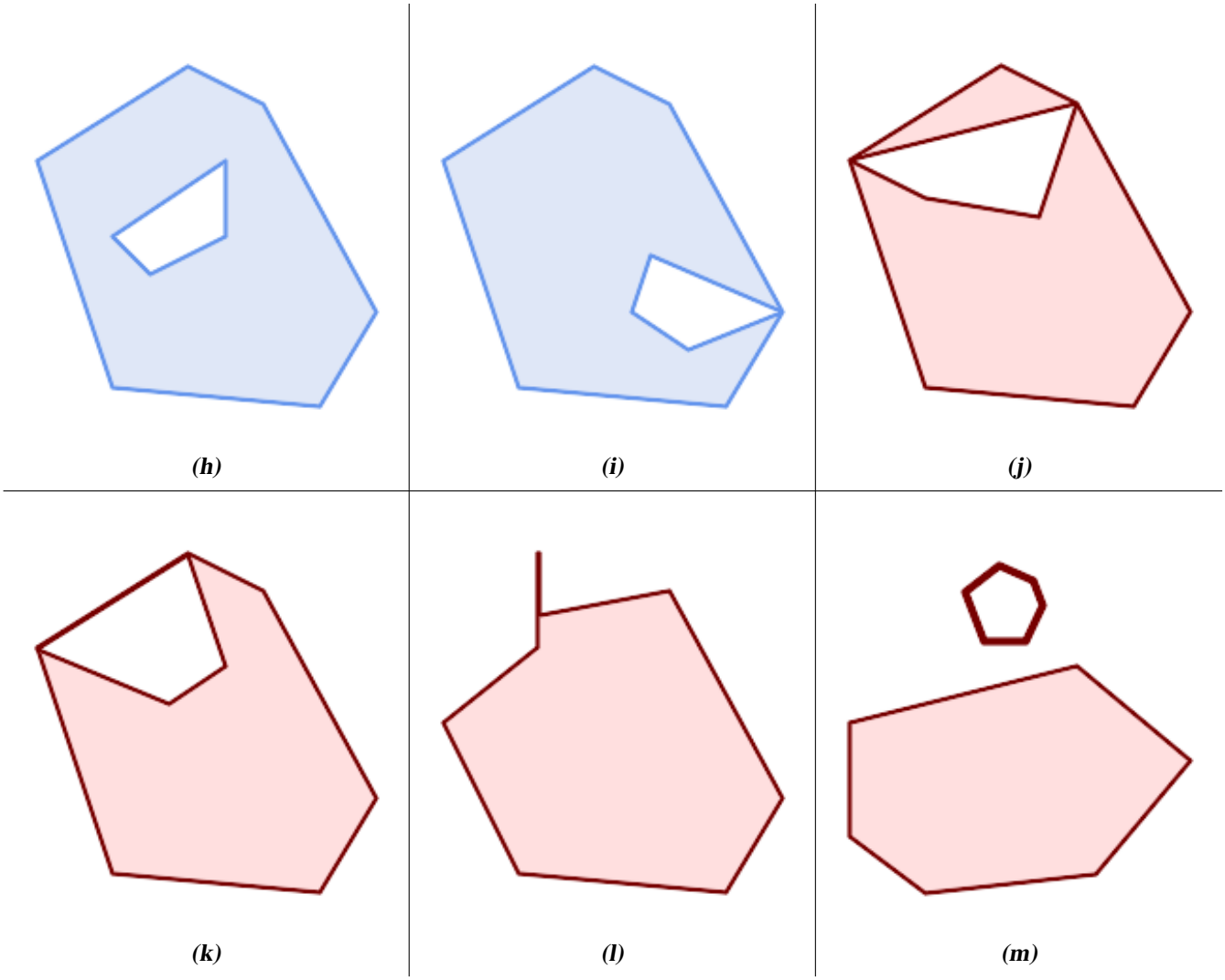
4.4.2 Géométrie valide

La validité géométrique s'applique principalement aux géométries bidimensionnelles (POLYGONs et MULTIPOLYGONs). La validité est définie par des règles qui permettent à la géométrie polygonale de modéliser des zones planes sans ambiguïté.

Un POLYGON est *valide* si :

1. les anneaux de délimitation du polygone (l'anneau extérieur de la coquille et les anneaux intérieurs des trous) sont *simples* (ils ne se croisent pas et ne se touchent pas). Pour cette raison, un polygone ne peut pas avoir de lignes de coupe, de pointes ou de boucles. Cela implique que les trous des polygones doivent être représentés par des anneaux intérieurs, plutôt que par l'anneau extérieur qui se touche (ce qu'on appelle un "trou inversé").
2. les anneaux de délimitation ne se croisent pas
3. Les anneaux de délimitation peuvent se toucher en certains points, mais uniquement sous la forme d'une tangente (c'est-à-dire pas sous la forme d'une ligne)
4. les anneaux intérieurs sont contenus dans l'anneau extérieur
5. l'intérieur du polygone est simplement connecté (c'est-à-dire que les anneaux ne doivent pas se toucher d'une manière qui divise le polygone en plus d'une partie)

***(h)** et **(i)** sont des `POLYGON` valides. **(j-m)** ne sont pas valides. **(j)** peut être représenté comme un `MULTIPOLYGON` valide.*



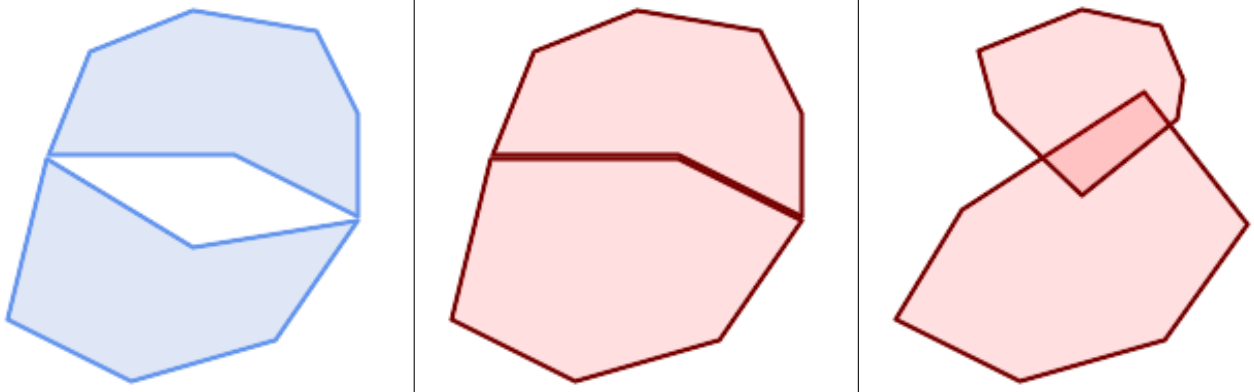
(h) ***(i)*** ***(j)***

(k) ***(l)*** ***(m)***

Un `MULTIPOLYGON` est *valide* si :

1. ses éléments `POLYGON` sont valides
2. les éléments ne se chevauchent pas (c'est-à-dire que leurs intérieurs ne doivent pas se croiser)
3. les éléments ne se touchent que par des points (c'est-à-dire pas le long d'une ligne)

***(n)** est un `MULTIPOLYGON` valide. **(o)** et **(p)** ne sont pas valides.*



(n) *(o)* *(p)*

Ces règles signifient que la géométrie polygonale valide est également *simple*.

Pour la géométrie linéaire, la seule règle de validité est que les `LINESTRING` doivent avoir au moins deux points et une longueur non nulle (ou, de façon équivalente, au moins deux points distincts).

```
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;

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

Les géométries `POINT` et `MULTIPOINT` n'ont pas de règles de validité.

4.4.3 Gestion de la validité

PostGIS permet de créer et de stocker des géométries valides et non valides. Cela permet de détecter les géométries non valides et de les signaler ou de les corriger. Il existe également des situations où les règles de validité de l'OGC sont plus strictes que souhaité (par exemple, les lignes de longueur nulle et les polygones avec des trous inversés.)

De nombreuses fonctions fournies par PostGIS reposent sur l'hypothèse que les arguments géométriques sont valides. Par exemple, il n'est pas logique de calculer la surface d'un polygone dont le trou est défini à l'extérieur du polygone, ou de construire un polygone à partir d'une ligne de démarcation non simple. Le fait de supposer que les entrées géométriques sont valides permet aux fonctions de fonctionner plus efficacement, puisqu'elles n'ont pas besoin de vérifier l'exactitude topologique. (Les exceptions notables sont que les lignes de longueur nulle et les polygones avec des inversions sont généralement traités correctement). En outre, la plupart des fonctions PostGIS produisent une géométrie valide si les données d'entrée sont valides. Cela permet d'enchaîner les fonctions PostGIS en toute sécurité.

Si vous rencontrez des messages d'erreur inattendus lors de l'appel de fonctions PostGIS (tels que "GEOS Intersection() threw an error !"), vous devez d'abord vous assurer que les arguments de la fonction sont valides. Si ce n'est pas le cas, envisagez d'utiliser l'une des techniques ci-dessous pour vous assurer que les données que vous traitez sont valides.

Note
Si une fonction signale une erreur avec des entrées valides, il se peut que vous ayez trouvé une erreur dans PostGIS ou dans l'une des bibliothèques qu'il utilise, et vous devez le signaler au projet PostGIS. Il en va de même si une fonction PostGIS renvoie une géométrie non valide pour des données d'entrée valides.

Pour vérifier si une géométrie est valide, utilisez la fonction ST_IsValid :

```
SELECT ST_IsValid('POLYGON ((20 180, 180 180, 180 20, 20 20, 20 180))');
-----------------
 t
```

Les informations relatives à la nature et à la localisation d'une invalidité géométrique sont fournies par la fonction ST_IsValidDetail :

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

 valid |      reason       |                  location
-------+-------------------+---------------------------------------------
 f     | Self-intersection | POINT(91.51162790697674 141.56976744186045)
```

Dans certaines situations, il est souhaitable de corriger automatiquement la géométrie non valide. Utilisez la fonction ST_MakeValid pour cela. (`ST_MakeValid` est un exemple de fonction spatiale qui *doit* permettre une entrée non valide !)

Par défaut, PostGIS ne vérifie pas la validité lors du chargement de la géométrie, car les tests de validité peuvent prendre beaucoup de temps CPU pour les géométries complexes. Si vous ne faites pas confiance à vos sources de données, vous pouvez imposer un contrôle de validité à vos tables en ajoutant une contrainte de contrôle :

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

4.5 Systèmes de référence spatiale

Un système de référence spatiale (SRS) (également appelé système de référence des coordonnées (SRC)) définit la manière dont la géométrie est référencée par rapport à des emplacements sur la surface de la Terre. Il existe trois types de SRS :

- Un SRS **géodésique** utilise des coordonnées angulaires (longitude et latitude) qui correspondent directement à la surface de la terre.
- Un SRS **projeté** utilise une transformation mathématique de projection pour "aplatir" la surface de la terre sphéroïdale sur un plan. Il attribue des coordonnées de localisation d'une manière qui permet de mesurer directement des quantités telles que la distance, la surface et l'angle. Le système de coordonnées est cartésien, ce qui signifie qu'il a un point d'origine défini et deux axes perpendiculaires (généralement orientés vers le nord et l'est). Chaque SRS projeté utilise une unité de longueur définie (généralement des mètres ou des pieds). Un SRS projeté peut être limité dans sa zone d'applicabilité afin d'éviter toute distorsion et de s'inscrire dans les limites des coordonnées définies.
- Un SRS **local** est un système de coordonnées cartésiennes qui n'est pas référencé à la surface de la terre. Dans PostGIS, ce système est spécifié par une valeur SRID de 0.

Il existe de nombreux systèmes de référence spatiale différents. Les SRS courants sont normalisés dans la base de données de l'European Petroleum Survey Group EPSG database. Par commodité, PostGIS (et de nombreux autres systèmes spatiaux) se réfère aux définitions des SRS à l'aide d'un identifiant entier appelé SRID.

Une géométrie est associée à un système de référence spatiale par sa valeur SRID, à laquelle on accède par ST_SRID. Le SRID d'une géométrie peut être assigné en utilisant ST_SetSRID. Certaines fonctions de construction de géométrie permettent de fournir un SRID (telles que ST_Point et ST_MakeEnvelope). Le format EWKT prend en charge les SRID avec le préfixe `SRID=n;`.

Les fonctions spatiales qui traitent des paires de géométries (telles que les fonctions overlay et relationship) exigent que les géométries d'entrée soient dans le même système de référence spatiale (qu'elles aient le même SRID). Les données géométriques peuvent être transformées dans un système de référence spatiale différent en utilisant ST_Transform et ST_TransformPipeline. Les géométries renvoyées par les fonctions ont le même SRS que les géométries d'entrée.

4.5.1 Table SPATIAL_REF_SYS

La table `SPATIAL_REF_SYS` utilisée par PostGIS est une table de base de données conforme à l'OGC qui définit les systèmes de référence spatiale disponibles. Elle contient les SRID numériques et les descriptions textuelles des systèmes de coordonnées.

La définition de la table `spatial_ref_sys` est la suivante :

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

Les colonnes sont :

`srid` Un code entier qui identifie de manière unique le Système de référence spatiale (SRS) dans la base de données.

`auth_name` Le nom de la norme ou de l'organisme de normalisation qui est cité pour ce système de référence. Par exemple, "EPSG" est un `auth_name` valide.

`auth_srid` L'ID du système de référence spatiale tel que défini par l'autorité citée dans le `auth_name`. Dans le cas de l'EPSG, il s'agit du code EPSG.

`srtext` Représentation Well-Known Text du système de référence spatiale. Un exemple de représentation WKT du SRS est le suivant :

```
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]
]
```

Pour une discussion sur le SRS WKT, voir la norme OGC Well-known text representation of coordinate reference systems.

`proj4text` PostGIS utilise la bibliothèque PROJ pour fournir des capacités de transformation de coordonnées. La colonne `proj4text` contient la chaîne de caractères de définition des coordonnées PROJ pour un SRID particulier. Par exemple :

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

Pour plus d'informations, consultez le site web de PROJ. Le fichier `spatial_ref_sys.sql` contient les définitions `srtext` et `proj4text` pour toutes les projections EPSG.

Lors de la récupération des définitions des systèmes de référence spatiale pour les utiliser dans les transformations, PostGIS utilise la stratégie suivante :

- Si `auth_name` et `auth_srid` sont présents (non NULL), utiliser le PROJ SRS basé sur ces entrées (si au moins un existe).
- Si `srtext` est présent, créez un SRS en l'utilisant, si possible.
- Si `proj4text` est présent, créez un SRS en l'utilisant, si possible.

4.5.2 Systèmes de référence spatiale définis par l'utilisateur

La table PostGIS `spatial_ref_sys` contient plus de 3000 définitions des systèmes de référence spatiale les plus courants qui sont gérés par la bibliothèque de projection PROJ. Mais il existe de nombreux systèmes de coordonnées qu'elle ne contient pas. Vous pouvez ajouter des définitions de SRS à la table si vous disposez des informations requises sur le système de référence spatiale. Vous pouvez également définir votre propre système de référence spatiale personnalisé si vous êtes familiarisé avec les constructions PROJ. Gardez à l'esprit que la plupart des systèmes de référence spatiale sont régionaux et n'ont aucune signification lorsqu'ils sont utilisés en dehors des limites pour lesquelles ils ont été conçus.

Une ressource permettant de trouver des systèmes de référence spatiale non définis dans le jeu de base est http://spatialreference.org/-

Les systèmes de référence spatiale les plus couramment utilisés sont les suivants : 4326 - WGS 84 Long Lat, 4269 - NAD 83 Long Lat, 3395 - WGS 84 World Mercator, 2163 - US National Atlas Equal Area, et les 60 zones UTM WGS84. Les zones UTM sont parmi les plus idéales pour les mesures, mais elles ne couvrent que des régions de 6 degrés. (Pour déterminer la zone UTM à utiliser pour votre zone d'intérêt, voir la utmzone PostGIS plpgsql helper function.)

Les États américains utilisent des systèmes de référence spatiale State Plane (basés sur le mètre ou le pied) - il en existe généralement un ou deux par État. La plupart des systèmes basés sur le mètre se trouvent dans le jeu de base, mais de nombreux systèmes basés sur le pied ou créés par ESRI devront être copiés à partir de spatialreference.org.

Vous pouvez même définir des systèmes de coordonnées non terrestres, tels que Mars 2000 Ce système de coordonnées martien n'est pas planaire (il est en degrés sphéroïdaux), mais vous pouvez l'utiliser avec le type `géographie` pour obtenir des mesures de longueur et de proximité en mètres au lieu de degrés.

Voici un exemple de chargement d'un système de coordonnées personnalisé à l'aide d'un SRID non attribué et de la définition PROJ pour une projection Lambert conforme centrée sur les États-Unis :

```
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 Tables spatiales

4.6.1 Créer une table spatiale

Vous pouvez créer une table pour stocker des données géométriques en utilisant l'instruction SQL CREATE TABLE avec une colonne de type `geometry`. L'exemple suivant crée une table avec une colonne géométrie stockant des chaînes de lignes 2D (XY) dans le système de coordonnées BC-Albers (SRID 3005) :

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

Le type `geometry` prend en charge deux **modificateurs de type** facultatifs :

- le **modificateur de type spatial** restreint le type de formes et de dimensions autorisées dans la colonne. La valeur peut être l'un des sous-types de géométrie pris en charge (par exemple POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION, etc). Le modificateur prend en charge les restrictions de dimensionnalité des coordonnées en ajoutant des suffixes : Z, M et ZM. Par exemple, le modificateur "LINESTRINGM" n'autorise que les lignes à trois dimensions et traite la troisième dimension comme une mesure. De même, "POINTZM" requiert des données à quatre dimensions (XYZM).
- le **modificateur SRID** limite le système de référence spatiale SRID à un nombre particulier. S'il est omis, le SRID prend par défaut la valeur 0.

Exemples de création de tableaux avec des colonnes géométriques :

- Créer une table contenant n'importe quel type de géométrie avec le SRID par défaut :

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

- Créer une table avec une géométrie POINT 2D avec le SRID par défaut :

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

- Créer une table avec des POINTS 3D (XYZ) et un SRID explicite de 3005 :

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

- Créer une table avec une géométrie 4D (XYZM) LINESTRING avec le SRID par défaut :

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

- Créer une table avec une géométrie POLYGON 2D avec le SRID 4267 (NAD 1927 long lat) :

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

Il est possible d'avoir plus d'une colonne de géométrie dans une table. Cela peut être spécifié lors de la création de la table, ou une colonne peut être ajoutée à l'aide de l'instruction SQL ALTER TABLE. Cet exemple ajoute une colonne qui peut contenir des LineStrings 3D :

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

4.6.2 Vue GEOMETRY_COLUMNS

L'OGC *Simple Features Specification for SQL* définit la table de métadonnées `GEOMETRY_COLUMNS` pour décrire la structure de la table géométrique. Dans PostGIS, `geometry_columns` est une vue qui lit les tables de catalogue du système de base de données. Cela garantit que les informations de métadonnées spatiales sont toujours cohérentes avec les tables et les vues actuellement définies. La structure de la vue est la suivante :

```
\d geometry_columns
```

```
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)  |
```

Les colonnes sont :

`f_table_catalog`, `f_table_schema`, `f_table_name` Le nom complet de la table d'entités contenant la colonne géométrie. Il n'y a pas d'analogue PostgreSQL de "catalog", cette colonne est donc laissée vide. Pour "schema", le nom du schéma PostgreSQL est utilisé (`public` est la valeur par défaut).

`f_geometry_column` Le nom de la colonne géométrique dans la table des caractéristiques.

`coord_dimension` La dimension des coordonnées (2, 3 ou 4) de la colonne.

srid L'ID du système de référence spatiale utilisé pour la géométrie des coordonnées dans cette table. Il s'agit d'une référence de clé étrangère à la table `spatial_ref_sys` (voir Section 4.5.1).

type Le type de l'objet spatial. Pour limiter la colonne spatiale à un seul type, utilisez l'une des options suivantes : POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION ou les versions XYM correspondantes POINTM, LINESTRINGM, POLYGONM, MULTIPOINTM, MULTILINESTRINGM, MULTIPOLYGONM, GEOMETRYCOLLECTIONM. Pour les collections hétérogènes (de type mixte), vous pouvez utiliser "GEOMETRY" comme type.

4.6.3 Enregistrement manuel des colonnes de géométrie

Deux des cas où vous pouvez avoir besoin de cela sont les cas de vues SQL et les insertions en masse. Dans le cas des insertions en masse, vous pouvez corriger l'enregistrement dans la table geometry_columns en contraignant la colonne ou en effectuant une modification de la table. Pour les vues, vous pouvez exposer en utilisant une opération CAST. Notez que si votre colonne est basée sur un typmod, le processus de création l'enregistrera correctement, il n'est donc pas nécessaire de faire quoi que ce soit. De même, les vues qui n'ont pas de fonction spatiale appliquée à la géométrie s'enregistreront de la même manière que la colonne géométrique de la table sous-jacente.

```
-- Lets say you have a view created like this
CREATE VIEW public.vwmytablemercator AS
        SELECT gid, ST_Transform(geom, 3395) As geom, f_name
        FROM public.mytable;

-- For it to register correctly
-- You need to cast the geometry
--
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 you know the geometry type for sure is a 2D POLYGON then you could do
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;
```

```
--Lets say you created a derivative table by doing a bulk insert
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 2D index on new table
CREATE INDEX idx_myschema_myspecialpois_geom_gist
  ON myschema.my_special_pois USING gist(geom);

-- If your points are 3D points or 3M points,
-- then you might want to create an nd index instead of a 2D index
CREATE INDEX my_special_pois_geom_gist_nd
        ON my_special_pois USING gist(geom gist_geometry_ops_nd);

-- To manually register this new table's geometry column in geometry_columns.
-- Note it will also change the underlying structure of the table to
-- to make the column typmod based.
SELECT populate_geometry_columns('myschema.my_special_pois'::regclass);

-- If you are using PostGIS 2.0 and for whatever reason, you
-- you need the constraint based definition behavior
-- (such as case of inherited tables where all children do not have the same type and srid)
-- set optional use_typmod argument to false
```

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

Bien que l'ancienne méthode basée sur les contraintes soit toujours supportée, une colonne géométrique basée sur les contraintes utilisée directement dans une vue ne s'enregistrera pas correctement dans geometry_columns, tout comme une colonne typmod. Dans cet exemple, nous définissons une colonne à l'aide de typmod et une autre à l'aide de contraintes.

```
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);
```

Si nous exécutons dans psql

```
\d pois_ny;
```

Nous observons qu'ils sont définis différemment - l'un est un typmod, l'autre une contrainte

```
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)
```

Dans geometry_columns, les deux s'enregistrent correctement

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

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

Cependant, si nous devions créer une vue comme celle-ci

```
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';
```

La colonne de la vue géométrique basée sur le modèle s'enregistre correctement, mais pas celle basée sur les contraintes.

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

Cela pourrait changer dans les prochaines versions de PostGIS, mais pour l'instant, pour forcer la colonne de vue basée sur les contraintes à s'enregistrer correctement, vous devez procéder comme suit :

```
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';
```

```
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 Chargement des données spatiales

Une fois que vous avez créé une table spatiale, vous êtes prêt à charger des données spatiales dans la base de données. Il existe deux façons d'intégrer des données spatiales dans une base de données PostGIS/PostgreSQL : à l'aide d'instructions SQL formatées ou à l'aide de l'utilitaire qui permet de charger des Shapefiles.

4.7.1 Utilisation de SQL pour charger des données

Si les données spatiales peuvent être converties en une représentation textuelle (WKT ou WKB), l'utilisation de SQL pourrait être le moyen le plus simple d'introduire les données dans PostGIS. Les données peuvent être chargées en masse dans PostGIS/-PostgreSQL en chargeant un fichier texte d'instructions SQL `INSERT` à l'aide de l'utilitaire SQL `psql`.

Un fichier de chargement SQL (`roads.sql` par exemple) peut ressembler à ceci :

```
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;
```

Le fichier SQL peut être chargé dans PostgreSQL en utilisant `psql` :

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

4.7.2 Utilisation de l'utilitaire qui permet de charger des fichiers Shapefile

Le chargeur de données `shp2pgsql` convertit les Shapefiles en SQL pour les insérer dans une base de données PostGIS/Post-greSQL au format geometry ou geography. Le chargeur a plusieurs modes de fonctionnement sélectionnés par des indicateurs de ligne de commande.

Il existe également une interface graphique `shp2pgsql-gui` qui offre la plupart des options du chargeur en ligne de commande. Cette interface peut être plus facile à utiliser pour un chargement ponctuel sans script ou si vous êtes novice en matière de PostGIS. Il peut également être configuré comme un plugin pour PgAdminIII.

(c|a|d|p) Ces options s'excluent mutuellement :

-c Crée une nouvelle table et la remplit à partir du fichier Shapefile. *C'est le mode par défaut.*

-a Ajoute les données du fichier Shapefile dans la table de la base de données. Notez que pour utiliser cette option afin de charger plusieurs fichiers, les fichiers doivent avoir les mêmes attributs et les mêmes types de données.

-d Supprime la table de la base de données avant de créer une nouvelle table avec les données du fichier Shapefile.

-p Produit uniquement le code SQL de création de la table, sans ajouter de données réelles. Cette option peut être utilisée si vous souhaitez séparer complètement les étapes de création de la table et de chargement des données.

-? Afficher l'écran d'aide.

-D Utilise le format "dump" de PostgreSQL pour les données de sortie. Ce format peut être combiné avec -a, -c et -d. Il est beaucoup plus rapide à charger que le format SQL "insert" par défaut. Utilisez-le pour les très grands ensembles de données.

-s [<FROM_SRID>:]<SRID> Crée et remplit les tables de géométrie avec le SRID spécifié. Il est possible de spécifier que le fichier de forme d'entrée utilise le FROM_SRID donné, auquel cas les géométries seront reprojetées dans le SRID cible.

-k Conserver la casse des identifiants (colonne, schéma et attributs). Notez que les attributs dans Shapefile sont tous en majuscules.

-i Contraindre tous les entiers à des entiers 32 bits standard, ne pas créer de bigints 64 bits, même si la signature de l'en-tête DBF semble le justifier.

-I Créer un index GiST sur la colonne géométrie.

-m -m `nom_de_fichier` Spécifier un fichier contenant un ensemble de correspondances entre les noms de colonnes (longs) et les noms de colonnes DBF à 10 caractères. Le contenu du fichier est une ou plusieurs lignes de deux noms séparées par un espace blanc, sans espace de départ ni de fin. Par exemple :

```
COLUMNNAME DBFFIELD1
AVERYLONGCOLUMNNAME DBFFIELD2
```

-S Génère des géométries simples au lieu de géométries MULTI. Ne réussira que si toutes les géométries sont en fait simples (c'est-à-dire un MULTIPOLYGON avec une seule enveloppe, ou un MULTIPOINT avec un seul sommet).

-t <dimensionality> Force la géométrie de sortie à avoir la dimensionnalité spécifiée. Utilisez les chaînes de caractères suivantes pour indiquer la dimensionnalité : 2D, 3DZ, 3DM, 4D.

Si l'entrée a moins de dimensions que celles spécifiées, la sortie aura ces dimensions remplies avec des zéros. Si l'entrée a plus de dimensions que celles spécifiées, les dimensions non désirées seront supprimées.

-w Sortie au format WKT, au lieu de WKB. Notez que cela peut introduire des dérives de coordonnées dues à la perte de précision.

-e Exécuter chaque instruction séparément, sans utiliser de transaction. Cela permet de charger la majorité des bonnes données lorsqu'il y a quelques mauvaises géométries qui génèrent des erreurs. Notez que cela ne peut pas être utilisé avec l'option -D car le format "dump" utilise toujours une transaction.

-W <encoding> Spécifie l'encodage des données d'entrée (fichier dbf). Lorsqu'il est utilisé, tous les attributs du fichier dbf sont convertis en UTF8 à partir de l'encodage spécifié. La sortie SQL résultante contiendra une commande `SET CLIENT_ENCODING to UTF8`, afin que le backend puisse reconvertir de UTF8 à n'importe quel encodage que la base de données est configurée pour utiliser en interne.

-N <policy> Politique de gestion des géométries NULL (insert*, skip, abort)

-n -n Importer uniquement le fichier DBF. Si vos données n'ont pas de fichier de forme correspondant, il passera automatiquement à ce mode et ne chargera que le fichier DBF. Cette option n'est donc nécessaire que si vous disposez d'un fichier de forme complet et que vous ne voulez que les données d'attributs et pas de géométrie.

-G Utiliser le type geography au lieu de geometry (nécessite des données lon/lat) dans WGS84 long lat (SRID=4326)

-T <tablespace> Spécifiez le tablespace pour la nouvelle table. Les index utiliseront toujours le tablespace par défaut à moins que le paramètre -X ne soit également utilisé. La documentation de PostgreSQL contient une bonne description de l'utilisation des tablespaces personnalisés.

-X <tablespace> Spécifier le tablespace pour les index de la nouvelle table. Ceci s'applique à l'index de clé primaire et à l'index spatial GIST si -I est également utilisé.

-Z Lorsqu'il est utilisé, cet indicateur empêche la génération des instructions `ANALYZE`. Sans l'option -Z (comportement par défaut), les instructions `ANALYZE` seront générées.

Un exemple de session utilisant le chargeur pour créer un fichier d'entrée et le charger peut ressembler à ceci :

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

Une conversion et un chargement peuvent être effectués en une seule étape à l'aide de pipes UNIX :

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

4.8 Extraction de données spatiales

Les données spatiales peuvent être extraites de la base de données à l'aide de SQL ou de Shapefile dumper. La section sur SQL présente certaines des fonctions disponibles pour effectuer des comparaisons et des requêtes sur les tables spatiales.

4.8.1 Utiliser SQL pour extraire des données

La manière la plus simple d'extraire des données spatiales de la base de données consiste à utiliser une requête SQL `SELECT` pour définir l'ensemble de données à extraire et à transférer les colonnes résultantes dans un fichier texte analysable :

```
db=# SELECT road_id, ST_AsText(road_geom) AS geom, road_name FROM roads;

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)
```

Il peut arriver qu'une restriction soit nécessaire pour réduire le nombre d'enregistrements renvoyés. Dans le cas de restrictions basées sur des attributs, utilisez la même syntaxe SQL que pour une table non spatiale. Dans le cas de restrictions spatiales, les fonctions suivantes sont utiles :

ST_Intersects Cette fonction indique si deux géométries partagent un espace.

= Cette fonction permet de vérifier si deux géométries sont géométriquement identiques. Par exemple, si 'POLYGON((0 0,1 1,1 0,0 0))' est identique à 'POLYGON((0 0,1 1,1 0,0 0))' (c'est le cas).

Vous pouvez ensuite utiliser ces opérateurs dans des requêtes. Notez que lorsque vous spécifiez des géométries et des box sur la ligne de commande SQL, vous devez explicitement transformer les représentations de chaînes de caractères en fonction géométrique. Le 312 est un système de référence spatiale fictif qui correspond à nos données. Ainsi, par exemple, le 312 est un système de référence spatiale fictif qui correspond à nos données :

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

La requête ci-dessus renverrait le seul enregistrement de la table "ROADS_GEOM" dont la géométrie est égale à cette valeur.

Vérifier si certaines routes passent dans la zone définie par un polygone :

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

La requête spatiale la plus courante sera probablement une requête "basée sur un cadre", utilisée par les logiciels clients, tels que les navigateurs de données et les cartographes web, pour saisir un "cadre de carte" d'une valeur de données pour l'affichage.

Lorsque vous utilisez l'opérateur "&&", vous pouvez spécifier soit une BOX3D comme élément de comparaison, soit une GEOMETRY. Toutefois, lorsque vous spécifiez une GEOMETRY, c'est sa boîte de délimitation qui sera utilisée pour la comparaison.

En utilisant un objet "BOX3D" pour le cadre, une telle requête se présente comme suit :

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

Notez l'utilisation du SRID 312 pour spécifier la projection de l'enveloppe.

4.8.2 Utilisation de Shapefile Dumper

Le table dumper `pgsql2shp` se connecte à la base de données et convertit une table (éventuellement définie par une requête) en un fichier shape. La syntaxe de base est la suivante :

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

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

Les options de la ligne de commande sont les suivantes :

-f <filename> Écriture de la sortie dans un fichier particulier.

-h <host> L'hôte de la base de données auquel se connecter.

-p <port> Le port auquel se connecter sur l'hôte de la base de données.

-P <password> Le mot de passe à utiliser lors de la connexion à la base de données.

-u <user> Le nom d'utilisateur à utiliser lors de la connexion à la base de données.

-g <geometry column> Dans le cas de tableaux comportant plusieurs colonnes géométriques, la colonne géométrique à utiliser lors de l'écriture du fichier de forme.

-b Utiliser un curseur binaire. L'opération sera plus rapide, mais ne fonctionnera pas si un attribut NON-geometry de la table n'est pas converti en texte.

-r Mode brut. Ne pas supprimer le champ `gid`, ni échapper les noms de colonnes.

-m filename Remapper les identifiants en noms de dix caractères. Le contenu du fichier est constitué de lignes de deux symboles séparées par un seul espace blanc, sans espace de départ ni de fin : VERYLONGSYMBOL SHORTONE ANOTHERYLONGSYMBOL SHORTER etc.

4.9 Index spatiaux

Les index spatiaux permettent d'utiliser une base de données spatiales pour de grands ensembles de données. Sans indexation, la recherche de caractéristiques nécessite un balayage séquentiel de chaque enregistrement de la base de données. L'indexation accélère la recherche en organisant les données dans une structure qui peut être rapidement parcourue pour trouver les enregistrements correspondants.

La méthode d'indexation B-tree couramment utilisée pour les données d'attributs n'est pas très utile pour les données spatiales, car elle ne permet de stocker et d'interroger les données que dans une seule dimension. Les données telles que la géométrie (qui a 2 dimensions ou plus) requièrent une méthode d'indexation qui supporte les requêtes sur toutes les dimensions des données. L'un des principaux avantages de PostgreSQL pour le traitement des données spatiales est qu'il offre plusieurs types de méthodes d'indexation qui fonctionnent bien pour les données multidimensionnelles : GiST, BRIN et SP-GiST.

- Les index **GiST (Generalized Search Tree)** décomposent les données en "choses à côté", "choses qui se chevauchent", "choses qui sont à l'intérieur" et peuvent être utilisés sur un large éventail de types de données, y compris les données SIG. PostGIS utilise un index R-Tree implémenté au-dessus de GiST pour indexer les données spatiales. GiST est la méthode d'indexation spatiale la plus couramment utilisée et la plus polyvalente, et offre de très bonnes performances en matière de requêtes.
- Les index **BRIN (Block Range Index)** fonctionnent en résumant l'étendue spatiale des plages d'enregistrements de la table. La recherche s'effectue par le biais d'un balayage des plages. L'index BRIN n'est approprié que pour certains types de données (triées dans l'espace, avec des mises à jour peu fréquentes ou inexistantes). Mais il permet une création d'index beaucoup plus rapide et une taille d'index beaucoup plus petite.
- **SP-GiST (Space-Partitioned Generalized Search Tree)** est une méthode d'indexation générique qui prend en charge les arbres de recherche partitionnés tels que les arbres quadratiques, les arbres k-d et les arbres radix (tries).

Les index spatiaux ne stockent que la boîte de délimitation des géométries. Les requêtes spatiales utilisent l'index comme **filtre primaire** pour déterminer rapidement un ensemble de géométries correspondant potentiellement à la condition de la requête. La plupart des requêtes spatiales nécessitent un **filtre secondaire** qui utilise une fonction de prédicat spatial pour tester une condition spatiale plus spécifique. Pour plus d'informations sur les requêtes avec des prédicats spatiaux, voir Section 5.2.

Voir également la section de l'atelier PostGIS sur les index spatiaux et le manuel de PostgreSQL.

4.9.1 Index GiST

GiST signifie "Generalized Search Tree" (arbre de recherche généralisé) et constitue une forme générique d'indexation pour les données multidimensionnelles. PostGIS utilise un index R-Tree implémenté au-dessus de GiST pour indexer les données spatiales. GiST est la méthode d'indexation spatiale la plus couramment utilisée et la plus polyvalente, et offre de très bonnes performances en matière de requêtes. D'autres implémentations de GiST sont utilisées pour accélérer les recherches sur toutes sortes de structures de données irrégulières (tableaux d'entiers, données spectrales, etc.) qui ne se prêtent pas à l'indexation B-Tree normale. Pour plus d'informations, voir le manuel PostgreSQL.

Dès qu'une table de données spatiales dépasse quelques milliers de lignes, vous devez créer un index pour accélérer les recherches spatiales dans les données (sauf si toutes vos recherches sont basées sur des attributs, auquel cas vous devez créer un index normal sur les champs d'attributs).

La syntaxe pour construire un index GiST sur une colonne "geometry" est la suivante :

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

La syntaxe ci-dessus permet toujours de créer un index 2D. Pour obtenir un index à n dimensions pour le type de géométrie, vous pouvez en créer un à l'aide de cette syntaxe :

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

La construction d'un index spatial est un exercice de calcul intensif. Il bloque également l'accès en écriture à votre table pendant le temps de sa création, donc sur un système de production, vous voudrez peut-être le faire d'une manière plus lente, en tenant compte de la notion de CONCURRENCE :

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

Après avoir construit un index, il est parfois utile de forcer PostgreSQL à collecter les statistiques de la table, qui sont utilisées pour optimiser les plans de requête :

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

4.9.2 Index BRIN

BRIN signifie "Block Range Index". C'est une méthode d'indexation générale introduite dans PostgreSQL 9.5. BRIN est une méthode d'indexation *lossy*, ce qui signifie qu'une vérification secondaire est nécessaire pour confirmer qu'un enregistrement correspond à une condition de recherche donnée (ce qui est le cas pour tous les index spatiaux fournis). Elle permet une création d'index beaucoup plus rapide et une taille d'index beaucoup plus petite, avec des performances de lecture raisonnables. Son objectif principal est de permettre l'indexation de très grandes tables sur des colonnes qui ont une corrélation avec leur emplacement physique dans la table. Outre l'indexation spatiale, BRIN peut accélérer les recherches sur divers types de structures de données d'attributs (entiers, tableaux, etc.). Pour plus d'informations, voir le Manuel de PostgreSQL.

Dès qu'une table spatiale dépasse quelques milliers de lignes, vous voudrez construire un index pour accélérer les recherches spatiales dans les données. Les index GiST sont très performants tant que leur taille ne dépasse pas la quantité de mémoire vive disponible pour la base de données, et tant que vous pouvez vous permettre la taille de stockage de l'index et le coût de la mise à jour de l'index en écriture. Sinon, pour les très grandes tables, l'index BRIN peut être considéré comme une alternative.

Un index BRIN stocke la boîte englobant toutes les géométries contenues dans les lignes d'un ensemble contigu de blocs de table, appelé *plage de blocs*. Lors de l'exécution d'une requête à l'aide de l'index, les plages de blocs sont examinées pour trouver celles qui recoupent l'étendue de la requête. Cette méthode n'est efficace que si les données sont physiquement ordonnées de manière à ce que les zones de délimitation des plages de blocs se chevauchent le moins possible (et, dans l'idéal, s'excluent mutuellement). L'index qui en résulte est de très petite taille, mais il est généralement moins performant en lecture qu'un index GiST sur les mêmes données.

La construction d'un index BRIN est beaucoup moins gourmande en ressources CPU que la construction d'un index GiST. Il est courant de constater qu'un index BRIN est dix fois plus rapide à construire qu'un index GiST pour les mêmes données. Et comme un index BRIN ne stocke qu'une seule boîte englobante pour chaque plage de blocs de table, il est courant d'utiliser jusqu'à mille fois moins d'espace disque qu'un index GiST.

Vous pouvez choisir le nombre de blocs à résumer dans une plage. Si vous diminuez ce nombre, l'index sera plus volumineux mais offrira probablement de meilleures performances.

Pour que le BRIN soit efficace, les données de la table doivent être stockées dans un ordre physique qui minimise le chevauchement de l'étendue du bloc. Il se peut que les données soient déjà triées de manière appropriée (par exemple, si elles sont chargées à partir d'un autre jeu de données déjà trié dans l'ordre spatial). Dans le cas contraire, il est possible de trier les données en fonction d'une clé spatiale unidimensionnelle. Une façon de procéder consiste à créer une nouvelle table triée par les valeurs géométriques (qui, dans les versions récentes de PostGIS, utilisent un classement efficace par courbe de Hilbert) :

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

Il est également possible de trier les données en place en utilisant un GeoHash comme index (temporaire) et en effectuant un regroupement sur cet index :

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

La syntaxe pour construire un index BRIN sur une colonne `geometry` est la suivante :

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

La syntaxe ci-dessus permet de créer un index en 2D. Pour construire un index à trois dimensions, utilisez la syntaxe suivante :

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

Vous pouvez également obtenir un index quadridimensionnel à l'aide de la classe d'opérateurs 4D :

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

Les commandes ci-dessus utilisent le nombre par défaut de blocs dans une plage, qui est de 128. Pour spécifier le nombre de blocs à résumer dans une plage, utilisez la syntaxe suivante

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

Gardez à l'esprit qu'un index BRIN ne stocke qu'une entrée d'index pour un grand nombre de lignes. Si votre table stocke des géométries avec un nombre variable de dimensions, il est probable que l'index qui en résultera sera peu performant. Vous pouvez éviter cette pénalité de performance en choisissant la classe d'opérateur avec le plus petit nombre de dimensions des géométries stockées

Le type de données `geography` est pris en charge pour l'indexation BRIN. La syntaxe pour construire un index BRIN sur une colonne géographique est la suivante :

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

La syntaxe ci-dessus permet de créer un index 2D pour les objets géospatiaux sur le sphéroïde.

Actuellement, seul le "support d'inclusion" est fourni, ce qui signifie que seuls les opérateurs `&&`, `~` et `@` peuvent être utilisés pour les cas 2D (à la fois pour `geometry` et `geography`), et seulement l'opérateur `&&&` pour les géométries 3D. Il n'y a actuellement pas de support pour les recherches kNN.

Une différence importante entre BRIN et les autres types d'index est que la base de données ne maintient pas l'index de manière dynamique. Les modifications apportées aux données spatiales de la table sont simplement ajoutées à la fin de l'index. Cela entraîne une dégradation des performances de recherche de l'index au fil du temps. L'index peut être mis à jour en effectuant un `VACUUM`, ou en utilisant une fonction spéciale `brin_summarize_new_values(regclass)`. C'est la raison pour laquelle BRIN peut être plus approprié pour les données qui sont en lecture seule ou qui ne changent que rarement. Pour plus d'informations, consultez le manuel.

En résumé, l'utilisation de BRIN pour les données spatiales :

- Le temps de construction de l'index est très rapide et la taille de l'index est très réduite.
- Le temps d'interrogation de l'index est plus lent que celui de GiST, mais reste très acceptable.
- Nécessite que les données du tableau soient triées dans un ordre spatial.
- Nécessite une maintenance manuelle de l'index.
- Plus approprié pour les tables de très grande taille, avec peu ou pas de chevauchement (par exemple des points), qui sont statiques ou qui changent peu souvent.
- Plus efficace pour les requêtes qui renvoient un nombre relativement important d'enregistrements de données.

4.9.3 Index SP-GiST

SP-GiST signifie "Space-Partitioned Generalized Search Tree" et est une forme générique d'indexation pour les types de données multidimensionnelles qui prend en charge les arbres de recherche partitionnés, tels que les arbres quadratiques, les arbres k-d et les arbres radix (tries). La caractéristique commune de ces structures de données est qu'elles divisent de manière répétée l'espace de recherche en partitions qui ne sont pas nécessairement de taille égale. Outre l'indexation spatiale, SP-GiST est utilisé pour accélérer les recherches sur de nombreux types de données, telles que le routage téléphonique, le routage ip, la recherche de substrats, etc. Pour plus d'informations, voir le manuel PostgreSQL.

Comme pour les index GiST, les index SP-GiST sont avec perte, dans le sens où ils stockent la boîte englobante qui entoure les objets spatiaux. Les index SP-GiST peuvent être considérés comme une alternative aux index GiST.

Lorsqu'une table de données SIG dépasse quelques milliers de lignes, un index SP-GiST peut être utilisé pour accélérer les recherches spatiales dans les données. La syntaxe pour construire un index SP-GiST sur une colonne "géométrie" est la suivante :

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

La syntaxe ci-dessus permet de créer un index bidimensionnel. Un index tridimensionnel pour le type de géométrie peut être créé à l'aide de la classe d'opérateurs 3D :

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

La création d'un index spatial est une opération très gourmande en ressources informatiques. Elle bloque également l'accès en écriture à votre table pendant la durée de la création, donc sur un système de production, vous voudrez peut-être le faire d'une manière plus lente, en tenant compte de la notion de CONCURRENCE :

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

Après avoir construit un index, il est parfois utile de forcer PostgreSQL à collecter les statistiques de la table, qui sont utilisées pour optimiser les plans de requête :

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

Un index SP-GiST peut accélérer les requêtes impliquant les opérateurs suivants :

- <<, &<, &>, >>, <<|, &<|, |&>, |>>, &&, @>, <@, et ~=, pour les index à 2 dimensions,
- &/&, ~==, @>>, et <<@, pour les index à 3 dimensions.

Il n'y a pas de support pour les recherches kNN pour le moment.

4.9.4 Optimisation de l'utilisation de l'index

Normalement, les index accélèrent de manière invisible l'accès aux données : une fois qu'un index est construit, le planificateur de requêtes de PostgreSQL décide automatiquement quand l'utiliser pour améliorer les performances de la requête. Mais dans certaines situations, le planificateur ne choisit pas d'utiliser les index existants, et les requêtes finissent par utiliser des balayages séquentiels lents au lieu d'un index spatial.

Si vous constatez que vos index spatiaux ne sont pas utilisés, il y a plusieurs choses que vous pouvez faire :

- Examinez le plan de requête et vérifiez que votre requête calcule effectivement ce dont vous avez besoin. Un JOIN erroné, oublié ou vers la mauvaise table, peut récupérer plusieurs fois des enregistrements de la table de manière inattendue. Pour obtenir le plan de la requête, exécutez avec `EXPLAIN` devant la requête.
- Assurez-vous que des statistiques sont collectées sur le nombre et la distribution des valeurs dans une table, afin de fournir au planificateur de requêtes de meilleures informations pour prendre des décisions concernant l'utilisation de l'index. La **VACUUM ANALYZE** calculera les deux.

  Vous devriez de toute façon vacuum régulièrement vos bases de données. De nombreux administrateurs de bases de données PostgreSQL exécutent **VACUUM** en tant que tâche cron en dehors des heures de pointe sur une base régulière.

- Si vacuum ne suffit pas, vous pouvez temporairement forcer le planificateur à utiliser les informations d'index en utilisant la commande **SET ENABLE_SEQSCAN TO OFF;**. De cette façon, vous pouvez vérifier si le planificateur est capable de générer un plan de requête accéléré par l'index pour votre requête. Vous ne devez utiliser cette commande qu'à des fins de débogage ; en règle générale, le planificateur sait mieux que vous quand utiliser les index. Une fois que vous avez exécuté votre requête, n'oubliez pas de lancer la commande **SET ENABLE_SEQSCAN TO ON;** afin que le planificateur fonctionne normalement pour les autres requêtes.
- Si **SET ENABLE_SEQSCAN TO OFF;** aide votre requête à s'exécuter plus rapidement, votre Postgres n'est probablement pas adapté à votre matériel. Si vous trouvez que le planificateur se trompe sur le coût des balayages séquentiels par rapport aux balayages d'index, essayez de réduire la valeur de `RANDOM_PAGE_COST` dans `postgresql.conf`, ou utilisez **SET RANDOM_PAGE_COST TO 1.1;**. La valeur par défaut de `RANDOM_PAGE_COST` est 4.0. Essayez de la fixer à 1.1 (pour les disques SSD) ou à 2.0 (pour les disques magnétiques rapides). En diminuant la valeur, le planificateur est plus enclin à utiliser les balayages d'index.
- Si la **SET ENABLE_SEQSCAN TO OFF;** n'aide pas votre requête, il se peut qu'elle utilise une construction SQL que le planificateur Postgres n'est pas encore en mesure d'optimiser. Il peut être possible de réécrire la requête de manière à ce que le planificateur soit en mesure de la traiter. Par exemple, une sous-requête avec un SELECT en ligne peut ne pas produire un plan efficace, mais peut être réécrite en utilisant un JOIN LATERAL.

Pour plus d'informations, voir la section du manuel Postgres sur Query Planning.

Chapter 5

Requêtes spatiales

La *raison d'être* des bases de données spatiales est de réaliser à l'intérieur de la base de données des requêtes qui normalement nécessiteraient des fonctionnalités d'un SIG Desktop. Utiliser PostGIS requiert en effet la connaissance de quelles fonctions spatiales sont disponibles, comment les utiliser dans une requête, et de s'assurer de la mise en place adéquate des index, pour une bonne performance.

5.1 Déterminer les relations spatiales

Les relations spatiales indiquent comment deux géométries interagissent l'une avec l'autre. Elles constituent une fonctionnalité fondamentale pour effectuer des requêtes sur des géométries.

5.1.1 Modèle à 9 intersections dimensionnellement étendu

Selon le OpenGIS Simple Features Implementation Specification for SQL, "l'approche de base pour comparer deux géométries consiste à effectuer des tests par paire des intersections entre les intérieurs, les limites et les extérieurs des deux géométries et à classer la relation entre les deux géométries sur la base des entrées de la matrice "intersection" résultante."

Dans la théorie de la topologie des ensembles de points, les points d'une géométrie intégrée dans un espace à deux dimensions sont classés en trois ensembles :

Limites

La limite d'une géométrie est l'ensemble des géométries de la dimension immédiatement inférieure. Pour les `POINT`, qui ont une dimension de 0, la limite est l'ensemble vide. La limite d'une `LINESTRING` est constituée par ses deux extrémités. Pour les `POLYGON`, la limite est le réseau de lignes des anneaux extérieur et intérieur.

Intérieur

L'intérieur d'une géométrie est constitué des points de la géométrie qui ne se trouvent pas dans la limite. Pour les `POINT`, l'intérieur est le point lui-même. L'intérieur d'une `LINESTRING` est l'ensemble des points situés entre les extrémités. Pour les `POLYGON`, l'intérieur est la surface aréale à l'intérieur du polygone.

Extérieur

L'extérieur d'une géométrie est le reste de l'espace dans lequel la géométrie est intégrée ; en d'autres termes, tous les points qui ne se trouvent pas à l'intérieur ou sur la limite de la géométrie. Il s'agit d'une surface non fermée à 2 dimensions.

Le Dimensionally Extended 9-Intersection Model (DE-9IM) décrit la relation spatiale entre deux géométries en spécifiant les dimensions des 9 intersections entre les ensembles ci-dessus pour chaque géométrie. Les dimensions des intersections peuvent être représentées formellement dans une **matrice d'intersections** de 3x3.

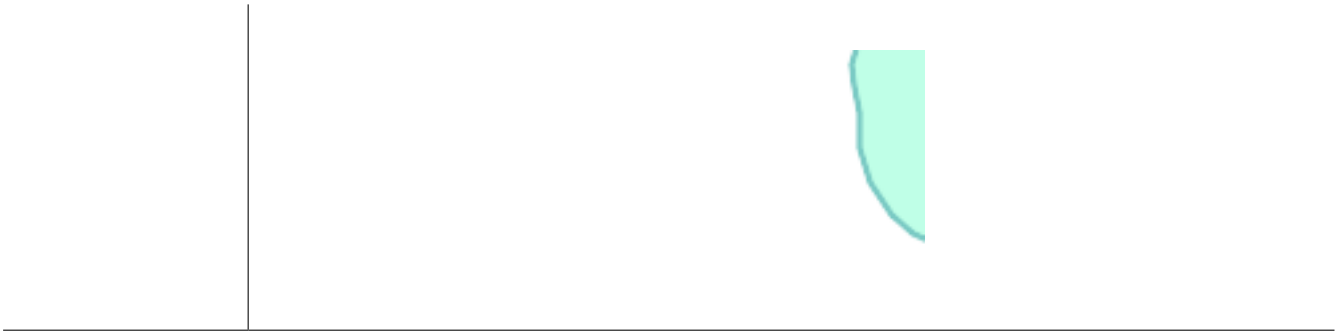
Pour une géométrie *g*, les *Limites*, *Intérieures*, et *Extérieures* sont désignés par la notation *I(g)*, *B(g)*, et *E(g)*. En outre, *dim(s)* désigne la dimension d'un ensemble *s* dont le domaine est `{0,1,2,F}` :

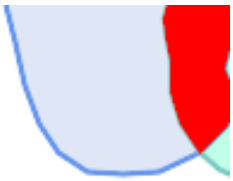
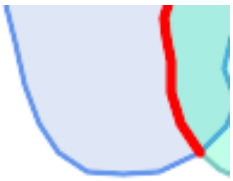
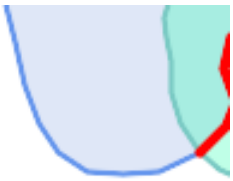
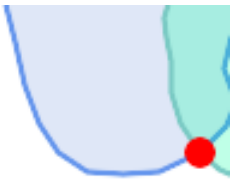
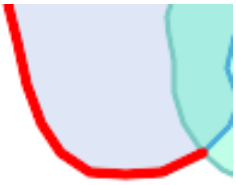
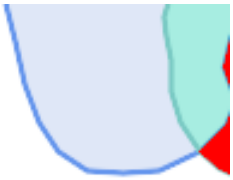
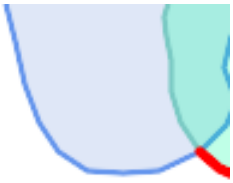
- `0` => point
- `1` => ligne
- `2` => area
- `F` => ensemble vide

En utilisant cette notation, la matrice d'intersection pour deux géométries *a* et *b* est :

	Intérieur	**Limite**	**Extérieur**
Intérieur	*dim(I(a) ∩ I(b))*	*dim(I(a) ∩ B(b))*	*dim(I(a) ∩ E(b))*
Limite	*dim(B(a) ∩ I(b))*	*dim(B(a) ∩ B(b))*	*dim(B(a) ∩ E(b))*
Extérieur	*dim(E(a) ∩ I(b))*	*dim(E(a) ∩ B(b))*	*dim(E(a) ∩ E(b))*

Visuellement, pour deux géométries polygonales qui se chevauchent, cela ressemble à ce qui suit :



	Intérieur	**Limite**	**Extérieur**
Intérieur	 *dim(I(a) ∩ I(b))* = **2**	 *dim(I(a) ∩ B(b)* = **1**	*dim(I(a) ∩ E(b))* = **2**
Limite	 *dim(B(a) ∩ I(b))* = **1**	 *dim(B(a) ∩ B(b))* = **0**	 *dim(B(a) ∩ E(b))* = **1**
Extérieur	 *dim(E(a) ∩ I(b))* = **2**	 *dim(E(a) ∩ B(b))* = **1**	*dim(E(a) ∩ E(b)* = **2**

En lisant de gauche à droite et de haut en bas, la matrice d'intersection est représentée par la chaîne de texte "**212101212**".

Pour plus d'informations, voir :

- OpenGIS Simple Features Implementation Specification for SQL (version 1.1, section 2.1.13.2)
- Wikipedia : Dimensionally Extended Nine-Intersection Model (DE-9IM)
- GeoTools : Théorie des ensembles de points et matrice DE-9IM

5.1.2 Relations spatiales nommées

Pour faciliter la détermination des relations spatiales communes, l'OGC SFS définit un ensemble de *prédicats de relations spatiales*. PostGIS fournit ces prédicats sous la forme des fonctions ST_Contains, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within. Il définit également les prédicats de relation non standard ST_Covers, ST_CoveredBy, et ST_ContainsProperly.

Les prédicats spatiaux sont généralement utilisés comme conditions dans les clauses SQL `WHERE` ou `JOIN`. Les prédicats spatiaux nommés utilisent automatiquement un index spatial s'il existe, de sorte qu'il n'est pas nécessaire d'utiliser également l'opérateur de boîte de délimitation `&&`. Par exemple :

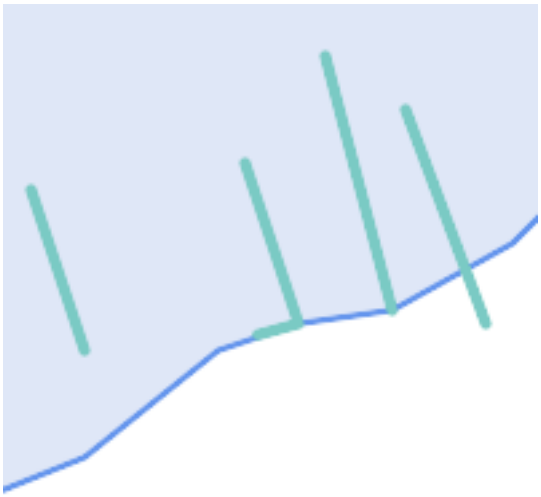
```
SELECT city.name, state.name, city.geom
FROM city JOIN state ON ST_Intersects(city.geom, state.geom);
```

Pour plus de détails et d'illustrations, voir l' Atelier PostGIS.

5.1.3 Relations spatiales générales

Dans certains cas, les relations spatiales nommées sont insuffisantes pour fournir une condition de filtrage spatial souhaitée.

Prenons l'exemple d'un ensemble de données linéaires représentant un réseau routier. Il peut être nécessaire d'identifier tous les segments de route qui se croisent, non pas en un point, mais en une ligne (peut-être pour valider une règle commerciale). Dans ce cas, ST_Crosses ne fournit pas le filtre spatial nécessaire, puisque pour les caractéristiques linéaires, il renvoie `true` uniquement lorsqu'elles se croisent en un point.
Une solution en deux étapes consisterait à calculer d'abord l'intersection réelle (ST_Intersection) des paires de lignes routières qui se croisent dans l'espace (ST_Intersects), puis vérifier si le ST_GeometryType de l'intersection est "`LINESTRING`" (en traitant correctement les cas qui renvoient des `GEOMETRYCOLLECTION`s de `[MULTI]POINT`s, `[MULTI]LINESTRING`s, etc.).
Il est évident qu'une solution plus simple et plus rapide est souhaitable.



Un deuxième exemple est la localisation des quais qui coupent les limites d'un lac sur une ligne et dont l'une des extrémités se trouve sur le rivage. En d'autres termes, lorsqu'un quai est situé à l'intérieur d'un lac mais n'est pas complètement contenu par celui-ci, qu'il coupe la limite d'un lac sur une ligne et que l'une des extrémités du quai se trouve à l'intérieur ou sur la limite du lac. Il est possible d'utiliser une combinaison de prédicats spatiaux pour trouver les caractéristiques requises :

- ST_Contains(lake, wharf) = TRUE
- ST_ContainsProperly(lake, wharf) = FALSE
- ST_GeometryType(ST_Intersection(wharf, lake)) = 'LINESTRING'
- ST_NumGeometries(ST_Multi(ST_Intersection(ST_Boundary(wharf), ST_Boundary(lake)))) = 1

  ... mais inutile de dire que c'est assez compliqué.

Ces exigences peuvent être satisfaites en calculant la matrice d'intersection DE-9IM complète. PostGIS fournit la fonction ST_Relate pour ce faire :

```
SELECT ST_Relate( 'LINESTRING (1 1, 5 5)',
                  'POLYGON ((3 3, 3 7, 7 7, 7 3, 3 3))' );
st_relate
-----------
1010F0212
```

Pour tester une relation spatiale particulière, un **modèle de matrice d'intersection** est utilisé. Il s'agit de la représentation matricielle augmentée des symboles supplémentaires `{T,*}` :

- `T` => la dimension d'intersection est non vide ; c'est-à-dire qu'elle se trouve dans `{0,1,2}`
- `*` => indifférent

Les modèles de matrice d'intersection permettent d'évaluer des relations spatiales spécifiques de manière plus succincte. Les fonctions ST_Relate et ST_RelateMatch peuvent être utilisées pour tester les modèles de matrice d'intersection. Pour le premier exemple ci-dessus, le modèle de matrice d'intersection spécifiant deux lignes se croisant dans une ligne est '**1*1***1****' :

```
-- Find road segments that intersect in a line
SELECT a.id
FROM roads a, roads b
WHERE a.id != b.id
```

```
      AND a.geom && b.geom
      AND ST_Relate(a.geom, b.geom, '1*1***1**');
```

Pour le deuxième exemple, le modèle de matrice d'intersection spécifiant une ligne située en partie à l'intérieur et en partie à l'extérieur d'un polygone est "**102101FF2**" :

```
-- Find wharves partly on a lake's shoreline
SELECT a.lake_id, b.wharf_id
FROM lakes a, wharfs b
WHERE a.geom && b.geom
      AND ST_Relate(a.geom, b.geom, '102101FF2');
```

5.2 Utilisation des index spatiaux

Lors de la construction de requêtes utilisant des conditions spatiales, il est important, pour de meilleures performances, de s'assurer qu'un index spatial est utilisé, s'il existe (voir Section 4.9). Pour ce faire, un opérateur spatial ou une fonction sensible à l'index doit être utilisé dans une clause `WHERE` ou `ON` de la requête.

Les opérateurs spatiaux comprennent les opérateurs de boîte de délimitation (dont le plus couramment utilisé est && ; voir Section 7.10.1 pour la liste complète) et les opérateurs de distance utilisés dans les requêtes sur les plus proches voisins (le plus courant étant <-> ; voir Section 7.10.2 pour la liste complète.)

Les fonctions tenant compte de l'index ajoutent automatiquement un opérateur de boîte de délimitation à la condition spatiale. Les fonctions tenant compte de l'index comprennent les prédicats de relation spatiale nommés ST_Contains, ST_ContainsProperly, ST_CoveredBy, ST_Covers, ST_Crosses, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within, ST_Within, et ST_3DIntersects, et les prédicats de distance ST_DWithin, ST_DFullyWithin, ST_3DDFullyWithin, et ST_3DDWithin .)

Les fonctions telles que ST_Distance n'utilisent *pas* les index pour optimiser leur fonctionnement. Par exemple, la requête suivante serait assez lente sur une grande table :

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

Cette requête sélectionne toutes les géométries de la `geom_table` qui se trouvent à moins de 100 unités du point (100000, 200000). Elle sera lente car elle calcule la distance entre chaque point du tableau et le point spécifié, c'est-à-dire qu'un calcul `ST_Distance()` est effectué pour **chaque** ligne du tableau.

Le nombre de lignes traitées peut être considérablement réduit en utilisant la fonction d'indexation ST_DWithin :

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

Cette requête sélectionne les mêmes géométries, mais de manière plus efficace. Ceci est possible en `ST_DWithin()` utilisant l'opérateur `&&` en interne sur une boîte de délimitation élargie de la géométrie de la requête. S'il existe un index spatial sur `geom`, le planificateur de requêtes reconnaîtra qu'il peut utiliser l'index pour réduire le nombre de lignes analysées avant de calculer la distance. L'index spatial permet d'extraire uniquement les enregistrements dont les boîtes de délimitation chevauchent l'étendue élargie et qui, par conséquent, *pourraient* se trouver à l'intérieur de la distance requise. La distance réelle est ensuite calculée pour confirmer l'inclusion de l'enregistrement dans l'ensemble des résultats.

Pour plus d'informations et d'exemples, voir l' atelier de travail PostGIS.

5.3 Exemples de SQL spatial

Les exemples de cette section utilisent une table de routes linéaires et une table de limites de communes polygonales. La définition de la table `bc_roads` est la suivante :

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

La définition de la table `bc_municipality` est la suivante :

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

1. *Quelle est la longueur totale de toutes les routes, exprimée en kilomètres ?*

   Vous pouvez répondre à cette question à l'aide d'un code SQL très simple :

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

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

2. *Quelle est la superficie de la ville de Prince George, en hectares ?*

   Cette requête combine une condition d'attribut (sur le nom de la municipalité) avec un calcul spatial (de la surface du polygone) :

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

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

3. *Quelle est la plus grande municipalité de la province, en termes de superficie ?*

   Cette requête utilise une mesure spatiale comme valeur de référence. Il existe plusieurs façons d'aborder ce problème, mais la plus efficace est la suivante :

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

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

   Notez que pour répondre à cette question, nous devons calculer la surface de chaque polygone. Si nous faisions cela souvent, il serait logique d'ajouter une colonne de surface à la table qui pourrait être indexée pour des raisons de performance. En ordonnant les résultats dans une direction décroissante et en utilisant la commande "LIMIT" de PostgreSQL, nous pouvons facilement sélectionner la plus grande valeur sans utiliser une fonction d'agrégation comme MAX().

4. *Quelle est la longueur des routes entièrement comprises dans chaque municipalité ?*

   Il s'agit d'un exemple de "jointure spatiale", qui rassemble les données de deux tables (avec une jointure) en utilisant une interaction spatiale ("contenu") comme condition de jointure (plutôt que l'approche relationnelle habituelle de jointure sur une clé commune) :

```
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;

name                        | roads_km
----------------------------+------------------
SURREY                      | 1539.47553551242
VANCOUVER                   | 1450.33093486576
LANGLEY DISTRICT            | 833.793392535662
BURNABY                     | 773.769091404338
PRINCE GEORGE               | 694.37554369147
...
```

   Cette requête prend un certain temps, car chaque route du tableau est résumée dans le résultat final (environ 250 000 routes pour le tableau de l'exemple). Pour des ensembles de données plus petits (plusieurs milliers d'enregistrements sur plusieurs centaines), la réponse peut être très rapide.

5. *Créez un nouveau tableau avec toutes les routes de la ville de Prince George.*

   Il s'agit d'un exemple de "superposition", qui prend en compte deux tableaux et produit un nouveau tableau composé de résultats coupés dans l'espace. Contrairement à la "jointure spatiale" présentée ci-dessus, cette requête crée de nouvelles géométries. Une superposition est comme une jointure spatiale turbocompressée, et est utile pour des travaux d'analyse plus précis :

```
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. *Quelle est la longueur en kilomètres de "Douglas St" à Victoria ?*

```
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';

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

7. *Quel est le plus grand polygone municipal comportant un trou ?*

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

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

Chapter 6

Conseils sur les performances

6.1 Petites tables de grandes géométries

6.1.1 Description du problème

Les versions de PostgreSQL actuelles (y compris 8.0) souffrent d'une faiblesse optimiseur de requête relative les tables TOAST. Tables TOAST sont une sorte de «salle de l'extension" utilisé pour stocker de grandes valeurs (dans le sens de la taille des données) qui ne rentrent pas dans les pages de données normales (comme de longs textes, images ou des géométries complexes avec beaucoup de sommets), voir Documentation PostgreSQL pour TOAST pour plus d'informations).

Le problème apparaît s'il vous arrive d'avoir une table avec d'assez grandes géométries, mais pas beaucoup de lignes d'entre elles (comme un tableau contenant les frontières de tous les pays européens en haute résolution). Ensuite, le tableau lui-même est petit, mais il utilise beaucoup d'espace TOAST. Dans notre exemple, le cas, la table elle-même avait environ 80 lignes et seulement 3 pages de données utilisées, mais la table TOAST 8225 pages utilisé.

Maintenant émettre une requête en utilisant l'opérateur de géométrie && pour rechercher une boîte englobante qui correspond que très peu de ces lignes. Maintenant l'optimiseur de requêtes voit que la table n'a que 3 pages et 80 lignes. Il estime qu'une analyse séquentielle sur une telle petite table est beaucoup plus rapide que d'utiliser un index. Et alors il décide d'ignorer l'index GIST. Habituellement, cette estimation est correcte. Mais dans notre cas, l'opérateur && doit aller chercher chaque géométrie à partir du disque pour comparer les boîtes englobantes, et par conséquent la lecture de toutes les pages TOAST également.

Pour voir si vous souffrez de ce problème, utilisez la commande postgresql "EXPLAIN ANALYZE". Pour plus d'informations et les détails techniques, vous pouvez lire le fil de discussion sur la liste de diffusion sur les performances de PostgreSQL : http://archives.postgresql.org/pgsql-performance/2005-02/msg00030.php

et un fil d'actualités plus récent sur PostGIS https://lists.osgeo.org/pipermail/postgis-devel/2017-June/026209.html

6.1.2 Solutions de contournement

Les personnes de PostgreSQL essayent de résoudre ce problème en faisant l'estimation de la requête TOAST-courant. Pour l'instant, voici deux solutions :

La première solution consiste à forcer le planificateur de requêtes à utiliser l'index. Envoyer "SET enable_seqscan TO off;" au serveur avant d'émettre la requête. Cela force le planificateur de requêtes à éviter balayages séquentiels lorsque cela est possible. Donc, il utilise l'index GIST comme d'habitude. Mais cet indicateur doit être fixé à chaque connexion, et il provoque le planificateur de requêtes à faire des erreurs d'estimation dans les autres cas, vous devrez donc faire "SET POUR enable_seqscan sur;" après la requête.

La deuxième solution consiste à faire le balayage séquentielle aussi vite que le planificateur de requêtes pense. Ceci peut être réalisé en créant une colonne supplémentaire qui "cache" la bbox, et contre cette correspondance. Dans notre exemple, les commandes sont comme :

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

Modifiez maintenant votre requête pour utiliser l'opérateur && contre bbox au lieu de geom_column, comme suit :

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

Bien sûr, si vous changez ou ajoutez des lignes à mytable, vous devez garder la bbox "synchro". La façon la plus transparente pour ce faire serait des déclencheurs, mais vous pouvez également modifier votre application afin de maintenir la colonne bbox courante ou exécuter la requête UPDATE ci-dessus après chaque modification.

6.2 CLUSTER d'index géométriques

Pour les tables qui sont pour la plupart en lecture seule, et où un seul index est utilisé pour la majorité des requêtes, PostgreSQL offre la commande CLUSTER. Cette commande réorganise physiquement toutes les lignes de données dans le même ordre que les critères de l'index, ce qui donne deux avantages de performance : d'abord, pour des analyses d'intervalle de l'index, le nombre de recherche sur la table de données est considérablement réduit. Deuxièmement, si votre jeu de travail se concentre à quelques petits intervalles sur les index, vous avez une mise en cache plus efficace parce que les lignes de données sont dispersées sur moins de pages de données. (N'hésitez pas à lire la documentation de la commande CLUSTER du manuel PostgreSQL à ce stade)

Cependant, PostgreSQL ne permet actuellement pas le clustering sur les index GIST de PostGIS car les indices GIST ignorent les valeurs NULL, vous obtenez un message d'erreur comme :

```
lwgeom=# CLUSTER my_geom_index ON my_table;
ERROR: cannot cluster when index access method does not handle null values
HINT: You may be able to work around this by marking column "geom" NOT NULL.
```

Comme le message d'ASTUCES vous le dit, on peut contourner cette lacune en ajoutant une contrainte "not null" à la table :

```
lwgeom=# ALTER TABLE my_table ALTER COLUMN geom SET not null;
ALTER TABLE
```

Bien sûr, cela ne fonctionnera pas si vous avez besoin, dans les faits, de valeurs NULL dans la colonne de géométrie. En outre, vous devez utiliser la méthode ci-dessus pour ajouter la contrainte, en utilisant une contrainte CHECK comme "ALTER TABLE blubb ADD CHECK (geometry is not null)" ne fonctionnera pas.

6.3 Eviter les conversions de dimension

Il arrive parfois que vous ayez des données 3D ou 4D dans votre table, mais que vous y accédiez toujours à l'aide des fonctions ST_AsText() ou ST_AsBinary() conformes à OpenGIS qui ne produisent que des géométries 2D. Pour ce faire, elles appellent en interne la fonction ST_Force2D(), qui introduit une surcharge importante pour les géométries de grande taille. Pour éviter cette surcharge, il peut être possible de pré-déposer ces dimensions supplémentaires une fois pour toutes :

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

Notez que si vous avez ajouté votre colonne de géométrie à l'aide AddGeometryColumn (), il y aura une contrainte sur la dimension de la géométrie. Pour contourner vous devrez supprimer la contrainte. N'oubliez pas de mettre à jour l'entrée dans la table geometry_columns et recréer la contrainte par la suite.

En cas de grandes tables, il peut être judicieux de diviser cette mise à jour en petites portions en restreignant l'UPDATE à une partie de la table via une clause WHERE et votre clé primaire ou d'un autre critère, et exécutant un simple «VACUUM» ;

entre votre mises à jour. Cela réduit considérablement le besoin d'espace disque temporaire. En outre, si vous avez des données géométriques de dimension mixte, restreindre la mise à jour en "WHERE dimension(the_geom)>2" saute la ré-écriture des géométries qui sont déjà en 2D.

Chapter 7

Référence PostGIS

Les fonctions suivantes sont celles dont un utilisateur normal de PostGIS peut avoir besoin. Il existe d'autres fonctions nécessaires au fonctionnement de PostGIS. Elles ne sont normalement pas destinées à un utilisateur standard.

Note
PostGIS a entamé une phase de transition concernant le nom des fonctions pour les faire correspondre à la norme SQL-MM. La plupart des fonctions ont été renommées en utilisant le préfixe des types spatiaux (ST, pour Spatial Type). Les anciennes fonctions, toujours disponibles, ne sont pas listées ci-après si leur équivalent est disponible. Les fonctions non préfixées par ST, qui ne sont pas listées dans cette documentation, sont dépréciées et seront supprimées des prochaines version de PostGIS. IL NE FAUT PLUS LES UTILISER.

7.1 Types de données PostGIS Geometry/Geography/Box

7.1.1 box2d

box2d — Le type représentant une boîte de délimitation bidimensionnelle.

Description

`box2d` est un type de données spatiales utilisé pour représenter la boîte de délimitation bidimensionnelle entourant une géométrie ou une collection de géométries. Par exemple, la fonction d'agrégation ST_Extent renvoie un objet `box2d`.

La représentation contient les valeurs `xmin, ymin, xmax, ymax`. Ce sont les valeurs minimales et maximales des étendues X et Y.

Les objets `box2d` ont une représentation textuelle qui ressemble à `BOX(1 2,5 6)`.

Transtypages

Ce tableau énumère les casts automatiques et explicites autorisés pour ce type de données :

Transtypage vers	Comportement
box3d	automatique
geometry	automatique

Voir aussi

Section 13.7

7.1.2 box3d

box3d — Le type représentant une boîte de délimitation tridimensionnelle.

Description

`box3d` est un type de données spatiales PostGIS utilisé pour représenter la boîte de délimitation tridimensionnelle entourant une géométrie ou une collection de géométries. Par exemple, la fonction d'agrégation ST_3DExtent renvoie un objet `box3d`.

La représentation contient les valeurs `xmin, ymin, zmin, xmax, ymax, zmax`. Il s'agit des valeurs minimales et maximales des étendues X, Y et Z.

Les objets `box3d` ont une représentation textuelle qui ressemble à `BOX3D(1 2 3,5 6 5)`.

Transtypages

Ce tableau énumère les casts automatiques et explicites autorisés pour ce type de données :

Transtypage vers	Comportement
box	automatique
box2d	automatique
geometry	automatique

Voir aussi

Section 13.7

7.1.3 geometry

geometry — Le type représentant les caractéristiques spatiales avec des systèmes de coordonnées planaires.

Description

`géométrie` est un type de données spatiales fondamental de PostGIS utilisé pour représenter une caractéristique dans des systèmes de coordonnées planes (euclidiennes).

Toutes les opérations spatiales sur la géométrie utilisent les unités du système de référence spatiale dans lequel se trouve la géométrie.

Transtypages

Ce tableau énumère les casts automatiques et explicites autorisés pour ce type de données :

Transtypage vers	Comportement
box	automatique
box2d	automatique
box3d	automatique
bytea	automatique
geography	automatique
text	automatique

Voir aussi

Section 4.1, Section 13.3

7.1.4 geometry_dump

geometry_dump — Un type composite utilisé pour décrire les parties d'une géométrie complexe.

Description

`geometry_dump` est un type de données composite contenant les champs :

- `geom` - une géométrie représentant un composant de la géométrie explosée. Le type de géométrie dépend de la fonction d'origine.
- `path[]` - un tableau d'entiers qui définit le chemin de navigation dans la géométrie explosée vers le composant `geom`. Le tableau de chemins est basé sur 1 (c'est-à-dire que `path[1]` est le premier élément)

Il est utilisé par la famille de fonctions `ST_Dump*` comme type de sortie pour exploser une géométrie complexe en ses parties constitutives.

Voir aussi

Section 13.6

7.1.5 geography

geography — Le type représentant les entités spatiales avec des systèmes de coordonnées géodésiques (ellipsoïdales).

Description

`geography` est un type de données spatiales utilisé pour représenter une entité dans les systèmes de coordonnées géodésiques. Les systèmes de coordonnées géodésiques modélisent la terre à l'aide d'un ellipsoïde.

Les opérations spatiales sur le type geography fournissent des résultats plus précis en prenant en compte le modèle ellipsoïdal.

Transtypages

Ce tableau énumère les casts automatiques et explicites autorisés pour ce type de données :

Transtypage vers	Comportement
geometry	explicite

Voir aussi

Section 4.3, Section 13.4

7.2 Fonctions de gestion des tables

7.2.1 AddGeometryColumn

AddGeometryColumn — Ajoute une colonne de géométrie à une table existante.

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);

Description

Ajoute une colonne géométrique à une table attributaire existante. `schema_name` est le nom du schéma de la table. `srid` est un entier positif présent dans la table SPATIAL_REF_SYS. `type` est le type de géométrie en texte, par exemple 'POLYGON' ou 'MULTILINESTRING'. Une erreur est renvoyée si le schéma n'existe pas (ou n'est pas visible dans le search_path courant) ou si le SRID, type de géométrie ou dimension est invalide.

Note
Changement : 2.0.0 Cette fonction ne met plus à jour geometry_columns maintenant que geometry_columns est une vue basée sur le catalogue système. Par défaut, elle ne créée plus de contraintes mais utilise le modificateur de type de PostgreSQL. Ainsi, par exemple, créer une colonne de type POINT WGS84 est désormais équivalent à : `ALTER TABLE some_table ADD COLUMN geom geometry(Point,4326) ;`
Changement : 2.0.0 Si l'ancien mécanisme basé sur les contraintes est nécessaire, utiliser le paramètre `use_typmod` avec la valeur false.

Note
Changement : 2.0.0 Les vues ne peuvent plus être enregistrées dans geometry_columns. Cependant, les vues construites à partir de tables contenant des géométries définies avec le modificateur de type et n'utilisant pas de fonctions d'encapsulation seront enregistrées dans la vue geometry_columns car elles héritent du mécanisme des tables dont elles sont issues. Les vues utilisant des fonctions renvoyant d'autres géométries doivent être transtypées vers des géométries avec modificateur de type pour pouvoir être correctement référencées dans la vue geometry_columns. Cf. Section 4.6.3.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Amélioration : 2.0.0 introduction du paramètre use_typmod. Le comportement par défaut est de créer une colonne géométrique avec modificateur de type au lieu de contraintes sur la colonne.

Exemples

```
-- Create schema to hold data
CREATE SCHEMA my_schema;
-- Create a new simple PostgreSQL table
CREATE TABLE my_schema.my_spatial_table (id serial);

-- Describing the table shows a simple table with a single "id" column.
postgis=# \d my_schema.my_spatial_table
                                                    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
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geom',4326,'POINT',2);

-- Add a point using the old constraint based behavior
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geom_c',4326,'POINT',2, false);

--Add a curvepolygon using old constraint behavior
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geomcp_c',4326,'CURVEPOLYGON',2,  ↩
    false);

-- Describe the table again reveals the addition of a new geometry columns.
\d my_schema.my_spatial_table
                           addgeometrycolumn
-----------------------------------------------------------------
 my_schema.my_spatial_table.geomcp_c SRID:4326 TYPE:CURVEPOLYGON DIMS:2
(1 row)

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

 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)

-- geometry_columns view also registers the new columns --
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';

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

Voir aussi

DropGeometryColumn, DropGeometryTable, Section 4.6.2, Section 4.6.3

7.2.2 DropGeometryColumn

DropGeometryColumn — Supprime une colonne géométrique d'une table spatiale.

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);

Description

Supprime une colonne géométrique d'une table spatiale. Note : schema_name doit correspondre au champ f_table_schema de la table geometry_columns.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

> **Note**
> Changement : 2.0.0 Fonction assurant la rétro compatibilité. Maintenant que geometry_columns est une vue basée sur les catalogues du système, la colonne géométrique peut être supprimée d'une table comme tout autre colonne en utilisant `ALTER TABLE`

Exemples

```
SELECT DropGeometryColumn ('my_schema','my_spatial_table','geom');
                        ----RESULT output ---
                                          dropgeometrycolumn
----------------------------------------------------
 my_schema.my_spatial_table.geom effectively removed.

-- In PostGIS 2.0+ the above is also equivalent to the standard
-- the standard alter table.  Both will deregister from geometry_columns
ALTER TABLE my_schema.my_spatial_table DROP column geom;
```

Voir aussi

AddGeometryColumn, DropGeometryTable, Section 4.6.2

7.2.3 DropGeometryTable

DropGeometryTable — Supprime une table et toutes ces références dans 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);

Description

Supprime une table et toutes ces références dans geometry_columns. Note : utilise la fonction current_schema() sur les installations PostgreSQL le supportant, si le schéma n'est pas fourni.

Note
Changement : 2.0.0 Function assurant la rétro compatibilité. Maintenant que geometry_columns est une vue basée sur les catalogues du système, une table spatiale peut etre supprimée comme tout autre table en utilisant `ALTER TABLE`

Exemples

```
SELECT DropGeometryTable ('my_schema','my_spatial_table');
----RESULT output ---
my_schema.my_spatial_table dropped.

-- The above is now equivalent to --
DROP TABLE my_schema.my_spatial_table;
```

Voir aussi

AddGeometryColumn, DropGeometryColumn, Section 4.6.2

7.2.4 Find_SRID

Find_SRID — Renvoie le SRID défini pour une colonne géométrique.

Synopsis

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

Description

Renvoie le SRID entier de la colonne géométrique spécifiée en effectuant une recherche dans la table GEOMETRY_COLUMNS. Si la colonne géométrique n'a pas été correctement ajoutée (par exemple avec la fonction AddGeometryColumn), cette fonction ne fonctionnera pas.

Exemples

```
SELECT Find_SRID('public', 'tiger_us_state_2007', 'geom_4269');
find_srid
----------
4269
```

Voir aussi

ST_SRID

7.2.5 Populate_Geometry_Columns

Populate_Geometry_Columns — Assure que les colonnes géométriques sont définies avec des modificateurs de type ou qu'elles sont soumises à des contraintes spatiales appropriées.

Synopsis

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

Description

S'assure que les colonnes géométriques ont des modificateurs de type ou des contraintes spatiales appropriés pour garantir qu'elles sont enregistrées correctement dans la vue `geometry_columns`. Par défaut, convertira toutes les colonnes géométriques sans modificateur de type en colonnes avec modificateur de type.

Pour conserver la rétro compatibilité et pour des besoins particuliers comme par exemple des tables héritées ayant des types géométriques différents, l'ancien mécanisme est toujours supporté. Si ce mécanisme est nécessaire, le nouveau paramètre optionnel doit être mis à false : `use_typmod=false`. Avec cette valeur, la colonne géométrique sera créée sans modificateur de type mais 3 contraintes seront définies. Cela signifie concrètement que chaque colonne géométrique de la table aura au moins 3 contraintes :

- `enforce_dims_geom` - garantit que chaque géométrie a la même dimension (voir ST_NDims)
- `enforce_geotype_geom` - garantit que chaque géométrie est du même type (voir GeometryType)
- `enforce_srid_the_geom` - s'assure que toutes les géométries sont dans la même projection (see ST_SRID)

Si un identifiant de table `oid` est fourni, cette fonction tente de déterminer le SRID, la dimension et le type géométrique de toutes les colonnes géométriques de la table, ajoutant des contraintes si nécessaire. En cas de succès, une ligne est insérée dans la table geometry_columns, sinon, une erreur est affichée indiquant le problème.

Si le `oid` d'une vue est fourni, comme avec un oid de table, cette fonction essaie de déterminer le srid, la dimension et le type de toutes les géométries de la vue, en insérant les entrées appropriées dans la table `geometry_columns`, mais rien n'est fait pour appliquer les contraintes.

La version sans paramètre est un raccourci pour la version avec paramètres. Elle vide puis remplit la table geometry_columns pour chaque table ou vue spatiale de la base, ajoutant les contraintes aux tables si besoin. Retourne un résumé montrant le nombre de colonnes géométriques identifiées dans la base et le nombre inséré dans la table `geometry_columns`. La version avec paramètres renvoie juste le nombre de lignes insérées dans la table `geometry_columns` .

Disponibilité : 1.4.0

Changement : 2.0.0 Par défaut, utilise les modificateurs de type au lieu de contraintes de vérification pour contraindre les types géométriques. Le comportement basé sur les contraintes peut être activé en mettant le nouveau paramètre `use_typmod` à false.

Amélioration : 2.0.0 L'argument optionnel `use_typmod` a été introduit pour controler si les colonnes sont créés avec des modificateurs de type ou des contraintes de vérification.

Exemples

```
CREATE TABLE public.myspatial_table(gid serial, geom geometry);
INSERT INTO myspatial_table(geom) VALUES(ST_GeomFromText('LINESTRING(1 2, 3 4)',4326) );
-- This will now use typ modifiers.  For this to work, there must exist data
SELECT Populate_Geometry_Columns('public.myspatial_table'::regclass);

populate_geometry_columns
--------------------------
                         1


\d myspatial_table

                                  Table "public.myspatial_table"
 Column |           Type            |                           Modifiers
--------+---------------------------+-----------------------------------------------------------------

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

```
-- This will change the geometry columns to use constraints if they are not typmod or have  ↩
   constraints already.
--For this to work, there must exist data
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);
populate_geometry_columns
--------------------------
                         1
\d myspatial_table_cs

                          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 — Met à jour le SRID de tous les éléments d'une colonne géométrique et les métadonnées de la table.

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);

Description

Met à jour le SRID de tous les objets d'une colonne géométrique et met à jour les métadonnées de geometry_columns et la contrainte sur le SRID. Note : utilise la fonction current_schema() sur les installations PostgreSQL le supportant, si le

schéma n'est pas fourni.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

Insérer des géométries dans une table roads avec un jeu de SRID utilisant déjà le format EWKT :

```
COPY roads (geom) FROM STDIN;
SRID=4326;LINESTRING(0 0, 10 10)
SRID=4326;LINESTRING(10 10, 15 0)
\.
```

Cela va changer le srid de la table roads à 4326 quelle que soit sa valeur avant :

```
SELECT UpdateGeometrySRID('roads','geom',4326);
```

L'exemple précédent est équivalent à cette requête DDL :

```
ALTER TABLE roads
  ALTER COLUMN geom TYPE geometry(MULTILINESTRING, 4326)
    USING ST_SetSRID(geom,4326);
```

Si vous vous êtes trompé de projection (ou si vous l'avez introduite en tant qu'inconnue) dans le chargement et que vous voulez transformer en mercator web en une seule fois, vous pouvez le faire avec DDL mais il n'y a pas de fonction de gestion PostGIS équivalente pour le faire en une seule fois.

```
ALTER TABLE roads
 ALTER COLUMN geom TYPE geometry(MULTILINESTRING, 3857) USING ST_Transform(ST_SetSRID(geom ↩
     ,4326),3857) ;
```

Voir aussi

UpdateRasterSRID, ST_SetSRID, ST_Transform, ST_GeomFromEWKT

7.3 Constructeurs de géométries

7.3.1 ST_Collect

ST_Collect — Crée une géométrie GeometryCollection ou Multi à partir d'un ensemble de géométries.

Synopsis

geometry **ST_Collect**(geometry g1, geometry g2);
geometry **ST_Collect**(geometry[] g1_array);
geometry **ST_Collect**(geometry set g1field);

Description

Rassemble les géométries dans une collection de géométries. Le résultat est soit un Multi*, soit une GeometryCollection, selon que les géométries d'entrée ont des types identiques ou différents (homogènes ou hétérogènes). Les géométries d'entrée restent inchangées dans la collection.

Variante 1: accepte géométries en entrée

Variante 2: accepte un tableau de géométries

Variante 3: fonction agrégée acceptant un ensemble de géométries.

Note
Si l'une des géométries en entrée est une collection (Multi* ou GeometryCollection), ST_Collect renvoie une GeometryCollection (puisque c'est le seul type qui peut contenir des collections imbriquées). Pour éviter cela, utilisez ST_Dump dans une sous-requête pour développer les collections d'entrée en leurs éléments atomiques (voir l'exemple ci-dessous).

Note
ST_Collect et ST_Union semblent similaires, mais fonctionnent en fait assez différemment. ST_Collect agrège les géométries dans une collection sans les modifier en aucune façon. ST_Union fusionne géométriquement les géométries lorsqu'elles se chevauchent, et divise les lignes aux intersections. Elle peut retourner des géométries uniques lorsqu'elle dissout les frontières.

Disponibilité : 1.4.0 - ST_Collect(geomarray) a été introduit. ST_Collect a été amélioré pour gérer plus de géométries plus rapidement.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples - Variante à deux entrées

Collectez les points 2D.

```
SELECT ST_AsText( ST_Collect( ST_GeomFromText('POINT(1 2)'),
        ST_GeomFromText('POINT(-2 3)') ));

st_astext
----------
MULTIPOINT((1 2),(-2 3))
```

Collectez les points 3D.

```
SELECT ST_AsEWKT( ST_Collect( ST_GeomFromEWKT('POINT(1 2 3)'),
                ST_GeomFromEWKT('POINT(1 2 4)') ) );

                st_asewkt
--------------------------
 MULTIPOINT(1 2 3,1 2 4)
```

Collectez les courbes.

```
SELECT ST_AsText( ST_Collect( 'CIRCULARSTRING(220268 150415,220227 150505,220227 150406)',
                'CIRCULARSTRING(220227 150406,2220227 150407,220227 150406)'));

                st_astext
```

```
-------------------------------------------------------------------------------------
MULTICURVE(CIRCULARSTRING(220268 150415,220227 150505,220227 150406),
 CIRCULARSTRING(220227 150406,2220227 150407,220227 150406))
```

Exemples - Variante avec un tableau

Utilisation d'un constructeur de tableau pour une sous-requête.

```
SELECT ST_Collect( ARRAY( SELECT geom FROM sometable ) );
```

Utilisation d'un constructeur de tableau pour les valeurs.

```
SELECT ST_AsText(  ST_Collect(
                ARRAY[ ST_GeomFromText('LINESTRING(1 2, 3 4)'),
                        ST_GeomFromText('LINESTRING(3 4, 4 5)') ] )) As wktcollect;

--wkt collect --
MULTILINESTRING((1 2,3 4),(3 4,4 5))
```

Exemples - Variante avec un agrégat

Création de collections multiples en regroupant les géométries dans une table.

```
SELECT stusps, ST_Collect(f.geom) as geom
         FROM (SELECT stusps, (ST_Dump(geom)).geom As geom
                              FROM
                              somestatetable ) As f
        GROUP BY stusps
```

Voir aussi

ST_Dump, ST_Union

7.3.2 ST_LineFromMultiPoint

ST_LineFromMultiPoint — Crée une LineString à partir d'une géométrie MultiPoint.

Synopsis

geometry **ST_LineFromMultiPoint**(geometry aMultiPoint);

Description

Crée une LineString à partir d'une géométrie MultiPoint.

Utilisez ST_MakeLine pour créer des lignes à partir des entrées Point ou LineString.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Créer une LineString 3D à partir d'un MultiPoint 3D

```
SELECT ST_AsEWKT(  ST_LineFromMultiPoint('MULTIPOINT(1 2 3, 4 5 6, 7 8 9)')  ));

--result--
LINESTRING(1 2 3,4 5 6,7 8 9)
```

Voir aussi

ST_AsEWKT, ST_MakeLine

7.3.3 ST_MakeEnvelope

ST_MakeEnvelope — Crée un polygone rectangulaire à partir des coordonnées minimales et maximales.

Synopsis

geometry **ST_MakeEnvelope**(float xmin, float ymin, float xmax, float ymax, integer srid=unknown);

Description

Crée un polygone rectangulaire à partir des valeurs minimales et maximales de X et Y. Les valeurs entrées doivent être dans le système de référence spatiale spécifié par le SRID. Si aucun SRID n'est spécifié, le système de référence spatiale inconnu (SRID 0) est utilisé.

Disponibilité : 1.5

Amélioré : 2.0 : La possibilité de spécifier une enveloppe sans spécifier un SRID a été introduite.

Exemple : Construction d'un polygone de la l'enveloppe englobante

```
SELECT ST_AsText( ST_MakeEnvelope(10, 10, 11, 11, 4326) );

st_asewkt
-----------
POLYGON((10 10, 10 11, 11 11, 11 10, 10 10))
```

Voir aussi

ST_MakePoint, ST_MakeLine, ST_MakePolygon, ST_TileEnvelope

7.3.4 ST_MakeLine

ST_MakeLine — Crée une LineString à partir de géométries Point, MultiPoint ou LineString.

Synopsis

geometry **ST_MakeLine**(geometry geom1, geometry geom2);
geometry **ST_MakeLine**(geometry[] geoms_array);
geometry **ST_MakeLine**(geometry set geoms);

Description

Crée une LineString contenant les points des géométries Point, MultiPoint ou LineString. Les autres types de géométrie provoquent une erreur.

Variante 1: accepte géométries en entrée

Variante 2: accepte un tableau de géométries

Variante 3: fonction agrégée acceptant un ensemble de géométries. Pour garantir l'ordre des géométries d'entrée, utilisez `ORDER BY` dans l'appel de fonction, ou une sous-requête avec une clause `ORDER BY`.

Les nœuds répétés au début des LineStrings d'entrée sont réduits à un seul point. Les points répétés dans les entrées Point et MultiPoint ne sont pas réduits. ST_RemoveRepeatedPoints peut être utilisé pour réduire les points répétés de la LineString de sortie.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Disponibilité : 2.3.0 - La prise en charge des éléments d'entrée MultiPoint a été introduite

Disponibilité : 2.0.0 - La prise en charge des éléments d'entrée LineString a été introduite

Disponibilité : 1.4.0 - création de ST_MakeLine(geomarray). L'agrégat spatial ST_MakeLine amélioré pour supporter plus de points plus rapidement.

Exemples : Variante à deux entrées

Créez une ligne composée de deux points.

```
SELECT ST_AsText( ST_MakeLine(ST_Point(1,2), ST_Point(3,4)) );

          st_astext
---------------------
 LINESTRING(1 2,3 4)
```

Créer une ligne 3D à partir de deux points 3D.

```
SELECT ST_AsEWKT( ST_MakeLine(ST_MakePoint(1,2,3), ST_MakePoint(3,4,5) ));

                st_asewkt
-------------------------
 LINESTRING(1 2 3,3 4 5)
```

Crée une ligne à partir de deux LineStrings disjointes.

```
select ST_AsText( ST_MakeLine( 'LINESTRING(0 0, 1 1)', 'LINESTRING(2 2, 3 3)' ) );

          st_astext
-----------------------------
 LINESTRING(0 0,1 1,2 2,3 3)
```

Exemples : Variante avec un tableau

Créer une ligne à partir d'un tableau formé par une sous-requête avec ordonnancement.

```
SELECT ST_MakeLine( ARRAY( SELECT ST_Centroid(geom) FROM visit_locations ORDER BY  ↩
    visit_time) );
```

Créer une ligne 3D à partir d'un tableau de points 3D

```
SELECT ST_AsEWKT( ST_MakeLine(
          ARRAY[ ST_MakePoint(1,2,3), ST_MakePoint(3,4,5), ST_MakePoint(6,6,6) ]  ));

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

Exemples : Variante avec un agrégat

Cet exemple interroge des séquences temporelles de points GPS à partir d'un ensemble de pistes et crée un enregistrement pour chaque piste. Les géométries résultantes sont des LineStrings composées des points GPS dans l'ordre de leur déplacement.

L'utilisation de l'agrégat `ORDER BY` fournit une LineString correctement ordonnée.

```
SELECT gps.track_id, ST_MakeLine(gps.geom ORDER BY gps_time) As geom
        FROM gps_points As gps
        GROUP BY track_id;
```

Avant PostgreSQL 9, l'ordre dans une sous-requête peut être utilisé. Cependant, il arrive que le plan de requête ne respecte pas l'ordre de la sous-requête.

```
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 track_id;
```

Voir aussi

ST_RemoveRepeatedPoints, ST_AsEWKT, ST_AsText, ST_GeomFromText, ST_MakePoint, ST_Point

7.3.5 ST_MakePoint

ST_MakePoint — Crée un point 2D, 3DZ ou 4D.

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);

Description

Crée une géométrie de point 2D XY, 3D XYZ ou 4D XYZM. Utilisez ST_MakePointM pour créer des points avec des coordonnées XYM.

Utilisez ST_SetSRID pour spécifier un SRID pour le point créé.

Bien que non conforme à l'OGC, `ST_MakePoint` est plus rapide et plus précis que ST_GeomFromText et ST_PointFromText. Elle est également plus facile à utiliser pour les valeurs de coordonnées numériques.

Note
Pour les coordonnées géodésiques, `X` est la longitude et `Y` la latitude

Note
Les fonctions ST_Point, ST_PointZ, ST_PointM et ST_PointZM peuvent être utilisées pour créer des points avec un SRID donné.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
-- Create a point with unknown SRID
SELECT ST_MakePoint(-71.1043443253471, 42.3150676015829);

-- Create a point in the WGS 84 geodetic CRS
SELECT ST_SetSRID(ST_MakePoint(-71.1043443253471, 42.3150676015829),4326);

-- Create a 3D point (e.g. has altitude)
SELECT ST_MakePoint(1, 2,1.5);

-- Get z of point
SELECT ST_Z(ST_MakePoint(1, 2,1.5));
result
-------
1.5
```

Voir aussi

ST_GeomFromText, ST_PointFromText, ST_SetSRID, ST_MakePointM, ST_Point, ST_PointZ, ST_PointM, ST_PointZM

7.3.6 ST_MakePointM

ST_MakePointM — Crée un point à partir des valeurs X, Y et M.

Synopsis

geometry **ST_MakePointM**(float x, float y, float m);

Description

Crée un point avec des coordonnées X, Y et M (mesure). Utilisez ST_MakePoint pour créer des points avec des coordonnées XY, XYZ ou XYZM.

Utilisez ST_SetSRID pour spécifier un SRID pour le point créé.

Note
Pour les coordonnées géodésiques, `X` est la longitude et `Y` la latitude

Note
Les fonctions ST_PointM et ST_PointZM peuvent être utilisées pour créer des points avec une valeur M et un SRID donné.

Exemples

Note
ST_AsEWKT est utilisé pour la sortie texte car ST_AsText ne prend pas en charge les valeurs M.

Créer un point avec un SRID inconnu.

```
SELECT ST_AsEWKT(  ST_MakePointM(-71.1043443253471, 42.3150676015829, 10)  );

                                    st_asewkt
-----------------------------------------------
 POINTM(-71.1043443253471 42.3150676015829 10)
```

Créer un point avec une mesure dans le système de coordonnées géodésiques WGS 84.

```
SELECT ST_AsEWKT( ST_SetSRID(  ST_MakePointM(-71.104, 42.315, 10),  4326));

                                                st_asewkt
----------------------------------------------------------
SRID=4326;POINTM(-71.104 42.315 10)
```

Obtenir la mesure du point créé.

```
SELECT ST_M(  ST_MakePointM(-71.104, 42.315, 10)  );

result
-------
10
```

Voir aussi

ST_MakePoint, ST_SetSRID, ST_PointM, ST_PointZM

7.3.7 ST_MakePolygon

ST_MakePolygon — Crée un polygone à partir d'une collection et d'une liste facultative de trous.

Synopsis

geometry **ST_MakePolygon**(geometry linestring);

geometry **ST_MakePolygon**(geometry outerlinestring, geometry[] interiorlinestrings);

Description

Crée un polygone formé par la collection donnée et un tableau optionnel de trous. Les géométries d'entrée doivent être des LineStrings (anneaux) fermés.

Variante 1: Accepte une collection de LineString.

Variante 2: Accepte une collection de LineString et un tableau de LineStrings internes (trous). Un tableau de géométrie peut être construit en utilisant les constructions PostgreSQL array_agg(), ARRAY[] ou ARRAY().

Note
Cette fonction n'accepte pas les MultiLineStrings. Utilisez ST_LineMerge pour générer une LineString, ou ST_Dump pour extraire les LineStrings.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples : variante avec une entrée unique

Créez un polygone à partir d'une LineString 2D.

```
SELECT ST_MakePolygon( ST_GeomFromText('LINESTRING(75 29,77 29,77 29, 75 29)'));
```

Créez un polygone à partir d'une LineString ouverte, en utilisant ST_StartPoint et ST_AddPoint pour la fermer.

```
SELECT ST_MakePolygon( ST_AddPoint(foo.open_line, ST_StartPoint(foo.open_line)) )
FROM (
  SELECT ST_GeomFromText('LINESTRING(75 29,77 29,77 29, 75 29)') As open_line) As foo;
```

Créer un polygone à partir d'une LineString 3D

```
SELECT ST_AsEWKT( ST_MakePolygon( 'LINESTRING(75.15 29.53 1,77 29 1,77.6 29.5 1, 75.15  ↩
    29.53 1)'));

st_asewkt
-----------
POLYGON((75.15 29.53 1,77 29 1,77.6 29.5 1,75.15 29.53 1))
```

Créer un polygone à partir d'une LineString avec des mesures

```
SELECT ST_AsEWKT( ST_MakePolygon( 'LINESTRINGM(75.15 29.53 1,77 29 1,77.6 29.5 2, 75.15  ↩
    29.53 2)' ));

st_asewkt
----------
POLYGONM((75.15 29.53 1,77 29 1,77.6 29.5 2,75.15 29.53 2))
```

Exemples : Enveloppe extérieure avec variante de trous intérieurs

Créer un polygone en forme de donut avec un trou supplémentaire

```
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;
```

Créez un ensemble de frontières de province avec des trous représentant des lacs. L'entrée est un tableau de polygones/multi-polygones de province et un tableau de lignes d'eau. Les lignes formant des lacs sont déterminées en utilisant ST_IsClosed. Le réseau de la province est extrait en utilisant ST_Boundary. Comme requis par `ST_MakePolygon`, la frontière est forcée à être une seule LineString en utilisant ST_LineMerge. (Cependant, notez que si une province a plus d'une région ou a des îles, cela produira un polygone invalide). L'utilisation d'un LEFT JOIN garantit que toutes les provinces sont incluses, même si elles n'ont pas de lacs.

Note
La construction CASE est utilisée car le passage d'un tableau nul dans ST_MakePolygon entraîne une valeur de retour NULL.

```
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
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;
```

Une autre technique consiste à utiliser une sous-requête corrélée et le constructeur ARRAY() qui convertit un ensemble de lignes en un tableau.

```
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;
```

Voir aussi

ST_BuildArea ST_Polygon

7.3.8 ST_Point

ST_Point — Crée un point avec des valeurs X, Y et SRID.

Synopsis

geometry **ST_Point**(float x, float y);

geometry **ST_Point**(float x, float y, integer srid=unknown);

Description

Renvoie un point avec les valeurs de coordonnées X et Y données. C'est l'équivalent SQL-MM de ST_MakePoint qui ne prend que X et Y.

Note
Pour les coordonnées géodésiques, `X` est la longitude et `Y` la latitude

Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 6.1.2

Exemple : Géométrie

```
SELECT ST_Point( -71.104, 42.315);
```

Création d'un point avec le SRID spécifié :

```
SELECT ST_Point( -71.104, 42.315, 4326);
```

Autre façon de spécifier le SRID :

```
SELECT ST_SetSRID( ST_Point( -71.104, 42.315), 4326);
```

Exemples : Géographie

Créez des points geography en utilisant la syntaxe `::` :

```
SELECT ST_Point( -71.104, 42.315, 4326)::geography;
```

Code Pre-PostGIS 3.2, utilisant `CAST` :

```
SELECT CAST( ST_SetSRID(ST_Point( -71.104, 42.315), 4326) AS geography);
```

Si les coordonnées du point ne sont pas dans un système de coordonnées géodésiques (tel que WGS84), elles doivent être reprojetées avant d'être projetées dans une géographie. Dans cet exemple, un point en pieds du plan de l'État de Pennsylvanie (SRID 2273) est projeté en WGS84 (SRID 4326).

```
SELECT ST_Transform( ST_Point( 3637510, 3014852, 2273), 4326)::geography;
```

Voir aussi

ST_MakePoint, ST_PointZ, ST_PointM, ST_PointZM, ST_SetSRID, ST_Transform

7.3.9 ST_PointZ

ST_PointZ — Crée un point avec des valeurs X, Y, Z et SRID.

Synopsis

geometry **ST_PointZ**(float x, float y, float z, integer srid=unknown);

Description

Renvoie un point avec les valeurs de coordonnées X, Y et Z données, et éventuellement un numéro SRID.

Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie.

Exemples

```
SELECT ST_PointZ(-71.104, 42.315, 3.4, 4326)
```

```
SELECT ST_PointZ(-71.104, 42.315, 3.4, srid => 4326)
```

```
SELECT ST_PointZ(-71.104, 42.315, 3.4)
```

Voir aussi

ST_MakePoint, ST_Point, ST_PointM, ST_PointZM

7.3.10 ST_PointM

ST_PointM — Crée un point avec des valeurs X, Y, M et SRID.

Synopsis

geometry **ST_PointM**(float x, float y, float m, integer srid=unknown);

Description

Renvoie un point avec les valeurs de coordonnées X, Y et M données, et éventuellement un numéro SRID.

Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie.

Exemples

```
SELECT ST_PointM(-71.104, 42.315, 3.4, 4326)
```

```
SELECT ST_PointM(-71.104, 42.315, 3.4, srid => 4326)
```

```
SELECT ST_PointM(-71.104, 42.315, 3.4)
```

Voir aussi

ST_MakePoint, ST_Point, ST_PointZ, ST_PointZM

7.3.11 ST_PointZM

ST_PointZM — Crée un point avec des valeurs X, Y, Z, M et SRID.

Synopsis

geometry **ST_PointZM**(float x, float y, float z, float m, integer srid=unknown);

Description

Renvoie un point avec les valeurs de coordonnées X, Y, Z et M données, et éventuellement un numéro SRID.

Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie.

Exemples

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5, 4326)
```

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5, srid => 4326)
```

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5)
```

Voir aussi

ST_MakePoint, ST_Point, ST_PointM, ST_PointZ, ST_SetSRID

7.3.12 ST_Polygon

ST_Polygon — Crée un polygone à partir d'une LineString avec un SRID spécifié.

Synopsis

geometry **ST_Polygon**(geometry lineString, integer srid);

Description

Renvoie un polygone construit à partir de la LineString donnée et définit le système de référence spatiale à partir du `srid`.

ST_Polygon est similaire à la variante 1 de ST_MakePolygon avec l'ajout de la définition du SRID.

Pour créer des polygones avec des trous, utilisez ST_MakePolygon la variante 2 et ensuite ST_SetSRID.

Note
Cette fonction n'accepte pas les MultiLineStrings. Utilisez ST_LineMerge pour générer une LineString, ou ST_Dump pour extraire les LineStrings.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 8.3.2

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Créer un polygone 2D.

```
SELECT ST_AsText( ST_Polygon('LINESTRING(75 29, 77 29, 77 29, 75 29)'::geometry, 4326) );

-- result --
POLYGON((75 29, 77 29, 77 29, 75 29))
```

Créer un polygone 3D.

```
SELECT ST_AsEWKT( ST_Polygon( ST_GeomFromEWKT('LINESTRING(75 29 1, 77 29 2, 77 29 3, 75 29 ↩
    1)'), 4326) );

-- result --
SRID=4326;POLYGON((75 29 1, 77 29 2, 77 29 3, 75 29 1))
```

Voir aussi

ST_AsEWKT, ST_AsText, ST_GeomFromEWKT, ST_GeomFromText, ST_LineMerge, ST_MakePolygon

7.3.13 ST_TileEnvelope

ST_TileEnvelope — Crée un polygone rectangulaire dans Web Mercator (SRID:3857) en utilisant le système de tuiles XYZ.

Synopsis

geometry **ST_TileEnvelope**(integer tileZoom, integer tileX, integer tileY, geometry bounds=SRID=3857;LINESTRING(-20037508.342 -20037508.342789,20037508.342789 20037508.342789), float margin=0.0);

Description

Crée un polygone rectangulaire donnant l'étendue d'une tuile dans le système de tuiles XYZ. La tuile est spécifiée par le niveau de zoom Z et l'indice XY de la tuile dans la grille à ce niveau. Peut être utilisé pour définir les limites de la tuile requises par ST_AsMVTGeom pour convertir la géométrie dans l'espace de coordonnées de la tuile MVT.

Par défaut, l'enveloppe de la tuile est dans le système de coordonnées Web Mercator (SRID:3857) utilisant la plage standard du système Web Mercator (-20037508.342789, 20037508.342789). Il s'agit du système de coordonnées le plus couramment utilisé pour les tuiles MVT. Le paramètre facultatif `bounds` peut être utilisé pour générer des tuiles dans n'importe quel système de coordonnées. Il s'agit d'une géométrie qui possède le SRID et l'étendue du carré "Niveau de zoom zéro" dans lequel le système de tuiles XYZ est inscrit.

Le paramètre facultatif `margin` peut être utilisé pour étendre une tuile du pourcentage donné. Par exemple, `margin=0.125` étend la tuile de 12,5%, ce qui équivaut à buffer=512 lorsque la taille de l'étendue de la tuile est de 4096, comme utilisé dans ST_AsMVTGeom. Ceci est utile pour créer un tampon de tuile afin d'inclure des données situées en dehors de la zone visible de la tuile, mais dont l'existence affecte le rendu de la tuile. Par exemple, le nom d'une ville (un point) peut se trouver près du bord d'une tuile, son étiquette doit donc être rendue sur deux tuiles, même si le point se trouve dans la zone visible d'une seule tuile. L'utilisation de tuiles étendues dans une requête inclura le point de la ville dans les deux tuiles. Utilisez une valeur négative pour réduire la tuile à la place. Les valeurs inférieures à -0,5 sont interdites, car cela éliminerait complètement la tuile. Ne spécifiez pas de marge lors de l'utilisation avec `ST_AsMVTGeom`. Voir l'exemple pour ST_AsMVT.

Amélioré : 3.1.0 Ajout d'un paramètre de marge.

Disponibilité : 3.0.0

Exemple : Construction d'une enveloppe de tuiles

```
SELECT ST_AsText( ST_TileEnvelope(2, 1, 1) );

 st_astext
------------------------------
 POLYGON((-10018754.1713945 0,-10018754.1713945 10018754.1713945,0 10018754.1713945,0  ↩
     0,-10018754.1713945 0))

SELECT ST_AsText( ST_TileEnvelope(3, 1, 1, ST_MakeEnvelope(-180, -90, 180, 90, 4326) ) );

                      st_astext
------------------------------------------------------
 POLYGON((-135 45,-135 67.5,-90 67.5,-90 45,-135 45))
```

Voir aussi

ST_MakeEnvelope

7.3.14 ST_HexagonGrid

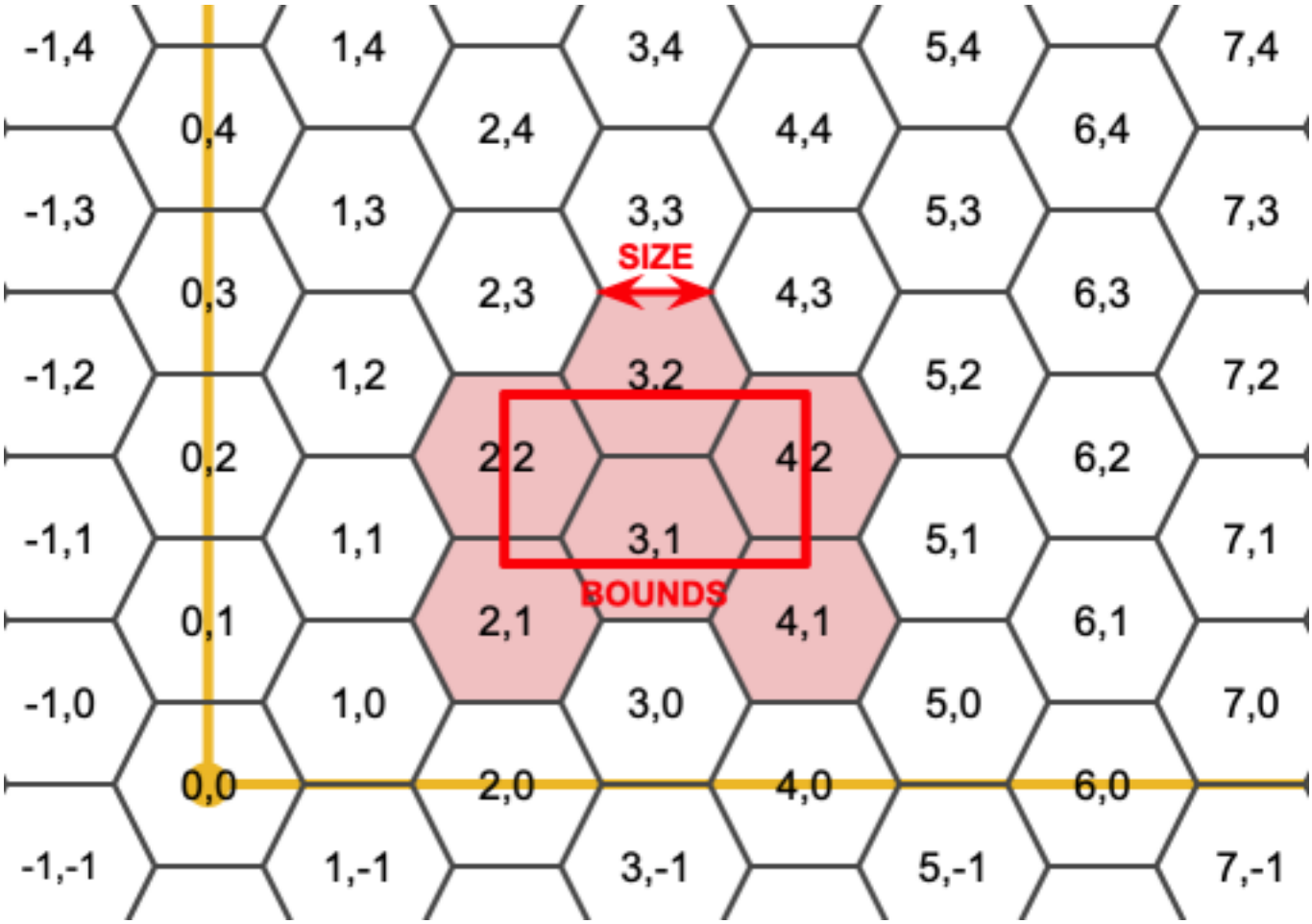
ST_HexagonGrid — Renvoie un ensemble d'hexagones et d'indices de cellules qui couvrent complètement les limites de l'argument géométrie.

Synopsis

setof record **ST_HexagonGrid**(float8 size, geometry bounds);

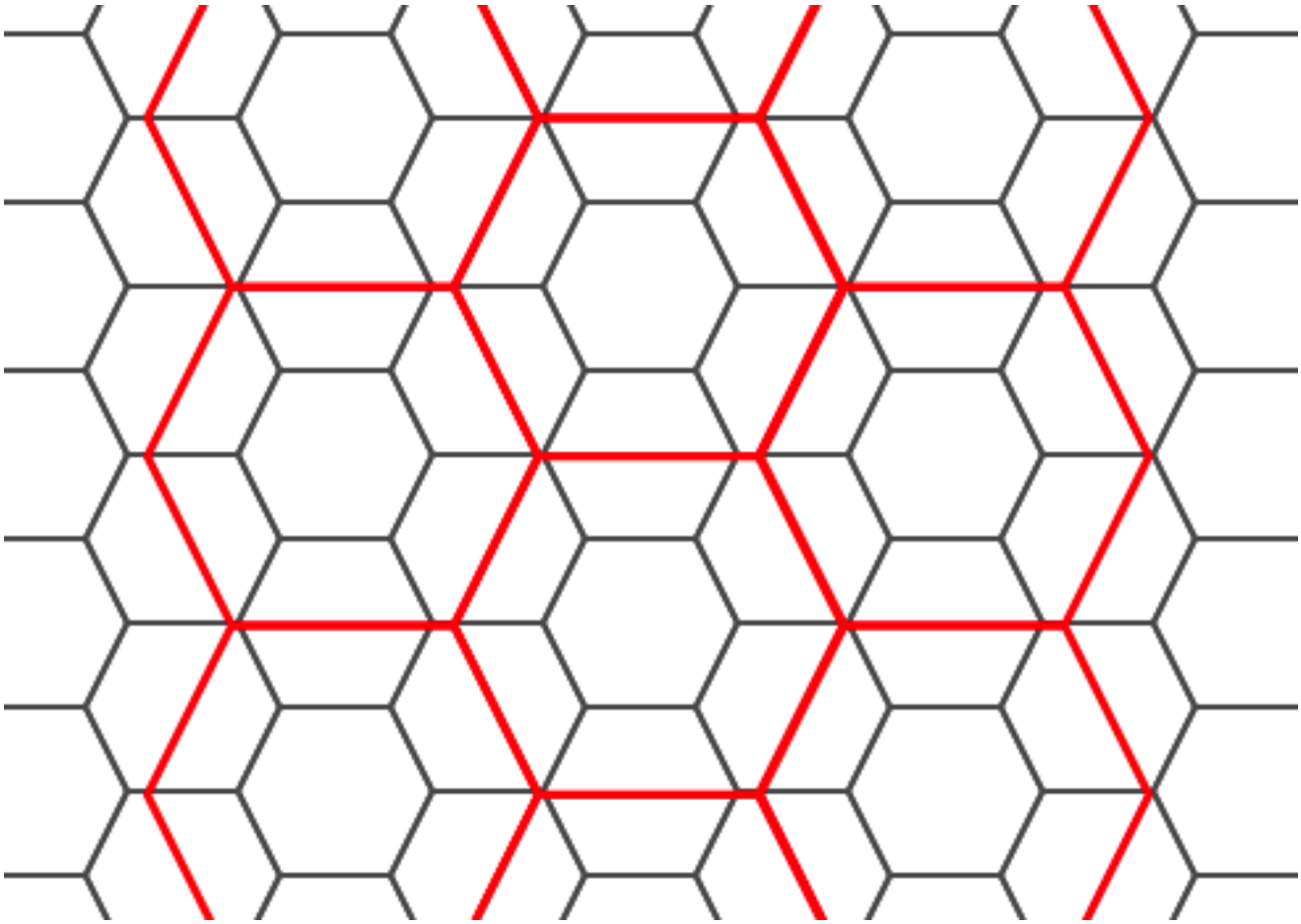
Description

Commence par le concept d'un tuilage hexagonal du plan. (Pas un pavage hexagonal du globe, ce n'est pas le schéma de pavage H3). Pour un SRS plan donné, et une taille d'arête donnée, en commençant à l'origine du SRS, il existe un unique tuilage hexagonal du plan, Tiling(SRS, Size). Cette fonction répond à la question : quels hexagones dans un Tiling(SRS, Size) donné se chevauchent avec une limite donnée.



Le SRS pour les hexagones de sortie est le SRS fourni par la géométrie des limites.

Doubler ou tripler la taille des bords de l'hexagone génère un nouveau pavage parent qui s'adapte au pavage d'origine. Malheureusement, il n'est pas possible de générer des tuiles d'hexagones parents dans lesquelles les tuiles enfants s'insèrent parfaitement.



Disponibilité : 3.1.0

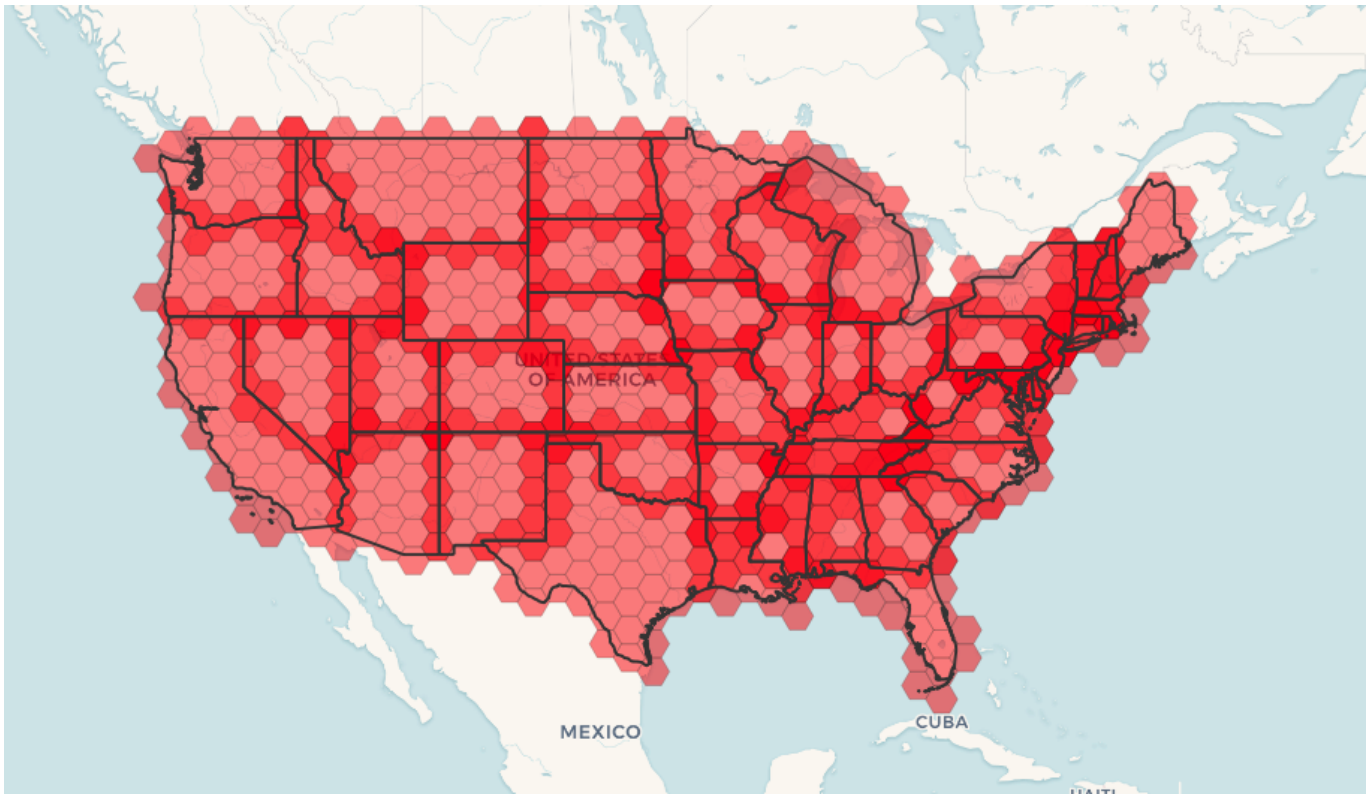
Exemple : Compter les points dans les hexagones

Pour faire un résumé des points par rapport à un pavage hexagonal, générez une grille hexagonale en utilisant l'étendue des points comme limites, puis joignez spatialement cette grille.

```
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;
```

Exemple : Génération de la couverture hexagonale des polygones

Si nous générons un ensemble d'hexagones pour chaque limite de polygone et que nous éliminons par filtrage ceux qui n'intersectent pas leurs hexagones, nous obtenons un pavage pour chaque polygone.



La mise en mosaïque des états donne lieu à une couverture hexagonale de chaque état, et à des hexagones multiples se chevauchant aux frontières entre les états.

Note

Le mot-clé LATERAL est implicite pour les fonctions de retour d'ensemble lorsqu'elles font référence à une table antérieure dans la liste FROM. Ainsi, CROSS JOIN LATERAL, CROSS JOIN, ou tout simplement , sont des constructions équivalentes pour cet exemple.

```
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)
```

Voir aussi

ST_EstimatedExtent, ST_SetSRID, ST_SquareGrid, ST_TileEnvelope

7.3.15 ST_Hexagon

ST_Hexagon — Renvoie un seul hexagone, en utilisant la taille du bord et les coordonnées de la cellule fournies dans l'espace de la grille de l'hexagone.

Synopsis

geometry **ST_Hexagon**(float8 size, integer cell_i, integer cell_j, geometry origin);

Description

Utilise le même concept de tuilage d'hexagones que ST_HexagonGrid, mais génère un seul hexagone à la coordonnée de cellule souhaitée. En option, peut ajuster la coordonnée d'origine du tuilage, l'origine par défaut est à 0,0.

Les hexagones sont générés sans SRID défini, utilisez donc ST_SetSRID pour définir le SRID à celui que vous attendez.

Disponibilité : 3.1.0

Exemple : Création d'un hexagone à l'origine

```
SELECT ST_AsText(ST_SetSRID(ST_Hexagon(1.0, 0, 0), 3857));

POLYGON((-1 0,-0.5
         -0.866025403784439,0.5
         -0.866025403784439,1
         0,0.5
         0.866025403784439,-0.5
         0.866025403784439,-1 0))
```

Voir aussi

ST_TileEnvelope, ST_HexagonGrid, ST_Square

7.3.16 ST_SquareGrid

ST_SquareGrid — Renvoie un ensemble de carrés de grille et d'indices de cellules qui couvrent complètement les limites de l'argument géométrie.

Synopsis

setof record **ST_SquareGrid**(float8 size, geometry bounds);

Description

Commence par le concept de tuilage carré du plan. Pour un SRS plan donné, et une taille d'arête donnée, en commençant à l'origine du SRS, il existe un unique pavage carré du plan, Tiling(SRS, Size). Cette fonction répond à la question : quelles grilles dans un Tiling(SRS, Size) donné se chevauchent avec une limite donnée.

Le SRS des carrés de sortie est le SRS fourni par la géométrie des limites.

Le doublement de la taille du carré ou de son bord génère un nouveau pavage parent qui s'adapte parfaitement au pavage d'origine. Les carrelages standard des cartes Web dans mercator ne sont que des puissances de deux grilles carrées dans le plan mercator.

Disponibilité : 3.1.0

Exemple : Générer une grille de 1 degré pour un pays

La grille remplira toutes les limites du pays, donc si vous voulez seulement des carrés qui touchent le pays, vous devrez filtrer ensuite avec ST_Intersects.

```
WITH grid AS (
SELECT (ST_SquareGrid(1, ST_Transform(geom,4326))).*
FROM admin0 WHERE name = 'Canada'
)
  SELEcT ST_AsText(geom)
  FROM grid
```

Exemple : Compter les points dans les carrés (en utilisant une seule grille découpée)

Pour faire un résumé des points par rapport à un tuilage carré, générez une grille carrée en utilisant l'étendue des points comme limites, puis joignez spatialement à cette grille. Notez que l'étendue estimée peut être différente de l'étendue réelle, soyez donc prudent et assurez-vous au moins d'avoir analysé votre tableau.

```
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
```

Exemple : Compter des points dans des carrés en utilisant un jeu de grille par point

Cette méthode donne le même résultat que le premier exemple mais sera plus lente pour un grand nombre de points

```
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
```

Voir aussi

ST_TileEnvelope, ST_HexagonGrid , ST_EstimatedExtent , ST_SetSRID

7.3.17 ST_Square

ST_Square — Renvoie un seul carré, en utilisant la taille du bord et la coordonnée de la cellule fournies dans l'espace de la grille du carré.

Synopsis

geometry **ST_Square**(float8 size, integer cell_i, integer cell_j, geometry origin);

Description

Utilise le même concept de tuilage carré que ST_SquareGrid, mais génère un seul carré à la coordonnée de cellule souhaitée. En option, peut ajuster la coordonnée d'origine du tuilage, l'origine par défaut est à 0,0.

Les carrés sont générés sans SRID défini, utilisez donc ST_SetSRID pour définir le SRID à celui que vous attendez.

Disponibilité : 3.1.0

Exemple : Création d'un carré à l'origine

```
SELECT ST_AsText(ST_SetSRID(ST_Square(1.0, 0, 0), 3857));

 POLYGON((0 0,0 1,1 1,1 0,0 0))
```

Voir aussi

ST_TileEnvelope, ST_SquareGrid, ST_Hexagon

7.3.18 ST_Letters

ST_Letters — Renvoie les lettres d'entrée rendues sous forme de géométrie avec une position de départ par défaut à l'origine et une hauteur de texte par défaut de 100.

Synopsis

geometry **ST_Letters**(text letters, json font);

Description

Utilise une police intégrée pour rendre une chaîne de caractères sous forme de géométrie multipolygonale. La hauteur de texte par défaut est de 100,0, soit la distance entre le bas d'une descendante et le haut d'une capitale. La position de départ par défaut place le début de la ligne de base à l'origine. Pour remplacer la police, il faut passer une carte json, avec un caractère comme clé, et des TWKB encodés en base64 pour la forme de la police, les polices ayant une hauteur de 1000 unités du bas des descendantes au haut des capitales.

Le texte est généré à l'origine par défaut, donc pour repositionner et redimensionner le texte, appliquez d'abord la fonction `ST_Scale` et ensuite appliquez la fonction `ST_Translate`.

Disponibilité : 3.3.0

Exemple : Génération du mot 'Yo'

```
SELECT ST_AsText(ST_Letters('Yo'), 1);
```

Lettres générées par ST_Letters

Exemple : Mise à l'échelle et déplacement des mots

```
SELECT ST_Translate(ST_Scale(ST_Letters('Yo'), 10, 10), 100,100);
```

Voir aussi

ST_AsTWKB, ST_Scale, ST_Translate

7.4 Fonctions d'accès aux géométries

7.4.1 GeometryType

GeometryType — Renvoie le type d'une géométrie sous forme de texte.

Synopsis

text **GeometryType**(geometry geomA);

Description

Retourne le type de la géométrie, par exemple : 'LINESTRING', 'POLYGON', 'MULTIPOINT', etc.

OGC SPEC s2.1.1.1 - Retourne le nom du sous type instanciable de la géométrie. Le nom est retourné sous forme de texte.

Note
Cette fonction indique si la géométrie comporte une dimension de type mesure, en retournant un texte de la forme 'POINTM'.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29  ↩
    29.07)'));
 geometrytype
--------------
 LINESTRING
```

```
SELECT ST_GeometryType(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) ↩
                    ) )'));
                        --result
                        POLYHEDRALSURFACE
```

```
SELECT GeometryType(geom) as result
  FROM
    (SELECT
       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 geom
    ) AS g;
 result
--------
 TIN
```

Voir aussi

ST_GeometryType

7.4.2 ST_Boundary

ST_Boundary — Renvoie la limite d'une géométrie.

Synopsis

geometry **ST_Boundary**(geometry geomA);

Description

Renvoie l'ensemble formant la frontière finie de cette géométrie. La notion de frontière est définie dans la section 3.12.3.2 des spécifications OGC. Le résultat de cette fonction est un ensemble topologiquement fermé, représentable avec les types de base, comme décrit dans la section 3.12.2 des spécifications OGC.

Effectué par le module GEOS

Note
Avant la version 2.0.0, cette fonction renvoie une exception si une `GEOMETRYCOLLECTION` est passée en paramètre. A partir de la 2.0.0, la fonction renvoie null (paramètre non supporté).

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. OGC SPEC s2.1.1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1.17

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

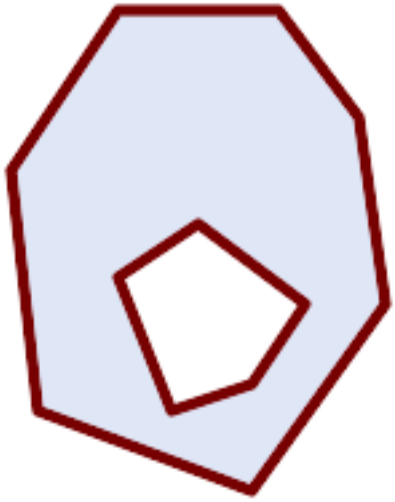
Amélioration : 2.1.0 introduction du support pour Triangle

Modifié : 3.2.0 support pour TIN, n'utilise pas geos, ne linéarise pas les courbes

Exemples

Ligne avec les points de démarcation superposés

```
SELECT ST_Boundary(geom)
FROM (SELECT 'LINESTRING(100 150,50 60,  ↩
     70 80, 160 170)'::geometry As geom) As f;

ST_AsText output

MULTIPOINT((100 150),(160 170))
```



trous de polygone avec une multi-lignes en limite

```
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;

ST_AsText 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))
```

```
SELECT ST_AsText(ST_Boundary(ST_GeomFromText('LINESTRING(1 1,0 0, -1 1)')));
st_astext
-----------
MULTIPOINT((1 1),(-1 1))

SELECT ST_AsText(ST_Boundary(ST_GeomFromText('POLYGON((1 1,0 0, -1 1, 1 1))')));
st_astext
----------
LINESTRING(1 1,0 0,-1 1,1 1)
```

```
--Using a 3d polygon
SELECT ST_AsEWKT(ST_Boundary(ST_GeomFromEWKT('POLYGON((1 1 1,0 0 1, -1 1 1, 1 1 1))')));

st_asewkt
-----------------------------------
LINESTRING(1 1 1,0 0 1,-1 1 1,1 1 1)

--Using a 3d multilinestring
SELECT ST_AsEWKT(ST_Boundary(ST_GeomFromEWKT('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) )')));

st_asewkt
----------
MULTIPOINT((-1 1 1),(1 1 0.75))
```

Voir aussi

ST_AsText, ST_ExteriorRing, ST_MakePolygon

7.4.3 ST_BoundingDiagonal

ST_BoundingDiagonal — Retourne la diagonale de la boîte englobante pour la géométrie en argument.

Synopsis

geometry **ST_BoundingDiagonal**(geometry geom, boolean fits=false);

Description

Renvoie la diagonale de la boîte de délimitation de la géométrie fournie sous la forme d'une LineString. La diagonale est une LineString à 2 points avec les valeurs minimales de chaque dimension dans son point de départ et les valeurs maximales dans son point d'arrivée. Si la géométrie d'entrée est vide, la diagonale est une LINESTRING EMPTY.

Le paramètre facultatif `fits` spécifie si le meilleur ajustement est nécessaire. S'il est faux, la diagonale d'une boîte de délimitation un peu plus grande peut être acceptée (ce qui est plus rapide à calculer pour les géométries comportant de nombreux sommets). Dans les deux cas, la boîte de délimitation de la ligne diagonale renvoyée couvre toujours la géométrie d'entrée.

La géométrie renvoyée conserve le SRID et la dimensionnalité (présence Z et M) de la géométrie en entrée.

Note
Dans les cas dégénérés (c'est-à-dire un seul sommet dans l'entrée), la ligne retournée sera formellement invalide (pas d'intérieur). Le résultat est toujours topologiquement valide.

Disponibilité : 2.2.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Exemples

```
-- Get the minimum X in a buffer around a point
SELECT ST_X(ST_StartPoint(ST_BoundingDiagonal(
  ST_Buffer(ST_Point(0,0),10)
)));
 st_x
------
  -10
```

Voir aussi

ST_StartPoint, ST_EndPoint, ST_X, ST_Y, ST_Z, ST_M, ST_Envelope

7.4.4 ST_CoordDim

ST_CoordDim — Renvoie la dimension des coordonnées d'une géométrie.

Synopsis

integer **ST_CoordDim**(geometry geomA);

Description

Retourne la dimension des coordonnées d'une valeur ST_Geometry.

Alias SQL/MM pour la fonction ST_NDims

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.3

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_CoordDim('CIRCULARSTRING(1 2 3, 1 3 4, 5 6 7, 8 9 10, 11 12 13)');
                        ---result--
                                3

                                SELECT ST_CoordDim(ST_Point(1,2));
                        --result--
                                2
```

Voir aussi

ST_NDims

7.4.5 ST_Dimension

ST_Dimension — Renvoie la dimension topologique d'une géométrie.

Synopsis

integer **ST_Dimension**(geometry g);

Description

Renvoie la dimension topologique de cet objet Geometry, qui doit être inférieure ou égale à la dimension des coordonnées. OGC SPEC s2.1.1.1 - renvoie 0 pour `POINT`, 1 pour `LINESTRING`, 2 pour `POLYGON`, et la plus grande dimension des composants d'une `GEOMETRYCOLLECTION`. Si la dimension est inconnue (par exemple, pour une `GEOMETRYCOLLECTION` vide), 0 est renvoyé.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.2

Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques. Ne renvoie plus une exception si une GEOMETRY EMPTY est passée.

Note
Avant la version 2.0.0, cette fonction lève une exception si elle est utilisée avec une géométrie vide.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_Dimension('GEOMETRYCOLLECTION(LINESTRING(1 1,0 0),POINT(0 0))');
ST_Dimension
------------
1
```

Voir aussi

ST_NDims

7.4.6 ST_Dump

ST_Dump — Renvoie un ensemble de lignes `geometry_dump` pour les composants d'une géométrie.

Synopsis

geometry_dump[] **ST_Dump**(geometry g1);

Description

Une fonction de retour d'ensemble (SRF) qui extrait les composants d'une géométrie. Elle renvoie un ensemble de geometry_dump lignes, chacune contenant une géométrie (champ *geom*) et un tableau d'entiers (champ *path*).

Pour un type de géométrie atomique (POINT, LINESTRING, POLYGONE), un seul enregistrement est renvoyé avec un tableau *path* vide et la géométrie en entrée en tant que *geom*. Pour une collection ou une géométrie multiple, un enregistrement est renvoyé pour chacun des composants de la collection, et le *path* indique la position du composant à l'intérieur de la collection.

ST_Dump est utile pour développer les géométries. C'est l'inverse d'un ST_Collect / GROUP BY, en ce sens qu'il crée de nouvelles lignes. Par exemple, il peut être utilisé pour développer les MULTIPOLYGONES en POLYGONES.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Disponibilité : PostGIS 1.0.0RC1. Nécessite PostgreSQL 7.3 ou plus.

Note
Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples standard

```
SELECT sometable.field1, sometable.field1,
      (ST_Dump(sometable.geom)).geom AS geom
FROM sometable;

-- Break a compound curve into its constituent linestrings and circularstrings
SELECT ST_AsEWKT(a.geom), ST_HasArc(a.geom)
  FROM ( SELECT (ST_Dump(p_geom)).geom AS geom
         FROM (SELECT ST_GeomFromEWKT('COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0  ↩
             1))') AS p_geom) AS b
        ) AS a;
          st_asewkt           | st_hasarc
------------------------------+----------
 CIRCULARSTRING(0 0,1 1,1 0) | t
 LINESTRING(1 0,0 1)         | f
(2 rows)
```

Exemples TIN, Triangle et Surfaces Polyédriques

```
-- Polyhedral surface example
-- Break a Polyhedral surface into its faces
SELECT (a.p_geom).path[1] As path, ST_AsEWKT((a.p_geom).geom) As geom_ewkt
  FROM (SELECT ST_Dump(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 p_geom )  AS a;

 path |                geom_ewkt
------+-----------------------------------------
    1 | POLYGON((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0))
    2 | POLYGON((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0))
    3 | POLYGON((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0))
    4 | POLYGON((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0))
    5 | POLYGON((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0))
    6 | POLYGON((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
```

```
-- TIN --
SELECT (g.gdump).path, ST_AsEWKT((g.gdump).geom) as wkt
  FROM
    (SELECT
       ST_Dump( 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 gdump
    ) AS g;
-- result --
 path |                  wkt
------+---------------------------------------
 {1}  | TRIANGLE((0 0 0,0 0 1,0 1 0,0 0 0))
 {2}  | TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
```

Voir aussi

geometry_dump, Section 13.6, ST_Collect, ST_GeometryN

7.4.7 ST_DumpPoints

ST_DumpPoints — Renvoie un ensemble de lignes `geometry_dump` pour les coordonnées dans une géométrie.

Synopsis

geometry_dump[] **ST_DumpPoints**(geometry geom);

Description

Une fonction de retour d'ensemble (SRF) qui extrait les coordonnées (sommets) d'une géométrie. Elle renvoie un ensemble de geometry_dump lignes, chacune contenant une géométrie (champ *geom*) et un tableau d'entiers (champ *path*).

- le champ *geom* `POINT` représente les coordonnées de la géométrie fournie.
- le champ *path* (un `integer[]`) est un index énumérant les positions des coordonnées dans les éléments de la géométrie fournie. Les indices sont basés sur 1. Par exemple, pour un `LINESTRING`, les chemins sont `{i}` où `i` est la `nième` coordonnée dans le `LINESTRING`. Pour un `POLYGONE`, les chemins sont `{i,j}` où `i` est le numéro de l'anneau (1 est l'anneau extérieur ; les anneaux intérieurs suivent) et `j` est la position de la coordonnée dans l'anneau.

Pour obtenir une géométrie unique contenant les coordonnées, utilisez ST_Points.

Amélioré : 2.1.0 Vitesse plus rapide. Réimplémentation en C natif.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Disponibilité : 1.5.0

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

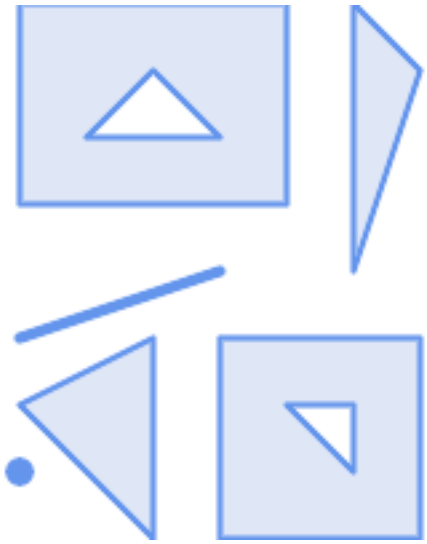
Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Classique Éclater un tableau de lignes en nœuds

```
SELECT edge_id, (dp).path[1] As index, ST_AsText((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;
 edge_id | index |    wktnode
---------+-------+--------------
       1 |     1 | POINT(1 2)
       1 |     2 | POINT(3 4)
       1 |     3 | POINT(10 10)
       2 |     1 | POINT(3 5)
       2 |     2 | POINT(5 6)
       2 |     3 | POINT(9 10)
```

Exemples de géométrie standard



```
SELECT path, ST_AsText(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;

   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)
```

Exemples TIN, Triangle et Surfaces Polyédriques

```
-- Polyhedral surface cube --
SELECT (g.gdump).path, ST_AsEWKT((g.gdump).geom) as wkt
  FROM
    (SELECT
       ST_DumpPoints(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 gdump
```

```
    ) AS g;
-- result --
  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)
```

```
-- Triangle --
SELECT (g.gdump).path, ST_AsText((g.gdump).geom) as wkt
  FROM
    (SELECT
       ST_DumpPoints( ST_GeomFromEWKT('TRIANGLE ((
                0 0,
                0 9,
                9 0,
                0 0
            ))') ) AS gdump
    ) AS g;
-- result --
 path |    wkt
------+------------
 {1}  | POINT(0 0)
 {2}  | POINT(0 9)
 {3}  | POINT(9 0)
 {4}  | POINT(0 0)
```

```
-- TIN --
SELECT (g.gdump).path, ST_AsEWKT((g.gdump).geom) as wkt
  FROM
    (SELECT
       ST_DumpPoints( 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 gdump
    ) AS g;
-- result --
  path   |     wkt
---------+--------------
 {1,1,1} | POINT(0 0 0)
 {1,1,2} | POINT(0 0 1)
 {1,1,3} | POINT(0 1 0)
 {1,1,4} | 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(0 0 0)
(8 rows)
```

Voir aussi

geometry_dump, Section 13.6, ST_Dump, ST_DumpRings, ST_Points

7.4.8 ST_DumpSegments

ST_DumpSegments — Renvoie un ensemble de lignes `geometry_dump` pour les segments d'une géométrie.

Synopsis

geometry_dump[] **ST_DumpSegments**(geometry geom);

Description

Une fonction de retour d'ensemble (SRF) qui extrait les segments d'une géométrie. Elle renvoie un ensemble de geometry_dump lignes, chacune contenant une géométrie (champ *`geom`*) et un tableau d'entiers (champ *`path`*).

- le champ *`geom`* `LINESTRING`s représente les segments linéaires de la géométrie fournie, tandis que `CIRCULARSTRING`s représente les segments d'arc.
- le champ *`path`* (un `integer[]`) est un index énumérant les positions des points de départ des segments dans les éléments de la géométrie fournie. Les indices sont basés sur 1. Par exemple, pour un `LINESTRING`, les chemins sont `{i}` où `i` est le `nième` point de départ du segment dans le `LINESTRING`. Pour un `POLYGONE`, les chemins sont `{i,j}` où `i` est le numéro de l'anneau (1 est l'anneau extérieur ; les anneaux intérieurs suivent) et `j` est la position du point de départ du segment dans l'anneau.

Disponibilité : 3.2.0

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples de géométrie standard

```
SELECT path, ST_AsText(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;

  path   &#x2502;      st_astext
---------------------------------
 {1,1}   &#x2502; LINESTRING(1 1,3 3)
 {1,2}   &#x2502; LINESTRING(3 3,4 4)
 {2,1,1} &#x2502; LINESTRING(5 5,6 6)
 {2,1,2} &#x2502; LINESTRING(6 6,7 7)
 {2,1,3} &#x2502; LINESTRING(7 7,5 5)
(5 rows)
```

Exemples de TIN et de triangles

```
-- Triangle --
SELECT path, ST_AsText(geom)
FROM (
    SELECT (ST_DumpSegments(g.geom)).*
    FROM (SELECT 'TRIANGLE((
        0 0,
        0 9,
        9 0,
        0 0
    ))'::geometry AS geom
        ) AS g
) j;

 path  &#x2502;      st_astext
 ---------------------------------
 {1,1} &#x2502; LINESTRING(0 0,0 9)
 {1,2} &#x2502; LINESTRING(0 9,9 0)
 {1,3} &#x2502; LINESTRING(9 0,0 0)
(3 rows)
```

```
-- TIN --
SELECT path, ST_AsEWKT(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;

  path   &#x2502;        st_asewkt
  --------------------------------
 {1,1,1} &#x2502; LINESTRING(0 0 0,0 0 1)
 {1,1,2} &#x2502; LINESTRING(0 0 1,0 1 0)
 {1,1,3} &#x2502; LINESTRING(0 1 0,0 0 0)
 {2,1,1} &#x2502; LINESTRING(0 0 0,0 1 0)
 {2,1,2} &#x2502; LINESTRING(0 1 0,1 1 0)
 {2,1,3} &#x2502; LINESTRING(1 1 0,0 0 0)
(6 rows)
```

Voir aussi

geometry_dump, Section 13.6, ST_Dump, ST_DumpRings

7.4.9 ST_DumpRings

ST_DumpRings — Renvoie un ensemble de lignes `geometry_dump` pour les anneaux extérieurs et intérieurs d'un polygone.

Synopsis

geometry_dump[] **ST_DumpRings**(geometry a_polygon);

Description

Une fonction de retour d'ensemble (SRF) qui extrait les anneaux d'un polygone. Elle renvoie un ensemble de geometry_dump lignes, chacune contenant une géométrie (champ *`geom`*) et un tableau d'entiers (champ *`path`*).

Le champ *`geom`* contient chaque anneau sous forme de POLYGONE. Le champ *`path`* est un tableau d'entiers de longueur 1 contenant l'indice de l'anneau du polygone. L'anneau extérieur (coquille) a l'indice 0. Les anneaux intérieurs (trous) ont des indices de 1 et plus.

Note
Cela ne fonctionne que pour les géométries POLYGON. Il ne fonctionne pas pour les MULTIPOLYGONS

Disponibilité : PostGIS 1.1.3. Nécessite PostgreSQL 7.3 ou plus.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Forme générale de la requête.

```
SELECT polyTable.field1, polyTable.field1,
          (ST_DumpRings(polyTable.geom)).geom As geom
FROM polyTable;
```

Un polygone avec un seul trou.

```
SELECT path, ST_AsEWKT(geom) As geom
        FROM ST_DumpRings(
                ST_GeomFromEWKT('POLYGON((-8149064 5133092 1,-8149064 5132986 1,-8148996  ↩
                    5132839 1,-8148972 5132767 1,-8148958 5132508 1,-8148941 5132466  ↩
                    1,-8148924 5132394 1,
                -8148903 5132210 1,-8148930 5131967 1,-8148992 5131978 1,-8149237 5132093  ↩
                    1,-8149404 5132211 1,-8149647 5132310 1,-8149757 5132394 1,
                -8150305 5132788 1,-8149064 5133092 1),
                (-8149362 5132394 1,-8149446 5132501 1,-8149548 5132597 1,-8149695 5132675  ↩
                    1,-8149362 5132394 1))')
                )  as foo;
 path |                                              geom
------------------------------------------------------------------------------------------------------------

  {0} | POLYGON((-8149064 5133092 1,-8149064 5132986 1,-8148996 5132839 1,-8148972 5132767  ↩
      1,-8148958 5132508 1,
          |           -8148941 5132466 1,-8148924 5132394 1,
          |           -8148903 5132210 1,-8148930 5131967 1,
          |           -8148992 5131978 1,-8149237 5132093 1,
          |           -8149404 5132211 1,-8149647 5132310 1,-8149757 5132394 1,-8150305  ↩
              5132788 1,-8149064 5133092 1))
  {1} | POLYGON((-8149362 5132394 1,-8149446 5132501 1,
          |           -8149548 5132597 1,-8149695 5132675 1,-8149362 5132394 1))
```

Voir aussi

geometry_dump, Section 13.6, ST_Dump, ST_ExteriorRing, ST_InteriorRingN

7.4.10 ST_EndPoint

ST_EndPoint — Renvoie le dernier point d'une LineString ou CircularLineString.

Synopsis

geometry **ST_EndPoint**(geometry g);

Description

Renvoie le dernier point d'une géométrie `LINESTRING` ou `CIRCULARLINESTRING` comme un `POINT`. Renvoie `NULL` si l'entrée n'est pas une `LINESTRING` ou `CIRCULARLINESTRING`.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.1.4

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Note

Modifié : 2.0.0 ne fonctionne plus avec les MultiLineStrings à géométrie unique. Dans les anciennes versions de PostGIS, une MultiLineString à géométrie unique fonctionnait avec cette fonction et renvoyait le point final. Dans la version 2.0.0, elle renvoie NULL comme toute autre MultiLineString. L'ancien comportement était une fonctionnalité non documentée, mais les personnes qui supposaient que leurs données étaient stockées en tant que LINESTRING peuvent voir ces dernières retourner NULL dans la version 2.0.0.

Exemples

Point final d'une ligne

```
postgis=# SELECT ST_AsText(ST_EndPoint('LINESTRING(1 1, 2 2, 3 3)'::geometry));
 st_astext
------------
 POINT(3 3)
```

Le point final d'une ligne qui n'en est pas une est NULL

```
SELECT ST_EndPoint('POINT(1 1)'::geometry) IS NULL AS is_null;
  is_null
----------
 t
```

Point final d'une ligne 3D

```
--3d endpoint
SELECT ST_AsEWKT(ST_EndPoint('LINESTRING(1 1 2, 1 2 3, 0 0 5)'));
  st_asewkt
--------------
 POINT(0 0 5)
```

Point d'arrivée d'une CircularString

```
SELECT ST_AsText(ST_EndPoint('CIRCULARSTRING(5 2,-3 1.999999, -2 1, -4 2, 6 3)'::geometry)) ↩
    ;
 st_astext
------------
 POINT(6 3)
```

Voir aussi

ST_PointN, ST_StartPoint

7.4.11 ST_Envelope

ST_Envelope — Renvoie une géométrie représentant la boîte de délimitation d'une géométrie.

Synopsis

geometry **ST_Envelope**(geometry g1);

Description

Renvoie la boîte de délimitation minimale en double précision (float8) pour la géométrie fournie, en tant que géométrie. Le polygone est défini par les points d'angle de la boîte de délimitation ((MINX, MINY), (MINX, MAXY), (MAXX, MAXY), (MAXX, MINY), (MINX, MINY)). (PostGIS ajoutera également une coordonnée ZMIN/ZMAX).

D'autres cas (lignes verticales, points) retourneront une géométrie de dimension inférieure à POLYGON, c'est-à-dire POINT ou LINESTRING.

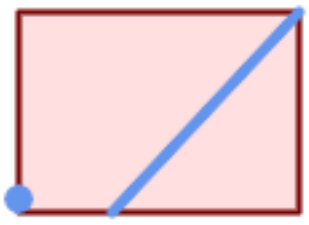
Disponibilité : 1.5.0 changement pour renvoyer un type double précision à la place de float4

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.19

Exemples

```
SELECT ST_AsText(ST_Envelope('POINT(1 3)'::geometry));
 st_astext
------------
 POINT(1 3)
(1 row)


SELECT ST_AsText(ST_Envelope('LINESTRING(0 0, 1 3)'::geometry));
                  st_astext
-------------------------------
 POLYGON((0 0,0 3,1 3,1 0,0 0))
(1 row)


SELECT ST_AsText(ST_Envelope('POLYGON((0 0, 0 1, 1.0000001 1, 1.0000001 0, 0 0))'::geometry ↩
    ));
                                                        st_astext
------------------------------------------------------------------
 POLYGON((0 0,0 1,1.00000011920929 1,1.00000011920929 0,0 0))
(1 row)
SELECT ST_AsText(ST_Envelope('POLYGON((0 0, 0 1, 1.0000000001 1, 1.0000000001 0, 0 0))':: ↩
    geometry));
                                                        st_astext
------------------------------------------------------------------
 POLYGON((0 0,0 1,1.00000011920929 1,1.00000011920929 0,0 0))
(1 row)

SELECT Box3D(geom), Box2D(geom), ST_AsText(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;
```



Enveloppe d'un point et d'une ligne.

```
SELECT ST_AsText(ST_Envelope(
                ST_Collect(
                        ST_GeomFromText('LINESTRING(55 75,125 150)'),
                                ST_Point(20, 80))
```

```
                                    )) As wktenv;
wktenv
-----------
POLYGON((20 75,20 150,125 150,125 75,20 75))
```

Voir aussi

Box2D, Box3D, ST_OrientedEnvelope

7.4.12 ST_ExteriorRing

ST_ExteriorRing — Renvoie une ligne représentant l'anneau extérieur d'un polygone.

Synopsis

geometry **ST_ExteriorRing**(geometry a_polygon);

Description

Renvoie une LINESTRING représentant l'anneau extérieur (coquille) d'un POLYGONE. Renvoie NULL si la géométrie n'est pas un polygone.

Note
Cette fonction ne prend pas en charge les MULTIPOLYGONES. Pour les MULTIPOLYGONs, utilisez conjointement avec ST_GeometryN ou ST_Dump

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. 2.1.5.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 8.2.3, 8.3.3

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
--If you have a table of polygons
SELECT gid, ST_ExteriorRing(geom) AS ering
FROM sometable;

--If you have a table of MULTIPOLYGONs
--and want to return a MULTILINESTRING composed of the exterior rings of each polygon
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;

--3d Example
SELECT ST_AsEWKT(
        ST_ExteriorRing(
        ST_GeomFromEWKT('POLYGON((0 0 1, 1 1 1, 1 2 1, 1 1 1, 0 0 1))')
        )
);
```

```
st_asewkt
---------
LINESTRING(0 0 1,1 1 1,1 2 1,1 1 1,0 0 1)
```

Voir aussi

ST_InteriorRingN, ST_Boundary, ST_NumInteriorRings

7.4.13 ST_GeometryN

ST_GeometryN — Renvoie un élément d'une collection de géométries.

Synopsis

geometry **ST_GeometryN**(geometry geomA, integer n);

Description

Renvoie la géométrie du Nième élément basé sur 1 d'une géométrie d'entrée qui est une GEOMETRYCOLLECTION, MULTIPOINT, MULTILINESTRING, MULTICURVE, MULTI)POLYGON ou POLYHEDRALSURFACE. Sinon, renvoie NULL.

Note
L'index commence à 1 pour respecter les spécificarions OGC depuis la version 0.8.0. Dans les versions antérieures, l'index commençait à 0.

Note
Pour extraire tous les éléments d'une géométrie, ST_Dump est plus efficace et fonctionne pour les géométries atomiques.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Changement : 2.0.0. Les versions antérieures renvoient NULL pour les géometries simples (un seul objet). Renvoie désormais la géométrie pour le cas ST_GeometryN(..,1).

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 9.1.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples standard

```
--Extracting a subset of points from a 3d multipoint
SELECT n, ST_AsEWKT(ST_GeometryN(geom, n)) As geomewkt
FROM (
VALUES (ST_GeomFromEWKT('MULTIPOINT((1 2 7), (3 4 7), (5 6 7), (8 9 10))') ),
( ST_GeomFromEWKT('MULTICURVE(CIRCULARSTRING(2.5 2.5,4.5 2.5, 3.5 3.5), (10 11, 12 11))') )
        )As foo(geom)
        CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(geom);

 n |               geomewkt
---+----------------------------------------
 1 | POINT(1 2 7)
 2 | POINT(3 4 7)
 3 | POINT(5 6 7)
 4 | POINT(8 9 10)
 1 | CIRCULARSTRING(2.5 2.5,4.5 2.5,3.5 3.5)
 2 | LINESTRING(10 11,12 11)


--Extracting all geometries (useful when you want to assign an id)
SELECT gid, n, ST_GeometryN(geom, n)
FROM sometable CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(geom);
```

Exemples TIN, Triangle et Surfaces Polyédriques

```
-- Polyhedral surface example
-- Break a Polyhedral surface into its faces
SELECT ST_AsEWKT(ST_GeometryN(p_geom,3)) As geom_ewkt
  FROM (SELECT 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 p_geom )  AS a;

                geom_ewkt
-----------------------------------------
 POLYGON((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0))
```

```
-- TIN --
SELECT ST_AsEWKT(ST_GeometryN(geom,2)) as wkt
  FROM
    (SELECT
       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 geom
    ) AS g;
-- result --
                 wkt
------------------------------------
 TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
```

Voir aussi

ST_Dump, ST_NumGeometries

7.4.14 ST_GeometryType

ST_GeometryType — Renvoie le type SQL-MM d'une géométrie sous forme de texte.

Synopsis

text **ST_GeometryType**(geometry g1);

Description

Renvoie le type de la géométrie sous forme de texte, par exemple : 'ST_LineString', 'ST_Polygon','ST_MultiPolygon' etc. Cette fonction diffère de GeometryType(geometry) par la casse du texte renvoyé et par le préfixe ST_ en début de texte. N'indique pas si la géométrie comporte une dimension MESURE.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.4

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27  ↩
    29.31,77.29 29.07)'));
                        --result
                        ST_LineString
```

```
SELECT ST_GeometryType(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) ↩
                    ) )'));
                        --result
                        ST_PolyhedralSurface
```

```
SELECT ST_GeometryType(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) ↩
                    ) )'));
                        --result
                        ST_PolyhedralSurface
```

```
SELECT ST_GeometryType(geom) as result
  FROM
    (SELECT
       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 geom
    ) AS g;
 result
--------
 ST_Tin
```

Voir aussi

GeometryType

7.4.15 ST_HasArc

ST_HasArc — Teste si une géométrie contient un arc de cercle

Synopsis

boolean **ST_HasArc**(geometry geomA);

Description

Renvoie true si une géométrie ou une collection de géométries contient une circular string

Disponibilité : 1.2.3 ?

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_HasArc(ST_Collect('LINESTRING(1 2, 3 4, 5 6)', 'CIRCULARSTRING(1 1, 2 3, 4 5, 6 ↩
    7, 5 6)'));
               st_hasarc
               --------
               t
```

Voir aussi

ST_CurveToLine, ST_LineToCurve

7.4.16 ST_InteriorRingN

ST_InteriorRingN — Renvoie le Nième anneau intérieur (trou) d'un polygone.

Synopsis

geometry **ST_InteriorRingN**(geometry a_polygon, integer n);

Description

Renvoie le Nième anneau intérieur (trou) d'une géométrie POLYGONE sous forme de LINESTRING. L'indice commence à 1. Renvoie NULL si la géométrie n'est pas un polygone ou si l'indice est hors de la plage.

Note
Cette fonction ne prend pas en charge les MULTIPOLYGONES. Pour les MULTIPOLYGONs, utilisez conjointement avec ST_GeometryN ou ST_Dump

---

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 8.2.6, 8.3.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsText(ST_InteriorRingN(geom, 1)) As geom
FROM (SELECT ST_BuildArea(
               ST_Collect(ST_Buffer(ST_Point(1,2), 20,3),
                       ST_Buffer(ST_Point(1, 2), 10,3))) As geom
               )  as foo;
```

Voir aussi

ST_ExteriorRing, ST_BuildArea, ST_Collect, ST_Dump, ST_NumInteriorRing, ST_NumInteriorRings

7.4.17 ST_NumCurves

ST_NumCurves — Renvoie le nombre de courbes composantes d'une CompoundCurve.

Synopsis

integer **ST_NumCurves**(geometry a_compoundcurve);

Description

Renvoie le nombre de courbes composantes d'une CompoundCurve, zéro pour une CompoundCurve vide, ou NULL pour une entrée qui n'est pas une CompoundCurve.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 8.2.6, 8.3.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
-- Returns 3
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)
  )');

-- Returns 0
SELECT ST_NumCurves('COMPOUNDCURVE EMPTY');
```

Voir aussi

ST_CurveN, ST_Dump, ST_ExteriorRing, ST_NumInteriorRings, ST_NumGeometries

7.4.18 ST_CurveN

ST_CurveN — Renvoie la Nième courbe composante d'une CompoundCurve.

Synopsis

geometry **ST_CurveN**(geometry a_compoundcurve, integer index);

Description

Renvoie la Nièmecourbe composante d'une CompoundCurve. L'index commence à 1. Renvoie NULL si la géométrie n'est pas une CompoundCurve ou si l'index est hors de la plage.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 8.2.6, 8.3.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsText(ST_CurveN('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)
  )', 1));
```

Voir aussi

ST_NumCurves, ST_Dump, ST_ExteriorRing, ST_NumInteriorRings, ST_NumGeometries

7.4.19 ST_IsClosed

ST_IsClosed — Teste si les points de départ et d'arrivée d'une LineString coïncident. Pour une PolyhedralSurface, teste si elle est fermée (volumétrique).

Synopsis

boolean **ST_IsClosed**(geometry g);

Description

Renvoie `TRUE` si les premier et dernier points de la `LINESTRING` sont identiques. Pour les surface polyhédriques, indique si la surface est surfacique (ouverte) ou volumétrique (fermée).

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.1.5, 9.3.3

Note
SQL-MM définit le résultat de `ST_IsClosed(NULL)` comme étant 0, alors que PostGIS renvoie `NULL`.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples de lignes et de points

```
postgis=# SELECT ST_IsClosed('LINESTRING(0 0, 1 1)'::geometry);
 st_isclosed
-------------
 f
(1 row)

postgis=# SELECT ST_IsClosed('LINESTRING(0 0, 0 1, 1 1, 0 0)'::geometry);
 st_isclosed
-------------
 t
(1 row)

postgis=# SELECT ST_IsClosed('MULTILINESTRING((0 0, 0 1, 1 1, 0 0),(0 0, 1 1))'::geometry);
 st_isclosed
-------------
 f
(1 row)
```

```
postgis=# SELECT ST_IsClosed('POINT(0 0)'::geometry);
 st_isclosed
-------------
 t
(1 row)

postgis=# SELECT ST_IsClosed('MULTIPOINT((0 0), (1 1))'::geometry);
 st_isclosed
-------------
 t
(1 row)
```

Exemples : surfaces polyédriques

```
-- A cube --
                SELECT ST_IsClosed(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) ↩
                    ) )'));

 st_isclosed
-------------
 t


 -- Same as cube but missing a side --
 SELECT ST_IsClosed(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)) )'));

 st_isclosed
-------------
 f
```

Voir aussi

ST_IsRing

7.4.20 ST_IsCollection

ST_IsCollection — Teste si une géométrie est un type de collection de géométrie.

Synopsis

boolean **ST_IsCollection**(geometry g);

Description

Renvoie `TRUE` si le type de géométrie de l'argument est un type de collection de géométrie. Les types de collection sont les suivants :

- GEOMETRYCOLLECTION
- MULTI{POINT,POLYGON,LINESTRING,CURVE,SURFACE}
- COMPOUNDCURVE

Note
Cette fonction analyse le type de la géométrie. Renvoie `TRUE` pour les collections vides ou ne contenant qu'un seul élément.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
postgis=# SELECT ST_IsCollection('LINESTRING(0 0, 1 1)'::geometry);
 st_iscollection
-----------------
 f
(1 row)

postgis=# SELECT ST_IsCollection('MULTIPOINT EMPTY'::geometry);
 st_iscollection
-----------------
 t
(1 row)

postgis=# SELECT ST_IsCollection('MULTIPOINT((0 0))'::geometry);
 st_iscollection
-----------------
 t
(1 row)

postgis=# SELECT ST_IsCollection('MULTIPOINT((0 0), (42 42))'::geometry);
 st_iscollection
-----------------
 t
(1 row)

postgis=# SELECT ST_IsCollection('GEOMETRYCOLLECTION(POINT(0 0))'::geometry);
 st_iscollection
-----------------
 t
(1 row)
```

Voir aussi

ST_NumGeometries

7.4.21 ST_IsEmpty

ST_IsEmpty — Teste si une géométrie est vide.

Synopsis

boolean **ST_IsEmpty**(geometry geomA);

Description

Renvoie true si cette géométrie est une géométrie vide. Si vrai, alors cette géométrie représente une collection de géométrie vide, un polygone, un point, etc.

Note
La norme SQL-MM stipule que ST_IsEmpty(NULL) doit renvoyer 0. PostGIS renvoie NULL.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.7

Cette méthode prend en charge les types Circular String et Curve.

Warning
Modifié : 2.0.0 Dans les versions précédentes de PostGIS, ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)') était autorisé. Ceci est maintenant illégal dans PostGIS 2.0.0 pour mieux se conformer aux normes SQL/MM

Exemples

```
SELECT ST_IsEmpty(ST_GeomFromText('GEOMETRYCOLLECTION EMPTY'));
 st_isempty
------------
 t
(1 row)

 SELECT ST_IsEmpty(ST_GeomFromText('POLYGON EMPTY'));
 st_isempty
------------
 t
(1 row)

SELECT ST_IsEmpty(ST_GeomFromText('POLYGON((1 2, 3 4, 5 6, 1 2))'));

 st_isempty
------------
 f
(1 row)

 SELECT ST_IsEmpty(ST_GeomFromText('POLYGON((1 2, 3 4, 5 6, 1 2))')) = false;
 ?column?
----------
 t
```

```
(1 row)

 SELECT ST_IsEmpty(ST_GeomFromText('CIRCULARSTRING EMPTY'));
  st_isempty
------------
 t
(1 row)
```

7.4.22 ST_IsPolygonCCW

ST_IsPolygonCCW — Teste si les polygones ont des anneaux extérieurs orientés dans le sens inverse des aiguilles d'une montre et des anneaux intérieurs orientés dans le sens des aiguilles d'une montre.

Synopsis

boolean **ST_IsPolygonCCW** (geometry geom);

Description

Renvoie un résultat positif si tous les composants polygonaux de la géométrie en entrée utilisent une orientation anti-horaire pour leur anneau extérieur et une orientation horaire pour tous les anneaux intérieurs.

Retourne vrai si la géométrie n'a pas de composants polygonaux.

Note
Les lignes fermées ne sont pas considérées comme des composants polygonaux, de sorte que vous obtiendrez toujours un retour vrai en passant une seule ligne fermée, quelle que soit son orientation.

Note
Si une géométrie polygonale n'utilise pas l'orientation inversée pour les anneaux intérieurs (c'est-à-dire si un ou plusieurs anneaux intérieurs sont orientés dans la même direction qu'un anneau extérieur), ST_IsPolygonCW et ST_IsPolygonCCW renvoient tous deux la valeur false.

Disponibilité : 2.4.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Voir aussi

ST_ForcePolygonCW , ST_ForcePolygonCCW , ST_IsPolygonCW

7.4.23 ST_IsPolygonCW

ST_IsPolygonCW — Teste si les polygones ont des anneaux extérieurs orientés dans le sens des aiguilles d'une montre et des anneaux intérieurs orientés dans le sens inverse des aiguilles d'une montre.

Synopsis

boolean **ST_IsPolygonCW** (geometry geom);

Description

Renvoie un résultat positif si tous les composants polygonaux de la géométrie d'entrée utilisent une orientation dans le sens des aiguilles d'une montre pour leur anneau extérieur, et dans le sens inverse des aiguilles d'une montre pour tous les anneaux intérieurs.

Retourne vrai si la géométrie n'a pas de composants polygonaux.

Note
Les lignes fermées ne sont pas considérées comme des composants polygonaux, de sorte que vous obtiendrez toujours un retour vrai en passant une seule ligne fermée, quelle que soit son orientation.

Note
Si une géométrie polygonale n'utilise pas l'orientation inversée pour les anneaux intérieurs (c'est-à-dire si un ou plusieurs anneaux intérieurs sont orientés dans la même direction qu'un anneau extérieur), ST_IsPolygonCW et ST_IsPolygonCCW renvoient tous deux la valeur false.

Disponibilité : 2.4.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Voir aussi

ST_ForcePolygonCW , ST_ForcePolygonCCW , ST_IsPolygonCW

7.4.24 ST_IsRing

ST_IsRing — Teste si une ligne est fermée et simple.

Synopsis

boolean **ST_IsRing**(geometry g);

Description

Renvoie `TRUE` si cette `LINESTRING` est à la fois ST_IsClosed (`ST_StartPoint(g) ~= ST_Endpoint(g)`) et ST_IsSimple (ne s'intersecte pas).

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. 2.1.5.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.1.6

Note
SQL-MM définit le résultat de `ST_IsRing(NULL)` comme étant 0, alors que PostGIS renvoie `NULL`.

Exemples

```
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;
 st_isring | st_isclosed | st_issimple
-----------+-------------+-------------
 t         | t           | t
(1 row)

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;
 st_isring | st_isclosed | st_issimple
-----------+-------------+-------------
 f         | t           | f
(1 row)
```

Voir aussi

ST_IsClosed, ST_IsSimple, ST_StartPoint, ST_EndPoint

7.4.25 ST_IsSimple

ST_IsSimple — Teste si une géométrie n'a pas de points d'auto-intersection ou d'auto-tangente.

Synopsis

boolean **ST_IsSimple**(geometry geomA);

Description

Renvoie TRUE si cette géométrie ne présente pas d'anomalie comme une auto intersection ou des segments tangentiels. Pour plus d'information sur les notions OGC de simplicité et de validité, se référer à "Ensuring OpenGIS compliance of geometries"

Note
La norme SQL-MM indique que le résultat de la fonction ST_IsSimple(NULL) doit être 0 ; PostGIS renvoie NULL.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.8

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_IsSimple(ST_GeomFromText('POLYGON((1 2, 3 4, 5 6, 1 2))'));
 st_issimple
-------------
 f
(1 row)

 SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(1 1,2 2,2 3.5,1 3,1 2,2 1)'));
```

```
 st_issimple
-------------
 f
(1 row)
```

Voir aussi

ST_IsValid

7.4.26 ST_M

ST_M — Renvoie la coordonnée M d'un point.

Synopsis

float **ST_M**(geometry a_point);

Description

Retourne les coordonnées M d'un point, ou NULL si non disponible. L'entrée doit être un point.

Note
Cette fonction ne fait pas (encore) partie de la spécification de l'OGC, mais elle est mentionnée ici pour compléter la liste des fonctions de l'extracteur de coordonnées de points.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_M(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_m
------
        4
(1 row)
```

Voir aussi

ST_GeomFromEWKT, ST_X, ST_Y, ST_Z

7.4.27 ST_MemSize

ST_MemSize — Renvoie la quantité d'espace mémoire que prend une géométrie.

Synopsis

integer **ST_MemSize**(geometry geomA);

Description

Renvoie la quantité d'espace mémoire (en octets) que prend la géométrie.

Ceci complète les fonctions intégrées de PostgreSQL fonctions d'objet de base de données pg_column_size, pg_size_pretty, pg_relation_size, pg_total_relation_size.

Note

pg_relation_size qui donne la taille en octets d'une table peut renvoyer une taille en octets inférieure à ST_MemSize. Cela est dû au fait que pg_relation_size n'ajoute pas la contribution des tables toasted et que les grandes géométries sont stockées dans les tables TOAST.
pg_total_relation_size - comprend la table, les tables toasted et les index.
pg_column_size indique l'espace que prendrait une géométrie dans une colonne en tenant compte de la compression, et peut donc être inférieur à ST_MemSize

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Modifié : 2.2.0 nom modifié en ST_MemSize pour respecter la convention de nommage.

Exemples

```
--Return how much byte space Boston takes up  in our Mass data set
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)*1.00 /
                SUM(ST_MemSize(geom))*100 As numeric(10,2)) As perbos
FROM towns;

totgeomsum        bossum        perbos
----------        ------        ------
1522 kB                30 kB        1.99


SELECT ST_MemSize(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227  ↩
    150406)'));

---
73

--What percentage of our table is taken up by just the geometry
SELECT pg_total_relation_size('public.neighborhoods') As fulltable_size, sum(ST_MemSize( ↩
    geom)) As geomsize,
sum(ST_MemSize(geom))*1.00/pg_total_relation_size('public.neighborhoods')*100 As pergeom
FROM neighborhoods;
fulltable_size geomsize  pergeom
-----------------------------------------------
262144         96238         36.71188354492187500000
```

7.4.28 ST_NDims

ST_NDims — Renvoie la dimension des coordonnées d'une géométrie.

Synopsis

integer **ST_NDims**(geometry g1);

Description

Renvoie la dimension des coordonnées de la géométrie. PostGIS supporte 2 - (x,y) , 3 - (x,y,z) ou 2D avec mesure - x,y,m, et 4 - 3D avec espace de mesure x,y,z,m

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_NDims(ST_GeomFromText('POINT(1 1)')) As d2point,
        ST_NDims(ST_GeomFromEWKT('POINT(1 1 2)')) As d3point,
        ST_NDims(ST_GeomFromEWKT('POINTM(1 1 0.5)')) As d2pointm;

         d2point | d3point | d2pointm
---------+---------+----------
           2 |       3 |        3
```

Voir aussi

ST_CoordDim, ST_Dimension, ST_GeomFromEWKT

7.4.29 ST_NPoints

ST_NPoints — Retourne le nombre de points (vertex) d'un objet géométrique.

Synopsis

integer **ST_NPoints**(geometry g1);

Description

Retourne le nombre de points d'un objet géométrique. Cela fonctionne pour tous les types de géométrie.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Note
Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_NPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29  ↩
    29.07)'));
--result
4

--Polygon in 3D space
SELECT ST_NPoints(ST_GeomFromEWKT('LINESTRING(77.29 29.07 1,77.42 29.26 0,77.27 29.31  ↩
    -1,77.29 29.07 3)'))
--result
4
```

Voir aussi

ST_NumPoints

7.4.30 ST_NRings

ST_NRings — Renvoie le nombre d'anneaux dans une géométrie polygonale.

Synopsis

integer **ST_NRings**(geometry geomA);

Description

Si la géométrie est un polygone ou un multi-polygone, renvoie le nombre d'anneaux. Contrairement à NumInteriorRings, il compte également les anneaux extérieurs.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_NRings(geom) As Nrings, ST_NumInteriorRings(geom) As ninterrings
                                        FROM (SELECT ST_GeomFromText('POLYGON((1 2, 3 4, 5  ↩
                                            6, 1 2))') As geom) As foo;
         nrings | ninterrings
--------+-------------
          1 |           0
(1 row)
```

Voir aussi

ST_NumInteriorRings

7.4.31 ST_NumGeometries

ST_NumGeometries — Renvoie le nombre d'éléments dans une collection de géométrie.

Synopsis

integer **ST_NumGeometries**(geometry geom);

Description

Renvoie le nombre d'éléments d'une collection de géométries (GEOMETRYCOLLECTION ou MULTI*). Pour les géométries atomiques non vides, le résultat est 1. Pour les géométries vides, le résultat est 0.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Modifié : 2.0.0 Dans les versions précédentes, cette fonction renvoyait NULL si la géométrie n'était pas de type collection/-MULTI. 2.0.0+ renvoie maintenant 1 pour les géométries simples, par exemple POLYGONE, LINESTRING, POINT.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 9.1.4

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
--Prior versions would have returned NULL for this -- in 2.0.0 this returns 1
SELECT ST_NumGeometries(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27  ↩
    29.31,77.29 29.07)'));
--result
1

--Geometry Collection Example - multis count as one geom in a collection
SELECT ST_NumGeometries(ST_GeomFromEWKT('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)))'));
--result
3
```

Voir aussi

ST_GeometryN, ST_Multi

7.4.32 ST_NumInteriorRings

ST_NumInteriorRings — Renvoie le nombre d'anneaux intérieurs (trous) d'un polygone.

Synopsis

integer **ST_NumInteriorRings**(geometry a_polygon);

Description

Renvoie le nombre d'anneaux intérieurs d'une géométrie polygonale. Retourne NULL si la géométrie n'est pas un polygone.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 8.2.5

Modifié : 2.0.0 - dans les versions antérieures, il permettait de passer un MULTIPOLYGONE, renvoyant le nombre d'anneaux intérieurs du premier POLYGONE.

Exemples

```
--If you have a regular polygon
SELECT gid, field1, field2, ST_NumInteriorRings(geom) AS numholes
FROM sometable;

--If you have multipolygons
--And you want to know the total number of interior rings in the MULTIPOLYGON
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;
```

Voir aussi

ST_NumInteriorRing, ST_InteriorRingN

7.4.33 ST_NumInteriorRing

ST_NumInteriorRing — Renvoie le nombre d'anneaux intérieurs (trous) d'un polygone. Alias pour ST_NumInteriorRings

Synopsis

integer **ST_NumInteriorRing**(geometry a_polygon);

Voir aussi

ST_NumInteriorRings, ST_InteriorRingN

7.4.34 ST_NumPatches

ST_NumPatches — Renvoie le nombre de faces d'une surface polyédrique. Retourne null pour les géométries non polyédriques.

Synopsis

integer **ST_NumPatches**(geometry g1);

Description

Renvoie le nombre de faces d'une surface polyédrique. Retourne null pour les géométries non polyédriques. C'est un alias de ST_NumGeometries pour supporter le nommage MM. Il est plus rapide d'utiliser ST_NumGeometries si vous ne vous souciez pas de la convention MM.

Disponibilité : 2.0.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM ISO/IEC 13249-3 : 8.5

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_NumPatches(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) ↩
                    ) )'));
                --result
                6
```

Voir aussi

ST_GeomFromEWKT, ST_NumGeometries

7.4.35 ST_NumPoints

ST_NumPoints — Renvoie le nombre de points dans une LineString ou une CircularString.

Synopsis

integer **ST_NumPoints**(geometry g1);

Description

Retourne le nombre de points dans une ST_LineString ou ST_CircularString. Avant la version 1.4, cette fonction ne fonctionnait qu'avec les lignes, comme l'indiquent les spécifications. A partir de la version 1.4, il s'agit d'un alias de ST_NPoints qui renvoie le nombre de sommets, et pas seulement pour les lignes. Envisagez d'utiliser ST_NPoints à la place, qui est polyvalent et fonctionne avec de nombreux types de géométrie.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.2.4

Exemples

```
SELECT ST_NumPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29  ↩
    29.07)'));
                --result
                4
```

Voir aussi

ST_NPoints

7.4.36 ST_PatchN

ST_PatchN — Renvoie la Nième géométrie (face) d'une PolyhedralSurface.

Synopsis

geometry **ST_PatchN**(geometry geomA, integer n);

Description

Renvoie la Nième géométrie (face) basée sur 1 si la géométrie est une POLYHEDRALSURFACE ou une POLYHEDRALSURFACEM. Sinon, elle renvoie NULL. Cette fonction renvoie la même réponse que ST_GeometryN pour PolyhedralSurfaces. L'utilisation de ST_GeometryN est plus rapide.

Note
L'indice est basé sur 1.

Note
Si vous voulez extraire tous les éléments d'une géométrie, ST_Dump est plus efficace.

Disponibilité : 2.0.0

Cette méthode implémente la spécification SQL/MM. SQL-MM ISO/IEC 13249-3 : 8.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
--Extract the 2nd face of the polyhedral surface
SELECT ST_AsEWKT(ST_PatchN(geom, 2)) As geomewkt
FROM (
VALUES (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 foo(geom);

              geomewkt
---+----------------------------------------
 POLYGON((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0))
```

Voir aussi

ST_AsEWKT, ST_GeomFromEWKT, ST_Dump, ST_GeometryN, ST_NumGeometries

7.4.37 ST_PointN

ST_PointN — Renvoie le Nième point de la première LineString ou LineString circulaire d'une géométrie.

Synopsis

geometry **ST_PointN**(geometry a_linestring, integer n);

Description

Renvoie le Nième point d'une ligne ou d'une ligne circulaire dans la géométrie. Les valeurs négatives sont comptées à rebours à partir de la fin de la LineString, de sorte que -1 est le dernier point. Renvoie NULL s'il n'y a pas de ligne dans la géométrie.

Note
L'index est basé sur 1 comme pour les spécifications de l'OGC depuis la version 0.8.0. L'indexation à rebours (index négatif) n'est pas prévue dans les spécifications de l'OGC. Les versions précédentes l'ont implémentée en la basant sur 0.

Note
Si vous souhaitez obtenir le Nième point de chaque ligne dans une multiligne, utilisez ST_Dump en conjonction avec ST_Dump

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.2.5, 7.3.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Note
Modifié : la version 2.0.0 ne fonctionne plus avec les multi-lignes à géométrie unique. Dans les anciennes versions de PostGIS, une multi-ligne d'une seule ligne fonctionnait parfaitement avec cette fonction et renvoyait le point de départ. Dans la version 2.0.0, elle renvoie simplement NULL comme n'importe quelle autre multi-ligne.
Modifié : 2.3.0 : indexation négative disponible (-1 est le dernier point)

Exemples

```
-- Extract all POINTs from a LINESTRING
SELECT ST_AsText(
   ST_PointN(
          column1,
          generate_series(1, ST_NPoints(column1))
   ))
FROM ( VALUES ('LINESTRING(0 0, 1 1, 2 2)'::geometry) ) AS foo;

 st_astext
------------
 POINT(0 0)
 POINT(1 1)
 POINT(2 2)
(3 rows)

--Example circular string
SELECT ST_AsText(ST_PointN(ST_GeomFromText('CIRCULARSTRING(1 2, 3 2, 1 2)'), 2));
```

```
 st_astext
------------
 POINT(3 2)
(1 row)

SELECT ST_AsText(f)
FROM ST_GeomFromText('LINESTRING(0 0 0, 1 1 1, 2 2 2)') AS g
  ,ST_PointN(g, -2) AS f; -- 1 based index

    st_astext
-----------------
 POINT Z (1 1 1)
(1 row)
```

Voir aussi

ST_NPoints

7.4.38 ST_Points

ST_Points — Renvoie un MultiPoint contenant les coordonnées d'une géométrie.

Synopsis

geometry **ST_Points**(geometry geom);

Description

Renvoie un MultiPoint contenant toutes les coordonnées d'une géométrie. Les points en double sont conservés, y compris les points de départ et d'arrivée des géométries en anneau. (Si vous le souhaitez, les points en double peuvent être supprimés en appelant ST_RemoveRepeatedPoints sur le résultat).

Pour obtenir des informations sur la position de chaque coordonnée dans la géométrie parente, utilisez ST_DumpPoints.

Les coordonnées M et Z sont conservées si elles sont présentes.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Disponibilité : 2.3.0

Exemples

```
SELECT ST_AsText(ST_Points('POLYGON Z ((30 10 4,10 30 5,40 40 6, 30 10))'));

--result
MULTIPOINT Z ((30 10 4),(10 30 5),(40 40 6),(30 10 4))
```

Voir aussi

ST_RemoveRepeatedPoints, ST_DumpPoints

7.4.39 ST_StartPoint

ST_StartPoint — Renvoie le premier point d'une LineString.

Synopsis

geometry **ST_StartPoint**(geometry geomA);

Description

Renvoie le premier point d'une géométrie `LINESTRING` ou `CIRCULARLINESTRING` comme un `POINT`. Renvoie `NULL` si l'entrée n'est pas une `LINESTRING` ou `CIRCULARLINESTRING`.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.1.3

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Note

Amélioré : 3.2.0 renvoie un point pour toutes les géométries. Le comportement précédent renvoyait NULL si l'entrée n'était pas une LineString.
Modifié : 2.0.0 ne fonctionne plus avec les MultiLineStrings à géométrie unique. Dans les anciennes versions de PostGIS, une MultiLineString à géométrie unique fonctionnait sans problème avec cette fonction et renvoyait le point de départ. Dans la version 2.0.0, elle renvoie simplement NULL comme toute autre MultiLineString. L'ancien comportement était une fonctionnalité non documentée, mais les personnes qui supposaient que leurs données étaient stockées en tant que LINESTRING peuvent voir ces données retourner NULL dans la version 2.0.0.

Exemples

Point de départ d'une LineString

```
SELECT ST_AsText(ST_StartPoint('LINESTRING(0 1, 0 2)'::geometry));
 st_astext
------------
 POINT(0 1)
```

Le point de départ d'une géométrie qui n'est pas une LineString est NULL

```
SELECT ST_StartPoint('POINT(0 1)'::geometry) IS NULL AS is_null;
  is_null
----------
 t
```

Point de départ d'une LineString 3D

```
SELECT ST_AsEWKT(ST_StartPoint('LINESTRING(0 1 1, 0 2 2)'::geometry));
 st_asewkt
------------
 POINT(0 1 1)
```

Point de départ d'une CircularString

```
SELECT ST_AsText(ST_StartPoint('CIRCULARSTRING(5 2,-3 1.999999, -2 1, -4 2, 6 3)'::geometry ↩
    ));
 st_astext
------------
 POINT(5 2)
```

Voir aussi

ST_EndPoint, ST_PointN

7.4.40 ST_Summary

ST_Summary — Renvoie un résumé textuel du contenu d'une géométrie.

Synopsis

text **ST_Summary**(geometry g);
text **ST_Summary**(geography g);

Description

Renvoie un résumé textuel du contenu de la géométrie.

Les indicateurs indiqués entre crochets après le type de géométrie ont la signification suivante :

- M : possède une coordonnée M
- Z : possède une coordonnée Z
- B : possède une bounding box en cache
- G : est géodésique (geography)
- S : possède un système de référence spatiale

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Disponibilité : 1.2.2

Amélioré : la version 2.0.0 a ajouté la prise en charge du type geography

Amélioré : 2.1.0 Indicateur S pour indiquer si le système de référence spatiale est connu

Amélioré : 2.2.0 Ajout de la prise en charge des TIN et des courbes

Exemples

```
=# SELECT ST_Summary(ST_GeomFromText('LINESTRING(0 0, 1 1)')) as geom,
        ST_Summary(ST_GeogFromText('POLYGON((0 0, 1 1, 1 2, 1 1, 0 0))')) geog;
            geom             |          geog
-----------------------------+-------------------------
 LineString[B] with 2 points | Polygon[BGS] with 1 rings
                             | ring 0 has 5 points
                             :
(1 row)


=# SELECT ST_Summary(ST_GeogFromText('LINESTRING(0 0 1, 1 1 1)')) As geog_line,
        ST_Summary(ST_GeomFromText('SRID=4326;POLYGON((0 0 1, 1 1 2, 1 2 3, 1 1 1, 0 0 1)) ↩
            ')) As geom_poly;
```

```
;
           geog_line             |        geom_poly
---------------------------------- +--------------------------
 LineString[ZBGS] with 2 points | Polygon[ZBS] with 1 rings
                                :    ring 0 has 5 points
                                :
(1 row)
```

Voir aussi

PostGIS_DropBBox, PostGIS_AddBBox, ST_Force3DM, ST_Force3DZ, ST_Force2D, geography

ST_IsValid, ST_IsValid, ST_IsValidReason, ST_IsValidDetail

7.4.41 ST_X

ST_X — Renvoie la coordonnée X d'un point.

Synopsis

float **ST_X**(geometry a_point);

Description

Renvoie la coordonnée X du point, ou NULL si elle n'est pas disponible. L'entrée doit être un point.

Note
Pour obtenir les valeurs X minimale et maximale des coordonnées géométriques, utilisez les fonctions ST_XMin et ST_XMax.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 6.1.3

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_X(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_x
------
        1
(1 row)

SELECT ST_Y(ST_Centroid(ST_GeomFromEWKT('LINESTRING(1 2 3 4, 1 1 1 1)')));
 st_y
------
  1.5
(1 row)
```

Voir aussi

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_XMax, ST_XMin, ST_Y, ST_Z

7.4.42 ST_Y

ST_Y — Renvoie la coordonnée Y d'un point.

Synopsis

float **ST_Y**(geometry a_point);

Description

Renvoie la coordonnée Y du point, ou NULL si elle n'est pas disponible. L'entrée doit être un point.

Note
Pour obtenir les valeurs minimale et maximale en Y des coordonnées géométriques, utilisez les fonctions ST_YMin et ST_YMax.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 6.1.4

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_Y(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_y
------
        2
(1 row)

SELECT ST_Y(ST_Centroid(ST_GeomFromEWKT('LINESTRING(1 2 3 4, 1 1 1 1)')));
 st_y
------
  1.5
(1 row)
```

Voir aussi

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_X, ST_YMax, ST_YMin, ST_Z

7.4.43 ST_Z

ST_Z — Renvoie la coordonnée Z d'un point.

Synopsis

float **ST_Z**(geometry a_point);

Description

Renvoie la coordonnée Z du point, ou NULL si elle n'est pas disponible. L'entrée doit être un point.

Note
Pour obtenir les valeurs minimale et maximale de Z des coordonnées géométriques, utilisez les fonctions ST_ZMin et ST_ZMax.

Cette méthode implémente la spécification SQL/MM.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_Z(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_z
------
        3
(1 row)
```

Voir aussi

ST_GeomFromEWKT, ST_M, ST_X, ST_Y, ST_ZMax, ST_ZMin

7.4.44 ST_Zmflag

ST_Zmflag — Retourne un code indiquant la dimension des coordonnées ZM d'une géométrie.

Synopsis

smallint **ST_Zmflag**(geometry geomA);

Description

Retourne un code indiquant la dimension des coordonnées ZM d'une géométrie.

Les valeurs sont les suivantes : 0 = 2D, 1 = 3D-M, 2 = 3D-Z, 3 = 4D.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_Zmflag(ST_GeomFromEWKT('LINESTRING(1 2, 3 4)'));
 st_zmflag
-----------
                0

SELECT ST_Zmflag(ST_GeomFromEWKT('LINESTRINGM(1 2 3, 3 4 3)'));
 st_zmflag
-----------
                1

SELECT ST_Zmflag(ST_GeomFromEWKT('CIRCULARSTRING(1 2 3, 3 4 3, 5 6 3)'));
 st_zmflag
-----------
                2
SELECT ST_Zmflag(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_zmflag
-----------
                3
```

Voir aussi

ST_CoordDim, ST_NDims, ST_Dimension

7.4.45 ST_HasZ

ST_HasZ — Vérifie si une géométrie possède une dimension Z.

Synopsis

boolean **ST_HasZ**(geometry geom);

Description

Vérifie si la géométrie d'entrée a une dimension Z et renvoie une valeur booléenne. Si la géométrie a une dimension Z, le résultat est vrai ; sinon, il est faux.

Les objets géométriques dotés d'une dimension Z représentent généralement des géométries tridimensionnelles (3D), tandis que ceux qui en sont dépourvus sont des géométries bidimensionnelles (2D).

Cette fonction est utile pour déterminer si une géométrie contient des informations sur l'altitude ou la hauteur.

Disponibilité : 3.5.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Exemples

```
SELECT ST_HasZ(ST_GeomFromText('POINT(1 2 3)'));
 --result
 true
```

```
SELECT ST_HasZ(ST_GeomFromText('LINESTRING(0 0, 1 1)'));
 --result
 false
```

Voir aussi

ST_Zmflag

ST_HasM

7.4.46 ST_HasM

ST_HasM — Vérifie si une géométrie a une dimension M (mesure).

Synopsis

boolean **ST_HasM**(geometry geom);

Description

Vérifie si la géométrie d'entrée a une dimension M (mesure) et renvoie une valeur booléenne. Si la géométrie a une dimension M, il renvoie vrai ; sinon, il renvoie faux.

Les objets géométriques de dimension M représentent généralement des mesures ou des données supplémentaires associées à des entités spatiales.

Cette fonction est utile pour déterminer si une géométrie contient des informations de mesure.

Disponibilité : 3.5.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Exemples

```
SELECT ST_HasM(ST_GeomFromText('POINTM(1 2 3)'));
 --result
 true
```

```
SELECT ST_HasM(ST_GeomFromText('LINESTRING(0 0, 1 1)'));
 --result
 false
```

Voir aussi

ST_Zmflag

ST_HasZ

7.5 Éditeurs de géométrie

7.5.1 ST_AddPoint

ST_AddPoint — Ajoute un point à une LineString.

Synopsis

geometry **ST_AddPoint**(geometry linestring, geometry point);

geometry **ST_AddPoint**(geometry linestring, geometry point, integer position = -1);

Description

Ajoute un point à une LineString avant l'index *position* (en utilisant un index basé sur 0). Si le paramètre *position* est omis ou vaut -1, le point est ajouté à la fin de la LineString.

Disponibilité : 1.1.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Ajouter un point à la fin d'une ligne 3D

```
SELECT ST_AsEWKT(ST_AddPoint('LINESTRING(0 0 1, 1 1 1)', ST_MakePoint(1, 2, 3)));

    st_asewkt
    ----------
    LINESTRING(0 0 1,1 1 1,1 2 3)
```

Garantir que toutes les lignes d'une table sont fermées en ajoutant le point de départ de chaque ligne à la fin de la ligne uniquement pour celles qui ne sont pas fermées.

```
UPDATE sometable
SET geom = ST_AddPoint(geom, ST_StartPoint(geom))
FROM sometable
WHERE ST_IsClosed(geom) = false;
```

Voir aussi

ST_RemovePoint, ST_SetPoint

7.5.2 ST_CollectionExtract

ST_CollectionExtract — Pour une collection de géométries spécifiée, renvoie une multi-géométrie contenant uniquement des éléments d'un type spécifié.

Synopsis

geometry **ST_CollectionExtract**(geometry collection);

geometry **ST_CollectionExtract**(geometry collection, integer type);

Description

Pour une collection de géométries spécifiée, renvoie une multi-géométrie homogène.

Si le *type* n'est pas spécifié, il renvoie une multi-géométrie contenant uniquement des géométries de la dimension la plus élevée. Les polygones sont donc préférés aux lignes, qui sont préférées aux points.

Si le *type* est spécifié, renvoie une multi-géométrie contenant uniquement ce type. S'il n'y a pas de sous-géométrie du bon type, une géométrie VIDE est renvoyée. Seuls les points, les lignes et les polygones sont pris en charge. Les numéros de type sont les suivants :

- 1 == POINT
- 2 == LINESTRING
- 3 == POLYGON

Pour les entrées de géométrie atomique, la géométrie est restituée inchangée si le type d'entrée correspond au type demandé. Sinon, le résultat est une géométrie VIDE du type spécifié. Si nécessaire, ces géométries peuvent être converties en géométries multiples en utilisant ST_Multi.

Warning
La validité des résultats MultiPolygon n'est pas vérifiée. Si les composants du polygone sont adjacents ou se chevauchent, le résultat ne sera pas valide. (Par exemple, cela peut se produire lorsque cette fonction est appliquée à un résultat ST_Split). Cette situation peut être vérifiée avec ST_IsValid et réparée avec ST_MakeValid.

Disponibilité : 1.5.0

Note
Avant la version 1.5.3, cette fonction renvoyait les entrées atomiques inchangées, quel que soit leur type. Dans la version 1.5.3, les géométries simples non correspondantes renvoyaient un résultat NULL. Dans la version 2.0.0, les géométries simples non correspondantes renvoient un résultat VIDE du type demandé.

Exemples

Extraire le type de dimension le plus élevé :

```
SELECT ST_AsText(ST_CollectionExtract(
        'GEOMETRYCOLLECTION( POINT(0 0), LINESTRING(1 1, 2 2) )'));
    st_astext
    ---------------
    MULTILINESTRING((1 1, 2 2))
```

Extraire des points (type 1 == POINT) :

```
SELECT ST_AsText(ST_CollectionExtract(
        'GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(POINT(0 0)))',
        1 ));
    st_astext
    ---------------
    MULTIPOINT((0 0))
```

Extraire des lignes (type 2 == LINESTRING) :

```
SELECT ST_AsText(ST_CollectionExtract(
        'GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(LINESTRING(0 0, 1 1)),LINESTRING(2 2, 3 3)) ↩
            ',
        2 ));
    st_astext
    ---------------
    MULTILINESTRING((0 0, 1 1), (2 2, 3 3))
```

Voir aussi

ST_CollectionHomogenize, ST_Multi, ST_IsValid, ST_MakeValid

7.5.3 ST_CollectionHomogenize

ST_CollectionHomogenize — Renvoie la représentation la plus simple d'une collection de géométries.

Synopsis

geometry **ST_CollectionHomogenize**(geometry collection);

Description

Pour une collection de géométries spécifiée, renvoie la représentation la plus simple du contenu.

- Les collections homogènes (uniformes) sont renvoyées sous la forme de la multi-géométrie appropriée.
- Les collections hétérogènes (mixtes) sont écrasées en une seule GeometryCollection.
- Les collections contenant un seul élément atomique sont renvoyées sous la forme de cet élément.
- Les géométries atomiques sont renvoyées telles quelles. Si nécessaire, elles peuvent être converties en une géométrie multiple en utilisant ST_Multi.

Warning
Cette fonction ne garantit pas la validité du résultat. En particulier, une collection contenant des polygones adjacents ou se chevauchant créera un MultiPolygon invalide. Cette situation peut être vérifiée avec ST_IsValid et réparée avec ST_MakeValid.

Disponibilité : 2.0.0

Exemples

Collection d'un seul élément convertie en géométrie atomique

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0))'));

        st_astext
        ------------
        POINT(0 0)
```

Collection imbriquée d'éléments simples convertie en géométrie atomique :

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(MULTIPOINT((0 0)))'));

        st_astext
        ------------
        POINT(0 0)
```

Collection convertie en une géométrie multiple :

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0),POINT(1 1))'));

        st_astext
        ---------------------
        MULTIPOINT((0 0),(1 1))
```

Collection hétérogène imbriquée écrasée en une GeometryCollection :

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0), GEOMETRYCOLLECTION ↩
    ( LINESTRING(1 1, 2 2)))'));

        st_astext
        ---------------------
        GEOMETRYCOLLECTION(POINT(0 0),LINESTRING(1 1,2 2))
```

Collection de polygones convertis en un MultiPolygon (non valide) :

```
SELECT ST_AsText(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)))'));

        st_astext
        ---------------------
        MULTIPOLYGON(((10 50,50 50,50 10,10 10,10 50)),((90 50,90 10,50 10,50 50,90 50)))
```

Voir aussi

ST_CollectionExtract, ST_Multi, ST_IsValid, ST_MakeValid

7.5.4 ST_CurveToLine

ST_CurveToLine — Convertit une géométrie contenant des courbes en une géométrie linéaire.

Synopsis

geometry **ST_CurveToLine**(geometry curveGeom, float tolerance, integer tolerance_type, integer flags);

Description

Convertit une CIRCULAR STRING en LINESTRING régulière ou CURVEPOLYGON en POLYGON ou MULTISURFACE en MULTIPOLYGON. Utile pour les sorties vers des appareils qui ne supportent pas les types de géométrie CIRCULARSTRING

Convertit une géométrie donnée en une géométrie linéaire. Chaque géométrie ou segment courbe est converti en une approximation linéaire en utilisant la `tolérance` et les options données (32 segments par quadrant et aucune option par défaut).

L'argument 'tolerance_type' détermine l'interprétation de l'argument `tolerance`. Il peut prendre les valeurs suivantes :

- 0 (par défaut) : La tolérance est le nombre maximum de segments par quadrant.
- 1 : La tolérance est l'écart maximal de la ligne par rapport à la courbe, en unités de source.
- 2 : La tolérance est l'angle maximal, en radians, entre les rayons générateurs.

L'argument 'flags' est un champ de bits. 0 par défaut. Les bits supportés sont :

- 1 : Sortie symétrique (indépendante de l'orientation).
- 2 : Conserver l'angle, évite de réduire les angles (longueur des segments) lors de la production d'une sortie symétrique. N'a aucun effet lorsque l'indicateur de symétrie est désactivé.

Disponibilité : 1.3.0

Amélioration : 2.4.0 a ajouté la prise en charge de la tolérance de l'écart maximal et de l'angle maximal, ainsi que de la sortie symétrique.

Amélioration : la version 3.0.0 a mis en place un nombre minimum de segments par arc linéarisé afin d'éviter une rupture topologique.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 7.1.7

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_AsText(ST_CurveToLine(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 ↩
    150505,220227 150406)')));

--Result --
 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)

--3d example
SELECT ST_AsEWKT(ST_CurveToLine(ST_GeomFromEWKT('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
SELECT ST_AsText(ST_CurveToLine(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227  ↩
    150505,220227 150406)'),2));
st_astext
------------------------------
 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)

-- Ensure approximated line is no further than 20 units away from
-- original curve, and make the result direction-neutral
SELECT ST_AsText(ST_CurveToLine(
 'CIRCULARSTRING(0 0,100 -100,200 0)'::geometry,
    20, -- Tolerance
    1, -- Above is max distance between curve and line
    1  -- Symmetric flag
));
st_astext
------------------------------------------------------------------------------------------
 LINESTRING(0 0,50 -86.6025403784438,150 -86.6025403784439,200 -1.1331077795296e-13,200 0)
```

Voir aussi

ST_LineToCurve

7.5.5 ST_Scroll

ST_Scroll — Modifier le point de départ d'une LineString fermée.

Synopsis

geometry **ST_Scroll**(geometry linestring, geometry point);

Description

Modifie le point de départ/fin d'une LineString fermée en fonction du sommet *point* donné.

Disponibilité : 3.2.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Exemples

Faire partir une ligne fermée de son 3ème sommet

```
SELECT ST_AsEWKT(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)'));

st_asewkt
----------
SRID=4326;LINESTRING(5 5 4 2,0 0 0 1,10 0 2 0,5 5 4 2)
```

Voir aussi

ST_Normalize

7.5.6 ST_FlipCoordinates

ST_FlipCoordinates — Renvoie une version d'une géométrie dont les axes X et Y sont inversés.

Synopsis

geometry **ST_FlipCoordinates**(geometry geom);

Description

Renvoie une version de la géométrie donnée avec les axes X et Y inversés. Utile pour fixer les géométries qui contiennent des coordonnées exprimées en latitude/longitude (Y,X).

Disponibilité : 2.0.0

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemple

```
SELECT ST_AsEWKT(ST_FlipCoordinates(GeomFromEWKT('POINT(1 2)')));
 st_asewkt
------------
POINT(2 1)
```

Voir aussi

ST_SwapOrdinates

7.5.7 ST_Force2D

ST_Force2D — Forcer les géométries à passer en "mode bidimensionnel".

Synopsis

geometry **ST_Force2D**(geometry geomA);

Description

Force les géométries à passer en "mode bidimensionnel", de sorte que toutes les représentations de sortie n'auront que les coordonnées X et Y. Ceci est utile pour forcer une sortie conforme à l'OGC (puisque l'OGC ne spécifie que des géométries à 2 dimensions).

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_2D.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsEWKT(ST_Force2D(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5 6 ↩
    2)')));
               st_asewkt
-------------------------------------
CIRCULARSTRING(1 1,2 3,4 5,6 7,5 6)

SELECT  ST_AsEWKT(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)) ↩
   '));

                                st_asewkt
---------------------------------------------
 POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))
```

Voir aussi

ST_Force3D

7.5.8 ST_Force3D

ST_Force3D — Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ.

Synopsis

geometry **ST_Force3D**(geometry geomA, float Zvalue = 0.0);

Description

Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ. Si une géométrie n'a pas de composante Z, une *Zvalue* coordonnée Z est ajoutée.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_3D.

Modifié : 3.1.0. Ajout de la prise en charge pour pouvoir passer une valeur Z non nulle.

Cette fonction prend en charge les surfaces Polyhedral.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
--Nothing happens to an already 3D geometry
               SELECT ST_AsEWKT(ST_Force3D(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 ↩
                   5 2, 6 7 2, 5 6 2)')));
                              st_asewkt
-----------------------------------------------
 CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)


SELECT  ST_AsEWKT(ST_Force3D('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))'));
```

```
                                                    st_asewkt
-------------------------------------------------------------
 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))
```

Voir aussi

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3DZ

7.5.9 ST_Force3DZ

ST_Force3DZ — Forcer les géométries en mode XYZ.

Synopsis

geometry **ST_Force3DZ**(geometry geomA, float Zvalue = 0.0);

Description

Force les géométries à passer en mode XYZ. Si une géométrie n'a pas de composante Z, une `Zvalue` coordonnée Z est ajoutée.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_3DZ.

Modifié : 3.1.0. Ajout de la prise en charge pour pouvoir passer une valeur Z non nulle.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
--Nothing happens to an already 3D geometry
SELECT ST_AsEWKT(ST_Force3DZ(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5  ↩
    6 2)')));
                                    st_asewkt
-----------------------------------------------
 CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)


SELECT  ST_AsEWKT(ST_Force3DZ('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))'));

                                                    st_asewkt
-------------------------------------------------------------
 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))
```

Voir aussi

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D

7.5.10 ST_Force3DM

ST_Force3DM — Forcer les géométries en mode XYM.

Synopsis

geometry **ST_Force3DM**(geometry geomA, float Mvalue = 0.0);

Description

Force les géométries à passer en mode XYM. Si une géométrie n'a pas de composante M, une `Mvalue` coordonnée M est ajoutée. Si la géométrie a une composante Z, la coordonnée Z est supprimée

Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_3DM.

Modifié : 3.1.0. Ajout de la prise en charge de pouvoir passer une valeur M non nulle.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
--Nothing happens to an already 3D geometry
SELECT ST_AsEWKT(ST_Force3DM(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5  ↩
    6 2)')));
                                st_asewkt
-----------------------------------------------
 CIRCULARSTRINGM(1 1 0,2 3 0,4 5 0,6 7 0,5 6 0)


SELECT  ST_AsEWKT(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)) ↩
    '));

                                                st_asewkt
------------------------------------------------------------------
 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))
```

Voir aussi

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

7.5.11 ST_Force4D

ST_Force4D — Forcer les géométries en mode XYZM.

Synopsis

geometry **ST_Force4D**(geometry geomA, float Zvalue = 0.0, float Mvalue = 0.0);

Description

Force les géométries en mode XYZM. *Zvalue* et *Mvalue* sont ajoutés pour les dimensions Z et M manquantes, respectivement.

Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_4D.

Modifié : 3.1.0. Ajout de la prise en charge de pouvoir passer des valeurs Z et M non nulles.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
--Nothing happens to an already 3D geometry
SELECT ST_AsEWKT(ST_Force4D(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5 6 ↩
    2)')));
                                                st_asewkt
----------------------------------------------------------
 CIRCULARSTRING(1 1 2 0,2 3 2 0,4 5 2 0,6 7 2 0,5 6 2 0)



SELECT  ST_AsEWKT(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))'));

                                                                                st_asewkt
-----------------------------------------------------------------------------------------
 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))
```

Voir aussi

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D

7.5.12 ST_ForceCollection

ST_ForceCollection — Convertir la géométrie en une GEOMETRYCOLLECTION.

Synopsis

geometry **ST_ForceCollection**(geometry geomA);

Description

Convertit la géométrie en une GEOMETRYCOLLECTION. Ceci est utile pour simplifier la représentation WKB.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Disponibilité : 1.2.2, avant la version 1.3.4 cette fonction plante avec les courbes. Ceci est corrigé dans la version 1.3.4+

Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Force_Collection.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT  ST_AsEWKT(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))'));

                                                                                st_asewkt
-------------------------------------------------------------------------------------------
 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)))


  SELECT ST_AsText(ST_ForceCollection('CIRCULARSTRING(220227 150406,2220227 150407,220227  ↩
      150406)'));
                                                                                st_astext
-------------------------------------------------------------------------------------------
 GEOMETRYCOLLECTION(CIRCULARSTRING(220227 150406,2220227 150407,220227 150406))
(1 row)
```

```
-- POLYHEDRAL example --
SELECT ST_AsEWKT(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)))'))

                                                                                st_asewkt
-------------------------------------------------------------------------------------------
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))
)
```

Voir aussi

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

7.5.13 ST_ForceCurve

ST_ForceCurve — Retransformation d'une géométrie dans son type de courbure, le cas échéant.

Synopsis

geometry **ST_ForceCurve**(geometry g);

Description

Transforme une géométrie en sa représentation courbe, le cas échéant : les lignes deviennent des courbes composées, les multilignes deviennent des multicourbes, les polygones deviennent des polygones courbes, les multipolygones deviennent des multisurfaces. Si la géométrie en entrée est déjà une représentation courbe, le résultat est identique à celui de l'entrée.

Disponibilité : 2.2.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_AsText(
  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
  )
);
                              st_astext
---------------------------------------------------------------------
 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))
(1 row)
```

Voir aussi

ST_LineToCurve

7.5.14 ST_ForcePolygonCCW

ST_ForcePolygonCCW — Oriente tous les anneaux extérieurs dans le sens inverse des aiguilles d'une montre et tous les anneaux intérieurs dans le sens des aiguilles d'une montre.

Synopsis

geometry **ST_ForcePolygonCCW** (geometry geom);

Description

Force les (Multi)Polygones à utiliser une orientation dans le sens inverse des aiguilles d'une montre pour leur anneau extérieur, et une orientation dans le sens des aiguilles d'une montre pour leurs anneaux intérieurs. Les géométries non polygonales sont renvoyées inchangées.

Disponibilité : 2.4.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Voir aussi

ST_ForcePolygonCW , ST_IsPolygonCCW , ST_IsPolygonCW

7.5.15 ST_ForcePolygonCW

ST_ForcePolygonCW — Oriente tous les anneaux extérieurs dans le sens des aiguilles d'une montre et tous les anneaux intérieurs dans le sens inverse des aiguilles d'une montre.

Synopsis

geometry **ST_ForcePolygonCW** (geometry geom);

Description

Force les (Multi)Polygones à utiliser une orientation dans le sens des aiguilles d'une montre pour leur anneau extérieur, et une orientation dans le sens inverse des aiguilles d'une montre pour leurs anneaux intérieurs. Les géométries non polygonales sont renvoyées inchangées.

Disponibilité : 2.4.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Voir aussi

ST_ForcePolygonCCW , ST_IsPolygonCCW , ST_IsPolygonCW

7.5.16 ST_ForceSFS

ST_ForceSFS — Forcer les géométries à utiliser uniquement les types de géométrie SFS 1.1.

Synopsis

geometry **ST_ForceSFS**(geometry geomA);
geometry **ST_ForceSFS**(geometry geomA, text version);

Description

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

7.5.17 ST_ForceRHR

ST_ForceRHR — Force l'orientation des sommets d'un polygone à suivre la règle de la main droite.

Synopsis

geometry **ST_ForceRHR**(geometry g);

Description

Force l'orientation des sommets d'un polygone à suivre la règle de la main droite, dans laquelle la zone délimitée par le polygone se trouve à droite de la limite. En particulier, l'anneau extérieur est orienté dans le sens des aiguilles d'une montre et les anneaux intérieurs dans le sens inverse. Cette fonction est synonyme de ST_ForcePolygonCW

Note
La définition ci-dessus de la règle de la main droite est en conflit avec des définitions utilisées dans d'autres contextes. Pour éviter toute confusion, il est recommandé d'utiliser ST_ForcePolygonCW.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_AsEWKT(
  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))'
  )
);
                                                    st_asewkt
-------------------------------------------------------------
 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))
(1 row)
```

Voir aussi

ST_ForcePolygonCCW , ST_ForcePolygonCW , ST_IsPolygonCCW , ST_IsPolygonCW , ST_BuildArea, ST_Polygonize, ST_Reverse

7.5.18 ST_LineExtend

ST_LineExtend — Renvoie une ligne prolongée vers l'avant et vers l'arrière selon les distances spécifiées.

Synopsis

geometry **ST_LineExtend**(geometry line, float distance_forward, float distance_backward=0.0);

Description

Renvoie une ligne prolongée vers l'avant et vers l'arrière en ajoutant de nouveaux points de départ (et d'arrivée) à la (aux) distance(s) donnée(s). Une distance de zéro n'ajoute pas de point. Seules les distances non négatives sont autorisées. La direction du ou des points ajoutés est déterminée par les deux premiers (et derniers) points distincts de la ligne. Les points en double sont ignorés.

Disponibilité : 3.4.0

Exemple : Prolonge une ligne de 5 unités vers l'avant et de 6 unités vers l'arrière

```
SELECT ST_AsText(ST_LineExtend('LINESTRING(0 0, 0 10)'::geometry, 5, 6));
------------------------------------------
LINESTRING(0 -6,0 0,0 10,0 15)
```

Voir aussi

ST_LineSubstring, ST_LocateAlong, ST_Project

7.5.19 ST_LineToCurve

ST_LineToCurve — Convertit une géométrie linéaire en une géométrie courbe.

Synopsis

geometry **ST_LineToCurve**(geometry geomANoncircular);

Description

Convertit les lignes et polygones simples en lignes circulaires et polygones courbes. Notez qu'il faut beaucoup moins de points pour décrire l'équivalent courbe.

Note
Si l'entrée LINESTRING/POLYGON n'est pas suffisamment courbée pour représenter clairement une courbe, la fonction renverra la même géométrie d'entrée.

Disponibilité : 1.3.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
-- 2D Example
SELECT ST_AsText(ST_LineToCurve(foo.geom)) As curvedastext,ST_AsText(foo.geom) As  ↩
    non_curvedastext
    FROM (SELECT ST_Buffer('POINT(1 3)'::geometry, 3) As geom) As foo;

curvedatext                                                              non_curvedastext
-------------------------------------------------------------------------|-------------------------------

CURVEPOLYGON(CIRCULARSTRING(4 3,3.12132034355964 0.878679656440359, | POLYGON((4  ↩
    3,3.94235584120969 2.41472903395162,3.77163859753386 1.85194970290473,
1 0,-1.12132034355965 5.12132034355963,4 3))                         |  3.49440883690764  ↩
    1.33328930094119,3.12132034355964 0.878679656440359,
                                                                         |  2.66671069905881  ↩
                                                                             0.505591163092366,2.14805029
                                                                              0.228361402466141,
```

```
                                                                    |  1.58527096604839  ↩
                                                                       0.0576441587903094,1 ↩
                                                                        0,
                                                                    |  0.414729033951621  ↩
                                                                       0.0576441587903077,-0.1480502
                                                                        0.228361402466137,
                                                                    |  -0.666710699058802  ↩
                                                                       0.505591163092361,-1.1213203
                                                                        0.878679656440353,
                                                                    |  -1.49440883690763  ↩
                                                                       1.33328930094119,-1.7716385
                                                                        1.85194970290472
                                                                    |  --ETC--  ↩
                                                                       ,3.94235584120969  ↩
                                                                       3.58527096604839,4  ↩
                                                                       3))

--3D example
SELECT ST_AsText(ST_LineToCurve(geom)) As curved, ST_AsText(geom) AS not_curved
FROM (SELECT ST_Translate(ST_Force3D(ST_Boundary(ST_Buffer(ST_Point(1,3), 2,2))),0,0,3) AS  ↩
    geom) AS foo;

                        curved                         |               not_curved
-------------------------------------------------------+-----------------------------------------

 CIRCULARSTRING Z (3 3 3,-1 2.99999999999999 3,3 3 3) | LINESTRING Z (3 3 3,2.4142135623731 ↩
     1.58578643762691 3,1 1 3,
                                                       | -0.414213562373092 1.5857864376269  ↩
                                                          3,-1 2.99999999999999 3,
                                                       | -0.414213562373101 4.41421356237309 ↩
                                                          3,
                                                       | 0.999999999999991 5  ↩
                                                          3,2.41421356237309 4.4142135623731 ↩
                                                          3,3 3 3)
(1 row)
```

Voir aussi

ST_CurveToLine

7.5.20 ST_Multi

ST_Multi — Renvoie la géométrie sous la forme d'une géométrie MULTI*.

Synopsis

geometry **ST_Multi**(geometry geom);

Description

Renvoie la géométrie sous la forme d'une collection de géométries MULTI*. Si la géométrie est déjà une collection, elle est renvoyée inchangée.

Exemples

```
SELECT ST_AsText(ST_Multi('POLYGON ((10 30, 30 30, 30 10, 10 10, 10 30))'));
                    st_astext
    ------------------------------------------------
    MULTIPOLYGON(((10 30,30 30,30 10,10 10,10 30)))
```

Voir aussi

ST_AsText

7.5.21 ST_Normalize

ST_Normalize — Renvoie la géométrie sous sa forme canonique.

Synopsis

geometry **ST_Normalize**(geometry geom);

Description

Renvoie la géométrie sous sa forme normalisée/canonique. Peut réorganiser les sommets dans les anneaux de polygones, les anneaux dans un polygone, les éléments dans un complexe multi-géométrique.

La plupart du temps, il n'est utile qu'à des fins de test (comparaison des résultats attendus et obtenus).

Disponibilité : 2.3.0

Exemples

```
SELECT ST_AsText(ST_Normalize(ST_GeomFromText(
  '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)
    )
  )'
)));
                                                                    st_astext
-----------------------------------------------------------------------------------------------------------

 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))
(1 row)
```

Voir aussi

ST_Equals,

7.5.22 ST_Project

ST_Project — Renvoie un point projeté à partir d'un point de départ en fonction d'une distance et d'un 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);

Description

Renvoie un point projeté à partir d'un point le long d'une géodésique en utilisant une distance et un azimut donnés. C'est ce qu'on appelle le problème de la géodésique directe.

La version à deux points utilise la trajectoire entre le premier et le deuxième point pour définir implicitement l'azimut et utilise la distance comme précédemment.

La distance est indiquée en mètres. Les valeurs négatives sont prises en charge.

L'azimut (également appelé cap ou relèvement) est indiqué en radians. Il est mesuré dans le sens des aiguilles d'une montre à partir du vrai nord.

- Le nord est l'azimut zéro (0 degré)
- L'est est l'azimut $\pi/2$ (90 degrés)
- Le sud est l'azimut π (180 degrés)
- Ouest est l'azimut $3\pi/2$ (270 degrés)

Les valeurs d'azimut négatives et les valeurs supérieures à 2π (360 degrés) sont prises en charge.

Disponibilité : 2.0.0

Amélioration : 2.4.0 Autorise les distances négatives et les azimuts non normalisés.

Amélioration : 3.4.0 Autorise les arguments géométriques et la forme en deux points omettant l'azimut.

Exemple : Point projeté à 100,000 mètres et relèvement de 45 degrés

```
SELECT ST_AsText(ST_Project('POINT(0 0)'::geography, 100000, radians(45.0)));
------------------------------------------
 POINT(0.635231029125537 0.639472334729198)
```

Voir aussi

ST_Azimuth, ST_Distance, Fonction PostgreSQL radians()

7.5.23 ST_QuantizeCoordinates

ST_QuantizeCoordinates — Met à zéro les bits de poids faible des coordonnées

Synopsis

geometry **ST_QuantizeCoordinates** (geometry g , int prec_x , int prec_y , int prec_z , int prec_m);

Description

`ST_QuantizeCoordinates` détermine le nombre de bits (`N`) requis pour représenter une valeur de coordonnées avec un nombre spécifié de chiffres après la virgule, puis met à zéro tous les bits sauf les `N` les plus significatifs. La valeur de coordonnées résultante sera toujours arrondie à la valeur d'origine, mais sa compressibilité sera améliorée. Il peut en résulter une réduction significative de l'utilisation du disque, à condition que la colonne géométrique utilise un type de stockage compressible. La fonction permet de spécifier un nombre différent de chiffres après la virgule dans chaque dimension ; les dimensions non spécifiées sont supposées avoir la précision de la dimension `x`. Les chiffres négatifs sont interprétés comme des chiffres à gauche de la virgule (c'est-à-dire que `prec_x=-2` conservera les valeurs des coordonnées à la centaine la plus proche).

Les coordonnées produites par `ST_QuantizeCoordinates` sont indépendantes de la géométrie qui contient ces coordonnées et de la position relative de ces coordonnées dans la géométrie. Par conséquent, les relations topologiques existantes entre les géométries ne sont pas affectées par l'utilisation de cette fonction. La fonction peut produire une géométrie non valide lorsqu'elle est appelée avec un nombre de chiffres inférieur à la précision intrinsèque de la géométrie.

Disponibilité : 2.5.0

Informations techniques

PostGIS stocke toutes les valeurs de coordonnées sous forme d'entiers à virgule flottante en double précision, qui peuvent représenter de manière fiable 15 chiffres significatifs. Cependant, PostGIS peut être utilisé pour gérer des données qui ont intrinsèquement moins de 15 chiffres significatifs. C'est le cas des données TIGER, qui sont fournies sous forme de coordonnées géographiques avec six chiffres de précision après la virgule (ce qui ne nécessite que neuf chiffres significatifs pour la longitude et huit chiffres significatifs pour la latitude)

Lorsque 15 chiffres significatifs sont disponibles, il existe de nombreuses représentations possibles d'un nombre à 9 chiffres significatifs. Un nombre à virgule flottante en double précision utilise 52 bits explicites pour représenter le significand (mantissa) de la coordonnée. Seuls 30 bits sont nécessaires pour représenter un mantissa de 9 chiffres significatifs, ce qui laisse 22 bits non significatifs ; nous pouvons leur donner la valeur que nous voulons et obtenir un nombre qui s'arrondit à notre valeur d'entrée. Par exemple, la valeur 100,123456 peut être représentée par les nombres à virgule flottante les plus proches de 100,123456000000, 100,123456000001 et 100,123456432199. Tous sont également valables, en ce sens que `ST_AsText(geom, 6)` renverra le même résultat avec n'importe laquelle de ces entrées. Comme nous pouvons fixer ces bits à n'importe quelle valeur, `ST_QuantizeCoo` fixe les 22 bits non significatifs à zéro. Pour une longue séquence de coordonnées, cela crée un motif de blocs de zéros consécutifs qui est compressé par PostgreSQL de manière plus efficace.

Note
Seule la taille sur disque de la géométrie est potentiellement affectée par `ST_QuantizeCoordinates`. ST_MemSize, qui indique l'utilisation en mémoire de la géométrie, renverra la même valeur quel que soit l'espace disque utilisé par une géométrie.

Exemples

```
SELECT ST_AsText(ST_QuantizeCoordinates('POINT (100.123456 0)'::geometry, 4));
st_astext
-------------------------
POINT(100.123455047607 0)
```

```
WITH test AS (SELECT 'POINT (123.456789123456 123.456789123456)'::geometry AS geom)
SELECT
  digits,
  encode(ST_QuantizeCoordinates(geom, digits), 'hex'),
  ST_AsText(ST_QuantizeCoordinates(geom, digits))
FROM test, generate_series(15, -15, -1) AS digits;

digits  |                  encode                  |               st_astext
--------+------------------------------------------+---------------------------------------- ↩
```

```
15      | 01010000005f9a72083cdd5e405f9a72083cdd5e40 | POINT(123.456789123456  ↩
    123.456789123456)
14      | 01010000005f9a72083cdd5e405f9a72083cdd5e40 | POINT(123.456789123456  ↩
    123.456789123456)
13      | 01010000005f9a72083cdd5e405f9a72083cdd5e40 | POINT(123.456789123456  ↩
    123.456789123456)
12      | 01010000005c9a72083cdd5e405c9a72083cdd5e40 | POINT(123.456789123456  ↩
    123.456789123456)
11      | 0101000000409a72083cdd5e40409a72083cdd5e40 | POINT(123.456789123456  ↩
    123.456789123456)
10      | 0101000000009a72083cdd5e40009a72083cdd5e40 | POINT(123.456789123455  ↩
    123.456789123455)
9       | 0101000000009072083cdd5e40009072083cdd5e40 | POINT(123.456789123418  ↩
    123.456789123418)
8       | 0101000000008072083cdd5e40008072083cdd5e40 | POINT(123.45678912336  ↩
    123.45678912336)
7       | 0101000000000070083cdd5e40000070083cdd5e40 | POINT(123.456789121032  ↩
    123.456789121032)
6       | 0101000000000040083cdd5e40000040083cdd5e40 | POINT(123.456789076328  ↩
    123.456789076328)
5       | 0101000000000000083cdd5e40000000083cdd5e40 | POINT(123.456789016724  ↩
    123.456789016724)
4       | 0101000000000000003cdd5e40000000003cdd5e40 | POINT(123.456787109375  ↩
    123.456787109375)
3       | 0101000000000000003cdd5e40000000003cdd5e40 | POINT(123.456787109375  ↩
    123.456787109375)
2       | 0101000000000000038dd5e400000000038dd5e40 | POINT(123.45654296875  ↩
    123.45654296875)
1       | 0101000000000000000dd5e400000000000dd5e40 | POINT(123.453125 123.453125)
0       | 0101000000000000000dc5e400000000000dc5e40 | POINT(123.4375 123.4375)
-1      | 0101000000000000000c05e400000000000c05e40 | POINT(123 123)
-2      | 01010000000000000000005e400000000000005e40 | POINT(120 120)
-3      | 010100000000000000000058400000000000005840 | POINT(96 96)
-4      | 010100000000000000000058400000000000005840 | POINT(96 96)
-5      | 010100000000000000000058400000000000005840 | POINT(96 96)
-6      | 010100000000000000000058400000000000005840 | POINT(96 96)
-7      | 010100000000000000000058400000000000005840 | POINT(96 96)
-8      | 010100000000000000000058400000000000005840 | POINT(96 96)
-9      | 010100000000000000000058400000000000005840 | POINT(96 96)
-10     | 010100000000000000000058400000000000005840 | POINT(96 96)
-11     | 010100000000000000000058400000000000005840 | POINT(96 96)
-12     | 010100000000000000000058400000000000005840 | POINT(96 96)
-13     | 010100000000000000000058400000000000005840 | POINT(96 96)
-14     | 010100000000000000000058400000000000005840 | POINT(96 96)
-15     | 010100000000000000000058400000000000005840 | POINT(96 96)
```

Voir aussi

ST_SnapToGrid

7.5.24 ST_RemovePoint

ST_RemovePoint — Supprime un point d'une ligne.

Synopsis

geometry **ST_RemovePoint**(geometry linestring, integer offset);

Description

Supprime un point d'une LineString, en fonction de son index (basé sur 0). Utile pour transformer une ligne fermée (anneau) en une ligne ouverte.

Amélioration : 3.2.0

Disponibilité : 1.1.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Garantit qu'aucune ligne n'est fermée en supprimant le point final des lignes fermées (anneaux). Suppose que geom est de type LINESTRING

```
UPDATE sometable
        SET geom = ST_RemovePoint(geom, ST_NPoints(geom) - 1)
        FROM sometable
        WHERE ST_IsClosed(geom);
```

Voir aussi

ST_AddPoint, ST_NPoints, ST_NumPoints

7.5.25 ST_RemoveRepeatedPoints

ST_RemoveRepeatedPoints — Renvoie une version d'une géométrie dont les points en double ont été supprimés.

Synopsis

geometry **ST_RemoveRepeatedPoints**(geometry geom, float8 tolerance);

Description

Renvoie une version de la géométrie donnée dont les points consécutifs en double ont été supprimés. La fonction ne traite que les (Multi)LineStrings, les (Multi)Polygons et les MultiPoints, mais elle peut être appelée avec n'importe quel type de géométrie. Les éléments des collections de géométries sont traités individuellement. Les extrémités des lignes sont préservées.

Si le paramètre *tolérance* est fourni, les sommets situés dans la distance de tolérance les uns des autres sont considérés comme des doublons.

Amélioration : 3.2.0

Disponibilité : 2.2.0

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'MULTIPOINT ((1 1), (2 2), (3 3), (2 2))'));
-------------------------
 MULTIPOINT(1 1,2 2,3 3)
```

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'LINESTRING (0 0, 0 0, 1 1, 0 0, 1 1, 2 2)'));
--------------------------------
 LINESTRING(0 0,1 1,0 0,1 1,2 2)
```

Exemple: Les éléments de la collection sont traités individuellement.

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'GEOMETRYCOLLECTION (LINESTRING (1 1, 2 2, 2 2,  ↩
    3 3), POINT (4 4), POINT (4 4), POINT (5 5))'));
-----------------------------------------------------------------------------
 GEOMETRYCOLLECTION(LINESTRING(1 1,2 2,3 3),POINT(4 4),POINT(4 4),POINT(5 5))
```

Exemple: Suppression répété de points avec une tolérance de distance.

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'LINESTRING (0 0, 0 0, 1 1, 5 5, 1 1, 2 2)', 2)) ↩
    ;
-------------------------
 LINESTRING(0 0,5 5,2 2)
```

Voir aussi

ST_Simplify

7.5.26 ST_RemoveIrrelevantPointsForView

ST_RemoveIrrelevantPointsForView — Removes points that are irrelevant for rendering a specific rectangluar view of a geometry.

Synopsis

geometry **ST_RemoveIrrelevantPointsForView**(geometry geom, box2d bounds, boolean cartesian_hint = false);

Description

Returns a geometry without points being irrelevant for rendering the geometry within a given rectangluar view.

This function can be used to quickly preprocess geometries that should be rendered only within certain bounds.

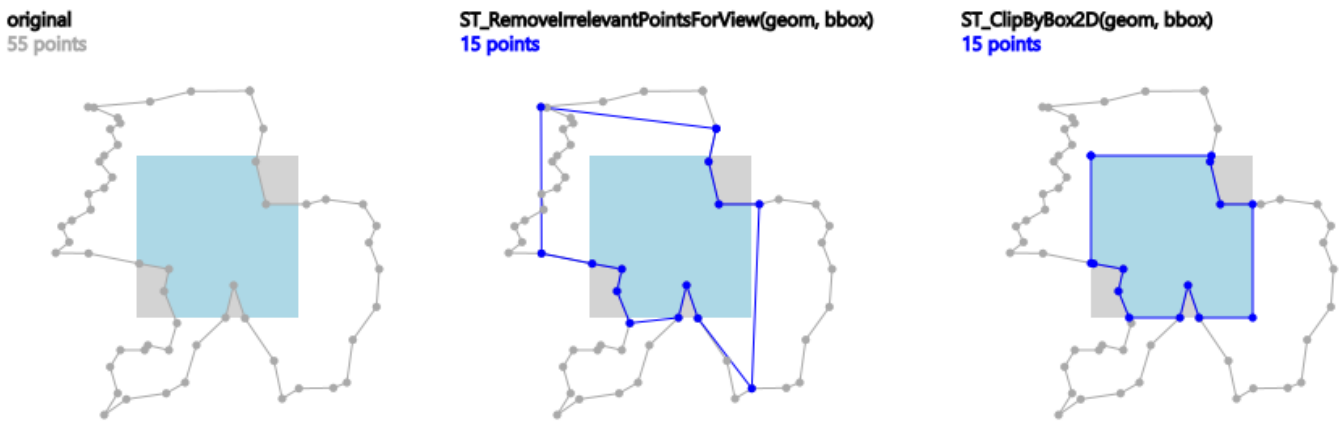
Only geometries of type (MULTI)POLYGON and (MULTI)LINESTRING are evaluated. Other geometries keep unchanged.

In contrast to `ST_ClipByBox2D()` this function

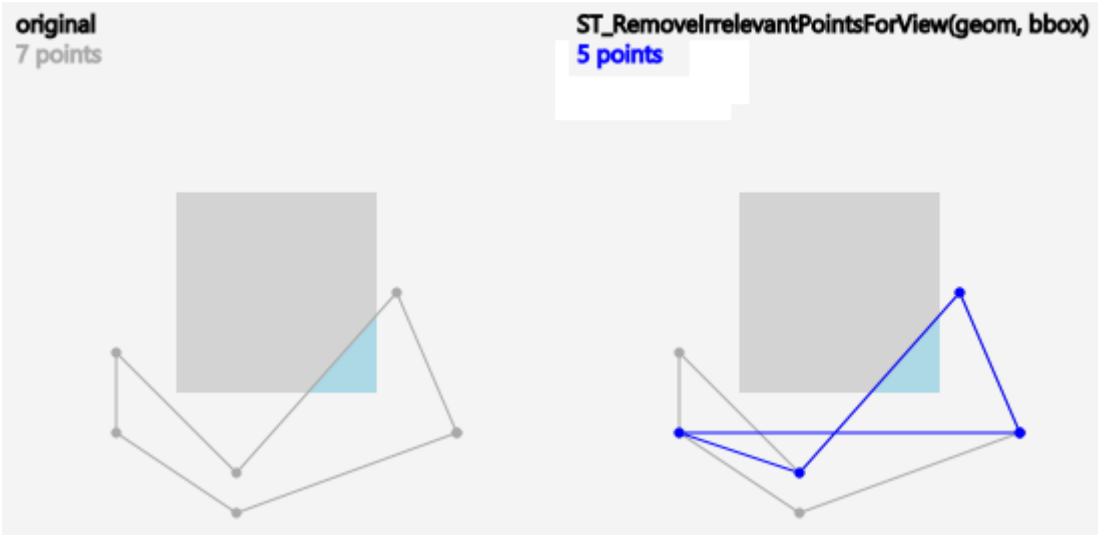
- sorts out points without computing new intersection points which avoids rounding errors and usually increases performance,
- returns a geometry with equal or similar point number,
- leads to the same rendering result within the specified view, and
- may introduce self-intersections which would make the resulting geometry invalid (see example below).

If `cartesian_hint` is set to `true`, the algorithm applies additional optimizations involving cartesian math to further reduce the resulting point number. Please note that using this option might introduce rendering artifacts if the resulting coordinates are projected into another (non-cartesian) coordinate system before rendering.

Warning
For polygons, this function does currently not ensure that the result is valid. This situation can be checked with ST_IsValid and repaired with ST_MakeValid.



Example: ST_RemoveIrrelevantPointsForView() applied to a polygon. Blue points remain, the rendering result (light-blue area) within the grey view box remains as well.



Example: Due to the fact that points are just sorted out and no new points are computed, the result of ST_RemoveIrrelevantPointsForView() may contain self-intersections.

Disponibilité : 3.5.0

Exemples

```
SELECT ST_AsText(
                        ST_RemoveIrrelevantPointsForView(
                        ST_GeomFromText('MULTIPOLYGON(((10 10, 20 10, 30 10, 40 10, 20 20,  ↩
                            10 20, 10 10)),((10 10, 20 10, 20 20, 10 20, 10 10)))'),
                        ST_MakeEnvelope(12,12,18,18), true));

                st_astext
                ---------
```

```
                    MULTIPOLYGON(((10 10,40 10,20 20,10 20,10 10)),((10 10,20 10,20 20,10  ↩
                        20,10 10)))
```

```
SELECT ST_AsText(
                        ST_RemoveIrrelevantPointsForView(
                        ST_GeomFromText('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))'),
                        ST_MakeEnvelope(12,12,18,18), true));

                st_astext
                ---------
                        MULTILINESTRING((10 15,15 15,20 15),(13 13,15 15,17 17))
```

```
SELECT ST_AsText(
                        ST_RemoveIrrelevantPointsForView(
                        ST_GeomFromText('LINESTRING(0 0, 10 0,20 0,30 0)'),
                        ST_MakeEnvelope(12,12,18,18), true));

                st_astext
                ---------
                    LINESTRING EMPTY
```

```
SELECT ST_AsText(
                        ST_RemoveIrrelevantPointsForView(
                        ST_GeomFromText('POLYGON((0 30, 15 30, 30 30, 30 0, 0 0, 0 30))'),
                        ST_MakeEnvelope(12,12,18,18), true));

                st_astext
                ---------
                    POLYGON((15 30,30 0,0 0,15 30))
```

```
SELECT ST_AsText(
                        ST_RemoveIrrelevantPointsForView(
                        ST_GeomFromText('POLYGON((0 30, 15 30, 30 30, 30 0, 0 0, 0 30))'),
                        ST_MakeEnvelope(12,12,18,18)));

                st_astext
                ---------
                    POLYGON((0 30,30 30,30 0,0 0,0 30))
```

Voir aussi

ST_ClipByBox2D, ST_Intersection

7.5.27 ST_RemoveSmallParts

ST_RemoveSmallParts — Removes small parts (polygon rings or linestrings) of a geometry.

Synopsis

geometry **ST_RemoveSmallParts**(geometry geom, double precision minSizeX, double precision minSizeY);

Description

Returns a geometry without small parts (exterior or interior polygon rings, or linestrings).

This function can be used as preprocessing step for creating simplified maps, e. g. to remove small islands or holes.

It evaluates only geometries of type (MULTI)POLYGON and (MULTI)LINESTRING. Other geometries remain unchanged.

If `minSizeX` is greater than 0, parts are sorted out if their width is smaller than `minSizeX`.

If `minSizeY` is greater than 0, parts are sorted out if their height is smaller than `minSizeY`.

Both `minSizeX` and `minSizeY` are measured in coordinate system units of the geometry.

For polygon types, evaluation is done separately for each ring which can lead to one of the following results:

- the original geometry,
- a POLYGON with all rings with less vertices,
- a POLYGON with a reduced number of interior rings (having possibly less vertices),
- a POLYGON EMPTY, or
- a MULTIPOLYGON with a reduced number of polygons (having possibly less interior rings or vertices), or
- a MULTIPOLYGON EMPTY.

For linestring types, evaluation is done for each linestring which can lead to one of the following results:

- the original geometry,
- a LINESTRING with a reduced number of vertices,
- a LINESTRING EMPTY,
- a MULTILINESTRING with a reduced number of linestrings (having possibly less vertices), or
- a MULTILINESTRING EMPTY.

Example: ST_RemoveSmallParts() applied to a multi-polygon. Blue parts remain.

Disponibilité : 3.5.0

Exemples

```
SELECT ST_AsText(
                        ST_RemoveSmallParts(
                        ST_GeomFromText('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));

                st_astext
                ---------
                        MULTIPOLYGON(((60 160,120 160,120 220,60 220,60 160)),((85 75,155  ↩
                            75,155 145,85 145,85 75)))
```

```
SELECT ST_AsText(
                        ST_RemoveSmallParts(
                        ST_GeomFromText('LINESTRING(10 10, 20 20)'),
                                50, 50));

                st_astext
                ---------
                        LINESTRING EMPTY
```

7.5.28 ST_Reverse

ST_Reverse — Retourne la géométrie avec l'ordre des sommets inversé.

Synopsis

geometry **ST_Reverse**(geometry g1);

Description

Peut être utilisé sur n'importe quelle géométrie et inverse l'ordre des sommets.

Amélioration : la prise en charge des courbes a été introduite dans la version 2.4.0.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_AsText(geom) as line, ST_AsText(ST_Reverse(geom)) As reverseline
FROM
(SELECT ST_MakeLine(ST_Point(1,2),
                ST_Point(1,10)) As geom) as foo;
--result
                line         |     reverseline
---------------------+----------------------
LINESTRING(1 2,1 10) | LINESTRING(1 10,1 2)
```

7.5.29 ST_Segmentize

ST_Segmentize — Renvoie une geometry/geography modifiée dont aucun segment ne dépasse une distance donnée.

Synopsis

geometry **ST_Segmentize**(geometry geom, float max_segment_length);
geography **ST_Segmentize**(geography geog, float max_segment_length);

Description

Renvoie une geometry/geography modifiée dont aucun segment n'est plus long que `max_segment_length`. La longueur est calculée en 2D. Les segments sont toujours divisés en sous-segments de même longueur.

- Pour la géométrie, la longueur maximale est exprimée dans les unités du système de référence spatiale.
- En géographie, la longueur maximale est exprimée en mètres. Les distances sont calculées sur la sphère. Les sommets ajoutés sont créés le long des arcs de grands cercles sphériques définis par les extrémités des segments.

Note
Cette opération ne fait que raccourcir les segments longs. Elle n'allonge pas les segments plus courts que la longueur maximale.

Warning
Pour les entrées contenant de longs segments, la spécification d'un `max_segment_length` relativement court peut entraîner l'ajout d'un très grand nombre de sommets. Cela peut se produire involontairement si l'argument est spécifié accidentellement comme un nombre de segments, plutôt que comme une longueur maximale.

Disponibilité : 1.2.2

Amélioration : 3.0.0 La segmentation de géométrie produit désormais des sous-segments de longueur égale

Amélioration : 2.3.0 La segmentation d'objets geography produit désormais des sous-segments de longueur égale

Amélioration : la prise en charge des objets de type geography a été introduite dans la version 2.1.0.

Modifié : 2.1.0 Suite à l'introduction de la prise en charge du type geography, l'utilisation `ST_Segmentize('LINESTRING(1 2, 3 4)', 0.5)` provoque une erreur de fonction ambiguë. L'entrée doit être correctement typée en tant que geometry ou geography. Utilisez ST_GeomFromText, ST_GeogFromText ou un cast vers le type requis (par exemple, `ST_Segmentize('LINESTRING(1 2, 3 4)'::geometry, 0.5) )`

Exemples

La segmentation d'une ligne. Les segments longs sont divisés de manière égale et les segments courts ne sont pas divisés.

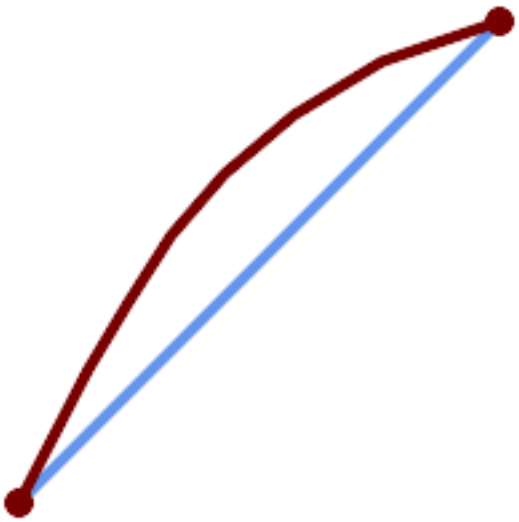
```
SELECT ST_AsText(ST_Segmentize(
    'MULTILINESTRING((0 0, 0 1, 0 9),(1 10, 1 18))'::geometry,
        5 ) );
--------------------------------------------------
MULTILINESTRING((0 0,0 1,0 5,0 9),(1 10,1 14,1 18))
```

Segmentation d'un polygone :

```
SELECT ST_AsText(
        ST_Segmentize(('POLYGON((0 0, 0 8, 30 0, 0 0))'::geometry), 10));
-------------------------------------------------------
POLYGON((0 0,0 8,7.5 6,15 4,22.5 2,30 0,20 0,10 0,0 0))
```

Segmentation d'une ligne géographique, en utilisant une longueur de segment maximale de 2000 kilomètres. Les sommets sont ajoutés le long de l'arc de grand cercle reliant les points d'extrémité.

```
SELECT ST_AsText(
        ST_Segmentize(('LINESTRING (0 0, 60 60)'::geography), 2000000));
------------------------------------------------------------
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)
```



Une ligne géographique segmentée le long d'un arc de grand cercle

Voir aussi

ST_LineSubstring

7.5.30 ST_SetPoint

ST_SetPoint — Remplacer le point d'une ligne par un point donné.

Synopsis

geometry **ST_SetPoint**(geometry linestring, integer zerobasedposition, geometry point);

Description

Remplace le point N de la ligne par le point donné. L'index est basé sur 0. Les index négatifs sont comptés à rebours, de sorte que -1 est le dernier point. Cette fonction est particulièrement utile dans les triggers lorsqu'il s'agit de maintenir la relation entre les articulations lorsqu'un sommet se déplace.

Disponibilité : 1.1.0

Mise à jour 2.3.0 : indexation négative

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
--Change first point in line string from -1 3 to -1 1
SELECT ST_AsText(ST_SetPoint('LINESTRING(-1 2,-1 3)', 0, 'POINT(-1 1)'));
           st_astext
-----------------------
 LINESTRING(-1 1,-1 3)

---Change last point in a line string (lets play with 3d linestring this time)
SELECT ST_AsEWKT(ST_SetPoint(foo.geom, ST_NumPoints(foo.geom) - 1, ST_GeomFromEWKT('POINT ↩
    (-1 1 3)')))
FROM (SELECT ST_GeomFromEWKT('LINESTRING(-1 2 3,-1 3 4, 5 6 7)') As geom) As foo;
           st_asewkt
-----------------------
LINESTRING(-1 2 3,-1 3 4,-1 1 3)

SELECT ST_AsText(ST_SetPoint(g, -3, p))
FROM ST_GEomFromText('LINESTRING(0 0, 1 1, 2 2, 3 3, 4 4)') AS g
        , ST_PointN(g,1) as p;
           st_astext
-----------------------
LINESTRING(0 0,1 1,0 0,3 3,4 4)
```

Voir aussi

ST_AddPoint, ST_NPoints, ST_NumPoints, ST_PointN, ST_RemovePoint

7.5.31 ST_ShiftLongitude

ST_ShiftLongitude — Décale les coordonnées de longitude d'une géométrie entre -180..180 et 0..360.

Synopsis

geometry **ST_ShiftLongitude**(geometry geom);

Description

Lit chaque point/vertex d'une géométrie et déplace sa coordonnée de longitude de -180..0 à 180..360 et vice versa si elle se trouve entre ces plages. Cette fonction est symétrique, de sorte que le résultat est une représentation 0..360 d'une donnée -180..180 et une représentation -180..180 d'une donnée 0..360.

Note
Ceci n'est utile que pour les données dont les coordonnées sont exprimées en longitude/latitude ; par exemple, SRID 4326 (WGS 84 géographique)

Warning
Un bug antérieur à la version 1.3.4 empêchait le fonctionnement pour MULTIPOINT. La version 1.3.4+ fonctionne également avec MULTIPOINT.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques.

NOTE : cette fonction a été renommée "ST_Shift_Longitude" dans la version 2.2.0

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
--single point forward transformation
SELECT ST_AsText(ST_ShiftLongitude('SRID=4326;POINT(270 0)'::geometry))

st_astext
----------
POINT(-90 0)


--single point reverse transformation
SELECT ST_AsText(ST_ShiftLongitude('SRID=4326;POINT(-90 0)'::geometry))

st_astext
----------
POINT(270 0)


--for linestrings the functions affects only to the sufficient coordinates
SELECT ST_AsText(ST_ShiftLongitude('SRID=4326;LINESTRING(174 12, 182 13)'::geometry))

st_astext
----------
LINESTRING(174 12,-178 13)
```

Voir aussi

ST_WrapX

7.5.32 ST_WrapX

ST_WrapX — Enveloppe une géométrie autour d'une valeur X.

Synopsis

geometry **ST_WrapX**(geometry geom, float8 wrap, float8 move);

Description

Cette fonction divise les géométries d'entrée et déplace ensuite chaque composant résultant tombant à droite (pour un "move" négatif) ou à gauche (pour un "move" positif) de la ligne "wrap" donnée dans la direction spécifiée par le paramètre "move", pour finalement réassembler les morceaux.

Note
Cette fonction est utile pour "recentrer" les données à long terme afin que les caractéristiques intéressantes ne soient pas produites d'un côté à l'autre.

Disponibilité : 2.3.0 nécessite GEOS

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
-- Move all components of the given geometries whose bounding box
-- falls completely on the left of x=0 to +360
select ST_WrapX(geom, 0, 360);

-- Move all components of the given geometries whose bounding box
-- falls completely on the left of x=-30 to +360
select ST_WrapX(geom, -30, 360);
```

Voir aussi

ST_ShiftLongitude

7.5.33 ST_SnapToGrid

ST_SnapToGrid — Accrocher tous les points de la géométrie d'entrée à une grille régulière.

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);

Description

Variante 1,2,3 : Accrocher tous les points de la géométrie d'entrée à la grille définie par l'origine et la taille des cellules. Supprime les points consécutifs tombant sur la même cellule, et renvoie éventuellement NULL si les points de sortie ne sont pas suffisants pour définir une géométrie du type donné. Les géométries écrasées d'une collection sont supprimées de celle-ci. Utile pour réduire la précision.

Variante 4 : Introduite dans la version 1.1.0 - Accroche tous les points de la géométrie d'entrée à la grille définie par son origine (le deuxième argument doit être un point) et la taille des cellules. Spécifiez 0 comme taille pour toute dimension que vous ne voulez pas accrocher à une grille.

Note
La géométrie renvoyée peut perdre sa simplicité (voir ST_IsSimple).

Note
Avant la version 1.1.0, cette fonction renvoyait toujours une géométrie 2D. A partir de la version 1.1.0, la géométrie renvoyée aura la même dimensionnalité que la géométrie d'entrée, les valeurs de dimensions supérieures n'étant pas modifiées. Utilisez la version prenant un second argument géométrique pour définir toutes les dimensions de la grille.

Disponibilité : 1.0.0RC1

Disponibilité : 1.1.0 - Prise en charge des Z et M

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
--Snap your geometries to a precision grid of 10^-3
UPDATE mytable
   SET geom = ST_SnapToGrid(geom, 0.001);

SELECT ST_AsText(ST_SnapToGrid(
                        ST_GeomFromText('LINESTRING(1.1115678 2.123, 4.111111 3.2374897,  ↩
                            4.11112 3.23748667)'),
                        0.001)
                );
                          st_astext
-------------------------------------
 LINESTRING(1.112 2.123,4.111 3.237)
 --Snap a 4d geometry
SELECT ST_AsEWKT(ST_SnapToGrid(
        ST_GeomFromEWKT('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)'),
 ST_GeomFromEWKT('POINT(1.12 2.22 3.2 4.4444)'),
 0.1, 0.1, 0.1, 0.01) );
                                                                        st_asewkt
------------------------------------------------------------------------------------
 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)


--With a 4d geometry - the ST_SnapToGrid(geom,size) only touches x and y coords but keeps m ↩
     and z the same
SELECT ST_AsEWKT(ST_SnapToGrid(ST_GeomFromEWKT('LINESTRING(-1.1115678 2.123 3 2.3456,
                4.111111 3.2374897 3.1234 1.1111)'),
           0.01)      );
                                              st_asewkt
-------------------------------------------------------
 LINESTRING(-1.11 2.12 3 2.3456,4.11 3.24 3.1234 1.1111)
```

Voir aussi

ST_Snap, ST_AsEWKT, ST_AsText, ST_GeomFromText, ST_GeomFromEWKT, ST_Simplify

7.5.34 ST_Snap

ST_Snap — Accrocher les segments et les sommets de la géométrie d'entrée aux sommets d'une géométrie de référence.

Synopsis

geometry **ST_Snap**(geometry input, geometry reference, float tolerance);

Description

Accroche les sommets et les segments d'une géométrie aux sommets d'une autre géométrie. Une tolérance de distance d'accrochage est utilisée pour contrôler l'endroit où l'accrochage est effectué. La géométrie résultante est la géométrie d'entrée avec les sommets accrochés. Si aucun accrochage n'a lieu, la géométrie d'entrée est renvoyée inchangée.

L'accrochage d'une géométrie à une autre peut améliorer la robustesse des opérations de superposition en éliminant les arêtes presque coïncidentes (qui posent des problèmes lors du noding et du calcul de l'intersection).

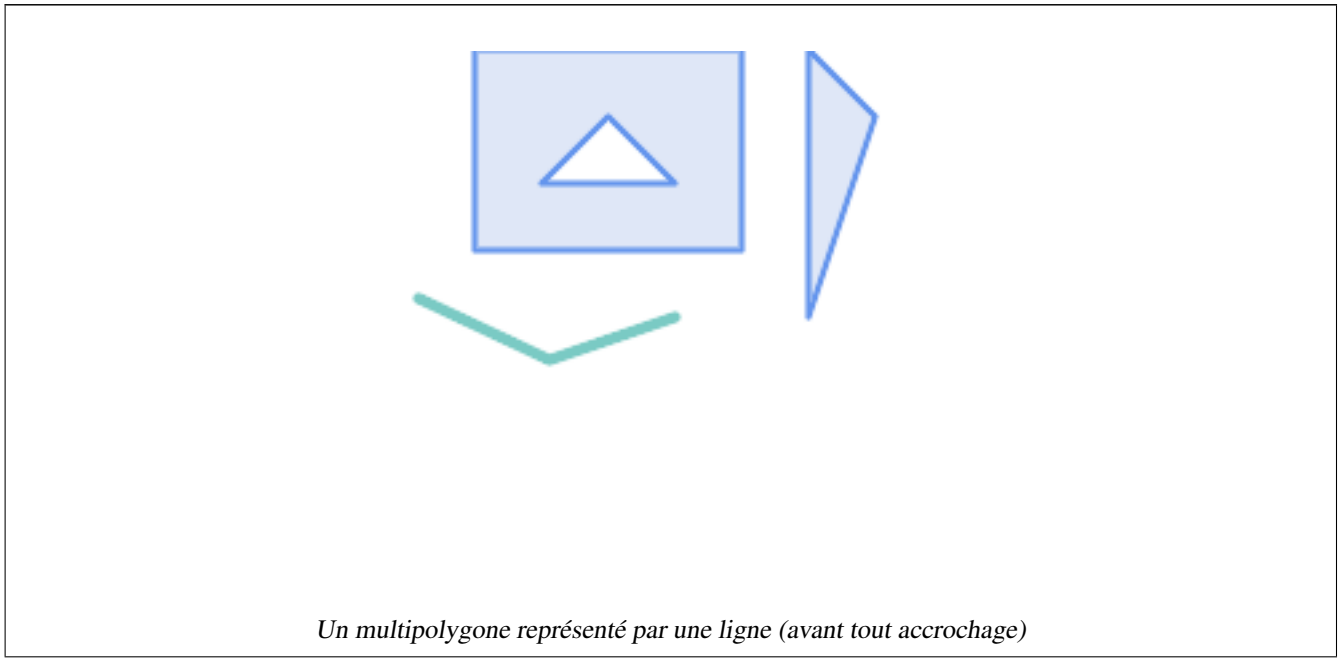
Un accrochage trop important peut entraîner la création d'une topologie non valide, c'est pourquoi le nombre et l'emplacement des sommets accrochés sont décidés à l'aide d'une heuristique pour déterminer quand il est sûr d'accrocher. Cela peut toutefois entraîner l'omission de certains accrochages potentiels.

Note
La géométrie renvoyée peut perdre sa simplicité (voir ST_IsSimple) et sa validité (voir ST_IsValid).

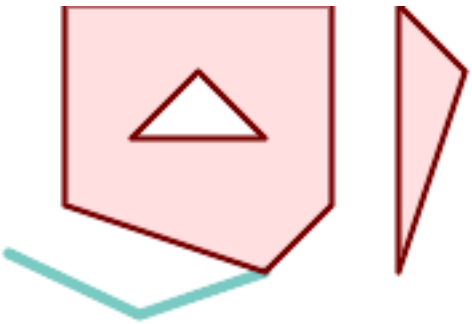
Effectué par le module GEOS.

Disponibilité : 2.0.0

Exemples

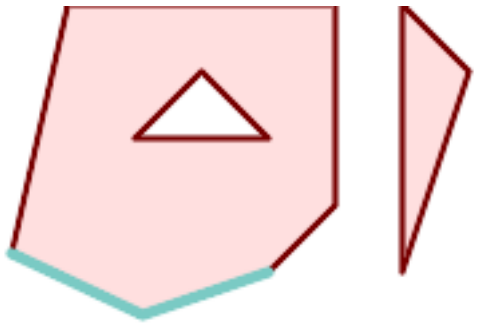


Un multipolygone représenté par une ligne (avant tout accrochage)



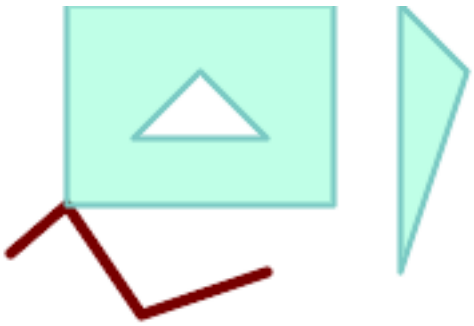
Un multipolygone est accroché à une ligne de référence avec une tolérance de 1,01 de la distance. Le nouveau multipolygone est représenté avec la ligne de référence

```
SELECT ST_AsText(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;

                              polysnapped
------------------------------------------------------------------------------
 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)))
```



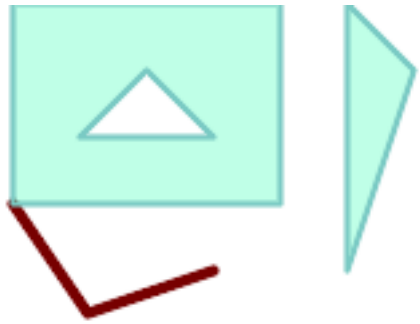
Un multipolygone est accroché à une ligne de référence avec une tolérance de 1,25 de la distance. Le nouveau multipolygone est représenté avec la ligne de référence

```
SELECT ST_AsText(
    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;

------------------------------------------------------------------------------ ↩
   polysnapped
------------------------------------------------------------------------------

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)))
```



La ligne s'est accrochée au multipolygone d'origine avec une tolérance de 1,01 de la distance. La nouvelle ligne est représentée avec le multipolygone de référence

```
SELECT ST_AsText(
   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;

              linesnapped
----------------------------------------
 LINESTRING(5 107,26 125,54 84,101 100)
```



La ligne s'est accrochée au multipolygone d'origine avec une tolérance de 1,25 de la distance. La nouvelle ligne est représentée avec le multipolygone de référence

```
SELECT ST_AsText(
 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;
              linesnapped
----------------------------------------
LINESTRING(26 125,54 84,101 100)
```

Voir aussi

ST_SnapToGrid

7.5.35 ST_SwapOrdinates

ST_SwapOrdinates — Renvoie une version de la géométrie donnée avec les valeurs d'ordonnées permutées.

Synopsis

geometry **ST_SwapOrdinates**(geometry geom, cstring ords);

Description

Renvoie une version de la géométrie donnée dont les ordonnées ont été interverties.

Le paramètre `ords` est une chaîne de 2 caractères désignant les ordonnées à permuter. Les noms valides sont : x, y, z et m.

Disponibilité : 2.2.0

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemple

```
-- Scale M value by 2
SELECT ST_AsText(
  ST_SwapOrdinates(
    ST_Scale(
      ST_SwapOrdinates(g,'xm'),
      2, 1
    ),
  'xm')
) FROM ( SELECT 'POINT ZM (0 0 0 2)'::geometry g ) foo;
     st_astext
--------------------
 POINT ZM (0 0 0 4)
```

Voir aussi

ST_FlipCoordinates

7.6 Validation de la géométrie

7.6.1 ST_IsValid

ST_IsValid — Teste si une géométrie est bien formée en 2D.

Synopsis

boolean **ST_IsValid**(geometry g);
boolean **ST_IsValid**(geometry g, integer flags);

Description

Teste si une valeur ST_Geometry est bien formée et valide en 2D selon les règles de l'OGC. Pour les géométries à 3 et 4 dimensions, la validité est toujours testée uniquement en 2 dimensions. Pour les géométries qui ne sont pas valides, une NOTICE PostgreSQL est émise fournissant les détails de la raison pour laquelle elle n'est pas valide.

Pour la version avec le paramètre `flags`, les valeurs prises en charge sont documentées dans ST_IsValidDetail Cette version n'imprime pas de NOTICE expliquant l'invalidité.

Pour plus d'informations sur la définition de la validité des géométries, reportez-vous à Section 4.4

Note
SQL-MM définit le résultat de ST_IsValid(NULL) comme étant 0, alors que PostGIS renvoie NULL.

Effectué par le module GEOS.

La version acceptant les flags est disponible à partir de la version 2.0.0.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.9

Note
Les spécifications de l'OGC-SFS et de SQL-MM ne comprennent pas d'argument de type flag pour ST_IsValid. L'indicateur est une extension de PostGIS.

Exemples

```
SELECT ST_IsValid(ST_GeomFromText('LINESTRING(0 0, 1 1)')) As good_line,
        ST_IsValid(ST_GeomFromText('POLYGON((0 0, 1 1, 1 2, 1 1, 0 0))')) As bad_poly
--results
NOTICE:  Self-intersection at or near point 0 0
 good_line | bad_poly
-----------+----------
 t         | f
```

Voir aussi

ST_IsSimple, ST_IsValidReason, ST_IsValidDetail,

7.6.2 ST_IsValidDetail

ST_IsValidDetail — Renvoie une ligne `valid_detail` indiquant si une géométrie est valide ou sinon une raison et une localisation.

Synopsis

valid_detail **ST_IsValidDetail**(geometry geom, integer flags);

Description

Renvoie une ligne `valid_detail`, contenant un booléen (`valid`) indiquant si une géométrie est valide, un varchar (`reason`) indiquant une raison pour laquelle elle est invalide et une géométrie (`location`) indiquant où elle est invalide.

Utile pour améliorer la combinaison de ST_IsValid et ST_IsValidReason afin de générer un rapport détaillé des géométries invalides.

Le paramètre facultatif `flags` est un champ de type bit. Il peut avoir les valeurs suivantes :

- 0 : utiliser la sémantique de validité habituelle de l'OGC SFS.
- 1 : Considérer certains types d'anneaux auto-touchants (coquilles inversées et trous exverts) comme valides. Ceci est également connu sous le nom de "flag ESRI", car c'est le modèle de validité utilisé par ces outils. Notez que cela n'est pas valide selon le modèle OGC.

Effectué par le module GEOS.

Disponibilité : 2.0.0

Exemples

```
--First 3 Rejects from a successful quintuplet experiment
SELECT gid, reason(ST_IsValidDetail(geom)), ST_AsText(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;

 gid  |      reason       |  location
------+-------------------+-------------
 5330 | Self-intersection | POINT(32 5)
 5340 | Self-intersection | POINT(42 5)
 5350 | Self-intersection | POINT(52 5)

 --simple example
SELECT * FROM ST_IsValidDetail('LINESTRING(220227 150406,2220227 150407,222020 150410)');

 valid | reason | location
-------+--------+----------
 t     |        |
```

Voir aussi

ST_IsValid, ST_IsValidReason

7.6.3 ST_IsValidReason

ST_IsValidReason — Renvoie un texte indiquant si une géométrie est valide, ou la raison de son invalidité.

Synopsis

text **ST_IsValidReason**(geometry geomA);
text **ST_IsValidReason**(geometry geomA, integer flags);

Description

Renvoie un texte indiquant si une géométrie est valide ou, si elle est invalide, une raison pour laquelle elle l'est.

Utile en combinaison avec ST_IsValid pour générer un rapport détaillé des géométries invalides et des raisons.

Les `flags` autorisés sont documentés dans ST_IsValidDetail.

Effectué par le module GEOS.

Disponibilité : 1.4

Disponibilité : la version 2.0 prend des flags.

Exemples

```
-- invalid bow-tie polygon
SELECT ST_IsValidReason(
    'POLYGON ((100 200, 100 100, 200 200,
     200 100, 100 200))'::geometry) as validity_info;
validity_info
--------------------------
Self-intersection[150 150]
```

```
--First 3 Rejects from a successful quintuplet experiment
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;

 gid  |      validity_info
------+-------------------------
 5330 | Self-intersection [32 5]
 5340 | Self-intersection [42 5]
 5350 | Self-intersection [52 5]

 --simple example
SELECT ST_IsValidReason('LINESTRING(220227 150406,2220227 150407,222020 150410)');

 st_isvalidreason
------------------
 Valid Geometry
```

Voir aussi

ST_IsValid, ST_Summary

7.6.4 ST_MakeValid

ST_MakeValid — Tente de rendre valide une géométrie invalide sans perdre de sommets.

Synopsis

geometry **ST_MakeValid**(geometry input);
geometry **ST_MakeValid**(geometry input, text params);

Description

La fonction tente de créer une représentation valide d'une géométrie invalide donnée sans perdre aucun des sommets d'entrée. Les géométries valides sont retournées inchangées.

Les types de données pris en charge sont : POINTS, MULTIPOINTS, LINESTRINGS, MULTILINESTRINGS, POLYGONS, MULTIPOLYGONS et GEOMETRYCOLLECTIONS contenant n'importe quel mélange de ces éléments.

En cas de diminution de dimension (total ou partiel), la géométrie de sortie peut être une collection de géométries de dimension inférieure à égale, ou une géométrie de dimension inférieure.

Les polygones simples peuvent devenir des multigéométries en cas d'auto-intersections.

L'argument `params` peut être utilisé pour fournir une chaîne d'options permettant de sélectionner la méthode à utiliser pour construire une géométrie valide. La chaîne d'options est au format "method=linework|structure keepcollapsed=true|false". Si aucun argument "params" n'est fourni, l'algorithme "linework" sera utilisé par défaut.

La clé "method" a deux valeurs.

- Le "linework" est l'algorithme original, et construit des géométries valides en extrayant d'abord toutes les lignes, en codant ce linework ensemble, puis en construisant une sortie de valeur à partir du linework.
- La "structure" est un algorithme qui distingue les anneaux intérieurs et extérieurs, construisant une nouvelle géométrie en réunissant les anneaux extérieurs, puis en différenciant tous les anneaux intérieurs.

La clé "keepcollapsed" est uniquement valable pour l'algorithme "structure" et prend la valeur "true" ou "false". Lorsqu'elle est définie sur "false", les composants géométriques qui se réduisent à une dimension inférieure, par exemple une ligne à un point, sont abandonnés.

Effectué par le module GEOS.

Disponibilité : 2.0.0

Amélioré : 2.0.1, améliorations de la vitesse

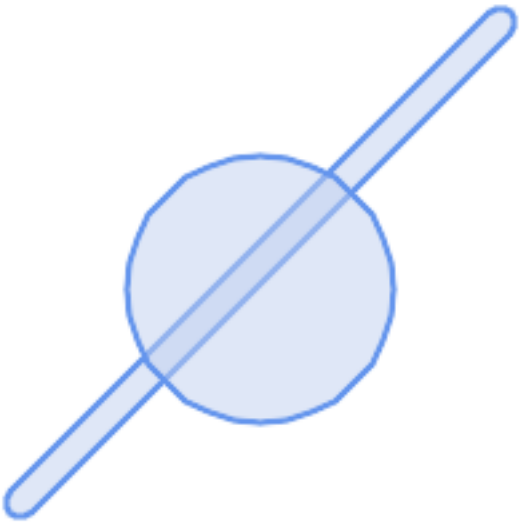
Amélioration : 2.1.0, ajout du support pour GEOMETRYCOLLECTION et MULTIPOINT.

Amélioration : 3.1.0, suppression des coordonnées avec des valeurs NaN.

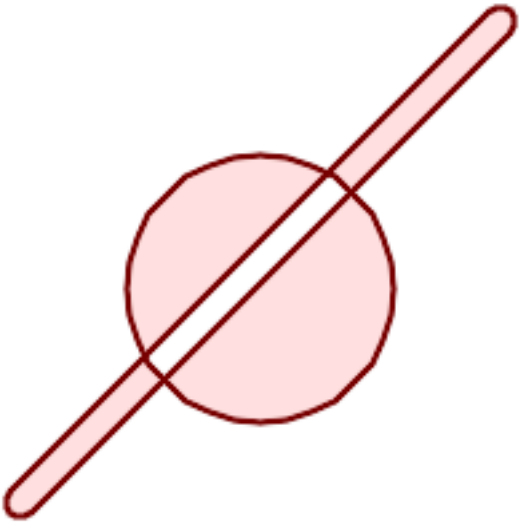
Amélioration : 3.2.0, ajout d'options d'algorithme, 'linework' et 'structure' qui nécessite GEOS >= 3.10.0.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

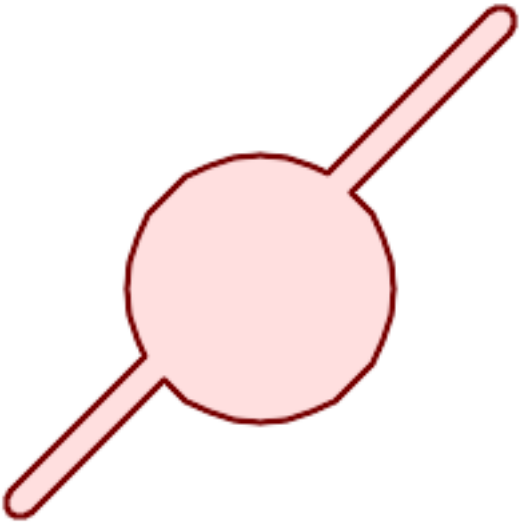
Exemples



before_geom : MULTIPOLYGON de 2 polygones qui se superposent

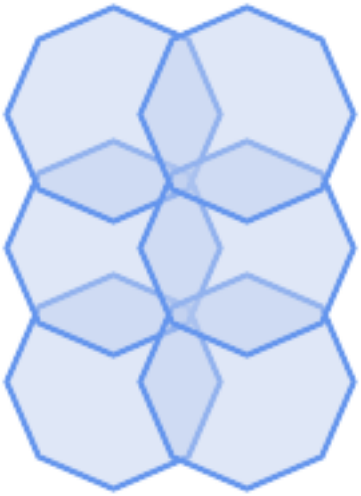


after_geom : MULTIPOLYGON de 4 polygones qui ne se superposent pas

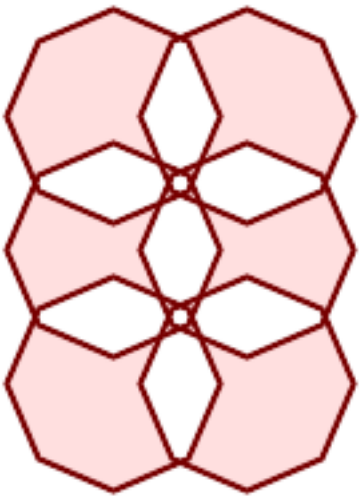


after_geom_structure : MULTIPOLYGON de 1 polygone qui ne se chevauche pas

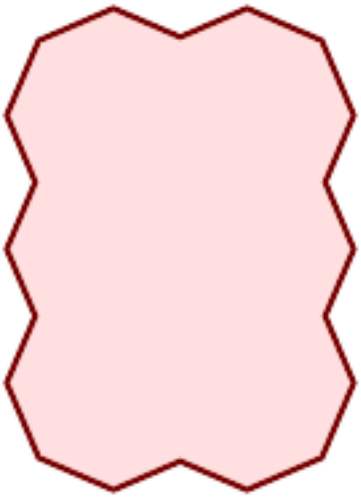
```
SELECT f.geom AS before_geom, ST_MakeValid(f.geom) AS after_geom, ST_MakeValid(f.geom,  ↩
     'method=structure') AS after_geom_structure
FROM (SELECT 'MULTIPOLYGON(((186 194,187 194,188 195,189 195,190 195,
191 195,192 195,193 194,194 194,194 194,193 195,192 195,191
```



before_geom : MULTIPOLYGON de 6 polygones qui se superposent



after_geom : MULTIPOLYGON de 14 polygones qui ne se superposent pas



after_geom_structure : MULTIPOLYGON de 1 polygone qui ne se chevauche pas

```
SELECT c.geom AS before_geom,
                    ST_MakeValid(c.geom) AS after_geom,
                    ST_MakeValid(c.geom, 'method=structure') AS after_geom_structure
       FROM (SELECT 'MULTIPOLYGON(((91 50,79 22,51 10,23 22,11 50,23 78,51 90,79 78,91 ↩
```

Exemples

```
SELECT ST_AsText(ST_MakeValid(
    'LINESTRING(0 0, 0 0)',
    'method=structure keepcollapsed=true'
    ));

 st_astext
------------
 POINT(0 0)


SELECT ST_AsText(ST_MakeValid(
    'LINESTRING(0 0, 0 0)',
    'method=structure keepcollapsed=false'
    ));

    st_astext
------------------
 LINESTRING EMPTY
```

Voir aussi

ST_IsValid, ST_Collect, ST_CollectionExtract

7.7 Fonctions des systèmes de référence spatiale

7.7.1 ST_InverseTransformPipeline

ST_InverseTransformPipeline — Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent en utilisant l'inverse d'un pipeline de transformation de coordonnées défini.

Synopsis

geometry **ST_InverseTransformPipeline**(geometry geom, text pipeline, integer to_srid);

Description

Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent à l'aide d'un pipeline de transformation de coordonnées défini pour aller dans la direction inverse.

Voir ST_TransformPipeline pour plus de détails sur l'écriture d'un pipeline de transformation.

Disponibilité : 3.4.0

Le SRID de la géométrie en entrée est ignoré et le SRID de la géométrie de sortie sera mis à zéro à moins qu'une valeur ne soit fournie via le paramètre facultatif `to_srid`. Lors de l'utilisation de ST_TransformPipeline, le pipeline est exécuté dans le sens direct. En utilisant `ST_InverseTransformPipeline()`, le pipeline est exécuté dans le sens inverse.

Les transformations utilisant des pipelines sont une version spéciale de ST_Transform. Dans la plupart des cas, `ST_Transform` choisira les opérations correctes pour convertir entre les systèmes de coordonnées, et devrait être préféré.

Exemples

Changer WGS 84 long lat en UTM 31N en utilisant la conversion EPSG:16031

```
-- Inverse direction
SELECT ST_AsText(ST_InverseTransformPipeline('POINT(426857.9877165967 5427937.523342293)':: ↩
    geometry,
  'urn:ogc:def:coordinateOperation:EPSG::16031')) AS wgs_geom;

          wgs_geom
----------------------------
 POINT(2 48.99999999999999)
(1 row)
```

Exemple GDA2020.

```
-- using ST_Transform with automatic selection of a conversion pipeline.
SELECT ST_AsText(ST_Transform('SRID=4939;POINT(143.0 -37.0)'::geometry, 7844)) AS  ↩
    gda2020_auto;

                gda2020_auto
---------------------------------------------
 POINT(143.00000635638918 -36.999986706128176)
(1 row)
```

Voir aussi

ST_Transform, ST_TransformPipeline

7.7.2 ST_SetSRID

ST_SetSRID — Définir le SRID d'une géométrie.

Synopsis

geometry **ST_SetSRID**(geometry geom, integer srid);

Description

Définit le SRID d'une géométrie à une valeur entière particulière. Utile pour construire des boîtes de délimitation pour les requêtes.

Note
Cette fonction ne transforme en aucune façon les coordonnées de la géométrie - elle définit simplement les métadonnées définissant le système de référence spatiale dans lequel la géométrie est supposée se trouver. Utilisez ST_Transform si vous souhaitez transformer la géométrie dans une nouvelle projection.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

-- Marquer un point comme WGS 84 long lat --

```
SELECT ST_SetSRID(ST_Point(-123.365556, 48.428611),4326) As wgs84long_lat;
-- the ewkt representation (wrap with ST_AsEWKT) -
SRID=4326;POINT(-123.365556 48.428611)
```

-- Marquer un point comme WGS 84 long lat puis le transformer en mercator web (Mercator Sphérique) --

```
SELECT ST_Transform(ST_SetSRID(ST_Point(-123.365556, 48.428611),4326),3785) As spere_merc;
-- the ewkt representation (wrap with ST_AsEWKT) -
SRID=3785;POINT(-13732990.8753491 6178458.96425423)
```

Voir aussi

Section 4.5, ST_SRID, ST_Transform, UpdateGeometrySRID

7.7.3 ST_SRID

ST_SRID — Renvoie l'identifiant de référence spatiale d'une géométrie.

Synopsis

integer **ST_SRID**(geometry g1);

Description

Renvoie l'identifiant de référence spatiale pour la ST_Geometry tel que défini dans la table spatial_ref_sys. Section 4.5

Note
La table spatial_ref_sys est une table qui répertorie tous les systèmes de référence spatiale connus de PostGIS et qui est utilisée pour les transformations d'un système de référence spatiale à un autre. Il est donc important de vérifier que vous disposez du bon identifiant de système de référence spatiale si vous envisagez de transformer vos géométries.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.5

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)',4326));
    --result
    4326
```

Voir aussi

Section 4.5, ST_SetSRID, ST_Transform, ST_SRID, ST_SRID

7.7.4 ST_Transform

ST_Transform — Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent.

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);

Description

Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatiale différent. La référence spatiale de destination `to_srid` peut être identifiée par un paramètre entier SRID valide (c'est-à-dire qu'elle doit exister dans la table `spatial_ref_sys`). Il est également possible d'utiliser une référence spatiale définie comme une chaîne PROJ.4 pour `to_proj` et/ou `from_proj`, mais ces méthodes ne sont pas optimisées. Si le système de référence spatiale de destination est exprimé avec une chaîne PROJ.4 au lieu d'un SRID, le SRID de la géométrie de sortie sera mis à zéro. A l'exception des fonctions avec `from_proj`, les géométries d'entrée doivent avoir un SRID défini.

ST_Transform est souvent confondu avec ST_SetSRID. ST_Transform modifie en fait les coordonnées d'une géométrie d'un système de référence spatiale à un autre, tandis que ST_SetSRID() modifie simplement l'identifiant SRID de la géométrie.

ST_Transform sélectionne automatiquement un pipeline de conversion approprié en fonction des systèmes de référence spatiale source et cible. Pour utiliser une méthode de conversion spécifique, utilisez ST_TransformPipeline.

Note
PostGIS doit être compilé avec le support PROJ. Utilisez PostGIS_Full_Version pour confirmer que vous avez compilé le support PROJ.

Note
Si l'on utilise plus d'une transformation, il est utile de disposer d'un index fonctionnel sur les transformations les plus couramment utilisées afin de tirer parti de l'utilisation de l'index.

Note
Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Amélioration : la version 2.3.0 a introduit la prise en charge du texte PROJ.4 direct.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.6

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

Changer la géométrie des pieds US du plan de l'État du Massachusetts en WGS 84 long lat

```
SELECT ST_AsText(ST_Transform(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450,
  743265 2967450,743265.625 2967416,743238 2967416))',2249),4326)) As wgs_geom;

 wgs_geom
---------------------------
 POLYGON((-71.1776848522251 42.3902896512902,-71.1776843766326 42.3903829478009,
-71.1775844305465 42.3903826677917,-71.1775825927231 42.3902893647987,-71.177684
8522251 42.3902896512902));
(1 row)

--3D Circular String example
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromEWKT('SRID=2249;CIRCULARSTRING(743238 2967416  ↩
    1,743238 2967450 2,743265 2967450 3,743265.625 2967416 3,743238 2967416 4)'),4326));

         st_asewkt
---------------------------------------------------------------------------------------
 SRID=4326;CIRCULARSTRING(-71.1776848522251 42.3902896512902 1,-71.1776843766326  ↩
     42.3903829478009 2,
 -71.1775844305465 42.3903826677917 3,
 -71.1775825927231 42.3902893647987 3,-71.1776848522251 42.3902896512902 4)
```

Exemple de création d'un index fonctionnel partiel. Pour les tables dont vous n'êtes pas sûr que toutes les géométries seront renseignées, il est préférable d'utiliser un index partiel qui laisse de côté les géométries nulles, ce qui permet à la fois de conserver de l'espace et de rendre votre index plus petit et plus efficace.

```
CREATE INDEX idx_geom_26986_parcels
  ON parcels
  USING gist
  (ST_Transform(geom, 26986))
  WHERE geom IS NOT NULL;
```

Exemples d'utilisation du texte PROJ.4 pour transformer avec des références spatiales personnalisées.

```
-- Find intersection of two polygons near the North pole, using a custom Gnomic projection
-- See http://boundlessgeo.com/2012/02/flattening-the-peel/
 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_AsText(
   ST_Transform(
     ST_Intersection(ST_Transform(p1, gnom), ST_Transform(p2, gnom)),
   gnom, 4326))
 FROM data;
                                          st_astext
 ------------------------------------------------------------------------------
  POLYGON((-170 74.053793645338,-141 73.4268621378904,-141 68,-170 68,-170 74.053793645338) ↩
      )
```

Configurer le comportement de transformation

Parfois, la transformation de coordonnées impliquant un décalage de grille peut échouer, par exemple si PROJ.4 n'a pas été construit avec des fichiers de décalage de grille ou si la coordonnée ne se trouve pas dans la plage pour laquelle le décalage de

grille est défini. Par défaut, PostGIS génère une erreur si un fichier de décalage de grille n'est pas présent, mais ce comportement peut être configuré pour chaque SRID, soit en testant différentes valeurs `to_proj` de texte PROJ.4, soit en modifiant la valeur `proj4text` dans la table `spatial_ref_sys`.

Par exemple, le paramètre proj4text +datum=NAD87 est une forme abrégée du paramètre +nadgrids suivant :

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat
```

Le préfixe @ signifie qu'aucune erreur n'est signalée si les fichiers ne sont pas présents, mais si la fin de la liste est atteinte sans qu'aucun fichier n'ait été approprié (c'est-à-dire trouvé et se chevauchant), une erreur est émise.

Si, à l'inverse, vous voulez vous assurer qu'au moins les fichiers standard sont présents, mais que si tous les fichiers ont été analysés sans résultat, une transformation nulle est appliquée, vous pouvez utiliser :

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat,null
```

Le fichier de décalage de grille nul est un fichier de décalage de grille valide couvrant le monde entier et n'appliquant aucun décalage. Ainsi, pour un exemple complet, si vous vouliez modifier PostGIS de manière à ce que les transformations du SRID 4267 qui ne se situent pas dans la plage correcte n'entraînent pas d'ERREUR, vous utiliseriez ce qui suit :

```
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;
```

Voir aussi

Section 4.5, ST_SetSRID, ST_SRID, UpdateGeometrySRID, ST_TransformPipeline

7.7.5 ST_TransformPipeline

ST_TransformPipeline — Retourner une nouvelle géométrie avec des coordonnées transformées dans un système de référence spatial différent à l'aide d'un pipeline de transformation de coordonnées défini.

Synopsis

geometry **ST_TransformPipeline**(geometry g1, text pipeline, integer to_srid);

Description

Retourner une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent à l'aide d'un pipeline de transformation de coordonnées défini.

Les pipelines de transformation sont définis à l'aide de l'un des formats de texte suivants :

- `urn:ogc:def:coordinateOperation:AUTHORITY::CODE`. Notez qu'une simple chaîne de caractère `EPSG:CODE` n'identifie pas de manière unique une opération de coordonnées : le même code EPSG peut être utilisé pour une définition CRS.
- Une chaîne de caractères de pipeline PROJ de la forme : `+proj=pipeline ...`. La normalisation automatique des axes ne sera pas appliquée et, si nécessaire, l'appelant devra ajouter une étape de pipeline supplémentaire ou supprimer les étapes `axisswap`.
- Opérations concaténées de la forme : `urn:ogc:def:coordinateOperation,coordinateOperation:EPSG::3895,c`

Disponibilité : 3.4.0

Le SRID de la géométrie d'entrée est ignoré, et le SRID de la géométrie de sortie sera mis à zéro à moins qu'une valeur ne soit fournie via le paramètre optionnel `to_srid`. Lors de l'utilisation de `ST_TransformPipeline()`, le pipeline est exécuté dans la direction avant. En utilisant ST_InverseTransformPipeline le pipeline est exécuté dans le sens inverse.

Les transformations utilisant des pipelines sont une version spéciale de ST_Transform. Dans la plupart des cas, `ST_Transform` choisira les opérations correctes pour convertir entre les systèmes de coordonnées, et devrait être préféré.

Exemples

Changer WGS 84 long lat en UTM 31N en utilisant la conversion EPSG:16031

```
-- Forward direction
SELECT ST_AsText(ST_TransformPipeline('SRID=4326;POINT(2 49)'::geometry,
  'urn:ogc:def:coordinateOperation:EPSG::16031')) AS utm_geom;

                 utm_geom
-------------------------------------------
 POINT(426857.9877165967 5427937.523342293)
(1 row)

-- Inverse direction
SELECT ST_AsText(ST_InverseTransformPipeline('POINT(426857.9877165967 5427937.523342293)':: ↩
    geometry,
  'urn:ogc:def:coordinateOperation:EPSG::16031')) AS wgs_geom;

          wgs_geom
----------------------------
 POINT(2 48.99999999999999)
(1 row)
```

Exemple GDA2020.

```
-- using ST_Transform with automatic selection of a conversion pipeline.
SELECT ST_AsText(ST_Transform('SRID=4939;POINT(143.0 -37.0)'::geometry, 7844)) AS  ↩
    gda2020_auto;

                gda2020_auto
---------------------------------------------
 POINT(143.00000635638918 -36.999986706128176)
(1 row)

-- using a defined conversion (EPSG:8447)
SELECT ST_AsText(ST_TransformPipeline('SRID=4939;POINT(143.0 -37.0)'::geometry,
  'urn:ogc:def:coordinateOperation:EPSG::8447')) AS gda2020_code;

                 gda2020_code
---------------------------------------------
 POINT(143.0000063280214 -36.999986718287545)
(1 row)

-- using a PROJ pipeline definition matching EPSG:8447, as returned from
-- 'projinfo -s EPSG:4939 -t EPSG:7844'.
-- NOTE: any 'axisswap' steps must be removed.
SELECT ST_AsText(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;

                 gda2020_pipeline
---------------------------------------------
 POINT(143.0000063280214 -36.999986718287545)
(1 row)
```

Voir aussi

ST_Transform, ST_InverseTransformPipeline

7.7.6 postgis_srs_codes

postgis_srs_codes — Renvoie la liste des codes SRS associés à l'autorité donnée.

Synopsis

setof text **postgis_srs_codes**(text auth_name);

Description

Renvoie un ensemble de tous les `auth_srid` pour le `auth_name` donné.

Disponibilité : 3.4.0

Proj version 6+

Exemples

Indiquez les dix premiers codes associés à l'autorité EPSG.

```
SELECT * FROM postgis_srs_codes('EPSG') LIMIT 10;

 postgis_srs_codes
-------------------
 2000
 20004
 20005
 20006
 20007
 20008
 20009
 2001
 20010
 20011
```

Voir aussi

postgis_srs, postgis_srs_all, postgis_srs_search

7.7.7 postgis_srs

postgis_srs — Renvoyer une fiche de métadonnées pour l'autorité et le srid demandés.

Synopsis

setof record **postgis_srs**(text auth_name, text auth_srid);

Description

Renvoie un enregistrement de métadonnées pour le `auth_srid` demandé pour le `auth_name` donné. L'enregistrement contiendra les éléments suivants : `auth_name`, `auth_srid`, `srname`, `srtext`, `proj4text`, et les angles de la zone d'utilisation, `point_sw` et `point_ne`.

Disponibilité : 3.4.0

Proj version 6+

Exemples

Obtenir les métadonnées pour EPSG:3005.

```
SELECT * FROM postgis_srs('EPSG', '3005');

auth_name | EPSG
auth_srid | 3005
srname    | NAD83 / BC Albers
srtext    | PROJCS["NAD83 / BC Albers", ... ]]
proj4text | +proj=aea +lat_0=45 +lon_0=-126 +lat_1=50 +lat_2=58.5 +x_0=1000000 +y_0=0 + ↩
    datum=NAD83 +units=m +no_defs +type=crs
point_sw  | 0101000020E6100000E17A14AE476161C00000000000204840
point_ne  | 0101000020E610000085EB51B81E855CC0E17A14AE47014E40
```

Voir aussi

postgis_srs_codes, postgis_srs_all, postgis_srs_search

7.7.8 postgis_srs_all

postgis_srs_all — Renvoie des enregistrements de métadonnées pour chaque système de référence spatiale dans la base de données Proj sous-jacente.

Synopsis

setof record **postgis_srs_all**(void);

Description

Renvoie un ensemble d'enregistrements de métadonnées dans la base de données Proj sous-jacente. Les enregistrements auront le `auth_name`, `auth_srid`, `srname`, `srtext`, `proj4text`, et les angles de la zone d'utilisation, `point_sw` et `point_ne`.

Disponibilité : 3.4.0

Proj version 6+

Exemples

Obtenir les 10 premiers enregistrements de métadonnées de la base de données Proj.

```
SELECT auth_name, auth_srid, srname FROM postgis_srs_all() LIMIT 10;

 auth_name | auth_srid |                  srname
-----------+-----------+-----------------------------------------
 EPSG      | 2000      | Anguilla 1957 / British West Indies Grid
 EPSG      | 20004     | Pulkovo 1995 / Gauss-Kruger zone 4
 EPSG      | 20005     | Pulkovo 1995 / Gauss-Kruger zone 5
 EPSG      | 20006     | Pulkovo 1995 / Gauss-Kruger zone 6
 EPSG      | 20007     | Pulkovo 1995 / Gauss-Kruger zone 7
 EPSG      | 20008     | Pulkovo 1995 / Gauss-Kruger zone 8
 EPSG      | 20009     | Pulkovo 1995 / Gauss-Kruger zone 9
 EPSG      | 2001      | Antigua 1943 / British West Indies Grid
 EPSG      | 20010     | Pulkovo 1995 / Gauss-Kruger zone 10
 EPSG      | 20011     | Pulkovo 1995 / Gauss-Kruger zone 11
```

Voir aussi

postgis_srs_codes, postgis_srs, postgis_srs_search

7.7.9 postgis_srs_search

postgis_srs_search — Renvoyer les enregistrements de métadonnées pour les systèmes de coordonnées projetées dont les zones d'utilisation contiennent entièrement le paramètre bounds.

Synopsis

setof record **postgis_srs_search**(geometry bounds, text auth_name=EPSG);

Description

Renvoie un ensemble d'enregistrements de métadonnées pour les systèmes de coordonnées projetées dont les zones d'utilisation contiennent entièrement le paramètre bounds. Chaque enregistrement contiendra le `auth_name`, `auth_srid`, `srname`, `srtext`, `proj4text`, et les angles de la zone d'utilisation, `point_sw` et `point_ne`.

La recherche ne porte que sur les systèmes de coordonnées projetées et vise à permettre aux utilisateurs d'explorer les systèmes possibles en fonction de l'étendue de leurs données.

Disponibilité : 3.4.0

Proj version 6+

Exemples

Recherche de systèmes de coordonnées projetées en Louisiane.

```
SELECT auth_name, auth_srid, srname,
  ST_AsText(point_sw) AS point_sw,
  ST_AsText(point_ne) AS point_ne
FROM postgis_srs_search('SRID=4326;LINESTRING(-90 30, -91 31)')
LIMIT 3;

 auth_name | auth_srid |                srname                |      point_sw       |       ↩
    point_ne
-----------+-----------+--------------------------------------+---------------------+---------------

 EPSG      | 2801      | NAD83(HARN) / Louisiana South        | POINT(-93.94 28.85) | POINT ↩
    (-88.75 31.07)
 EPSG      | 3452      | NAD83 / Louisiana South (ftUS)       | POINT(-93.94 28.85) | POINT ↩
    (-88.75 31.07)
 EPSG      | 3457      | NAD83(HARN) / Louisiana South (ftUS) | POINT(-93.94 28.85) | POINT ↩
    (-88.75 31.07)
```

Examinez une table pour connaître l'étendue maximale et trouvez les systèmes de coordonnées projetées qui pourraient convenir.

```
WITH ext AS (
  SELECT ST_Extent(geom) AS geom, Max(ST_SRID(geom)) AS srid
  FROM foo
)
SELECT auth_name, auth_srid, srname,
  ST_AsText(point_sw) AS point_sw,
  ST_AsText(point_ne) AS point_ne
FROM ext
CROSS JOIN postgis_srs_search(ST_SetSRID(ext.geom, ext.srid))
LIMIT 3;
```

Voir aussi

postgis_srs_codes, postgis_srs_all, postgis_srs

7.8 Import de géométrie

7.8.1 Well-Known Text (WKT)

7.8.1.1 ST_BdPolyFromText

ST_BdPolyFromText — Construit un Polygon à partir d'une collection de lignes fermées, exprimées sous forme de MultiLineString en représentation Well-Known text.

Synopsis

geometry **ST_BdPolyFromText**(text WKT, integer srid);

Description

Construit un Polygon à partir d'une collection de lignes fermées, exprimées sous forme de MultiLineString en représentation Well-Known text.

Note
Renvoie une erreur si le WKT n'est pas une MULTILINESTRING. Renvoie une erreur si le résultat est un MULTIPOLYGON. Utiliser ST_BdMPolyFromText dans ce cas, ou voir ST_BuildArea() pour une approche basée sur une fonction spécifique.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Effectué par le module GEOS.

Disponibilité : 1.1.0

Voir aussi

ST_BuildArea, ST_BdMPolyFromText

7.8.1.2 ST_BdMPolyFromText

ST_BdMPolyFromText — Construit un MultiPolygon à partir d'une collection de lignes fermées, exprimées sous forme de MultiLineString en représentation Well-Known text.

Synopsis

geometry **ST_BdMPolyFromText**(text WKT, integer srid);

Description

Construit un Polygon à partir d'une collection de lignes fermées, de polygones ou de MultiLineStrings exprimés en représentation Well-Known text.

Note
Renvoie une erreur si le WKT n'est pas une MULTILINESTRING. Force le type de retour en MULTIPOLYGON même si le résultat est en fait composé d'un seul POLYGON. Utiliser ST_BdPolyFromText si l'on est sûr que le résultat produit des Polygon, ou voir la fonction spécifique PostGIS ST_BuildArea().

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Effectué par le module GEOS.

Disponibilité : 1.1.0

Voir aussi

ST_BuildArea, ST_BdPolyFromText

7.8.1.3 ST_GeogFromText

ST_GeogFromText — Retourne un objet de type geography à partir de sa représentation Well-Know Text (WKT ou EWKT).

Synopsis

geography **ST_GeogFromText**(text EWKT);

Description

Retourne un objet de type geography à partir de sa représentation Well-Know Text (WKT ou EWKT). Le SRID 4326 est pris par défaut. Ceci est un alias pour ST_GeographyFromText. Les coordonnées des points sont exprimées en longitude latitude.

Exemples

```
--- converting lon lat coords to geography
ALTER TABLE sometable ADD COLUMN geog geography(POINT,4326);
UPDATE sometable SET geog = ST_GeogFromText('SRID=4326;POINT(' || lon || ' ' || lat || ')') ↩
    ;

--- specify a geography point using EPSG:4267, NAD27
SELECT ST_AsEWKT(ST_GeogFromText('SRID=4267;POINT(-77.0092 38.889588)'));
```

Voir aussi

ST_AsText, ST_GeographyFromText

7.8.1.4 ST_GeographyFromText

ST_GeographyFromText — Retourne un objet de type geography à partir de sa représentation Well-Know Text (WKT ou EWKT).

Synopsis

geography **ST_GeographyFromText**(text EWKT);

Description

Retourne un objet de type geography à partir de sa représentation Well-Know Text (WKT ou EWKT). SRID 4326 par défaut.

Voir aussi

ST_GeogFromText, ST_AsText

7.8.1.5 ST_GeomCollFromText

ST_GeomCollFromText — Crée une collection Geometry à partir de la collection WKT avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0.

Synopsis

geometry **ST_GeomCollFromText**(text WKT, integer srid);
geometry **ST_GeomCollFromText**(text WKT);

Description

Crée une géométrie de collection à partir de la représentation Well-Known-Text (WKT) avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0.

OGC SPEC 3.2.6.2 - l'option SRID est issue des tests de conformité

Retourne null si le WKT n'est pas une GEOMETRYCOLLECTION

Note
Si vous êtes absolument sûrs que toutes les géométries WKT sont des collections, ne pas utiliser cette fonction. Elle est plus lente que ST_GeomFromText à cause d'une étape de validation supplémentaire.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Cette méthode implémente la spécification SQL/MM.

Exemples

```
SELECT ST_GeomCollFromText('GEOMETRYCOLLECTION(POINT(1 2),LINESTRING(1 2, 3 4))');
```

Voir aussi

ST_GeomFromText, ST_SRID

7.8.1.6 ST_GeomFromEWKT

ST_GeomFromEWKT — Retourne un objet ST_Geometry à partir de sa représentation textuelle étendue (Extended Well-Known Text representation, EWKT).

Synopsis

geometry **ST_GeomFromEWKT**(text EWKT);

Description

Retourne un objet ST_Geometry à partir de sa représentation textuelle étendue OGC (Extended Well-Known Text representation, EWKT).

Note
Le format EWKT n'est pas une norme OGC, mais un format spécifique à PostGIS incluant l'identifiant du système de référence des coordonnées (SRID)

Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_GeomFromEWKT('SRID=4269;LINESTRING(-71.160281 42.258729,-71.160837  ↩
    42.259113,-71.161144 42.25932)');
SELECT ST_GeomFromEWKT('SRID=4269;MULTILINESTRING((-71.160281 42.258729,-71.160837  ↩
    42.259113,-71.161144 42.25932))');

SELECT ST_GeomFromEWKT('SRID=4269;POINT(-71.064544 42.28787)');

SELECT 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))');

SELECT 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)))');
```

```
--3d circular string
SELECT ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)');
```

```
--Polyhedral Surface example
SELECT 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))
)');
```

Voir aussi

ST_AsEWKT, ST_GeomFromText

7.8.1.7 ST_GeomFromMARC21

ST_GeomFromMARC21 — Prend les données géographiques MARC21/XML en entrée et renvoie un objet géométrique PostGIS.

Synopsis

geometry **ST_GeomFromMARC21** (text marcxml);

Description

Cette fonction crée une géométrie PostGIS à partir d'un enregistrement MARC21/XML, qui peut contenir un `POINT` ou un `POLYGON`. En cas d'entrées de données géographiques multiples dans le même enregistrement MARC21/XML, un `MULTIPOINT` ou `MULTIPOLYGON` sera renvoyé. Si la notice contient des types de géométrie mixtes, une `GEOMETRYCOLLECTION` sera retournée. Il renvoie NULL si l'enregistrement MARC21/XML ne contient pas de données géographiques (datafield :034).

Prise en charge des versions LOC MARC21/XML :

- MARC21/XML 1.1

Disponibilité : 3.3.0, nécessite libxml2 2.6+

Note
Les données mathématiques cartographiques codées MARC21/XML ne fournissent actuellement aucun moyen de décrire le système de référence spatiale des coordonnées codées, de sorte que cette fonction retournera toujours une géométrie avec `SRID 0`.

Note
Les géométries `POLYGON` retournées seront toujours orientées dans le sens des aiguilles d'une montre.

Exemples

Conversion de données géographiques MARC21/XML contenant un seul `POINT` encodé en `hddd.dddddd`

```
                SELECT
                ST_AsText(
                    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
>'));

                st_astext
                -------------------
                POINT(-4.5 54.25)
                (1 row)
```

Conversion de données géographiques MARC21/XML contenant un seul `POLYGON` encodé en `hdddmmss`

```
                SELECT
                ST_AsText(
                    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
>'));

                st_astext
```

```
                ----------------------------------------------------------------------------------------

                POLYGON((13.1 52.65,13.516666666666667 52.65,13.516666666666667  ↩
                    52.38333333333333,13.1 52.38333333333333,13.1 52.65))
                (1 row)
```

Conversion de données géographiques MARC21/XML contenant un `POLYGON` et un `POINT` :

```
                SELECT
                ST_AsText(
                    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
>'));
                                                                                        st_astext ↩

                ----------------------------------------------------------------------------------------

                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))
                (1 row)
```

Voir aussi

ST_AsMARC21

7.8.1.8 ST_GeometryFromText

ST_GeometryFromText — Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT). Alias pour ST_GeomFromText

Synopsis

geometry **ST_GeometryFromText**(text WKT);
geometry **ST_GeometryFromText**(text WKT, integer srid);

Description

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.40

Voir aussi

ST_GeomFromText

7.8.1.9 ST_GeomFromText

ST_GeomFromText — Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT).

Synopsis

geometry **ST_GeomFromText**(text WKT);
geometry **ST_GeomFromText**(text WKT, integer srid);

Description

Construit un objet Postgis de type geometry à partir d'une représentation OGC Well-Known Text WKT.

Note
Il existe deux variantes de la fonction ST_GeomFromText. La première ne prend pas de SRID et renvoie une géométrie sans système de référence spatiale défini (SRID=0). La seconde prend un SRID comme deuxième argument et renvoie une géométrie qui inclut ce SRID comme partie de ses métadonnées.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2 - l'option SRID est issue des tests de conformité.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.40

Cette méthode prend en charge les types Circular String et Curve.

Note
Bien que non conforme à l'OGC, ST_MakePoint est plus rapide que ST_GeomFromText et ST_PointFromText. Il est également plus facile à utiliser pour les valeurs de coordonnées numériques. ST_Point est une autre option similaire en vitesse à ST_MakePoint et est conforme à l'OGC, mais ne prend pas en charge autre chose que les points 2D.

Warning
Changement : 2.0.0 dans les version précédentes de PostGIS ST_GeomFromText('GEOMETRYCOLLECTION(EMPT etait autorisé. C'est désormais interdit dans PostGIS 2.0.0 pour respecter la norme SQL/MM. La forme privilégiée désormais est : ST_GeomFromText('GEOMETRYCOLLECTION EMPTY')

Exemples

```
SELECT ST_GeomFromText('LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 ↩
    42.25932)');
SELECT ST_GeomFromText('LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 ↩
    42.25932)',4269);

SELECT ST_GeomFromText('MULTILINESTRING((-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932))');

SELECT ST_GeomFromText('POINT(-71.064544 42.28787)');

SELECT ST_GeomFromText('POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 ↩
    42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))');

SELECT 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);

SELECT ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)');
```

Voir aussi

ST_GeomFromEWKT, ST_GeomFromWKB, ST_SRID

7.8.1.10 ST_LineFromText

ST_LineFromText — Construit une géométrie à partir d'une représentation WKT avec le SRID donné. Si aucun SRID n'est donné, la valeur par défaut est 0.

Synopsis

geometry **ST_LineFromText**(text WKT);
geometry **ST_LineFromText**(text WKT, integer srid);

Description

Crée une géométrie à partir de WKT avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0. Si le WKT passé n'est pas un LINESTRING, null est retourné.

Note
OGC SPEC 3.2.6.2 - option SRID issue des tests de conformité.

Note
Si vous êtes sûrs que toutes les géométries WKT sont des LINESTRINGS, la fonction ST_GeomFromText est plus efficace car elle ne contrôle pas le type de la géométrie renvoyée.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.2.8

Exemples

```
SELECT ST_LineFromText('LINESTRING(1 2, 3 4)') AS aline, ST_LineFromText('POINT(1 2)') AS  ↩
    null_return;
aline                              | null_return
-----------------------------------------------
0102000000020000000000000000F ... | t
```

Voir aussi

ST_GeomFromText

7.8.1.11 ST_MLineFromText

ST_MLineFromText — Retourne un objet de type ST_MultiLineString à partir de sa représentation WKT.

Synopsis

geometry **ST_MLineFromText**(text WKT, integer srid);
geometry **ST_MLineFromText**(text WKT);

Description

Crée une géométrie à partir du texte connu (WKT) avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0.

OGC SPEC 3.2.6.2 - l'option SRID est issue des tests de conformité

Retourne NULL si le WKT n'est pas une MULTILINESTRING

Note
Si vous êtes absolument sûrs que toutes les géométries WKT sont des points, ne pas utiliser cette fonction. Elle est plus lente que ST_GeomFromText à cause d'une étape de validation supplémentaire.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 9.4.4

Exemples

```
SELECT ST_MLineFromText('MULTILINESTRING((1 2, 3 4), (4 5, 6 7))');
```

Voir aussi

ST_GeomFromText

7.8.1.12 ST_MPointFromText

ST_MPointFromText — Créé une Geometry depuis un WKT avec le SRID donné. Si le SRID n'est pas fourni, il sera défini par défaut à 0.

Synopsis

geometry **ST_MPointFromText**(text WKT, integer srid);
geometry **ST_MPointFromText**(text WKT);

Description

Créé une Geometry depuis un WKT avec le SRID donné. Si le SRID n'est pas fourni, il sera défini par défaut à 0.

OGC SPEC 3.2.6.2 - l'option SRID est issue des tests de conformité

Retourne NULL si le WKT n'est pas une MULTIPOINT

Note
Si vous êtes absolument sûrs que toutes les géométries WKT sont des points, ne pas utiliser cette fonction. Elle est plus lente que ST_GeomFromText à cause d'une étape de validation supplémentaire.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. 3.2.6.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 9.2.4

Exemples

```
SELECT ST_MPointFromText('MULTIPOINT((1 2),(3 4))');
SELECT ST_MPointFromText('MULTIPOINT((-70.9590 42.1180),(-70.9611 42.1223))', 4326);
```

Voir aussi

ST_GeomFromText

7.8.1.13 ST_MPolyFromText

ST_MPolyFromText — Crée une géométrie multi-polygone à partir de WKT avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0.

Synopsis

geometry **ST_MPolyFromText**(text WKT, integer srid);
geometry **ST_MPolyFromText**(text WKT);

Description

Crée un MultiPolygone à partir de WKT avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0.

OGC SPEC 3.2.6.2 - l'option SRID est issue des tests de conformité

Retourne une erreur si le WKT n'est pas un MULTIPOLYGON

Note
Si vous êtes absolument sûrs que toutes les géométries WKT sont des multipolygones, ne pas utiliser cette fonction. Elle est plus lente que ST_GeomFromText à cause d'une étape de validation supplémentaire.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 9.6.4

Exemples

```
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)))');
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);
```

Voir aussi

ST_GeomFromText, ST_SRID

7.8.1.14 ST_PointFromText

ST_PointFromText — Construit une géométrie point à partir d'une représentation WKT et le SRID donné. Si aucun SRID n'est donné, la valeur par défaut est 0.

Synopsis

geometry **ST_PointFromText**(text WKT);
geometry **ST_PointFromText**(text WKT, integer srid);

Description

Construit un objet point ST_Geometry de PostGIS à partir de la représentation textuelle Well-Known de l'OGC. Si le SRID n'est pas donné, il prend par défaut la valeur inconnue (actuellement 0). Si la géométrie n'est pas une représentation de point WKT, retourne null. Si la représentation WKT n'est pas du tout valide, une erreur est générée.

Note
Il existe 2 versions de la fonction ST_PointFromText : la première ne prend pas de SRID en paramètre et retourne une geometry sans système de coordonnées. La seconde prend un SRID en second paramètre et retourne une ST_Geometry incluant un SRID dans ses métadonnées. Ce SRID doit obligatoirement exister dans la table spatial_ref_sys.

Note
Si vous êtes absolument sûrs que toutes les géométries WKT sont des points, ne pas utiliser cette fonction. Elle est plus lente que ST_GeomFromText à cause d'une étape de validation supplémentaire. Si le point doit être construit à partir de coordonnées latitude longitude et que la performance est recherchée, utiliser la fonction ST_MakePoint ou son équivalent OGC ST_Point.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2 - l'option SRID est issue des tests de conformité.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 6.1.8

Exemples

```
SELECT ST_PointFromText('POINT(-71.064544 42.28787)');
SELECT ST_PointFromText('POINT(-71.064544 42.28787)', 4326);
```

Voir aussi

ST_GeomFromText, ST_MakePoint, ST_Point, ST_SRID

7.8.1.15 ST_PolygonFromText

ST_PolygonFromText — Créé une Geometry depuis un WKT avec le SRID donné. Si le SRID n'est pas fourni, il sera défini par défaut à 0.

Synopsis

geometry **ST_PolygonFromText**(text WKT);
geometry **ST_PolygonFromText**(text WKT, integer srid);

Description

Crée une géométrie à partir de WKT avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0. Retourne null si WKT n'est pas un polygone.

OGC SPEC 3.2.6.2 - l'option SRID est issue des tests de conformité

Note
Si vous êtes absolument sûrs que toutes les géométries WKT sont des polygones, ne pas utiliser cette fonction. Elle est plus lente que ST_GeomFromText à cause d'une étape de validation supplémentaire.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 8.3.6

Exemples

```
SELECT ST_PolygonFromText('POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 ↩
    42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))');
st_polygonfromtext
------------------
0103000000010000000050000006...


SELECT ST_PolygonFromText('POINT(1 2)') IS NULL as point_is_notpoly;

point_is_not_poly
----------
t
```

Voir aussi

ST_GeomFromText

7.8.1.16 ST_WKTToSQL

ST_WKTToSQL — Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT). Alias pour ST_GeomFromText

Synopsis

geometry **ST_WKTToSQL**(text WKT);

Description

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.34

Voir aussi

ST_GeomFromText

7.8.2 Well-Known Binary (WKB)

7.8.2.1 ST_GeogFromWKB

ST_GeogFromWKB — Retourne un objet de type geography à partir de sa représentation binaire Well-Know Binary (WKB ou EWKB).

Synopsis

geography **ST_GeogFromWKB**(bytea wkb);

Description

`ST_GeogFromWKB` prend en paramètre une représentation binaire d'une géométrie (WKB ou EWKB) et crée une instance de type geography. Cette fonction assure le rôle de Geometry Factory en SQL.

Si le SRID n'est pas spécifié, la valeur 4326 est prise (WGS 84 long lat).

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
--Although bytea rep contains single \, these need to be escaped when inserting into a  ↩
    table
SELECT ST_AsText(
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@')
);
                                          st_astext
-----------------------------------------------------
 LINESTRING(-113.98 39.198,-113.981 39.195)
(1 row)
```

Voir aussi

ST_GeogFromText, ST_AsBinary

7.8.2.2 ST_GeomFromEWKB

ST_GeomFromEWKB — Retourne un objet ST_Geometry à partir de sa représentation binaire étendue (Extended Well-Known Binary representation, EWKB).

Synopsis

geometry **ST_GeomFromEWKB**(bytea EWKB);

Description

Retourne un objet ST_Geometry à partir de sa représentation textuelle étendue OGC (Extended Well-Known Text representation, EWKT).

Note
Le format EWKB n'est pas une norme OGC, mais un format spécifique à PostGIS incluant l'identifiant du système de référence des coordonnées (SRID)

Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

line string binary rep 0f LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.25932) in NAD 83 long lat (4269).

Note
NOTE : Si le paramètre standard_conforming_strings est à la valeur off, il est nécessaire d'échapper les caractères \ et ' avec \ et ". Ceci diffère de la représentation AsEWKB.

```
SELECT ST_GeomFromEWKB(E'\\001\\002\\000\\000 \\255\\020\\000\\000\\003\\000\\000\\000\\344 ↩
    J=
\\013B\\312Q\\300n\\303(\\010\\036!E@''\\277E''K
\\312Q\\300\\366{b\\235*!E@\\225|\\354.P\\312Q
\\300p\\231\\323e1!E@');
```

Note
Dans PostgreSQL 9.1+ - standard_conforming_strings est activé par défaut, alors que dans les versions précédentes il était désactivé. Vous pouvez modifier les valeurs par défaut selon vos besoins pour une seule requête ou au niveau de la base de données ou du serveur. Voici comment procéder avec standard_conforming_strings = on. Dans ce cas, nous échappons le ' avec le standard ansi '', mais les barres obliques ne sont pas échappées

---

```
set standard_conforming_strings = on;
SELECT ST_GeomFromEWKB('\001\002\000\000 \255\020\000\000\003\000\000\000\344J=\012\013B
    \312Q\300n\303(\010\036!E@''\277E''K\012\312Q\300\366{b\235*!E@\225|\354.P\312Q\012\300 ↩
        p\231\323e1')
```

Voir aussi

ST_AsBinary, ST_AsEWKB, ST_GeomFromWKB

7.8.2.3 ST_GeomFromWKB

ST_GeomFromWKB — Retourne un objet de type geometry à partir de sa représentation binaire Well-Know Binary (WKB) et d'un SRID optionnel.

Synopsis

geometry **ST_GeomFromWKB**(bytea geom);
geometry **ST_GeomFromWKB**(bytea geom, integer srid);

Description

ST_GeomFromWKB prend en paramètre une représentation binaire d'une géométrie (WKB ou EWKB) et un SRID optionnel (SRID) et crée une instance de type geometry. Cette fonction assure le rôle de Geometry Factory en SQL. Alias pour ST_WKBToSQL.

Si le SRID n'est pas précisé, la valeur 0 (indéfini) est prise par défaut.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.7.2 - le paramètre optionnel est issu des tests de conformité

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.41

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
--Although bytea rep contains single \, these need to be escaped when inserting into a  ↩
    table
                -- unless standard_conforming_strings is set to on.
SELECT ST_AsEWKT(
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)
);
                                          st_asewkt
-----------------------------------------------------
 SRID=4326;LINESTRING(-113.98 39.198,-113.981 39.195)
(1 row)

SELECT
  ST_AsText(
        ST_GeomFromWKB(
          ST_AsEWKB('POINT(2 5)'::geometry)
        )
  );
 st_astext
------------
 POINT(2 5)
(1 row)
```

Voir aussi

ST_WKBToSQL, ST_AsBinary, ST_GeomFromEWKB

7.8.2.4 ST_LineFromWKB

ST_LineFromWKB — Construit une LINESTRING depuis la représentation binaire WKB et le srid donné

Synopsis

geometry **ST_LineFromWKB**(bytea WKB);
geometry **ST_LineFromWKB**(bytea WKB, integer srid);

Description

ST_LineFromWKB prend en paramètre une représentation binaire d'une géométrie (WKB ou EWKB) et un SRID (SRID) et crée une instance du bon type géométrique, en l'occurence une LINESTRING. Cette fonction assure le rôle de Geometry Factory en SQL.

Si le SRID n'est pas précisé, la valeur 0 est prise par défaut. NULL est retourné si le paramètre bytea donné ne représente pas une LINESTRING.

Note
OGC SPEC 3.2.6.2 - option SRID issue des tests de conformité.

Note
Si vous savez que toutes vos géométries sont des LINESTRING, il est plus efficace d'utiliser ST_GeomFromWKB. Cette fonction appelle simplement ST_GeomFromWKB et ajoute une validation supplémentaire indiquant qu'elle renvoie une linestring.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.2.9

Exemples

```
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;
aline                              | null_return
-----------------------------------------------
0102000000020000000000000000F ... | t
```

Voir aussi

ST_GeomFromWKB, ST_LinestringFromWKB

7.8.2.5 ST_LinestringFromWKB

ST_LinestringFromWKB — Construit une géométrie depuis la représentation binaire WKB et le SRID donné.

Synopsis

geometry **ST_LinestringFromWKB**(bytea WKB);
geometry **ST_LinestringFromWKB**(bytea WKB, integer srid);

Description

La fonction ST_LinestringFromWKB prend en paramètre une représentation binaire d'une géométrie (WKB ou EWKB) et un SRID (SRID) et crée une instance du bon type géométrique, en l'occurence une LINESTRING. Cette fonction assure le rôle de Geometry Factory en SQL.

Si le SRID n'est pas précisé, la valeur 0 est prise par défaut. NULL est retourné si le paramètre bytea donné ne représente pas une LINESTRING. Alias pour ST_LineFromWKB.

Note
OGC SPEC 3.2.6.2 - SRID optionnel issu des tests de conformité.

Note
Si vous savez que toutes vos géométries sont des `LINESTRING`, il est plus efficace d'utiliser ST_GeomFromWKB. Cette fonction appelle simplement ST_GeomFromWKB et ajoute une validation supplémentaire indiquant qu'elle renvoie une `LINESTRING`.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.6.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 7.2.9

Exemples

```
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;
   aline                              | null_return
--------------------------------------------------
0102000000020000000000000000F ... | t
```

Voir aussi

ST_GeomFromWKB, ST_LineFromWKB

7.8.2.6 ST_PointFromWKB

ST_PointFromWKB — Construit une géométrie depuis la représentation binaire WKB et le SRID donné

Synopsis

geometry **ST_GeomFromWKB**(bytea geom);
geometry **ST_GeomFromWKB**(bytea geom, integer srid);

Description

`ST_PointFromWKB` prend en paramètre une représentation binaire d'une géométrie et un SRID (`SRID`) et crée une instance du bon type géométrique, en l'occurence une `POINT`. Cette fonction assure le rôle de Geometry Factory en SQL.

Si le SRID n'est pas précisé, la valeur 0 est prise par défaut. `NULL` est retourné si le paramètre `bytea` donné ne représente pas une géométrie `POINT`.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.7.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 6.1.9

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT
  ST_AsText(
        ST_PointFromWKB(
          ST_AsEWKB('POINT(2 5)'::geometry)
        )
  );
 st_astext
------------
 POINT(2 5)
(1 row)

SELECT
  ST_AsText(
        ST_PointFromWKB(
          ST_AsEWKB('LINESTRING(2 5, 2 6)'::geometry)
        )
  );
 st_astext
-----------

(1 row)
```

Voir aussi

ST_GeomFromWKB, ST_LineFromWKB

7.8.2.7 ST_WKBToSQL

ST_WKBToSQL — Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Binary (WKB). Alias pour ST_GeomFromWKB sans SRID

Synopsis

geometry **ST_WKBToSQL**(bytea WKB);

Description

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.36

Voir aussi

ST_GeomFromWKB

7.8.3 Autres formats

7.8.3.1 ST_Box2dFromGeoHash

ST_Box2dFromGeoHash — Retourne une BOX2D à partir d'une chaîne GeoHash.

Synopsis

box2d **ST_Box2dFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

Description

Retourne une BOX2D à partir d'une chaîne GeoHash.

Si aucune `precision` n'est spécifiée, ST_Box2dFromGeoHash retourne un BOX2D basé sur la précision complète de la chaîne GeoHash d'entrée.

Si `precision` est spécifié, ST_Box2dFromGeoHash utilise autant de caractère du GeoHash pour créer la BOX2D. Une précision plus basse retourne des BOXD2 plus grandes, et une valeur plus haute améliore la précision.

Disponibilité : 2.1.0

Exemples

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0');

                st_geomfromgeohash
-------------------------------------------------
 BOX(-115.172816 36.114646,-115.172816 36.114646)

SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 0);

 st_box2dfromgeohash
----------------------
 BOX(-180 -90,180 90)

 SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 10);
                            st_box2dfromgeohash
-------------------------------------------------------------------------------
 BOX(-115.17282128334 36.1146408319473,-115.172810554504 36.1146461963654)
```

Voir aussi

ST_GeoHash, ST_GeomFromGeoHash, ST_PointFromGeoHash

7.8.3.2 ST_GeomFromGeoHash

ST_GeomFromGeoHash — Retourne une geometry depuis une chaîne GeoHash.

Synopsis

geometry **ST_GeomFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

Description

Retourne une Geometry à partir d'une chaîne GeoHash. La géométrie sera un polygone représentant les limites du GeoHash.

Si aucune `precision` n'est spécifiée, ST_GeomFromGeoHash retourne un polygone basé sur la précision complète de la chaîne GeoHash en entrée.

Si `precision` est spécifié, ST_GeomFromGeoHash utilise autant de caractère du GeoHash pour créer le polygone.

Disponibilité : 2.1.0

Exemples

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0'));
                                                    st_astext
------------------------------------------------------------------------------------------------------------

 POLYGON((-115.172816 36.114646,-115.172816 36.114646,-115.172816 36.114646,-115.172816  ↩
     36.114646,-115.172816 36.114646))

SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 4));
                                                      st_astext
------------------------------------------------------------------------------------------------------------

 POLYGON((-115.3125 36.03515625,-115.3125 36.2109375,-114.9609375 36.2109375,-114.9609375  ↩
     36.03515625,-115.3125 36.03515625))

SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 10));
                                                                                  st_astext ↩

------------------------------------------------------------------------------------------------------------

 POLYGON((-115.17282128334 36.1146408319473,-115.17282128334  ↩
     36.1146461963654,-115.172810554504 36.1146461963654,-115.172810554504  ↩
     36.1146408319473,-115.17282128334 36.1146408319473))
```

Voir aussi

ST_GeoHash,ST_Box2dFromGeoHash, ST_PointFromGeoHash

7.8.3.3 ST_GeomFromGML

ST_GeomFromGML — Prend en paramètre une représentation GML d'une géométrie et renvoie un objet PostGIS de type geometry

Synopsis

geometry **ST_GeomFromGML**(text geomgml);
geometry **ST_GeomFromGML**(text geomgml, integer srid);

Description

Construit un objet PostGIS ST_Geometry à partir d'une représentation GML OGC.

La fonction ST_GeomFromGML fonctionne uniquement avec le fragment GML représentant la géométrie. Elle renvoie une error si un document GML complet est utilisé.

version OGC GML supportée :

- GML 3.2.1 Namespace
- GML 3.1.1 Simple Features profile SF-2 (with GML 3.1.0 and 3.0.0 backward compatibility)
- GML 2.1.2

OGC GML standards, cf : http://www.opengeospatial.org/standards/gml :

Disponibilité : 1.5, nécessite libxml2 1.6+

Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques.

Amélioration : 2.0.0 paramètre optionnel de srid par défaut ajouté.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Le format GML supporte des objets de dimensions différentes (2D et 3D dans la même MultiGeometry par exemple). PostGIS ne supportant pas cela, la fonction convertit toute la géometrie en 2D si une seule coordonnée Z manque.

Le format GML supporte des objets ayant des SRID différents dans la même MultiGeometry. PostGIS ne supportant pas cela, ST_GeomFromGML reprojète toutes les sous géométries dans le SRS du noeud racine. Si aucun attribut srsName n'est disponible pour le noeud racine GML, la fonction renvoie une erreur.

La fonction ST_GeomFromGML n'impose pas d'utiliser un espace de noms GML explicite. Pour les usages courants, il peut être ignoré. Il est en revanche nécessaire en cas d'utilisation de la fonctionnalité XLink dans le GML.

Note

La fonction ST_GeomFromGML ne supporte pas les géométries de type SQL/MM courbes.

Exemple : une géométrie unique avec srsName

```
SELECT ST_GeomFromGML($$
    <gml:LineString xmlns:gml="http://www.opengis.net/gml"
                        srsName="EPSG:4269">
        <gml:coordinates>
            -71.16028,42.258729 -71.160837,42.259112 -71.161143,42.25932
        </gml:coordinates>
    </gml:LineString>
$$);
```

Exemple - utilisation de XLink

```
SELECT ST_GeomFromGML($$
    <gml:LineString xmlns:gml="http://www.opengis.net/gml"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            srsName="urn:ogc:def:crs:EPSG::4269">
        <gml:pointProperty>
            <gml:Point gml:id="p1"
><gml:pos
>42.258729 -71.16028</gml:pos
></gml:Point>
        </gml:pointProperty>
        <gml:pos
>42.259112 -71.160837</gml:pos>
        <gml:pointProperty>
            <gml:Point xlink:type="simple" xlink:href="#p1"/>
        </gml:pointProperty>
    </gml:LineString>
$$);
```

Exemple - Surface Polyédrique

```
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 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
>'));

-- result --
 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)))
```

Voir aussi

Section 2.2.3, ST_AsGML, ST_GMLToSQL

7.8.3.4 ST_GeomFromGeoJSON

ST_GeomFromGeoJSON — Prend en entrée une géométrie au format geojson et renvoie un objet Postgis de type geometry

Synopsis

geometry **ST_GeomFromGeoJSON**(text geomjson);
geometry **ST_GeomFromGeoJSON**(json geomjson);
geometry **ST_GeomFromGeoJSON**(jsonb geomjson);

Description

Construit un objet Postgis de type geometry à partir d'une représentation GeoJSON.

La fonction ST_GeomFromGeoJSON fonctionne uniquement avec le fragment JSON représentant la géométrie. Elle renvoie une erreur si un document JSON complet est utilisé.

Amélioré : 3.0.0 La géométrie parsée prend par défaut la valeur SRID=4326 si elle n'est pas spécifiée autrement.

Amélioration : 2.5.0 peut maintenant accepter json et jsonb comme entrées.

Disponibilité : 2.0.0 nécessite JSON-C >= 0.9

Note
Si JSON-C n'est pas disponible sur le système, une erreur est renvoyée. Pour activer JSON-C, lancer configure --with-jsondir=/path/to/json-c. Cf. Section 2.2.3 pour plus de détails.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsText(ST_GeomFromGeoJSON('{"type":"Point","coordinates":[-48.23456,20.12345]}')) ↩
    As wkt;
wkt
------
POINT(-48.23456 20.12345)
```

```
-- a 3D linestring
SELECT ST_AsText(ST_GeomFromGeoJSON('{"type":"LineString","coordinates ↩
   ":[[1,2,3],[4,5,6],[7,8,9]]}')) As wkt;

wkt
-------------------
LINESTRING(1 2,4 5,7 8)
```

Voir aussi

ST_AsText, ST_AsGeoJSON, Section 2.2.3

7.8.3.5 ST_GeomFromKML

ST_GeomFromKML — Prend en entrée une géométrie au format KML et renvoie un objet Postgis de type geometry

Synopsis

geometry **ST_GeomFromKML**(text geomkml);

Description

Construit un objet Postgis de type geometry à partir d'une représentation OGC KML.

La fonction ST_GeomFromKML fonctionne uniquement avec le fragment KML représentant la géométrie. Elle renvoie une erreur si un document KML complet est utilisé.

versions OGC KML supportées :

- KML 2.2.0 Namespace

OGC KML standards, cf : http://www.opengeospatial.org/standards/kml :

Disponibilité : 1.5, nécessite libxml2 2.6+

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Note
La fonction ST_GeomFromGML ne supporte pas les géométries de type SQL/MM courbes.

Exemple : une géométrie unique avec srsName

```
SELECT ST_GeomFromKML($$
    <LineString>
        <coordinates
>-71.1663,42.2614
            -71.1667,42.2616</coordinates>
    </LineString>
$$);
```

Voir aussi

Section 2.2.3, ST_AsKML

7.8.3.6 ST_GeomFromTWKB

ST_GeomFromTWKB — Crée une instance de geometry depuis une représentation de géométrie en TWKB ("Tiny Well-Known Binary").

Synopsis

geometry **ST_GeomFromTWKB**(bytea twkb);

Description

La fonction `ST_GeomFromTWKB` prend une représentation de géométrie TWKB ("Tiny Well-Known Binary") et crée une instance du type de géométrie approprié.

Exemples

```
SELECT ST_AsText(ST_GeomFromTWKB(ST_AsTWKB('LINESTRING(126 34, 127 35)'::geometry)));

         st_astext
-----------------------------
 LINESTRING(126 34, 127 35)
(1 row)


SELECT ST_AsEWKT(
  ST_GeomFromTWKB(E'\\x620002f7f40dbce4040105')
);
                                          st_asewkt
-------------------------------------------------------------
LINESTRING(-113.98 39.198,-113.981 39.195)
(1 row)
```

Voir aussi

ST_AsTWKB

7.8.3.7 ST_GMLToSQL

ST_GMLToSQL — Retourne un objet de type ST_Geometry à partir de sa représentation GML. Alias pour ST_GeomFromGML

Synopsis

geometry **ST_GMLToSQL**(text geomgml);
geometry **ST_GMLToSQL**(text geomgml, integer srid);

Description

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.50 (sauf pour le support des courbes).

Disponibilité : 1.5, nécessite libxml2 1.6+

Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques.

Amélioration : 2.0.0 paramètre optionnel de srid par défaut ajouté.

Voir aussi

Section 2.2.3, ST_GeomFromGML, ST_AsGML

7.8.3.8 ST_LineFromEncodedPolyline

ST_LineFromEncodedPolyline — Crée une LineString depuis une polyligne encodée ("Encoded Polyline").

Synopsis

geometry **ST_LineFromEncodedPolyline**(text polyline, integer precision=5);

Description

Crée une LineString à partir d'une chaîne de polyligne encodée ("Encoded Polyline").

L'option `precision` spécifie le nombre de décimales qui seront préservées dans la polyligne encodée. La valeur doit être la même à l'encodage et au décodage, sinon les coordonnées seront incorrectes.

Voir http://developers.google.com/maps/documentation/utilities/polylinealgorithm

Disponibilité : 2.2.0

Exemples

```
-- Create a line string from a polyline
SELECT ST_AsEWKT(ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@'));
-- result --
SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)

-- Select different precision that was used for polyline encoding
SELECT ST_AsEWKT(ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@',6));
-- result --
SRID=4326;LINESTRING(-12.02 3.85,-12.095 4.07,-12.6453 4.3252)
```

Voir aussi

ST_AsEncodedPolyline

7.8.3.9 ST_PointFromGeoHash

ST_PointFromGeoHash — Retourne un point à partir d'une chaîne GeoHash.

Synopsis

point **ST_PointFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

Description

Retourne une point à partir d'une chaîne GeoHash. Le point représente le centre du GeoHash.

Si aucune `precision` n'est spécifiée, ST_PointFromGeoHash retourne une un point basé sur la précision complète de la chaîne GeoHash en entrée.

Si `precision` est spécifié, ST_PointFromGeoHash utilise autant de caractère du GeoHash pour créer le point.

Disponibilité : 2.1.0

Exemples

```
SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0'));
          st_astext
------------------------------
 POINT(-115.172816 36.114646)

SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 4));
             st_astext
-----------------------------------
 POINT(-115.13671875 36.123046875)

SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 10));
                 st_astext
-------------------------------------------
 POINT(-115.172815918922 36.1146435141563)
```

Voir aussi

ST_GeoHash, ST_Box2dFromGeoHash, ST_GeomFromGeoHash

7.8.3.10 ST_FromFlatGeobufToTable

ST_FromFlatGeobufToTable — Crée une table basée sur la structure des données FlatGeobuf.

Synopsis

void **ST_FromFlatGeobufToTable**(text schemaname, text tablename, bytea FlatGeobuf input data);

Description

Crée une table basée sur la structure des données FlatGeobuf. (http://flatgeobuf.org).

`schema` Nom du schéma.

`table` Nom de la table.

`data` Données FlatGeobuf en entrée.

Disponibilité : 3.2.0

7.8.3.11 ST_FromFlatGeobuf

ST_FromFlatGeobuf — Lit les données FlatGeobuf.

Synopsis

setof anyelement **ST_FromFlatGeobuf**(anyelement Table reference, bytea FlatGeobuf input data);

Description

Lit les données FlatGeobuf (http://flatgeobuf.org). REMARQUE : les chaînes binaires de PostgreSQL ne peuvent pas dépasser 1 Go.

`tabletype` référence à un type de table.

`data` données d'entrée FlatGeobuf.

Disponibilité : 3.2.0

7.9 Export de géométrie

7.9.1 Well-Known Text (WKT)

7.9.1.1 ST_AsEWKT

ST_AsEWKT — Renvoie la représentation Well-Known Text (WKT) de la géométrie avec les métadonnées SRID.

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);

Description

Renvoie la représentation Well-Known Text de la géométrie préfixée par le SRID. L'argument facultatif `maxdecimaldigits` peut être utilisé pour réduire le nombre maximal de chiffres décimaux après la virgule flottante utilisés dans la sortie (valeur par défaut : 15).

Pour effectuer la conversion inverse de la représentation EWKT en géométrie PostGIS, utilisez ST_GeomFromEWKT.

Warning
L'utilisation du paramètre `maxdecimaldigits` peut rendre la géométrie de sortie invalide. Pour éviter cela, utilisez d'abord ST_ReducePrecision avec une taille de grille appropriée.

Note
La spécification WKT ne comprend pas le SRID. Pour obtenir le format WKT de l'OGC, utilisez ST_AsText.

Warning
Le format WKT ne maintient pas la précision. Pour éviter la troncature flottante, utilisez le format ST_AsBinary ou ST_AsEWKB pour le transport.

Amélioré : support du paramètre optionnel de précision dans la version 3.1.0.

Amélioration : la version 2.0.0 a introduit la prise en charge de la geography, des surfaces polyédriques, des triangles et des TIN.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_AsEWKT('0103000020E6100000010000005000000000000000
      0000000000000000000000000000000000000000000000000000000000
      F03F000000000000F03F000000000000F03F000000000000F03F
      F0000000000000000000000000000000000000000000000000'::geometry);

       st_asewkt
---------------------------------
SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
(1 row)

SELECT ST_AsEWKT('0108000080030000000000000060 ↩
    E30A4100000000785C0241000000000000F03F0000000018
E20A4100000000485F0241000000000000400000000018
E20A4100000000305C0241000000000000840')

--st_asewkt---
CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)
```

Voir aussi

ST_AsBinary, ST_AsEWKB, ST_AsText, ST_GeomFromEWKT

7.9.1.2 ST_AsText

ST_AsText — Renvoie la représentation Well-Known Text (WKT) de la géométrie/geography sans métadonnées SRID.

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);

Description

Renvoie la représentation OGC Well-Known Text (WKT) de la géométrie/geography. L'argument facultatif *maxdecimaldigits* peut être utilisé pour limiter le nombre de chiffres après la virgule dans les ordonnées de sortie (valeur par défaut : 15).

Pour effectuer la conversion inverse de la représentation WKT en géométrie PostGIS, utilisez ST_GeomFromText.

Note
La représentation WKT standard de l'OGC n'inclut pas le SRID. Pour inclure le SRID dans la représentation de sortie, utilisez la fonction PostGIS non standard ST_AsEWKT

Warning
La représentation textuelle des nombres en WKT peut ne pas maintenir une précision totale en virgule flottante. Pour garantir une précision totale pour le stockage ou le transport des données, il est préférable d'utiliser le format Well-Known Binary (WKB) (voir ST_AsBinary et *maxdecimaldigits*).

Warning
L'utilisation du paramètre *maxdecimaldigits* peut rendre la géométrie de sortie invalide. Pour éviter cela, utilisez d'abord ST_ReducePrecision avec une taille de grille appropriée.

Disponibilité : 1.5 - le support de la geography a été introduit.

Amélioration : 2.5 - introduction de la précision des paramètres optionnels.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.25

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_AsText('0103000000010000000500000000000000000
0000000000000000000000000000000000000000000000000
F03F000000000000F03F000000000000F03F000000000000F03
F000000000000000000000000000000000000000000000000');

    st_astext
-------------------------------
 POLYGON((0 0,0 1,1 1,1 0,0 0))
```

La sortie de précision complète est la valeur par défaut.

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'));
    st_astext
-----------------------------
 POINT(111.1111111 1.1111111)
```

L'argument *maxdecimaldigits* peut être utilisé pour limiter la précision de la sortie.

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'), 2);
    st_astext
--------------------
 POINT(111.11 1.11)
```

Voir aussi

ST_AsBinary, ST_AsEWKB, ST_AsEWKT, ST_GeomFromText

7.9.2 Well-Known Binary (WKB)

7.9.2.1 ST_AsBinary

ST_AsBinary — Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.

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);

Description

Returns the OGC/ISO Well-Known Binary (WKB) representation of the geometry. The first function variant defaults to encoding using server machine endian. The second function variant takes a text argument specifying the endian encoding: either 'NDR' for little-endian; or 'XDR' for big-endian. Supplying unknown arguments will result in little-endian output.

Le format WKB est utile pour lire les données géométriques de la base de données et maintenir une précision numérique totale. Cela permet d'éviter les arrondis de précision qui peuvent se produire avec les formats texte tels que WKT.

Pour effectuer la conversion inverse de la géométrie WKB en géométrie PostGIS, utilisez ST_GeomFromWKB.

Note
Le format WKB de l'OGC/ISO ne comprend pas le SRID. Pour obtenir le format EWKB qui inclut le SRID, utilisez ST_AsEWKB

Note
Le comportement par défaut dans PostgreSQL 9.0 a été modifié pour sortir bytea en encodage hexagonal. Si vos outils GUI nécessitent l'ancien comportement, alors SET bytea_output='escape' dans votre base de données.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Amélioration : 2.0.0 le support pour des dimensions de coordonnées plus élevées a été introduit.

Amélioration : 2.0.0 le support pour spécifier endian avec geography a été introduit.

Disponibilité : 1.5.0 Le support de la géographie a été introduit.

Modifié : 2.0.0 Les entrées de cette fonction ne peuvent pas être inconnues, elles doivent être des géométries. Des constructions telles que `ST_AsBinary('POINT(1 2)')` ne sont plus valides et vous obtiendrez une erreur de type `n st_asbinary(unknown is not unique`. Un code comme celui-là doit être changé en `ST_AsBinary('POINT(1 2)'::geometry);`. Si cela n'est pas possible, alors installez `legacy.sql`.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.37

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

       st_asbinary
---------------------------------
\x0103000000010000000500000000000000000000000000000000000000000000000000f03f000000000000f03f000000000000f03f000000000000f03f00000000000000000000000000000000
```

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326), 'XDR');
       st_asbinary
--------------------------------
\x000000000300000001000000050000000000000000000000000000000000000000000000000003ff000
00000000003ff00000000000003ff00000000000003ff00000000000000000000000000000000000
0000000000000000000000000
```

Voir aussi

ST_GeomFromWKB, ST_AsEWKB, ST_AsTWKB, ST_AsText,

7.9.2.2 ST_AsEWKB

ST_AsEWKB — Renvoie la représentation Extended Well-Known Binary (EWKB) de la géométrie avec les métadonnées SRID.

Synopsis

bytea **ST_AsEWKB**(geometry g1);
bytea **ST_AsEWKB**(geometry g1, text NDR_or_XDR);

Description

Returns the Extended Well-Known Binary (EWKB) representation of the geometry with SRID metadata. The first function variant defaults to encoding using server machine endian. The second function variant takes a text argument specifying the endian encoding: either 'NDR' for little-endian; or 'XDR' for big-endian. Supplying unknown arguments will result in little-endian output.

Le format WKB est utile pour lire les données géométriques de la base de données et maintenir une précision numérique totale. Cela permet d'éviter les arrondis de précision qui peuvent se produire avec les formats texte tels que WKT.

Pour effectuer la conversion inverse de la géométrie EWKB en géométrie PostGIS, utilisez ST_GeomFromEWKB.

> **Note**
> Pour obtenir le format OGC/ISO WKB, utilisez ST_AsBinary. Notez que le format OGC/ISO WKB ne comprend pas le SRID.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

- Cette fonction prend en charge la 3D et ne supprime pas l'indice z.
- Cette méthode prend en charge les types Circular String et Curve.
- Cette fonction prend en charge les surfaces Polyhedral.
- Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

       st_asewkb
---------------------------------
\x0103000020e6100000010000000500000000000000000000000000000000000000000000000000000000000000000000
0000000000000f03f000000000000f03f000000000000f03f000000000000f03f000000000000000
0000000000000000000000000000000000
```

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326), 'XDR');
       st_asewkb
---------------------------------
\x0020000003000010e600000001000000050000000000000000000000000000000000000000000000000000000000000000
003ff0000000000003ff0000000000003ff0000000000003ff0000000000000000000000000000
0000000000000000000000000000000000
```

Voir aussi

ST_AsBinary, ST_GeomFromEWKB, ST_SRID

7.9.2.3 ST_AsHEXEWKB

ST_AsHEXEWKB — Renvoie une géométrie au format HEXEWKB (en tant que texte) en utilisant l'encodage little-endian (NDR) ou big-endian (XDR).

Synopsis

text **ST_AsHEXEWKB**(geometry g1, text NDRorXDR);
text **ST_AsHEXEWKB**(geometry g1);

Description

Renvoie une géométrie au format HEXEWKB (en tant que texte) en utilisant l'encodage little-endian (NDR) ou big-endian (XDR). Si aucun encodage n'est spécifié, NDR est utilisé.

Note
Disponibilité : 1.2.2

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_AsHEXEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
    which gives same answer as

    SELECT ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326)::text;

    st_ashexewkb
    --------
    0103000020E610000001000000500
```

```
    00000000000000000000000000000000
    00000000000000000000000000000F03F
    000000000000F03F000000000000F03F0000000000000F03
    F000000000000000000000000000000000000000000000000
```

7.9.3 Autres formats

7.9.3.1 ST_AsEncodedPolyline

ST_AsEncodedPolyline — Renvoie une polyligne encodée à partir d'une géométrie LineString.

Synopsis

text **ST_AsEncodedPolyline**(geometry geom, integer precision=5);

Description

Renvoie la géométrie sous forme de polyligne encodée. Ce format est utilisé par Google Maps avec précision=5 et par Open Source Routing Machine avec précision=5 et 6.

L'option `precision` spécifie le nombre de décimales qui seront préservées dans la polyligne encodée. La valeur doit être la même à l'encodage et au décodage, sinon les coordonnées seront incorrectes.

Disponibilité : 2.2.0

Exemples

Base

```
SELECT ST_AsEncodedPolyline(GeomFromEWKT('SRID=4326;LINESTRING(-120.2 38.5,-120.95  ↩
    40.7,-126.453 43.252)'));
  --result--
  |_p~iF~ps|U_ulLnnqC_mqNvxq`@
```

Utiliser en conjonction avec geography linestring et geography segmentize, et mettre sur google maps

```
-- the SQL for Boston to San Francisco, segments every 100 KM
  SELECT ST_AsEncodedPolyline(
    ST_Segmentize(
      ST_GeogFromText('LINESTRING(-71.0519 42.4935,-122.4483 37.64)'),
        100000)::geometry) As encodedFlightPath;
```

Le javascript ressemblera à quelque chose comme ceci où la variable $ est remplacée par le résultat de la requête

```
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?libraries= ↩
    geometry"
></script>
<script type="text/javascript">
   flightPath = new google.maps.Polyline({
      path:  google.maps.geometry.encoding.decodePath("$encodedFlightPath"),
      map: map,
      strokeColor: '#0000CC',
      strokeOpacity: 1.0,
      strokeWeight: 4
    });
</script>
```

Voir aussi

ST_LineFromEncodedPolyline, ST_Segmentize

7.9.3.2 ST_AsFlatGeobuf

ST_AsFlatGeobuf — Renvoie une représentation FlatGeobuf d'un ensemble de lignes.

Synopsis

bytea **ST_AsFlatGeobuf**(anyelement set row);
bytea **ST_AsFlatGeobuf**(anyelement row, bool index);
bytea **ST_AsFlatGeobuf**(anyelement row, bool index, text geom_name);

Description

Renvoie une représentation FlatGeobuf (http://flatgeobuf.org) d'un ensemble de lignes correspondant à une FeatureCollection. REMARQUE : les octets PostgreSQL ne peuvent pas dépasser 1 Go.

`row` données de ligne avec au moins une colonne de géométrie.

`index` basculer la création d'un index spatial. La valeur par défaut est false.

`geom_name` est le nom de la colonne de géométrie dans les données de la ligne. Si elle est NULL, elle prendra par défaut la première colonne de géométrie trouvée.

Disponibilité : 3.2.0

7.9.3.3 ST_AsGeobuf

ST_AsGeobuf — Retourne une représentation Geobuf d'un ensemble de lignes.

Synopsis

bytea **ST_AsGeobuf**(anyelement set row);
bytea **ST_AsGeobuf**(anyelement row, text geom_name);

Description

Renvoie une représentation Geobuf (https://github.com/mapbox/geobuf) d'un ensemble de lignes correspondant à une FeatureCollection. Chaque géométrie en entrée est analysée afin de déterminer la précision maximale pour un stockage optimal. Notez que Geobuf dans sa forme actuelle ne peut pas être streamé, donc la sortie complète sera assemblée en mémoire.

`row` données de ligne avec au moins une colonne de géométrie.

`geom_name` est le nom de la colonne de géométrie dans les données de la ligne. Si elle est NULL, elle prendra par défaut la première colonne de géométrie trouvée.

Disponibilité : 2.4.0

Exemples

```
SELECT encode(ST_AsGeobuf(q, 'geom'), 'base64')
    FROM (SELECT ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))') AS geom) AS q;
 st_asgeobuf
---------------------------------
 GAAiEAoOCgwIBBoIAAAAAgIAAAE=
```

7.9.3.4 ST_AsGeoJSON

ST_AsGeoJSON — Renvoyer une géométrie ou un élément au format GeoJSON.

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);

Description

Renvoie une géométrie sous forme d'objet GeoJSON "geometry", ou une ligne sous forme d'objet GeoJSON "feature".

Les représentations de géométrie et d'entités GeoJSON résultantes sont conformes aux spécifications GeoJSON RFC 7946, sauf lorsque les géométries analysées sont référencées avec un CRS autre que la longitude et la latitude WGS84 (EPSG:4326, urn:ogc:def:crs:OGC::CRS84) ; l'objet géométrique GeoJSON aura alors un identifiant SRID CRS court attaché par défaut. Les géométries 2D et 3D sont toutes deux supportées. GeoJSON ne supporte que les types de géométrie SFS 1.1 (pas de support pour les courbes par exemple).

Le paramètre `geom_column` est utilisé pour faire la distinction entre plusieurs colonnes géométriques. S'il est omis, la première colonne géométrique de l'enregistrement sera déterminée. Inversement, l'utilisation de ce paramètre permet d'économiser les recherches sur le type de colonne.

L'argument `maxdecimaldigits` peut être utilisé pour réduire le nombre maximum de décimales utilisées dans la sortie (par défaut 9). Si vous utilisez EPSG:4326 et que vous sortez la géométrie uniquement pour l'affichage, `maxdecimaldigits`=6 peut être un bon choix pour de nombreuses cartes.

Warning
L'utilisation du paramètre *`maxdecimaldigits`* peut rendre la géométrie de sortie invalide. Pour éviter cela, utilisez d'abord ST_ReducePrecision avec une taille de grille appropriée.

L'argument `options` peut être utilisé pour ajouter BBOX ou CRS dans la sortie GeoJSON :

- 0 : signifie aucune option
- 1 : GeoJSON BBOX
- 2 : GeoJSON Short CRS (e.g. EPSG:4326)
- 4 : GeoJSON Long CRS (e.g. urn:ogc:def:crs:EPSG::4326)
- 8 : GeoJSON Short CRS si pas EPSG:4326 (par défaut)

Le paramètre `id_column` est utilisé pour définir le membre "id" des caractéristiques GeoJSON renvoyées. Conformément à la RFC GeoJSON, ce paramètre DEVRAIT être utilisé chaque fois qu'un élément a un identifiant couramment utilisé, tel qu'une clé primaire. Si elle n'est pas spécifiée, les entités produites n'auront pas de membre "id" et toutes les colonnes autres que la géométrie, y compris les clés potentielles, se retrouveront dans le membre "properties" de l'entité.

La spécification GeoJSON indique que les polygones sont orientés selon la règle de la main droite, et certains clients exigent cette orientation. Ceci peut être assuré en utilisant ST_ForcePolygonCCW . La spécification exige également que la géométrie soit dans le système de coordonnées WGS84 (SRID = 4326). Si nécessaire, la géométrie peut être projetée en WGS84 en utilisant ST_Transform : `ST_Transform( geom, 4326 )`.

GeoJSON peut être testé et visualisé en ligne sur geojson.io et geojsonlint.com. Il est largement pris en charge par les frameworks de cartographie web :

- Exemple OpenLayers GeoJSON

- Exemple Leaflet GeoJSON
- Exemple Mapbox GL GeoJSON

Disponibilité : 1.3.4

Disponibilité : 1.5.0 Le support de la géographie a été introduit.

Modifié : 2.0.0 supporte les args par défaut et les args nommés.

Modifié : la version 3.0.0 prend en charge les enregistrements en tant que données d'entrée

Modifié : 3.0.0 SRID de sortie si ce n'est pas EPSG:4326.

Modifié : la version 3.5.0 permet de spécifier la colonne contenant l'identifiant de l'élément

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Générer une FeatureCollection :

```
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);
```

```
{"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"}}]}
```

Génère une Feature :

```
SELECT ST_AsGeoJSON(t.*, id_column =
> 'id')
FROM (VALUES (1, 'one', 'POINT(1 1)'::geometry)) AS t(id, name, geom);
```

```
st_asgeojson
----------------------------------------------------------------------------------------------------

 {"type": "Feature", "geometry": {"type":"Point","coordinates":[1,1]}, "id": 1, "properties ↩
     ": {"name": "one"}}
```

N'oubliez pas de transformer vos données en longitude et latitude WGS84 pour vous conformer à la spécification GeoJSON :

```
SELECT ST_AsGeoJSON(ST_Transform(geom,4326)) from fe_edges limit 1;
```

```
st_asgeojson
----------------------------------------------------------------------------------------------------


{"type":"MultiLineString","coordinates":[[[-89.734634999999997,31.492072000000000],
[-89.734955999999997,31.492237999999997]]]}
```

Les géométries 3D sont prises en charge :

```
SELECT ST_AsGeoJSON('LINESTRING(1 2 3, 4 5 6)');
```

```
{"type":"LineString","coordinates":[[1,2,3],[4,5,6]]}
```

Options argument can be used to add BBOX and CRS in GeoJSON output:

```
SELECT ST_AsGeoJSON(ST_SetSRID('POINT(1 1)'::geometry, 4326), 9, 4|1);
```

```
{"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]}
```

Voir aussi

ST_GeomFromGeoJSON, ST_ForcePolygonCCW , ST_Transform

7.9.3.5 ST_AsGML

ST_AsGML — Renvoyer la géométrie en tant qu'élément GML version 2 ou 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);

Description

Renvoie la géométrie sous la forme d'un élément GML (Geography Markup Language). Le paramètre de version, s'il est spécifié, peut être 2 ou 3. Si aucun paramètre de version n'est spécifié, la valeur par défaut est 2. L'argument `maxdecimaldigits` peut être utilisé pour réduire le nombre maximum de décimales utilisées dans la sortie (15 par défaut).

Warning
L'utilisation du paramètre *maxdecimaldigits* peut rendre la géométrie de sortie invalide. Pour éviter cela, utilisez d'abord ST_ReducePrecision avec une taille de grille appropriée.

GML 2 fait référence à la version 2.1.2, GML 3 à la version 3.1.1

L'argument "options" est un champ de bits. Il peut être utilisé pour définir le type de sortie CRS dans la sortie GML, et pour déclarer les données comme lat/lon :

- 0 : GML Short CRS (e.g. EPSG:4326), valeur par défaut
- 1: GML Long CRS (e.g urn:ogc:def:crs:EPSG::4326)
- 2 : Pour GML 3 uniquement, supprimer l'attribut srsDimension de la sortie.
- 4 : Pour GML 3 uniquement, utilisez la balise <LineString> plutôt que la balise <Curve> pour les lignes.
- 16 : Déclare que les données sont des lat/lon (par exemple srid=4326). Par défaut, on suppose que les données sont planaires. Cette option n'est utile que pour la sortie GML 3.1.1, en ce qui concerne l'ordre des axes. Ainsi, si vous la définissez, les coordonnées seront échangées pour que l'ordre soit lat lon au lieu de lon lat dans la base de données.

- 32 : Sortie de la boîte de la géométrie (enveloppe).

L'argument 'namespace prefix' peut être utilisé pour spécifier un préfixe d'espace de noms personnalisé ou aucun préfixe (s'il est vide). Si l'argument est nul ou omis, le préfixe 'gml' est utilisé

Disponibilité : 1.3.2

Disponibilité : 1.5.0 Le support de la géographie a été introduit.

Amélioration : la prise en charge du préfixe 2.0.0 a été introduite. L'option 4 pour GML3 a été introduite pour permettre l'utilisation de LineString au lieu de Curve tag pour les lignes. La prise en charge GML3 des surfaces polyédriques et des TINS a été introduite. L'option 32 a été introduite pour produire la boîte.

Modifié : 2.0.0 utiliser les args nommés par défaut

Amélioration : 2.1.0 La prise en charge des identifiants a été introduite pour GML 3.

Note
Seule la version 3+ de ST_AsGML prend en charge les surfaces polyédriques et les TINS.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 17.2

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples : Version 2

```
SELECT ST_AsGML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
    st_asgml
    --------
    <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>
```

Exemples : Version 3

```
-- Flip coordinates and output extended EPSG (16 | 1)--
SELECT ST_AsGML(3, ST_GeomFromText('POINT(5.234234233242 6.34534534534)',4326), 5, 17);
      st_asgml
      --------
    <gml:Point srsName="urn:ogc:def:crs:EPSG::4326"
><gml:pos
>6.34535 5.23423</gml:pos
></gml:Point>
```

```
-- Output the envelope (32) --
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 32);
    st_asgml
    --------
  <gml:Envelope srsName="EPSG:4326">
    <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 --
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 49);
  st_asgml
  --------
<gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
  <gml:lowerCorner
>2 1</gml:lowerCorner>
  <gml:upperCorner
>20 10</gml:upperCorner>
</gml:Envelope>
```

```
-- Polyhedral Example --
SELECT ST_AsGML(3, 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)) )'));
  st_asgml
  --------
 <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 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>
```

Voir aussi

ST_GeomFromGML

7.9.3.6 ST_AsKML

ST_AsKML — Renvoyer la géométrie sous forme d'élément KML.

Synopsis

text **ST_AsKML**(geometry geom, integer maxdecimaldigits=15, text nprefix=NULL);
text **ST_AsKML**(geography geog, integer maxdecimaldigits=15, text nprefix=NULL);

Description

Renvoie la géométrie sous la forme d'un élément KML (Keyhole Markup Language). Le nombre maximal de décimales par défaut est de 15, la valeur par défaut du namespace est sans préfixe.

Warning
L'utilisation du paramètre *maxdecimaldigits* peut rendre la géométrie de sortie invalide. Pour éviter cela, utilisez d'abord ST_ReducePrecision avec une taille de grille appropriée.

Note
Exige que PostGIS soit compilé avec le support Proj. Utilisez PostGIS_Full_Version pour confirmer que vous avez compilé le support Proj.

Note
Disponibilité : 1.2.2 - les variantes ultérieures qui incluent le paramétrage de la version sont disponibles dans la version 1.3.2

Note
Amélioré : 2.0.0 - Ajout d'un préfixe namespace, utilisation d'arguments par défaut et d'arguments nommés

Note
Modifié : 3.0.0 - Suppression de la signature de la variante "versioned"

Note
La sortie AsKML ne fonctionnera pas avec les géométries qui n'ont pas de SRID

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsKML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

    st_askml
    --------
    <Polygon
><outerBoundaryIs
><LinearRing
><coordinates
>0,0 0,1 1,1 1,0 0,0</coordinates
></LinearRing
></outerBoundaryIs
></Polygon>

    --3d linestring
    SELECT ST_AsKML('SRID=4326;LINESTRING(1 2 3, 4 5 6)');
    <LineString
><coordinates
>1,2,3 4,5,6</coordinates
></LineString>
```

Voir aussi

ST_AsSVG, ST_AsGML

7.9.3.7 ST_AsLatLonText

ST_AsLatLonText — Renvoie la représentation en degrés, minutes et secondes du point donné.

Synopsis

text **ST_AsLatLonText**(geometry pt, text format='');

Description

Renvoie la représentation en degrés, minutes et secondes du point.

Note
On suppose que le point se trouve dans une projection lat/lon. Les coordonnées X (lon) et Y (lat) sont normalisées dans la sortie à la plage "normale" (-180 à +180 pour lon, -90 à +90 pour lat).

Le paramètre text est une chaîne de caractères contenant le format du texte résultant, similaire à une chaîne de caractères de date. Les symboles autorisés sont "D" pour les degrés, "M" pour les minutes, "S" pour les secondes et "C" pour la direction cardinale (NSEW). Les jetons DMS peuvent être répétés pour indiquer la largeur et la précision souhaitées ("SSS.SSSS" signifie "1,0023").

Les lettres "M", "S" et "C" sont facultatives. Si "C" est omis, les degrés sont indiqués avec un signe "-" s'il s'agit du sud ou de l'ouest. Si "S" est omis, les minutes sont affichées sous forme décimale avec autant de chiffres de précision que vous le spécifiez. Si "M" est également omis, les degrés sont affichés sous forme décimale avec autant de chiffres de précision que vous le spécifiez.

Si la chaîne de caractères format est omise (ou de longueur nulle), un format par défaut sera utilisé.

Disponibilité : 2.0

Exemples

Format par défaut.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)'));
      st_aslatlontext
----------------------------
 2\textdegree{}19'29.928"S 3\textdegree{}14'3.243"W
```

Fournir un format (identique au format par défaut).

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D\textdegree{}M''S.SSS"C'));
      st_aslatlontext
----------------------------
 2\textdegree{}19'29.928"S 3\textdegree{}14'3.243"W
```

Les caractères autres que D, M, S, C et . ne sont pas pris en compte.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D degrees, M minutes, S seconds to ↩
   the C'));
                                st_aslatlontext
-------------------------------------------------------------------------------------
 2 degrees, 19 minutes, 30 seconds to the S 3 degrees, 14 minutes, 3 seconds to the W
```

Les degrés signés au lieu des directions cardinales.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D\textdegree{}M''S.SSS"'));
      st_aslatlontext
----------------------------
 -2\textdegree{}19'29.928" -3\textdegree{}14'3.243"
```

Degrés décimaux.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D.DDDD degrees C'));
          st_aslatlontext
-----------------------------------
 2.3250 degrees S 3.2342 degrees W
```

Les valeurs trop importantes sont normalisées.

```
SELECT (ST_AsLatLonText('POINT (-302.2342342 -792.32498)'));
        st_aslatlontext
------------------------------
 72\textdegree{}19'29.928"S 57\textdegree{}45'56.757"E
```

7.9.3.8 ST_AsMARC21

ST_AsMARC21 — Renvoie la géométrie sous forme d'enregistrement MARC21/XML avec un champ de données géographiques (034).

Synopsis

text **ST_AsMARC21** (geometry geom , text format='hdddmmss');

Description

Cette fonction renvoie un enregistrement MARC21/XML avec Coded Cartographic Mathematical Data représentant la boîte de délimitation d'une géométrie donnée. Le paramètre `format` permet de coder les coordonnées dans les sous-champs `$d`,`$e`,`$f` et `$g` dans tous les formats pris en charge par la norme MARC21/XML. Les formats valides sont les suivants :

- la direction cardinale, en degrés, minutes et secondes (par défaut) : `hdddmmss`
- degrés décimaux avec la direction cardinale : `hddd.dddddd`
- degrés décimaux sans direction cardinale : `ddd.dddddd`
- minutes décimales avec la direction cardinale : `hdddmm.mmmm`
- minutes décimales sans direction cardinale : `dddmm.mmmm`
- secondes décimales avec la direction cardinale : `hdddmmss.sss`

Le signe décimal peut également être une virgule, par exemple `hdddmm,mmmm`.

La précision des formats décimaux peut être limitée par le nombre de caractères après le signe décimal, par exemple `hdddmm.mm` pour des minutes décimales avec une précision de deux décimales.

Cette fonction ignore les dimensions Z et M.

Prise en charge des versions LOC MARC21/XML :

- MARC21/XML 1.1

Disponibilité : 3.3.0

Note
Cette fonction ne prend pas en charge les géométries non lon/lat, car elles ne sont pas prises en charge par la norme MARC21/XML (Coded Cartographic Mathematical Data).

Note
La norme MARC21/XML ne prévoit aucun moyen d'annoter le système de référence spatiale pour les données mathématiques cartographiques codées, ce qui signifie que cette information sera perdue après la conversion au format MARC21/XML.

Exemples

Conversion d'un `POINT` en MARC21/XML au format hdddmmss (par défaut)

```
                SELECT ST_AsMARC21('SRID=4326;POINT(-4.504289 54.253312)'::geometry);

                                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>
```

Conversion d'un `POLYGON` en MARC21/XML formaté en degrés décimaux

```
                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>
```

Conversion d'une GEOMETRYCOLLECTION en MARC21/XML en minutes décimales. L'ordre des géométries dans la sortie MARC21/XML correspond à leur ordre dans la collection.

```
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');

                    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>
```

Voir aussi

ST_GeomFromMARC21

7.9.3.9 ST_AsMVTGeom

ST_AsMVTGeom — Transforme une géométrie dans l'espace de coordonnées d'une tuile MVT.

Synopsis

geometry **ST_AsMVTGeom**(geometry geom, box2d bounds, integer extent=4096, integer buffer=256, boolean clip_geom=true);

Description

Transforme une géométrie dans l'espace de coordonnées d'une tuile MVT (Mapbox Vector Tile), en l'ajustant aux limites de la tuile si nécessaire. La géométrie doit être dans le système de coordonnées de la carte cible (en utilisant ST_Transform si nécessaire). Il s'agit généralement de Web Mercator (SRID:3857).

La fonction tente de préserver la validité de la géométrie et la corrige si nécessaire. Cela peut entraîner l'effondrement de la géométrie résultante à une dimension inférieure.

Les limites rectangulaires de la tuile dans l'espace de coordonnées de la carte cible doivent être fournies, afin que la géométrie puisse être transformée et coupée si nécessaire. Les limites peuvent être générées à l'aide de ST_TileEnvelope.

Cette fonction est utilisée pour convertir la géométrie dans l'espace de coordonnées de la tuile requis par ST_AsMVT.

`geom` est la géométrie à transformer, dans le système de coordonnées de la carte cible.

`bounds` est la limite rectangulaire de la tuile dans l'espace de coordonnées de la carte, sans tampon.

`extent` est la taille de l'étendue de la tuile dans l'espace de coordonnées de la tuile tel que défini par la spécification MVT. La valeur par défaut est 4096.

`buffer` est la taille du tampon dans l'espace de coordonnées de la tuile pour le découpage de la géométrie. La valeur par défaut est 256.

`clip_geom` est un booléen qui contrôle si les géométries sont découpées ou encodées telles quelles. La valeur par défaut est true.

Disponibilité : 2.4.0

Note
A partir de la version 3.0, Wagyu peut être choisi au moment de la configuration pour découper et valider les polygones MVT. Cette bibliothèque est plus rapide et produit des résultats plus corrects que la bibliothèque par défaut de GEOS, mais elle peut mettre de côté de petits polygones.

Exemples

```
SELECT ST_AsText(ST_AsMVTGeom(
  ST_GeomFromText('POLYGON ((0 0, 10 0, 10 5, 0 -5, 0 0))'),
  ST_MakeBox2D(ST_Point(0, 0), ST_Point(4096, 4096)),
  4096, 0, false));
                              st_astext
-----------------------------------------------------------------------
 MULTIPOLYGON(((5 4096,10 4091,10 4096,5 4096)),((5 4096,0 4101,0 4096,5 4096)))
```

Canonical example for a Web Mercator tile using a computed tile bounds to query and clip geometry. This assumes the data.geom column has srid of 4326.

```
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)
```

Voir aussi

ST_AsMVT, ST_TileEnvelope, PostGIS_Wagyu_Version

7.9.3.10 ST_AsMVT

ST_AsMVT — Fonction d'agrégation renvoyant une représentation MVT d'un ensemble de lignes.

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);

Description

Une fonction agrégée qui renvoie une représentation binaire Mapbox Vector Tile d'un ensemble de lignes correspondant à une couche de tuiles. Les lignes doivent contenir une colonne de géométrie qui sera encodée comme une géométrie d'élément. La géométrie doit être dans l'espace de coordonnées de la tuile et valide conformément à la spécification MVT. ST_AsMVTGeom peut être utilisé pour transformer la géométrie dans l'espace de coordonnées des tuiles. Les autres colonnes de la ligne sont encodées en tant qu'attributs de caractéristiques.

Le format Mapbox Vector Tile peut stocker des éléments avec différents ensembles d'attributs. Pour utiliser cette fonctionnalité, il convient de fournir une colonne JSONB dans les données de la ligne contenant des objets Json à un niveau de profondeur. Les clés et les valeurs des valeurs JSONB seront encodées en tant qu'attributs d'entités.

Les tuiles à couches multiples peuvent être créées en concaténant plusieurs appels à cette fonction en utilisant `||` ou `STRING_AGG`.

Important
Ne pas appeler avec un `GEOMETRYCOLLECTION` comme élément de la ligne. Cependant, vous pouvez utiliser ST_AsMVTGeom pour préparer une collection de géométrie à inclure.

`row` données de ligne avec au moins une colonne de géométrie.

`name` est le nom de la couche. La valeur par défaut est la chaîne de caractères "default".

`extent` est l'étendue de la tuile dans l'espace de l'écran tel que défini par la spécification. La valeur par défaut est 4096.

`geom_name` est le nom de la colonne géométrique dans les données de la ligne. La valeur par défaut est la première colonne géométrique. Notez que PostgreSQL, par défaut, plie automatiquement les identifiants non cités en minuscules, ce qui signifie qu'à moins que la colonne géométrique ne soit citée, par exemple `"MyMVTGeom"`, ce paramètre doit être fourni en minuscules.

`feature_id_name` est le nom de la colonne Feature ID dans les données de la ligne. S'il est NULL ou négatif, l'identifiant de la caractéristique n'est pas défini. La première colonne correspondant au nom et au type valide (smallint, integer, bigint) sera utilisée comme Feature ID, et toute colonne suivante sera ajoutée en tant que propriété. Les propriétés JSON ne sont pas prises en charge.

Amélioration : 3.0 - ajout de la prise en charge du Feature ID.

Amélioration : 2.5.0 - ajout de la prise en charge des requêtes parallèles.

Disponibilité : 2.4.0

Exemples

```
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;
```

Voir aussi

ST_AsMVTGeom, ST_TileEnvelope

7.9.3.11 ST_AsSVG

ST_AsSVG — Renvoie les données de chemin SVG pour une géométrie.

Synopsis

text **ST_AsSVG**(geometry geom, integer rel=0, integer maxdecimaldigits=15);
text **ST_AsSVG**(geography geog, integer rel=0, integer maxdecimaldigits=15);

Description

Renvoie la géométrie sous forme de données de chemin SVG (Scalar Vector Graphics). Utilisez 1 comme deuxième argument pour que les données de chemin soient implémentées en termes de déplacements relatifs, la valeur par défaut (ou 0) utilise des déplacements absolus. Le troisième argument peut être utilisé pour réduire le nombre maximal de chiffres décimaux utilisés dans la sortie (15 par défaut). Les géométries ponctuelles seront rendues sous la forme cx/cy lorsque l'argument 'rel' est 0, x/y lorsque 'rel' est 1. Les géométries multipoints sont délimitées par des virgules (","), les géométries GeometryCollection sont délimitées par des points-virgules (" ;").

Pour travailler avec les graphiques PostGIS SVG, consultez la bibliothèque pg_svg qui fournit des fonctions plpgsql pour travailler avec les résultats de ST_AsSVG.

Amélioration : 3.4.0 pour prendre en charge tous les types de courbes

Modifié : 2.0.0 pour utiliser les args par défaut et supporter les args nommés

---

Note
Disponibilité : 1.2.2. Disponibilité : 1.4.0 Modifié dans PostGIS 1.4.0 pour inclure la commande L dans le chemin absolu afin de se conformer à http://www.w3.org/TR/SVG/paths.html#PathDataBNF

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_AsSVG('POLYGON((0 0,0 1,1 1,1 0,0 0))'::geometry);

st_assvg
--------
M 0 0 L 0 -1 1 -1 1 0 Z
```

Circular string

```
SELECT ST_AsSVG( ST_GeomFromText('CIRCULARSTRING(-2 0,0 2,2 0,0 2,2 4)') );

st_assvg
--------
M -2 0 A 2 2 0 0 1 2 0 A 2 2 0 0 1 2 -4
```

Multi-curve

```
SELECT ST_AsSVG('MULTICURVE((5 5,3 5,3 3,0 3),
 CIRCULARSTRING(0 0,2 1,2 2))'::geometry, 0, 0);
 st_assvg
-----------------------------------------------
 M 5 -5 L 3 -5 3 -3 0 -3 M 0 0 A 2 2 0 0 0 2 -2
```

Multi-surface

```
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);

st_assvg
-------------------------------------------------------
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
```

7.9.3.12 ST_AsTWKB

ST_AsTWKB — Renvoie la géométrie sous forme de TWKB, diminutif de "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);

Description

Renvoie la géométrie au format TWKB (Tiny Well-Known Binary). TWKB est un format binaire compressé dont l'objectif est de minimiser la taille de la sortie.

Les paramètres relatifs aux chiffres décimaux déterminent le degré de précision stocké dans la sortie. Par défaut, les valeurs sont arrondies à l'unité la plus proche avant l'encodage. Si vous souhaitez transférer plus de précision, augmentez le nombre. Par exemple, une valeur de 1 implique que le premier chiffre à droite du point décimal sera préservé.

Les paramètres "sizes" et "bounding boxes" déterminent si les informations optionnelles concernant la longueur encodée de l'objet et les limites de l'objet sont incluses dans la sortie. Par défaut, elles ne le sont pas. Ne les activez pas à moins que votre logiciel client n'en ait l'utilité, car ils ne font qu'utiliser de l'espace (et économiser de l'espace est l'objectif de TWKB).

La forme tableau de la fonction est utilisée pour convertir une collection de géométries et d'identifiants uniques en une collection TWKB qui préserve les identifiants. Cette fonction est utile pour les clients qui souhaitent décompresser une collection et accéder ensuite à d'autres informations sur les objets qu'elle contient. Vous pouvez créer les tableaux à l'aide de la fonction array_agg. Les autres paramètres sont les mêmes que pour la forme simple de la fonction.

Note
La spécification du format est disponible en ligne à l'adresse https://github.com/TWKB/Specification, et le code permettant de créer un client JavaScript est disponible à l'adresse https://github.com/TWKB/twkb.js.

Amélioration : 2.4.0 amélioration de la mémoire et de la vitesse.

Disponibilité : 2.2.0

Exemples

```
SELECT ST_AsTWKB('LINESTRING(1 1,5 5)'::geometry);
                 st_astwkb
-------------------------------------------
\x02000202020808
```

Pour créer un objet TWKB agrégé comprenant des identifiants, il faut d'abord agréger les géométries et les objets souhaités en utilisant "array_agg()", puis appeler la fonction TWKB appropriée.

```
SELECT ST_AsTWKB(array_agg(geom), array_agg(gid)) FROM mytable;
                 st_astwkb
-------------------------------------------
\x040402020400000202
```

Voir aussi

ST_GeomFromTWKB, ST_AsBinary, ST_AsEWKB, ST_AsEWKT, ST_GeomFromText

7.9.3.13 ST_AsX3D

ST_AsX3D — Renvoie une géométrie au format X3D xml node element : ISO-IEC-19776-1.2-X3DEncodings-XML

Synopsis

text **ST_AsX3D**(geometry g1, integer maxdecimaldigits=15, integer options=0);

Description

Renvoie une géométrie sous la forme d'un élément de nœud X3D xml formaté http://www.web3d.org/standards/number/19776-1. Si `maxdecimaldigits` (précision) n'est pas spécifié, la valeur par défaut est 15.

Note
Il existe plusieurs options pour traduire les géométries PostGIS en X3D, car les types de géométrie X3D ne correspondent pas directement aux types de géométrie PostGIS, et nous avons évité certains types X3D plus récents qui pourraient constituer de meilleures correspondances, car la plupart des outils de rendu ne les prennent pas en charge actuellement. Ce sont les correspondances que nous avons choisies. N'hésitez pas à poster un ticket de bug si vous avez des idées sur l'idée ou sur la façon dont nous pouvons permettre aux gens de dénoter leurs correspondances préférées.
Voici comment nous faisons actuellement correspondre les types 2D/3D de PostGIS aux types X3D

L'argument 'options' est un champ de bits. Pour PostGIS 2.2+, il est utilisé pour indiquer si les coordonnées doivent être représentées avec le nœud géospatial X3D GeoCoordinates et si l'axe x/y doit être inversé. Par défaut, `ST_AsX3D` produit les données sous forme de base de données (long,lat ou X,Y), mais la valeur par défaut de lat/lon, y/x de X3D peut être préférée.

- 0 : X/Y dans l'ordre de la base de données (par exemple long/lat = X,Y est l'ordre standard de la base de données), valeur par défaut, et coordonnées non spatiales (juste un bon vieux Coordinate tag).
- 1 : Inverser X et Y. Si cette option est utilisée en conjonction avec l'option GeoCoordinate, la sortie sera par défaut "latitude_first" et les coordonnées seront également inversées.

- 2 : Sortie des coordonnées enGeoSpatial GeoCoordinates. Cette option génère une erreur si les géométries ne sont pas en WGS 84 long lat (srid : 4326). C'est actuellement le seul type de GeoCoordinate pris en charge. Référence aux spécifications X3D spécifiant un système de référence spatiale.. La sortie par défaut sera `GeoCoordinate geoSystem='"GD" "WE" "longitude_first"'`. Si vous préférez la sortie par défaut X3D de `GeoCoordinate geoSystem='"GD" "WE" "latitude_first"'`, utilisez `(2 + 1) = 3`

Type PostGIS	Type 2D X3D	Type 3D X3D
LINESTRING	pas encore implémenté - sera PolyLine2D	LineSet
MULTILINESTRING	pas encore implémenté - sera PolyLine2D	IndexedLineSet
MULTIPOINT	Polypoint2D	PointSet
POINT	produit les coordonnées délimitées par l'espace	produit les coordonnées délimitées par l'espace
(MULTI) POLYGON, POLYHEDRALSURFACE	Balises X3D non valides	IndexedFaceSet (les anneaux intérieurs sont actuellement édités sous la forme d'un autre jeu de faces)
TIN	TriangleSet2D (Pas encore implémenté)	IndexedTriangleSet

Note
La prise en charge de la géométrie 2D n'est pas encore terminée. Les anneaux intérieurs sont actuellement dessinés comme des polygones séparés. Nous y travaillons.

De nombreuses avancées ont lieu dans l'espace 3D, en particulier avec X3D Integration with HTML5

Il existe également une visionneuse X3D open source que vous pouvez utiliser pour visualiser les géométries rendues. Des binaires Wrl http://freewrl.sourceforge.net/ gratuits sont disponibles pour Mac, Linux et Windows. Utilisez le FreeWRL_Launcher pour visualiser les géométries.

Consultez également PostGIS minimalist X3D viewer qui utilise cette fonction et x3dDom html/js open source toolkit.

Disponibilité : 2.0.0: ISO-IEC-19776-1.2-X3DEncodings-XML

Amélioration : 2.2.0 : Prise en charge des coordonnées géographiques et de l'inversion des axes (x/y, long/lat). Voir les options pour plus de détails.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemple : Créer un document X3D entièrement fonctionnel - Cela permet de générer un cube qui peut être visualisé dans FreeWrl et d'autres visionneuses X3D.

```
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;

    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>
```

Bâtiments PostGIS

Copiez et collez le résultat de cette requête dans x3d scene viewer et cliquez sur Show

```
SELECT string_agg('<Shape
>' || ST_AsX3D(ST_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);
```

Bâtiments formés par subdivision PostGIS et extrusion

Exemple : Un octogone élevé à 3 unités et une précision décimale de 6

```
SELECT ST_AsX3D(
ST_Translate(
    ST_Force_3d(
        ST_Buffer(ST_Point(10,10),5, 'quad_segs=2')), 0,0,
    3)
  ,6) As x3dfrag;

x3dfrag
--------
<IndexedFaceSet 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>
```

Exemple : TIN

```
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;

    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>
```

Exemple : multiligne fermée (la limite d'un polygone avec des trous)

```
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;

    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 — Retourne une représentation GeoHash de la géométrie.

Synopsis

text **ST_GeoHash**(geometry geom, integer maxchars=full_precision_of_point);

Description

Calcule une représentation GeoHash d'une géométrie. Un GeoHash encode un point géographique sous forme de texte qui peut être trié et recherché en fonction du préfixe. Un GeoHash plus court est une représentation moins précise d'un point. Il peut être considéré comme une boîte qui contient le point.

Les valeurs géométriques non ponctuelles dont l'étendue n'est pas nulle peuvent également être mises en correspondance avec des codes GeoHash. La précision du code dépend de l'étendue géographique de la géométrie.

Si `maxchars` n'est pas spécifié, le code GeoHash renvoyé correspond à la plus petite cellule contenant la géométrie d'entrée. Les points renvoient un GeoHash avec 20 caractères de précision (ce qui est suffisant pour contenir la double précision de l'entrée). D'autres types géométriques peuvent renvoyer un GeoHash avec moins de précision, en fonction de l'étendue de la géométrie. Les géométries plus grandes sont représentées avec moins de précision, les plus petites avec plus de précision. La boîte déterminée par le code GeoHash contient toujours l'élément d'entrée.

Si `maxchars` est spécifié, le code GeoHash renvoyé comporte au maximum ce nombre de caractères. Il correspond à une représentation (éventuellement) de moindre précision de la géométrie d'entrée. Pour les non-points, le point de départ du calcul est le centre de la boîte de délimitation de la géométrie.

Disponibilité : 1.4.0

Note
ST_GeoHash exige que la géométrie d'entrée soit en coordonnées géographiques (lon/lat).

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_GeoHash( ST_Point(-126,48) );

   st_geohash
----------------------
 c0w3hf1s70w3hf1s70w3

SELECT ST_GeoHash( ST_Point(-126,48), 5);

 st_geohash
------------
 c0w3h

-- This line contains the point, so the GeoHash is a prefix of the point code
SELECT ST_GeoHash('LINESTRING(-126 48, -126.1 48.1)'::geometry);

 st_geohash
------------
 c0w3
```

Voir aussi

ST_GeomFromGeoHash, ST_PointFromGeoHash, ST_Box2dFromGeoHash

7.10 Opérateurs

7.10.1 Opérateurs de Bounding Box

7.10.1.1 &&

&& — Renvoi VRAI si la boite englobante 2D de A intersecte la boite englobante 2D de B.

Synopsis

boolean **&&**(geometry A , geometry B);
boolean **&&**(geography A , geography B);

Description

L'opérateur && renvoi VRAI si la boite englobante 2D de la géométrie A intersecte la boite englobante 2D de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Amélioration : 2.0.0 introduction du support des surfaces polyédriques.

Disponibilité : 1.5.0 le support de la géographie a été introduit.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
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;

 column1 | column1 | overlaps
---------+---------+----------
       1 |       3 | t
       2 |       3 | f
(2 rows)
```

Voir aussi

ST_Intersects, ST_Extent, |&>, &>, &<|, &<, ~, @

7.10.1.2 &&(geometry,box2df)

&&(geometry,box2df) — Renvoie TRUE si la boîte de délimitation 2D (en cache) d'une géométrie intersecte une boîte de délimitation 2D de précision flottante (BOX2DF).

Synopsis

boolean **&&**(geometry A , box2df B);

Description

L'opérateur `&&` renvoie `TRUE` si la boîte de délimitation 2D mise en cache de la géométrie A intersecte la boîte de délimitation 2D B, en utilisant la précision float. Cela signifie que si B est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D à précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_Point(1,1) && ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) AS overlaps;

 overlaps
----------
 t
(1 row)
```

Voir aussi

&&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.3 &&(box2df,geometry)

&&(box2df,geometry) — Renvoie `TRUE` si une boîte de délimitation 2D de précision flottante (BOX2DF) intersecte la boîte de délimitation 2D (mise en cache) d'une géométrie.

Synopsis

boolean **&&**(box2df A , geometry B);

Description

L'opérateur `&&` renvoie `TRUE` si la boîte de délimitation 2D A intersecte la boîte de délimitation 2D mise en cache de la géométrie B, en utilisant la précision float. Cela signifie que si A est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D à précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_Point(1,1) AS overlaps;

 overlaps
----------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.4 &&(box2df,box2df)

&&(box2df,box2df) — Renvoie `TRUE` si deux boîtes de délimitation 2D à précision flottante (BOX2DF) se croisent.

Synopsis

boolean **&&**(box2df A , box2df B);

Description

L'opérateur `&&` renvoie `TRUE` si deux boîtes de délimitation 2D A et B se croisent, en utilisant la précision float. Cela signifie que si A (ou B) est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D de précision flottante (BOX2DF)

Note
Cet opérateur est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_MakeBox2D(ST_Point(1,1), ST_Point ↩
    (3,3)) AS overlaps;

 overlaps
----------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,geometry), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.5 &&&

&&& — Renvoie `TRUE` si la boîte de délimitation n-D de A intersecte la boîte de délimitation n-D de B.

Synopsis

boolean **&&&**(geometry A , geometry B);

Description

L'opérateur `&&&` renvoie `TRUE` si la boîte de délimitation n-D de la géométrie A intersecte la boîte de délimitation n-D de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Disponibilité : 2.0.0

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples : LineStrings 3D

```
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;

 column1 | column1 | overlaps_3d | overlaps_2d
---------+---------+-------------+-------------
       1 |       3 | t           | t
       2 |       3 | f           | t
```

Exemples : LineStrings 3M

```
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;

 column1 | column1 | overlaps_3zm | overlaps_2d
---------+---------+--------------+-------------
       1 |       3 | t            | t
       2 |       3 | f            | t
```

Voir aussi

&&

7.10.1.6 &&&(geometry,gidx)

&&&(geometry,gidx) — Renvoie `TRUE` si la boîte de délimitation n-D (en cache) d'une géométrie intersecte une boîte de délimitation de précision flottante n-D (GIDX).

Synopsis

boolean **&&&**(geometry A , gidx B);

Description

L'opérateur `&&&` renvoie `TRUE` si la boîte de délimitation n-D mise en cache de la géométrie A intersecte la boîte de délimitation n-D B, en utilisant la précision float. Cela signifie que si B est un box3d (double précision), il sera converti en interne en une boîte de délimitation 3D de précision flottante (GIDX)

Note
Cet opérateur est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_MakePoint(1,1,1) &&& ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) AS  ↩
    overlaps;

 overlaps
----------
 t
(1 row)
```

Voir aussi

&&&(gidx,geometry), &&&(gidx,gidx)

7.10.1.7 &&&(gidx,geometry)

&&&(gidx,geometry) — Renvoie `TRUE` si une boîte de délimitation de précision flottante n-D (GIDX) intersecte la boîte de délimitation n-D (mise en cache) d'une géométrie.

Synopsis

boolean **&&&**(gidx A , geometry B);

Description

L'opérateur `&&&` renvoie `TRUE` si la boîte de délimitation n-D A intersecte la boîte de délimitation n-D mise en cache de la géométrie B, en utilisant la précision float. Cela signifie que si A est un box3d (à double précision), il sera converti en interne en une boîte de délimitation 3D à précision flottante (GIDX)

Note
Cet opérateur est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_MakePoint(1,1,1) AS  ↩
    overlaps;

 overlaps
----------
 t
(1 row)
```

Voir aussi

&&&(geometry,gidx), &&&(gidx,gidx)

7.10.1.8 &&&(gidx,gidx)

&&&(gidx,gidx) — Renvoie TRUE si deux boîtes de délimitation (GIDX) de précision flottante n-D se croisent.

Synopsis

boolean **&&&**(gidx A , gidx B);

Description

L'opérateur &&& renvoie TRUE si deux boîtes de délimitation n-D A et B se croisent, en utilisant la précision float. Cela signifie que si A (ou B) est un box3d (double précision), il sera converti en interne en une boîte de délimitation 3D de précision flottante (GIDX)

Note
Cet opérateur est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
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;

 overlaps
----------
 t
(1 row)
```

Voir aussi

&&&(geometry,gidx), &&&(gidx,geometry)

7.10.1.9 &<

&< — Renvoie TRUE si la boîte englobante de A chevauche ou est à gauche de celle de B.

Synopsis

boolean **&<**(geometry A , geometry B);

Description

L'opérateur `&<` renvoie `TRUE` si la boîte de délimitation de la géométrie A chevauche ou est à gauche de la boîte de délimitation de la géométrie B, ou plus exactement, chevauche ou n'est PAS à droite de la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | overleft
---------+---------+----------
           1 |       2 | f
           1 |       3 | f
           1 |       4 | t
(3 rows)
```

Voir aussi

&&, |&>, &>, &<|

7.10.1.10 &<|

&<| — Renvoie `TRUE` si la boîte englobante de A chevauche ou est inférieure à celle de B.

Synopsis

boolean **&<|**(geometry A , geometry B);

Description

L'opérateur `&<|` renvoie `TRUE` si la boîte de délimitation de la géométrie A chevauche ou est en dessous de la boîte de délimitation de la géométrie B, ou plus exactement, chevauche ou n'est PAS au-dessus de la boîte de délimitation de la géométrie B.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | overbelow
---------+---------+-----------
           1 |       2 | f
           1 |       3 | t
           1 |       4 | t
(3 rows)
```

Voir aussi

&&, |&>, &>, &<

7.10.1.11 &>

&> — Renvoie TRUE si la boîte de délimitation de A chevauche ou est à droite de celle de B.

Synopsis

boolean **&>**(geometry A , geometry B);

Description

L'opérateur &> renvoie TRUE si la boîte de délimitation de la géométrie A chevauche ou est à droite de la boîte de délimitation de la géométrie B, ou plus exactement, chevauche ou n'est PAS à gauche de la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | overright
---------+---------+-----------
           1 |       2 | t
           1 |       3 | t
           1 |       4 | f
(3 rows)
```

Voir aussi

&&, |&>, &<|, &<

7.10.1.12 <<

<< — Renvoie TRUE si la boîte de délimitation de A est strictement à gauche de celle de B.

Synopsis

boolean **<<**(geometry A , geometry B);

Description

L'opérateur << renvoie TRUE si la boîte de délimitation de la géométrie A est strictement à gauche de la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | left
---------+---------+------
           1 |       2 | f
           1 |       3 | t
           1 |       4 | t
(3 rows)
```

Voir aussi

>>, |>>, <<|

7.10.1.13 <<|

<<| — Renvoie TRUE si la boîte de délimitation de A est strictement inférieure à celle de B.

Synopsis

boolean **<<|**(geometry A , geometry B);

Description

L'opérateur <<| renvoie TRUE si la boîte de délimitation de la géométrie A est strictement inférieure à la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | below
---------+---------+-------
           1 |       2 | t
           1 |       3 | f
           1 |       4 | f
(3 rows)
```

Voir aussi

<<, >>, |>>

7.10.1.14 =

= — Renvoie TRUE si les coordonnées et l'ordre des coordonnées de la géométrie/géographie A sont les mêmes que les coordonnées et l'ordre des coordonnées de la géométrie/géographie B.

Synopsis

boolean **=**(geometry A , geometry B);
boolean **=**(geography A , geography B);

Description

L'opérateur = renvoie `TRUE` si les coordonnées et l'ordre des coordonnées de la géométrie/géographie A sont les mêmes que les coordonnées et l'ordre des coordonnées de la géométrie/géographie B. PostgreSQL utilise les opérateurs =, <, et > définis pour les géométries pour effectuer des classements internes et des comparaisons de géométries (c'est-à-dire dans une clause GROUP BY ou ORDER BY).

Note
Seules les géométries/géographies qui sont exactement égales à tous égards, avec les mêmes coordonnées, dans le même ordre, sont considérées comme égales par cet opérateur. Pour une "égalité spatiale", qui ignore des choses comme l'ordre des coordonnées, et peut détecter des caractéristiques qui couvrent la même zone spatiale avec des représentations différentes, utilisez ST_OrderingEquals ou ST_Equals

Caution
Cet opérande n'utilisera PAS les index qui peuvent être disponibles sur les géométries. Pour un test d'égalité exact assisté par index, combinez = avec &&.

Modifié : 2.4.0, dans les versions précédentes, il s'agissait d'une égalité de boîte de délimitation et non d'une égalité géométrique. Si vous avez besoin d'une égalité de boîte de délimitation, utilisez ~= à la place.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT 'LINESTRING(0 0, 0 1, 1 0)'::geometry = 'LINESTRING(1 1, 0 0)'::geometry;
 ?column?
----------
 f
(1 row)

SELECT ST_AsText(column1)
FROM ( VALUES
        ('LINESTRING(0 0, 1 1)'::geometry),
        ('LINESTRING(1 1, 0 0)'::geometry)) AS foo;
          st_astext
---------------------
 LINESTRING(0 0,1 1)
 LINESTRING(1 1,0 0)
(2 rows)

-- Note: the GROUP BY uses the "=" to compare for geometry equivalency.
SELECT ST_AsText(column1)
FROM ( VALUES
        ('LINESTRING(0 0, 1 1)'::geometry),
        ('LINESTRING(1 1, 0 0)'::geometry)) AS foo
GROUP BY column1;
      st_astext
---------------------
 LINESTRING(0 0,1 1)
 LINESTRING(1 1,0 0)
(2 rows)

-- In versions prior to 2.0, this used to return true --
```

```
 SELECT ST_GeomFromText('POINT(1707296.37 4820536.77)') =
        ST_GeomFromText('POINT(1707296.27 4820536.87)') As pt_intersect;

--pt_intersect --
f
```

Voir aussi

ST_Equals, ST_OrderingEquals, ~=

7.10.1.15 >>

>> — Renvoie `TRUE` si la boîte de délimitation de A est strictement à droite de celle de B.

Synopsis

boolean **>>**(geometry A , geometry B);

Description

L'opérateur >> renvoie `TRUE` si la boîte de délimitation de la géométrie A est strictement à droite de la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | right
---------+---------+-------
           1 |       2 | t
           1 |       3 | f
           1 |       4 | f
(3 rows)
```

Voir aussi

<<, |>>, <<|

7.10.1.16 @

@ — Renvoie TRUE si la boîte de délimitation de A est contenue par celle de B.

Synopsis

boolean @(geometry A , geometry B);

Description

L'opérateur @ renvoie TRUE si la boîte de délimitation de la géométrie A est complètement contenue par la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | contained
---------+---------+-----------
           1 |       2 | t
           1 |       3 | f
           1 |       4 | t
(3 rows)
```

Voir aussi

~, &&

7.10.1.17 @(geometry,box2df)

@(geometry,box2df) — Renvoie TRUE si la boîte de délimitation 2D d'une géométrie est contenue dans une boîte de délimitation 2D à précision flottante (BOX2DF).

Synopsis

boolean @(geometry A , box2df B);

Description

L'opérateur `@` renvoie `TRUE` si la boîte de délimitation 2D de la géométrie A est contenue dans la boîte de délimitation 2D B, en utilisant la précision float. Cela signifie que si B est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D de précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_Buffer(ST_GeomFromText('POINT(2 2)'), 1) @ ST_MakeBox2D(ST_Point(0,0), ST_Point ↩
    (5,5)) AS is_contained;

 is_contained
--------------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.18 @(box2df,geometry)

@(box2df,geometry) — Renvoie `TRUE` si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans la boîte de délimitation 2D d'une géométrie.

Synopsis

boolean @(box2df A , geometry B);

Description

L'opérateur `@` renvoie `TRUE` si la boîte de délimitation 2D A est contenue dans la boîte de délimitation 2D de la géométrie B, en utilisant la précision float. Cela signifie que si B est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D de précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_MakeBox2D(ST_Point(2,2), ST_Point(3,3)) @ ST_Buffer(ST_GeomFromText('POINT(1 1)') ↩
    , 10) AS is_contained;

 is_contained
--------------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,box2df)

7.10.1.19 @(box2df,box2df)

@(box2df,box2df) — Renvoie `TRUE` si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans une autre boîte de délimitation de précision flottante 2D.

Synopsis

boolean @(box2df A , box2df B);

Description

L'opérateur `@` renvoie `TRUE` si la boîte de délimitation 2D A est contenue dans la boîte de délimitation 2D B, en utilisant la précision float. Cela signifie que si A (ou B) est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D à précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_MakeBox2D(ST_Point(2,2), ST_Point(3,3)) @ ST_MakeBox2D(ST_Point(0,0), ST_Point ↩
    (5,5)) AS is_contained;

 is_contained
--------------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry)

7.10.1.20 |&>

|&> — Renvoie TRUE si la boîte de délimitation de A chevauche ou est au-dessus de celle de B.

Synopsis

boolean **|&>**(geometry A , geometry B);

Description

L'opérateur |&> renvoie TRUE si la boîte de délimitation de la géométrie A chevauche ou est au-dessus de la boîte de délimitation de la géométrie B, ou plus exactement, chevauche ou n'est PAS au-dessous de la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | overabove
---------+---------+-----------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Voir aussi

&&, &>, &<|, &<

7.10.1.21 |>>

|>> — Renvoie TRUE si la boîte de délimitation de A est strictement au-dessus de celle de B.

Synopsis

boolean **|>>**(geometry A , geometry B);

Description

L'opérateur |>> renvoie TRUE si la boîte de délimitation de la géométrie A est strictement au-dessus de la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | above
---------+---------+-------
           1 |       2 | t
           1 |       3 | f
           1 |       4 | f
(3 rows)
```

Voir aussi

<<, >>, <<|

7.10.1.22 ~

~ — Renvoie TRUE si la boîte de délimitation de A contient celle de B.

Synopsis

boolean **~**(geometry A , geometry B);

Description

L'opérateur ~ renvoie TRUE si la boîte de délimitation de la géométrie A contient complètement la boîte de délimitation de la géométrie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 column1 | column1 | contains
---------+---------+----------
           1 |       2 | f
           1 |       3 | t
           1 |       4 | t
(3 rows)
```

Voir aussi

@, &&

7.10.1.23 ~(geometry,box2df)

~(geometry,box2df) — Renvoie TRUE si la boîte de délimitation 2D d'une géométrie contient une boîte de délimitation de précision flottante 2D (GIDX).

Synopsis

boolean ~(geometry A , box2df B);

Description

L'opérateur ~ renvoie TRUE si la boîte de délimitation 2D d'une géométrie A contient la boîte de délimitation 2D B, en utilisant la précision float. Cela signifie que si B est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D de précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_Buffer(ST_GeomFromText('POINT(1 1)'), 10) ~ ST_MakeBox2D(ST_Point(0,0), ST_Point ↩
    (2,2)) AS contains;

 contains
----------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.24 ~(box2df,geometry)

~(box2df,geometry) — Renvoie `TRUE` si une boîte de délimitation de précision flottante 2D (BOX2DF) contient la boîte de délimitation 2D d'une géométrie.

Synopsis

boolean ~(box2df A , geometry B);

Description

L'opérateur ~ renvoie `TRUE` si la boîte de délimitation 2D A contient la boîte de délimitation de la géométrie B, en utilisant la précision float. Cela signifie que si A est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D de précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(5,5)) ~ ST_Buffer(ST_GeomFromText('POINT(2 2)') ↩
    , 1) AS contains;

 contains
----------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.25 ~(box2df,box2df)

~(box2df,box2df) — Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient une autre boîte de délimitation de précision flottante 2D (BOX2DF).

Synopsis

boolean ~(box2df A , box2df B);

Description

L'opérateur ~ renvoie TRUE si la boîte de délimitation 2D A contient la boîte de délimitation 2D B, en utilisant la précision float. Cela signifie que si A est un box2d (double précision), il sera converti en interne en une boîte de délimitation 2D de précision flottante (BOX2DF)

Note
Cet opérande est destiné à être utilisé en interne par les index BRIN, plus que par les utilisateurs.

Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(5,5)) ~ ST_MakeBox2D(ST_Point(2,2), ST_Point ↩
    (3,3)) AS contains;

 contains
----------
 t
(1 row)
```

Voir aussi

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.26 ~=

~= — Renvoie TRUE si la boîte de délimitation de A est la même que celle de B.

Synopsis

boolean **~=**(geometry A , geometry B);

Description

L'opérateur ~= renvoie TRUE si la boîte de délimitation de la géométrie/géographie A est la même que la boîte de délimitation de la géométrie/géographie B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Disponibilité : 1.5.0 comportement changé

Cette fonction prend en charge les surfaces Polyhedral.

Warning
Cet opérateur a changé de comportement dans PostGIS 1.5, passant de la vérification de l'égalité géométrique réelle à la vérification de l'égalité du rectangle de délimitation. Pour compliquer les choses, le comportement de votre base de données dépend également du type de mise à niveau (hard ou soft) que vous avez effectué. Pour savoir quel est le comportement de votre base de données, vous pouvez exécuter la requête ci-dessous. Pour vérifier l'égalité réelle, utilisez ST_OrderingEquals ou ST_Equals.

Exemples

```
select 'LINESTRING(0 0, 1 1)'::geometry ~= 'LINESTRING(0 1, 1 0)'::geometry as equality;
 equality   |
------------------+
          t     |
```

Voir aussi

ST_Equals, ST_OrderingEquals, =

7.10.2 Opérateurs de distance

7.10.2.1 <->

<-> — Renvoie la distance en 2D entre A et B.

Synopsis

double precision **<->**(geometry A , geometry B);
double precision **<->**(geography A , geography B);

Description

L'opérateur <-> renvoie la distance 2D entre deux géométries. Utilisé dans la clause "ORDER BY" fournit des ensembles de résultats de plus proches voisins assistés par index. Pour PostgreSQL inférieur à 9.5, donne uniquement la distance centroïde des boîtes englobantes et pour PostgreSQL 9.5+, fait une vraie recherche de distance KNN donnant la vraie distance entre les géométries, et la sphère de distance pour les géographies.

Note
Cet opérande utilise les index 2D GiST qui peuvent être disponibles sur les géométries. Il est différent des autres opérateurs qui utilisent des index spatiaux en ce sens que l'index spatial n'est utilisé que lorsque l'opérateur est dans la clause ORDER BY.

Note
L'index n'intervient que si l'une des géométries est une constante (pas dans une sous-requête/cte). Par exemple, 'SRID=3005;POINT(1011102 450541)'::geometry au lieu de a.geom

Reportez-vous à l'atelier PostGIS : La recherche du plus proche voisin pour un exemple détaillé.

Amélioré : 2.2.0 -- Comportement KNN ("K nearest neighbor") réel pour la géométrie et la géographie pour PostgreSQL 9.5+. Note : pour la géographie, KNN est basé sur la sphère plutôt que sur le sphéroïde. Pour PostgreSQL 9.4 et moins, le support de la géographie est nouveau mais ne supporte que le centroïde de la boîte de délimitation.

Modifié : 2.2.0 -- Pour les utilisateurs de PostgreSQL 9.5, l'ancienne syntaxe Hybrid peut être plus lente, donc vous voudrez vous débarrasser de ce hack si vous exécutez votre code uniquement sur PostGIS 2.2+ 9.5+. Voir les exemples ci-dessous.

Disponibilité : 2.0.0 -- Le KNN fournit des voisins les plus proches basés sur les distances entre les centroïdes géométriques au lieu des distances réelles. Résultats exacts pour les points, inexacts pour tous les autres types. Disponible pour PostgreSQL 9.1+

Exemples

```
SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d,edabbr, vaabbr
FROM va2005
ORDER BY d limit 10;

        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)
```

Puis la réponse brute KNN :

```
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;

        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)
```

Si vous exécutez "EXPLAIN ANALYZE" sur les deux requêtes, vous constaterez une amélioration des performances pour la seconde.

Pour les utilisateurs utilisant PostgreSQL < 9.5, utilisez une requête hybride pour trouver les vrais plus proches voisins. D'abord une requête CTE utilisant le KNN assisté par index, puis une requête exacte pour obtenir l'ordre correct :

```
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;

        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)
```

Voir aussi

ST_DWithin, ST_Distance, <#>

7.10.2.2 |=|

|=| — Renvoie la distance entre les trajectoires A et B à leur point d'approche le plus proche.

Synopsis

double precision **|=|**(geometry A , geometry B);

Description

L'opérateur |=| renvoie la distance 3D entre deux trajectoires (Voir ST_IsValidTrajectory). C'est la même chose que ST_DistanceCPA mais en tant qu'opérateur, il peut être utilisé pour effectuer des recherches du plus proche voisin en utilisant un index à N dimensions (nécessite PostgreSQL 9.5.0 ou plus).

Note
Cet opérande utilisera les index ND GiST qui peuvent être disponibles sur les géométries. Il est différent des autres opérateurs qui utilisent des index spatiaux en ce sens que l'index spatial n'est utilisé que lorsque l'opérateur est dans la clause ORDER BY.

Note
L'index n'intervient que si l'une des géométries est une constante (pas dans une sous-requête/cte). Par exemple, 'SRID=3005;LINESTRINGM(0 0 0,0 0 1)'::geometry au lieu de a.geom

Disponibilité : 2.2.0. La prise en charge des index est disponible uniquement pour PostgreSQL 9.5+

Exemples

```
-- Save a literal query trajectory in a psql variable...
\set qt 'ST_AddMeasure(ST_MakeLine(ST_MakePointM(-350,300,0),ST_MakePointM(-410,490,0)) ↩
    ,10,20)'
-- Run the query !
SELECT track_id, dist FROM (
  SELECT track_id, ST_DistanceCPA(tr,:qt) dist
  FROM trajectories
  ORDER BY tr |=| :qt
  LIMIT 5
) foo;
 track_id        dist
----------+------------------
      395 | 0.576496831518066
      380 |  5.06797130410151
      390 |  7.72262293958322
      385 |   9.8004461358071
      405 |  10.9534397988433
(5 rows)
```

Voir aussi

ST_DistanceCPA, ST_ClosestPointOfApproach, ST_IsValidTrajectory

7.10.2.3 <#>

<#> — Renvoie la distance 2D entre les boîtes de délimitation A et B.

Synopsis

double precision **<#>**(geometry A , geometry B);

Description

L'opérateur <#> renvoie la distance entre deux boîtes de délimitation en virgule flottante, en les lisant éventuellement à partir d'un index spatial (PostgreSQL 9.1+ requis). Utile pour effectuer un ordonnancement par distance du plus proche voisin **approximate**.

Note
Cet opérande utilisera tous les index qui peuvent être disponibles sur les géométries. Il est différent des autres opérateurs qui utilisent des index spatiaux en ce sens que l'index spatial n'est utilisé que lorsque l'opérateur est dans la clause ORDER BY.

Note
L'index n'intervient que si l'une des géométries est une constante, par exemple ORDER BY (ST_GeomFromText('POINT(1 2)') <#> geom) au lieu de g1.geom <#>.

Disponibilité : 2.0.0 -- KNN disponible uniquement pour PostgreSQL 9.1+

Exemples

```
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;

   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)
```

Voir aussi

ST_DWithin, ST_Distance, <->

7.10.2.4 <<->>

<<->> — Renvoie la distance n-D entre les géométries A et B ou les boîtes englobantes

Synopsis

double precision **<<->>**(geometry A , geometry B);

Description

L'opérateur <<−>> renvoie la distance n-D (euclidienne) entre les centroïdes des boîtes englobantes de deux géométries. Utile pour effectuer le classement par distance du plus proche voisin **approximate**.

Note
Cet opérande utilise les index GiST n-D qui peuvent être disponibles sur les géométries. Il est différent des autres opérateurs qui utilisent des index spatiaux en ce sens que l'index spatial n'est utilisé que lorsque l'opérateur est dans la clause ORDER BY.

Note
L'index n'intervient que si l'une des géométries est une constante (pas dans une sous-requête/cte). Par exemple, 'SRID=3005;POINT(1011102 450541)'::geometry au lieu de a.geom

Disponibilité : 2.2.0 -- KNN disponible uniquement pour PostgreSQL 9.1+

Voir aussi

<->

7.11 Relations spatiales

7.11.1 Relations topologiques

7.11.1.1 ST_3DIntersects

ST_3DIntersects — Teste si deux géométries se croisent dans l'espace en 3D - uniquement pour les points, les lignes, les polygones, les surfaces polyédriques (aire)

Synopsis

boolean **ST_3DIntersects**(geometry geomA , geometry geomB);

Description

Overlaps, Touches, Within impliquent tous une intersection spatiale. Si l'un des éléments susmentionnés renvoie un résultat positif, les géométries se recoupent également dans l'espace. Disjoint implique faux pour l'intersection spatiale.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries.

Note
En raison des défaillances de la robustesse flottante, les géométries ne se croisent pas toujours comme on s'y attendrait après le traitement géométrique. Par exemple, le point le plus proche d'une géométrie sur une ligne peut ne pas se trouver sur la ligne. Pour ce type de problèmes, lorsqu'une distance d'un centimètre est considérée comme une intersection, utilisez ST_3DDWithin.

Modifié : 3.0.0 SFCGAL backend supprimé, GEOS backend supporte les TINs.

Disponibilité : 2.0.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Exemples de géométrie

```
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;
 st_3dintersects | st_intersects
-----------------+---------------
 f               | t
(1 row)
```

Exemples TIN

```
SELECT ST_3DIntersects('TIN(((0 0 0,1 0 0,0 1 0,0 0 0)))'::geometry, 'POINT(.1 .1 0)':: ↩
    geometry);
 st_3dintersects
-----------------
 t
```

Voir aussi

ST_3DDWithin, ST_Intersects

7.11.1.2 ST_Contains

ST_Contains — Tests si chaque point de B est situé dans A, et que leurs intérieurs ont un point commun

Synopsis

boolean **ST_Contains**(geometry geomA, geometry geomB);

Description

Renvoie TRUE si la géométrie A contient la géométrie B. A contient B si et seulement si tous les points de B se trouvent à l'intérieur (c'est-à-dire à l'intérieur ou à la limite) de A (ou, de manière équivalente, si aucun point de B ne se trouve à l'extérieur de A), et si les intérieurs de A et de B ont au moins un point en commun.

En termes mathématiques : $ST\_Contains(A, B) \Leftrightarrow (A \bigcap B = B) \wedge (Int(A) \bigcap Int(B) \neq \varnothing)$.

La relation contains est réflexive : toute géométrie se contient elle-même. (En revanche, dans le prédicat ST_ContainsProperly, une géométrie ne se contient*pas* elle-même correctement). La relation est antisymétrique : si `ST_Contains(A,B) = true` et `ST_Contains(B,A) = true`, alors les deux géométries doivent être topologiquement égales (`ST_Equals(A,B) = true`).

ST_Contains est le contraire de ST_Within. Ainsi, `ST_Contains(A,B) = ST_Within(B,A)`.

Note
Comme les intérieurs doivent avoir un point commun, une subtilité de la définition est que les polygones et les lignes ne contiennent *pas* de lignes et de points se trouvant entièrement dans leur limite. Pour plus de détails, voir Subtleties of OGC Covers, Contains, Within. Le prédicat ST_Covers fournit une relation plus inclusive.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries. Pour éviter l'utilisation d'un index, utilisez la fonction `_ST_Contains`.

Effectué par le module GEOS

Amélioré : 2.3.0 Amélioration du court-circuit PIP étendu à la prise en charge des multipoints avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones.

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Important
N'utilisez pas cette fonction avec des géométries non valides. Vous obtiendrez des résultats inattendus.

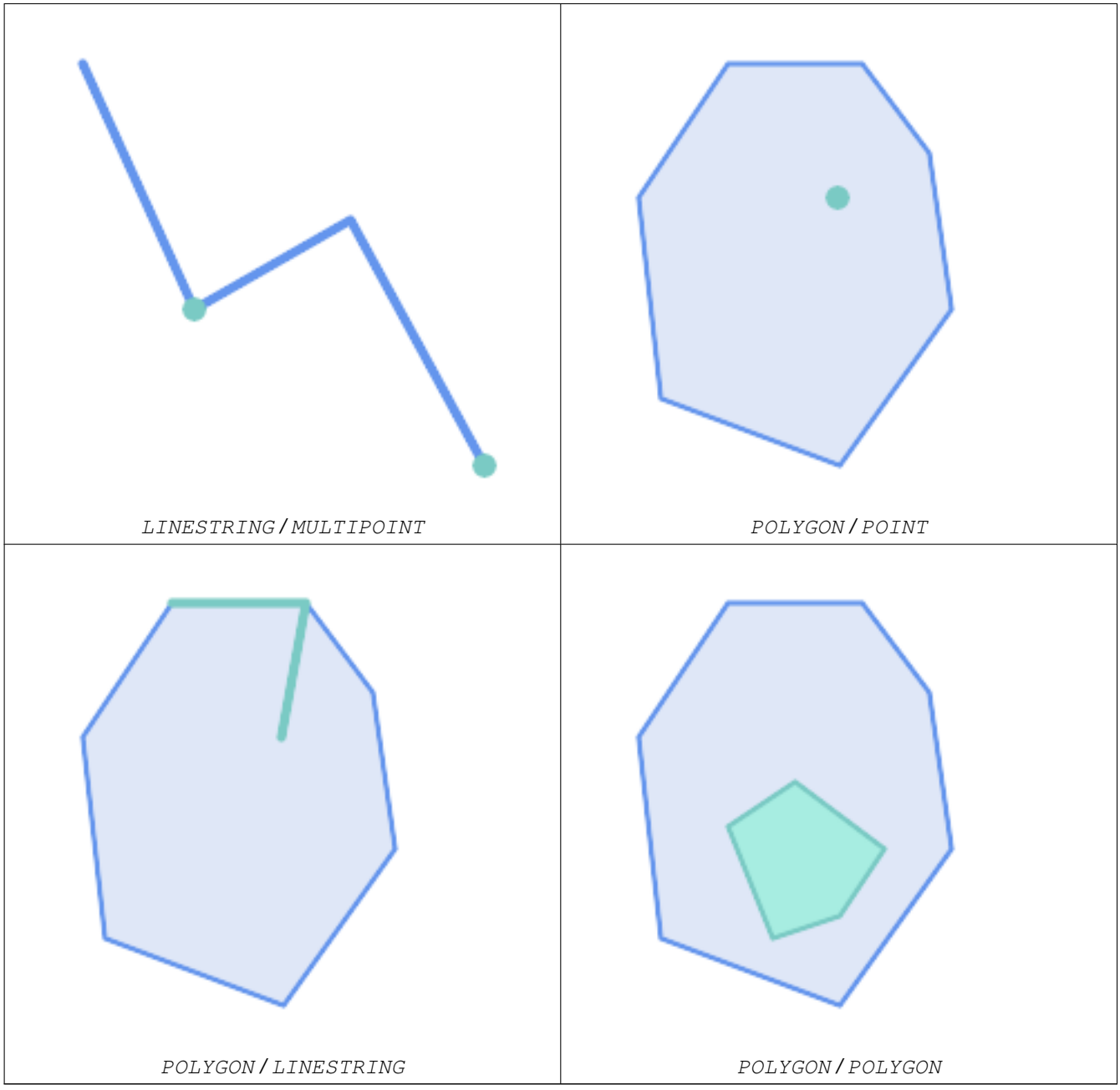
NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 // s2.1.13.3 - identique à within(geometry B, geometry A)

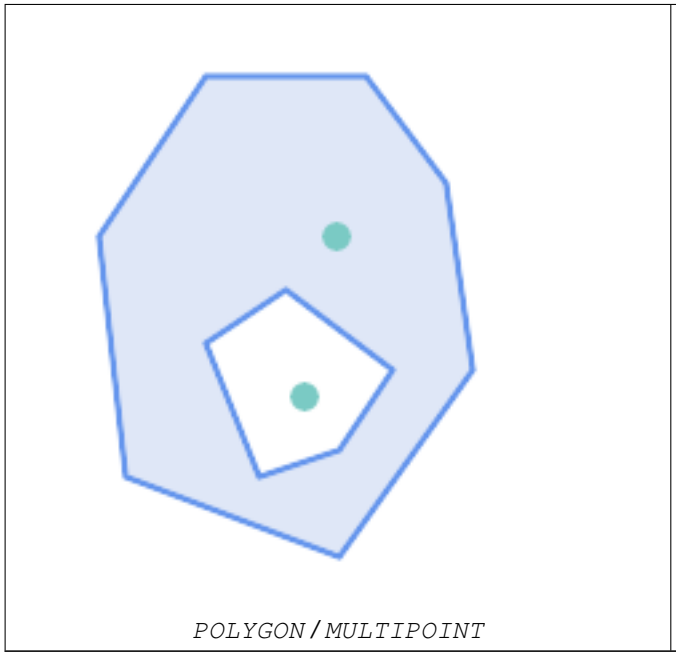
Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.31

Exemples

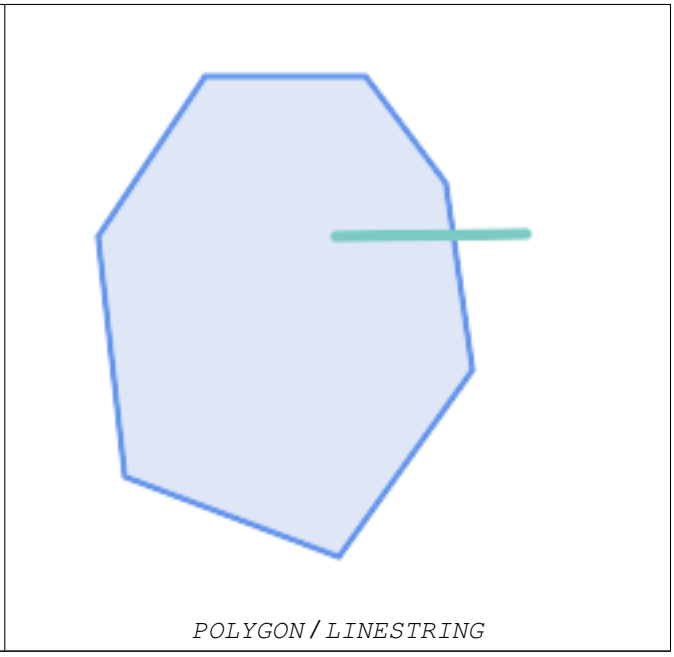
`ST_Contains` renvoie `TRUE` dans les situations suivantes :



La `ST_Contains` renvoie `FALSE` dans les situations suivantes :

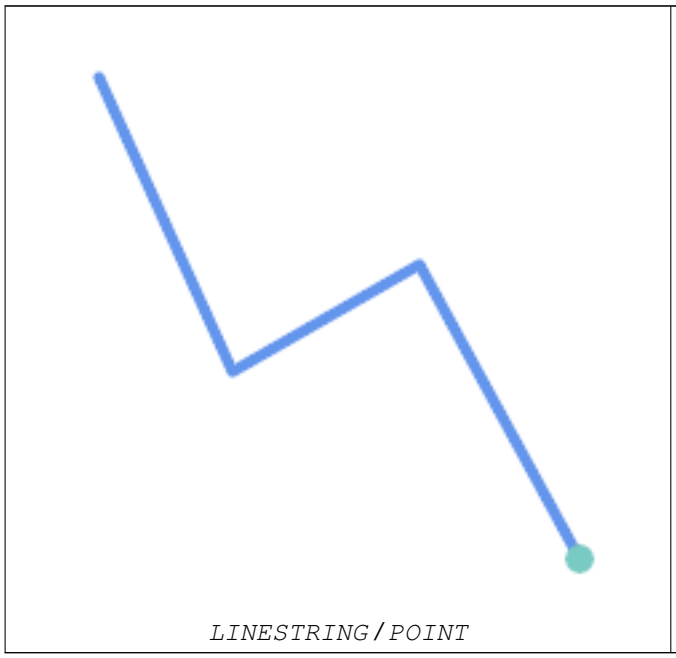


POLYGON/MULTIPOINT

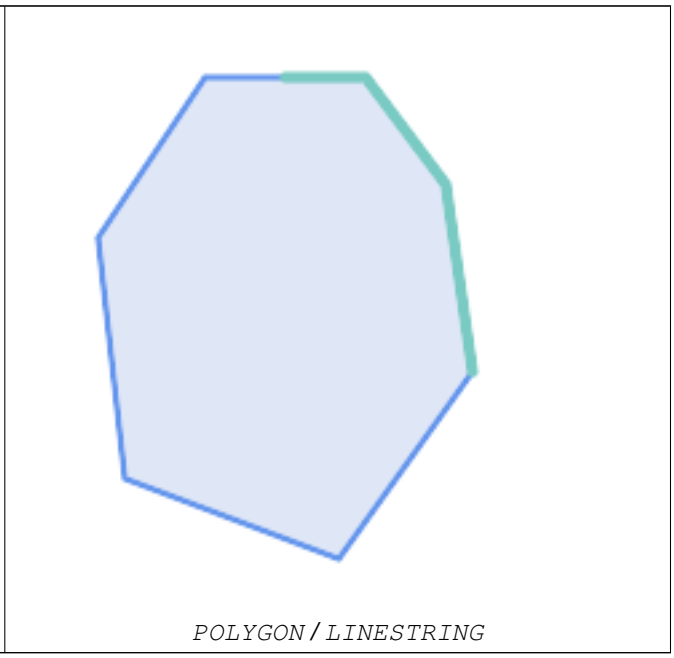


POLYGON/LINESTRING

En raison de la condition d'intersection intérieure, la `ST_Contains` renvoie `FALSE` dans les situations suivantes (alors que la `ST_Covers` renvoie `TRUE`) :



LINESTRING/POINT



POLYGON/LINESTRING

```
-- A circle within a circle
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 (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
       ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;
```

```
-- Result
  smallcontainsbig | bigcontainssmall | bigcontainsunion | bigisunion | bigcoversexterior | ↩
      bigcontainsexterior
-----------------+------------------+------------------+------------+-------------------+----------

 f                | t                | t                | t          | t         | f

-- Example demonstrating difference between contains and contains properly
SELECT ST_GeometryType(geomA) As geomtype, ST_Contains(geomA,geomA) AS acontainsa,  ↩
    ST_ContainsProperly(geomA, geomA) AS acontainspropa,
   ST_Contains(geomA, ST_Boundary(geomA)) As acontainsba, ST_ContainsProperly(geomA,  ↩
       ST_Boundary(geomA)) 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(geomA);

  geomtype    | acontainsa | acontainspropa | acontainsba | acontainspropba
--------------+------------+----------------+-------------+-----------------
ST_Polygon    | t          | f              | f           | f
ST_LineString | t          | f              | f           | f
ST_Point      | t          | t              | f           | f
```

Voir aussi

ST_Boundary, ST_ContainsProperly, ST_Covers, ST_CoveredBy, ST_Equals, ST_Within

7.11.1.3 ST_ContainsProperly

ST_ContainsProperly — Tests si chaque point de B se trouve à l'intérieur de A

Synopsis

boolean **ST_ContainsProperly**(geometry geomA, geometry geomB);

Description

Retourne `vrai` si chaque point de B se trouve à l'intérieur de A (ou, de façon équivalente, si aucun point de B ne se trouve à la limite ou à l'extérieur de A).

En termes mathématiques : *ST_ContainsProperly(A, B)* $\Leftrightarrow$ *Int(A)* $\bigcap$ *B* = *B*

A contient B correctement si la matrice d'intersection DE-9IM pour les deux géométries correspond à [T**FF*FF*]

A ne se contient pas proprement, mais se contient.

Ce prédicat peut être utilisé pour calculer les intersections d'un ensemble de géométries avec une grande géométrie polygonale. L'intersection étant une opération assez lente, il peut être plus efficace d'utiliser containsProperly pour filtrer les géométries de test qui se trouvent entièrement à l'intérieur de la zone. Dans ce cas, on sait a priori que l'intersection correspond exactement à la géométrie d'essai originale.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries. Pour éviter l'utilisation d'un index, utilisez la fonction `_ST_ContainsProperly`.

Note
L'avantage de ce prédicat par rapport à ST_Contains et ST_Intersects est qu'il peut être calculé plus efficacement, sans qu'il soit nécessaire de calculer la topologie en des points individuels.

Effectué par le module GEOS.

Disponibilité : 1.4.0

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Important
N'utilisez pas cette fonction avec des géométries non valides. Vous obtiendrez des résultats inattendus.

Exemples

```
--a circle within a circle
  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 (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
  ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;
  --Result
  smallcontainspropbig | bigcontainspropsmall | bigcontainspropunion | bigisunion |  ↩
      bigcoversexterior | bigcontainsexterior
------------------+------------------+-----------------+------------+------------------+----------

 f                    | t                   | f                    | t          | t  ↩
                     | f

 --example demonstrating difference between contains and contains properly
 SELECT ST_GeometryType(geomA) As geomtype, ST_Contains(geomA,geomA) AS acontainsa,  ↩
     ST_ContainsProperly(geomA, geomA) AS acontainspropa,
 ST_Contains(geomA, ST_Boundary(geomA)) As acontainsba, ST_ContainsProperly(geomA,  ↩
     ST_Boundary(geomA)) 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(geomA);

  geomtype    | acontainsa | acontainspropa | acontainsba | acontainspropba
--------------+------------+----------------+-------------+-----------------
ST_Polygon    | t          | f              | f           | f
ST_LineString | t          | f              | f           | f
ST_Point      | t          | t              | f           | f
```

Voir aussi

ST_GeometryType, ST_Boundary, ST_Contains, ST_Covers, ST_CoveredBy, ST_Equals, ST_Relate, ST_Within

7.11.1.4 ST_CoveredBy

ST_CoveredBy — Tests si chaque point de A se trouve dans B

Synopsis

boolean **ST_CoveredBy**(geometry geomA, geometry geomB);
boolean **ST_CoveredBy**(geography geogA, geography geogB);

Description

Renvoie `true` si chaque point de la geometry/geography A se trouve à l'intérieur (c'est-à-dire coupe l'intérieur ou la limite) de la geometry/geography B. De manière équivalente, teste qu'aucun point de A ne se trouve à l'extérieur (dans l'extérieur) de B.

En termes mathématiques : *ST_CoveredBy(A, B)* $\Leftrightarrow A \bigcap B = A$

ST_CoveredBy est le contraire de ST_Covers. Ainsi, `ST_CoveredBy(A,B) = ST_Covers(B,A)`.

En général, cette fonction devrait être utilisée à la place de ST_Within, car elle a une définition plus simple qui n'a pas la particularité que "les limites ne sont pas à l'intérieur de leur géométrie".

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries. Pour éviter l'utilisation d'un index, utilisez la fonction `_ST_CoveredBy`.

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Important
N'utilisez pas cette fonction avec des géométries non valides. Vous obtiendrez des résultats inattendus.

Effectué par le module GEOS

Disponibilité : 1.2.2

NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Il ne s'agit pas d'une norme de l'OGC, mais Oracle l'a également.

Exemples

```
--a circle coveredby a circle
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 exeriorwithinbig
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
  ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;
  --Result
 smallinsmall | smallcoveredbybig | exteriorcoveredbybig | exeriorwithinbig
--------------+-------------------+----------------------+------------------
 t            | t                 | t                    | f
(1 row)
```

Voir aussi

ST_Contains, ST_Covers, ST_ExteriorRing, ST_Within

7.11.1.5 ST_Covers

ST_Covers — Tests si chaque point de B est situé dans A

Synopsis

boolean **ST_Covers**(geometry geomA, geometry geomB);
boolean **ST_Covers**(geography geogpolyA, geography geogpointB);

Description

Renvoie `true` si chaque point de la geometry/geography B se trouve à l'intérieur (c'est-à-dire qu'il coupe l'intérieur ou la limite) de la geometry/geography A. De façon équivalente, teste qu'aucun point de B ne se trouve à l'extérieur (dans l'extérieur) de A.

En termes mathématiques : *ST_Covers(A, B)* $\Leftrightarrow A \bigcap B = B$

ST_Covers est le contraire de ST_CoveredBy. Ainsi, `ST_Covers(A,B) = ST_CoveredBy(B,A)`.

En général, cette fonction devrait être utilisée à la place de ST_Contains, car elle a une définition plus simple qui n'a pas la particularité que "les géométries ne contiennent pas leur frontière".

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries. Pour éviter l'utilisation d'un index, utilisez la fonction `_ST_Covers`.

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Important
N'utilisez pas cette fonction avec des géométries non valides. Vous obtiendrez des résultats inattendus.

Effectué par le module GEOS

Amélioration : 2.4.0 Ajout de la prise en charge des polygones dans les polygones et des lignes dans les polygones pour le type geography

Amélioration : 2.3.0 Amélioration du court-circuit PIP pour la géométrie étendue à la prise en charge des multipoints avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones.

Disponibilité : 1.5 - le support de la geography a été introduit.

Disponibilité : 1.2.2

NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Il ne s'agit pas d'une norme de l'OGC, mais Oracle l'a également.

Exemples

Exemple géométrique

```
--a circle covering a circle
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 (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
  ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;
  --Result
 smallinsmall | smallcoversbig | bigcoversexterior | bigcontainsexterior
--------------+----------------+-------------------+---------------------
 t            | f              | t                 | f
(1 row)
```

Exemple géographique

```
-- a point with a 300 meter buffer compared to a point, a point and its 10 meter buffer
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(ST_GeogFromText('SRID=4326;POINT(-99.327 31.4821)'), 300) As  ↩
      geog_poly,
        ST_GeogFromText('SRID=4326;POINT(-99.33 31.483)') As geog_pt ) As foo;

 poly_covers_pt | buff_10m_covers_cent
----------------+----------------------
 f              | t
```

Voir aussi

ST_Contains, ST_CoveredBy, ST_Within

7.11.1.6 ST_Crosses

ST_Crosses — Teste si deux géométries ont en commun certains points intérieurs, mais pas tous

Synopsis

boolean **ST_Crosses**(geometry g1, geometry g2);

Description

Compare deux objets géométriques et renvoie `true` si leur intersection "se croise spatialement", c'est-à-dire que les géométries ont certains points intérieurs en commun, mais pas tous. L'intersection des intérieurs des géométries doit être non vide et doit avoir une dimension inférieure à la dimension maximale des deux géométries d'entrée, et l'intersection des deux géométries ne doit pas être égale à l'une ou l'autre géométrie. Sinon, il renvoie `false`. La relation entre les croix est symétrique et irréflexive.

En termes mathématiques : $ST\_Crosses(A, B) \Leftrightarrow (dim(\ Int(A) \bigcap Int(B)\ ) < max(\ dim(\ Int(A)\ ), dim(\ Int(B)\ )\ )) \wedge (A \bigcap B \neq A) \wedge (A \bigcap B \neq B)$

Les géométries se croisent si leur matrice d'intersection DE-9IM correspond :

- `T*T******` pour les situations point/ligne, point/zone et ligne/zone
- `T*****T**` pour les situations Ligne/Point, Zone/Point et Zone/Ligne

- `0********` pour les situations ligne/ligne
- le résultat est `false` pour les situations Point/Point et Area/Area

---

Note
La spécification OpenGIS Simple Features définit ce prédicat uniquement pour les situations Point/Ligne, Point/Zone, Ligne/Ligne et Ligne/Zone. Le STC / GEOS étend la définition pour qu'elle s'applique également aux situations Ligne/-Point, Zone/Point et Zone/Ligne. Cela rend la relation symétrique.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries.

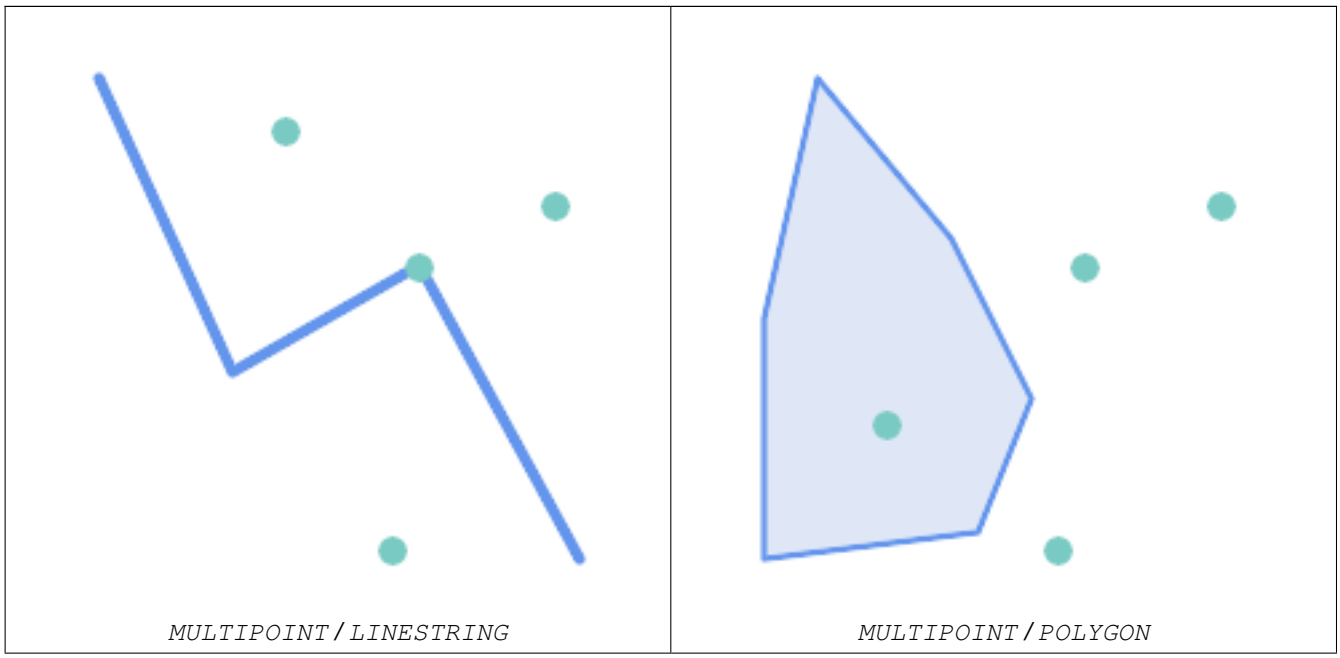
Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.13.3

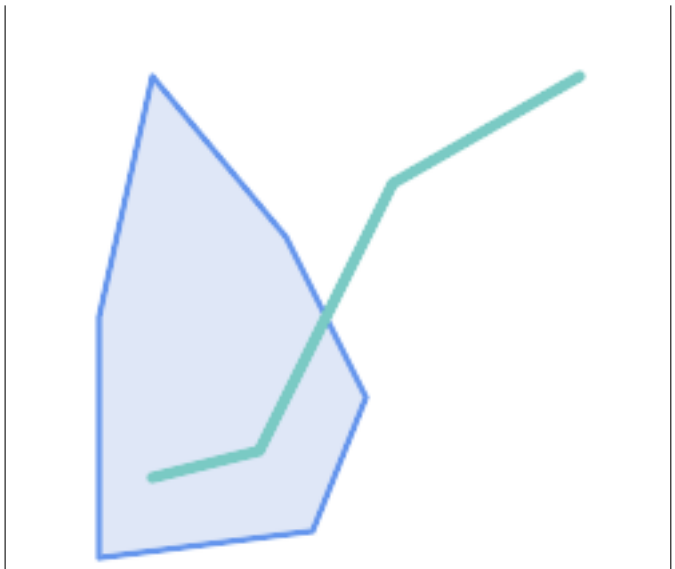
Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.29

Exemples

Les situations suivantes renvoient toutes `true`.



MULTIPOINT / LINESTRING | *MULTIPOINT / POLYGON*



LINESTRING/POLYGON

LINESTRING/LINESTRING

Prenons le cas d'un utilisateur qui dispose de deux tables : une table de routes et une table d'autoroutes.

```
CREATE TABLE roads (
  id serial NOT NULL,
  geom geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↩
     road_id)
);
```

```
CREATE TABLE highways (
  id serial NOT NULL,
  the_gem geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↩
     road_id)
);
```

Pour obtenir une liste des routes qui traversent une autoroute, utilisez une requête similaire à :

```
SELECT roads.id
FROM roads, highways
WHERE ST_Crosses(roads.geom, highways.geom);
```

Voir aussi

ST_Contains, ST_Overlaps

7.11.1.7 ST_Disjoint

ST_Disjoint — Teste si deux géométries n'ont pas de points communs

Synopsis

boolean **ST_Disjoint**(geometry A , geometry B);

Description

Renvoie `true` si deux géométries sont disjointes. Les géométries sont disjointes si elles n'ont aucun point en commun.

Si une autre relation spatiale est vraie pour une paire de géométries, celles-ci ne sont pas disjointes. La disjonction implique que ST_Intersects est fausse.

En termes mathématiques : *ST_Disjoint(A, B)* $\Leftrightarrow A \bigcap B = \varnothing$

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Effectué par le module GEOS

Note
Cet appel de fonction n'utilise pas d'index. Un prédicat négatif ST_Intersects peut être utilisé comme une alternative plus performante qui utilise des index : `ST_Disjoint(A,B) = NOT ST_Intersects(A,B)`

Note
NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 //s2.1.13.3 - a.Relate(b, 'FF*FF****')

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.26

Exemples

```
SELECT ST_Disjoint('POINT(0 0)'::geometry, 'LINESTRING ( 2 0, 0 2 )'::geometry);
 st_disjoint
---------------
 t
(1 row)
SELECT ST_Disjoint('POINT(0 0)'::geometry, 'LINESTRING ( 0 0, 0 2 )'::geometry);
 st_disjoint
---------------
 f
(1 row)
```

Voir aussi

ST_Intersects

7.11.1.8 ST_Equals

ST_Equals — Teste si deux géométries comprennent le même ensemble de points

Synopsis

boolean **ST_Equals**(geometry A, geometry B);

Description

Renvoie `true` si les géométries données sont "topologiquement égales". Utilisez ceci pour une "meilleure" réponse que "=". L'égalité topologique signifie que les géométries ont la même dimension et que leurs ensembles de points occupent le même espace. Cela signifie que l'ordre des sommets peut être différent dans des géométries topologiquement égales. Pour vérifier que l'ordre des points est cohérent, utilisez ST_OrderingEquals (il convient de noter que ST_OrderingEquals est un peu plus strict que la simple vérification de l'ordre des points).

En termes mathématiques : *ST_Equals(A, B)* $\Leftrightarrow A = B$

La relation suivante est valable : *ST_Equals(A, B)* $\Leftrightarrow$ *ST_Within(A,B)* $\wedge$ *ST_Within(B,A)*

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.24

Modifié : 2.2.0 Retourne vrai même pour les géométries invalides si elles sont binairement égales

Exemples

```
SELECT ST_Equals(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
    ST_GeomFromText('LINESTRING(0 0, 5 5, 10 10)'));
 st_equals
-----------
 t
(1 row)

SELECT ST_Equals(ST_Reverse(ST_GeomFromText('LINESTRING(0 0, 10 10)')),
    ST_GeomFromText('LINESTRING(0 0, 5 5, 10 10)'));
 st_equals
-----------
 t
(1 row)
```

Voir aussi

ST_IsValid, ST_OrderingEquals, ST_Reverse, ST_Within

7.11.1.9 ST_Intersects

ST_Intersects — Teste si deux géométries se croisent (elles ont au moins un point en commun)

Synopsis

boolean **ST_Intersects**(geometry geomA , geometry geomB);
boolean **ST_Intersects**(geography geogA , geography geogB);

Description

Renvoie `true` si deux géométries se croisent. Les géométries se croisent si elles ont un point commun.

Pour les objets de type geography, une tolérance de distance de 0,00001 mètre est utilisée (les points très proches sont donc considérés comme se croisant).

En termes mathématiques : $ST\_Intersects(A, B) \Leftrightarrow A \bigcap B \neq \varnothing$

Les géométries se croisent si leur matrice d'intersection DE-9IM correspond à l'un des éléments suivants :

- `T********`
- `*T*******`
- `***T*****`
- `****T****`

L'intersection spatiale est impliquée par tous les autres tests de relations spatiales, à l'exception de ST_Disjoint, qui teste que les géométries ne se croisent PAS.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries.

Modifié : 3.0.0 La version SFCGAL a été supprimée et la prise en charge native des TINS 2D a été ajoutée.

Amélioré : 2.5.0 Supporte GEOMETRYCOLLECTION.

Amélioré : 2.3.0 Amélioration du court-circuit PIP étendu à la prise en charge des multipoints avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones.

Effectuée par le module GEOS (pour geometry), la geography est native

Disponibilité : la version 1.5 a introduit la prise en charge du type geography.

Note
Pour geography, cette fonction a une tolérance de distance d'environ 0,00001 mètre et utilise la sphère plutôt que le calcul du sphéroïde.

Note
NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 //s2.1.13.3 - ST_Intersects(g1, g2) --> Not (ST_Disjoint(g1, g2))

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.27

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples de géométrie

```
SELECT ST_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 2 0, 0 2 )'::geometry);
 st_intersects
---------------
 f
(1 row)
SELECT ST_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 0 0, 0 2 )'::geometry);
 st_intersects
---------------
 t
(1 row)

-- Look up in table. Make sure table has a GiST index on geometry column for faster lookup.
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))');
 id | name
----+-------
  2 | Minsk
(1 row)
```

Exemples géographiques

```
SELECT ST_Intersects(
    'SRID=4326;LINESTRING(-43.23456 72.4567,-43.23456 72.4568)'::geography,
    'SRID=4326;POINT(-43.23456 72.4567772)'::geography
    );

 st_intersects
---------------
t
```

Voir aussi

&&, ST_3DIntersects, ST_Disjoint

7.11.1.10 ST_LineCrossingDirection

ST_LineCrossingDirection — Renvoie un nombre indiquant le comportement de croisement de deux LineStrings

Synopsis

integer **ST_LineCrossingDirection**(geometry linestringA, geometry linestringB);

Description

En ayant 2 lignes en entrée, renvoie un nombre entier entre -3 et 3 indiquant le type de croisement qui existe entre elles. 0 indique qu'il n'y a pas de croisement. Cette fonction n'est prise en charge que pour les `LINESTRING`s.

Le numéro de croisement a la signification suivante :

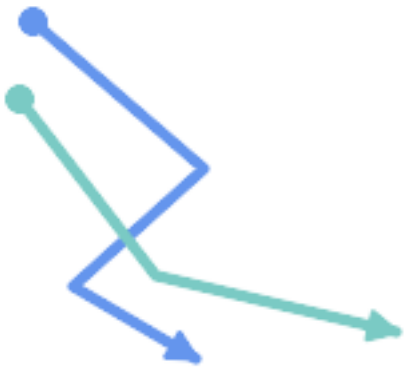
- 0 : LIGNE SANS CROISEMENT
- -1 : LIGNE CROISÉE À GAUCHE
- -1 : LIGNE CROISÉE À DROITE

- -2 : LIGNE MULTICROISEMENT EXTRÉMITÉ GAUCHE
- -2 : LIGNE MULTICROISEMENT EXTRÉMITÉ DROITE
- -3 : LIGNE MULTICROISEMENT FIN IDENTIQUE AU PREMIER GAUCHE
- -3 : LIGNE MULTICROISEMENT FIN IDENTIQUE AU PREMIER DROITE

Disponibilité : 1.4

Exemples

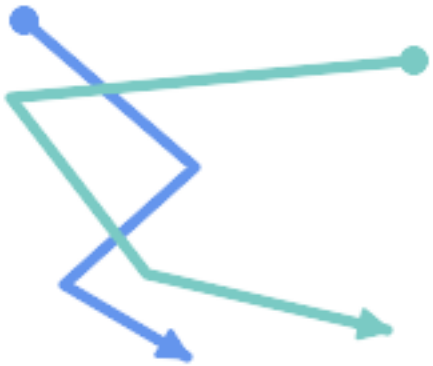
Exemple: LIGNE CROISEE A GAUCHE et LIGNE CROISEE A DROITE



Bleu : Ligne A ; Vert : Ligne B

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
  ST_GeomFromText('LINESTRING(25 169,89 114,40 70,86 43)') As lineA,
  ST_GeomFromText('LINESTRING (20 140, 71 74, 161 53)') As lineB
  ) As foo;

 A_cross_B | B_cross_A
-----------+-----------
        -1 |         1
```

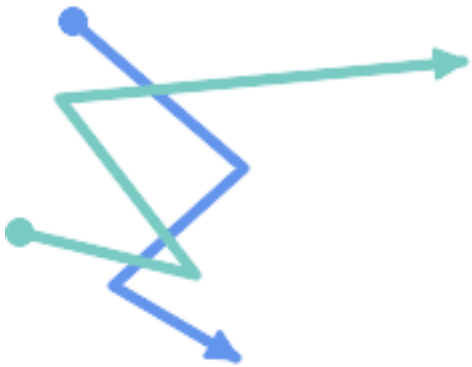
Exemple: LIGNE MULTICROISEMENT FIN IDENTIQUE AU PREMIER GAUCHE et LIGNE MULTICROISEMENT FIN IDENTIQUE AU PREMIER DROITE



Bleu : Ligne A ; Vert : Ligne B

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
 ST_GeomFromText('LINESTRING(25 169,89 114,40 70,86 43)') As lineA,
 ST_GeomFromText('LINESTRING(171 154,20 140,71 74,161 53)') As lineB
  ) As foo;

 A_cross_B | B_cross_A
-----------+-----------
         3 |        -3
```

Exemple: LIGNE MULTICROISEMENT EXTRÉMITÉ GAUCHE et LIGNE MULTICROISEMENT EXTRÉMITÉ DROITE



Bleu : Ligne A ; Vert : Ligne B

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
  ST_GeomFromText('LINESTRING(25 169,89 114,40 70,86 43)') As lineA,
  ST_GeomFromText('LINESTRING(5 90, 71 74, 20 140, 171 154)') As lineB
  ) As foo;
```

```
 A_cross_B | B_cross_A
-----------+-----------
        -2 |         2
```

Exemple: Trouve toutes les rues qui se croisent

```
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;
```

Voir aussi

ST_Crosses

7.11.1.11 ST_OrderingEquals

ST_OrderingEquals — Teste si deux géométries représentent la même géométrie et ont des points dans le même ordre directionnel

Synopsis

boolean **ST_OrderingEquals**(geometry A, geometry B);

Description

ST_OrderingEquals compare deux géométries et renvoie t (TRUE) si les géométries sont égales et si les coordonnées sont dans le même ordre ; sinon, il renvoie f (FALSE).

Note
Cette fonction est mise en œuvre conformément à la spécification SQL d'ArcSDE plutôt que SQL-MM. http://edndoc.esri.com/arcsde/9.1/sql_api/sqlapi3.htm#ST_OrderingEquals

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.43

Exemples

```
SELECT ST_OrderingEquals(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
    ST_GeomFromText('LINESTRING(0 0, 5 5, 10 10)'));
 st_orderingequals
-----------
 f
(1 row)

SELECT ST_OrderingEquals(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
    ST_GeomFromText('LINESTRING(0 0, 0 0, 10 10)'));
 st_orderingequals
-----------
 t
```

```
(1 row)

SELECT ST_OrderingEquals(ST_Reverse(ST_GeomFromText('LINESTRING(0 0, 10 10)')),
    ST_GeomFromText('LINESTRING(0 0, 0 0, 10 10)'));
 st_orderingequals
-------------------
 f
(1 row)
```

Voir aussi

&&, ST_Equals, ST_Reverse

7.11.1.12 ST_Overlaps

ST_Overlaps — Teste si deux géométries ont la même dimension et se croisent, mais si chacune a au moins un point qui n'est pas dans l'autre

Synopsis

boolean **ST_Overlaps**(geometry A, geometry B);

Description

Renvoie TRUE si les géométries A et B se "chevauchent spatialement". Deux géométries se chevauchent si elles ont la même dimension, si leurs intérieurs se croisent dans cette dimension et si chacune a au moins un point à l'intérieur de l'autre (ou, de manière équivalente, si aucune des deux ne recouvre l'autre). La relation de chevauchement est symétrique et irréflexive.

En termes mathématiques : $ST\_Overlaps(A, B) \Leftrightarrow (dim(A) = dim(B) = dim(Int(A) \bigcap Int(B))). \wedge (A \bigcap B \neq A) \wedge (A \bigcap B \neq B)$

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries. Pour éviter l'utilisation d'un index, utilisez la fonction `_ST_Overlaps`.

Effectué par le module GEOS

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

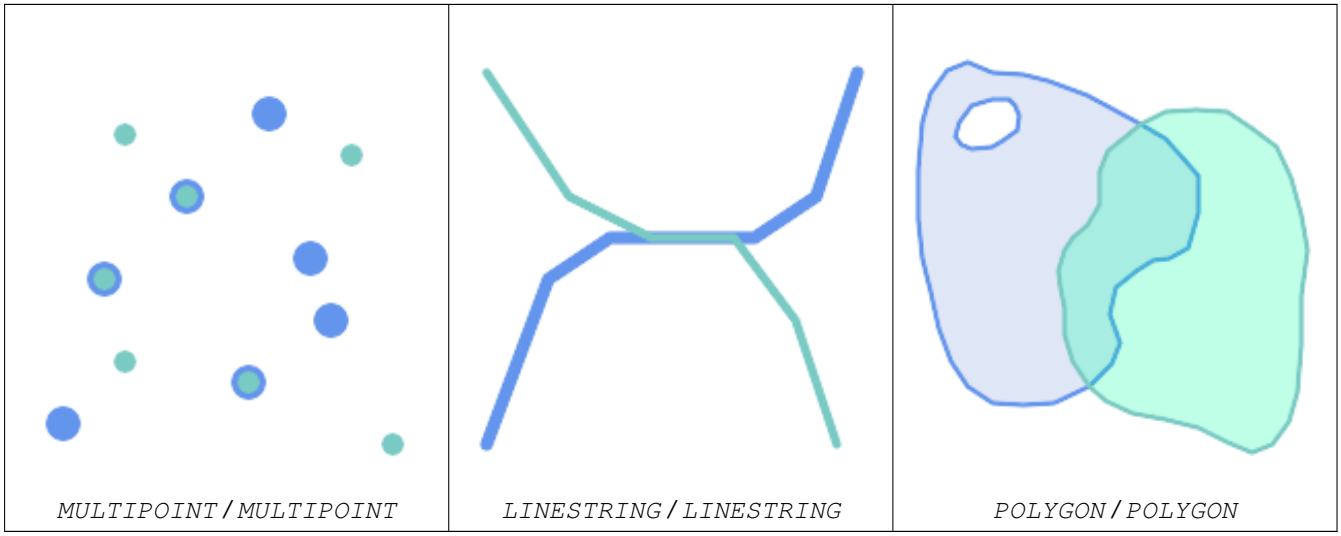
NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

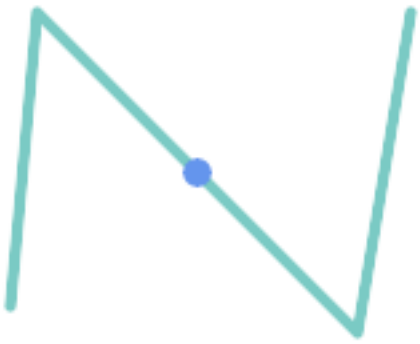
Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 // s2.1.13.3

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.32

Exemples

`ST_Overlaps` renvoie `TRUE` dans les situations suivantes :



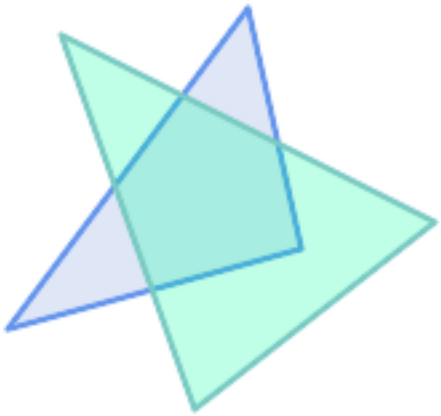


Un point sur une LineString est contenu, mais comme il a une dimension inférieure, il ne se chevauche pas et ne se croise pas.

```
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 ST_GeomFromText('POINT (100 100)') As a,
             ST_GeomFromText('LINESTRING (30 50, 40 160, 160 40, 180 160)')  AS b) AS t

overlaps | crosses | intersects | b_contains_a
---------+---------------------+--------------
f        | f       | t          | t
```

Une chaîne de lignes qui recouvre partiellement un polygone l'intersecte et le traverse, mais ne se chevauche pas car elle a des dimensions différentes.

```
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 ST_GeomFromText('POLYGON ((40 170, 90 30, 180 100, 40 170))') AS a,
             ST_GeomFromText('LINESTRING(10 10, 190 190)') AS b) AS t;

 overlap | crosses | intersects | contains
---------+---------+------------+--------------
 f       | t       | t          | f
```



Deux polygones qui se croisent mais dont aucun n'est contenu par l'autre se chevauchent, mais ne se croisent pas car leur intersection a la même dimension.

```
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 ST_GeomFromText('POLYGON ((40 170, 90 30, 180 100, 40 170))') AS a,
             ST_GeomFromText('POLYGON ((110 180, 20 60, 130 90, 110 180))') AS b) As t;
```

```
 overlaps | crosses | intersects | b_contains_a | dim_a | dim_b | dim_int
----------+---------+------------+--------------+-------+-------+-----------
 t        | f       | t          | f            |     2 |     2 |       2
```

Voir aussi

ST_Contains, ST_Crosses, ST_Dimension, ST_Intersects

7.11.1.13 ST_Relate

ST_Relate — Teste si deux géométries ont une relation topologique correspondant à un modèle de matrice d'intersection, ou calcule leur matrice d'intersection

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);

Description

Ces fonctions permettent de tester et d'évaluer la relation spatiale (topologique) entre deux géométries, telle que définie par le Dimensionally Extended 9-Intersection Model (DE-9IM).

Le DE-9IM est une matrice à 9 éléments indiquant la dimension des intersections entre l'intérieur, la frontière et l'extérieur de deux géométries. Elle est représentée par une chaîne de texte de 9 caractères utilisant les symboles "F", "0", "1", "2" (par exemple, `'FF1FF0102'`).

Un type spécifique de relation spatiale peut être testé en faisant correspondre la matrice d'intersection à un *motif de matrice d'intersection*. Les motifs peuvent inclure les symboles supplémentaires "T" (signifiant "l'intersection est non vide") et "*" (signifiant "n'importe quelle valeur"). Les relations spatiales communes sont fournies par les fonctions nommées ST_Contains, ST_ContainsProperly, ST_Covers, ST_CoveredBy, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps, ST_Touches, et ST_Within. L'utilisation d'un modèle explicite permet de tester plusieurs conditions d'intersection, de croisement, etc. en une seule étape. Elle permet également de tester des relations spatiales qui n'ont pas de fonction de relation spatiale nommée. Par exemple, la relation "Interior-Intersects" possède le motif DE-9IM `T********`, qui n'est évalué par aucun prédicat nommé.

Pour plus d'informations, voir Section 5.1.

Variante 1: Teste si deux géométries sont spatialement liées selon le `intersectionMatrixPattern` donné.

Note
Contrairement à la plupart des prédicats de relations spatiales nommées, ce prédicat n'inclut PAS automatiquement un appel d'index. La raison en est que certaines relations sont vraies pour des géométries qui ne s'intersectent PAS (par exemple Disjoint). Si vous utilisez un modèle de relation qui nécessite une intersection, incluez l'appel à l'index &&.

Note
Il est préférable d'utiliser une fonction de relation nommée si elle est disponible, car elle utilise automatiquement un index spatial lorsqu'il existe. En outre, elles peuvent mettre en œuvre des optimisations de performance qui ne sont pas disponibles avec l'évaluation de la relation complète.

Variante 2: Renvoie la chaîne matricielle DE-9IM pour la relation spatiale entre les deux géométries d'entrée. La chaîne matricielle peut être testée pour vérifier si elle correspond à un modèle DE-9IM en utilisant ST_RelateMatch.

Variante 3: Comme la variante 2, mais permet de spécifier une **Boundary Node Rule**. Une boundary node rule permet de contrôler plus finement si les extrémités des multilignes sont considérées comme se situant à l'intérieur ou à la limite du DE-9IM. Les valeurs de `boundaryNodeRule` sont les suivantes :

- `1` : **OGC-Mod2** - les extrémités des lignes sont dans la frontière si elles apparaissent un nombre impair de fois. C'est la règle définie par la norme SFS de l'OGC, et c'est la valeur par défaut de la `ST_Relate`.
- `2` : **Endpoint** - tous les points d'extrémité sont dans la frontière.
- `3` : **MultivalentEndpoint** - les points d'extrémité sont dans la frontière s'ils apparaissent plus d'une fois. En d'autres termes, la frontière est constituée de tous les points d'extrémité "attachés" ou "internes" (mais pas des points d'extrémité "non attachés/externes").
- `4` : **MonovalentEndpoint** - les points d'extrémité sont dans la frontière s'ils n'apparaissent qu'une seule fois. En d'autres termes, la frontière est constituée de tous les points d'extrémité "non attachés" ou "extérieurs".

Cette fonction ne figure pas dans la spécification de l'OGC, mais elle est implicite. voir s2.1.13.2

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 // s2.1.13.3

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.25

Effectué par le module GEOS

Amélioration : 2.0.0 - ajout de la prise en charge de la spécification de boundary node rule.

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Exemples

Utilisation de la fonction booléenne pour tester les relations spatiales.

```
SELECT ST_Relate('POINT(1 2)', ST_Buffer( 'POINT(1 2)', 2), '0FFFFF212');
st_relate
-----------
t

SELECT ST_Relate(POINT(1 2)', ST_Buffer( 'POINT(1 2)', 2), '*FF*FF212');
st_relate
-----------
t
```

Test d'un modèle de relation spatiale personnalisé comme condition de requête, avec `&&` pour permettre l'utilisation d'un index spatial.

```
-- Find compounds that properly intersect (not just touch) a poly (Interior Intersects)

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********');
```

Calcul de la matrice d'intersection pour les relations spatiales.

```
SELECT ST_Relate( 'POINT(1 2)',
                  ST_Buffer( 'POINT(1 2)', 2));
-----------
0FFFFF212
```

```
SELECT ST_Relate( 'LINESTRING(1 2, 3 4)',
                  'LINESTRING(5 6, 7 8)' );
-----------
FF1FF0102
```

Utilisation de différentes Boundary Node Rules pour calculer la relation spatiale entre une LineString et une MultiLineString avec une extrémité dupliquée `(3 3)` :

- En utilisant la règle **OGC-Mod2** (1), l'extrémité dupliquée se trouve dans l' **intérieur** de la MultiLineString, de sorte que l'entrée de la matrice DE-9IM [aB:bI] est `0` et [aB:bB] est `F`.
- En utilisant la règle **Endpoint** (2), l'extrémité dupliquée se trouve dans la **limite** de la MultiLineString, de sorte que l'entrée de la matrice DE-9IM [aB:bI] est `F` et [aB:bB] est `0`.

```
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;

 bnr_mod2  | bnr_endpoint
-----------+--------------
 FF10F0102 | FF1F00102
```

Voir aussi

Section 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 — Teste si une matrice d'intersection DE-9IM correspond à un modèle de matrice d'intersection

Synopsis

boolean **ST_RelateMatch**(text intersectionMatrix, text intersectionMatrixPattern);

Description

Teste si une Dimensionally Extended 9-Intersection Model (DE-9IM) `intersectionMatrix` satisfait un `intersectionMatrixP`
Les valeurs de la matrice d'intersection peuvent être calculées par ST_Relate.

Pour plus d'informations, voir Section 5.1.

Effectué par le module GEOS

Disponibilité : 2.0.0

Exemples

```
SELECT ST_RelateMatch('101202FFF', 'TTTTTTFFF') ;
-- result --
t
```

Modèles de relations spatiales communes mis en correspondance avec les valeurs de la matrice d'intersection, pour une ligne dans différentes positions par rapport à un polygone

```
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);

 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
```

Voir aussi

Section 5.1, ST_Relate

7.11.1.15 ST_Touches

ST_Touches — Teste si deux géométries ont au moins un point en commun, mais que leurs intérieurs ne se croisent pas

Synopsis

boolean **ST_Touches**(geometry A, geometry B);

Description

Retourne `TRUE` si A et B se croisent, mais que leurs intérieurs ne se croisent pas. De manière équivalente, A et B ont au moins un point en commun, et les points communs se situent dans au moins une frontière. Pour les entrées point/point, la relation est toujours `FALSE`, puisque les points n'ont pas de frontière.

In mathematical terms: $ST\_Touches(A, B) \Leftrightarrow (Int(A) \bigcap Int(B) = \varnothing) \wedge (A \bigcap B \neq \varnothing)$

Cette relation est valable si la matrice d'intersection DE-9IM pour les deux géométries correspond à l'une d'entre elles :

- FT*******
- F**T*****
- F***T****

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries. Pour éviter d'utiliser un index, utilisez plutôt la `_ST_Touches`.

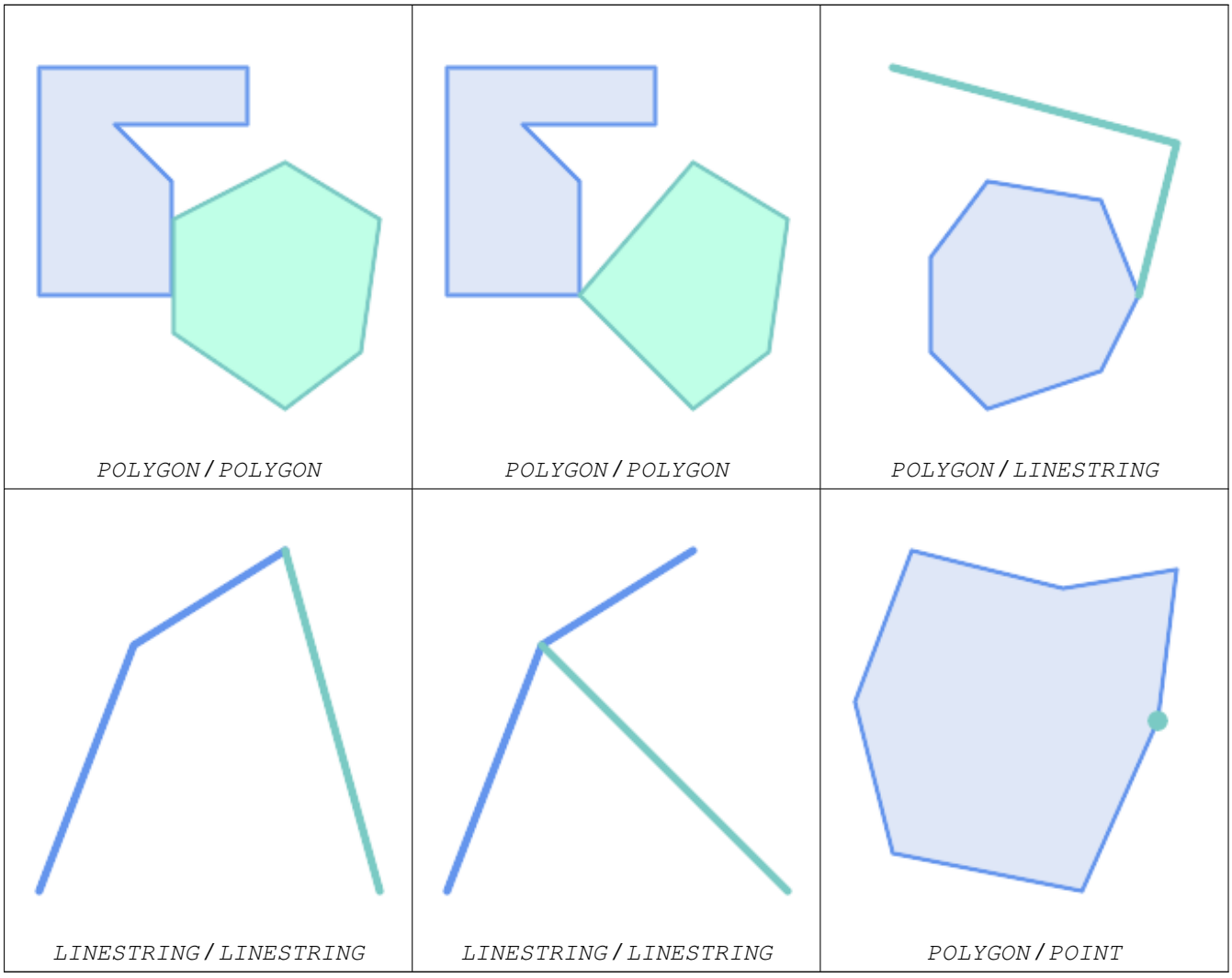
Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 // s2.1.13.3

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.28

Exemples

Le prédicat `ST_Touches` renvoie `TRUE` dans les exemples suivants.



```
SELECT ST_Touches('LINESTRING(0 0, 1 1, 0 2)'::geometry, 'POINT(1 1)'::geometry);
 st_touches
------------
 f
(1 row)

SELECT ST_Touches('LINESTRING(0 0, 1 1, 0 2)'::geometry, 'POINT(0 2)'::geometry);
 st_touches
------------
 t
(1 row)
```

7.11.1.16 ST_Within

ST_Within — Tests si chaque point de A se trouve dans B, et que leurs intérieurs ont un point commun

Synopsis

boolean **ST_Within**(geometry A, geometry B);

Description

Renvoie TRUE si la géométrie A est à l'intérieur de la géométrie B. A est à l'intérieur de B si et seulement si tous les points de A se trouvent à l'intérieur (c'est-à-dire à l'intérieur ou à la limite) de B (ou de manière équivalente, aucun point de A ne se trouve à l'extérieur de B), et si les intérieurs de A et de B ont au moins un point en commun.

Pour que cette fonction ait un sens, les géométries sources doivent toutes deux avoir la même projection de coordonnées et le même SRID.

En termes mathématiques : $ST\_Within(A, B) \Leftrightarrow (A \bigcap B = A) \wedge (Int(A) \bigcap Int(B) \neq \varnothing)$.

La relation within est réflexive : toute géométrie est à l'intérieur d'elle-même. La relation est antisymétrique : si `ST_Within(A,B) = true` et `ST_Within(B,A) = true`, alors les deux géométries doivent être topologiquement égales (`ST_Equals(A,B) = true`).

ST_Within est le contraire de ST_Contains. Ainsi, `ST_Within(A,B) = ST_Contains(B,A)`.

Note
Comme les intérieurs doivent avoir un point commun, une subtilité de la définition est que les lignes et les points situés entièrement dans la limite des polygones ou des lignes ne sont *pas* à l'intérieur de la géométrie. Pour plus de détails, voir Subtleties of OGC Covers, Contains, Within. Le prédicat ST_CoveredBy fournit une relation plus inclusive.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries. Pour éviter l'utilisation d'un index, utilisez la fonction `_ST_Within`.

Effectué par le module GEOS

Amélioration : 2.3.0 Amélioration du court-circuit PIP pour la géométrie étendue à la prise en charge des multipoints avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones.

Important
Amélioration : 3.0.0 a permis la prise en charge de `GEOMETRYCOLLECTION`

Important
N'utilisez pas cette fonction avec des géométries non valides. Vous obtiendrez des résultats inattendus.

NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 // s2.1.13.3 - a.Relate(b, 'T*F**F***')

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.30

Exemples

```
--a circle within a circle
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
(
SELECT ST_Buffer(ST_GeomFromText('POINT(50 50)'), 20) As smallc,
  ST_Buffer(ST_GeomFromText('POINT(50 50)'), 40) As bigc) As foo;
--Result
 smallinsmall | smallinbig | biginsmall | unioninbig | biginunion | bigisunion
--------------+------------+------------+------------+------------+------------
 t            | t          | f          | t          | t          | t
(1 row)
```

Voir aussi

ST_Contains, ST_CoveredBy, ST_Equals, ST_IsValid

7.11.2 Relations de distance

7.11.2.1 ST_3DDWithin

ST_3DDWithin — Teste si deux géométries 3D se trouvent à une distance 3D donnée

Synopsis

boolean **ST_3DDWithin**(geometry g1, geometry g2, double precision distance_of_srid);

Description

Renvoie true si la distance 3D entre deux valeurs géométriques n'est pas supérieure à la distance `distance_of_srid`. La distance est spécifiée en unités définies par le système de référence spatial des géométries. Pour que cette fonction ait un sens, les géométries sources doivent se trouver dans le même système de coordonnées (avoir le même SRID).

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette méthode implémente la spécification SQL/MM. SQL-MM ?

Disponibilité : 2.0.0

Exemples

```
-- Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point  ↩
    and line compared 2D point and line)
-- Note: currently no vertical datum support so Z is not transformed and assumed to be same ↩
     units as final.
SELECT ST_3DDWithin(
      ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 4)'),2163),
      ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 15, -72.123 42.1546 ↩
           20)'),2163),
      126.8
    ) As within_dist_3d,
ST_DWithin(
      ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 4)'),2163),
      ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 15, -72.123 42.1546 ↩
           20)'),2163),
      126.8
    ) As within_dist_2d;

 within_dist_3d | within_dist_2d
----------------+----------------
 f              | t
```

Voir aussi

ST_3DDFullyWithin, ST_DWithin, ST_DFullyWithin, ST_3DDistance, ST_Distance, ST_3DMaxDistance, ST_Transform

7.11.2.2 ST_3DDFullyWithin

ST_3DDFullyWithin — Teste si deux géométries 3D sont entièrement comprises dans une distance 3D donnée

Synopsis

boolean **ST_3DDFullyWithin**(geometry g1, geometry g2, double precision distance);

Description

Renvoie true si les géométries 3D se trouvent à la distance spécifiée l'une de l'autre. La distance est spécifiée en unités définies par le système de référence spatiale des géométries. Pour que cette fonction ait un sens, les géométries sources doivent toutes deux avoir la même projection de coordonnées et le même SRID.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries.

Disponibilité : 2.0.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

```
-- This compares the difference between fully within and distance within as well
    -- as the distance fully within for the 2D footprint of the line/point vs. the 3d fully ↩
        within
    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
    (select ST_GeomFromEWKT('POINT(1 1 2)') as geom_a,
    ST_GeomFromEWKT('LINESTRING(1 5 2, 2 7 20, 1 9 100, 14 12 3)') as geom_b) t1;
 d3dfullywithin10 | d3dwithin10 | d2dfullywithin20 | d3dfullywithin20
------------------+-------------+------------------+------------------
 f                | t           | t                | f
```

Voir aussi

ST_3DDWithin, ST_DWithin, ST_DFullyWithin, ST_3DMaxDistance

7.11.2.3 ST_DFullyWithin

ST_DFullyWithin — Teste si une géométrie se trouve entièrement à une distance d'une autre géométrie

Synopsis

boolean **ST_DFullyWithin**(geometry g1, geometry g2, double precision distance);

Description

Retourne vrai si `g2` se trouve entièrement à une distance `distance` de `g1`. Visuellement, la condition est vraie si `g2` est contenu dans un tampon de `distance` de `g1`. La distance est spécifiée en unités définies par le système de référence spatial des géométries.

Note
Cette fonction inclut une comparaison de la boîte englobante qui utilise tous les index disponibles sur les géométries.

Disponibilité : 1.5.0

Modifié : 3.5.0 : la logique implémentée utilise désormais un test de confinement dans un tampon, plutôt que l'algorithme ST_MaxDistance. Les résultats seront différents de ceux des versions précédentes, mais devraient être plus proches des attentes de l'utilisateur.

Exemples

```
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)', 'LINESTRING(1 5, 2 7, 1 9, 14 12)')
    ) AS v(geom_a, geom_b)

 dfullywithin10 | dwithin10 | dfullywithin20
----------------+-----------+----------------
 f              | t         | t
```

Voir aussi

ST_MaxDistance, ST_DWithin, ST_3DDWithin, ST_3DDFullyWithin

7.11.2.4 ST_DWithin

ST_DWithin — Teste si deux géométries se trouvent à une distance donnée

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_spheroid = true);

Description

Renvoie true si les géométries se trouvent à une distance donnée

Pour le type geometry : la distance est spécifiée en unités définies par le système de référence spatiale des géométries. Pour que cette fonction ait un sens, les géométries sources doivent se trouver dans le même système de coordonnées (avoir le même SRID).

Pour le type geography : les unités sont en mètres et la mesure de la distance est par défaut `use_spheroid = true`. Pour une évaluation plus rapide, utilisez `use_spheroid = false` pour mesurer sur la sphère.

Note
Utilisez ST_3DDWithin pour les géométries 3D.

Note
Cet appel de fonction inclut une comparaison de la boîte de délimitation qui utilise tous les index disponibles sur les géométries.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Disponibilité : la prise en charge du type geography a été introduite dans la version 1.5.0

Amélioration : la version 2.1.0 a amélioré la vitesse de la géographie. Voir Making Geography faster pour plus de détails.

Amélioration : la prise en charge des géométries courbes a été introduite dans la version 2.1.0.

Avant la version 1.3, ST_Expand était couramment utilisé en conjonction avec && et ST_Distance pour tester la distance, et avant la version 1.3.4, cette fonction utilisait cette logique. À partir de la version 1.3.4, ST_DWithin utilise une fonction de distance de court-circuit plus rapide.

Exemples

```
-- 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
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 is in units of spatial ref (e.g. meters, feet, degrees)
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 broadcasting towers that receiver with limited range can receive.
-- Data is geometry in Spherical Mercator (SRID=3857), ranges are approximate.

-- Create geometry index that will check proximity limit of user to tower
CREATE INDEX ON broadcasting_towers using gist (geom);

-- Create geometry index that will check proximity limit of tower to user
CREATE INDEX ON broadcasting_towers using gist (ST_Expand(geom, sending_range));

-- Query towers that 4-kilometer receiver in Minsk Hackerspace can get
-- Note: two conditions, because shorter LEAST(b.sending_range, 4000) will not use index.
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);
```

Voir aussi

ST_Distance, ST_3DDWithin

7.11.2.5 ST_PointInsideCircle

ST_PointInsideCircle — Teste si un point géométrique se trouve à l'intérieur d'un cercle défini par un centre et un rayon

Synopsis

boolean **ST_PointInsideCircle**(geometry a_point, float center_x, float center_y, float radius);

Description

Retourne true si la géométrie est un point et se trouve à l'intérieur du cercle de centre `center_x`,`center_y` et de rayon `radius`.

Warning
N'utilise pas les index spatiaux. Utilisez plutôt ST_DWithin.

Disponibilité : 1.2

Modifié : 2.2.0 Dans les versions précédentes, cette fonction était appelée ST_Point_Inside_Circle

Exemples

```
SELECT ST_PointInsideCircle(ST_Point(1,2), 0.5, 2, 3);
 st_pointinsidecircle
----------------------
 t
```

Voir aussi

ST_DWithin

7.12 Fonctions de mesure

7.12.1 ST_Area

ST_Area — Renvoie l'aire d'une géométrie polygonale.

Synopsis

float **ST_Area**(geometry g1);
float **ST_Area**(geography geog, boolean use_spheroid = true);

Description

Renvoie l'aire d'une géométrie polygonale. Pour les types geometry, une surface cartésienne 2D (planaire) est calculée, avec des unités spécifiées par le SRID. Pour les types geography, la surface est déterminée par défaut sur un sphéroïde avec des unités en mètres carrés. Pour calculer la surface en utilisant le modèle sphérique, plus rapide mais moins précis, utilisez `ST_Area(geog,false)`.

Amélioration : 2.0.0 - la prise en charge des surfaces polyédriques 2D a été introduite.

Amélioration : 2.2.0 - mesure sur sphéroïde effectuée avec GeographicLib pour une meilleure précision et robustesse. Nécessite PROJ >= 4.9.0 pour profiter de la nouvelle fonctionnalité.

Modifié : 3.0.0 - ne dépend plus de SFCGAL.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 8.1.2, 9.5.3

Cette fonction prend en charge les surfaces Polyhedral.

Note
Pour les surfaces polyédriques, ne prend en charge que les surfaces polyédriques 2D (pas 2.5D). Pour les surfaces 2.5D, la réponse peut être différente de zéro, mais uniquement pour les faces qui se trouvent entièrement dans le plan XY.

Exemples

Retourne la superficie en pieds carrés d'un terrain du Massachusetts et multipliez par la conversion pour obtenir des mètres carrés. Notez que la surface est exprimée en pieds carrés, car EPSG:2249 correspond aux pieds de plan de l'État du Massachusetts

```
select ST_Area(geom) sqft,
    ST_Area(geom) * 0.3048 ^ 2 sqm
from (
         select 'SRID=2249;POLYGON((743238 2967416,743238 2967450,
                                 743265 2967450,743265.625 2967416,743238 2967416))' ::  ↩
                                     geometry geom
     ) subquery;
&#x250c;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x252c;&#x2500;&#x2
&#x2502;   sqft    &#x2502;      sqm      &#x2502;
&#x251c;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x253c;&#x2500;&#x2
&#x2502; 928.625 &#x2502; 86.27208552 &#x2502;
&#x2514;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2534;&#x2500;&#x2
```

Retourne la surface en pieds carrés et transformez-la en mètres du plan de l'État du Massachusetts (EPSG:26986) pour obtenir des mètres carrés. Notez qu'il s'agit de pieds carrés car 2249 est le plan de l'État du Massachusetts en pieds et que la surface transformée est en mètres carrés car EPSG:26986 est le plan de l'État du Massachusetts en mètres

```
select ST_Area(geom) sqft,
    ST_Area(ST_Transform(geom, 26986)) As sqm
from (
         select
             'SRID=2249;POLYGON((743238 2967416,743238 2967450,
             743265 2967450,743265.625 2967416,743238 2967416))' :: geometry geom
     ) subquery;
&#x250c;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x252c;&#x2500;&#x2
&#x2502;   sqft    &#x2502;        sqm       &#x2502;
&#x251c;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x253c;&#x2500;&#x2
&#x2502; 928.625 &#x2502; 86.272430607008 &#x2502;
&#x2514;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2534;&#x2500;&#x2
```

Retourne la surface en pieds carrés et en mètres carrés en utilisant le type de données geography. Notez que nous transformons notre geometry en geography (avant de pouvoir le faire, assurez-vous que votre géométrie est en WGS 84 long lat 4326). La geography se mesure toujours en mètres. Il s'agit d'une démonstration à des fins de comparaison. Normalement, votre table sera déjà stockée dans le type de données geography.

```
select ST_Area(geog) / 0.3048 ^ 2 sqft_spheroid,
    ST_Area(geog, false) / 0.3048 ^ 2 sqft_sphere,
    ST_Area(geog) sqm_spheroid
from (
         select ST_Transform(
                    'SRID=2249;POLYGON((743238 2967416,743238 2967450,743265  ↩
                       2967450,743265.625 2967416,743238 2967416))'::geometry,
                    4326
                ) :: geography geog
     ) as subquery;
&#x250c;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2
&#x2502;  sqft_spheroid   &#x2502;   sqft_sphere    &#x2502;   sqm_spheroid   &#x2502;
&#x251c;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2
&#x2502; 928.684405784452 &#x2502; 927.049336105925 &#x2502; 86.2776044979692 &#x2502;
&#x2514;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2
```

Si vos données sont déjà dans le type geography :

```
select ST_Area(geog) / 0.3048 ^ 2 sqft,
    ST_Area(the_geog) sqm
from somegeogtable;
```

Voir aussi

ST_3DArea, ST_GeomFromText, ST_GeographyFromText, ST_SetSRID, ST_Transform

7.12.2 ST_Azimuth

ST_Azimuth — Renvoie l'azimut basé sur le nord d'une ligne entre deux points.

Synopsis

float **ST_Azimuth**(geometry origin, geometry target);
float **ST_Azimuth**(geography origin, geography target);

Description

Renvoie l'azimut en radians du point cible par rapport au point d'origine, ou NULL si les deux points coïncident. L'angle d'azimut est un angle positif dans le sens des aiguilles d'une montre, référencé à partir de l'axe Y positif (geometry) ou du méridien Nord (geography) : North = 0; Northeast = $\pi/4$; East = $\pi/2$; Southeast = $3\pi/4$; South = π; Southwest $5\pi/4$; West = $3\pi/2$; Northwest = $7\pi/4$.

Pour le type geography, la solution azimutale est connue sous le nom de inverse geodesic problem.

L'azimut est un concept mathématique défini comme l'angle entre un vecteur de référence et un point, avec des unités angulaires en radians. La valeur du résultat en radians peut être convertie en degrés à l'aide de la fonction PostgreSQL `degrees()`.

L'azimut peut être utilisé conjointement avec ST_Translate pour déplacer un objet le long de son axe perpendiculaire. Voir la fonction `upgis_lineshift()` dans le PostGIS wiki pour une implémentation de ceci.

Disponibilité : 1.1.0

Amélioration : la prise en charge du type geography a été introduite dans la version 2.0.0.

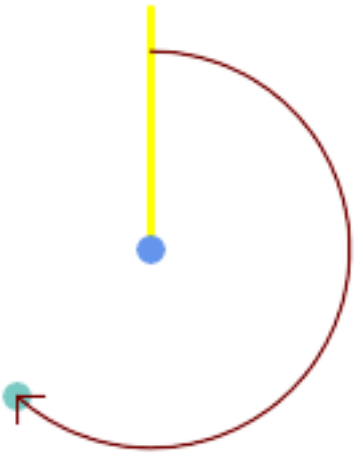
Amélioration : 2.2.0 mesure sur sphéroïde effectuée avec GeographicLib pour améliorer la précision et la robustesse. Nécessite PROJ >= 4.9.0 pour profiter de la nouvelle fonctionnalité.

Exemples

Geometry Azimut en degrés

```
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;

      dega_b       |      degb_a
-------------------+------------------
 42.2736890060937 | 222.273689006094
```

Bleu : origine Point(25,45) ; Vert : cible Point(75, 100) ; Jaune : axe Y ou Nord ; Rouge : angle d'azimut.



Bleu : origine Point(75, 100) ; Vert : cible Point(25, 45) ; Jaune : axe Y ou Nord ; Rouge : angle d'azimut.

Voir aussi

ST_Angle, ST_Point, ST_Translate, ST_Project, PostgreSQL Math Functions

7.12.3 ST_Angle

ST_Angle — Renvoie l'angle entre deux vecteurs définis par 3 ou 4 points, ou 2 lignes.

Synopsis

float **ST_Angle**(geometry point1, geometry point2, geometry point3, geometry point4);
float **ST_Angle**(geometry line1, geometry line2);

Description

Calcule l'angle dans le sens des aiguilles d'une montre entre deux vecteurs.

Variante 1: calcule l'angle formé par les points P1-P2-P3. Si un quatrième point est fourni, elle calcule l'angle formé par les points P1-P2 et P3-P4

Variante 2: calcule l'angle entre deux vecteurs S1-E1 et S2-E2, définis par les points de départ et d'arrivée des lignes d'entrée

Le résultat est un angle positif compris entre 0 et 2π radians. Le résultat en radians peut être converti en degrés à l'aide de la fonction PostgreSQL `degrees()`.

Notez que `ST_Angle(P1,P2,P3) = ST_Angle(P2,P1,P2,P3)`.

Disponibilité : 2.5.0

Exemples

Angle entre trois points

```
SELECT degrees( ST_Angle('POINT(0 0)', 'POINT(10 10)', 'POINT(20 0)') );

 degrees
---------
     270
```

Angle entre les vecteurs définis par quatre points

```
SELECT degrees( ST_Angle('POINT (10 10)', 'POINT (0 0)', 'POINT(90 90)', 'POINT (100 80)')  ↩
    );

      degrees
--------------------
 269.9999999999999
```

Angle entre les vecteurs définis par les points de départ et d'arrivée des lignes

```
SELECT degrees( ST_Angle('LINESTRING(0 0, 0.3 0.7, 1 1)', 'LINESTRING(0 0, 0.2 0.5, 1 0)')  ↩
    );

      degrees
--------------
           45
```

Voir aussi

ST_Azimuth

7.12.4 ST_ClosestPoint

ST_ClosestPoint — Renvoie le point 2D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne la plus courte d'une géométrie à l'autre.

Synopsis

geometry **ST_ClosestPoint**(geometry geom1, geometry geom2);
geography **ST_ClosestPoint**(geography geom1, geography geom2, boolean use_spheroid = true);

Description

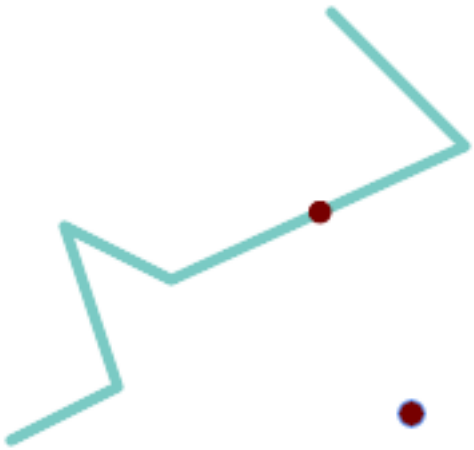
Renvoie le point bidimensionnel sur `geom1` qui est le plus proche de `geom2`. Il s'agit du premier point de la ligne la plus courte entre les géométries (telle que calculée par ST_ShortestLine).

Note
Si vous avez une géométrie 3D, vous préférerez peut-être utiliser ST_3DClosestPoint.

Amélioré : 3.4.0 - Prise en charge de la geography.

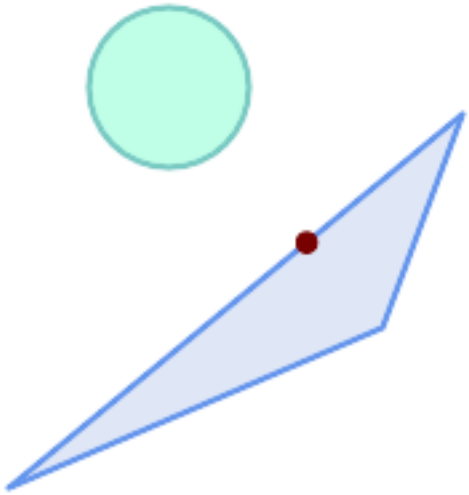
Disponibilité : 1.5.0

Exemples



Le point le plus proche d'un point et d'une LineString est le point lui-même. Le point le plus proche d'une LineString et d'un Point est un point de la ligne.

```
SELECT ST_AsText( ST_ClosestPoint(pt,line)) AS cp_pt_line,
       ST_AsText( 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;

   cp_pt_line    |                cp_line_pt
-----------------+-----------------------------------------
 POINT(160 40)   | POINT(125.75342465753425 115.34246575342466)
```



Le point du polygone A le plus proche du polygone B

```
SELECT ST_AsText( ST_ClosestPoint(
                'POLYGON ((190 150, 20 10, 160 70, 190 150))',
                ST_Buffer('POINT(80 160)', 30)        )) As ptwkt;
------------------------------------------
 POINT(131.59149149528952 101.89887534906197)
```

Voir aussi

ST_3DClosestPoint, ST_Distance, ST_LongestLine, ST_ShortestLine, ST_MaxDistance

7.12.5 ST_3DClosestPoint

ST_3DClosestPoint — Renvoie le point 3D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne 3D la plus courte.

Synopsis

geometry **ST_3DClosestPoint**(geometry g1, geometry g2);

Description

Renvoie le point tridimensionnel de g1 le plus proche de g2. Il s'agit du premier point de la ligne 3D la plus courte. La longueur 3D de la ligne 3D la plus courte est la distance 3D.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Disponibilité : 2.0.0

Modifié : 2.2.0 - si 2 géométries 2D sont saisies, un point 2D est renvoyé (au lieu de l'ancien comportement supposant 0 pour Z manquant). Dans le cas de 2D et 3D, Z n'est plus supposé être 0 pour Z manquant.

Exemples

ligne et point -- point le plus proche en 3d et 2d

```
SELECT ST_AsEWKT(ST_3DClosestPoint(line,pt)) AS cp3d_line_pt,
                ST_AsEWKT(ST_ClosestPoint(line,pt)) As cp2d_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;


 cp3d_line_pt                                                    |                ↩
    cp2d_line_pt
-----------------------------------------------------------------+-----------------------------------
 POINT(54.6993798867619 128.935022917228 11.5475869506606) | POINT(73.0769230769231  ↩
    115.384615384615)
```

ligne et multipoint - point le plus proche en 3d et en 2d

```
SELECT ST_AsEWKT(ST_3DClosestPoint(line,pt)) AS cp3d_line_pt,
                ST_AsEWKT(ST_ClosestPoint(line,pt)) As cp2d_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;


                       cp3d_line_pt                          | cp2d_line_pt
-------------------------------------------------------------+--------------
 POINT(54.6993798867619 128.935022917228 11.5475869506606) | POINT(50 75)
```

Multilignes et polygones en 3d et 2d point le plus proche

```
SELECT ST_AsEWKT(ST_3DClosestPoint(poly, mline)) As cp3d,
    ST_AsEWKT(ST_ClosestPoint(poly, mline)) As cp2d
        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;
                   cp3d                    |     cp2d
-------------------------------------------+--------------
 POINT(39.993580415989 54.1889925532825 5) | POINT(20 40)
```

Voir aussi

ST_AsEWKT, ST_ClosestPoint, ST_3DDistance, ST_3DShortestLine

7.12.6 ST_Distance

ST_Distance — Renvoie la distance entre deux valeurs de geometry ou geography.

Synopsis

float **ST_Distance**(geometry g1, geometry g2);
float **ST_Distance**(geography geog1, geography geog2, boolean use_spheroid = true);

Description

Pour les types geometry, renvoie la distance cartésienne (planaire) minimale en 2D entre deux géométries, en unités projetées (unités de référence spatiales).

Pour les types geography, la valeur par défaut est la distance géodésique minimale entre deux géographies en mètres, calculée sur le sphéroïde déterminé par le SRID. Si `use_spheroid` est faux, un calcul sphérique plus rapide est utilisé.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.23

Cette méthode prend en charge les types Circular String et Curve.

Disponibilité : 1.5.0 La prise en charge du type geography a été introduite dans la version 1.5. Amélioration de la vitesse pour les géométries planaires afin de mieux gérer les géométries de grande taille ou à nombreux sommets

Amélioration : la version 2.1.0 a amélioré la vitesse pour le type geography. Voir Making Geography faster pour plus de détails.

Amélioration : 2.1.0 - la prise en charge des géométries courbes a été introduite.

Amélioration : 2.2.0 - mesure sur sphéroïde effectuée avec GeographicLib pour une meilleure précision et robustesse. Nécessite PROJ >= 4.9.0 pour profiter de la nouvelle fonctionnalité.

Modifié : 3.0.0 - ne dépend plus de SFCGAL.

Exemples de géométrie

Exemple de géométrie - unités en degrés planaires 4326 est le long lat WGS 84, les unités sont des degrés.

```
SELECT ST_Distance(
    'SRID=4326;POINT(-72.1235 42.3521)'::geometry,
    'SRID=4326;LINESTRING(-72.1260 42.45, -72.123 42.1546)'::geometry );
------------------
0.00150567726382282
```

Exemple de géométrie - unités en mètres (SRID : 3857, proportionnel aux pixels sur les cartes web populaires). Bien que la valeur soit erronée, les valeurs proches peuvent être comparées correctement, ce qui en fait un bon choix pour des algorithmes tels que KNN ou KMeans.

```
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) ) ↩
        ;
------------------
167.441410065196
```

Exemple de géométrie - unités en mètres (SRID : 3857 comme ci-dessus, mais corrigé par cos(lat) pour tenir compte de la distorsion)

```
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);
------------------
123.742351254151
```

Exemple de géométrie - unités en mètres (SRID : 26986 Massachusetts state plane meters) (plus précis pour le Massachusetts)

```
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)  ↩
        );
-----------------
123.797937878454
```

Exemple de géométrie - unités en mètres (SRID : 2163 US National Atlas Equal area) (moins précis)

```
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) ) ↩
        ;
------------------
126.664256056812
```

Exemples géographiques

Identique à l'exemple du type geometry, mais avec des unités en mètres - utiliser la sphère pour un calcul légèrement plus rapide et moins précis.

```
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  ;

  spheroid_dist    |   sphere_dist
-------------------+------------------
 123.802076746848 | 123.475736916397
```

Voir aussi

ST_3DDistance, ST_DWithin, ST_DistanceSphere, ST_DistanceSpheroid, ST_MaxDistance, ST_HausdorffDistance, ST_FrechetDistan
ST_Transform

7.12.7 ST_3DDistance

ST_3DDistance — Renvoie la distance cartésienne minimale en 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.

Synopsis

float **ST_3DDistance**(geometry g1, geometry g2);

Description

Renvoie la distance cartésienne minimale tridimensionnelle entre deux géométries en unités projetées (unités de référence spatiales).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette méthode implémente la spécification SQL/MM. SQL-MM ISO/IEC 13249-3

Disponibilité : 2.0.0

Modifié : 2.2.0 - Dans le cas de la 2D et de la 3D, Z n'est plus considéré comme égal à 0 en cas de Z manquant.

Modifié : 3.0.0 - Version SFCGAL supprimée

Exemples

```
-- Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point  ↩
    and line compared 2D point and line)
-- Note: currently no vertical datum support so Z is not transformed and assumed to be same ↩
     units as final.
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;

     dist_3d      |     dist_2d
------------------+-----------------
 127.295059324629 | 126.66425605671
```

```
-- Multilinestring and polygon both 3d and 2d distance
-- Same example as 3D closest point example
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;
      dist3d       | dist2d
-------------------+--------
 0.716635696066337 |      0
```

Voir aussi

ST_Distance, ST_3DClosestPoint, ST_3DDWithin, ST_3DMaxDistance, ST_3DShortestLine, ST_Transform

7.12.8 ST_DistanceSphere

ST_DistanceSphere — Renvoie la distance minimale en mètres entre deux géométries lon/lat en utilisant un modèle de terre sphérique.

Synopsis

float **ST_DistanceSphere**(geometry geomlonlatA, geometry geomlonlatB, float8 radius=6371008);

Description

Renvoie la distance minimale en mètres entre deux points lon/lat. Utilise une terre sphérique et un rayon dérivé du sphéroïde défini par le SRID. Plus rapide que ST_DistanceSpheroid, mais moins précis. PostGIS Versions antérieures à la version 1.5 uniquement pour les points.

Disponibilité : 1.5 - la prise en charge d'autres types de géométrie que les points a été introduite. Les versions précédentes ne fonctionnaient qu'avec des points.

Modifié : 2.2.0 Dans les versions antérieures, cette fonction s'appelait ST_Distance_Sphere

Exemples

```
SELECT round(CAST(ST_DistanceSphere(ST_Centroid(geom), ST_GeomFromText('POINT(-118 38) ↩
    ',4326)) As numeric),2) As dist_meters,
round(CAST(ST_Distance(ST_Transform(ST_Centroid(geom),32611),
                ST_Transform(ST_GeomFromText('POINT(-118 38)', 4326),32611)) As numeric),2) ↩
                    As dist_utm11_meters,
round(CAST(ST_Distance(ST_Centroid(geom), ST_GeomFromText('POINT(-118 38)', 4326)) As  ↩
    numeric),5) As dist_degrees,
round(CAST(ST_Distance(ST_Transform(geom,32611),
                ST_Transform(ST_GeomFromText('POINT(-118 38)', 4326),32611)) As numeric),2) ↩
                    As min_dist_line_point_meters
FROM
        (SELECT ST_GeomFromText('LINESTRING(-118.584 38.374,-118.583 38.5)', 4326) As geom) ↩
            as foo;
         dist_meters | dist_utm11_meters | dist_degrees | min_dist_line_point_meters
        -------------+-------------------+--------------+----------------------------
                70424.47 |          70438.00 |      0.72900 |                   65871.18
```

Voir aussi

ST_Distance, ST_DistanceSpheroid

7.12.9 ST_DistanceSpheroid

ST_DistanceSpheroid — Renvoie la distance minimale entre deux géométries lon/lat en utilisant un modèle de terre sphéroïdale.

Synopsis

float **ST_DistanceSpheroid**(geometry geomlonlatA, geometry geomlonlatB, spheroid measurement_spheroid=WGS84);

Description

Renvoie la distance minimale en mètres entre deux géométries lon/lat pour un sphéroïde donné. Voir l'explication des sphéroïdes donnée pour ST_LengthSpheroid.

Note
Cette fonction ne prend pas en compte le SRID de la géométrie. Elle suppose que les coordonnées de la géométrie sont basées sur le sphéroïde fourni.

Disponibilité : 1.5 - la prise en charge d'autres types de géométrie que les points a été introduite. Les versions précédentes ne fonctionnaient qu'avec des points.

Modifié : 2.2.0 Dans les versions précédentes, cette fonction était appelée ST_Distance_Sphéroïde

Exemples

```
SELECT round(CAST(
                ST_DistanceSpheroid(ST_Centroid(geom), ST_GeomFromText('POINT(-118 38) ↩
                    ',4326), 'SPHEROID["WGS 84",6378137,298.257223563]')
                        As numeric),2) As dist_meters_spheroid,
                round(CAST(ST_DistanceSphere(ST_Centroid(geom), ST_GeomFromText('POINT(-118 ↩
                     38)',4326)) As numeric),2) As dist_meters_sphere,
round(CAST(ST_Distance(ST_Transform(ST_Centroid(geom),32611),
                ST_Transform(ST_GeomFromText('POINT(-118 38)', 4326),32611)) As numeric),2) ↩
                     As dist_utm11_meters
FROM
        (SELECT ST_GeomFromText('LINESTRING(-118.584 38.374,-118.583 38.5)', 4326) As geom) ↩
            as foo;
 dist_meters_spheroid | dist_meters_sphere | dist_utm11_meters
----------------------+--------------------+-------------------
                       70454.92 |           70424.47 |          70438.00
```

Voir aussi

ST_Distance, ST_DistanceSphere

7.12.10 ST_FrechetDistance

ST_FrechetDistance — Renvoie la distance de Fréchet entre deux géométries.

Synopsis

float **ST_FrechetDistance**(geometry g1, geometry g2, float densifyFrac = -1);

Description

Implémente l'algorithme de calcul de la distance de Fréchet limitée aux points discrets pour les deux géométries, basé sur Computing Discrete Fréchet Distance. La distance de Fréchet est une mesure de similarité entre les courbes qui tient compte de l'emplacement et de l'ordre des points le long des courbes. Elle est donc souvent meilleure que la distance de Hausdorff.

Lorsque le paramètre optionnel densifyFrac est spécifié, cette fonction effectue une densification des segments avant de calculer la distance de Fréchet discrète. Le paramètre densifyFrac définit la fraction par laquelle chaque segment doit être densifié. Chaque segment sera divisé en un certain nombre de sous-segments de longueur égale, dont la fraction de la longueur totale est la plus proche de la fraction donnée.

Les unités sont celles du système de référence spatiale des géométries.

Note
L'implémentation actuelle ne prend en charge que les sommets en tant qu'emplacements discrets. Elle pourrait être étendue pour permettre l'utilisation d'une densité arbitraire de points.

Note
Plus la valeur de densifyFrac spécifiée est petite, plus la distance de Fréchet est précise. Mais le temps de calcul et l'utilisation de la mémoire augmentent avec le carré du nombre de sous-segments.

Effectué par le module GEOS.

Disponibilité : 2.4.0 - nécessite GEOS >= 3.7.0

Exemples

```
postgres=# SELECT st_frechetdistance('LINESTRING (0 0, 100 0)'::geometry, 'LINESTRING (0 0, ↩
     50 50, 100 0)'::geometry);
 st_frechetdistance
--------------------
   70.7106781186548
(1 row)
```

```
SELECT st_frechetdistance('LINESTRING (0 0, 100 0)'::geometry, 'LINESTRING (0 0, 50 50, 100 ↩
     0)'::geometry, 0.5);
 st_frechetdistance
--------------------
                 50
(1 row)
```

Voir aussi

ST_HausdorffDistance

7.12.11 ST_HausdorffDistance

ST_HausdorffDistance — Renvoie la distance de Hausdorff entre deux géométries.

Synopsis

float **ST_HausdorffDistance**(geometry g1, geometry g2);
float **ST_HausdorffDistance**(geometry g1, geometry g2, float densifyFrac);

Description

Renvoie la distance de Hausdorff entre deux géométries. La distance de Hausdorff est une mesure de la similarité ou de la dissemblance de deux géométries.

La fonction calcule en fait la "Discrete Hausdorff Distance". Il s'agit de la distance de Hausdorff calculée en des points discrets des géométries. Le paramètre *densifyFrac* peut être spécifié pour fournir une réponse plus précise en densifiant les segments avant de calculer la distance discrète de Hausdorff. Chaque segment est divisé en un certain nombre de sous-segments de longueur égale dont la fraction de la longueur du segment est la plus proche de la fraction donnée.

Les unités sont celles du système de référence spatiale des géométries.

Note
Cet algorithme n'est PAS équivalent à la distance standard de Hausdorff. Cependant, il calcule une approximation qui est correcte pour un large sous-ensemble de cas utiles. Un cas important est celui des lignes qui sont à peu près parallèles les unes aux autres et dont la longueur est à peu près égale. Il s'agit d'une métrique utile pour l'appariement des lignes.

Disponibilité : 1.5.0

Exemples

Distance de Hausdorff (rouge) et distance (jaune) entre deux lignes

```
SELECT ST_HausdorffDistance(geomA, geomB),
       ST_Distance(geomA, geomB)
    FROM (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) AS t;
 st_hausdorffdistance | st_distance
----------------------+-------------
    37.26206567625497 |          20
```

Exemple: Distance de Hausdorff avec densification.

```
SELECT ST_HausdorffDistance(
            'LINESTRING (130 0, 0 0, 0 150)'::geometry,
            'LINESTRING (10 10, 10 150, 130 10)'::geometry,
            0.5);
 ----------------------
          70
```

Exemple: Pour chaque bâtiment, trouvez la parcelle qui le représente le mieux. Tout d'abord, nous exigeons que la parcelle intersecte la géométrie du bâtiment. `DISTINCT ON` garantit que chaque bâtiment ne sera listé qu'une seule fois. `ORDER BY .. ST_HausdorffDistance` sélectionne la parcelle qui est la plus similaire au bâtiment.

```
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);
```

Voir aussi

ST_FrechetDistance

7.12.12 ST_Length

ST_Length — Renvoie la longueur 2D d'une géométrie linéaire.

Synopsis

float **ST_Length**(geometry a_2dlinestring);
float **ST_Length**(geography geog, boolean use_spheroid = true);

Description

Pour les types geometry : renvoie la longueur cartésienne 2D de la géométrie s'il s'agit d'une LineString, MultiLineString, ST_Curve, ST_MultiCurve. Pour les géométries aréolaires, 0 est renvoyé ; utilisez ST_Perimeter à la place. Les unités de longueur sont déterminées par le système de référence spatial de la géométrie.

Pour les types geography : le calcul est effectué en utilisant le calcul géodésique inverse. Les unités de longueur sont exprimées en mètres. Si PostGIS est compilé avec PROJ version 4.8.0 ou ultérieure, le sphéroïde est spécifié par le SRID, sinon il est exclusif à WGS84. Si `use_spheroid = false`, le calcul est basé sur une sphère au lieu d'un sphéroïde.

Actuellement, pour la géométrie, il s'agit d'un alias de ST_Length2D, mais cela pourrait changer pour prendre en charge des dimensions plus élevées.

Warning
Modifié : 2.0.0 Rupture -- dans les versions précédentes, appliquer ceci à un MULTI/POLYGONE de type geography donnait le périmètre du POLYGONE/MULTIPOLYGONE. Dans la version 2.0.0, cette fonction a été modifiée pour retourner 0 afin d'être en ligne avec le comportement de la géométrie. Veuillez utiliser ST_Perimeter si vous souhaitez obtenir le périmètre d'un polygone

Note
Pour geography, le calcul utilise par défaut un modèle sphéroïdal. Pour utiliser le calcul sphérique, plus rapide mais moins précis, utilisez ST_Length(gg,false) ;

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.5.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 7.1.2, 9.3.4

Disponibilité : 1.5.0 La prise en charge du type geography a été introduite dans la version 1.5.

Exemples de géométrie

Renvoie la longueur en pieds de la ligne. Notez que cette longueur est exprimée en pieds car EPSG:2249 correspond aux pieds du plan de l'État du Massachusetts

```
SELECT ST_Length(ST_GeomFromText('LINESTRING(743238 2967416,743238 2967450,743265 2967450,
743265.625 2967416,743238 2967416)',2249));

st_length
---------
 122.630744000095


--Transforming WGS 84 LineString to Massachusetts state plane meters
SELECT ST_Length(
        ST_Transform(
                ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45, -72.1240 42.45666,  ↩
                    -72.123 42.1546)'),
                26986
        )
```

```
);

st_length
---------
34309.4563576191
```

Exemples géographiques

Retourne la longueur d'une ligne de type geography avec le SRID WGS 84

```
-- the default calculation uses a spheroid
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;

 length_spheroid  |  length_sphere
------------------+------------------
 34310.5703627288 | 34346.2060960742
```

Voir aussi

ST_GeographyFromText, ST_GeomFromEWKT, ST_LengthSpheroid, ST_Perimeter, ST_Transform

7.12.13 ST_Length2D

ST_Length2D — Renvoie la longueur 2D d'une géométrie linéaire. Alias de `ST_Length`

Synopsis

float **ST_Length2D**(geometry a_2dlinestring);

Description

Renvoie la longueur 2D de la géométrie s'il s'agit d'une ligne ou d'une multiligne. C'est un alias de `ST_Length`

Voir aussi

ST_Length, ST_3DLength

7.12.14 ST_3DLength

ST_3DLength — Renvoie la longueur 3D d'une géométrie linéaire.

Synopsis

float **ST_3DLength**(geometry a_3dlinestring);

Description

Renvoie la longueur tridimensionnelle ou bidimensionnelle de la géométrie s'il s'agit d'une LineString ou d'une MultiLineString. Pour les lignes à 2 dimensions, la longueur à 2 dimensions est renvoyée (comme ST_Length et ST_Length2D)

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 7.1, 10.3

Modifié : 2.0.0 Dans les versions précédentes, cette fonction était appelée ST_Length3D

Exemples

Longueur de retour en pieds pour un câble 3D. Notez que cette longueur est exprimée en pieds car EPSG:2249 correspond aux pieds du plan de l'État du Massachusetts

```
SELECT ST_3DLength(ST_GeomFromText('LINESTRING(743238 2967416 1,743238 2967450 1,743265  ↩
    2967450 3,
743265.625 2967416 3,743238 2967416 3)',2249));
ST_3DLength
-----------
122.704716741457
```

Voir aussi

ST_Length, ST_Length2D

7.12.15 ST_LengthSpheroid

ST_LengthSpheroid — Renvoie la longueur/périmètre 2D ou 3D d'une géométrie lon/lat sur un sphéroïde.

Synopsis

float **ST_LengthSpheroid**(geometry a_geometry, spheroid a_spheroid);

Description

Calcule la longueur ou le périmètre d'une géométrie sur un ellipsoïde. Ceci est utile si les coordonnées de la géométrie sont en longitude/latitude et qu'une longueur est souhaitée sans reprojection. Le sphéroïde est spécifié par une valeur textuelle comme suit :

```
SPHEROID[<NAME
>,<SEMI-MAJOR AXIS
>,<INVERSE FLATTENING
>]
```

Par exemple :

```
SPHEROID["GRS_1980",6378137,298.257222101]
```

Disponibilité : 1.2.2

Modifié : 2.2.0 Dans les versions précédentes, cette fonction s'appelait ST_Length_Spheroid et avait l'alias ST_3DLength_Spheroid

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_LengthSpheroid( geometry_column,
                          'SPHEROID["GRS_1980",6378137,298.257222101]' )
                          FROM geometry_table;

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 (SELECT ST_GeomFromText('MULTILINESTRING((-118.584  ↩
                              38.374,-118.583 38.5),
        (-71.05957 42.3589 , -71.061 43))') As geom,
CAST('SPHEROID["GRS_1980",6378137,298.257222101]' As spheroid) As sph_m)  as foo;
        tot_len      |    len_line1     |    len_line2
-----------------+------------------+-----------------
 85204.5207562955 | 13986.8725229309 | 71217.6482333646

 --3D
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 (SELECT ST_GeomFromEWKT('MULTILINESTRING((-118.584 38.374  ↩
                              20,-118.583 38.5 30),
        (-71.05957 42.3589 75, -71.061 43 90))') As geom,
CAST('SPHEROID["GRS_1980",6378137,298.257222101]' As spheroid) As sph_m)  as foo;

         tot_len      |    len_line1    |    len_line2
------------------+-----------------+------------------
 85204.5259107402 | 13986.876097711 | 71217.6498130292
```

Voir aussi

ST_GeometryN, ST_Length

7.12.16 ST_LongestLine

ST_LongestLine — Renvoie la ligne 2D la plus longue entre deux géométries.

Synopsis

geometry **ST_LongestLine**(geometry g1, geometry g2);

Description

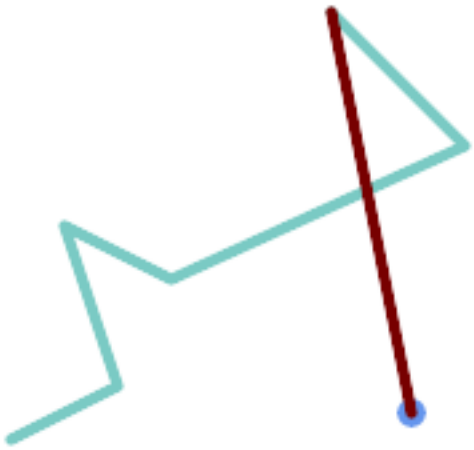
Renvoie la plus longue ligne bidimensionnelle entre les points de deux géométries. La ligne renvoyée commence sur `g1` et se termine sur `g2`.

La ligne la plus longue se trouve toujours entre deux sommets. La fonction renvoie la première ligne la plus longue si plusieurs sont trouvées. La longueur de la ligne est égale à la distance renvoyée par ST_MaxDistance.

Si g1 et g2 sont la même géométrie, renvoie la ligne entre les deux sommets les plus éloignés l'un de l'autre dans la géométrie. Les extrémités de la ligne se trouvent sur le cercle calculé par ST_MinimumBoundingCircle.

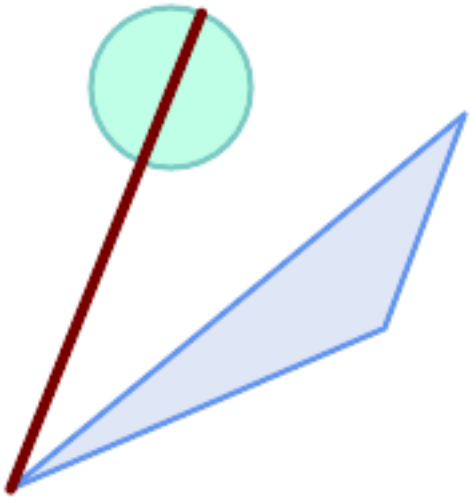
Disponibilité : 1.5.0

Exemples



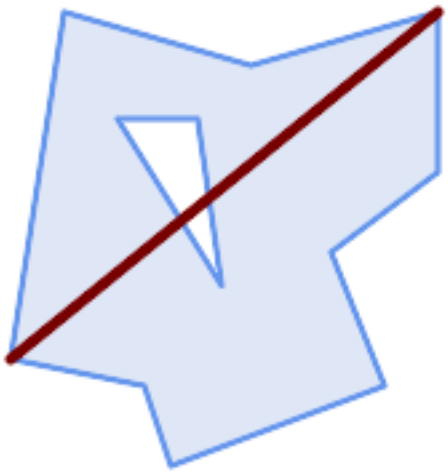
Ligne la plus longue entre un point et une ligne

```
SELECT ST_AsText( ST_LongestLine(
        'POINT (160 40)',
        'LINESTRING (10 30, 50 50, 30 110, 70 90, 180 140, 130 190)' )
        ) AS lline;
-----------------
LINESTRING(160 40,130 190)
```



Ligne la plus longue entre deux polygones

```
SELECT ST_AsText( ST_LongestLine(
        'POLYGON ((190 150, 20 10, 160 70, 190 150))',
        ST_Buffer('POINT(80 160)', 30)
            ) ) AS llinewkt;
-----------------
LINESTRING(20 10,105.3073372946034 186.95518130045156)
```



Ligne la plus longue traversant une seule géométrie. La longueur de la ligne est égale à la distance maximale. Les extrémités de la ligne sont situées sur le cercle de délimitation minimal.

```
SELECT ST_AsText( ST_LongestLine( geom, geom)) AS llinewkt,
                  ST_MaxDistance( geom, geom) AS max_dist,
                  ST_Length( ST_LongestLine(geom, 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;

         llinewkt          |      max_dist      |       lenll
---------------------------+--------------------+--------------------
 LINESTRING(20 50,180 180) | 206.15528128088303 | 206.15528128088303
```

Voir aussi

ST_MaxDistance, ST_ShortestLine, ST_3DLongestLine, ST_MinimumBoundingCircle

7.12.17 ST_3DLongestLine

ST_3DLongestLine — Renvoie la ligne 3D la plus longue entre deux géométries

Synopsis

geometry **ST_3DLongestLine**(geometry g1, geometry g2);

Description

Renvoie la ligne tridimensionnelle la plus longue entre deux géométries. La fonction renvoie la première ligne la plus longue s'il y en a plusieurs. La ligne retournée commence en g1 et se termine en g2. La longueur 3D de la ligne est égale à la distance renvoyée par ST_3DMaxDistance.

Disponibilité : 2.0.0

Modifié : 2.2.0 - si 2 géométries 2D sont saisies, un point 2D est renvoyé (au lieu de l'ancien comportement supposant 0 pour Z manquant). Dans le cas de 2D et 3D, Z n'est plus supposé être 0 pour Z manquant.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

linestring et point -- ligne la plus longue 3d et 2d

```
SELECT ST_AsEWKT(ST_3DLongestLine(line,pt)) AS lol3d_line_pt,
                ST_AsEWKT(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;


           lol3d_line_pt           |       lol2d_line_pt
-----------------------------------+---------------------------
 LINESTRING(50 75 1000,100 100 30) | LINESTRING(98 190,100 100)
```

ligne et multipoint - ligne la plus longue en 3d et 2d

```
SELECT ST_AsEWKT(ST_3DLongestLine(line,pt)) AS lol3d_line_pt,
                ST_AsEWKT(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;


          lol3d_line_pt          |      lol2d_line_pt
---------------------------------+-------------------------
 LINESTRING(98 190 1,50 74 1000) | LINESTRING(98 190,50 74)
```

MultiLineString et Polygon ligne la plus longue 3d et 2d

```
SELECT ST_AsEWKT(ST_3DLongestLine(poly, mline)) As lol3d,
    ST_AsEWKT(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;
            lol3d             |          lol2d
------------------------------+-------------------------
 LINESTRING(175 150 5,1 10 2) | LINESTRING(175 150,1 10)
```

Voir aussi

ST_3DClosestPoint, ST_3DDistance, ST_LongestLine, ST_3DShortestLine, ST_3DMaxDistance

7.12.18 ST_MaxDistance

ST_MaxDistance — Renvoie la plus grande distance 2D entre deux géométries en unités projetées.

Synopsis

float **ST_MaxDistance**(geometry g1, geometry g2);

Description

Renvoie la distance maximale en 2D entre deux géométries, en unités projetées. La distance maximale se situe toujours entre deux sommets. C'est la longueur de la ligne renvoyée par ST_LongestLine.

Si g1 et g2 sont la même géométrie, renvoie la distance entre les deux sommets les plus éloignés dans cette géométrie.

Disponibilité : 1.5.0

Exemples

Distance maximale entre un point et des lignes.

```
SELECT ST_MaxDistance('POINT(0 0)'::geometry, 'LINESTRING ( 2 0, 0 2 )'::geometry);
-----------------
 2

SELECT ST_MaxDistance('POINT(0 0)'::geometry, 'LINESTRING ( 2 2, 2 2 )'::geometry);
------------------
 2.82842712474619
```

Distance maximale entre les sommets d'une même géométrie.

```
SELECT ST_MaxDistance('POLYGON ((10 10, 10 0, 0 0, 10 10))'::geometry,
                      'POLYGON ((10 10, 10 0, 0 0, 10 10))'::geometry);
------------------
 14.142135623730951
```

Voir aussi

ST_Distance, ST_LongestLine, ST_DFullyWithin

7.12.19 ST_3DMaxDistance

ST_3DMaxDistance — Renvoie la distance maximale cartésienne 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.

Synopsis

float **ST_3DMaxDistance**(geometry g1, geometry g2);

Description

Renvoie la distance cartésienne maximale tridimensionnelle entre deux géométries en unités projetées (unités de référence spatiales).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Disponibilité : 2.0.0

Modifié : 2.2.0 - Dans le cas de la 2D et de la 3D, Z n'est plus considéré comme égal à 0 en cas de Z manquant.

Exemples

```
-- Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point ↩
    and line compared 2D point and line)
-- Note: currently no vertical datum support so Z is not transformed and assumed to be same ↩
     units as final.
SELECT ST_3DMaxDistance(
                        ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 ↩
                            10000)'),2163),
                        ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 ↩
                            15, -72.123 42.1546 20)'),2163)
                ) As dist_3d,
                ST_MaxDistance(
                        ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 ↩
                            10000)'),2163),
                        ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 ↩
                            15, -72.123 42.1546 20)'),2163)
                ) As dist_2d;

     dist_3d      |     dist_2d
------------------+------------------
 24383.7467488441 | 22247.8472107251
```

Voir aussi

ST_Distance, ST_3DDWithin, ST_3DMaxDistance, ST_Transform

7.12.20 ST_MinimumClearance

ST_MinimumClearance — Renvoie la clearance (le dégagement) d'une géométrie, une mesure de la robustesse d'une géométrie.

Synopsis

float **ST_MinimumClearance**(geometry g);

Description

Il est possible qu'une géométrie réponde aux critères de validité selon ST_IsValid (polygones) ou ST_IsSimple (lignes), mais qu'elle devienne invalide si l'un de ses sommets est déplacé d'une petite distance. Cela peut se produire en raison d'une perte de précision lors de la conversion vers des formats texte (tels que WKT, KML, GML, GeoJSON) ou des formats binaires qui n'utilisent pas de coordonnées en virgule flottante à double précision (par exemple, MapInfo TAB).

La clearance (le dégagement) minimum est une mesure quantitative de la résistance d'une géométrie aux changements de précision des coordonnées. Il s'agit de la plus grande distance à laquelle les sommets de la géométrie peuvent être déplacés sans créer une géométrie invalide. Des valeurs plus élevées de l'espace libre minimum indiquent une plus grande robustesse.

Si une géométrie a un dégagement minimal de `e`, alors.. :

- Aucun des deux sommets distincts de la géométrie n'est plus proche que la distance `e`.
- Aucun sommet n'est plus proche que `e` d'un segment de ligne dont il n'est pas l'extrémité.

S'il n'existe pas de dégagement minimum pour une géométrie (par exemple un point unique ou un MultiPoint dont les points sont identiques), la valeur de retour est `Infinity`.

Pour éviter les problèmes de validité causés par la perte de précision, ST_ReducePrecision peut réduire la précision des coordonnées tout en s'assurant que la géométrie polygonale reste valide.

Disponibilité : 2.3.0

Exemples

```
SELECT ST_MinimumClearance('POLYGON ((0 0, 1 0, 1 1, 0.5 3.2e-4, 0 0))');
 st_minimumclearance
---------------------
             0.00032
```

Voir aussi

ST_MinimumClearanceLine, ST_IsSimple, ST_IsValid, ST_ReducePrecision

7.12.21 ST_MinimumClearanceLine

ST_MinimumClearanceLine — Renvoie la chaîne de lignes à deux points couvrant le dégagement (clearance) minimum d'une géométrie.

Synopsis

Geometry **ST_MinimumClearanceLine**(geometry g);

Description

Renvoie les 2 points d'une LineString couvrant le dégagement (clearance) minimum d'une géométrie. Si la géométrie n'a pas de dégagement (clearance) minimum, `LINESTRING EMPTY` est retourné.

Effectué par le module GEOS.

Disponibilité : 2.3.0 - nécessite GEOS >= 3.6.0

Exemples

```
SELECT ST_AsText(ST_MinimumClearanceLine('POLYGON ((0 0, 1 0, 1 1, 0.5 3.2e-4, 0 0))'));
------------------------------
LINESTRING(0.5 0.00032,0.5 0)
```

Voir aussi

ST_MinimumClearance

7.12.22 ST_Perimeter

ST_Perimeter — Renvoie la longueur de la limite d'une géométrie polygonale ou d'une géographie.

Synopsis

float **ST_Perimeter**(geometry g1);
float **ST_Perimeter**(geography geog, boolean use_spheroid = true);

Description

Renvoie le périmètre 2D d'une geometry/geography s'il s'agit d'une ST_Surface, ST_MultiSurface (Polygone, MultiPolygone). 0 est retourné pour les géométries non réelles. Pour les géométries linéaires, utiliser ST_Length. Pour les types geometry, les unités de mesure du périmètre sont spécifiées par le système de référence spatiale de la géométrie.

Pour les types geography, les calculs sont effectués en utilisant le problème géodésique inverse, où les unités de périmètre sont en mètres. Si PostGIS est compilé avec PROJ version 4.8.0 ou ultérieure, le sphéroïde est spécifié par le SRID, sinon il est exclusif à WGS84. Si `use_spheroid = false`, les calculs approcheront une sphère au lieu d'un sphéroïde.

Il s'agit actuellement d'un alias de ST_Perimeter2D, mais cela pourrait changer pour prendre en charge des dimensions plus élevées.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.5.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 8.1.3, 9.5.4

Disponibilité 2.0.0 : La prise en charge du type geography a été introduite

Exemple : Géométrie

Retourne le périmètre en pieds pour les polygones et les multi-polygones. Notez que ce périmètre est en pieds car EPSG:2249 est le plan de l'État du Massachusetts en pieds

```
SELECT ST_Perimeter(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450,743265 2967450,
743265.625 2967416,743238 2967416))', 2249));
st_perimeter
---------
 122.630744000095
(1 row)

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));
st_perimeter
---------
 845.227713366825
(1 row)
```

Exemples : Géographie

Renvoie le périmètre en mètres et en pieds pour les polygones et les multi-polygones. Notez qu'il s'agit de geography (WGS 84 long lat)

```
SELECT  ST_Perimeter(geog) As per_meters, ST_Perimeter(geog)/0.3048 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;

   per_meters    |      per_ft
-----------------+------------------
```

```
37.3790462565251 | 122.634666195949


-- MultiPolygon example --
SELECT  ST_Perimeter(geog) As per_meters, ST_Perimeter(geog,false) As per_sphere_meters,  ↩
    ST_Perimeter(geog)/0.3048 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;

    per_meters     | per_sphere_meters |      per_ft
-------------------+-------------------+------------------
 257.634283683311 |  257.412311446337 | 845.256836231335
```

Voir aussi

ST_GeogFromText, ST_GeomFromText, ST_Length

7.12.23 ST_Perimeter2D

ST_Perimeter2D — Renvoie le périmètre 2D d'une géométrie polygonale. Alias de `ST_Perimeter`.

Synopsis

float **ST_Perimeter2D**(geometry geomA);

Description

Renvoie le périmètre bidimensionnel d'une géométrie polygonale.

Note
Il s'agit actuellement d'un alias de ST_Perimeter. Dans les versions futures, ST_Perimeter pourrait renvoyer le périmètre de la dimension la plus élevée pour une géométrie. Cette possibilité est encore à l'étude

Voir aussi

ST_Perimeter

7.12.24 ST_3DPerimeter

ST_3DPerimeter — Renvoie le périmètre 3D d'une géométrie polygonale.

Synopsis

float **ST_3DPerimeter**(geometry geomA);

Description

Renvoie le périmètre tridimensionnel de la géométrie, s'il s'agit d'un polygone ou d'un multi-polygone. Si la géométrie est bidimensionnelle, le périmètre bidimensionnel est renvoyé.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode implémente la spécification SQL/MM. SQL-MM ISO/IEC 13249-3: 8.1, 10.5

Modifié : 2.0.0 Dans les versions antérieures, il s'appelait ST_Perimeter3D

Exemples

Périmètre d'un polygone légèrement surélevé en l'air dans l'état du Massachusetts, en pieds de plan

```
SELECT ST_3DPerimeter(geom), ST_Perimeter2d(geom), ST_Perimeter(geom) FROM
                        (SELECT ST_GeomFromEWKT('SRID=2249;POLYGON((743238 2967416 2,743238 ↩
                            2967450 1,
743265.625 2967416 1,743238 2967416 2))') As geom) As foo;

  ST_3DPerimeter   |  st_perimeter2d   |   st_perimeter
-------------------+-------------------+------------------
 105.465793597674 | 105.432997272188 | 105.432997272188
```

Voir aussi

ST_GeomFromEWKT, ST_Perimeter, ST_Perimeter2D

7.12.25 ST_ShortestLine

ST_ShortestLine — Renvoie la ligne 2D la plus courte entre deux géométries

Synopsis

geometry **ST_ShortestLine**(geometry geom1, geometry geom2);
geography **ST_ShortestLine**(geography geom1, geography geom2, boolean use_spheroid = true);

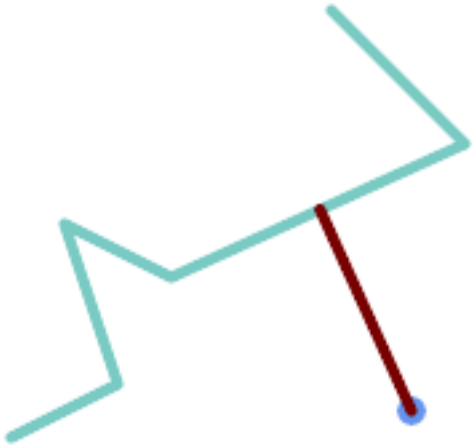
Description

Renvoie la ligne bidimensionnelle la plus courte entre deux géométries. La ligne renvoyée commence dans `geom1` et se termine dans `geom2`. Si `geom1` et `geom2` se croisent, le résultat est une ligne dont le début et la fin se situent à un point d'intersection. La longueur de la ligne est la même que celle que ST_Distance renvoie pour g1 et g2.

Amélioré : 3.4.0 - Prise en charge de la geography.

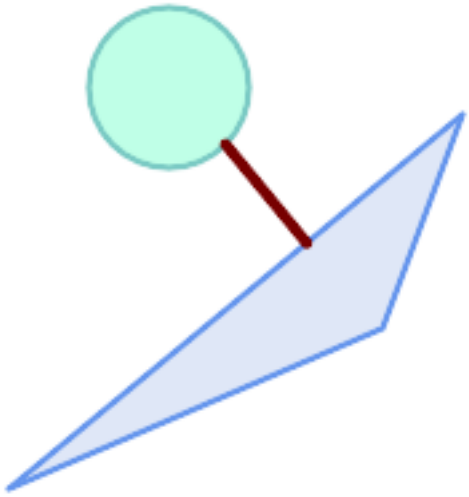
Disponibilité : 1.5.0

Exemples



Ligne la plus courte entre un point et une ligne

```
SELECT ST_AsText(  ST_ShortestLine(
        'POINT (160 40)',
        'LINESTRING (10 30, 50 50, 30 110, 70 90, 180 140, 130 190)')
        ) As sline;
-------------------------------------------------------
 LINESTRING(160 40,125.75342465753425 115.34246575342466)
```



Ligne la plus courte entre les polygones

```
SELECT ST_AsText( ST_ShortestLine(
         'POLYGON ((190 150, 20 10, 160 70, 190 150))',
         ST_Buffer('POINT(80 160)', 30)
              ) ) AS llinewkt;
-----------------
LINESTRING(131.59149149528952 101.89887534906197,101.21320343559644 138.78679656440357)
```

Voir aussi

ST_ClosestPoint, ST_Distance, ST_LongestLine, ST_MaxDistance

7.12.26 ST_3DShortestLine

ST_3DShortestLine — Renvoie la ligne 3D la plus courte entre deux géométries

Synopsis

geometry **ST_3DShortestLine**(geometry g1, geometry g2);

Description

Renvoie la ligne tridimensionnelle la plus courte entre deux géométries. La fonction ne renvoie que la première ligne la plus courte, si la fonction en trouve plusieurs. Si g1 et g2 se croisent en un seul point, la fonction renvoie une ligne dont le début et la fin se trouvent à ce point d'intersection. Si g1 et g2 se croisent en plus d'un point, la fonction renvoie une ligne dont le début et la fin se situent au même point, mais qui peut être n'importe lequel des points d'intersection. La ligne renvoyée commencera toujours en g1 et se terminera en g2. La longueur 3D de la ligne renvoyée par cette fonction sera toujours la même que ST_3DDistance renvoie pour g1 et g2.

Disponibilité : 2.0.0

Modifié : 2.2.0 - si 2 géométries 2D sont saisies, un point 2D est renvoyé (au lieu de l'ancien comportement supposant 0 pour Z manquant). Dans le cas de 2D et 3D, Z n'est plus supposé être 0 pour Z manquant.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Exemples

ligne et point -- ligne la plus courte 3d et 2d

```
SELECT ST_AsEWKT(ST_3DShortestLine(line,pt)) AS shl3d_line_pt,
                ST_AsEWKT(ST_ShortestLine(line,pt)) As shl2d_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;


 shl3d_line_pt                                                                 |  ↩
                  shl2d_line_pt
-------------------------------------------------------------------------------+-----------------------

 LINESTRING(54.6993798867619 128.935022917228 11.5475869506606,100 100 30)  |  ↩
    LINESTRING(73.0769230769231 115.384615384615,100 100)
```

ligne et multipoint -- ligne la plus courte en 3d et 2d

```
SELECT ST_AsEWKT(ST_3DShortestLine(line,pt)) AS shl3d_line_pt,
                ST_AsEWKT(ST_ShortestLine(line,pt)) As shl2d_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;


                        shl3d_line_pt                                          |  ↩
    shl2d_line_pt
-----------------------------------------------------------------------------------+----------------------

 LINESTRING(54.6993798867619 128.935022917228 11.5475869506606,100 100 30) | LINESTRING ↩
    (50 75,50 74)
```

MultiLineString et polygone -- ligne la plus courte 3d et 2d

```
SELECT ST_AsEWKT(ST_3DShortestLine(poly, mline)) As shl3d,
    ST_AsEWKT(ST_ShortestLine(poly, mline)) As shl2d
        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;
                   shl3d  ↩
                                                                                |     shl2d
-----------------------------------------------------------------------------------------------------------

 LINESTRING(39.993580415989 54.1889925532825 5,40.4078575708294 53.6052383805529  ↩
    5.03423778139177) | LINESTRING(20 40,20 40)
```

Voir aussi

ST_3DClosestPoint, ST_3DDistance, ST_LongestLine, ST_ShortestLine, ST_3DMaxDistance

7.13 Fonctions de superposition

7.13.1 ST_ClipByBox2D

ST_ClipByBox2D — Calcule la partie d'une géométrie située à l'intérieur d'un rectangle.

Synopsis

geometry **ST_ClipByBox2D**(geometry geom, box2d box);

Description

Coupe une géométrie par une boîte 2D d'une manière rapide et tolérante, mais qui peut être invalide. Les géométries d'entrée topologiquement invalides n'entraînent pas la levée d'exceptions. La géométrie de sortie n'est pas garantie valide (en particulier, des auto-intersections pour un polygone peuvent être introduites).

Effectué par le module GEOS.

Disponibilité : 2.2.0

Exemples

```
-- Rely on implicit cast from geometry to box2d for the second parameter
SELECT ST_ClipByBox2D(geom, ST_MakeEnvelope(0,0,10,10)) FROM mytab;
```

Voir aussi

ST_Intersection, ST_MakeBox2D, ST_MakeEnvelope

7.13.2 ST_Difference

ST_Difference — Calcule une géométrie représentant la partie de la géométrie A qui n'intersecte pas la géométrie B.

Synopsis

geometry **ST_Difference**(geometry geomA, geometry geomB, float8 gridSize = -1);

Description

Renvoie une géométrie représentant la partie de la géométrie A qui n'intersecte pas la géométrie B. Ceci est équivalent à `A - ST_Intersection(A,B)`. Si A est entièrement contenue dans B, une géométrie atomique vide du type approprié est renvoyée.

Note
Il s'agit de la seule fonction de superposition pour laquelle l'ordre d'entrée est important. ST_Difference(A, B) renvoie toujours une partie de A.

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a grid of the specified size. For the operation to give predictable results all the input vertices must fall already on the specified grid, see ST_ReducePrecision.

Effectué par le module GEOS

Amélioration : 3.1.0 accepte un paramètre gridSize.

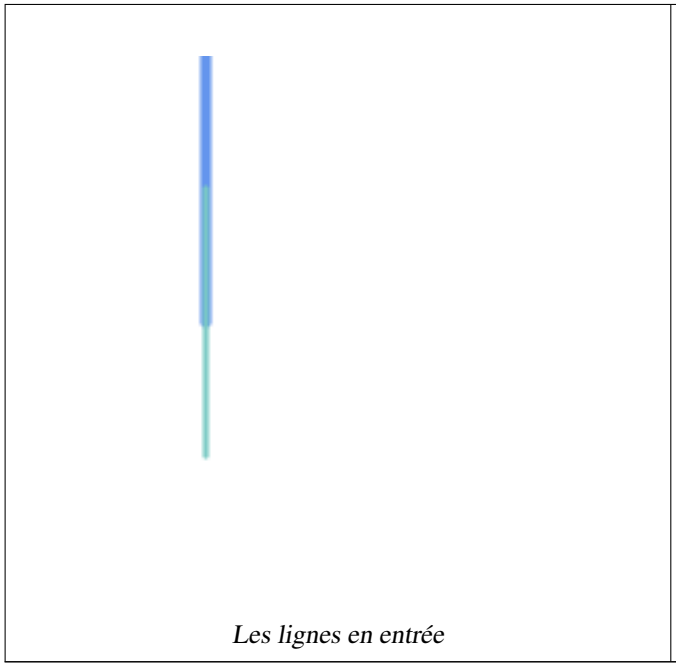
Nécessite GEOS >= 3.9.0 pour utiliser le paramètre gridSize.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.3

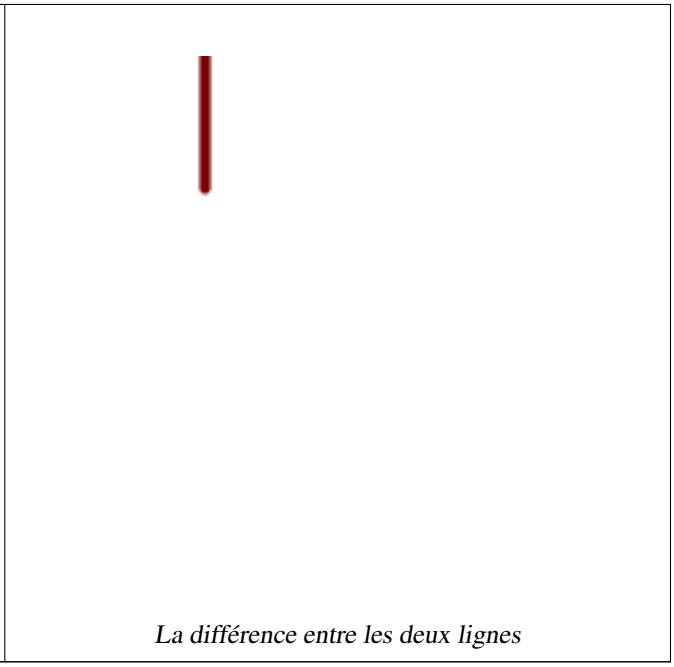
Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.20

Cette fonction prend en charge la 3D et ne supprime pas l'indice z. Cependant, le résultat est calculé en utilisant uniquement XY. Les valeurs Z résultantes sont copiées, moyennées ou interpolées.

Exemples



Les lignes en entrée



La différence entre les deux lignes

La différence entre les lignes 2D.

```
SELECT ST_AsText(
    ST_Difference(
            'LINESTRING(50 100, 50 200)'::geometry,
            'LINESTRING(50 50, 50 150)'::geometry
        )
    );

st_astext
---------
LINESTRING(50 150,50 200)
```

La différence entre les points 3D.

```
SELECT ST_AsEWKT( 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
                  ) );

st_asewkt
---------
MULTIPOINT(-118.6 38.329 6,-118.58 38.38 5)
```

Voir aussi

ST_SymDifference, ST_Intersection, ST_Union, ST_ReducePrecision

7.13.3 ST_Intersection

ST_Intersection — Calcule une géométrie représentant la partie partagée des géométries A et B.

Synopsis

geometry **ST_Intersection**(geometry geomA , geometry geomB , float8 gridSize = -1);
geography **ST_Intersection**(geography geogA , geography geogB);

Description

Renvoie une géométrie représentant l'intersection des points de deux géométries. En d'autres termes, la partie de la géométrie A et de la géométrie B qui est partagée entre les deux géométries.

Si les géométries n'ont aucun point commun (c'est-à-dire qu'elles sont disjointes), une géométrie atomique vide du type approprié est renvoyée.

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a grid of the specified size. For the operation to give predictable results all the input vertices must fall already on the specified grid, see ST_ReducePrecision.

ST_Intersection en conjonction avec ST_Intersects est utile pour découper les géométries, comme dans les requêtes sur les boîtes de délimitation, les tampons ou les régions, lorsque vous n'avez besoin que de la partie d'une géométrie qui se trouve à l'intérieur d'un pays ou d'une région d'intérêt.

Note
Pour le type geography, il s'agit d'un fin wrapper autour de l'implémentation de la géométrie. Il détermine d'abord le meilleur SRID qui correspond à la boîte de délimitation des 2 objets géographiques (si les objets géographiques sont dans une demi-zone UTM mais pas le même UTM choisira l'un d'eux) (en favorisant UTM ou Lambert Azimuthal Equal Area (LAEA) pôle nord/sud, et en se rabattant sur mercator dans le pire des cas) et ensuite l'intersection dans ce ref spatial planaire le mieux adapté et retransforme à nouveau à la géographie WGS84.

Warning
Cette fonction abandonne les valeurs de coordonnées M si elles sont présentes.

Warning
Si vous travaillez avec des géométries 3D, vous pouvez utiliser la fonction ST_3DIntersection basée sur SFGCAL qui réalise une intersection 3D correcte pour les géométries 3D. Bien que cette fonction fonctionne avec la coordonnée Z, elle effectue une moyenne de la coordonnée Z.

Effectué par le module GEOS

Amélioration : 3.1.0 accepte un paramètre gridSize

Nécessite GEOS >= 3.9.0 pour utiliser le paramètre gridSize

Modifié : 3.0.0 ne dépend pas de SFCGAL.

Disponibilité : La version 1.5 a introduit la prise en charge du type de données geography.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.3

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.18

Cette fonction prend en charge la 3D et ne supprime pas l'indice z. Cependant, le résultat est calculé en utilisant uniquement XY. Les valeurs Z résultantes sont copiées, moyennées ou interpolées.

Exemples

```
SELECT ST_AsText(ST_Intersection('POINT(0 0)'::geometry, 'LINESTRING ( 2 0, 0 2 )':: ↩
    geometry));
 st_astext
---------------
GEOMETRYCOLLECTION EMPTY

SELECT ST_AsText(ST_Intersection('POINT(0 0)'::geometry, 'LINESTRING ( 0 0, 0 2 )':: ↩
    geometry));
 st_astext
---------------
POINT(0 0)
```

Découper toutes les lignes (pistes) par pays. Nous supposons ici que les géométries des pays sont des POLYGONES ou des MULTIPOLYGONES. NOTE : nous ne gardons que les intersections qui résultent en un LINESTRING ou MULTILINESTRING car nous ne nous soucions pas des tracés qui partagent juste un point. Le dump est nécessaire pour étendre une collection géométrique en parties individuelles MULT*. La méthode ci-dessous est assez générique et fonctionnera pour les polys, etc. en changeant simplement la clause where.

```
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;
```

Pour les polys, par exemple les repères polygonaux, vous pouvez également utiliser la méthode parfois plus rapide qui consiste à mettre en mémoire tampon tout ce qui a une valeur de 0,0, à l'exception d'un polygone, afin d'obtenir une collection géométrique vide. (Ainsi, une collection géométrique contenant des polygones, des lignes et des points mis en mémoire tampon par 0.0 ne laisserait que les polygones et dissoudrait l'enveloppe de la collection.)

```
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));
```

Exemples : 2.5Dish

Notez qu'il ne s'agit pas d'une véritable intersection, comparez avec le même exemple en utilisant ST_3DIntersection.

```
select ST_AsText(ST_Intersection(linestring, polygon)) As wkt
from  ST_GeomFromText('LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)') AS  ↩
    linestring
 CROSS JOIN ST_GeomFromText('POLYGON((0 0 8, 0 1 8, 1 1 8, 1 0 8, 0 0 8))') AS polygon;

               st_astext
---------------------------------------
 LINESTRING Z (1 1 8,0.5 0.5 8,0 0 10)
```

Voir aussi

ST_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 — Fonction d'agrégation qui fusionne les géométries d'une manière efficace sur le plan de la mémoire mais plus lente

Synopsis

geometry **ST_MemUnion**(geometry set geomfield);

Description

Une fonction d'agrégation qui fusionne les géométries en entrée, en les fusionnant pour produire une géométrie de résultat sans chevauchement. Le résultat peut être une géométrie unique, une MultiGéométrie ou une Collection de Géométries.

Note
Produit le même résultat que ST_Union, mais utilise moins de mémoire et plus de temps processeur. Cette fonction d'agrégation fonctionne par l'union des géométries de manière incrémentale, contrairement à l'agrégation ST_Union qui accumule d'abord un tableau et en unit ensuite le contenu à l'aide d'un algorithme rapide.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z. Cependant, le résultat est calculé en utilisant uniquement XY. Les valeurs Z résultantes sont copiées, moyennées ou interpolées.

Exemples

```
SELECT id,
       ST_MemUnion(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

Voir aussi

ST_Union

7.13.5 ST_Node

ST_Node — Nœuds d'une collection de lignes.

Synopsis

geometry **ST_Node**(geometry geom);

Description

Renvoie une (Multi)LineString représentant la version entièrement "nodée" d'une collection de lignes. Le "noding" préserve tous les nœuds d'entrée et introduit le plus petit nombre possible de nouveaux nœuds. La ligne résultante est dissoute (les lignes en double sont supprimées).

Il s'agit d'un bon moyen de créer des lignes entièrement "nodées" pouvant être utilisées dans ST_Polygonize.

ST_UnaryUnion peut également être utilisé pour créer des nœuds et dissoudre des lignes. Il est possible de spécifier une taille de grille, ce qui permet d'obtenir des résultats plus simples et plus robustes. Voir aussi ST_Union pour une variante agrégée.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Effectué par le module GEOS.

Disponibilité : 2.0.0

Modifié : 2.4.0 cette fonction utilise GEOSNode en interne au lieu de GEOSUnaryUnion. Cela peut entraîner un ordre et une direction différents des lignes résultantes par rapport à PostGIS < 2.4.

Exemples

"Noding" d'une LineString 3D qui s'auto-intersecte

```
SELECT ST_AsText(
        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))
```

"Noding" de deux lignes qui partagent une structure linéaire commune. Notez que la ligne de résultat est dissoute.

```
SELECT ST_AsText(
        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))
```

Voir aussi

ST_UnaryUnion, ST_Union

7.13.6 ST_Split

ST_Split — Renvoie une collection de géométries créées en divisant une géométrie par une autre géométrie.

Synopsis

geometry **ST_Split**(geometry input, geometry blade);

Description

Cette fonction permet de diviser une ligne par un (Multi)Point, une (Multi)Ligne ou un (Multi)Polygone, ou un (Multi)Polygone par une ligne. Lorsqu'un (Multi)Polygone est utilisé, ses composantes linéaires (la frontière) sont utilisées pour diviser l'entrée. La géométrie résultante est toujours une collection.

Cette fonction est en quelque sorte l'inverse de ST_Union. L'application de ST_Union à la collection retournée devrait théoriquement donner la géométrie originale (bien qu'en raison de l'arrondi numérique, cela puisse ne pas être exactement le cas).

Note
Si l'entrée et la "lame" ne se croisent pas en raison de problèmes de précision numérique, l'entrée peut ne pas être divisée comme prévu. Pour éviter cette situation, il peut être nécessaire d'accrocher d'abord l'entrée à la "lame", en utilisant ST_Snap avec une petite tolérance.

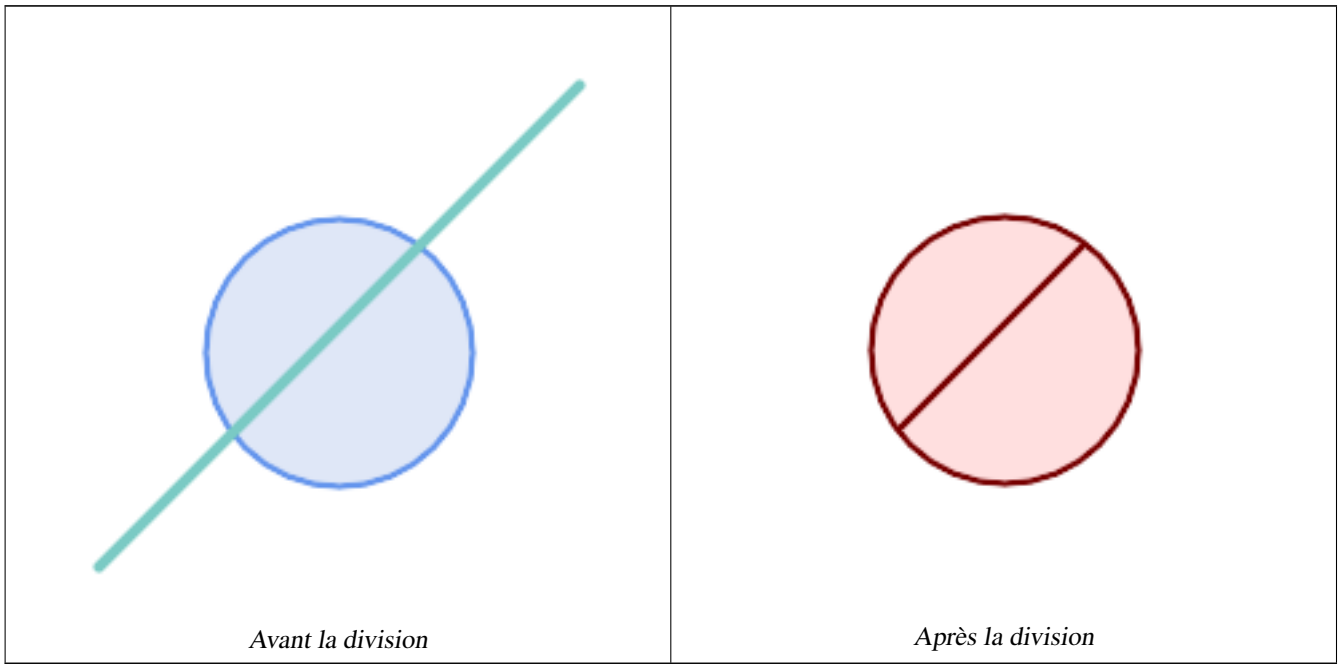
Disponibilité : 2.0.0 nécessite GEOS

Amélioration : la version 2.2.0 prend en charge la division d'une ligne par une limite multiligne, multipoint ou (multi)polygone.

Amélioration : la prise en charge de la division d'un polygone par une ligne multiple a été introduite dans la version 2.5.0.

Exemples

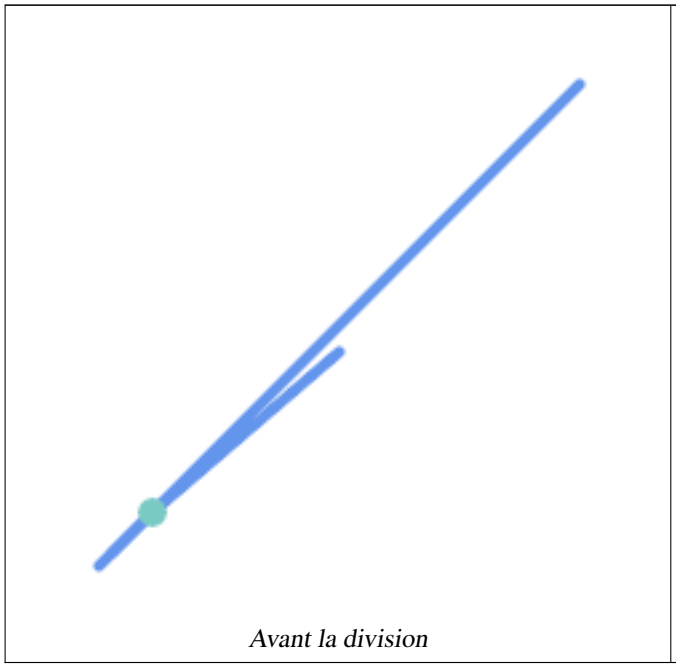
Diviser un polygone par une ligne.



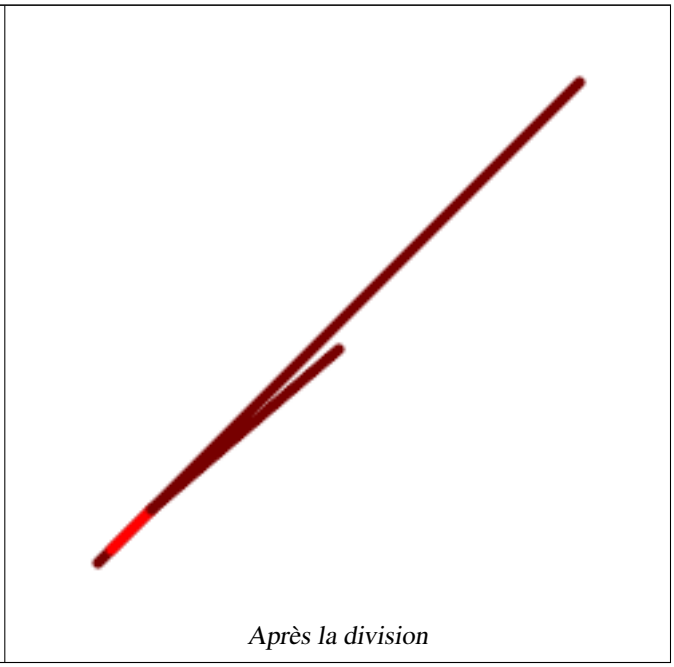
Avant la division | *Après la division*

```
SELECT ST_AsText( ST_Split(
                ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50), -- circle
                ST_MakeLine(ST_Point(10, 10),ST_Point(190, 190)) -- line
    ));

-- result --
 GEOMETRYCOLLECTION(
            POLYGON((150 90,149.039264020162 80.2454838991936,146.193976625564  ↩
                70.8658283817455,..),
            POLYGON(..))
)
```

Divise une MultiLineString par un point, où le point se trouve exactement sur les deux éléments de la LineString.



Avant la division



Après la division

```
SELECT ST_AsText(ST_Split(
    'MULTILINESTRING((10 10, 190 190), (15 15, 30 30, 100 90))',
    ST_Point(30,30))) As split;

split
------
GEOMETRYCOLLECTION(
    LINESTRING(10 10,30 30),
    LINESTRING(30 30,190 190),
    LINESTRING(15 15,30 30),
    LINESTRING(30 30,100 90)
)
```

Diviser une ligne par un point, lorsque le point ne se trouve pas exactement sur la ligne. Montre l'utilisation de ST_Snap pour accrocher la ligne au point afin de permettre son découpage.

```
WITH data AS (SELECT
  'LINESTRING(0 0, 100 100)'::geometry AS line,
  'POINT(51 50)':: geometry AS point
)
SELECT ST_AsText( ST_Split( ST_Snap(line, point, 1), point)) AS snapped_split,
       ST_AsText( ST_Split(line, point)) AS not_snapped_not_split
       FROM data;

                             snapped_split                                |             ↩
                                 not_snapped_not_split
------------------------------------------------------------------------+-------------------------------

 GEOMETRYCOLLECTION(LINESTRING(0 0,51 50),LINESTRING(51 50,100 100)) | GEOMETRYCOLLECTION( ↩
    LINESTRING(0 0,100 100))
```

Voir aussi

ST_Snap, ST_Union

7.13.7 ST_Subdivide

ST_Subdivide — Calcule une subdivision rectiligne d'une géométrie.

Synopsis

setof geometry **ST_Subdivide**(geometry geom, integer max_vertices=256, float8 gridSize = -1);

Description

Renvoie un ensemble de géométries résultant de la division de `geom` en parties à l'aide de lignes rectilignes, chaque partie ne contenant pas plus de `max_vertices`.

`max_vertices` must be 5 or more, as 5 points are needed to represent a closed box.

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a grid of the specified size. For the operation to give predictable results all the input vertices must fall already on the specified grid, see ST_ReducePrecision.

Les opérations "point dans polygone" et autres opérations spatiales sont normalement plus rapides pour les ensembles de données subdivisés et indexés. Étant donné que les boîtes de délimitation des parties couvrent généralement une zone plus petite que la boîte b de la géométrie d'origine, les requêtes d'index produisent moins de cas "hit". Les cas "hit" sont plus rapides car les opérations spatiales exécutées par la revérification de l'index traitent moins de points.

Note
Il s'agit d'une fonction de retour d'ensemble (SRF) qui renvoie un ensemble de lignes contenant des valeurs de géométrie uniques. Elle peut être utilisée dans une liste SELECT ou une clause FROM pour produire un ensemble de résultats avec un enregistrement pour chaque géométrie de résultat.

Effectué par le module GEOS.

Disponibilité : 2.2.0

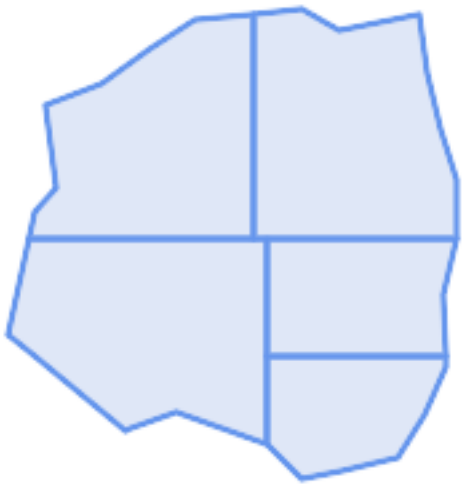
Amélioration : 2.5.0 réutilise les points existants lors de la division d'un polygone, le nombre de vertex est réduit de 8 à 5.

Amélioration : 3.1.0 accepte un paramètre gridSize.

Nécessite GEOS >= 3.9.0 pour utiliser le paramètre gridSize

Exemples

Exemple: Subdivisez un polygone en parties ne comportant pas plus de 10 sommets et attribuez à chaque partie un identifiant unique.



Subdivisé en un maximum de 10 sommets

```
SELECT row_number() OVER() As rn, ST_AsText(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);
```

```
rn &#x2502;                                                                    wkt
&#x2500;&#x2500;&#x2500;&#x2500;&#x253c;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2
  1 &#x2502; POLYGON((119 23,85 35,68 29,66 28,32 56,22 64,29.8260869565217 100,119 100,119 ↩
       23))
  2 &#x2502; POLYGON((132 10,119 23,119 56,186 56,186 52,178 34,168 18,147 13,132 10))
  3 &#x2502; POLYGON((119 56,119 100,190 100,185 79,186 56,119 56))
  4 &#x2502; POLYGON((29.8260869565217 100,32 110,40 119,36 150,57 158,75 171,92 182,114  ↩
      184,114 100,29.8260869565217 100))
  5 &#x2502; POLYGON((114 184,132 186,146 178,176 184,179 162,184 141,190 122,190 100,114  ↩
      100,114 184))
```

Exemple: Densifier une longue ligne de type geography en utilisant ST_Segmentize(geography, distance), et utiliser ST_Subdivide pour diviser la ligne résultante en sous-lignes de 8 sommets.

Les lignes densifiées et divisées.

```
SELECT ST_AsText( ST_Subdivide(
            ST_Segmentize('LINESTRING(0 0, 85 85)'::geography,
                          1200000)::geometry,    8));
```

```
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)
```

Exemple: Subdiviser les géométries complexes d'une table existante. Les enregistrements de la géométrie d'origine sont supprimés de la table source et de nouveaux enregistrements sont insérés pour chaque géométrie de résultat subdivisée.

```
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;
```

Exemple: Créer une nouvelle table contenant les géométries subdivisées, en conservant la clé de la géométrie d'origine afin que la nouvelle table puisse être jointe à la table source. Étant donné que ST_Subdivide est une fonction (tableau) qui renvoie un ensemble de lignes à valeur unique, cette syntaxe produit automatiquement un tableau avec une ligne pour chaque partie du résultat.

```
CREATE TABLE subdivided_geoms AS
    SELECT pkey, ST_Subdivide(geom) AS geom
    FROM original_geoms;
```

Voir aussi

ST_ClipByBox2D, ST_Segmentize, ST_Split, ST_NPoints, ST_ReducePrecision

7.13.8 ST_SymDifference

ST_SymDifference — Calcule une géométrie représentant les parties des géométries A et B qui ne s'intersectent pas.

Synopsis

geometry **ST_SymDifference**(geometry geomA, geometry geomB, float8 gridSize = -1);

Description

Renvoie une géométrie représentant les parties des géométries A et B qui ne s'intersectent pas. Ceci est équivalent à `ST_Union(A,B) - ST_Intersection(A,B)`. On parle de différence symétrique car `ST_SymDifférence(A,B) = ST_SymDifférence(B,`

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a grid of the specified size. For the operation to give predictable results all the input vertices must fall already on the specified grid, see ST_ReducePrecision.

Effectué par le module GEOS

Amélioration : 3.1.0 accepte un paramètre gridSize.

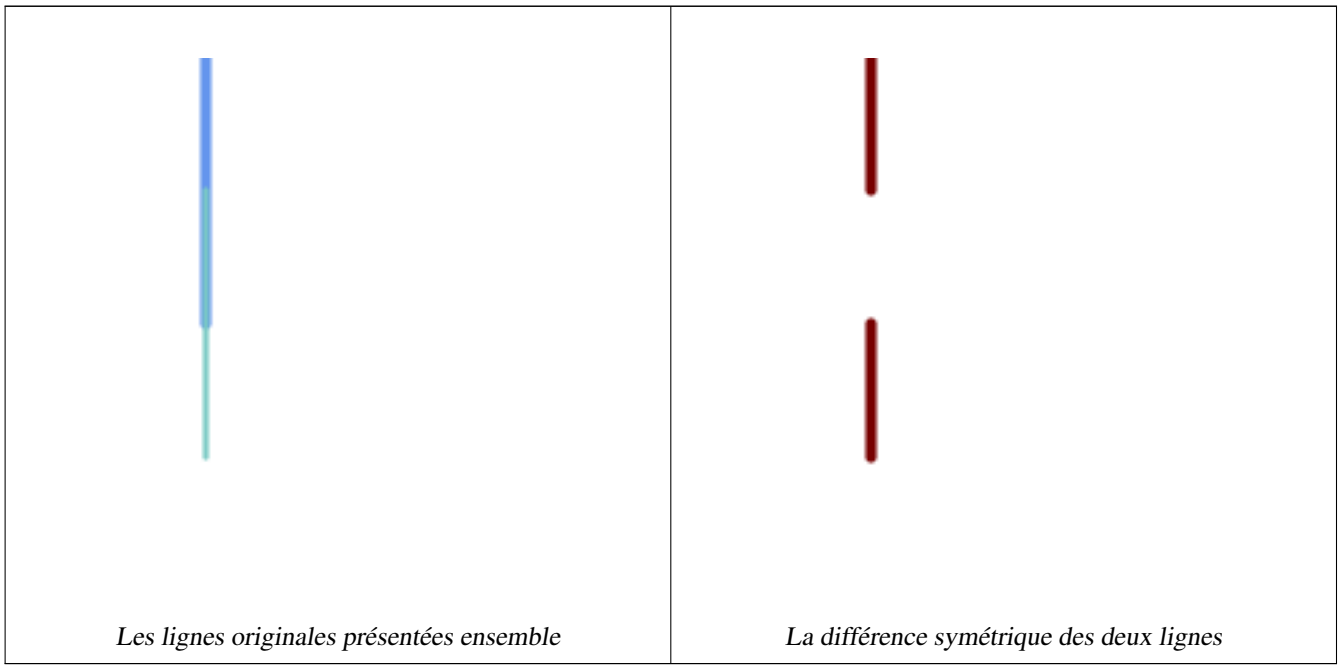
Nécessite GEOS >= 3.9.0 pour utiliser le paramètre gridSize

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.3

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 5.1.21

Cette fonction prend en charge la 3D et ne supprime pas l'indice z. Cependant, le résultat est calculé en utilisant uniquement XY. Les valeurs Z résultantes sont copiées, moyennées ou interpolées.

Exemples



Les lignes originales présentées ensemble

La différence symétrique des deux lignes

```
--Safe for 2d - symmetric difference of 2 linestrings
SELECT ST_AsText(
    ST_SymDifference(
        ST_GeomFromText('LINESTRING(50 100, 50 200)'),
        ST_GeomFromText('LINESTRING(50 50, 50 150)')
    )
);

st_astext
---------
MULTILINESTRING((50 150,50 200),(50 50,50 100))
```

```
--When used in 3d doesn't quite do the right thing
SELECT ST_AsEWKT(ST_SymDifference(ST_GeomFromEWKT('LINESTRING(1 2 1, 1 4 2)'),
    ST_GeomFromEWKT('LINESTRING(1 1 3, 1 3 4)')))

st_astext
------------
MULTILINESTRING((1 3 2.75,1 4 2),(1 1 3,1 2 2.25))
```

Voir aussi

ST_Difference, ST_Intersection, ST_Union, ST_ReducePrecision

7.13.9 ST_UnaryUnion

ST_UnaryUnion — Calcule l'union des composantes d'une seule géométrie.

Synopsis

geometry **ST_UnaryUnion**(geometry geom, float8 gridSize = -1);

Description

Variante à entrée unique de ST_Union. L'entrée peut être une géométrie unique, une MultiGeometry ou une GeometryCollection. L'union est appliquée aux éléments individuels de l'entrée.

Cette fonction peut être utilisée pour corriger les MultiPolygones qui ne sont pas valides en raison du chevauchement de leurs composants. Toutefois, les composants d'entrée doivent tous être valides. Un composant d'entrée non valide, tel qu'un polygone en nœud papillon, peut provoquer une erreur. Pour cette raison, il peut être préférable d'utiliser ST_MakeValid.

Une autre utilisation de cette fonction est de nouer et de dissoudre une collection de lignes qui se croisent ou se chevauchent pour les rendre simple. (ST_Node fait également cela, mais ne fournit pas l'option `gridSize`.)

Il est possible de combiner ST_UnaryUnion avec ST_Collect pour affiner le nombre de géométries à unir en une seule fois. Cela permet de faire un compromis entre l'utilisation de la mémoire et le temps de calcul, en trouvant un équilibre entre ST_Union et 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 grid of the specified size. For the operation to give predictable results all the input vertices must fall already on the specified grid, see ST_ReducePrecision.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z. Cependant, le résultat est calculé en utilisant uniquement XY. Les valeurs Z résultantes sont copiées, moyennées ou interpolées.

Amélioration : 3.1.0 accepte un paramètre gridSize.

Nécessite GEOS >= 3.9.0 pour utiliser le paramètre gridSize

Disponibilité : 2.0.0

Voir aussi

ST_Union, ST_MemUnion, ST_MakeValid, ST_Collect, ST_Node, ST_ReducePrecision

7.13.10 ST_Union

ST_Union — Calcule une géométrie représentant l'union des ensembles de points des géométries d'entrée.

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);

Description

Unifie les géométries d'entrée, fusionne les géométries pour produire une géométrie de résultat sans chevauchement. Le résultat peut être une géométrie atomique, une MultiGeometry ou une Geometry Collection. Il existe plusieurs variantes :

Variante à deux entrées: renvoie une géométrie qui est l'union de deux géométries d'entrée. Si l'une des entrées est NULL, NULL est renvoyé.

Variante tableau: renvoie une géométrie qui est l'union d'un tableau de géométries.

Variante agrégée: renvoie une géométrie qui est l'union d'un ensemble de géométries. La fonction ST_Union() est une fonction "agrégée" dans la terminologie de PostgreSQL. Cela signifie qu'elle opère sur des lignes de données, de la même manière que les fonctions SUM() et AVG() et, comme la plupart des agrégats, elle ignore les géométries NULL.

Voir ST_UnaryUnion pour une variante non agrégée, à entrée unique.

Le tableau ST_Union et ses variantes utilisent l'algorithme rapide d'union en cascade décrit dans 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 grid of the specified size. For the operation to give predictable results all the input vertices must fall already on the specified grid, see ST_ReducePrecision.

Note
ST_Collect peut parfois être utilisé à la place de ST_Union, si le résultat n'a pas besoin d'être non chevauchant. ST_Collect est généralement plus rapide que ST_Union car il n'effectue aucun traitement sur les géométries collectées.

Effectué par le module GEOS.

ST_Union crée une MultiLineString et ne fusionne pas les LineStrings en une seule LineString. Utilisez ST_LineMerge pour fusionner les LineStrings.

NOTE : cette fonction s'appelait auparavant GeomUnion(), qui a été renommée "Union" car UNION est un mot réservé de SQL.

Amélioration : 3.1.0 accepte un paramètre gridSize.

Nécessite GEOS >= 3.9.0 pour utiliser le paramètre gridSize

Modifié : 3.0.0 ne dépend pas de SFCGAL.

Disponibilité : 1.4.0 - ST_Union a été amélioré. ST_Union(geomarray) a été introduit ainsi qu'une collecte d'agrégats plus rapide dans PostgreSQL.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.3

Note
La version agrégée n'est pas explicitement définie dans l'OGC SPEC.

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 : 5.1.19 l'indice z (élévation) lorsque des polygones sont impliqués.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z. Cependant, le résultat est calculé en utilisant uniquement XY. Les valeurs Z résultantes sont copiées, moyennées ou interpolées.

Exemples

Exemple d'agrégat

```
SELECT id,
       ST_Union(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

Exemple non agrégé

```
select ST_AsText(ST_Union('POINT(1 2)' :: geometry, 'POINT(-2 3)' :: geometry))

st_astext
----------
MULTIPOINT(-2 3,1 2)

select ST_AsText(ST_Union('POINT(1 2)' :: geometry, 'POINT(1 2)' :: geometry))

st_astext
----------
POINT(1 2)
```

Exemple 3D - en quelque sorte des supports 3D (et avec des dimensions mixtes !)

```
select ST_AsEWKT(ST_Union(geom))
from (
         select 'POLYGON((-7 4.2,-7.1 4.2,-7.1 4.3, -7 4.2))'::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;

st_asewkt
---------
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)));
```

Exemple en 3D ne mélangeant pas les dimensions

```
select ST_AsEWKT(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;
```

```
st_asewkt
---------
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)))

--Examples using new Array construct
SELECT ST_Union(ARRAY(SELECT geom FROM sometable));

SELECT ST_AsText(ST_Union(ARRAY[ST_GeomFromText('LINESTRING(1 2, 3 4)'),
            ST_GeomFromText('LINESTRING(3 4, 4 5)')])) As wktunion;

--wktunion---
MULTILINESTRING((3 4,4 5),(1 2,3 4))
```

Voir aussi

ST_Collect, ST_UnaryUnion, ST_MemUnion, ST_Intersection, ST_Difference, ST_SymDifference, ST_ReducePrecision

7.14 Traitement des géométries

7.14.1 ST_Buffer

ST_Buffer — Calcule une géométrie couvrant tous les points situés à une distance donnée d'une géométrie.

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);

Description

Calcule un POLYGONE ou un MULTIPOLYGONE représentant tous les points dont la distance par rapport à une geometry/-geography est inférieure ou égale à une distance donnée. Une distance négative rétrécit la géométrie au lieu de l'étendre. Une distance négative peut réduire complètement un polygone, auquel cas POLYGON EMPTY est renvoyé. Pour les points et les lignes, les distances négatives renvoient toujours des résultats vides.

Pour les types geometry, la distance est spécifiée dans les unités du système de référence spatiale de la géométrie. Pour les types geography, la distance est spécifiée en mètres.

Le troisième paramètre facultatif contrôle la précision et le style du tampon. La précision des arcs de cercle dans le tampon est spécifiée en tant que nombre de segments de ligne utilisés pour approximer un quart de cercle (la valeur par défaut est de 8). Le style du tampon peut être spécifié en fournissant une liste de paires clé=valeur séparées par des blancs, comme suit :

- 'quad_segs=#' : nombre de segments de ligne utilisés pour approximer un quart de cercle (8 par défaut).
- 'endcap=round|flat|square' : style d'endcap (la valeur par défaut est "round"). 'butt' est accepté comme synonyme de 'flat'.
- 'join=round|mitre|bevel' : style de jointure (la valeur par défaut est "round"). 'miter' est accepté comme synonyme de 'mitre'.
- 'mitre_limit=#.#' : limite du rapport d'angle (n'affecte que le style d'assemblage en angle). 'miter_limit' est accepté comme synonyme de 'mitre_limit'.

- side=both|left|right' : 'left' ou 'right' effectue une mise en mémoire tampon unilatérale sur la géométrie, le côté mis en mémoire tampon étant relatif à la direction de la ligne. Ceci ne s'applique qu'à la géométrie LINESTRING et n'affecte pas les géométries POINT ou POLYGONE. Par défaut, les embouts sont carrés.

Note

Pour le type geography, il s'agit d'un fin wrapper autour de l'implémentation de la géométrie. Il détermine un système de référence spatiale planaire qui correspond le mieux à la boîte de délimitation de l'objet géographique (en essayant UTM, le pôle Nord/Sud de Lambert Azimuthal Equal Area (LAEA), et enfin Mercator). Le tampon est calculé dans l'espace planaire, puis retransformé en WGS84. Cela peut ne pas produire le comportement souhaité si l'objet d'entrée est beaucoup plus grand qu'une zone UTM ou s'il traverse la ligne de changement de date

---

Note

La sortie du tampon est toujours une géométrie polygonale valide. La mémoire tampon peut gérer des entrées non valides, de sorte que la mise en mémoire tampon par la distance 0 est parfois utilisée comme moyen de réparer les polygones non valides. ST_MakeValid peut également être utilisé à cette fin.

Note

La mise en mémoire tampon est parfois utilisée pour effectuer une recherche à l'intérieur de la distance. Dans ce cas, il est plus efficace d'utiliser ST_DWithin.

Note

Cette fonction ignore la dimension Z. Elle donne toujours un résultat en 2D, même lorsqu'elle est utilisée sur une géométrie en 3D.

Amélioration : 2.5.0 - La prise en charge de la géométrie ST_Buffer a été améliorée pour permettre la spécification de la mise en mémoire tampon latérale `side=both|left|right`.

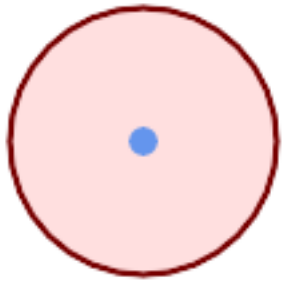
Disponibilité : 1.5 - ST_Buffer a été amélioré pour prendre en charge différents types de terminaisons et de jointures. Ceux-ci sont utiles, par exemple, pour convertir les lignes de route en routes polygonales avec des bords plats ou carrés au lieu de bords arrondis. Un petit wrapper pour la geography a été ajouté.

Effectué par le module GEOS.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.3

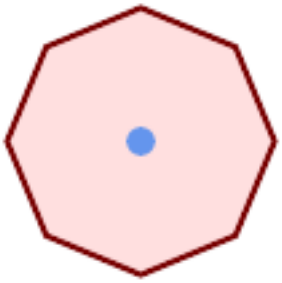
Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1.30

Exemples



quad_segs=8 (par défaut)

```
SELECT ST_Buffer(
 ST_GeomFromText('POINT(100 90)'),
 50, 'quad_segs=8');
```



quad_segs=2 (faible)

```
SELECT ST_Buffer(
 ST_GeomFromText('POINT(100 90)'),
 50, 'quad_segs=2');
```

endcap=round join=round (par défaut)

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'endcap=round join=round');
```

endcap=square

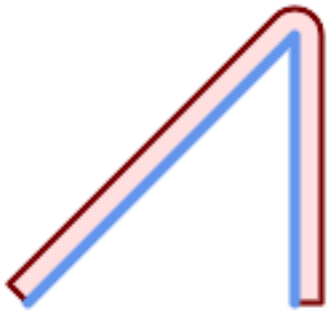
```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'endcap=square join=round');
```

join=bevel

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'join=bevel');
```

join=mitre mitre_limit=5.0 (limite de mitre par défaut)

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'join=mitre mitre_limit=5.0');
```

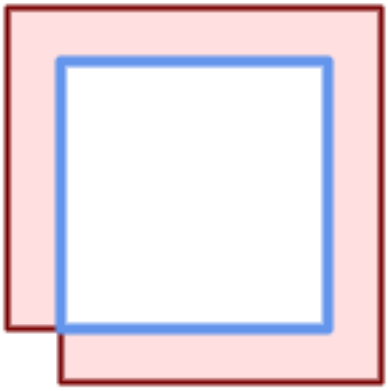


side=left

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'side=left');
```

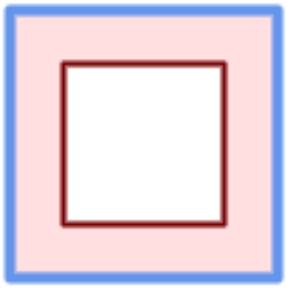
side=right

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'side=right');
```



enroulement à droite, limite du polygone side=left

```
SELECT ST_Buffer(
ST_ForceRHR(
ST_Boundary(
 ST_GeomFromText(
'POLYGON ((50 50, 50 150, 150 150, 150  ↩
    50, 50 50))'))),
 ), 20, 'side=left');
```



enroulement à droite, limite du polygone side=right

```
SELECT ST_Buffer(
ST_ForceRHR(
ST_Boundary(
 ST_GeomFromText(
'POLYGON ((50 50, 50 150, 150 150, 150  ↩
    50, 50 50))'))
), 20,'side=right')
```

```
--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)
SELECT ST_NPoints(ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50)) As  ↩
    promisingcircle_pcount,
ST_NPoints(ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50, 2)) As lamecircle_pcount;

promisingcircle_pcount | lamecircle_pcount
-----------------------+-------------------
                    33 |                 9

--A lighter but lamer circle
-- only 2 points per quarter circle is an octagon
--Below is a 100 meter octagon
-- Note coordinates are in NAD 83 long lat which we transform
to Mass state plane meter and then buffer to get measurements in meters;
SELECT ST_AsText(ST_Buffer(
ST_Transform(
ST_SetSRID(ST_Point(-71.063526, 42.35785),4269), 26986)
,100,2)) As octagon;
----------------------
POLYGON((236057.59057465 900908.759918696,236028.301252769 900838.049240578,235
957.59057465 900808.759918696,235886.879896532 900838.049240578,235857.59057465
900908.759918696,235886.879896532 900979.470596815,235957.59057465 901008.759918
696,236028.301252769 900979.470596815,236057.59057465 900908.759918696))
```

Voir aussi

ST_Collect, ST_DWithin, ST_SetSRID, ST_Transform, ST_Union, ST_MakeValid

7.14.2 ST_BuildArea

ST_BuildArea — Crée une géométrie polygonale formée par le tracé d'une géométrie.

Synopsis

geometry **ST_BuildArea**(geometry geom);

Description

Crée une géométrie aréolaire formée par les lignes constitutives de la géométrie d'entrée. L'entrée peut être une LineString, MultiLineString, Polygon, MultiPolygon ou une GeometryCollection. Le résultat est un Polygone ou un MultiPolygone, en fonction de l'entrée. Si la ligne d'entrée ne forme pas de polygone, NULL est renvoyé.

Contrairement à ST_MakePolygon, cette fonction accepte les anneaux formés par plusieurs lignes et peut former un nombre quelconque de polygones.

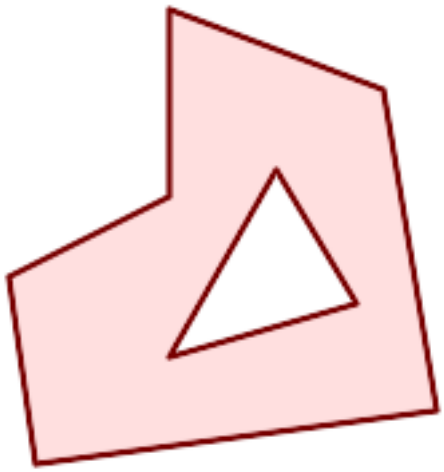
Cette fonction convertit les anneaux intérieurs en trous. Pour transformer les anneaux intérieurs en polygones, utilisez ST_Polygonize.

Note
Les lignes d'entrée doivent être correctement nouées pour que cette fonction fonctionne correctement. ST_Node peut être utilisé pour nouer des lignes.
Si le réseau de lignes d'entrée est croisé, cette fonction produira des polygones non valides. ST_MakeValid peut être utilisé pour s'assurer que la sortie est valide.

Disponibilité : 1.1.0

Exemples

Lignes en entrée



Résultat de la zone

```
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_AsText( ST_BuildArea( ST_Collect( geom )))
    FROM data;

------------------------------------------------------------------------------------------
POLYGON((180 40,30 20,20 90,80 120,80 190,160 160,180 40),(150 80,120 130,80 60,150 80))
```

Créer un donut à partir de deux polygones circulaires

```
SELECT ST_BuildArea(ST_Collect(inring,outring))
FROM (SELECT
    ST_Buffer('POINT(100 90)', 25) As inring,
    ST_Buffer('POINT(100 90)', 50) As outring) As t;
```

Voir aussi

ST_Collect, ST_MakePolygon, ST_MakeValid, ST_Node, ST_Polygonize, ST_BdPolyFromText, ST_BdMPolyFromText (wrappers de cette fonction avec l'interface standard de l'OGC)

7.14.3 ST_Centroid

ST_Centroid — Renvoie le centre géométrique d'une géométrie.

Synopsis

geometry **ST_Centroid**(geometry g1);
geography **ST_Centroid**(geography g1, boolean use_spheroid = true);

Description

Calcule un point qui est le centre de masse géométrique d'une géométrie. Pour [MULTI]POINTs, le centroïde est la moyenne arithmétique des coordonnées entrées. Pour les [MULTI]LINESTRINGs, le centroïde est calculé en utilisant la longueur pondérée de chaque segment de ligne. Pour les [MULTI]POLYGONs, le centroïde est calculé en termes de surface. Si une

géométrie vide est fournie, une GEOMETRYCOLLECTION vide est renvoyée. Si NULL est fourni, NULL est renvoyé. Si CIRCULARSTRING ou COMPOUNDCURVE sont fournis, ils sont convertis en linestring avec CurveToLine d'abord, puis de la même manière que pour LINESTRING

Pour les entrées de dimensions mixtes, le résultat est égal au centroïde de la composante Géométries de la dimension la plus élevée (puisque les géométries de dimension inférieure n'apportent aucun "poids" au centroïde).

Notez que pour les géométries polygonales, le centroïde n'est pas nécessairement situé à l'intérieur du polygone. Par exemple, voir le diagramme ci-dessous du centroïde d'un polygone en forme de C. Pour construire un point garanti à l'intérieur d'un polygone, utilisez ST_PointOnSurface.

Nouveau dans la version 2.3.0 : prend en charge CIRCULARSTRING et COMPOUNDCURVE (en utilisant CurveToLine)

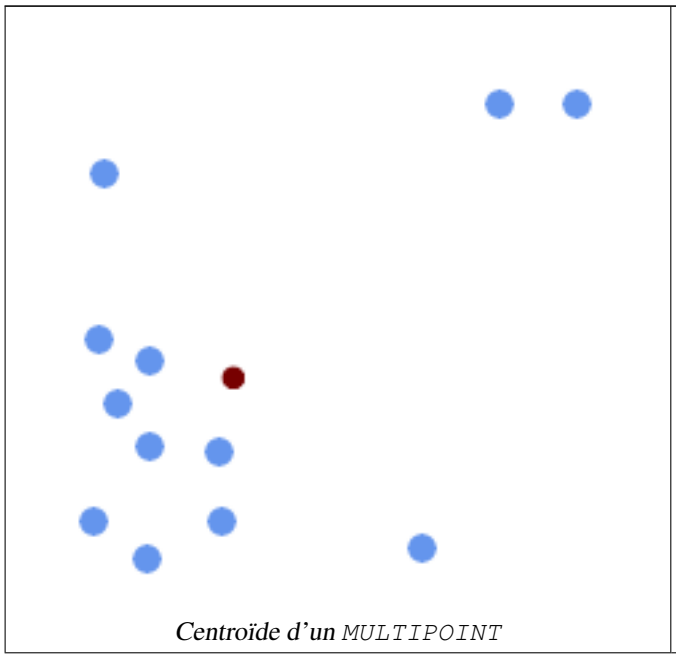
Disponibilité : la prise en charge du type geography a été introduite dans la version 2.4.0.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1.

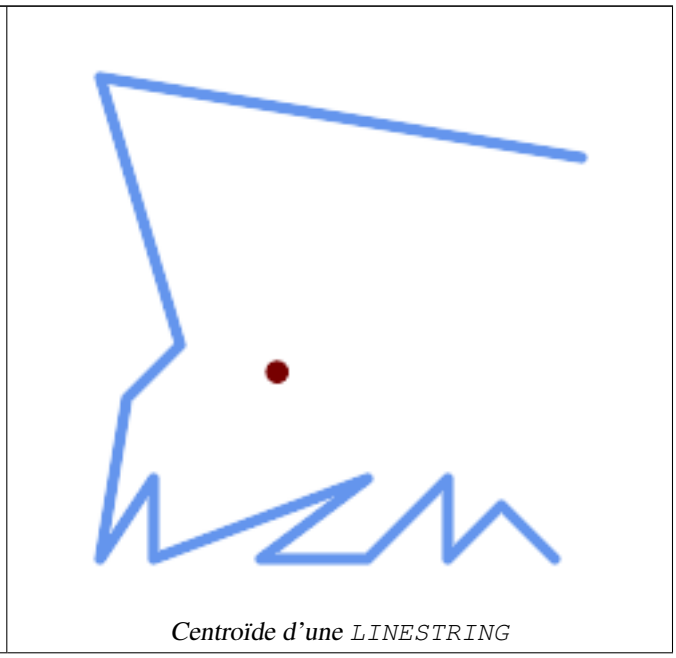
Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 8.1.4, 9.5.5

Exemples

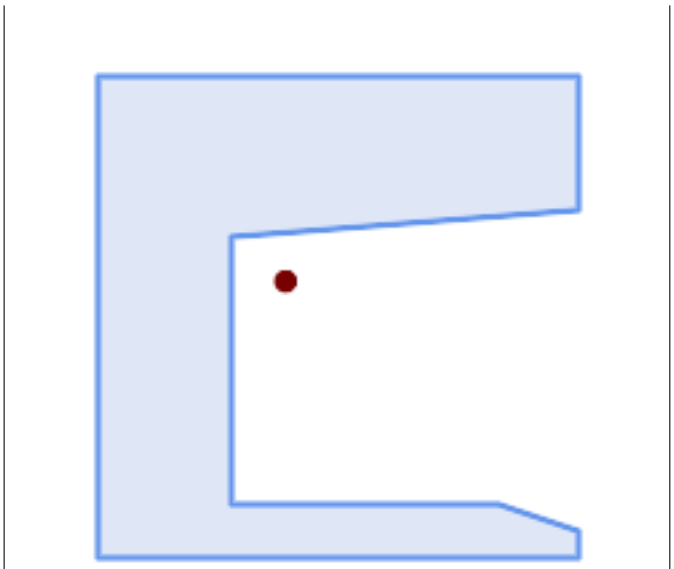
Dans les illustrations suivantes, le point rouge est le centroïde de la géométrie de la source.



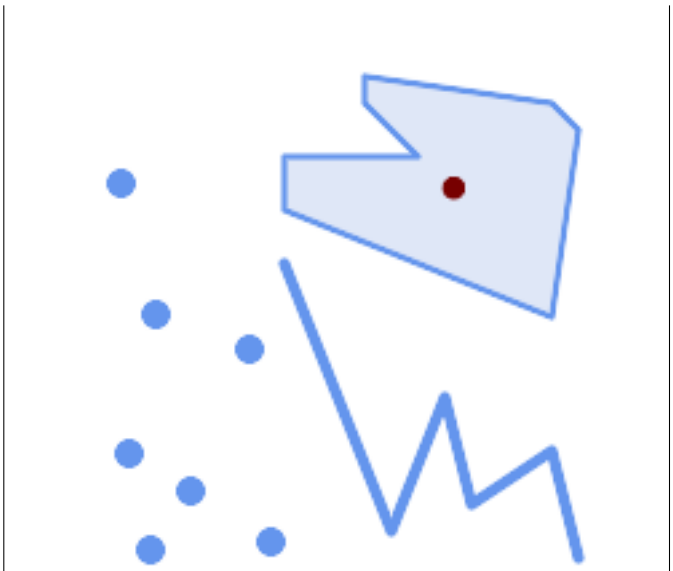
Centroïde d'un MULTIPOINT



Centroïde d'une LINESTRING



Centroïde d'un `POLYGON`



Centroïde d'une `GEOMETRYCOLLECTION`

```
SELECT ST_AsText(ST_Centroid('MULTIPOINT ( -1 0, -1 2, -1 3, -1 4, -1 7, 0 1, 0 3, 1 1, 2  ↩
    0, 6 0, 7 8, 9 8, 10 6 )'));
               st_astext
-----------------------------------------
 POINT(2.30769230769231 3.30769230769231)
(1 row)

SELECT ST_AsText(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 ;
----------------------------------------
POINT(0.5 1)


SELECT ST_AsText(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;
----------------------------------------
POINT(0.5 1)
```

Voir aussi

ST_PointOnSurface, ST_GeometricMedian

7.14.4 ST_ChaikinSmoothing

ST_ChaikinSmoothing — Renvoie une version lissée d'une géométrie, en utilisant l'algorithme Chaikin

Synopsis

geometry **ST_ChaikinSmoothing**(geometry geom, integer nIterations = 1, boolean preserveEndPoints = false);

Description

Lisse une géométrie linéaire ou polygonale en utilisant Chaikin's algorithm. Le degré de lissage est contrôlé par le paramètre `nIterations`. À chaque itération, chaque sommet intérieur est remplacé par deux sommets situés à 1/4 de la longueur des segments de ligne précédant et suivant le sommet. Un degré raisonnable de lissage est fourni par 3 itérations ; le maximum est limité à 5.

Si `preserveEndPoints` est égal à true, les extrémités des anneaux de polygone ne sont pas lissées. Les extrémités des lignes sont toujours préservées.

> **Note**
> Le nombre de sommets double à chaque itération, de sorte que la géométrie résultante peut comporter beaucoup plus de points que la géométrie d'entrée. Pour réduire le nombre de points, utilisez une fonction de simplification sur le résultat (voir ST_Simplify, ST_SimplifyPreserveTopology et ST_SimplifyVW).

Le résultat comporte des valeurs interpolées pour les dimensions Z et M lorsqu'elles sont présentes.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

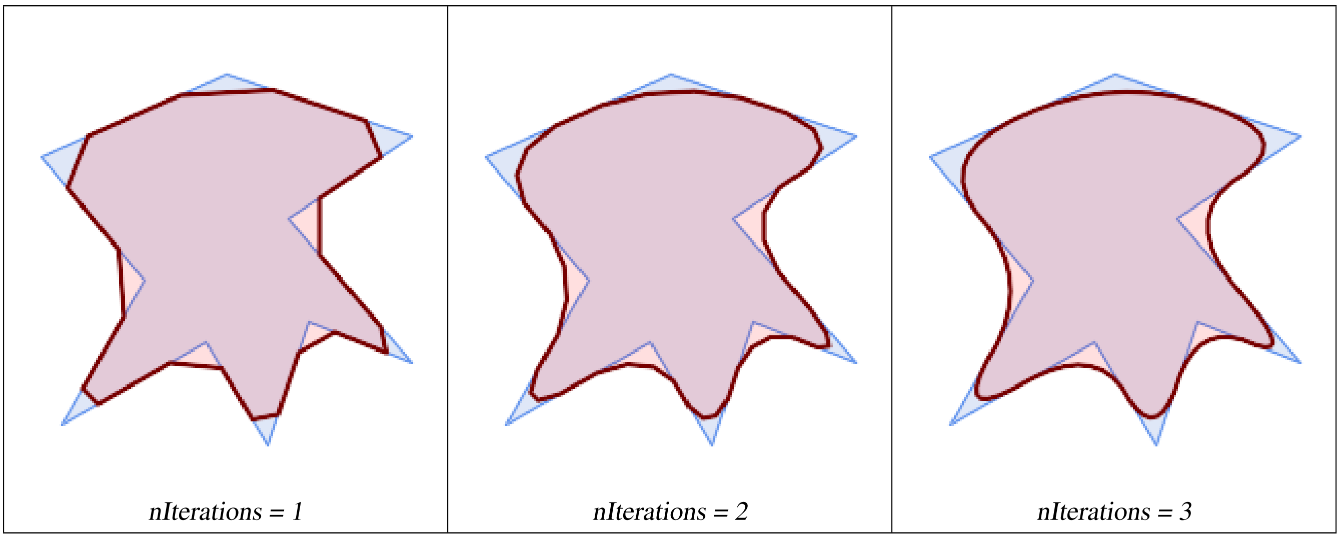
Disponibilité : 2.5.0

Exemples

Lissage d'un triangle :

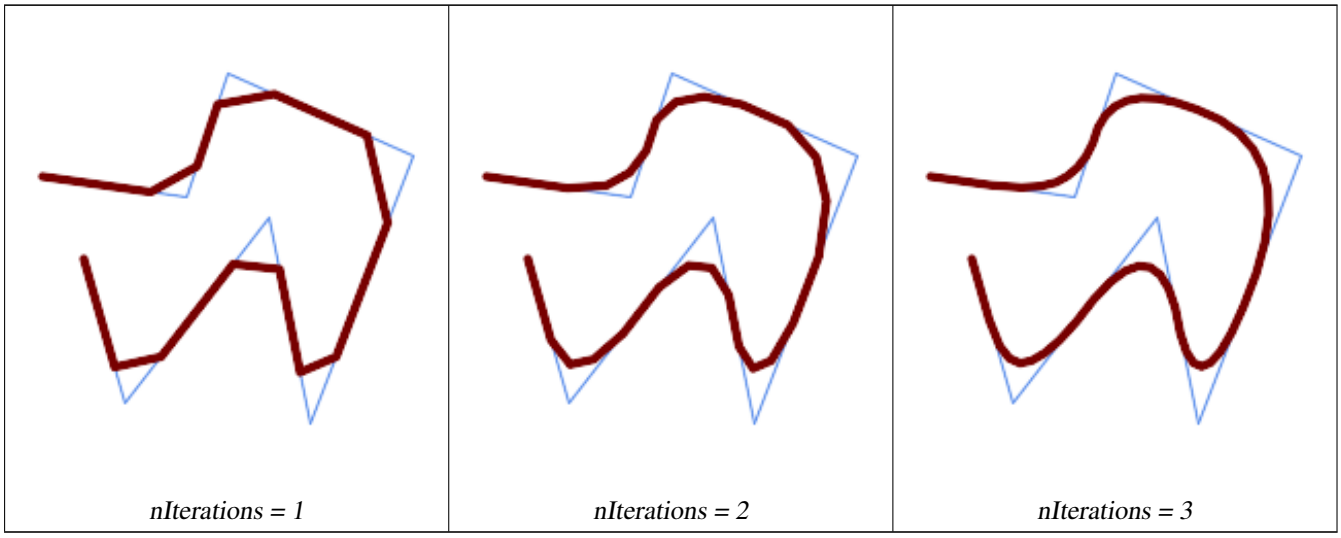
```
SELECT ST_AsText(ST_ChaikinSmoothing(geom)) smoothed
FROM (SELECT  'POLYGON((0 0, 8 8, 0 16, 0 0))'::geometry geom) AS foo;

                 smoothed
&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2500;&#x2
 POLYGON((2 2,6 6,6 10,2 14,0 12,0 4,2 2))
```

Lissage d'un polygone en utilisant 1, 2 et 3 itérations :




```
SELECT 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))',
            generate_series(1, 3) );
```

Lissage d'une LineString en utilisant 1, 2 et 3 itérations :



nIterations = 1 | *nIterations = 2* | *nIterations = 3*

```
SELECT ST_ChaikinSmoothing(
            'LINESTRING (10 140, 80 130, 100 190, 190 150, 140 20, 120 120, 50 30, 30 100) ↩
                ',
            generate_series(1, 3) );
```

Voir aussi

ST_Simplify, ST_SimplifyPreserveTopology, ST_SimplifyVW

7.14.5 ST_ConcaveHull

ST_ConcaveHull — Calcule une géométrie éventuellement concave qui contient tous les sommets de la géométrie d'entrée

Synopsis

geometry **ST_ConcaveHull**(geometry param_geom, float param_pctconvex, boolean param_allow_holes = false);

Description

Une enveloppe concave est une géométrie (généralement) concave qui contient l'entrée et dont les sommets sont un sous-ensemble des sommets de l'entrée. Dans le cas général, l'enveloppe concave est un polygone. L'enveloppe concave de deux ou plusieurs points colinéaires est une ligne à deux points. L'enveloppe concave d'un ou plusieurs points identiques est un Point. Le polygone ne contiendra pas de trous à moins que l'argument facultatif `param_allow_holes` ne soit spécifié comme vrai.

On peut considérer qu'une enveloppe concave "rétrécit" un ensemble de points. Cela diffère de la convex hull, qui revient à enrouler un élastique autour des points. Une enveloppe concave a généralement une surface plus petite et représente une limite plus naturelle pour les points d'entrée.

La `param_pctconvex` contrôle la concavité de l'enveloppe calculée. Une valeur de 1 produit une enveloppe convexe. Les valeurs comprises entre 1 et 0 produisent des enveloppes de concavité croissante. Une valeur de 0 produit une enveloppe avec une concavité maximale (mais toujours un polygone unique). Le choix d'une valeur appropriée dépend de la nature des données d'entrée, mais des valeurs comprises entre 0,3 et 0,1 produisent souvent des résultats raisonnables.

Note
Techniquement, la `param_pctconvex` détermine une longueur comme une fraction de la différence entre les arêtes les plus longues et les plus courtes dans la triangulation de Delaunay des points d'entrée. Les arêtes plus longues que cette longueur sont "érodées" de la triangulation. Les triangles restants forment l'enveloppe concave.

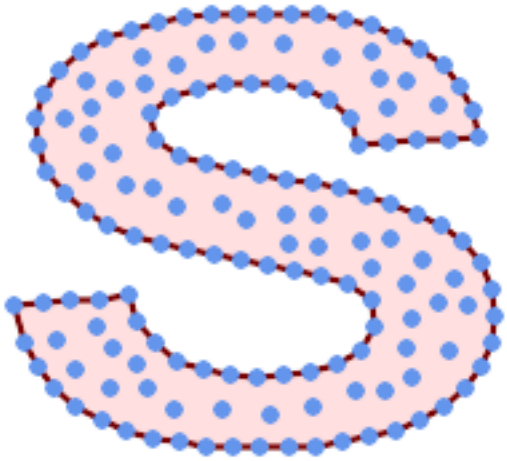
Pour les entrées ponctuelles et linéaires, l'enveloppe englobe tous les points de l'entrée. Pour les entrées polygonales, l'enveloppe englobera tous les points de l'entrée *et aussi* toutes les zones couvertes par l'entrée. Si vous souhaitez obtenir une enveloppe ponctuelle d'une entrée polygonale, convertissez-la d'abord en points à l'aide de ST_Points.

Il ne s'agit pas d'une fonction d'agrégation. Pour calculer l'enveloppe concave d'un ensemble de géométries, utilisez ST_Collect (par exemple `ST_ConcaveHull( ST_Collect( geom ), 0.80)`.

Disponibilité : 2.0.0

Amélioré : 3.3.0, implémentation native de GEOS activée pour GEOS 3.11+

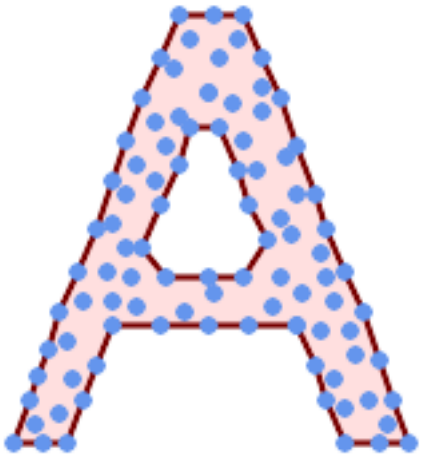
Exemples



Enveloppe concave d'un MultiPoint

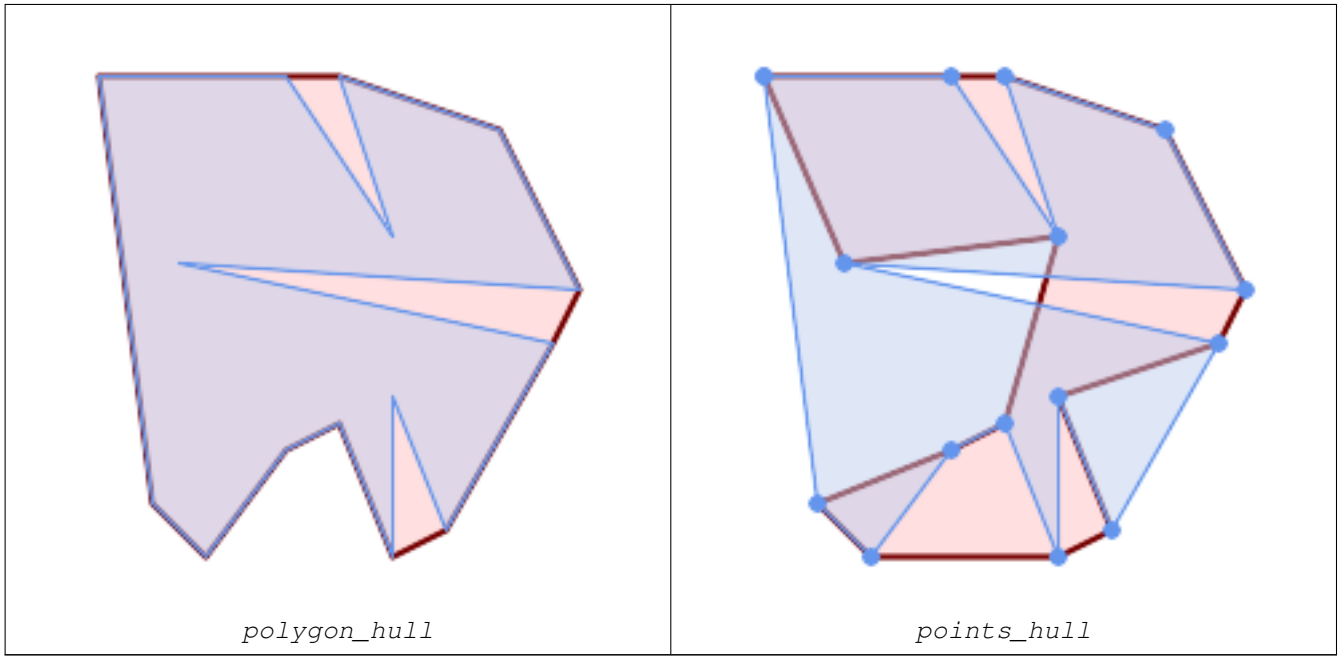
```
SELECT ST_AsText( ST_ConcaveHull(
        'MULTIPOINT ((10 72), (53 76), (56 66), (63 58), (71 51), (81 48), (91 46), (101  ↩
            45), (111 46), (121 47), (131 50), (140 55), (145 64), (144 74), (135 80), (125  ↩
            83), (115 85), (105 87), (95 89), (85 91), (75 93), (65 95), (55 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 ) );
---st_astext--
POLYGON ((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, 173 134, 161 134, 150 133, 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, 10 72, 21 73, 31  ↩
    74, 42 74, 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))
```



Enveloppe concave d'un MultiPoint, autorisant des trous

```
SELECT ST_AsText( 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 ) );
---st_astext--
POLYGON ((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, 114 64, 99 64, 81 64, 63 64, 57 49, 52 36, 46 20,  ↩
    37 20, 26 20, 32 36, 35 45, 39 55, 43 69), (88 124, 81 109, 74 93, 83 82, 99 82, 112 82, ↩
     121 96, 114 109, 110 122, 103 138, 92 138, 88 124))
```



Comparaison entre l'enveloppe concave d'un polygone et l'enveloppe concave des points qui le composent. L'enveloppe respecte les limites du polygone, ce qui n'est pas le cas de l'enveloppe basée sur les points.

```
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;
```

Utilisation avec ST_Collect pour calculer l'enveloppe concave d'un ensemble géométrique.

```
-- Compute estimate of infected area based on point observations
SELECT disease_type,
    ST_ConcaveHull( ST_Collect(obs_pnt), 0.3 ) AS geom
  FROM disease_obs
  GROUP BY disease_type;
```

Voir aussi

ST_ConvexHull, ST_Collect, ST_AlphaShape, ST_OptimalAlphaShape

7.14.6 ST_ConvexHull

ST_ConvexHull — Calcule l'enveloppe convexe d'une géométrie.

Synopsis

geometry **ST_ConvexHull**(geometry geomA);

Description

Calcule l'enveloppe convexe d'une géométrie. L'enveloppe convexe est la plus petite géométrie convexe qui englobe toutes les géométries de l'entrée.

On peut considérer l'enveloppe convexe comme la géométrie obtenue en enroulant un élastique autour d'un ensemble de géométries. Ceci est différent d'une concave hull qui est analogue à un "rétrécissement" des géométries. Une enveloppe convexe est souvent utilisée pour déterminer une zone affectée sur la base d'un ensemble d'observations ponctuelles.

Dans le cas général, l'enveloppe convexe est un polygone. L'enveloppe convexe de deux ou plusieurs points colinéaires est une LineString à deux points. L'enveloppe convexe d'un ou plusieurs points identiques est un Point.

Il ne s'agit pas d'une fonction d'agrégation. Pour calculer l'enveloppe convexe d'un ensemble de géométries, utilisez ST_Collect pour les agréger dans une collection de géométries (par exemple `ST_ConvexHull(ST_Collect(geom))`.

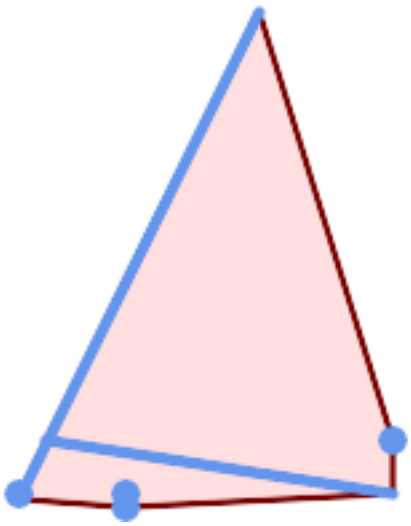
Effectué par le module GEOS

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s2.1.1.3

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1.16

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples



Enveloppe convexe d'une MultiLinestring et d'un MultiPoint

```
SELECT ST_AsText(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)')
            )) );
---st_astext--
POLYGON((50 5,10 8,10 10,100 190,150 30,150 10,50 5))
```

Utilisation avec ST_Collect pour calculer les enveloppes convexes des ensembles géométriques.

```
--Get estimate of infected area based on point observations
SELECT d.disease_type,
    ST_ConvexHull(ST_Collect(d.geom)) As geom
    FROM disease_obs As d
    GROUP BY d.disease_type;
```

Voir aussi

ST_Collect, ST_ConcaveHull, ST_MinimumBoundingCircle

7.14.7 ST_DelaunayTriangles

ST_DelaunayTriangles — Renvoie la triangulation de Delaunay des sommets d'une géométrie.

Synopsis

geometry **ST_DelaunayTriangles**(geometry g1, float tolerance = 0.0, int4 flags = 0);

Description

Calcule la Delaunay triangulation des sommets de la géométrie d'entrée. La `tolerance` optionnelle peut être utilisée pour regrouper les sommets proches, ce qui améliore la robustesse dans certaines situations. La géométrie résultante est délimitée par l'enveloppe convexe des sommets de la géométrie d'entrée. La représentation de la géométrie résultante est déterminée par le code `flags` :

- `0` - une collection de géométries de POLYGONES triangulaires (par défaut)
- `1` - une MULTILINESTRING des arêtes de la triangulation
- `2` - Un TIN de la triangulation

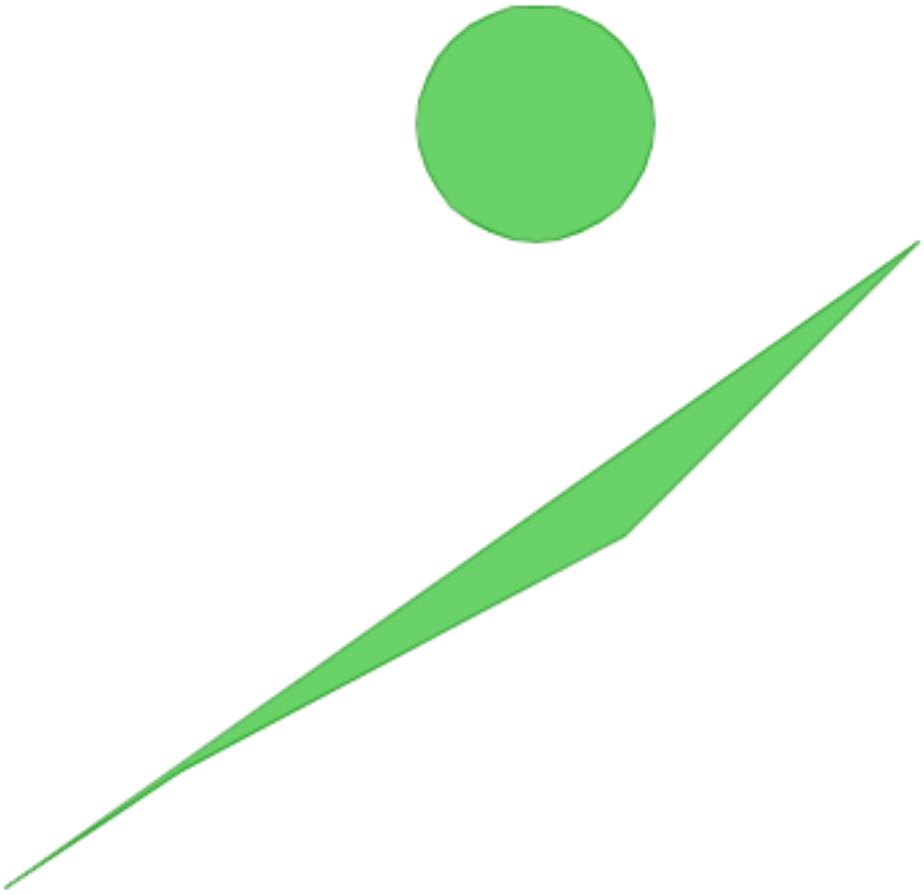
Effectué par le module GEOS.

Disponibilité : 2.1.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

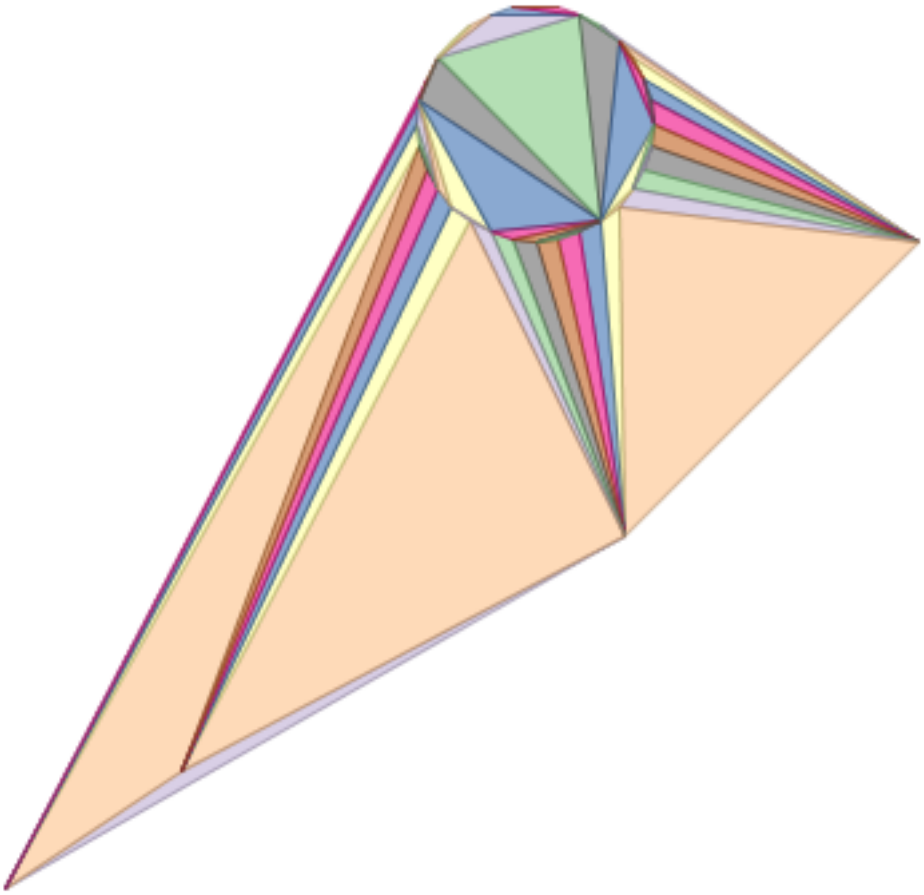
Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples



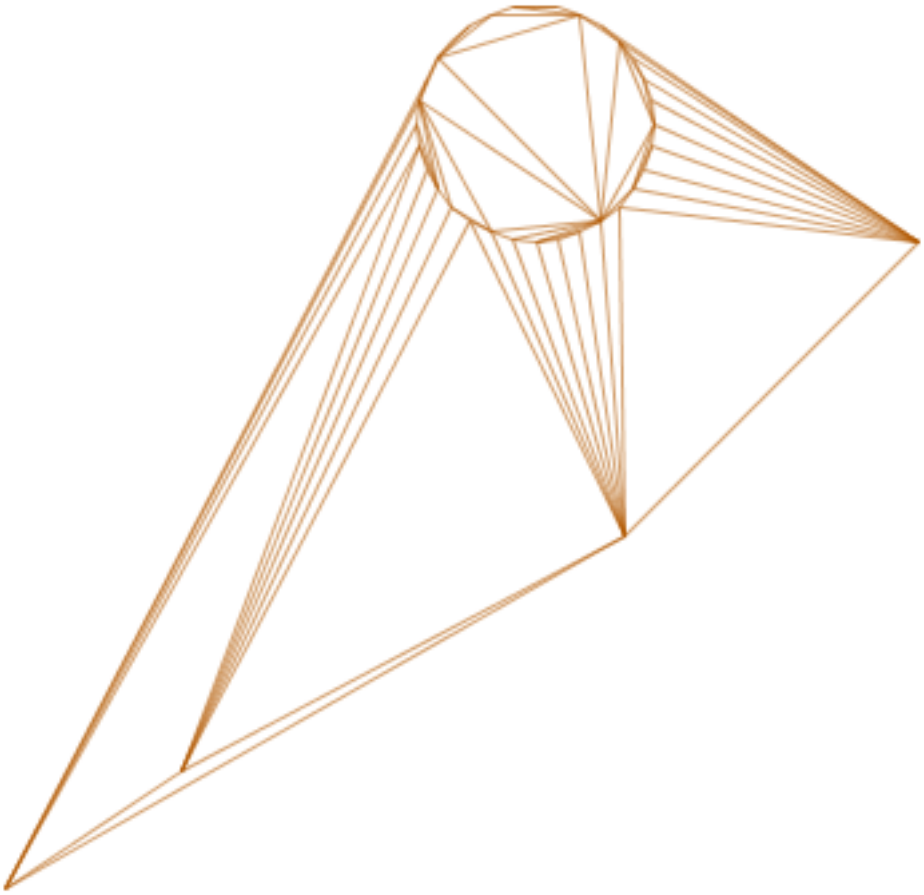
Polygones d'origine

```
our original geometry
    ST_Union(ST_GeomFromText('POLYGON((175 150, 20 40,
            50 60, 125 100, 175 150))'),
        ST_Buffer(ST_GeomFromText('POINT(110 170)'), 20)
        )
```



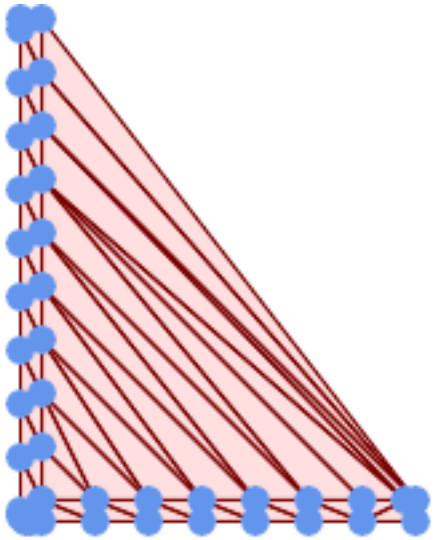
ST_DelaunayTriangles de 2 polygones : polygones de triangle delaunay, chaque triangle est thématisé dans une couleur différente

```
geometries overlaid multilinestring triangles

SELECT
    ST_DelaunayTriangles(
        ST_Union(ST_GeomFromText('POLYGON((175 150, 20 40,
            50 60, 125 100, 175 150))'),
        ST_Buffer(ST_GeomFromText('POINT(110 170)'), 20)
        ))
     As  dtriag;
```



-- triangles de delaunay en tant que multilignes

```
SELECT
    ST_DelaunayTriangles(
        ST_Union(ST_GeomFromText('POLYGON((175 150, 20 40,
            50 60, 125 100, 175 150))'),
        ST_Buffer(ST_GeomFromText('POINT(110 170)'), 20)
        ),0.001,1)
     As  dtriag;
```



-- triangles de delaunay de 45 points sous forme de 55 polygones triangulaires

```
this produces a table of 42 points that form an L shape

SELECT (ST_DumpPoints(ST_GeomFromText(
'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)'))).geom
    INTO TABLE l_shape;

output as individual polygon triangles

SELECT ST_AsText((ST_Dump(geom)).geom) As wkt
FROM ( SELECT ST_DelaunayTriangles(ST_Collect(geom)) As geom
FROM l_shape) As foo;


wkt

POLYGON((6 194,6 190,14 194,6 194))
POLYGON((14 194,6 190,14 174,14 194))
POLYGON((14 194,14 174,154 14,14 194))
POLYGON((154 14,14 174,14 154,154 14))
POLYGON((154 14,14 154,150 14,154 14))
POLYGON((154 14,150 14,154 6,154 14))
```

Exemple utilisant des sommets avec des valeurs Z.

```
3D multipoint

SELECT ST_AsText(ST_DelaunayTriangles(ST_GeomFromText(
         'MULTIPOINT Z(14 14 10, 150 14 100,34 6 25, 20 10 150)'))) As wkt;


wkt

GEOMETRYCOLLECTION Z (POLYGON Z ((14 14 10,20 10 150,34 6 25,14 14 10))
 ,POLYGON Z ((14 14 10,34 6 25,150 14 100,14 14 10)))
```

Voir aussi

ST_VoronoiPolygons, ST_TriangulatePolygon, ST_ConstrainedDelaunayTriangles, ST_VoronoiLines, ST_ConvexHull

7.14.8 ST_FilterByM

ST_FilterByM — Supprime les vertices en fonction de leur valeur M

Synopsis

geometry **ST_FilterByM**(geometry geom, double precision min, double precision max = null, boolean returnM = false);

Description

Filtre les points de sommet en fonction de leur valeur M. Renvoie une géométrie contenant uniquement les points de sommet dont la valeur M est supérieure ou égale à la valeur min et inférieure ou égale à la valeur max. Si l'argument max-value est omis, seule la valeur min est prise en compte. Si le quatrième argument est omis, la valeur m ne figurera pas dans la géométrie résultante. Si la géométrie résultante a trop peu de points de vertex pour son type de géométrie, une géométrie vide sera renvoyée. Dans une collection de géométries, les géométries qui n'ont pas assez de points seront simplement éliminées silencieusement.

Cette fonction est principalement destinée à être utilisée en conjonction avec ST_SetEffectiveArea. ST_EffectiveArea définit la surface effective d'un sommet dans sa valeur m. Avec ST_FilterByM, il est possible d'obtenir une version simplifiée de la géométrie sans aucun calcul, simplement en filtrant

Note
Il existe une différence entre ST_SimplifyVW et ST_FilterByM en ce qui concerne la géométrie renvoyée lorsque le nombre de points répondant aux critères n'est pas suffisant. ST_SimplifyVW renvoie la géométrie avec suffisamment de points alors que ST_FilterByM renvoie une géométrie vide

Note
Notez que la géométrie renvoyée peut être invalide

Note
Cette fonction renvoie toutes les dimensions, y compris les valeurs Z et M

Disponibilité : 2.5.0

Exemples

Une ligne est filtrée

```
SELECT ST_AsText(ST_FilterByM(geom,30)) simplified
FROM (SELECT  ST_SetEffectiveArea('LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)'::geometry) geom ↩
    ) As foo;

result

         simplified
----------------------------
 LINESTRING(5 2,7 25,10 10)
```

Voir aussi

ST_SetEffectiveArea, ST_SimplifyVW

7.14.9 ST_GeneratePoints

ST_GeneratePoints — Génère un multipoint de points aléatoires contenus dans un polygone ou un multipolygone.

Synopsis

geometry **ST_GeneratePoints**(geometry g, integer npoints, integer seed = 0);

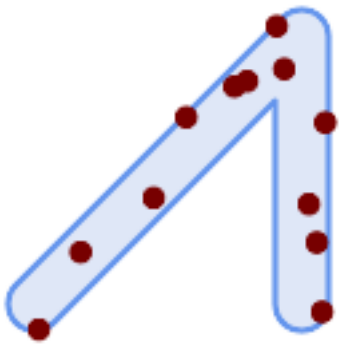
Description

ST_GeneratePoints génère un multipoint composé d'un nombre donné de points pseudo-aléatoires situés dans la zone d'entrée. Le paramètre facultatif `seed` est utilisé pour régénérer une séquence déterministe de points et doit être supérieur à zéro.

Disponibilité : 2.3.0

Amélioration : 3.0.0, ajout du paramètre seed

Exemples



Génération d'un multipoint composé de 12 points superposés au polygone d'origine en utilisant une valeur de départ aléatoire 1996

```
SELECT ST_GeneratePoints(geom, 12, 1996)
FROM (
    SELECT ST_Buffer(
        ST_GeomFromText(
        'LINESTRING(50 50,150 150,150 50)'),
        10, 'endcap=round join=round') AS geom
) AS s;
```

Etant donné un tableau de polygones s, retourner 12 points individuels par polygone. Les résultats seront différents à chaque exécution.

```
SELECT s.id, dp.path[1] AS pt_id, dp.geom
FROM s, ST_DumpPoints(ST_GeneratePoints(s.geom,12)) AS dp;
```

Voir aussi

ST_DumpPoints

7.14.10 ST_GeometricMedian

ST_GeometricMedian — Renvoie la médiane géométrique d'un MultiPoint.

Synopsis

geometry **ST_GeometricMedian** (geometry geom, float8 tolerance = NULL, int max_iter = 10000, boolean fail_if_not_converged = false);

Description

Calcule la médiane géométrique approximative d'une géométrie MultiPoint à l'aide de l'algorithme de Weiszfeld. La médiane géométrique est le point qui minimise la somme des distances aux points d'entrée. Elle fournit une mesure de centralité moins sensible aux points aberrants que le centroïde (centre de masse).

L'algorithme itère jusqu'à ce que la variation de la distance entre les itérations successives soit inférieure au paramètre `tolerance` fourni. Si cette condition n'est pas remplie après `max_iterations` itérations, la fonction produit une erreur et se termine, à moins que `fail_fif_not_converged` ne soit fixé à `false` (par défaut).

Si l'argument `tolerance` n'est pas fourni, la valeur de la tolérance est calculée sur la base de l'étendue de la géométrie d'entrée.

Si elle est présente, les valeurs M des points d'entrée sont interprétées comme leurs poids relatifs.

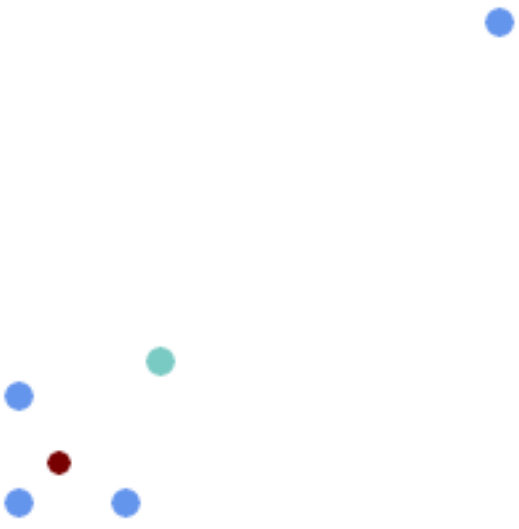
Disponibilité : 2.3.0

Amélioré : 2.5.0 Ajout de la prise en charge de M comme poids des points.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Exemples



Comparaison de la médiane géométrique (rouge) et du centroïde (turquoise) d'un MultiPoint.

```
WITH test AS (
SELECT 'MULTIPOINT((10 10), (10 40), (40 10), (190 190))'::geometry geom)
SELECT
  ST_AsText(ST_Centroid(geom)) centroid,
  ST_AsText(ST_GeometricMedian(geom)) median
FROM test;

      centroid      |                median
--------------------+---------------------------------------
   POINT(62.5 62.5) | POINT(25.01778421249728 25.01778421249728)
(1 row)
```

Voir aussi

ST_Centroid

7.14.11 ST_LineMerge

ST_LineMerge — Renvoie les lignes formées par la couture d'une MultiLineString.

Synopsis

geometry **ST_LineMerge**(geometry amultilinestring);
geometry **ST_LineMerge**(geometry amultilinestring, boolean directed);

Description

Renvoie une LineString ou une MultiLineString formée par l'assemblage des éléments de ligne d'une MultiLineString. Les lignes sont jointes à leurs extrémités aux intersections à 2 voies. Les lignes ne sont pas jointes aux intersections à 3 voies ou plus.

Si **directed** est TRUE, ST_LineMerge ne modifiera pas l'ordre des points dans les LineStrings, de sorte que les lignes ayant des directions opposées ne seront pas fusionnées

Note
A n'utiliser qu'avec les MultiLineString/LineStrings. Les autres types de géométrie renvoient une GeometryCollection vide

Effectué par le module GEOS.

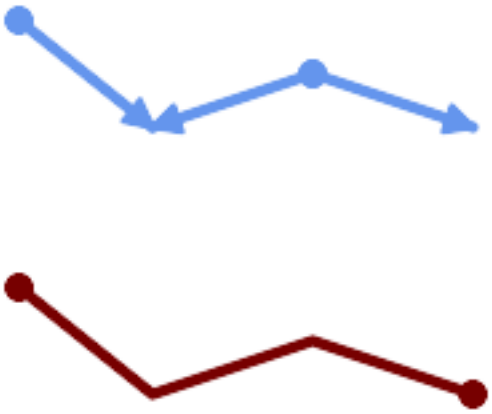
Amélioration : 3.3.0 accepte un paramètre direct.

GEOS >= 3.11.0 est nécessaire pour utiliser le paramètre direct.

Disponibilité : 1.1.0

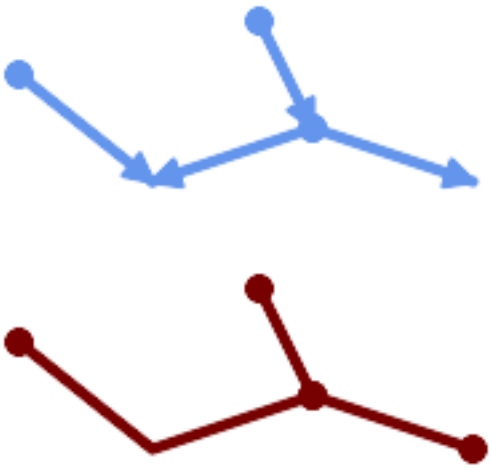
Warning
Cette fonction supprime la dimension M.

Exemples



Fusionner des lignes d'orientation différente.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((10 160, 60 120), (120 140, 60 120), (120 140, 180 120))'
                ));
------------------------------------------
 LINESTRING(10 160,60 120,120 140,180 120)
```

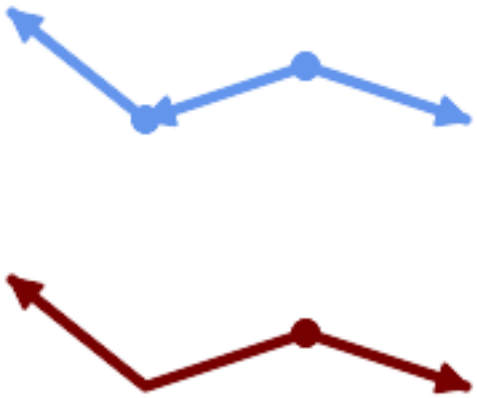


Les lignes ne sont pas fusionnées aux intersections de degré > 2.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((10 160, 60 120), (120 140, 60 120), (120 140, 180 120), (100 180, 120  ↩
    140))'
                ));
------------------------------------------
 MULTILINESTRING((10 160,60 120,120 140),(100 180,120 140),(120 140,180 120))
```

Si la fusion n'est pas possible en raison de lignes qui ne se touchent pas, la MultiLineString d'origine est renvoyée.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33),(-45.2 -33.2,-46 -32))'
));
----------------
MULTILINESTRING((-45.2 -33.2,-46 -32),(-29 -27,-30 -29.7,-36 -31,-45 -33))
```



Les lignes de directions opposées ne sont pas fusionnées si directed = TRUE.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((60 30, 10 70), (120 50, 60 30), (120 50, 180 30))',
TRUE));
----------------------------------------------------
 MULTILINESTRING((120 50,60 30,10 70),(120 50,180 30))
```

Exemple de traitement de la dimension Z.

```
SELECT ST_AsText(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))'
        ));
------------------------------------------------------------------------------------------------ ←

LINESTRING Z (-30 -29.7 5,-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 1,-46 -32 11)
```

Voir aussi

ST_Segmentize, ST_LineSubstring

7.14.12 ST_MaximumInscribedCircle

ST_MaximumInscribedCircle — Calcule le plus grand cercle contenu dans une géométrie.

Synopsis

(geometry, geometry, double precision) **ST_MaximumInscribedCircle**(geometry geom);

Description

Trouve le plus grand cercle contenu dans un (multi)polygone, ou qui ne chevauche aucune ligne ni aucun point. Renvoie un enregistrement avec des champs :

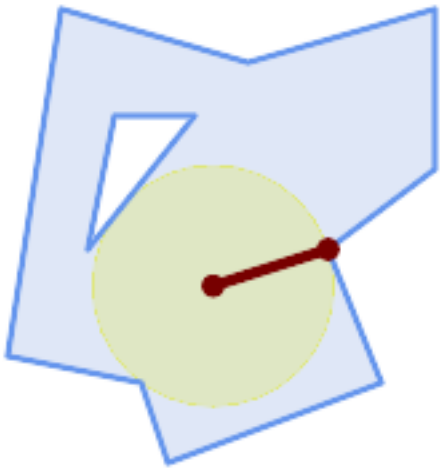
- `center` - point central du cercle
- `nearest` - un point de la géométrie le plus proche du centre
- `radius` - rayon du cercle

Pour les entrées polygonales, le cercle est inscrit dans les anneaux de délimitation, en utilisant les anneaux internes comme limites. Pour les entrées linéaires et ponctuelles, le cercle est inscrit dans l'enveloppe convexe de l'entrée, en utilisant les lignes et les points de l'entrée comme autres limites.

Disponibilité : 3.1.0.

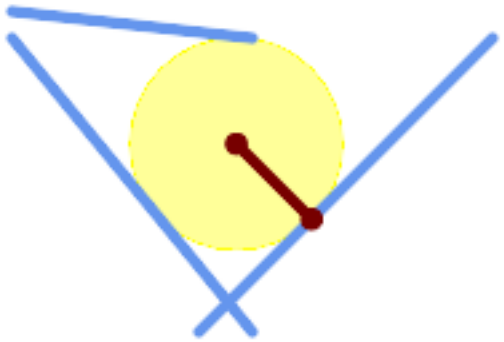
Nécessite GEOS >= 3.9.0.

Exemples



Cercle inscrit maximal d'un polygone. Le centre, le point le plus proche et le rayon sont renvoyés.

```
SELECT radius, ST_AsText(center) AS center, ST_AsText(nearest) AS nearest
    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))');

     radius      |           center           |    nearest
-----------------+----------------------------+---------------
 45.165845650018 | POINT(96.953125 76.328125) | POINT(140 90)
```



Cercle inscrit maximal d'une multiligne. Le centre, le point le plus proche et le rayon sont renvoyés.

Voir aussi

ST_MinimumBoundingRadius, ST_LargestEmptyCircle

7.14.13 ST_LargestEmptyCircle

ST_LargestEmptyCircle — Calcule le plus grand cercle ne recouvrant pas une géométrie.

Synopsis

(geometry, geometry, double precision) **ST_LargestEmptyCircle**(geometry geom, double precision tolerance=0.0, geometry boundary=POINT EMPTY);

Description

Trouve le plus grand cercle qui ne chevauche pas un ensemble d'obstacles de points et de lignes. (Les géométries polygonales peuvent être incluses en tant qu'obstacles, mais seules leurs lignes de démarcation sont utilisées). Le centre du cercle est contraint de se situer à l'intérieur d'une limite polygonale, qui par défaut est l'enveloppe convexe de la géométrie d'entrée. Le centre du cercle est le point à l'intérieur de la frontière qui est le plus éloigné des obstacles. Le cercle lui-même est fourni par le point central et un point le plus proche situé sur un obstacle déterminant le rayon du cercle.

Le centre du cercle est déterminé avec une précision donnée, spécifiée par une tolérance de distance, à l'aide d'un algorithme itératif. Si la distance de précision n'est pas spécifiée, une valeur par défaut raisonnable est utilisée.

Renvoie un enregistrement avec des champs :

- `center` - point central du cercle
- `nearest` - un point de la géométrie le plus proche du centre
- `radius` - rayon du cercle

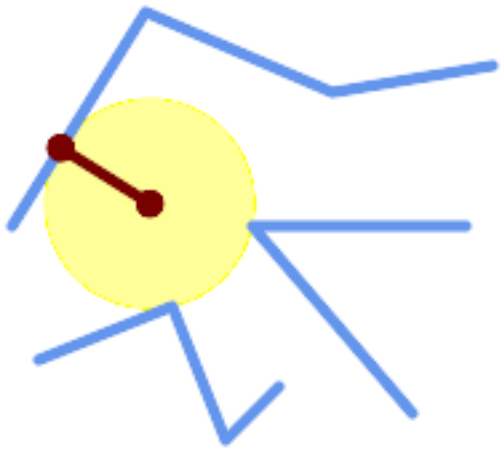
Pour trouver le plus grand cercle vide à l'intérieur d'un polygone, voir ST_MaximumInscribedCircle.

Disponibilité : 3.4.0.

Nécessite GEOS >= 3.9.0.

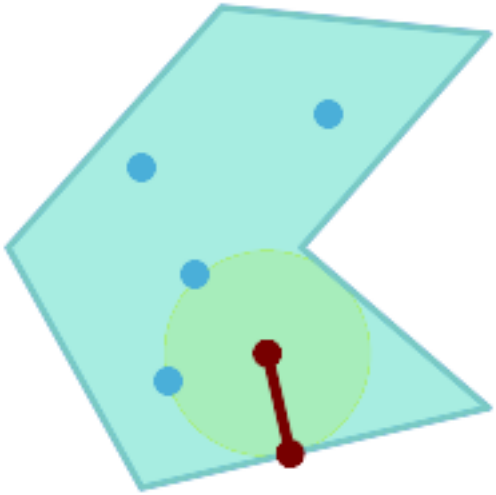
Exemples

```
SELECT radius,
      center,
      nearest
  FROM ST_LargestEmptyCircle(
       'MULTILINESTRING (
         (10 100, 60 180, 130 150, 190 160),
         (20 50, 70 70, 90 20, 110 40),
         (160 30, 100 100, 180 100))');
```



Le plus grand cercle vide à l'intérieur d'un ensemble de lignes.

```
SELECT radius,
       center,
       nearest
  FROM ST_LargestEmptyCircle(
         ST_Collect(
           '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) ↩
              ,
           0,
         'POLYGON ((90 190, 10 100, 60 10, 190 40, 120 100, 190 180, 90 190))'::geometry
       );
```



Le plus grand cercle vide à l'intérieur d'un ensemble de points, contraint à se situer dans un polygone. La limite du polygone de contrainte doit être incluse en tant qu'obstacle et spécifiée en tant que contrainte pour le centre du cercle.

Voir aussi

ST_MinimumBoundingRadius

7.14.14 ST_MinimumBoundingCircle

ST_MinimumBoundingCircle — Retourne le plus petit cercle polygonal qui contient une géométrie.

Synopsis

geometry **ST_MinimumBoundingCircle**(geometry geomA, integer num_segs_per_qt_circ=48);

Description

Retourne le plus petit cercle polygonal qui contient une géométrie.

Note
Le cercle de délimitation est approximé par un polygone avec une valeur par défaut de 48 segments par quart de cercle. Le polygone étant une approximation du cercle de délimitation minimal, certains points de la géométrie d'entrée peuvent ne pas être contenus dans le polygone. L'approximation peut être améliorée en augmentant le nombre de segments. Pour les applications où une approximation ne convient pas, on peut utiliser ST_MinimumBoundingRadius.

A utiliser avec ST_Collect pour obtenir le cercle minimal de délimitation d'un ensemble de géométries.

Pour calculer deux points situés sur le cercle minimum (le "diamètre maximum"), utilisez ST_LongestLine.

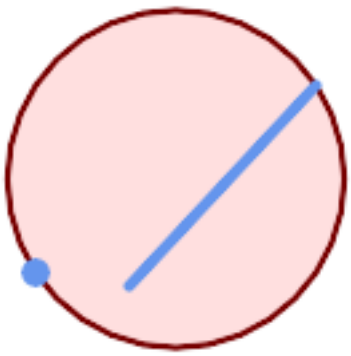
Le rapport de la surface d'un polygone divisée par la surface de son cercle circonscrit minimal est appelé *Score de compacité de Reock*.

Effectué par le module GEOS.

Disponibilité : 1.4.0

Exemples

```
SELECT d.disease_type,
    ST_MinimumBoundingCircle(ST_Collect(d.geom)) As geom
    FROM disease_obs As d
    GROUP BY d.disease_type;
```



Cercle minimal de délimitation d'un point et d'une ligne de démarcation. Utilisation de 8 segments pour approximer un quart de cercle

```
SELECT ST_AsText(ST_MinimumBoundingCircle(
        ST_Collect(
            ST_GeomFromText('LINESTRING(55 75,125 150)'),
                ST_Point(20, 80)), 8
                )) As wktmbc;
wktmbc
-----------
POLYGON((135.59714732062 115,134.384753327498 102.690357210921,130.79416296937  ↩
    90.8537670908995,124.963360620072 79.9451031602111,117.116420743937  ↩
    70.3835792560632,107.554896839789 62.5366393799277,96.6462329091006  ↩
    56.70583703063,84.8096427890789 53.115246672502,72.5000000000001  ↩
    51.9028526793802,60.1903572109213 53.1152466725019,48.3537670908996  ↩
    56.7058370306299,37.4451031602112 62.5366393799276,27.8835792560632  ↩
    70.383579256063,20.0366393799278 79.9451031602109,14.20583703063  ↩
    90.8537670908993,10.615246672502 102.690357210921,9.40285267938019 115,10.6152466725019  ↩
    127.309642789079,14.2058370306299 139.1462329091,20.0366393799275  ↩
    150.054896839789,27.883579256063 159.616420743937,
37.4451031602108 167.463360620072,48.3537670908992 173.29416296937,60.190357210921  ↩
    176.884753327498,
72.4999999999998 178.09714732062,84.8096427890786 176.884753327498,96.6462329091003  ↩
    173.29416296937,107.554896839789 167.463360620072,
117.116420743937 159.616420743937,124.963360620072 150.054896839789,130.79416296937  ↩
    139.146232909101,134.384753327498 127.309642789079,135.59714732062 115))
```

Voir aussi

ST_Collect, ST_MinimumBoundingRadius, ST_LargestEmptyCircle, ST_LongestLine

7.14.15 ST_MinimumBoundingRadius

ST_MinimumBoundingRadius — Renvoie le point central et le rayon du plus petit cercle contenant une géométrie.

Synopsis

(geometry, double precision) **ST_MinimumBoundingRadius**(geometry geom);

Description

Calcule le point central et le rayon du plus petit cercle contenant une géométrie. Retourne un enregistrement avec des champs :

- `center` - point central du cercle
- `radius` - rayon du cercle

A utiliser avec ST_Collect pour obtenir le cercle minimal de délimitation d'un ensemble de géométries.

Pour calculer deux points situés sur le cercle minimum (le "diamètre maximum"), utilisez ST_LongestLine.

Disponibilité - 2.3.0

Exemples

```
SELECT ST_AsText(center), radius FROM ST_MinimumBoundingRadius('POLYGON((26426 65078,26531 ↩
    65242,26075 65136,26096 65427,26426 65078))');

                st_astext                 |      radius
------------------------------------------+------------------
 POINT(26284.8418027133 65267.1145090825) | 247.436045591407
```

Voir aussi

ST_Collect, ST_MinimumBoundingCircle, ST_LongestLine

7.14.16 ST_OrientedEnvelope

ST_OrientedEnvelope — Renvoie un rectangle de surface minimale contenant une géométrie.

Synopsis

geometry **ST_OrientedEnvelope**(geometry geom);

Description

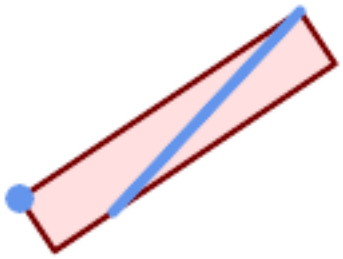
Renvoie le rectangle rotatif de surface minimale entourant une géométrie. Notez qu'il peut exister plus d'un rectangle de ce type. Peut renvoyer un point ou une ligne dans le cas d'entrées dégénérées.

Disponibilité : 2.5.0.

Nécessite GEOS >= 3.6.0.

Exemples

```
SELECT ST_AsText(ST_OrientedEnvelope('MULTIPOINT ((0 0), (-1 -1), (3 2))'));

                st_astext
                ----------------------------------------------
                POLYGON((3 2,2.88 2.16,-1.12 -0.84,-1 -1,3 2))
```



Enveloppe orientée d'un point et d'une ligne.

```
SELECT ST_AsText(ST_OrientedEnvelope(
        ST_Collect(
            ST_GeomFromText('LINESTRING(55 75,125 150)'),
                ST_Point(20, 80))
                )) As wktenv;
wktenv
-----------
POLYGON((19.9999999999997 79.9999999999999,33.0769230769229  ↩
    60.3846153846152,138.076923076924 130.384615384616,125.000000000001  ↩
    150.000000000001,19.9999999999997 79.9999999999999))
```

Voir aussi

ST_Envelope ST_MinimumBoundingCircle

7.14.17 ST_OffsetCurve

ST_OffsetCurve — Renvoie une ligne décalée par rapport à une distance et un côté donnés à partir d'une ligne en entrée.

Synopsis

geometry **ST_OffsetCurve**(geometry line, float signed_distance, text style_parameters='');

Description

Renvoie une ligne décalée par rapport à une distance et un côté donnés à partir d'une ligne en entrée. Tous les points des géométries renvoyées ne sont pas plus éloignés que la distance donnée de la géométrie d'entrée. Utile pour calculer des lignes parallèles autour d'une ligne centrale.

Pour une distance positive, le décalage se situe à gauche de la ligne d'entrée et conserve la même direction. Pour une distance négative, il se trouve sur le côté droit et dans la direction opposée.

Les unités de distance sont mesurées en unités du système de référence spatiale.

Notez que la sortie peut être un MULTILINESTRING ou un EMPTY pour certaines géométries d'entrée en forme de puzzle.

Le troisième paramètre facultatif permet de spécifier une liste de paires clé/valeur séparées par des blancs afin de modifier les opérations comme suit :

- 'quad_segs=#' : nombre de segments utilisés pour approximer un quart de cercle (8 par défaut).
- 'join=round|mitre|bevel' : style de jointure (la valeur par défaut est "round"). 'miter' est également accepté comme synonyme de 'mitre'.
- 'mitre_limit=#.#' : limite du rapport de "mitre" (n'affecte que le style d'assemblage "mitred"). 'miter_limit' est également accepté comme synonyme de 'mitre_limit'.

Effectué par le module GEOS.

Behavior changed in GEOS 3.11 so offset curves now have the same direction as the input line, for both positive and negative offsets.

Disponibilité : 2.0

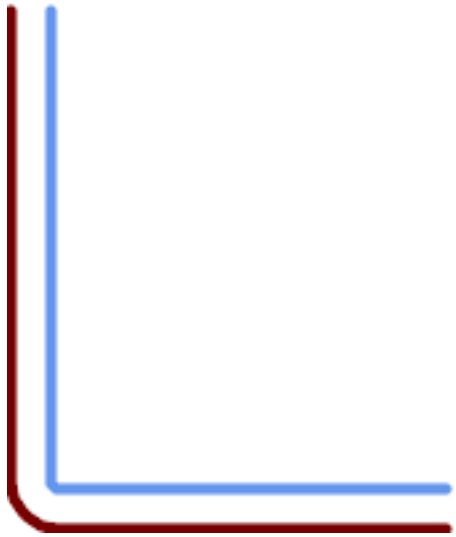
Amélioration : 2.5 - ajout de la prise en charge de GEOMETRYCOLLECTION et MULTILINESTRING

Note
Cette fonction ignore la dimension Z. Elle donne toujours un résultat en 2D, même lorsqu'elle est utilisée sur une géométrie en 3D.

Exemples

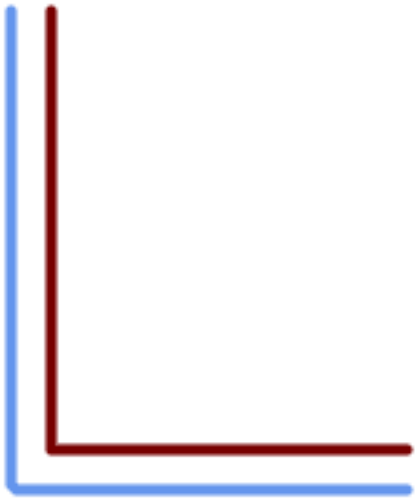
Calculer une zone tampon ouverte autour des routes

```
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;
```



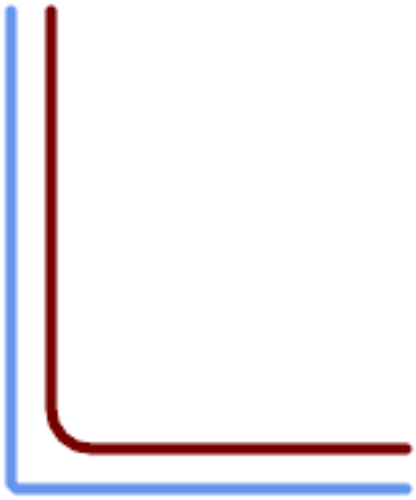
15, 'quad_segs=4 join=round' ligne originale et son décalage de 15 unités.

```
SELECT ST_AsText(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 1,12.2597485145237  ↩
    2.1418070123307,
    7.39339828220179 5.39339828220179,
    5.39339828220179 7.39339828220179,
    2.14180701233067 12.2597485145237,1  ↩
    18,1 195)
```



-15, 'quad_segs=4 join=round' ligne originale et son décalage de -15 unités

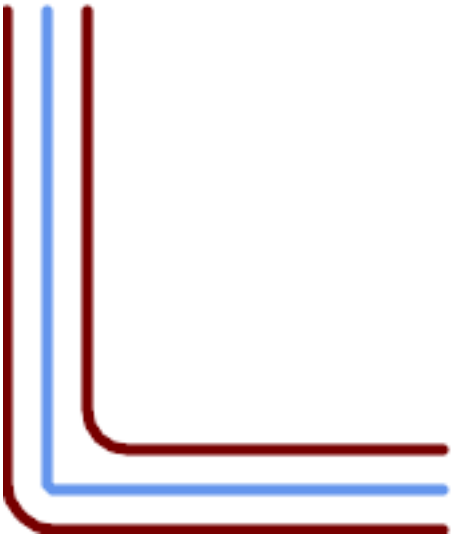
```
SELECT ST_AsText(ST_OffsetCurve(geom,
    -15, 'quad_segs=4 join=round')) As  ↩
    notsocurvy
    FROM ST_GeomFromText(
'LINESTRING(164 16,144 16,124 16,104  ↩
    16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') ↩
     As geom;

notsocurvy

LINESTRING(31 195,31 31,164 31)
```



double décalage pour obtenir plus de courbes, notez que le premier inverse la direction, donc -30 + 15 = -15

```
SELECT ST_AsText(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;
```

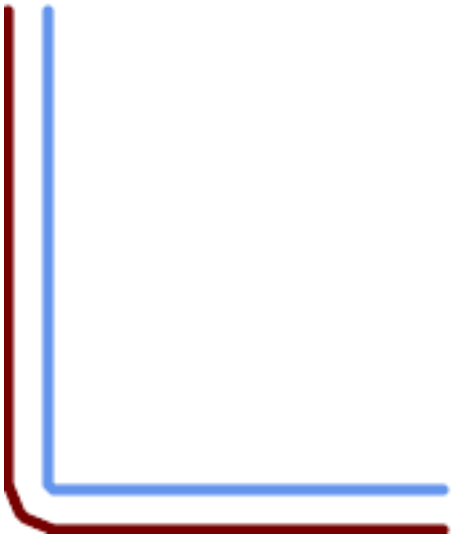
```
morecurvy

LINESTRING(164 31,46 31,40.2597485145236  ↩
    32.1418070123307,
35.3933982822018 35.3933982822018,
32.1418070123307 40.2597485145237,31  ↩
    46,31 195)
```



Double décalage pour obtenir des courbes plus prononcées, combiné avec le décalage régulier 15 pour obtenir des lignes parallèles. Superposition avec l'original.

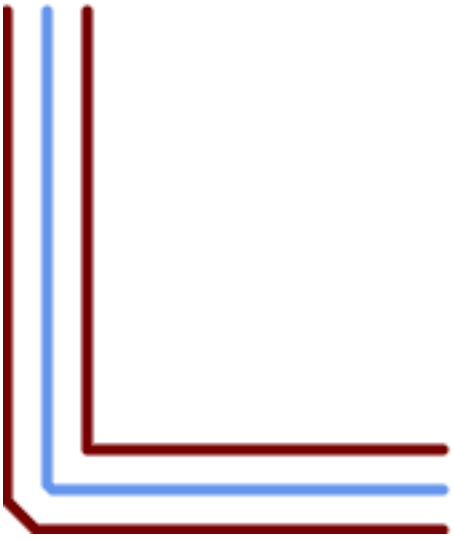
```
SELECT ST_AsText(ST_Collect(
    ST_OffsetCurve(geom, 15, 'quad_segs=4 ↩
     join=round'),
    ST_OffsetCurve(ST_OffsetCurve(geom,
    -30, 'quad_segs=4 join=round'), -15,  ↩
    'quad_segs=4 join=round')
    )
) As parallel_curves
    FROM ST_GeomFromText(
'LINESTRING(164 16,144 16,124 16,104  ↩
    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;
```

```
parallel curves

MULTILINESTRING((164 1,18  ↩
    1,12.2597485145237 2.1418070123307,
7.39339828220179  ↩
    5.39339828220179,5.39339828220179 7.393398282201
2.14180701233067 12.2597485145237,1 18,1  ↩
    195),
(164 31,46 31,40.2597485145236  ↩
    32.1418070123307,35.3933982822018 35.39339828220
32.1418070123307 40.2597485145237,31  ↩
    46,31 195))
```



15, "quad_segs=4 join=bevel" montré avec la ligne originale

```
SELECT ST_AsText(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 1,7.39339828220179  ↩
    5.39339828220179,
    5.39339828220179 7.39339828220179,1  ↩
    18,1 195)
```



15,-15 collecté, join=mitre mitre_limit=2.1

```
SELECT ST_AsText(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 1,11.7867965644036  ↩
    1,1 11.7867965644036,1 195),
    (31 195,31 31,164 31))
```

Voir aussi

ST_Buffer

7.14.18 ST_PointOnSurface

ST_PointOnSurface — Calcule un point dont on garantit qu'il se trouve dans un polygone ou sur une géométrie.

Synopsis

geometry **ST_PointOnSurface**(geometry g1);

Description

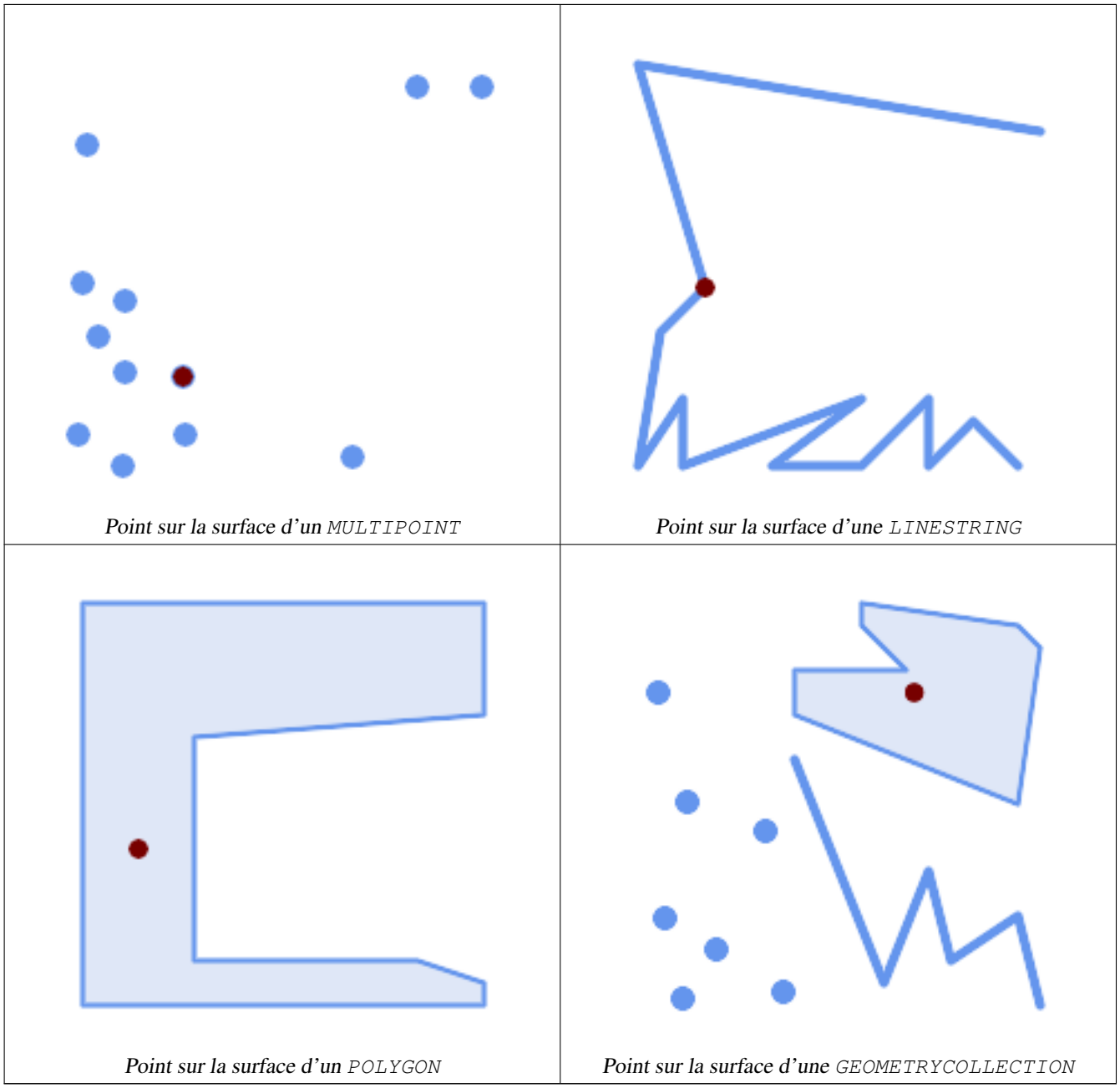
Renvoie un `POINT` qui est garanti de se trouver à l'intérieur d'une surface (`POLYGON`, `MULTIPOLYGON`, et `CURVEPOLYGON`). Dans PostGIS, cette fonction fonctionne également pour les géométries de lignes et de points.

Cette méthode implémente la spécification OGC Simple Features Implementation Specification for SQL 1.1. s3.2.14.2 // s3.2.18.2

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 8.1.5, 9.5.6. Les spécifications définissent ST_PointOnSurface pour les géométries de surface uniquement. PostGIS étend la fonction pour prendre en charge tous les types de géométrie courants. D'autres bases de données (Oracle, DB2, ArcSDE) semblent ne prendre en charge cette fonction que pour les surfaces. SQL Server 2008 prend en charge tous les types de géométrie courants.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples



Point sur la surface d'un `MULTIPOINT`

Point sur la surface d'une `LINESTRING`

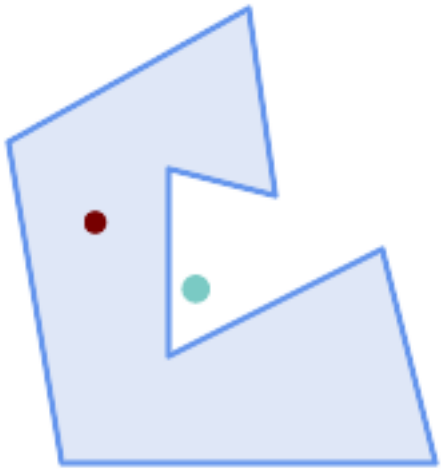
Point sur la surface d'un `POLYGON`

Point sur la surface d'une `GEOMETRYCOLLECTION`

```
SELECT ST_AsText(ST_PointOnSurface('POINT(0 5)'::geometry));
```

```
------------
 POINT(0 5)

SELECT ST_AsText(ST_PointOnSurface('LINESTRING(0 5, 0 10)'::geometry));
------------
 POINT(0 5)

SELECT ST_AsText(ST_PointOnSurface('POLYGON((0 0, 0 5, 5 5, 5 0, 0 0))'::geometry));
----------------
 POINT(2.5 2.5)

SELECT ST_AsEWKT(ST_PointOnSurface(ST_GeomFromEWKT('LINESTRING(0 5 1, 0 0 1, 0 10 2)')));
----------------
 POINT(0 0 1)
```

Exemple: Le résultat de ST_PointOnSurface est garanti à l'intérieur des polygones, alors que le point calculé par ST_Centroid peut se trouver à l'extérieur.



Rouge : point sur la surface ; Vert : centroïde

```
SELECT ST_AsText(ST_PointOnSurface(geom)) AS pt_on_surf,
       ST_AsText(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;

   pt_on_surf    |                  centroid
-----------------+---------------------------------------------
 POINT(62.5 110) | POINT(100.18264840182648 85.11415525114155)
```

Voir aussi

ST_Centroid, ST_MaximumInscribedCircle

7.14.19 ST_Polygonize

ST_Polygonize — Calcule une collection de polygones formés à partir du tracé d'un ensemble de géométries.

Synopsis

geometry **ST_Polygonize**(geometry set geomfield);
geometry **ST_Polygonize**(geometry[] geom_array);

Description

Crée une GeometryCollection contenant les polygones formés par le tracé d'un ensemble de géométries. Si le tracé en entrée ne forme aucun polygone, une GeometryCollection vide est renvoyée.

Cette fonction crée des polygones couvrant toutes les zones délimitées. Si le résultat est destiné à former une géométrie polygonale valide, utilisez ST_BuildArea pour éviter que des trous ne soient remplis.

Note
La géométrie d'entrée doit être correctement nouée pour que cette fonction fonctionne correctement. Pour s'assurer que l'entrée est nouée, utilisez ST_Node sur la géométrie d'entrée avant de la polygoniser.

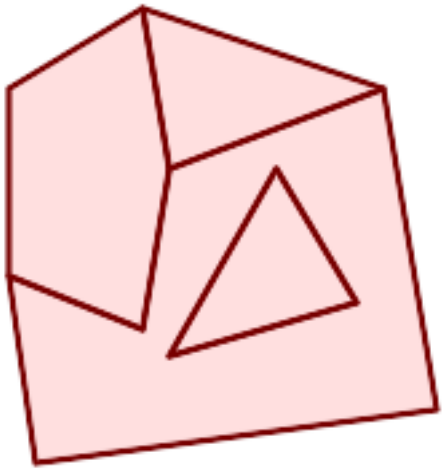
Note
Les collections de géométries peuvent être difficiles à manipuler avec des outils externes. Utilisez ST_Dump pour convertir le résultat polygonisé en polygones séparés.

Effectué par le module GEOS.

Disponibilité : 1.0.0RC1

Exemples

Lignes en entrée



Résultat polygonisé

```
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_AsText( ST_Polygonize( geom ))
    FROM data;

------------------------------------------------------------------------------------------
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 ((20 90, 20 160, 70 190, 80 130, 70 70, 20 90)),
                    POLYGON ((160 160, 80 130, 70 190, 160 160)),
                    POLYGON ((80 60, 120 130, 150 80, 80 60)))
```

Polygonisation d'une table de lignes :

```
SELECT ST_AsEWKT(ST_Polygonize(geom_4269)) As geomtextrep
FROM (SELECT geom_4269 FROM ma.suffolk_edges) As foo;

------------------------------------
 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)))

--Use ST_Dump to dump out the polygonize geoms into individual polygons
SELECT ST_AsEWKT((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;

------------------------
 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))
```

Voir aussi

ST_BuildArea, ST_Dump, ST_Node

7.14.20 ST_ReducePrecision

ST_ReducePrecision — Renvoie une géométrie valide dont les points sont arrondis en fonction de la tolérance de la grille.

Synopsis

geometry **ST_ReducePrecision**(geometry g, float8 gridsize);

Description

Renvoie une géométrie valide dont tous les points sont arrondis à la tolérance de grille fournie, et dont les caractéristiques inférieures à la tolérance sont supprimées.

Contrairement à ST_SnapToGrid, la géométrie renvoyée sera valide, sans auto-intersections d'anneaux ni composants réduits.

La réduction de précision peut être utilisée pour :

- faire correspondre la précision des coordonnées à la précision des données
- réduire le nombre de coordonnées nécessaires pour représenter une géométrie
- garantir une sortie géométrique valide vers des formats qui utilisent une précision moindre (par exemple, des formats de texte tels que WKT, GeoJSON ou KML lorsque le nombre de décimales de sortie est limité).
- exporter une géométrie valide vers des systèmes qui utilisent une précision plus faible ou limitée (par exemple, SDE, valeur de tolérance Oracle)

Disponibilité : 3.1.0.

Nécessite GEOS >= 3.9.0.

Exemples

```
SELECT ST_AsText(ST_ReducePrecision('POINT(1.412 19.323)', 0.1));
    st_astext
-----------------
 POINT(1.4 19.3)

SELECT ST_AsText(ST_ReducePrecision('POINT(1.412 19.323)', 1.0));
  st_astext
-------------
 POINT(1 19)

SELECT ST_AsText(ST_ReducePrecision('POINT(1.412 19.323)', 10));
  st_astext
-------------
 POINT(0 20)
```

La réduction de la précision permet de réduire le nombre de sommets

```
SELECT ST_AsText(ST_ReducePrecision('LINESTRING (10 10, 19.6 30.1, 20 30, 20.3 30, 40 40)', ↩
    1));
  st_astext
-------------
 LINESTRING (10 10, 20 30, 40 40)
```

La réduction de la précision divise les polygones si nécessaire pour garantir la validité

```
SELECT ST_AsText(ST_ReducePrecision('POLYGON ((10 10, 60 60.1, 70 30, 40 40, 50 10, 10 10)) ↩
    ', 10));
  st_astext
-------------
 MULTIPOLYGON (((60 60, 70 30, 40 40, 60 60)), ((40 40, 50 10, 10 10, 40 40)))
```

Voir aussi

ST_SnapToGrid, ST_Simplify, ST_SimplifyVW

7.14.21 ST_SharedPaths

ST_SharedPaths — Renvoie une collection contenant les chemins partagés par les deux lignes/multilignes en entrée.

Synopsis

geometry **ST_SharedPaths**(geometry lineal1, geometry lineal2);

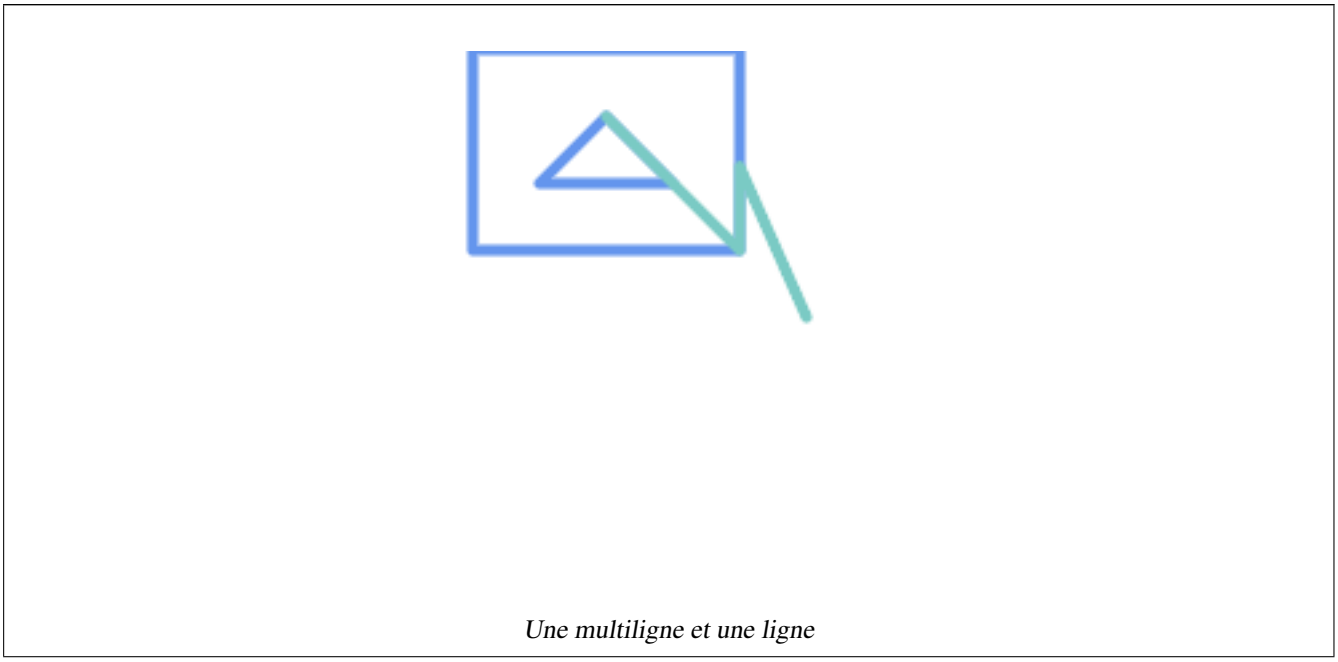
Description

Renvoie une collection contenant les chemins partagés par les deux géométries d'entrée. Ceux qui vont dans la même direction sont dans le premier élément de la collection, ceux qui vont dans la direction opposée sont dans le deuxième élément. Les chemins eux-mêmes sont donnés dans la direction de la première géométrie.

Effectué par le module GEOS.

Disponibilité : 2.0.0

Exemples : Trouver des chemins communs



Une multiligne et une ligne

Le chemin partagé de la multiligne et de la ligne superposé aux géométries originales.

```
SELECT ST_AsText(
  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(151 100,126 156.25,126 125,90 161, 76 175)')
    )
  ) As wkt

                                wkt
------------------------------------------------------------
GEOMETRYCOLLECTION(MULTILINESTRING((126 156.25,126 125),
 (101 150,90 161),(90 161,76 175)),MULTILINESTRING EMPTY)
```

```
same example but linestring orientation flipped

SELECT ST_AsText(
  ST_SharedPaths(
   ST_GeomFromText('LINESTRING(76 175,90 161,126 125,126 156.25,151 100)'),
   ST_GeomFromText('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),
       (51 150,101 150,76 175,51 150))')
    )
  ) As wkt

                                wkt
------------------------------------------------------------
GEOMETRYCOLLECTION(MULTILINESTRING EMPTY,
MULTILINESTRING((76 175,90 161),(90 161,101 150),(126 125,126 156.25)))
```

Voir aussi

ST_Dump, ST_GeometryN, ST_NumGeometries

7.14.22 ST_Simplify

ST_Simplify — Renvoie une représentation simplifiée d'une géométrie, en utilisant l'algorithme de Douglas-Peucker.

Synopsis

geometry **ST_Simplify**(geometry geom, float tolerance);
geometry **ST_Simplify**(geometry geom, float tolerance, boolean preserveCollapsed);

Description

Calcule une représentation simplifiée d'une géométrie en utilisant l'algorithme Douglas-Peucker. La variable `tolerance` de simplification est une valeur de distance, dans les unités du SRS d'entrée. La simplification supprime les sommets dont la distance au réseau simplifié est inférieure à la distance de tolérance. Le résultat peut ne pas être valide même si l'entrée l'est.

La fonction peut être appelée avec n'importe quel type de géométrie (y compris les GeometryCollections), mais seuls les lignes et polygones sont simplifiés. Les extrémités des lignes sont préservées.

L'indicateur `preserveCollapsed` permet de conserver les petites géométries qui seraient autrement supprimées à la tolérance donnée. Par exemple, si une ligne de 1 m de long est simplifiée avec une tolérance de 10 m, lorsque `preserveCollapsed` est vrai, la ligne ne disparaîtra pas. Cette option est utile pour le rendu, afin d'éviter que de très petits éléments ne disparaissent de la carte.

Note
La géométrie renvoyée peut perdre sa simplicité (voir ST_IsSimple), la topologie peut ne pas être préservée et les résultats polygonaux peuvent être invalides (voir ST_IsValid). Utilisez ST_SimplifyPreserveTopology pour préserver la topologie et garantir la validité.

Note
Cette fonction ne préserve pas les limites partagées entre les polygones. Utilisez ST_CoverageSimplify si cela est nécessaire.

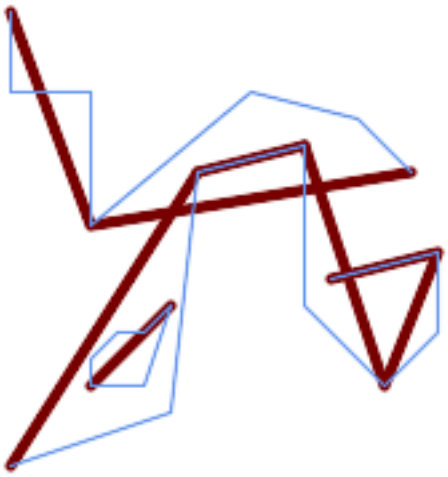
Disponibilité : 1.2.2

Exemples

Un cercle trop simplifié devient un triangle, moyennement un octogone,

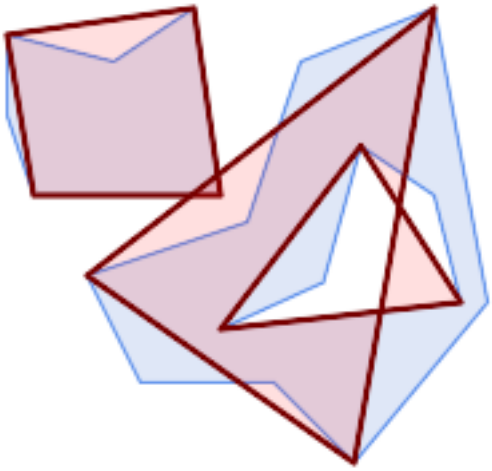
```
SELECT ST_Npoints(geom) AS np_before,
       ST_NPoints(ST_Simplify(geom, 0.1)) AS np01_notbadcircle,
       ST_NPoints(ST_Simplify(geom, 0.5)) AS np05_notquitecircle,
       ST_NPoints(ST_Simplify(geom, 1)) AS np1_octagon,
       ST_NPoints(ST_Simplify(geom, 10)) AS np10_triangle,
       (ST_Simplify(geom, 100) is null) AS  np100_geometrygoesaway
  FROM (SELECT ST_Buffer('POINT(1 3)', 10,12) As geom) AS t;

 np_before | np01_notbadcircle | np05_notquitecircle | np1_octagon | np10_triangle |  ↩
     np100_geometrygoesaway
-----------+-------------------+---------------------+-------------+---------------+----------------

        49 |                33 |                  17 |           9 |             4 | t
```

Simplification d'un ensemble de lignes. Les lignes peuvent s'intersecter après simplification.



```
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))',
    40);
```

Simplification d'un polygone multiple. Les résultats polygonaux peuvent être invalides.



```
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)))',
    40);
```

Voir aussi

ST_IsSimple, ST_SimplifyPreserveTopology, ST_SimplifyVW, ST_CoverageSimplify, Topologie ST_Simplify

7.14.23 ST_SimplifyPreserveTopology

ST_SimplifyPreserveTopology — Renvoie une représentation simplifiée et valide d'une géométrie, en utilisant l'algorithme de Douglas-Peucker.

Synopsis

geometry **ST_SimplifyPreserveTopology**(geometry geom, float tolerance);

Description

Calcule une représentation simplifiée d'une géométrie en utilisant une variante de l'algorithme Douglas-Peucker qui limite la simplification pour garantir que le résultat a la même topologie que l'entrée. Le paramètre de `tolerance` de simplification est une valeur de distance, dans les unités du SRS d'entrée. La simplification supprime les sommets dont la distance au réseau simplifiée est inférieure à la distance de tolérance, tant que la topologie est préservée. Le résultat sera valide et simple si l'entrée est.

La fonction peut être appelée avec n'importe quel type de géométrie (y compris les GeometryCollections), mais seuls les éléments linéaires et polygonaux sont simplifiés. Pour les éléments polygonaux, le résultat aura le même nombre d'anneaux (coquilles et trous), et les anneaux ne se croiseront pas. Les extrémités des anneaux peuvent être simplifiées. Pour les entrées linéaires, le résultat aura le même nombre de lignes et les lignes ne se croiseront pas si elles ne le faisaient pas dans la géométrie d'origine. Les extrémités de la géométrie linéaire sont préservées.

Note
Cette fonction ne préserve pas les limites partagées entre les polygones. Utilisez ST_CoverageSimplify si cela est nécessaire.

Effectué par le module GEOS.

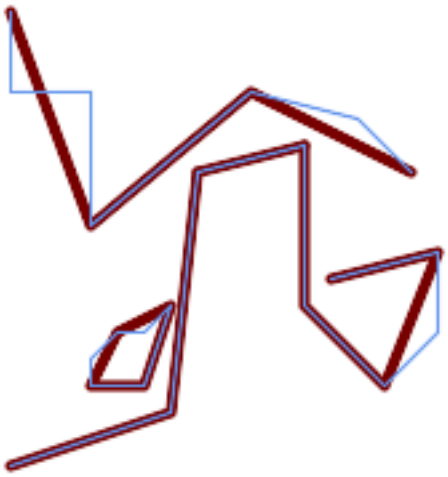
Disponibilité : 1.3.3

Exemples

Pour le même exemple que ST_Simplify, ST_SimplifyPreserveTopology empêche toute simplification excessive. Le cercle peut au maximum devenir un carré.

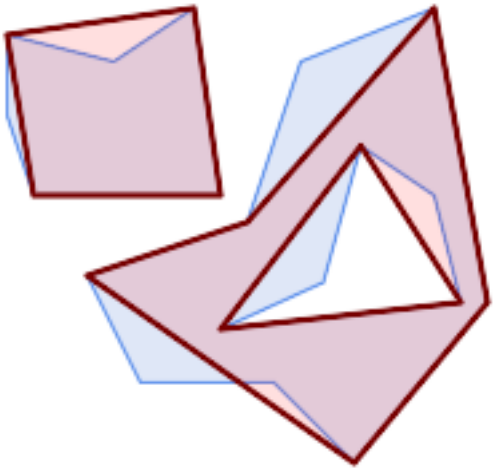
```
SELECT  ST_Npoints(geom) AS np_before,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.1)) AS np01_notbadcircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.5)) AS np05_notquitecircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 1))   AS np1_octagon,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 10))  AS np10_square,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 100)) AS np100_stillsquare
FROM (SELECT ST_Buffer('POINT(1 3)', 10,12) AS geom) AS t;

 np_before | np01_notbadcircle | np05_notquitecircle | np1_octagon | np10_square |  ↩
    np100_stillsquare
-----------+-------------------+---------------------+-------------+-------------+------------------

        49 |                33 |                  17 |           9 |           5 |  ↩
                              5
```

Simplification d'un ensemble de lignes, en préservant la topologie des lignes qui ne se croisent pas.



```
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))',
    40);
```

Simplification d'un multipolygone, en préservant la topologie des coquilles et des trous.



```
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)))',
    40);
```

Voir aussi

ST_Simplify, ST_SimplifyVW, ST_CoverageSimplify

7.14.24 ST_SimplifyPolygonHull

ST_SimplifyPolygonHull — Calcule une enveloppe extérieure ou intérieure simplifiée préservant la topologie d'une géométrie polygonale.

Synopsis

geometry **ST_SimplifyPolygonHull**(geometry param_geom, float vertex_fraction, boolean is_outer = true);

Description

Calcule une enveloppe extérieure ou intérieure simplifiée préservant la topologie d'une géométrie polygonale. Une enveloppe extérieure couvre complètement la géométrie d'entrée. Une enveloppe intérieure est entièrement couverte par la géométrie d'entrée. Le résultat est une géométrie polygonale formée par un sous-ensemble de sommets d'entrée. Les multipolygones et les trous sont gérés et produisent un résultat ayant la même structure que la géométrie d'entrée.

La réduction du nombre de sommets est contrôlée par le paramètre `vertex_fraction`, qui est un nombre compris entre 0 et 1. Les valeurs inférieures produisent des résultats plus simples, avec un nombre de vertex plus faible et une concavité moindre. Pour les enveloppes extérieures et intérieures, une fraction de sommet de 1,0 produit la géométrie originale. Pour les enveloppes extérieures, une valeur de 0,0 produit l'enveloppe convexe (pour un polygone unique) ; pour les enveloppes intérieures, elle produit un triangle.

Le processus de simplification consiste à supprimer progressivement les angles concaves qui contiennent le moins de surface, jusqu'à ce que l'objectif de nombre de vertex soit atteint. Il empêche les arêtes de se croiser, de sorte que le résultat est toujours une géométrie polygonale valide.

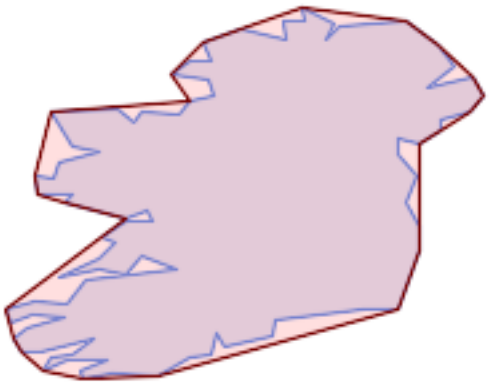
Pour obtenir de meilleurs résultats avec des géométries contenant des segments de ligne relativement longs, il peut être nécessaire de "segmenter" l'entrée, comme indiqué ci-dessous.

Effectué par le module GEOS.

Disponibilité : 3.3.0.

Nécessite GEOS >= 3.11.0.

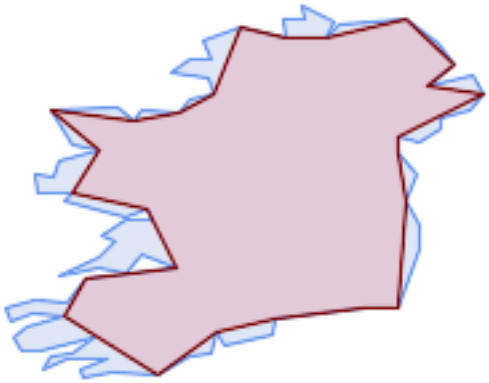
Exemples



Enveloppe extérieure d'un polygone

```
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))',
```

```
    0.3);
```



Enveloppe intérieure d'un polygone

```
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))',
    0.3, false);
```

Simplification de l'enveloppe extérieure d'un MultiPolygone, avec segmentation

```
SELECT ST_SimplifyPolygonHull(
  ST_Segmentize(ST_Letters('xt'), 2.0),
    0.1);
```

Voir aussi

ST_ConvexHull, ST_SimplifyVW, ST_ConcaveHull, ST_Segmentize

7.14.25 ST_SimplifyVW

ST_SimplifyVW — Renvoie une représentation simplifiée d'une géométrie, en utilisant l'algorithme de Visvalingam-Whyatt

Synopsis

geometry **ST_SimplifyVW**(geometry geom, float tolerance);

Description

Renvoie une représentation simplifiée d'une géométrie à l'aide de l'algorithme Visvalingam-Whyatt. La `tolérance` de simplification est une valeur de surface, dans les unités du SRS d'entrée. La simplification supprime les sommets qui forment des "coins" dont la surface est inférieure à la tolérance. Le résultat peut ne pas être valide même si l'entrée l'est.

La fonction peut être appelée avec n'importe quel type de géométrie (y compris les GeometryCollections), mais seuls les lignes et polygones sont simplifiés. Les extrémités des lignes sont préservées.

Note
La géométrie renvoyée peut perdre sa simplicité (voir ST_IsSimple), la topologie peut ne pas être préservée et les résultats polygonaux peuvent être invalides (voir ST_IsValid). Utilisez ST_SimplifyPreserveTopology pour préserver la topologie et garantir la validité. ST_CoverageSimplify préserve également la topologie et la validité.

Note
Cette fonction ne préserve pas les limites partagées entre les polygones. Utilisez ST_CoverageSimplify si cela est nécessaire.

Note
Cette fonction gère la 3D et la troisième dimension affectera le résultat.

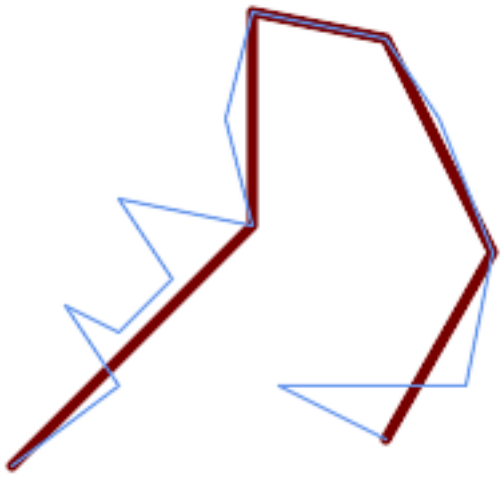
Disponibilité : 2.2.0

Exemples

Une LineString est simplifiée avec une tolérance de surface minimale de 30.

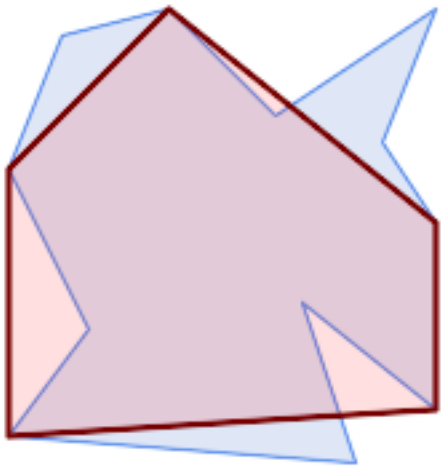
```
SELECT ST_AsText(ST_SimplifyVW(geom,30)) simplified
  FROM (SELECT 'LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)'::geometry AS geom) AS t;

 simplified
------------------------------
LINESTRING(5 2,7 25,10 10)
```

Simplification d'une ligne.



```
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)',
    1600);
```

Simplification d'un polygone.



```
SELECT ST_SimplifyVW(
  '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)))',
    40);
```

Voir aussi

ST_SetEffectiveArea, ST_Simplify, ST_SimplifyPreserveTopology, ST_CoverageSimplify, Topologie ST_Simplify

7.14.26 ST_SetEffectiveArea

ST_SetEffectiveArea — Définit la surface effective de chaque sommet, en utilisant l'algorithme Visvalingam-Whyatt.

Synopsis

geometry **ST_SetEffectiveArea**(geometry geom, float threshold = 0, integer set_area = 1);

Description

Définit la surface effective de chaque sommet, en utilisant l'algorithme Visvalingam-Whyatt. La surface effective est stockée comme la valeur M du sommet. Si le paramètre optionnel "threshold" est utilisé, une géométrie simplifiée sera renvoyée, contenant uniquement les sommets dont la surface effective est supérieure ou égale à la valeur seuil.

Cette fonction peut être utilisée pour la simplification côté serveur lorsqu'un seuil est spécifié. Une autre option consiste à utiliser une valeur seuil de zéro. Dans ce cas, la géométrie complète sera renvoyée avec les surfaces effectives sous forme de valeurs M, qui peuvent être utilisées par le client pour simplifier très rapidement.

Cette fonction n'agit qu'avec des (multi)lignes et des (multi)polygones, mais vous pouvez l'appeler en toute sécurité avec n'importe quel type de géométrie. Comme la simplification se produit objet par objet, vous pouvez également utiliser une collection de géométries dans cette fonction.

Note
Notez que la géométrie retournée peut perdre sa simplicité (voir ST_IsSimple)

Note
Notez que la topologie peut ne pas être préservée et donner lieu à des géométries non valides. Utilisez (voir ST_SimplifyPreserveTopology) pour préserver la topologie.

Note
La géométrie de sortie perdra toutes les informations précédentes concernant les valeurs M

Note
Cette fonction gère la 3D et la troisième dimension affecte la surface effective

Disponibilité : 2.2.0

Exemples

Calcul de la surface effective d'une LineString. Comme nous utilisons une valeur seuil de zéro, tous les sommets de la géométrie d'entrée sont renvoyés.

```
select ST_AsText(ST_SetEffectiveArea(geom)) all_pts, ST_AsText(ST_SetEffectiveArea(geom,30) ↩
    ) thrshld_30
FROM (SELECT  'LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)'::geometry geom) As foo;
-result
 all_pts | thrshld_30
-----------+-------------------+
LINESTRING M (5 2 3.40282346638529e+38,3 8 29,6 20 1.5,7 25 49.5,10 10 3.40282346638529e ↩
    +38) | LINESTRING M (5 2 3.40282346638529e+38,7 25 49.5,10 10 3.40282346638529e+38)
```

Voir aussi

ST_SimplifyVW

7.14.27 ST_TriangulatePolygon

ST_TriangulatePolygon — Calcule la triangulation de Delaunay contrainte des polygones

Synopsis

geometry **ST_TriangulatePolygon**(geometry geom);

Description

Calcule la triangulation de Delaunay contrainte des polygones. Les trous et les multipolygones sont pris en charge.

La "constrained Delaunay triangulation" d'un polygone est un ensemble de triangles formés à partir des sommets du polygone et le couvrant exactement, avec l'angle intérieur total maximal parmi toutes les triangulations possibles. Elle constitue la triangulation de "meilleure qualité" du polygone.

Disponibilité : 3.3.0.

Nécessite GEOS >= 3.11.0.

Exemple

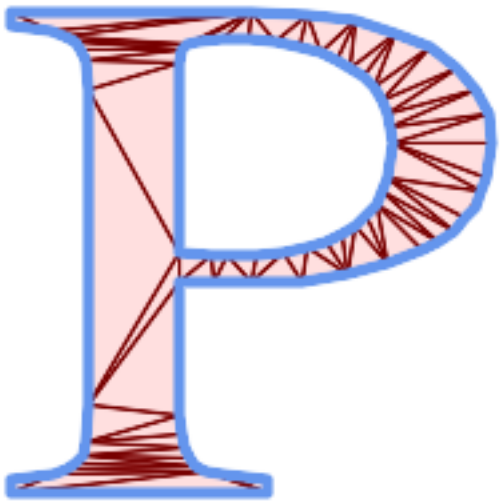
Triangulation d'un carré.

```
SELECT ST_AsText(
    ST_TriangulatePolygon('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))'));

                                st_astext
-------------------------------------------------------------------------
 GEOMETRYCOLLECTION(POLYGON((0 0,0 1,1 1,0 0)),POLYGON((1 1,1 0,0 0,1 1)))
```

Exemple

Triangulation de la lettre P.

```
SELECT ST_AsText(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))'
    ));
```



Triangulation de polygone

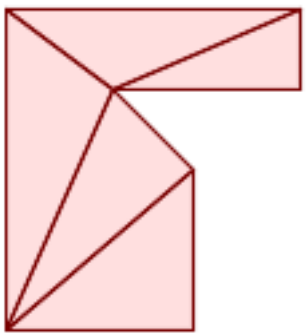
Même exemple que ST_Tesselate

```
SELECT ST_TriangulatePolygon(
                'POLYGON (( 10 190, 10 70, 80 70, 80 130, 50 160, 120 160, 120 190, 10 190  ↩
                    ))'::geometry
                );
```

Sortie ST_AsText

```
GEOMETRYCOLLECTION(POLYGON((50 160,120 190,120 160,50 160))
    ,POLYGON((10 70,80 130,80 70,10 70))
    ,POLYGON((50 160,10 70,10 190,50 160))
    ,POLYGON((120 190,50 160,10 190,120 190))
    ,POLYGON((80 130,10 70,50 160,80 130)))
```

Polygone d'origine



Polygone triangulé

Voir aussi

ST_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_Tesselate

7.14.28 ST_VoronoiLines

ST_VoronoiLines — Renvoie les limites des polygones de Voronoï des sommets d'une géométrie.

Synopsis

geometry **ST_VoronoiLines**(geometry geom , float8 tolerance = 0.0 , geometry extend_to = NULL);

Description

Calcule un Voronoi diagram bidimensionnel à partir des sommets de la géométrie fournie et renvoie les limites entre les cellules de la représentation sous la forme d'une MultiLineString. Renvoie null si la géométrie d'entrée est null. Renvoie une collection de géométries vide si la géométrie d'entrée ne contient qu'un seul sommet. Renvoie une collection de géométries vide si l'enveloppe `extend_to` a une surface nulle.

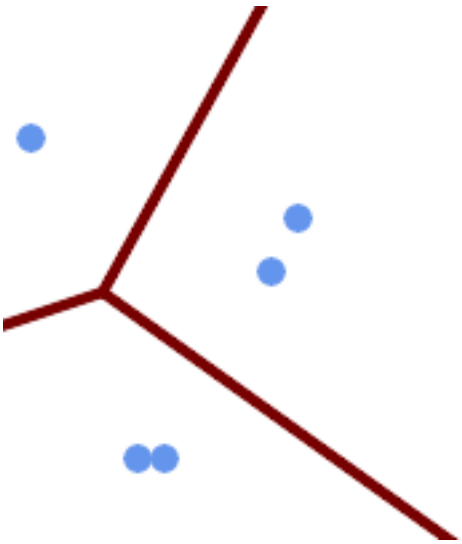
Paramètres optionnels :

- `tolerance` : distance à laquelle les sommets seront considérés comme équivalents. La robustesse de l'algorithme peut être améliorée en fournissant une distance de tolérance non nulle. (par défaut = 0.0)
- `extend_to` : S'il est présent, la représentation est étendue pour couvrir l'enveloppe de la géométrie fournie, à moins qu'elle ne soit plus petite que l'enveloppe par défaut (default = NULL, l'enveloppe par défaut est la boîte de délimitation de l'entrée élargie d'environ 50 %).

Effectué par le module GEOS.

Disponibilité : 2.3.0

Exemples



Lignes de la représentation de Voronoï, avec une tolérance de 30 unités

```
SELECT ST_VoronoiLines(
            'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120)'::geometry,
            30) AS geom;
```

```
ST_AsText output
MULTILINESTRING((135.555555555556 270,36.8181818181818 92.2727272727273),(36.8181818181818 ↩
    92.2727272727273,-110 43.3333333333333),(230 -45.7142857142858,36.8181818181818 ↩
    92.2727272727273))
```

Voir aussi

ST_DelaunayTriangles, ST_VoronoiPolygons

7.14.29 ST_VoronoiPolygons

ST_VoronoiPolygons — Renvoie les cellules de la représentation de Voronoï des sommets d'une géométrie.

Synopsis

geometry **ST_VoronoiPolygons**(geometry geom , float8 tolerance = 0.0 , geometry extend_to = NULL);

Description

Calcule un diagramme de Voronoï bidimensionnel à partir des sommets de la géométrie fournie. Le résultat est une GEOMETRYCOLLECTION de POLYGONES qui couvre une enveloppe plus grande que l'étendue des sommets d'entrée. Renvoie null si la géométrie d'entrée est null. Renvoie une collection de géométries vide si la géométrie d'entrée ne contient qu'un seul sommet. Renvoie une collection de géométries vide si l'enveloppe `extend_to` a une surface nulle.

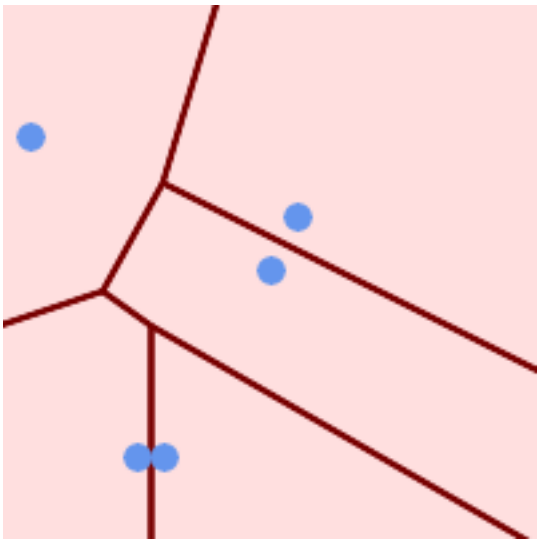
Paramètres optionnels :

- `tolerance` : distance à laquelle les sommets seront considérés comme équivalents. La robustesse de l'algorithme peut être améliorée en fournissant une distance de tolérance non nulle. (par défaut = 0.0)
- `extend_to` : S'il est présent, la représentation est étendue pour couvrir l'enveloppe de la géométrie fournie, à moins qu'elle ne soit plus petite que l'enveloppe par défaut (default = NULL, l'enveloppe par défaut est la boîte de délimitation de l'entrée élargie d'environ 50 %).

Effectué par le module GEOS.

Disponibilité : 2.3.0

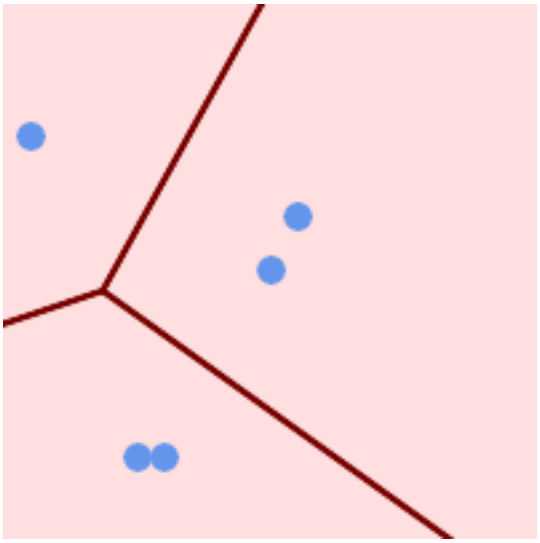
Exemples



Points superposés à la représentation de Voronoï

```
SELECT ST_VoronoiPolygons(
                'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120)'::geometry
            ) AS geom;
```

```
ST_AsText output
GEOMETRYCOLLECTION(POLYGON((-110 43.3333333333333,-110 270,100.5 270,59.3478260869565  ↩
    132.826086956522,36.8181818181818 92.2727272727273,-110 43.3333333333333)),
POLYGON((55 -90,-110 -90,-110 43.3333333333333,36.8181818181818 92.2727272727273,55  ↩
    79.2857142857143,55 -90)),
POLYGON((230 47.5,230 -20.7142857142857,55 79.2857142857143,36.8181818181818  ↩
    92.2727272727273,59.3478260869565 132.826086956522,230 47.5)),POLYGON((230  ↩
    -20.7142857142857,230 -90,55 -90,55 79.2857142857143,230 -20.7142857142857)),
POLYGON((100.5 270,230 270,230 47.5,59.3478260869565 132.826086956522,100.5 270)))
```



Représentation de Voronoï, avec une tolérance de 30 unités

```
SELECT ST_VoronoiPolygons(
            'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120)'::geometry,
            30) AS geom;
```

```
ST_AsText output
GEOMETRYCOLLECTION(POLYGON((-110 43.3333333333333,-110 270,100.5 270,59.3478260869565  ↩
    132.826086956522,36.8181818181818 92.2727272727273,-110 43.3333333333333)),
POLYGON((230 47.5,230 -45.7142857142858,36.8181818181818 92.2727272727273,59.3478260869565  ↩
    132.826086956522,230 47.5)),POLYGON((230 -45.7142857142858,230 -90,-110 -90,-110  ↩
    43.3333333333333,36.8181818181818 92.2727272727273,230 -45.7142857142858)),
POLYGON((100.5 270,230 270,230 47.5,59.3478260869565 132.826086956522,100.5 270)))
```

Voir aussi

ST_DelaunayTriangles, ST_VoronoiLines

7.15 Couvertures

7.15.1 ST_CoverageInvalidEdges

ST_CoverageInvalidEdges — Fonction window qui trouve les endroits où les polygones ne forment pas une couverture valide.

Synopsis

geometry **ST_CoverageInvalidEdges**(geometry winset geom, float8 tolerance = 0);

Description

Une fonction window qui vérifie si les polygones de la partition window forment une couverture polygonale valide. Elle renvoie des indicateurs linéaires indiquant l'emplacement des arêtes non valides (le cas échéant) dans chaque polygone.

Un ensemble de polygones valides est une couverture valide si les conditions suivantes sont remplies :

- **Non-overlapping** - les polygones ne se chevauchent pas (leurs intérieurs ne s'intersectent pas)
- **Edge-Matched** - les sommets situés le long d'arêtes communes sont identiques

En tant que fonction window, une valeur est renvoyée pour chaque polygone d'entrée. Pour les polygones qui ne respectent pas une ou plusieurs des conditions de validité, la valeur renvoyée est un MULTILINESTRING contenant les arêtes problématiques. Les polygones valides du point de vue de la couverture renvoient la valeur NULL. Les géométries non polygonales ou vides produisent également des valeurs NULL.

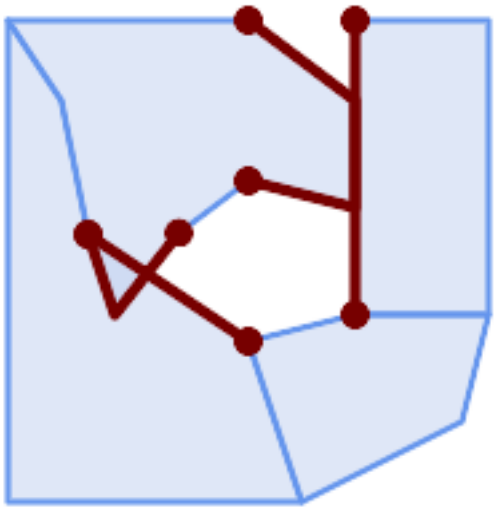
Les conditions permettent à une couverture valide de contenir des trous (espaces entre les polygones), pour autant que les polygones environnants soient adaptés aux bords. Toutefois, des espaces très étroits ne sont souvent pas souhaitables. Si le paramètre *tolérance* est spécifié avec une distance non nulle, les arêtes formant des espaces plus étroits seront également renvoyées comme non valides.

Les polygones dont la validité de la couverture est vérifiée doivent également être des géométries valides. Ceci peut être vérifié avec ST_IsValid.

Disponibilité : 3.4.0

Nécessite GEOS >= 3.12.0

Exemples



Arêtes non valides dues au chevauchement et à la non-concordance des sommets

```
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_AsText(ST_CoverageInvalidEdges(geom) OVER ())
  FROM coverage;

 id |               st_astext
----+----------------------------------------
  1 | LINESTRING (40 110, 100 70)
  2 | MULTILINESTRING ((100 130, 140 120, 140 160, 100 190), (40 110, 50 80, 74 110.5))
  3 | LINESTRING (140 80, 140 190)
  4 | null
```

```
-- Test entire table for coverage validity
SELECT true = ALL (
    SELECT ST_CoverageInvalidEdges(geom) OVER () IS NULL
    FROM coverage
    );
```

Voir aussi

ST_IsValid, ST_CoverageUnion, ST_CoverageSimplify

7.15.2 ST_CoverageSimplify

ST_CoverageSimplify — Fonction window qui simplifie les bords d'une couverture polygonale.

Synopsis

geometry **ST_CoverageSimplify**(geometry winset geom, float8 tolerance, boolean simplifyBoundary = true);

Description

Une fonction window qui simplifie les bords des polygones dans une couverture polygonale. La simplification préserve la topologie de la couverture. Cela signifie que les polygones de sortie simplifiés sont cohérents le long des arêtes communes et forment toujours une couverture valide.

La simplification utilise une variante de l'algorithme Visvalingam-Whyatt. Le paramètre `tolérance` a des unités de distance et est approximativement égal à la racine carrée des zones triangulaires à simplifier.

Pour simplifier uniquement les bords "internes" de la couverture (ceux qui sont partagés par deux polygones), définissez le paramètre `simplifyBoundary` à false.

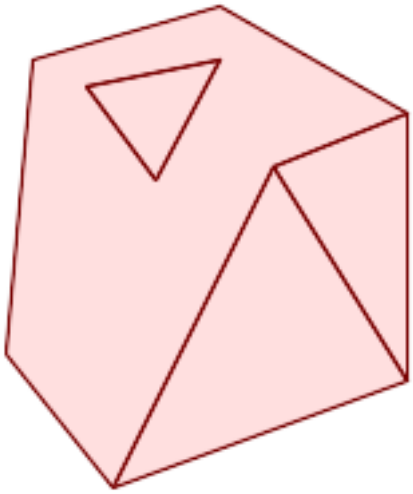
Note
Si l'entrée n'est pas une couverture valide, il peut y avoir des artefacts inattendus dans la sortie (tels que des intersections de frontières, ou des frontières séparées qui semblaient être partagées). Utilisez ST_CoverageInvalidEdges pour déterminer si une couverture est valide.

Disponibilité : 3.4.0

Nécessite GEOS >= 3.12.0

Exemples

Couverture d'entrée



Couverture simplifiée

```
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_AsText(ST_CoverageSimplify(geom, 30) OVER ())
```

```
 FROM coverage;

 id |              st_astext
----+---------------------------------------
  1 | POLYGON ((160 150, 110 130, 50 10, 10 60, 20 170, 90 190, 160 150), (40 160, 66 125,  ↩
    90 170, 40 160))
  2 | POLYGON ((40 160, 66 125, 90 170, 40 160))
  3 | POLYGON ((110 130, 160 50, 50 10, 110 130))
  4 | POLYGON ((160 150, 160 50, 110 130, 160 150))
```

Voir aussi

ST_CoverageInvalidEdges

7.15.3 ST_CoverageUnion

ST_CoverageUnion — Calcule l'union d'un ensemble de polygones formant une couverture en supprimant les arêtes communes.

Synopsis

geometry **ST_CoverageUnion**(geometry set geom);

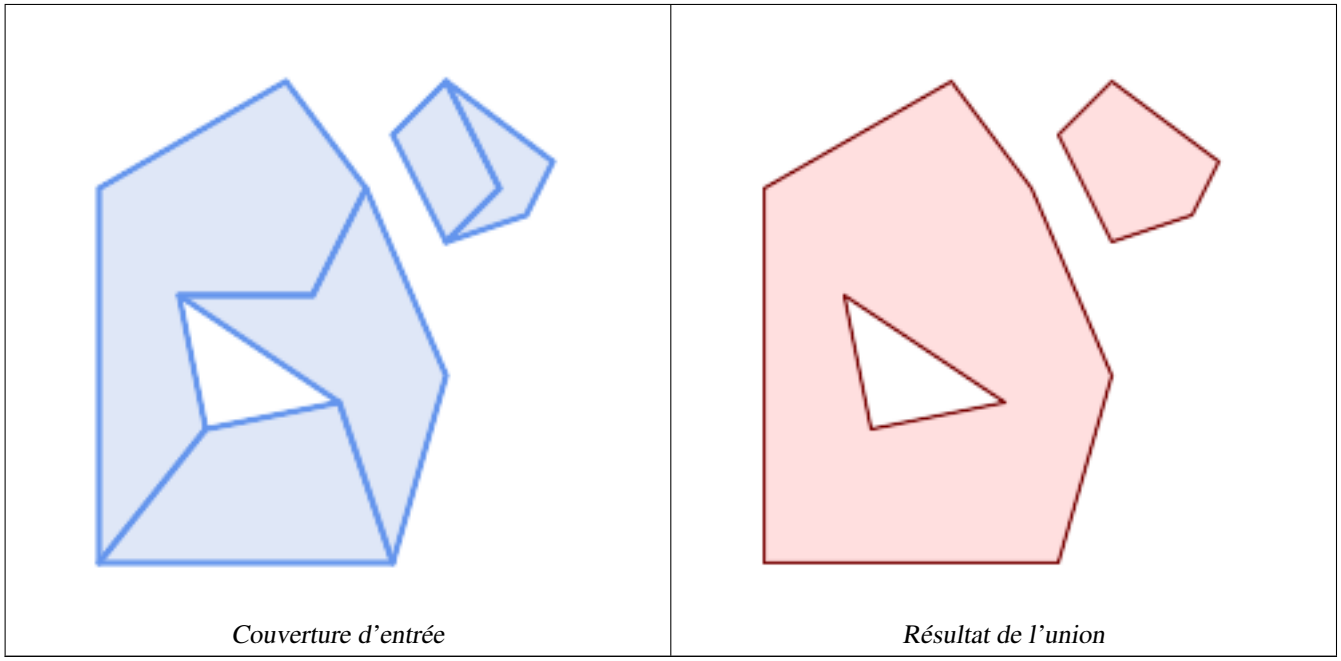
Description

Une fonction agrégée qui réunit un ensemble de polygones formant une couverture polygonale. Le résultat est une géométrie polygonale couvrant la même zone que la couverture. Cette fonction produit le même résultat que ST_Union, mais utilise la structure de la couverture pour calculer l'union beaucoup plus rapidement.

Note
Si l'entrée n'est pas une couverture valide, il peut y avoir des artefacts inattendus dans la sortie (tels que des polygones non fusionnés ou se chevauchant). Utilisez ST_CoverageInvalidEdges pour déterminer si une couverture est valide.

Disponibilité : 3.4.0 - nécessite GEOS >= 3.8.0

Exemples



Couverture d'entrée | *Résultat de l'union*

```
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_AsText(ST_CoverageUnion(geom))
  FROM coverage;

--------------------------------------
MULTIPOLYGON (((10 150, 80 190, 110 150, 140 80, 120 10, 10 10, 10 150), (50 60, 100 70, 40 ↩
     110, 50 60)), ((120 170, 140 190, 180 160, 170 140, 140 130, 120 170)))
```

Voir aussi

ST_CoverageInvalidEdges, ST_Union

7.16 Transformations affines

7.16.1 ST_Affine

ST_Affine — Appliquer une transformation affine 3D à une géométrie.

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);

Description

Applique une transformation affine 3D à la géométrie pour effectuer des opérations telles que la translation, la rotation et la mise à l'échelle en une seule étape.

Version 1: L'appel

```
ST_Affine(geom, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff)
```

représente la matrice de transformation

```
/ a  b  c  xoff \
| d  e  f  yoff |
| g  h  i  zoff |
\ 0  0  0     1 /
```

et les sommets sont transformés comme suit :

```
x' = a*x + b*y + c*z + xoff
y' = d*x + e*y + f*z + yoff
z' = g*x + h*y + i*z + zoff
```

Toutes les fonctions de translation / mise à l'échelle ci-dessous sont exprimées par une telle transformation affine.

Version 2: Applique une transformation affine 2d à la géométrie. L'appel

```
ST_Affine(geom, a, b, d, e, xoff, yoff)
```

représente la matrice de transformation

```
/  a  b  0  xoff  \       /  a  b  xoff  \
|  d  e  0  yoff  | rsp.  |  d  e  yoff  |
|  0  0  1     0  |       \  0  0     1  /
\  0  0  0     1  /
```

et les sommets sont transformés comme suit :

```
x' = a*x + b*y + xoff
y' = d*x + e*y + yoff
z' = z
```

Cette méthode est un sous-cas de la méthode 3D ci-dessus.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Disponibilité : 1.1.2. Le nom a été changé de Affine à ST_Affine dans la version 1.2.2

Note
Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
--Rotate a 3d line 180 degrees about the z axis.  Note this is long-hand for doing  ↩
    ST_Rotate();
 SELECT ST_AsEWKT(ST_Affine(geom,  cos(pi()), -sin(pi()), 0,  sin(pi()), cos(pi()), 0,  0,  ↩
     0, 1,  0, 0, 0)) As using_affine,
          ST_AsEWKT(ST_Rotate(geom, pi())) As using_rotate
        FROM (SELECT ST_GeomFromEWKT('LINESTRING(1 2 3, 1 4 3)') As geom) As foo;
        using_affine          |        using_rotate
------------------------------+-----------------------------
 LINESTRING(-1 -2 3,-1 -4 3) | LINESTRING(-1 -2 3,-1 -4 3)
(1 row)

--Rotate a 3d line 180 degrees in both the x and z axis
SELECT ST_AsEWKT(ST_Affine(geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), -sin(pi()) ↩
    , 0, sin(pi()), cos(pi()), 0, 0, 0))
        FROM (SELECT ST_GeomFromEWKT('LINESTRING(1 2 3, 1 4 3)') As geom) As foo;
           st_asewkt
--------------------------------
 LINESTRING(-1 -2 -3,-1 -4 -3)
(1 row)
```

Voir aussi

ST_Rotate, ST_Scale, ST_Translate, ST_TransScale

7.16.2 ST_Rotate

ST_Rotate — Fait pivoter une géométrie autour d'un point d'origine.

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);

Description

Fait pivoter la géométrie rotRadians dans le sens inverse des aiguilles d'une montre autour du point d'origine. Le point d'origine de la rotation peut être spécifié sous la forme d'une géométrie POINT, ou sous la forme de coordonnées x et y. Si l'origine n'est pas spécifiée, la géométrie est tournée autour de POINT(0 0).

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Amélioration : 2.0.0 des paramètres supplémentaires ont été ajoutés pour spécifier l'origine de la rotation.

Disponibilité : 1.1.2. Nom modifié de Rotate en ST_Rotate dans la version 1.2.2

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
--Rotate 180 degrees
SELECT ST_AsEWKT(ST_Rotate('LINESTRING (50 160, 50 50, 100 50)', pi()));
               st_asewkt
-----------------------------------
 LINESTRING(-50 -160,-50 -50,-100 -50)
(1 row)

--Rotate 30 degrees counter-clockwise at x=50, y=160
SELECT ST_AsEWKT(ST_Rotate('LINESTRING (50 160, 50 50, 100 50)', pi()/6, 50, 160));
                               st_asewkt
-------------------------------------------------------------------------
 LINESTRING(50 160,105 64.7372055837117,148.301270189222 89.7372055837117)
(1 row)

--Rotate 60 degrees clockwise from centroid
SELECT ST_AsEWKT(ST_Rotate(geom, -pi()/3, ST_Centroid(geom)))
FROM (SELECT 'LINESTRING (50 160, 50 50, 100 50)'::geometry AS geom) AS foo;
                           st_asewkt
-----------------------------------------------------------
 LINESTRING(116.4225 130.6721,21.1597 75.6721,46.1597 32.3708)
(1 row)
```

Voir aussi

ST_Affine, ST_RotateX, ST_RotateY, ST_RotateZ

7.16.3 ST_RotateX

ST_RotateX — Fait pivoter une géométrie autour de l'axe X.

Synopsis

geometry **ST_RotateX**(geometry geomA, float rotRadians);

Description

Fait pivoter une géométrie geomA - rotRadians autour de l'axe X.

> **Note**
> `ST_RotateX(geomA, rotRadians)` est un raccourci pour `ST_Affine(geomA, 1, 0, 0, 0, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0).`

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Disponibilité : 1.1.2. Nom modifié de RotateX en ST_RotateX dans la version 1.2.2

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
--Rotate a line 90 degrees along x-axis
SELECT ST_AsEWKT(ST_RotateX(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
                 st_asewkt
---------------------------
 LINESTRING(1 -3 2,1 -1 1)
```

Voir aussi

ST_Affine, ST_RotateY, ST_RotateZ

7.16.4 ST_RotateY

ST_RotateY — Fait pivoter une géométrie autour de l'axe Y.

Synopsis

geometry **ST_RotateY**(geometry geomA, float rotRadians);

Description

Fait pivoter une géométrie geomA - rotRadians autour de l'axe y.

Note

`ST_RotateY(geomA, rotRadians)` est un raccourci pour `ST_Affine(geomA, cos(rotRadians), 0, sin(rotRadians), 0, 1, 0, -sin(rotRadians), 0, cos(rotRadians), 0, 0, 0).`

Disponibilité : 1.1.2. Nom modifié de RotateY en ST_RotateY dans la version 1.2.2

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
--Rotate a line 90 degrees along y-axis
 SELECT ST_AsEWKT(ST_RotateY(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
                 st_asewkt
---------------------------
 LINESTRING(3 2 -1,1 1 -1)
```

Voir aussi

ST_Affine, ST_RotateX, ST_RotateZ

7.16.5 ST_RotateZ

ST_RotateZ — Fait pivoter une géométrie autour de l'axe Z.

Synopsis

geometry **ST_RotateZ**(geometry geomA, float rotRadians);

Description

Fait pivoter une géométrie geomA - rotRadians autour de l'axe Z.

Note
C'est un synonyme de ST_Rotate

Note
`ST_RotateZ(geomA, rotRadians)` est un raccourci `SELECT ST_Affine(geomA, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0, 1, 0, 0, 0).`

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Disponibilité : 1.1.2. Nom modifié de RotateZ en ST_RotateZ dans la version 1.2.2

Note
Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
--Rotate a line 90 degrees along z-axis
SELECT ST_AsEWKT(ST_RotateZ(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
                 st_asewkt
---------------------------
 LINESTRING(-2 1 3,-1 1 1)

 --Rotate a curved circle around z-axis
SELECT ST_AsEWKT(ST_RotateZ(geom, pi()/2))
FROM (SELECT ST_LineToCurve(ST_Buffer(ST_GeomFromText('POINT(234 567)'), 3)) As geom) As ↩
    foo;
```

```
----------------------------------------------------------------------------------------------------------------

 CURVEPOLYGON(CIRCULARSTRING(-567 237,-564.87867965644 236.12132034356,-564  ↩
     234,-569.12132034356 231.87867965644,-567 237))
```

Voir aussi

ST_Affine, ST_RotateX, ST_RotateY

7.16.6 ST_Scale

ST_Scale — Met à l'échelle une géométrie en fonction de facteurs donnés.

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);

Description

Met la géométrie à l'échelle en multipliant les ordonnées par les paramètres factor correspondants.

La version prenant une géométrie comme paramètre `factor` permet de passer un point 2d, 3dm, 3dz ou 4d pour définir le facteur d'échelle pour toutes les dimensions prises en charge. Les dimensions manquantes dans le point `factor` sont équivalentes à l'absence de mise à l'échelle de la dimension correspondante.

La variante à trois géométries permet de transmettre une "fausse origine" pour la mise à l'échelle. Cela permet une "mise à l'échelle sur place", par exemple en utilisant le centroïde de la géométrie comme fausse origine. Sans fausse origine, la mise à l'échelle s'effectue par rapport à l'origine réelle, de sorte que toutes les coordonnées sont simplement multipliées par le facteur d'échelle.

Note
Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Disponibilité : 1.1.0.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Amélioration : La prise en charge de la mise à l'échelle de toutes les dimensions (paramètre `factor`) a été introduite dans la version 2.2.0.

Amélioration : la prise en charge de la mise à l'échelle par rapport à une origine locale (paramètre `origin`) a été introduite dans la version 2.5.0.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge les coordonnées M.

Exemples

```
--Version 1: scale X, Y, Z
SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 0.75, 0.8));
                          st_asewkt
-------------------------------------
 LINESTRING(0.5 1.5 2.4,0.5 0.75 0.8)

--Version 2: Scale X Y
 SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 0.75));
                         st_asewkt
---------------------------------
 LINESTRING(0.5 1.5 3,0.5 0.75 1)

--Version 3: Scale X Y Z M
 SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3 4, 1 1 1 1)'),
   ST_MakePoint(0.5, 0.75, 2, -1)));
                                st_asewkt
-----------------------------------------
 LINESTRING(0.5 1.5 6 -4,0.5 0.75 2 -1)

--Version 4: Scale X Y using false origin
SELECT ST_AsText(ST_Scale('LINESTRING(1 1, 2 2)', 'POINT(2 2)', 'POINT(1 1)'::geometry));
      st_astext
---------------------
 LINESTRING(1 1,3 3)
```

Voir aussi

ST_Affine, ST_TransScale

7.16.7 ST_Translate

ST_Translate — Traduit une géométrie en fonction de décalages donnés.

Synopsis

geometry **ST_Translate**(geometry g1, float deltax, float deltay);
geometry **ST_Translate**(geometry g1, float deltax, float deltay, float deltaz);

Description

Renvoie une nouvelle géométrie dont les coordonnées sont des unités de translation delta x,delta y,delta z. Les unités sont basées sur les unités définies dans la référence spatiale (SRID) pour cette géométrie. Les unités sont basées sur les unités définies dans la référence spatiale (SRID) pour cette géométrie.

Note
Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Disponibilité : 1.2.2

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

Déplacer un point de 1 degré de longitude

```
SELECT ST_AsText(ST_Translate(ST_GeomFromText('POINT(-71.01 42.37)',4326),1,0)) As  ↩
    wgs_transgeomtxt;

        wgs_transgeomtxt
        --------------------
        POINT(-70.01 42.37)
```

Déplacer une ligne d'un degré de longitude et d'un demi-degré de latitude

```
SELECT ST_AsText(ST_Translate(ST_GeomFromText('LINESTRING(-71.01 42.37,-71.11 42.38)',4326) ↩
    ,1,0.5)) As wgs_transgeomtxt;
                    wgs_transgeomtxt
        ---------------------------------------
        LINESTRING(-70.01 42.87,-70.11 42.88)
```

Déplacer un point 3D

```
SELECT ST_AsEWKT(ST_Translate(CAST('POINT(0 0 0)' As geometry), 5, 12,3));
        st_asewkt
        ---------
        POINT(5 12 3)
```

Déplacer une courbe et un point

```
SELECT ST_AsText(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));


----------------------------------------------------------------------------------------------------------------

 GEOMETRYCOLLECTION(CURVEPOLYGON(CIRCULARSTRING(5 5,4.12 2.878,2 2,-0.121 7.1213,7 9,9 11,5 ↩
      5)),POINT(2 5))
```

Voir aussi

ST_Affine, ST_AsText, ST_GeomFromText

7.16.8 ST_TransScale

ST_TransScale — Traduit et met à l'échelle une géométrie en fonction des paramètres offset et factor spécifiés.

Synopsis

geometry **ST_TransScale**(geometry geomA, float deltaX, float deltaY, float XFactor, float YFactor);

Description

Traduit la géométrie à l'aide des paramètres deltaX et deltaY, puis la met à l'échelle à l'aide des paramètres XFactor et YFactor, uniquement en 2D.

> **Note**
> `ST_TransScale(geomA, deltaX, deltaY, XFactor, YFactor)` est un raccourci pour `ST_Affine(geomA, XFactor, 0, 0, 0, YFactor, 0, 0, 0, 1, deltaX*XFactor, deltaY*YFactor, 0)`.

> **Note**
> Avant la version 1.3.4, cette fonction se bloquait si elle était utilisée avec des géométries contenant des CURVES. Ce problème est corrigé dans la version 1.3.4+

Disponibilité : 1.1.0.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_AsEWKT(ST_TransScale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 1, 1, 2));
                  st_asewkt
-----------------------------
 LINESTRING(1.5 6 3,1.5 4 1)


--Buffer a point to get an approximation of a circle, convert to curve and then translate ↩
    1,2 and scale it 3,4
  SELECT ST_AsText(ST_Transscale(ST_LineToCurve(ST_Buffer('POINT(234 567)', 3)),1,2,3,4));


-----------------------------------------------------------------------------------------------------

 CURVEPOLYGON(CIRCULARSTRING(714 2276,711.363961030679 2267.51471862576,705 ↩
     2264,698.636038969321 2284.48528137424,714 2276))
```

Voir aussi

ST_Affine, ST_Translate

7.17 Fonctions de clustering

7.17.1 ST_ClusterDBSCAN

ST_ClusterDBSCAN — Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie d'entrée en utilisant l'algorithme DBSCAN.

Synopsis

integer **ST_ClusterDBSCAN**(geometry winset geom, float8 eps, integer minpoints);

Description

Une fonction window qui renvoie un numéro de cluster pour chaque géométrie d'entrée, en utilisant l'algorithme 2D Density-based spatial clustering of applications with noise (DBSCAN). Contrairement à ST_ClusterKMeans, elle ne nécessite pas la spécification du nombre de clusters, mais utilise les paramètres de distance (`eps`) et de densité (`minpoints`) pour déterminer chaque cluster.

Une géométrie d'entrée est ajoutée à un cluster si elle est l'une ou l'autre :

- Une géométrie "core", qui se trouve à `eps` distance d'au moins `minpoints` géométries d'entrée (y compris la sienne) ; ou
- Une géométrie "en bordure", qui se trouve dans `eps` distance d'une géométrie centrale.

Notez que les géométries situés en bordure peuvent se trouver à `eps` distance des géométries centrales dans plus d'un cluster. L'une ou l'autre assignation serait correcte, de sorte que la géométrie de bordure sera arbitrairement assignée à l'un des clusters disponibles. Dans cette situation, il est possible qu'un cluster correct soit généré avec moins de géométries `minpoints`. Pour garantir une affectation déterministe des géométries de bordure (de sorte que des appels répétés à ST_ClusterDBSCAN produisent des résultats identiques), utilisez une clause `ORDER BY` dans la définition de la fonction window. Les affectations de clusters ambigus peuvent différer d'autres implémentations DBSCAN.

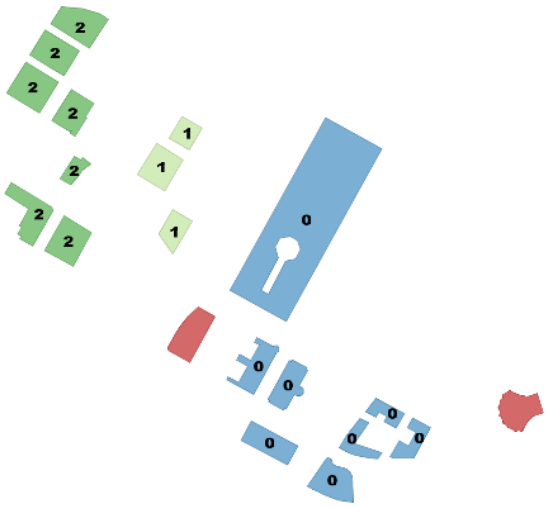
Note
Les géométries qui ne répondent pas aux critères d'appartenance à un cluster se voient attribuer un numéro de cluster NULL.

Disponibilité : 2.3.0

Cette méthode prend en charge les types Circular String et Curve.

Exemples

Regroupement des polygones à moins de 50 mètres les uns des autres, et nécessité d'avoir au moins 2 polygones par cluster.



Clusters dans un rayon de 50 mètres avec au moins 2 éléments par cluster. Les singletons ont NULL pour cid

```
SELECT name, ST_ClusterDBSCAN(geom, eps =
> 50, minpoints =
> 2) over () AS cid
FROM boston_polys
WHERE name
> '' AND building
> ''
        AND ST_DWithin(geom,
        ST_Transform(
            ST_GeomFromText('POINT ↩
    (-71.04054 42.35141)', 4326), 26986),
          500);
```

```
                name                   |  ↩
    bucket
-------------------------------------+-------- ↩

 Manulife Tower                        |  ↩
          0
 Park Lane Seaport I                   |  ↩
          0
 Park Lane Seaport II                  |  ↩
          0
 Renaissance Boston Waterfront Hotel   |  ↩
          0
 Seaport Boston Hotel                  |  ↩
          0
 Seaport Hotel & World Trade Center    |  ↩
          0
 Waterside Place                       |  ↩
          0
 World Trade Center East               |  ↩
          0
 100 Northern Avenue                   |  ↩
          1
 100 Pier 4                            |  ↩
          1
 The Institute of Contemporary Art     |  ↩
          1
 101 Seaport                           |  ↩
          2
 District Hall                         |  ↩
          2
 One Marina Park Drive                 |  ↩
          2
 Twenty Two Liberty                    |  ↩
          2
 Vertex                                |  ↩
          2
 Vertex                                |  ↩
          2
 Watermark Seaport                     |  ↩
          2
 Blue Hills Bank Pavilion              |    ↩
     NULL
 World Trade Center West               |    ↩
     NULL
(20 rows)
```

Un exemple montrant la combinaison de parcelles ayant le même numéro de cluster dans des collections géométriques.

```
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;
```

Voir aussi

ST_DWithin, ST_ClusterKMeans, ST_ClusterIntersecting, ST_ClusterIntersectingWin, ST_ClusterWithin, ST_ClusterWithinWin

7.17.2 ST_ClusterIntersecting

ST_ClusterIntersecting — Fonction d'agrégation qui regroupe les géométries en entrée en ensembles connectés.

Synopsis

geometry[] **ST_ClusterIntersecting**(geometry set g);

Description

Une fonction agrégée qui renvoie un tableau de GeometryCollections partitionnant les géométries d'entrée en clusters connectés et disjoints. Chaque géométrie d'un cluster intersecte au moins une autre géométrie du cluster et n'intersecte aucune géométrie d'autre cluster.

Disponibilité : 2.2.0

Exemples

```
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 ST_AsText(unnest(ST_ClusterIntersecting(geom))) FROM testdata;

--result

st_astext
---------
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))
```

Voir aussi

ST_ClusterIntersectingWin, ST_ClusterWithin, ST_ClusterWithinWin

7.17.3 ST_ClusterIntersectingWin

ST_ClusterIntersectingWin — Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, en regroupant les géométries en entrée en ensembles connectés.

Synopsis

integer **ST_ClusterIntersectingWin**(geometry winset geom);

Description

Une fonction window qui construit des clusters connectés de géométries qui se croisent. Il est possible de parcourir toutes les géométries d'un cluster sans quitter le cluster. La valeur de retour est le numéro du cluster à laquelle participe l'argument géométrie, ou null pour les entrées nulles.

Disponibilité : 3.4.0

Exemples

```
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,
  ST_AsText(geom),
  ST_ClusterIntersectingWin(geom) OVER () AS cluster
FROM testdata;

 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
```

Voir aussi

ST_ClusterIntersecting, ST_ClusterWithin, ST_ClusterWithinWin

7.17.4 ST_ClusterKMeans

ST_ClusterKMeans — Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée en utilisant l'algorithme K-means.

Synopsis

integer **ST_ClusterKMeans**(geometry winset geom, integer number_of_clusters, float max_radius);

Description

Renvoie le numéro de cluster K-means pour chaque géométrie en entrée. La distance utilisée pour le clustering est la distance entre les centroïdes pour les géométries 2D, et la distance entre les centres des boîtes de délimitation pour les géométries 3D. Pour les entrées POINT, la coordonnée M sera traitée comme le poids de l'entrée et doit être supérieure à 0.

`max_radius`, si elle est définie, fera en sorte que ST_ClusterKMeans génère plus de clusters que `k` en s'assurant qu'aucun cluster en sortie n'a un rayon plus grand que `max_radius`. Ceci est utile dans l'analyse d'accessibilité.

Amélioré : 3.2.0 Support pour `max_radius`

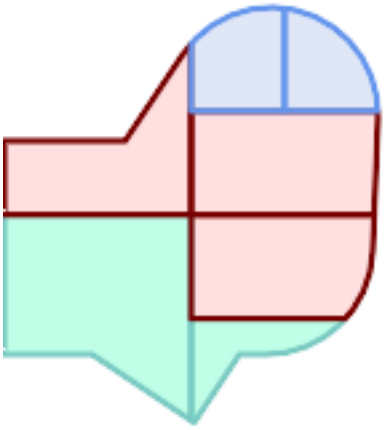
Amélioration : 3.1.0 Prise en charge des géométries et des poids en 3D

Disponibilité : 2.3.0

Exemples

Générer un ensemble de parcelles fictives pour les exemples :

```
CREATE TABLE parcels AS
SELECT lpad((row_number() over())::text,3,'0') As parcel_id, geom,
('{residential, commercial}'::text[])[1 + mod(row_number()OVER(),2)] As type
FROM
    ST_Subdivide(ST_Buffer('SRID=3857;LINESTRING(40 100, 98 100, 100 150, 60 90)'::geometry ↩
        ,
    40, 'endcap=square'),12) As geom;
```



Parcelles codées en couleur par numéro de cluster (cid)

```
SELECT ST_ClusterKMeans(geom, 3) OVER() AS cid, parcel_id, geom
    FROM parcels;
```

```
cid | parcel_id |   geom
-----+-----------+---------------
   0 | 001       | 0103000000...
   0 | 002       | 0103000000...
   1 | 003       | 0103000000...
   0 | 004       | 0103000000...
   1 | 005       | 0103000000...
   2 | 006       | 0103000000...
   2 | 007       | 0103000000...
```

Partitionnement des clusters de parcelles par type :

```
SELECT ST_ClusterKMeans(geom, 3) over (PARTITION BY type) AS cid, parcel_id, type
    FROM parcels;
```

```
cid | parcel_id |    type
-----+-----------+-------------
   1 | 005       | commercial
   1 | 003       | commercial
   2 | 007       | commercial
   0 | 001       | commercial
   1 | 004       | residential
   0 | 002       | residential
   2 | 006       | residential
```

Exemple : Regroupement d'un ensemble de données de population préagrégées à l'échelle planétaire en utilisant le clusering 3D et la pondération. Identifiez au moins 20 régions basées sur Kontur Population Data qui ne s'étendent pas à plus de 3000 km de leur centre :

```
create table kontur_population_3000km_clusters as
select
    geom,
    ST_ClusterKMeans(
        ST_Force4D(
            ST_Transform(ST_Force3D(geom), 4978), -- cluster in 3D XYZ CRS
            mvalue => population -- set clustering to be weighed by population
        ),
        20,                      -- aim to generate at least 20 clusters
        max_radius => 3000000    -- but generate more to make each under 3000 km radius
    ) over () as cid
from
    kontur_population;
```

La population mondiale regroupée selon les spécifications ci-dessus produit 46 clusters. Les clusters sont centrés sur des régions bien peuplées (New York, Moscou). Le Groenland est un cluster. Il y a des groupes d'îles qui s'étendent sur l'antiméridien. Les bords des clusters suivent la courbure de la Terre.

Voir aussi

ST_ClusterDBSCAN, ST_ClusterIntersectingWin, ST_ClusterWithinWin, ST_ClusterIntersecting, ST_ClusterWithin, ST_Subdivide, ST_Force3D, ST_Force4D,

7.17.5 ST_ClusterWithin

ST_ClusterWithin — Fonction agrégée qui regroupe les géométries en fonction de la distance de séparation.

Synopsis

geometry[] **ST_ClusterWithin**(geometry set g, float8 distance);

Description

Une fonction agrégée qui renvoie un tableau de GeometryCollections, où chaque collection est un cluster contenant certaines géométries d'entrée. Le regroupement partitionne les géométries d'entrée en ensembles dans lesquels chaque géométrie se trouve dans la `distance` spécifiée d'au moins une autre géométrie dans le même cluster. Les distances sont des distances cartésiennes dans les unités de l'ISDR.

ST_ClusterWithin est équivalent à l'exécution de ST_ClusterDBSCAN avec `minpoints => 0`.

Disponibilité : 2.2.0

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
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 ST_AsText(unnest(ST_ClusterWithin(geom, 1.4))) FROM testdata;

--result

st_astext
---------
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))
```

Voir aussi

ST_ClusterWithinWin, ST_ClusterDBSCAN, ST_ClusterIntersecting, ST_ClusterIntersectingWin

7.17.6 ST_ClusterWithinWin

ST_ClusterWithinWin — Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, regroupement en utilisant la distance de séparation.

Synopsis

integer **ST_ClusterWithinWin**(geometry winset geom, float8 distance);

Description

Une fonction window qui renvoie un numéro de cluster pour chaque géométrie d'entrée. Le regroupement partitionne les géométries en ensembles dans lesquels chaque géométrie se trouve dans la `distance` spécifiée d'au moins une autre géométrie dans le même groupe. Les distances sont des distances cartésiennes dans les unités du SRID.

ST_ClusterWithinWin est équivalent à l'exécution de ST_ClusterDBSCAN avec `minpoints => 0`.

Disponibilité : 3.4.0

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
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,
  ST_AsText(geom),
```

```
  ST_ClusterWithinWin(geom, 1.4) OVER () AS cluster
FROM testdata;


 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
```

Voir aussi

ST_ClusterWithin, ST_ClusterDBSCAN, ST_ClusterIntersecting, ST_ClusterIntersectingWin,

7.18 Fonctions des boîtes de délimitation

7.18.1 Box2D

Box2D — Renvoie une BOX2D représentant l'étendue 2D d'une géométrie.

Synopsis

box2d **Box2D**(geometry geom);

Description

Renvoie une box2d représentant l'étendue 2D de la géométrie.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT Box2D(ST_GeomFromText('LINESTRING(1 2, 3 4, 5 6)'));

box2d
---------
BOX(1 2,5 6)
```

```
SELECT Box2D(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'));

box2d
--------
BOX(220186.984375 150406,220288.25 150506.140625)
```

Voir aussi

Box3D, ST_GeomFromText

7.18.2 Box3D

Box3D — Renvoie une BOX3D représentant l'étendue 3D d'une géométrie.

Synopsis

box3d **Box3D**(geometry geom);

Description

Renvoie une box3d représentant l'étendue 3D de la géométrie.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT Box3D(ST_GeomFromEWKT('LINESTRING(1 2 3, 3 4 5, 5 6 5)'));

Box3d
---------
BOX3D(1 2 3,5 6 5)
```

```
SELECT Box3D(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 1,220227 150406 ↩
    1)'));

Box3d
--------
BOX3D(220227 150406 1,220268 150415 1)
```

Voir aussi

Box2D, ST_GeomFromEWKT

7.18.3 ST_EstimatedExtent

ST_EstimatedExtent — Renvoie l'étendue estimée d'une table spatiale.

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);

Description

Renvoie l'étendue estimée d'une table spatiale sous la forme d'une box2d. Le schéma actuel est utilisé s'il n'est pas spécifié. L'étendue estimée est tirée des statistiques de la colonne géométrique. Cette méthode est généralement plus rapide que le calcul de l'étendue exacte de la table à l'aide de ST_Extent ou ST_3DExtent.

Le comportement par défaut est d'utiliser également les statistiques collectées à partir des tables enfants (tables avec INHERITS) si elles sont disponibles. Si `parent_only` est défini à VRAI, seules les statistiques de la table donnée sont utilisées et les tables enfants sont ignorées.

Pour PostgreSQL >= 8.0.0 les statistiques sont collectées par VACUUM ANALYZE et l'étendue du résultat sera d'environ 95% de l'étendue réelle. Pour PostgreSQL < 8.0.0 les statistiques sont collectées en exécutant `update_geometry_stats()` et l'étendue du résultat sera exacte.

Note
En l'absence de statistiques (table vide ou pas d'ANALYZE), cette fonction renvoie NULL. Avant la version 1.5.4, une exception était levée.

Note
Escaping names for tables and/or namespaces that include special characters and quotes may require special handling. A user notes: "For schemas and tables, use identifier escaping rules to produce a double-quoted string, and afterwards remove the first and last double-quote character. For geometry column pass as is."

Disponibilité : 1.0.0

Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Estimated_Extent.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_EstimatedExtent('ny', 'edges', 'geom');
--result--
BOX(-8877653 4912316,-8010225.5 5589284)

SELECT ST_EstimatedExtent('feature_poly', 'geom');
--result--
BOX(-124.659652709961 24.6830825805664,-67.7798080444336 49.0012092590332)
```

Voir aussi

ST_Extent, ST_3DExtent

7.18.4 ST_Expand

ST_Expand — Renvoie une boîte de délimitation développée à partir d'une autre boîte de délimitation ou d'une géométrie.

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);

Description

Renvoie une boîte de délimitation développée à partir de la boîte de délimitation de l'entrée, soit en spécifiant une distance unique avec laquelle la boîte doit être développée sur les deux axes, soit en spécifiant une distance de développement pour chaque axe. Utilise la double précision. Peut être utilisé pour les requêtes de distance, ou pour ajouter un filtre de boîte de délimitation à une requête afin de tirer parti d'un index spatial.

Outre la version de ST_Expand acceptant et renvoyant une géométrie, des variantes sont fournies qui acceptent et renvoient des types de données box2d et box3d.

Les distances sont exprimées dans les unités du système de référence spatiale de l'entrée.

ST_Expand est similaire à ST_Buffer, mais alors que la mise en mémoire tampon étend une géométrie dans toutes les directions, ST_Expand étend la boîte de délimitation le long de chaque axe.

Note
Avant la version 1.3, ST_Expand était utilisé en conjonction avec ST_Distance pour effectuer des requêtes de distance indexables. Par exemple, `geom && ST_Expand('POINT(10 20)', 10) AND ST_Distance(geom, 'POINT(10 20)') < 10`. Cette fonction a été remplacée par la fonction ST_DWithin, plus simple et plus efficace.

Disponibilité : 1.5.0 comportement modifié pour afficher les coordonnées en double précision au lieu des coordonnées float4.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Amélioration : 2.3.0 : prise en charge de l'expansion d'une boîte par différentes quantités dans différentes dimensions.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

Note
Les exemples ci-dessous utilisent l'Atlas national américain Equal Area (SRID=2163) qui est une projection en mètres

```
--10 meter expanded box around bbox of a linestring
SELECT CAST(ST_Expand(ST_GeomFromText('LINESTRING(2312980 110676,2312923 110701,2312892  ↩
    110714)', 2163),10) As box2d);
                                        st_expand
------------------------------------
 BOX(2312882 110666,2312990 110724)

--10 meter expanded 3D box of a 3D box
SELECT ST_Expand(CAST('BOX3D(778783 2951741 1,794875 2970042.61545891 10)' As box3d),10)
                                                        st_expand
------------------------------------------------------
 BOX3D(778773 2951731 -9,794885 2970052.61545891 20)

 --10 meter geometry astext rep of a expand box around a point geometry
 SELECT ST_AsEWKT(ST_Expand(ST_GeomFromEWKT('SRID=2163;POINT(2312980 110676)'),10));
                                                                                st_asewkt ↩

------------------------------------------------------------------------------------------ ↩

 SRID=2163;POLYGON((2312970 110666,2312970 110686,2312990 110686,2312990 110666,2312970  ↩
     110666))
```

Voir aussi

ST_Buffer, ST_DWithin, ST_SRID

7.18.5 ST_Extent

ST_Extent — Fonction agrégée qui renvoie la boîte de délimitation des géométries.

Synopsis

box2d **ST_Extent**(geometry set geomfield);

Description

Une fonction agrégée qui renvoie une box2d boîte de délimitation qui délimite un ensemble de géométries.

Les coordonnées de la boîte de délimitation sont dans le système de référence spatiale des géométries d'entrée.

Le concept de ST_Extent est similaire à celui de SDO_AGGR_MBR d'Oracle Spatial/Locator.

Note
ST_Extent renvoie des boîtes avec seulement les coordonnées X et Y, même avec des géométries 3D. Pour renvoyer les coordonnées XYZ, utilisez ST_3DExtent.

Note
La valeur `box3d` renvoyée ne contient pas de SRID. Utilisez ST_SetSRID pour la convertir en une géométrie avec des métadonnées SRID. Le SRID est le même que celui des géométries d'entrée.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

Note
Les exemples ci-dessous utilisent le plan de l'État du Massachusetts (SRID=2249)

```
SELECT ST_Extent(geom) as bextent FROM sometable;
                                          st_bextent
------------------------------------
BOX(739651.875 2908247.25,794875.8125 2970042.75)


--Return extent of each category of geometries
SELECT ST_Extent(geom) as bextent
FROM sometable
```

```
GROUP BY category ORDER BY category;

                                          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

 --Force back into a geometry
 -- and render the extended text representation of that geometry
SELECT ST_SetSRID(ST_Extent(geom),2249) as bextent FROM sometable;

                               bextent
-------------------------------------------------------------------------------
 SRID=2249;POLYGON((739651.875 2908247.25,739651.875 2970042.75,794875.8125 2970042.75,
 794875.8125 2908247.25,739651.875 2908247.25))
```

Voir aussi

ST_EstimatedExtent, ST_3DExtent, ST_SetSRID

7.18.6 ST_3DExtent

ST_3DExtent — Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.

Synopsis

box3d **ST_3DExtent**(geometry set geomfield);

Description

Une fonction d'agrégation qui renvoie une box3d (y compris l'ordonnée Z) qui délimite un ensemble de géométries.

Les coordonnées de la boîte de délimitation sont dans le système de référence spatiale des géométries d'entrée.

Note
La valeur `box3d` renvoyée ne contient pas de SRID. Utilisez ST_SetSRID pour la convertir en une géométrie avec des métadonnées SRID. Le SRID est le même que celui des géométries d'entrée.

Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques.

Modifié : 2.0.0 Dans les versions précédentes, cette fonction était appelée ST_Extent3D

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
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;
          b3extent
--------------------
 BOX3D(1 1 0,3 2 2)

--Get the extent of various elevated circular strings
SELECT ST_3DExtent(foo.geom) As b3extent
FROM (SELECT ST_Translate(ST_Force_3DZ(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;

        b3extent
--------------------
 BOX3D(1 0 0,4 2 2)
```

Voir aussi

ST_Extent, ST_Force3DZ, ST_SetSRID

7.18.7 ST_MakeBox2D

ST_MakeBox2D — Crée un BOX2D défini par deux géométries de points 2D.

Synopsis

box2d **ST_MakeBox2D**(geometry pointLowLeft, geometry pointUpRight);

Description

Crée une box2d définie par deux géométries de points. Ceci est utile pour effectuer des requêtes sur des intervalles.

Exemples

```
--Return all features that fall reside 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)
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)
```

Voir aussi

ST_Point, ST_SetSRID, ST_SRID

7.18.8 ST_3DMakeBox

ST_3DMakeBox — Crée un BOX3D défini par deux géométries de points 3D.

Synopsis

box3d **ST_3DMakeBox**(geometry point3DLowLeftBottom, geometry point3DUpRightTop);

Description

Crée une box3d définie par deux géométries de points 3D.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Modifié : 2.0.0 Dans les versions précédentes, cette fonction s'appelait ST_MakeBox3D

Exemples

```
SELECT ST_3DMakeBox(ST_MakePoint(-989502.1875, 528439.5625, 10),
        ST_MakePoint(-987121.375 ,529933.1875, 10)) As abb3d

--bb3d--
--------
BOX3D(-989502.1875 528439.5625 10,-987121.375 529933.1875 10)
```

Voir aussi

ST_MakePoint, ST_SetSRID, ST_SRID

7.18.9 ST_XMax

ST_XMax — Retourne les maxima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Synopsis

float **ST_XMax**(box3d aGeomorBox2DorBox3D);

Description

Retourne les maxima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

> **Note**
> Bien que cette fonction ne soit définie que pour box3d, elle fonctionne également pour box2d et les valeurs géométriques en raison du moulage automatique. Cependant, elle n'acceptera pas de représentation textuelle de la géométrie ou de la box2d, car ces dernières ne sont pas soumises à la conversion automatique.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_XMax('BOX3D(1 2 3, 4 5 6)');
st_xmax
-------
4

SELECT ST_XMax(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_xmax
-------
5

SELECT ST_XMax(CAST('BOX(-3 2, 3 4)' As box2d));
st_xmax
-------
3
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to ↩
    a BOX3D
SELECT ST_XMax('LINESTRING(1 3, 5 6)');

--ERROR:  BOX3D parser - doesn't start with BOX3D(

SELECT ST_XMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ↩
    150406 3)'));
st_xmax
--------
220288.248780547
```

Voir aussi

ST_XMin, ST_YMax, ST_YMin, ST_ZMax, ST_ZMin

7.18.10 ST_XMin

ST_XMin — Retourne les minima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Synopsis

float **ST_XMin**(box3d aGeomorBox2DorBox3D);

Description

Retourne les minima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Note
Bien que cette fonction ne soit définie que pour box3d, elle fonctionne également pour box2d et les valeurs géométriques en raison du moulage automatique. Cependant, elle n'acceptera pas de représentation textuelle de la géométrie ou de la box2d, car celles-ci ne sont pas coulées automatiquement.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_XMin('BOX3D(1 2 3, 4 5 6)');
st_xmin
-------
1

SELECT ST_XMin(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_xmin
-------
1

SELECT ST_XMin(CAST('BOX(-3 2, 3 4)' As box2d));
st_xmin
-------
-3
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to ↩
    a BOX3D
SELECT ST_XMin('LINESTRING(1 3, 5 6)');

--ERROR:  BOX3D parser - doesn't start with BOX3D(

SELECT ST_XMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ↩
    150406 3)'));
st_xmin
--------
220186.995121892
```

Voir aussi

ST_XMax, ST_YMax, ST_YMin, ST_ZMax, ST_ZMin

7.18.11 ST_YMax

ST_YMax — Retourne les maxima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Synopsis

float **ST_YMax**(box3d aGeomorBox2DorBox3D);

Description

Retourne les maxima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Note
Bien que cette fonction ne soit définie que pour box3d, elle fonctionne également pour box2d et les valeurs géométriques en raison du moulage automatique. Cependant, elle n'acceptera pas de représentation textuelle de la géométrie ou de la box2d, car celles-ci ne sont pas coulées automatiquement.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_YMax('BOX3D(1 2 3, 4 5 6)');
st_ymax
-------
5

SELECT ST_YMax(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_ymax
-------
6

SELECT ST_YMax(CAST('BOX(-3 2, 3 4)' As box2d));
st_ymax
-------
4
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to ↩
    a BOX3D
SELECT ST_YMax('LINESTRING(1 3, 5 6)');

--ERROR:  BOX3D parser - doesn't start with BOX3D(

SELECT ST_YMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227  ↩
    150406 3)'));
st_ymax
--------
150506.126829327
```

Voir aussi

ST_XMin, ST_XMax, ST_YMin, ST_ZMax, ST_ZMin

7.18.12 ST_YMin

ST_YMin — Retourne les minima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Synopsis

float **ST_YMin**(box3d aGeomorBox2DorBox3D);

Description

Retourne les minima Y d'une boîte englobante 2D ou 3D ou d'une géométrie.

Note
Bien que cette fonction ne soit définie que pour box3d, elle fonctionne également pour box2d et les valeurs géométriques en raison du moulage automatique. Cependant, elle n'acceptera pas de représentation textuelle de la géométrie ou de la box2d, car celles-ci ne sont pas coulées automatiquement.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_YMin('BOX3D(1 2 3, 4 5 6)');
st_ymin
-------
2

SELECT ST_YMin(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_ymin
-------
3

SELECT ST_YMin(CAST('BOX(-3 2, 3 4)' As box2d));
st_ymin
-------
2
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to ↩
    a BOX3D
SELECT ST_YMin('LINESTRING(1 3, 5 6)');

--ERROR:  BOX3D parser - doesn't start with BOX3D(

SELECT ST_YMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227  ↩
    150406 3)'));
st_ymin
--------
150406
```

Voir aussi

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMax, ST_ZMax, ST_ZMin

7.18.13 ST_ZMax

ST_ZMax — Retourne les maxima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Synopsis

float **ST_ZMax**(box3d aGeomorBox2DorBox3D);

Description

Retourne les maxima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Note
Bien que cette fonction ne soit définie que pour box3d, elle fonctionne également pour box2d et les valeurs géométriques en raison du moulage automatique. Cependant, elle n'acceptera pas de représentation textuelle de la géométrie ou de la box2d, car celles-ci ne sont pas coulées automatiquement.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_ZMax('BOX3D(1 2 3, 4 5 6)');
st_zmax
-------
6

SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING(1 3 4, 5 6 7)'));
st_zmax
-------
7

SELECT ST_ZMax('BOX3D(-3 2 1, 3 4 1)' );
st_zmax
-------
1
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to ↩
    a BOX3D
SELECT ST_ZMax('LINESTRING(1 3 4, 5 6 7)');

--ERROR:  BOX3D parser - doesn't start with BOX3D(

SELECT ST_ZMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227  ↩
    150406 3)'));
st_zmax
--------
3
```

Voir aussi

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMax, ST_YMin, ST_ZMax

7.18.14 ST_ZMin

ST_ZMin — Retourne les minima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Synopsis

float **ST_ZMin**(box3d aGeomorBox2DorBox3D);

Description

Retourne les minima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

Note
Bien que cette fonction ne soit définie que pour box3d, elle fonctionne également pour box2d et les valeurs géométriques en raison du moulage automatique. Cependant, elle n'acceptera pas de représentation textuelle de la géométrie ou de la box2d, car celles-ci ne sont pas coulées automatiquement.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT ST_ZMin('BOX3D(1 2 3, 4 5 6)');
st_zmin
-------
3

SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING(1 3 4, 5 6 7)'));
st_zmin
-------
4

SELECT ST_ZMin('BOX3D(-3 2 1, 3 4 1)' );
st_zmin
-------
1
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to ↩
    a BOX3D
SELECT ST_ZMin('LINESTRING(1 3 4, 5 6 7)');

--ERROR:  BOX3D parser - doesn't start with BOX3D(

SELECT ST_ZMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227  ↩
    150406 3)'));
st_zmin
--------
1
```

Voir aussi

ST_GeomFromEWKT, ST_GeomFromText, ST_XMin, ST_XMax, ST_YMax, ST_YMin, ST_ZMax

7.19 Référencement linéaire

7.19.1 ST_LineInterpolatePoint

ST_LineInterpolatePoint — Renvoie un point interpolé le long d'une ligne à un emplacement fractionnaire.

Synopsis

geometry **ST_LineInterpolatePoint**(geometry a_linestring, float8 a_fraction);
geography **ST_LineInterpolatePoint**(geography a_linestring, float8 a_fraction, boolean use_spheroid = true);

Description

Retourne un point interpolé sur une ligne. Le premier argument doit être une LINESTRING. Le second argument est un float8 entre 0 et 1 représentant la fraction de la longueur totale de la ligne où le point doit être situé.

Voir ST_LineLocatePoint pour calculer l'emplacement de la ligne la plus proche d'un point.

Note
Cette fonction calcule des points en 2D et interpole ensuite les valeurs de Z et M, tandis que ST_3DLineInterpolatePoint calcule des points en 3D et n'interpole que la valeur M.

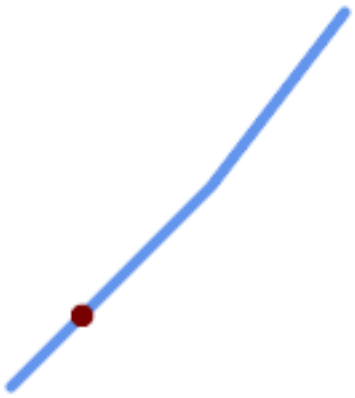
Note
Depuis la version 1.1.1 cette fonction interpole aussi les valeurs M et Z (lorsqu'elles sont présentes), les versions précédentes renvoient des valeurs M et Z à 0.0.

Disponibilité : 0.8.2, support de Z et M ajouté en 1.1.1

Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Line_Interpolate_Point.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples



Une ligne avec le point interpolé à la position 20% (0.20)

```
-- The point 20% along a line

SELECT ST_AsEWKT(  ST_LineInterpolatePoint(
        'LINESTRING(25 50, 100 125, 150 190)',
        0.2 ));
----------------
 POINT(51.5974135047432 76.5974135047432)
```

Le point central d'une ligne 3D :

```
SELECT ST_AsEWKT(  ST_LineInterpolatePoint('
        LINESTRING(1 2 3, 4 5 6, 6 7 8)',
        0.5 ));
--------------------
 POINT(3.5 4.5 5.5)
```

Le point le plus proche d'un point sur une ligne :

```
SELECT ST_AsText( ST_LineInterpolatePoint( line.geom,
                    ST_LineLocatePoint( line.geom, 'POINT(4 3)')))
FROM (SELECT ST_GeomFromText('LINESTRING(1 2, 4 5, 6 7)') As geom) AS line;
------------
 POINT(3 4)
```

Voir aussi

ST_LineInterpolatePoints, ST_3DLineInterpolatePoint, ST_LineLocatePoint

7.19.2 ST_3DLineInterpolatePoint

ST_3DLineInterpolatePoint — Renvoie un point interpolé le long d'une ligne 3D à un emplacement fractionnaire.

Synopsis

geometry **ST_3DLineInterpolatePoint**(geometry a_linestring, float8 a_fraction);

Description

Retourne un point interpolé sur une ligne. Le premier argument doit être une LINESTRING. Le second argument est un float8 entre 0 et 1 représentant la fraction de la longueur totale de la ligne où le point doit être situé.

Note
ST_LineInterpolatePoint calcule des points en 2D et interpole ensuite les valeurs de Z et M, alors que cette fonction calcule des points en 3D et n'interpole que la valeur M.

Disponibilité : 3.0.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Point de retour 20% le long de la ligne 3D

```
SELECT ST_AsText(
    ST_3DLineInterpolatePoint('LINESTRING(25 50 70, 100 125 90, 150 190 200)',
        0.20));

   st_asetext
----------------
 POINT Z (59.0675892910822 84.0675892910822 79.0846904776219)
```

Voir aussi

ST_LineInterpolatePoint, ST_LineInterpolatePoints, ST_LineLocatePoint

7.19.3 ST_LineInterpolatePoints

ST_LineInterpolatePoints — Renvoie des points interpolés le long d'une ligne à un intervalle fractionnaire.

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);

Description

Renvoie un ou plusieurs points interpolés le long d'une ligne à un intervalle fractionnaire. Le premier argument doit être une LINESTRING. Le second argument est un float8 entre 0 et 1 représentant l'espacement entre les points en tant que fraction de la longueur de la ligne. Si le troisième argument est faux, un seul point sera construit (ce qui est équivalent à ST_LineInterpolatePoint.)

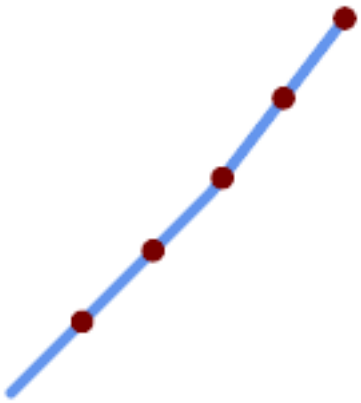
Si le résultat a zéro ou un point, il est retourné sous forme de POINT. S'il a deux points ou plus, il est retourné sous la forme d'un MULTIPOINT.

Disponibilité : 2.5.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les coordonnées M.

Exemples



Une ligne avec des points interpolés tous les 20 %

```
--Return points each 20% along a 2D line
SELECT ST_AsText(ST_LineInterpolatePoints('LINESTRING(25 50, 100 125, 150 190)', 0.20))
----------------
 MULTIPOINT((51.5974135047432 76.5974135047432),(78.1948270094864 103.194827009486) ↩
     ,(104.132163186446 130.37181214238),(127.066081593223 160.18590607119),(150 190))
```

Voir aussi

ST_LineInterpolatePoint, ST_LineLocatePoint

7.19.4 ST_LineLocatePoint

ST_LineLocatePoint — Renvoie l'emplacement fractionnaire du point le plus proche d'un point sur une ligne.

Synopsis

float8 **ST_LineLocatePoint**(geometry a_linestring, geometry a_point);
float8 **ST_LineLocatePoint**(geography a_linestring, geography a_point, boolean use_spheroid = true);

Description

Renvoie un flottant compris entre 0 et 1 représentant l'emplacement du point le plus proche du point donné sur une LineString, en tant que fraction de la ligne 2D longueur.

Vous pouvez utiliser l'emplacement renvoyé pour extraire un point (ST_LineInterpolatePoint) ou une sous-ligne (ST_LineSubstring).

Cette méthode est utile pour calculer approximativement le nombre d'adresses

Disponibilité : 1.1.0

Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Line_Locate_Point.

Exemples

```
--Rough approximation of finding the street number of a point along the street
--Note the whole foo thing is just to generate dummy data that looks
--like house centroids and street
--We use ST_DWithin to exclude
--houses too far away from the street to be considered on the street
SELECT ST_AsText(house_loc) As as_text_house_loc,
        startstreet_num +
                CAST( (endstreet_num - startstreet_num)
                        * ST_LineLocatePoint(street_line, house_loc) As integer) As  ↩
                            street_num
FROM
(SELECT ST_GeomFromText('LINESTRING(1 2, 3 4)') As street_line,
        ST_Point(x*1.01,y*1.03) As house_loc, 10 As startstreet_num,
                20 As endstreet_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);

 as_text_house_loc | street_num
-------------------+------------
 POINT(1.01 2.06)  |         10
 POINT(2.02 3.09)  |         15
 POINT(3.03 4.12)  |         20

 --find closest point on a line to a point or other geometry
 SELECT ST_AsText(ST_LineInterpolatePoint(foo.the_line, ST_LineLocatePoint(foo.the_line,  ↩
     ST_GeomFromText('POINT(4 3)'))))
FROM (SELECT ST_GeomFromText('LINESTRING(1 2, 4 5, 6 7)') As the_line) As foo;
   st_astext
----------------
 POINT(3 4)
```

Voir aussi

ST_DWithin, ST_Length2D, ST_LineInterpolatePoint, ST_LineSubstring

7.19.5 ST_LineSubstring

ST_LineSubstring — Renvoie la partie d'une ligne située entre deux emplacements fractionnaires.

Synopsis

geometry **ST_LineSubstring**(geometry a_linestring, float8 startfraction, float8 endfraction);
geography **ST_LineSubstring**(geography a_linestring, float8 startfraction, float8 endfraction);

Description

Calcule la ligne qui est la section de la ligne d'entrée commençant et se terminant aux emplacements fractionnaires donnés. Le premier argument doit être un LINESTRING. Les deuxième et troisième arguments sont des valeurs dans l'intervalle [0, 1] représentant les emplacements de début et de fin en tant que fractions de la longueur de la ligne. Les valeurs Z et M sont interpolées pour les points d'extrémité ajoutés, le cas échéant.

Si `startfraction` et `endfraction` ont la même valeur, cela équivaut à ST_LineInterpolatePoint.

Note
Cette méthode ne fonctionne qu'avec les LINESTRING. Pour l'utiliser sur des MULTILINESTRINGs contigues, il faut d'abord les joindre avec ST_LineMerge.

Note
Depuis la version 1.1.1, cette fonction interpole les valeurs de M et Z. Les versions précédentes fixent Z et M à des valeurs non spécifiées.

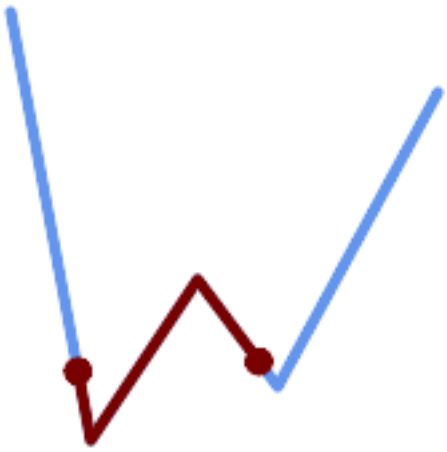
Amélioration : 3.4.0 - La prise en charge de la géographie a été introduite.

Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Line_Substring.

Disponibilité : 1.1.0, prise en charge des Z et M ajoutée dans la 1.1.1

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples



Une LineString vue avec un intervalle tous les 1/3 (0.333, 0.666)

```
SELECT ST_AsText(ST_LineSubstring( 'LINESTRING (20 180, 50 20, 90 80, 120 40, 180 150)',  ↩
    0.333, 0.666));
------------------------------------------------------------------------------------------ ↩

LINESTRING (45.17311810399485 45.74337011202746, 50 20, 90 80, 112.97593050157862  ↩
    49.36542599789519)
```

Si les points de départ et d'arrivée sont identiques, le résultat est un POINT.

```
SELECT ST_AsText(ST_LineSubstring( 'LINESTRING(25 50, 100 125, 150 190)', 0.333, 0.333));
----------------------------------------
 POINT(69.2846934853974 94.2846934853974)
```

Une requête pour découper une LineString en sections de longueur 100 ou moins. Elle utilise `generate_series()` avec un CROSS JOIN LATERAL pour produire l'équivalent d'une boucle FOR.

```
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_AsText( 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)
    -- skip last i if line length is exact multiple of sublen
    WHERE (sublen * i) / len <
> 1.0
    ) AS d2;

 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)
```

Mesures de mise en œuvre de la geography le long d'un sphéroïde, geometry le long d'une ligne

```
SELECT ST_AsText(ST_LineSubstring( 'LINESTRING(-118.2436 34.0522, -71.0570 42.3611)':: ↩
    geography, 0.333, 0.666),6) AS geog_sub
 , ST_AsText(ST_LineSubstring('LINESTRING(-118.2436 34.0522, -71.0570 42.3611)'::geometry,  ↩
     0.333, 0.666),6) AS geom_sub;
-------------------------------------------------------------
geog_sub | LINESTRING(-104.167064 38.854691,-87.674646 41.849854)
geom_sub | LINESTRING(-102.530462 36.819064,-86.817324 39.585927)
```

Voir aussi

ST_Length, ST_LineExtend, ST_LineInterpolatePoint, ST_LineMerge

7.19.6 ST_LocateAlong

ST_LocateAlong — Renvoie le(s) point(s) d'une géométrie qui correspond(ent) à une valeur de mesure.

Synopsis

geometry **ST_LocateAlong**(geometry geom_with_measure, float8 measure, float8 offset = 0);

Description

Renvoie le(s) emplacement(s) le long d'une géométrie mesurée qui possède(nt) les valeurs de mesure données. Le résultat est un point ou un multipoint. Les entrées polygonales ne sont pas prises en charge.

Si `offset` est fourni, le résultat est décalé à gauche ou à droite de la ligne d'entrée de la distance spécifiée. Un décalage positif se fera vers la gauche, et un décalage négatif vers la droite.

Note
Cette fonction n'est utilisée que pour les géométries linéaires avec une composante M

La sémantique est spécifiée par la norme *ISO/IEC 13249-3 SQL/MM Spatial*.

Disponibilité : 1.1.0 sous l'ancien nom ST_Locate_Along_Measure.

Modifié : 2.0.0 dans les versions précédentes, cette fonction était appelée ST_Locate_Along_Measure.

Cette fonction prend en charge les coordonnées M.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1.13

Exemples

```
SELECT ST_AsText(
  ST_LocateAlong(
    'MULTILINESTRINGM((1 2 3, 3 4 2, 9 4 3),(1 2 3, 5 4 5))'::geometry,
    3 ));

---------------------------------
 MULTIPOINT M ((1 2 3),(9 4 3),(1 2 3))
```

Voir aussi

ST_LocateBetween, ST_LocateBetweenElevations, ST_InterpolatePoint

7.19.7 ST_LocateBetween

ST_LocateBetween — Renvoie les parties d'une géométrie qui correspondent à un intervalle de mesure.

Synopsis

geometry **ST_LocateBetween**(geometry geom, float8 measure_start, float8 measure_end, float8 offset = 0);

Description

Renvoie une géométrie (collection) contenant les parties de la géométrie mesurée en entrée qui correspondent à l'intervalle de mesure spécifié (inclusivement).

Si le `offset` est fourni, le résultat est décalé à gauche ou à droite de la ligne d'entrée de la distance spécifiée. Un décalage positif se fera vers la gauche, et un décalage négatif vers la droite.

Couper un POLYGONE non convexe peut produire une géométrie non valide.

La sémantique est spécifiée par la norme *ISO/IEC 13249-3 SQL/MM Spatial*.

Disponibilité : 1.1.0 sous l'ancien nom ST_Locate_Between_Measures.

Modifié : 2.0.0 - dans les versions précédentes, cette fonction s'appelait ST_Locate_Between_Measures.

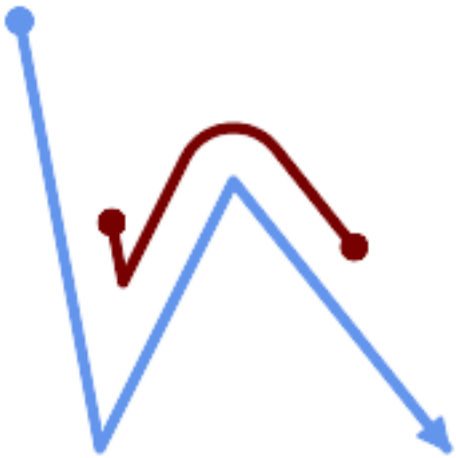
Amélioration : 3.0.0 - ajout de la prise en charge du POLYGONE, du TIN et du TRIANGLE.

Cette fonction prend en charge les coordonnées M.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Exemples

```
SELECT ST_AsText(
  ST_LocateBetween(
       'MULTILINESTRING M ((1 2 3, 3 4 2, 9 4 3),(1 2 3, 5 4 5))':: geometry,
       1.5, 3 ));
------------------------------------------------------------------------
 GEOMETRYCOLLECTION M (LINESTRING M (1 2 3,3 4 2,9 4 3),POINT M (1 2 3))
```



Une LineString avec la section entre les mesures 2 et 8, décalée vers la gauche

```
SELECT ST_AsText( ST_LocateBetween(
  ST_AddMeasure('LINESTRING (20 180, 50 20, 100 120, 180 20)', 0, 10),
  2, 8,
  20
));
------------------------------------------------------------------------
```

```
MULTILINESTRING((54.49835019899045 104.53426957938231,58.70056060327303  ↩
    82.12248075654186,69.16695286779743 103.05526528559065,82.11145618000168  ↩
    128.94427190999915,84.24893681714357 132.32493442618113,87.01636951231555  ↩
    135.21267035596549,90.30307285299679 137.49198684843182,93.97759758337769  ↩
    139.07172433557758,97.89298381958797 139.8887023914453,101.89263860095893  ↩
    139.9102465862721,105.81659870902816 139.13549527600819,109.50792827749828  ↩
    137.5954340631298,112.81899532549731 135.351656550512,115.6173761888606  ↩
    132.49390095108848,145.31017306064817 95.37790486135405))
```

Voir aussi

ST_LocateAlong, ST_LocateBetweenElevations

7.19.8 ST_LocateBetweenElevations

ST_LocateBetweenElevations — Renvoie les parties d'une géométrie qui se trouvent dans un intervalle d'élévation (Z).

Synopsis

geometry **ST_LocateBetweenElevations**(geometry geom, float8 elevation_start, float8 elevation_end);

Description

Renvoie une géométrie (collection) contenant les parties d'une géométrie qui se trouvent dans une plage d'élévation (Z).

Couper un POLYGONE non convexe peut produire une géométrie non valide.

Disponibilité : 1.4.0

Amélioration : 3.0.0 - ajout de la prise en charge du POLYGONE, du TIN et du TRIANGLE.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsText(
  ST_LocateBetweenElevations(
    'LINESTRING(1 2 3, 4 5 6)'::geometry,
    2, 4 ));

             st_astext
-----------------------------------
 MULTILINESTRING Z ((1 2 3,2 3 4))

SELECT ST_AsText(
    ST_LocateBetweenElevations(
      'LINESTRING(1 2 6, 4 5 -1, 7 8 9)',
      6, 9)) As ewelev;

                                ewelev
----------------------------------------------------------------------
 GEOMETRYCOLLECTION Z (POINT Z (1 2 6),LINESTRING Z (6.1 7.1 6,7 8 9))
```

Voir aussi

ST_Dump, ST_LocateBetween

7.19.9 ST_InterpolatePoint

ST_InterpolatePoint — Renvoie la mesure interpolée d'une géométrie la plus proche d'un point.

Synopsis

float8 **ST_InterpolatePoint**(geometry linear_geom_with_measure, geometry point);

Description

Renvoie une valeur de mesure interpolée d'une géométrie linéaire mesurée à l'emplacement le plus proche du point donné.

Note
Cette fonction n'est utilisée que pour les géométries linéaires avec une composante M

Disponibilité : 2.0.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_InterpolatePoint('LINESTRING M (0 0 0, 10 0 20)', 'POINT(5 5)');
 ---------------------
         10
```

Voir aussi

ST_AddMeasure, ST_LocateAlong, ST_LocateBetween

7.19.10 ST_AddMeasure

ST_AddMeasure — Interpole les mesures le long d'une géométrie linéaire.

Synopsis

geometry **ST_AddMeasure**(geometry geom_mline, float8 measure_start, float8 measure_end);

Description

Renvoie une géométrie dérivée avec des valeurs de mesure interpolées linéairement entre les points de départ et d'arrivée. Si la géométrie n'a pas de dimension de mesure, une dimension est ajoutée. Si la géométrie a une dimension de mesure, elle est remplacée par de nouvelles valeurs. Seuls les LINESTRINGS et MULTILINESTRINGS sont pris en charge.

Disponibilité : 1.5.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRING(1 0, 2 0, 4 0)'),1,4)) As ewelev;
           ewelev
-------------------------------
 LINESTRINGM(1 0 1,2 0 2,4 0 4)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRING(1 0 4, 2 0 4, 4 0 4)'),10,40)) As ewelev;
                ewelev
---------------------------------------
 LINESTRING(1 0 4 10,2 0 4 20,4 0 4 40)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRINGM(1 0 4, 2 0 4, 4 0 4)'),10,40)) As ewelev;
                ewelev
---------------------------------------
 LINESTRINGM(1 0 10,2 0 20,4 0 40)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('MULTILINESTRINGM((1 0 4, 2 0 4, 4 0 4),(1 0 4, 2 0 4, 4 0 4))'),10,70)) As ↩
     ewelev;
                             ewelev
-------------------------------------------------------------
 MULTILINESTRINGM((1 0 10,2 0 20,4 0 40),(1 0 40,2 0 50,4 0 70))
```

7.20 Fonctions de trajectoire

7.20.1 ST_IsValidTrajectory

ST_IsValidTrajectory — Teste si la géométrie est une trajectoire valide.

Synopsis

boolean **ST_IsValidTrajectory**(geometry line);

Description

Teste si une géométrie encode une trajectoire valide. Une trajectoire valide est représentée comme une `LINESTRING` avec des mesures (valeurs M). Les valeurs de mesure doivent augmenter de chaque sommet au suivant.

Les trajectoires valides sont attendues comme entrées pour des fonctions spatio-temporelles telles que ST_ClosestPointOfApproach

Disponibilité : 2.2.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
-- A valid trajectory
SELECT ST_IsValidTrajectory(ST_MakeLine(
  ST_MakePointM(0,0,1),
  ST_MakePointM(0,1,2))
);
```

```
 t

-- An invalid trajectory
SELECT ST_IsValidTrajectory(ST_MakeLine(ST_MakePointM(0,0,1), ST_MakePointM(0,1,0)));
NOTICE:  Measure of vertex 1 (0) not bigger than measure of vertex 0 (1)
 st_isvalidtrajectory
----------------------
 f
```

Voir aussi

ST_ClosestPointOfApproach

7.20.2 ST_ClosestPointOfApproach

ST_ClosestPointOfApproach — Renvoie une mesure au point d'approche le plus proche de deux trajectoires.

Synopsis

float8 **ST_ClosestPointOfApproach**(geometry track1, geometry track2);

Description

Renvoie la plus petite mesure pour laquelle les points interpolés le long des trajectoires données sont les moins éloignés les uns des autres.

Les entrées doivent être des trajectoires valides comme vérifié par ST_IsValidTrajectory. Null est retourné si les trajectoires ne se chevauchent pas dans leurs plages M.

Pour obtenir les points réels à la mesure calculée, utilisez ST_LocateAlong .

Disponibilité : 2.2.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
-- 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
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0, 10 0 5)'::geometry,
    extract(epoch from '2015-05-26 10:00'::timestamptz),
    extract(epoch from '2015-05-26 11:00'::timestamptz)
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10, 12 1 2)'::geometry,
    extract(epoch from '2015-05-26 10:00'::timestamptz),
    extract(epoch from '2015-05-26 11:00'::timestamptz)
  ) 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) t,
```

```
       ST_3DDistance(pa,pb) distance,
       ST_AsText(pa, 2) AS pa, ST_AsText(pb, 2) AS pb
FROM points, cpa;

               t               |      distance      |                  pa                  ↩
                   |                  pb
-------------------------------+--------------------+--------------------------------------+--------

 2015-05-26 10:45:31.034483-07 | 1.9652147377620688 | POINT ZM (7.59 0 3.79 1432662331.03)  ↩
    | POINT ZM (9.1 1.24 3.93 1432662331.03)
```

Voir aussi

ST_IsValidTrajectory, ST_DistanceCPA, ST_LocateAlong, ST_AddMeasure

7.20.3 ST_DistanceCPA

ST_DistanceCPA — Renvoie la distance entre le point d'approche le plus proche de deux trajectoires.

Synopsis

float8 **ST_DistanceCPA**(geometry track1, geometry track2);

Description

Renvoie la distance (en 2D) entre deux trajectoires à leur point d'approche le plus proche.

Les entrées doivent être des trajectoires valides comme vérifié par ST_IsValidTrajectory. Null est retourné si les trajectoires ne se chevauchent pas dans leurs plages M.

Disponibilité : 2.2.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
-- Return the minimum distance of two objects moving between 10:00 and 11:00
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0, 10 0 5)'::geometry,
    extract(epoch from '2015-05-26 10:00'::timestamptz),
    extract(epoch from '2015-05-26 11:00'::timestamptz)
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10, 12 1 2)'::geometry,
    extract(epoch from '2015-05-26 10:00'::timestamptz),
    extract(epoch from '2015-05-26 11:00'::timestamptz)
  ) b
)
SELECT ST_DistanceCPA(a,b) distance FROM inp;

     distance
------------------
 1.965214737762069
```

Voir aussi

ST_IsValidTrajectory, ST_ClosestPointOfApproach, ST_AddMeasure, |=|

7.20.4 ST_CPAWithin

ST_CPAWithin — Teste si le point d'approche le plus proche de deux trajectoires se trouve dans la distance spécifiée.

Synopsis

boolean **ST_CPAWithin**(geometry track1, geometry track2, float8 dist);

Description

Teste si deux objets en mouvement ont déjà été plus proches que la distance spécifiée.

Les entrées doivent être des trajectoires valides comme vérifié par ST_IsValidTrajectory. False est retourné si les trajectoires ne se chevauchent pas dans leurs plages M.

Disponibilité : 2.2.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0, 10 0 5)'::geometry,
    extract(epoch from '2015-05-26 10:00'::timestamptz),
    extract(epoch from '2015-05-26 11:00'::timestamptz)
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10, 12 1 2)'::geometry,
    extract(epoch from '2015-05-26 10:00'::timestamptz),
    extract(epoch from '2015-05-26 11:00'::timestamptz)
  ) b
)
SELECT ST_CPAWithin(a,b,2), ST_DistanceCPA(a,b) distance FROM inp;

 st_cpawithin |     distance
--------------+------------------
 t            | 1.96521473776207
```

Voir aussi

ST_IsValidTrajectory, ST_ClosestPointOfApproach, ST_DistanceCPA, |=|

7.21 Fonctions de version

7.21.1 PostGIS_Extensions_Upgrade

PostGIS_Extensions_Upgrade — Packages et mises à jour des extensions PostGIS (par exemple postgis_raster, postgis_topology, postgis_sfcgal) vers la version donnée ou la plus récente.

Synopsis

text **PostGIS_Extensions_Upgrade**(text target_version=null);

Description

Packages et mises à jour des extensions PostGIS vers la version donnée ou la plus récente. Seules les extensions que vous avez installées dans la base de données seront packagées et mises à jour si nécessaire. Rapporte la version complète de PostGIS et les informations sur la configuration de la compilation. C'est un raccourci pour faire plusieurs CREATE EXTENSION ... FROM non packagée et ALTER EXTENSION ... UPDATE pour chaque extension PostGIS. Actuellement, il n'essaie de mettre à jour que les extensions postgis, postgis_raster, postgis_sfcgal, postgis_topology, et postgis_tiger_geocoder.

Disponibilité : 2.5.0

Note
Modifié : 3.4.0 pour ajouter l'argument target_version.
Modifié : 3.3.0 support pour les mises à jour à partir de n'importe quelle version de PostGIS. Ne fonctionne pas sur tous les systèmes.
Modifié : 3.0.0 pour repackager les extensions libres et supporter postgis_raster.

Exemples

```
SELECT PostGIS_Extensions_Upgrade();
```

```
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
NOTICE:  Extension postgis_tiger_geocoder is not available or not packagable for some  ↩
    reason

                    postgis_extensions_upgrade
------------------------------------------------------------------
 Upgrade completed, run SELECT postgis_full_version(); for details
(1 row)
```

Voir aussi

Section 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 — Donne des informations complètes sur la version de PostGIS et la configuration du packaging.

Synopsis

text **PostGIS_Full_Version**();

Description

Donne des informations complètes sur la version de PostGIS et la configuration du packaging. Il fournit également des informations sur la synchronisation entre les bibliothèques et les scripts, suggérant des mises à niveau si nécessaire.

Amélioration : 3.4.0 inclut désormais les configurations supplémentaires PROJ NETWORK_ENABLED, URL_ENDPOINT et DATABASE_PATH pour l'emplacement proj.db

Exemples

```
SELECT PostGIS_Full_Version();
                                                          postgis_full_version
-------------------------------------------------------------------------------
POSTGIS="3.4.0dev 3.3.0rc2-993-g61bdf43a7" [EXTENSION] PGSQL="160" GEOS="3.12.0dev-CAPI ↩
    -1.18.0" SFCGAL="1.3.8" PROJ="7.2.1 NETWORK_ENABLED=OFF URL_ENDPOINT=https://cdn.proj. ↩
    org USER_WRITABLE_DIRECTORY=/tmp/proj DATABASE_PATH=/usr/share/proj/proj.db" GDAL="GDAL  ↩
    3.2.2, released 2021/03/05" LIBXML="2.9.10" LIBJSON="0.15" LIBPROTOBUF="1.3.3" WAGYU ↩
    ="0.5.0 (Internal)" TOPOLOGY RASTER
(1 row)
```

Voir aussi

Section 3.4, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Wagyu_Ve
PostGIS_Version

7.21.3 PostGIS_GEOS_Version

PostGIS_GEOS_Version — Renvoie le numéro de version de la librairie GEOS.

Synopsis

text **PostGIS_GEOS_Version**();

Description

Renvoie le numéro de version de la librairie GEOS, ou `NULL` si le support GEOS n'est pas activé.

Exemples

```
SELECT PostGIS_GEOS_Version();
 postgis_geos_version
----------------------
3.12.0dev-CAPI-1.18.0
(1 row)
```

Voir aussi

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 — Renvoie le numéro de version de la librairie GEOS avec laquelle PostGIS a été construit.

Synopsis

text **PostGIS_GEOS_Compiled_Version**();

Description

Renvoie le numéro de version de la librairie GEOS, ou par rapport à laquelle PostGIS a été construit.

Disponibilité : 3.4.0

Exemples

```
SELECT PostGIS_GEOS_Compiled_Version();
 postgis_geos_compiled_version
-------------------------------
 3.12.0
(1 row)
```

Voir aussi

PostGIS_GEOS_Version, PostGIS_Full_Version

7.21.5 PostGIS_Liblwgeom_Version

PostGIS_Liblwgeom_Version — Renvoie le numéro de version de la librairie liblwgeom. Cela devrait correspondre à la version de PostGIS.

Synopsis

text **PostGIS_Liblwgeom_Version**();

Description

Renvoie le numéro de version de la librairie liblwgeom

Exemples

```
SELECT PostGIS_Liblwgeom_Version();
postgis_liblwgeom_version
--------------------------
3.4.0dev 3.3.0rc2-993-g61bdf43a7
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Version

7.21.6 PostGIS_LibXML_Version

PostGIS_LibXML_Version — Renvoie le numéro de version de la librairie libxml2.

Synopsis

text **PostGIS_LibXML_Version**();

Description

Renvoie le numéro de version de la librairie LibXML2.

Disponibilité : 1.5

Exemples

```
SELECT PostGIS_LibXML_Version();
 postgis_libxml_version
----------------------
 2.9.10
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_PROJ_Version, PostGIS_GEOS_Version, PostGIS_Version

7.21.7 PostGIS_LibJSON_Version

PostGIS_LibJSON_Version — Returns the version number of the libjson-c library.

Synopsis

text **PostGIS_LibJSON_Version**();

Description

Returns the version number of the LibJSON-C library.

Exemples

```
SELECT PostGIS_LibJSON_Version();
 postgis_libjson_version
-------------------------
 0.17
```

Voir aussi

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_PROJ_Version, PostGIS_GEOS_Version, PostGIS_Version

7.21.8 PostGIS_Lib_Build_Date

PostGIS_Lib_Build_Date — Retourne la date de packaging de l'extension PostGIS.

Synopsis

text **PostGIS_Lib_Build_Date**();

Description

Retourne la date de packaging de l'extension PostGIS.

Exemples

```
SELECT PostGIS_Lib_Build_Date();
 postgis_lib_build_date
------------------------
 2023-06-22 03:56:11
(1 row)
```

7.21.9 PostGIS_Lib_Version

PostGIS_Lib_Version — Retourne le numéro de version de l'extension PostGIS.

Synopsis

text **PostGIS_Lib_Version**();

Description

Retourne le numéro de version de l'extension PostGIS.

Exemples

```
SELECT PostGIS_Lib_Version();
 postgis_lib_version
---------------------
 3.4.0dev
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Version

7.21.10 PostGIS_PROJ_Version

PostGIS_PROJ_Version — Renvoie le numéro de version de la librairie PROJ4.

Synopsis

text **PostGIS_PROJ_Version**();

Description

Retourne le numéro de version de la bibliothèque PROJ et certaines options de configuration de proj.

Amélioration : 3.4.0 inclut désormais PROJ NETWORK_ENABLED, URL_ENDPOINT et DATABASE_PATH pour l'emplacement proj.db

Exemples

```
SELECT PostGIS_PROJ_Version();
  postgis_proj_version
-------------------------
7.2.1 NETWORK_ENABLED=OFF URL_ENDPOINT=https://cdn.proj.org USER_WRITABLE_DIRECTORY=/tmp/ ↩
    proj DATABASE_PATH=/usr/share/proj/proj.db
(1 row)
```

Voir aussi

PostGIS_PROJ_Compiled_Version, PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Versio
PostGIS_Version

7.21.11 PostGIS_PROJ_Compiled_Version

PostGIS_PROJ_Compiled_Version — Returns the version number of the PROJ library against which PostGIS was built.

Synopsis

text **PostGIS_PROJ_Compiled_Version**();

Description

Returns the version number of the PROJ library, or against which PostGIS was built.

Disponibilité : 3.5.0

Exemples

```
SELECT PostGIS_PROJ_Compiled_Version();
 postgis_proj_compiled_version
-------------------------------
 9.1.1
(1 row)
```

Voir aussi

PostGIS_PROJ_Version, PostGIS_Full_Version

7.21.12 PostGIS_Wagyu_Version

PostGIS_Wagyu_Version — Renvoie le numéro de version de la librairie interne de Wagyu.

Synopsis

text **PostGIS_Wagyu_Version**();

Description

Renvoie le numéro de version de la librairie interne Wagyu, ou `NULL` si le support Wagyu n'est pas activé.

Exemples

```
SELECT PostGIS_Wagyu_Version();
 postgis_wagyu_version
-----------------------
 0.5.0 (Internal)
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_PROJ_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_Version

7.21.13 PostGIS_Scripts_Build_Date

PostGIS_Scripts_Build_Date — Retourne la date de packaging des scripts PostGIS.

Synopsis

text **PostGIS_Scripts_Build_Date**();

Description

Retourne la date de packaging des scripts PostGIS.

Disponibilité : 1.0.0RC1

Exemples

```
SELECT PostGIS_Scripts_Build_Date();
  postgis_scripts_build_date
--------------------------
 2023-06-22 03:56:11
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_Version

7.21.14 PostGIS_Scripts_Installed

PostGIS_Scripts_Installed — Retourne la version des scripts PostGIS installés dans cette base de données.

Synopsis

text **PostGIS_Scripts_Installed**();

Description

Retourne la version des scripts PostGIS installés dans cette base de données.

Note
Si la sortie de cette fonction ne correspond pas à la sortie de PostGIS_Scripts_Released, vous avez probablement manqué de mettre correctement à niveau une base de données existante. Consultez la section Upgrading pour plus d'informations.

Disponibilité : 0.9.0

Exemples

```
SELECT PostGIS_Scripts_Installed();
  postgis_scripts_installed
---------------------------
 3.4.0dev 3.3.0rc2-993-g61bdf43a7
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_Scripts_Released, PostGIS_Version

7.21.15 PostGIS_Scripts_Released

PostGIS_Scripts_Released — Retourne le numéro de version du script postgis.sql publié avec la librairie PostGIS installée.

Synopsis

text **PostGIS_Scripts_Released**();

Description

Retourne le numéro de version du script postgis.sql publié avec la librairie PostGIS installée.

Note
À partir de la version 1.1.0, cette fonction renvoie la même valeur de PostGIS_Lib_Version. Conservée pour des raisons de rétrocompatibilité.

Disponibilité : 0.9.0

Exemples

```
SELECT PostGIS_Scripts_Released();
  postgis_scripts_released
---------------------------
 3.4.0dev 3.3.0rc2-993-g61bdf43a7
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_Scripts_Installed, PostGIS_Lib_Version

7.21.16 PostGIS_Version

PostGIS_Version — Retourne le numéro de version de PostGIS et les options de compilation.

Synopsis

text **PostGIS_Version**();

Description

Retourne le numéro de version de PostGIS et les options de compilation.

Exemples

```
SELECT PostGIS_Version();
                        postgis_version
-------------------------------------
 3.4 USE_GEOS=1 USE_PROJ=1 USE_STATS=1
(1 row)
```

Voir aussi

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version

7.22 Variables PostGIS GUC (Grand Unified Custom Variables)

7.22.1 postgis.gdal_datapath

postgis.gdal_datapath — Une option de configuration pour régler la valeur de l'option GDAL_DATA de GDAL. Si elle n'est pas assignée, la valeur de la variable d'environnement GDAL_DATA est utilisée.

Description

Une variable GUC PostgreSQL pour régler la valeur de l'option GDAL_DATA de GDAL. La valeur `postgis.gdal_datapath` devrait être le chemin physique complet vers les fichiers de données de GDAL.

Cette option de configuration est principalement destinée aux plateformes Windows où le chemin des fichiers de données de GDAL n'est pas codé en dur. Cette option devrait aussi être réglée lorsque les fichiers de données GDAL ne sont pas situés dans le chemin attendu par GDAL.

Note
Cette option peut être réglée dans le fichier de configuration postgresql.conf de PostgreSQL. Elle peut aussi être réglée par connexion ou par transaction.

Disponibilité : 2.2.0

Note
Des informations complémentaires sur GDAL_DATA sont disponibles dans la description des Options de configuration.

Exemples

Régler et remettre à la valeur par défaut `postgis.gdal_datapath`

```
SET postgis.gdal_datapath TO '/usr/local/share/gdal.hidden';
SET postgis.gdal_datapath TO default;
```

Réglage pour une base de données spécifique, sous Windows

```
ALTER DATABASE gisdb
SET postgis.gdal_datapath = 'C:/Program Files/PostgreSQL/9.3/gdal-data';
```

Voir aussi

PostGIS_GDAL_Version, ST_Transform

7.22.2 postgis.gdal_enabled_drivers

postgis.gdal_enabled_drivers — Option de configuration permettant de définir les drivers GDAL activés dans l'environnement PostGIS. Affecte la variable de configuration GDAL GDAL_SKIP.

Description

Option de configuration permettant de définir les drivers GDAL activés dans l'environnement PostGIS. Affecte la variable de configuration GDAL GDAL_SKIP. Cette option peut être définie dans le fichier de configuration de PostgreSQL : postgresql.conf. Elle peut également être définie par connexion ou transaction.

La valeur initiale de `postgis.gdal_enabled_drivers` peut aussi être définie en passant la variable d'environnement `POSTGIS_GDAL_ENABLED_DRIVERS` avec la liste des drivers activés au processus de démarrage de PostgreSQL.

Les drivers GDAL activés peuvent être spécifiés par leur nom abrégé ou leur code. Les noms courts ou les codes des drivers sont disponibles sur GDAL Raster Formats. Il est possible de spécifier plusieurs drivers en insérant un espace entre chacun d'eux.

Note
Trois codes spéciaux sont disponibles pour `postgis.gdal_enabled_drivers`. Les codes sont sensibles à la casse.

- `DISABLE_ALL` désactive tous les drivers GDAL. S'il est présent, `DISABLE_ALL` remplace toutes les autres valeurs de `postgis.gdal_enabled_drivers`.
- `ENABLE_ALL` active tous les drivers GDAL.
- `VSICURL` active le système de fichiers virtuels `/vsicurl/` de GDAL.

Lorsque `postgis.gdal_enabled_drivers` est défini sur DISABLE_ALL, les tentatives d'utilisation des rasters out-db, ST_FromGDALRaster(), ST_AsGDALRaster(), ST_AsTIFF(), ST_AsJPEG() et ST_AsPNG() se traduisent par des messages d'erreur.

Note
Dans l'installation standard de PostGIS, `postgis.gdal_enabled_drivers` est défini sur DISABLE_ALL.

Note
Des informations supplémentaires sur GDAL_SKIP sont disponibles sur le site GDAL Configuration Options.

Disponibilité : 2.2.0

Exemples

To set and reset `postgis.gdal_enabled_drivers` for current session

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.gdal_enabled_drivers = default;
```

Set for all new connections to a specific database to specific drivers

```
ALTER DATABASE mygisdb SET postgis.gdal_enabled_drivers TO 'GTiff PNG JPEG';
```

Setting for whole database cluster to enable all drivers. Requires super user access. Also note that database, session, and user settings override this.

```
--writes to postgres.auto.conf
ALTER SYSTEM SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
 --Reloads postgres conf
SELECT pg_reload_conf();
```

Voir aussi

ST_FromGDALRaster, ST_AsGDALRaster, ST_AsTIFF, ST_AsPNG, ST_AsJPEG, postgis.enable_outdb_rasters

7.22.3 postgis.enable_outdb_rasters

postgis.enable_outdb_rasters — Une option de configuration booléenne pour permettre l'accès aux bandes matricielles de out-db.

Description

Une option de configuration booléenne pour permettre l'accès aux bandes matricielles de out-db. Cette option peut être définie dans le fichier de configuration de PostgreSQL : postgresql.conf. Elle peut également être définie par connexion ou transaction.

La valeur initiale de `postgis.enable_outdb_rasters` peut également être définie en passant la variable d'environnement `POSTGIS_ENABLE_OUTDB_RASTERS` avec une valeur non nulle au processus de démarrage de PostgreSQL.

Note
Même si `postgis.enable_outdb_rasters` est True, le GUC `postgis.gdal_enabled_drivers` détermine les formats raster accessibles.

Note
Dans l'installation standard de PostGIS, `postgis.enable_outdb_rasters` est défini sur False.

Disponibilité : 2.2.0

Exemples

Définir et réinitialiser `postgis.enable_outdb_rasters` pour la session actuelle

```
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 for all new connections to a specific database

```
ALTER DATABASE gisdb SET postgis.enable_outdb_rasters = true;
```

Setting for whole database cluster. Requires super user access. Also note that database, session, and user settings override this.

```
--writes to postgres.auto.conf
ALTER SYSTEM SET postgis.enable_outdb_rasters = true;
 --Reloads postgres conf
SELECT pg_reload_conf();
```

Voir aussi

postgis.gdal_enabled_drivers postgis.gdal_vsi_options

7.22.4 postgis.gdal_vsi_options

postgis.gdal_vsi_options — Une chaîne de configuration pour définir les options utilisées lors de l'utilisation d'un raster out-db.

Description

Une chaîne de configuration pour définir les options utilisées lors de l'utilisation d'un raster out-db. Les options de configuration contrôlent des éléments tels que l'espace alloué par GDAL au cache des données locales, la lecture ou non des aperçus et les clés d'accès à utiliser pour les sources de données out-db distantes.

Disponibilité : 3.2.0

Exemples

Définit `postgis.gdal_vsi_options` pour la session actuelle :

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx AWS_SECRET_ACCESS_KEY= ↩
    yyyyyyyyyyyyyyyyyyyyyyyyyy';
```

Définissez `postgis.gdal_vsi_options` uniquement pour la *transaction courante* en utilisant le mot clé `LOCAL` :

```
SET LOCAL postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx  ↩
    AWS_SECRET_ACCESS_KEY=yyyyyyyyyyyyyyyyyyyyyyyyyy';
```

Voir aussi

postgis.enable_outdb_rasters postgis.gdal_enabled_drivers

7.23 Fonctions de dépannage

7.23.1 PostGIS_AddBBox

PostGIS_AddBBox — Ajoute une bounding box à la géométrie.

Synopsis

geometry **PostGIS_AddBBox**(geometry geomA);

Description

Ajouter une boîte de délimitation à la géométrie. Cela rendrait les requêtes basées sur la boîte de délimitation plus rapides, mais augmenterait la taille de la géométrie.

Note
Les boîtes de délimitation sont automatiquement ajoutées aux géométries, de sorte qu'en général, cela n'est pas nécessaire, à moins que la boîte de délimitation générée ne soit corrompue d'une manière ou d'une autre ou que vous ayez une ancienne installation qui ne contient pas les boîtes de délimitation. Dans ce cas, vous devez supprimer l'ancienne et ajouter la nouvelle.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
UPDATE sometable
 SET geom =  PostGIS_AddBBox(geom)
 WHERE PostGIS_HasBBox(geom) = false;
```

Voir aussi

PostGIS_DropBBox, PostGIS_HasBBox

7.23.2 PostGIS_DropBBox

PostGIS_DropBBox — Supprime le cache de la boîte de délimitation de la géométrie.

Synopsis

geometry **PostGIS_DropBBox**(geometry geomA);

Description

Supprime le cache de la boîte de délimitation de la géométrie. Cela réduit la taille de la géométrie, mais rend les requêtes basées sur les boîtes de délimitation plus lentes. Elle est également utilisée pour supprimer une boîte de délimitation corrompue. Un signe révélateur d'une boîte de délimitation corrompue dans le cache est lorsque vos ST_Intersects et autres requêtes de relations laissent de côté des géométries qui devraient légitimement renvoyer un résultat vrai.

Note
Les boîtes de délimitation sont automatiquement ajoutées aux géométries et améliorent la vitesse des requêtes. En général, cela n'est donc pas nécessaire, sauf si le boîte de délimitation générée est corrompue d'une manière ou d'une autre ou si vous avez une ancienne installation qui ne contient pas de boîtes de délimitation. Dans ce cas, il est nécessaire de supprimer l'ancienne installation et de la remplacer par une nouvelle. Ce type de corruption a été observé dans les séries 8.3-8.3.6 où les bboxes mises en cache n'étaient pas toujours recalculées lorsqu'une géométrie changeait et lors des mises à jour vers une version plus récente où le dump non rechargé ne corrigeait pas les bboxes déjà corrompues. La mise à jour vers une version plus récente sans rechargement du dump ne corrigera pas les boîtes déjà corrompues. Il est donc possible de corriger manuellement en utilisant la méthode ci-dessous et de lire la bbox ou de recharger le dump.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
--This example drops bounding boxes where the cached box is not correct
                        --The force to ST_AsBinary before applying Box2D forces a  ↩
                            recalculation of the box, and Box2D applied to the table  ↩
                            geometry always
                        -- returns the cached bounding box.
                        UPDATE sometable
 SET geom =  PostGIS_DropBBox(geom)
 WHERE Not (Box2D(ST_AsBinary(geom)) = Box2D(geom));

        UPDATE sometable
 SET geom =  PostGIS_AddBBox(geom)
 WHERE Not PostGIS_HasBBOX(geom);
```

Voir aussi

PostGIS_AddBBox, PostGIS_HasBBox, Box2D

7.23.3 PostGIS_HasBBox

PostGIS_HasBBox — Renvoie TRUE si la bbox de cette géométrie est en cache, sinon FALSE.

Synopsis

boolean **PostGIS_HasBBox**(geometry geomA);

Description

Renvoie TRUE si la boîte de délimitation de cette géométrie est mise en cache, FALSE dans le cas contraire. Utilisez PostGIS_AddBBox et PostGIS_DropBBox pour contrôler la mise en cache.

Cette méthode prend en charge les types Circular String et Curve.

Exemples

```
SELECT geom
FROM sometable WHERE PostGIS_HasBBox(geom) = false;
```

Voir aussi

PostGIS_AddBBox, PostGIS_DropBBox

Chapter 8

Référence des fonctions SFCGAL

SFCGAL est une bibliothèque C++ autour de CGAL qui fournit des fonctions spatiales avancées en 2D et 3D. Pour des raisons de robustesse, les coordonnées géométriques sont représentées par des nombres rationnels exacts.

Les instructions d'installation de la bibliothèque se trouvent sur la page d'accueil de SFCGAL (http://www.sfcgal.org). Pour activer les fonctions, utilisez `create extension postgis_sfcgal`.

8.1 Fonctions de gestion de SFCGAL

8.1.1 postgis_sfcgal_version

postgis_sfcgal_version — Retourne la version de SFCGAL utilisée

Synopsis

text **postgis_sfcgal_version**(void);

Description

Retourne la version de SFCGAL utilisée

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

Voir aussi

postgis_sfcgal_full_version

8.1.2 postgis_sfcgal_full_version

postgis_sfcgal_full_version — Retourne la version complète de SFCGAL en cours d'utilisation, y compris les versions CGAL et Boost

Synopsis

text **postgis_sfcgal_full_version**(void);

Description

Retourne la version complète de SFCGAL utilisée, y compris les versions CGAL et Boost

Disponibilité : 3.3.0

Cette méthode nécessite le backend SFCGAL.

Voir aussi

postgis_sfcgal_version

8.2 Fonctions d'accès et de modifications SFCGAL

8.2.1 CG_ForceLHR

CG_ForceLHR — Force l'orientation LHR d'un objet

Synopsis

geometry **CG_ForceLHR**(geometry geom);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.2 CG_IsPlanar

CG_IsPlanar — Vérifie si une surface est planaire ou non

Synopsis

boolean **CG_IsPlanar**(geometry geom);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.3 CG_IsSolid

CG_IsSolid — Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.

Synopsis

boolean **CG_IsSolid**(geometry geom1);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.4 CG_MakeSolid

CG_MakeSolid — Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.

Synopsis

geometry **CG_MakeSolid**(geometry geom1);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.5 CG_Orientation

CG_Orientation — Détermine l'orientation d'une surface

Synopsis

integer **CG_Orientation**(geometry geom);

Description

Cette fonction ne s'applique qu'aux polygones. Elle renvoie -1 si le polygone est orienté dans le sens inverse des aiguilles d'une montre et 1 s'il est orienté dans le sens des aiguilles d'une montre.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

8.2.6 CG_Area

CG_Area — Calcule la surface d'une géométrie

Synopsis

double precision **CG_Area**(geometry geom);

Description

Calcule la surface d'une géométrie.

Réalisé par le module SFCGAL

Note
NOTE : cette fonction renvoie une valeur en double précision représentant la surface.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples de géométrie

```
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))');
               cg_area
               --------
               25
               (1 row)
```

Voir aussi

ST_3DArea, ST_Area

8.2.7 CG_3DArea

CG_3DArea — Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.

Synopsis

float**CG_3DArea**(geometry geom1);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 8.1, 10.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

Remarque : par défaut, une PolyhedralSurface construite à partir de WKT est une géométrie de surface, et non un solide. Elle possède donc une surface. Une fois convertie en solide, elle n'a plus de surface.

```
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);

            cube_surface_area | solid_surface_area
            ------------------+--------------------
            6 |                  0
```

Voir aussi

CG_Area, CG_MakeSolid, CG_IsSolid, CG_Area

8.2.8 CG_Volume

CG_Volume — Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.

Synopsis

float **CG_Volume**(geometry geom1);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 9.1 (same as CG_3DVolume)

Exemple

When closed surfaces are created with WKT, they are treated as areal rather than solid. To make them solid, you need to use CG_MakeSolid. Areal geometries have no volume. Here is an example to demonstrate.

```
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);

    cube_surface_vol | solid_surface_vol
    -----------------+------------------
    0 |                1
```

Voir aussi

CG_3DArea, CG_MakeSolid, CG_IsSolid

8.2.9 ST_ForceLHR

ST_ForceLHR — Force l'orientation LHR d'un objet

Synopsis

geometry **ST_ForceLHR**(geometry geom);

Description

Warning
ST_ForceLHR is deprecated as of 3.5.0. Use CG_ForceLHR instead.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.10 ST_IsPlanar

ST_IsPlanar — Vérifie si une surface est planaire ou non

Synopsis

boolean **ST_IsPlanar**(geometry geom);

Description

Warning
ST_IsPlanar is deprecated as of 3.5.0. Use CG_IsPlanar instead.

Disponibilité : 2.2.0 : Ce point était documenté dans la version 2.1.0 mais a été accidentellement omis dans la version 2.1.

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.11 ST_IsSolid

ST_IsSolid — Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.

Synopsis

boolean **ST_IsSolid**(geometry geom1);

Description

Warning
ST_IsSolid is deprecated as of 3.5.0. Use CG_IsSolid instead.

Disponibilité : 2.2.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.12 ST_MakeSolid

ST_MakeSolid — Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.

Synopsis

geometry **ST_MakeSolid**(geometry geom1);

Description

Warning
ST_MakeSolid is deprecated as of 3.5.0. Use CG_MakeSolid instead.

Disponibilité : 2.2.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.2.13 ST_Orientation

ST_Orientation — Détermine l'orientation d'une surface

Synopsis

integer **ST_Orientation**(geometry geom);

Description

Warning
ST_Orientation is deprecated as of 3.5.0. Use CG_Orientation instead.

Cette fonction ne s'applique qu'aux polygones. Elle renvoie -1 si le polygone est orienté dans le sens inverse des aiguilles d'une montre et 1 s'il est orienté dans le sens des aiguilles d'une montre.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

8.2.14 ST_3DArea

ST_3DArea — Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.

Synopsis

float**ST_3DArea**(geometry geom1);

Description

Warning
ST_3DArea is deprecated as of 3.5.0. Use CG_3DArea instead.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 8.1, 10.5

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

Remarque : par défaut, une PolyhedralSurface construite à partir de WKT est une géométrie de surface, et non un solide. Elle possède donc une surface. Une fois convertie en solide, elle n'a plus de surface.

```
SELECT ST_3DArea(geom) As cube_surface_area,
            ST_3DArea(ST_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);

            cube_surface_area | solid_surface_area
            ------------------+--------------------
            6 |                 0
```

Voir aussi

ST_Area, ST_MakeSolid, ST_IsSolid, ST_Area

8.2.15 ST_Volume

ST_Volume — Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.

Synopsis

float **ST_Volume**(geometry geom1);

Description

Warning
ST_Volume is deprecated as of 3.5.0. Use CG_Volume instead.

Disponibilité : 2.2.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3 : 9.1 (identique à ST_3DVolume)

Exemple

Lorsque des surfaces fermées sont créées avec WKT, elles sont traitées comme des surfaces aréolaires plutôt que comme des surfaces solides. Pour les rendre solides, vous devez utiliser ST_MakeSolid. Les géométries aréolaires n'ont pas de volume. Voici un exemple pour le démontrer.

```
SELECT ST_Volume(geom) As cube_surface_vol,
    ST_Volume(ST_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);

    cube_surface_vol | solid_surface_vol
    -----------------+------------------
    0 |                1
```

Voir aussi

ST_3DArea, ST_MakeSolid, ST_IsSolid

8.3 Fonctions de traitement et de relation SFCGAL

8.3.1 CG_Intersection

CG_Intersection — Calcul l'intersection de deux géométries

Synopsis

geometry **CG_Intersection**(geometry geomA , geometry geomB);

Description

Calcule l'intersection de deux géométries.

Réalisé par le module SFCGAL

Note
NOTE : cette fonction renvoie une géométrie représentant l'intersection.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples de géométrie

```
SELECT ST_AsText(CG_Intersection('LINESTRING(0 0, 5 5)', 'LINESTRING(5 0, 0 5)'));
                cg_intersection
                -----------------
                POINT(2.5 2.5)
                (1 row)
```

Voir aussi

ST_3DIntersection, ST_Intersection

8.3.2 CG_Intersects

CG_Intersects — Teste si deux géométries se croisent (elles ont au moins un point en commun)

Synopsis

boolean **CG_Intersects**(geometry geomA , geometry geomB);

Description

Renvoie `true` si deux géométries se croisent. Les géométries se croisent si elles ont un point commun.

Réalisé par le module SFCGAL

Note
NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples de géométrie

```
SELECT CG_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 2 0, 0 2 )'::geometry);
    cg_intersects
    ---------------
    f
    (1 row)
    SELECT CG_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 0 0, 0 2 )'::geometry);
    cg_intersects
    ---------------
    t
    (1 row)
```

Voir aussi

CG_3DIntersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.3 CG_3DIntersects

CG_3DIntersects — Teste si deux géométries 3D intersectent

Synopsis

boolean **CG_3DIntersects**(geometry geomA , geometry geomB);

Description

Teste si deux géométries 3D intersectent. Les géométries 3D intersectent si elles ont au moins un point commun dans l'espace tridimensionnel.

Réalisé par le module SFCGAL

Note
NOTE : il s'agit de la version "autorisée" qui renvoie un booléen et non un entier.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples de géométrie

```
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)))');
        cg_3dintersects
        ---------------
        t
        (1 row)
```

Voir aussi

CG_Intersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.4 CG_Difference

CG_Difference — Calcul la différence géométrique entre deux géométries

Synopsis

geometry **CG_Difference**(geometry geomA , geometry geomB);

Description

Calcule la différence géométrique entre deux géométries. La géométrie résultante est un ensemble de points présents dans la géométrie A mais pas dans la géométrie B.

Réalisé par le module SFCGAL

Note
NOTE : cette fonction renvoie une géométrie.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples de géométrie

```
SELECT ST_AsText(CG_Difference('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))'::geometry, 'LINESTRING ↩
    (0 0, 2 2)'::geometry));
    cg_difference
    ---------------
    POLYGON((0 0,1 0,1 1,0 1,0 0))
    (1 row)
```

Voir aussi

ST_3DDifference, ST_Difference

8.3.5 ST_3DDifference

ST_3DDifference — Effectuer une différence 3D

Synopsis

geometry **ST_3DDifference**(geometry geom1, geometry geom2);

Description

Warning
ST_3DDifference is deprecated as of 3.5.0. Use CG_3DDifference instead.

Renvoie la partie de geom1 qui ne fait pas partie de geom2.
Disponibilité : 2.2.0

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.3.6 CG_3DDifference

CG_3DDifference — Effectuer une différence 3D

Synopsis

geometry **CG_3DDifference**(geometry geom1, geometry geom2);

Description

Renvoie la partie de geom1 qui ne fait pas partie de geom2.
Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

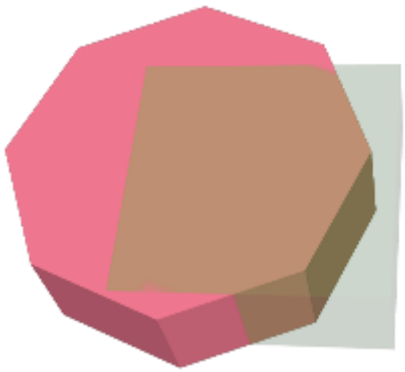
Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

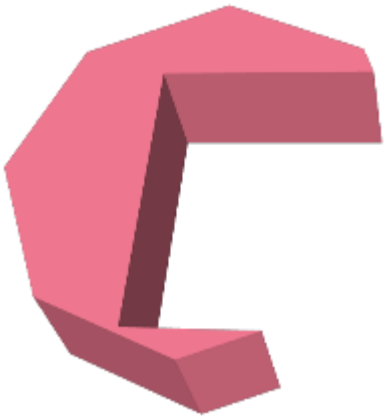
Les images 3D ont été générées à l'aide de PostGIS ST_AsX3D et rendues en HTML à l'aide de X3Dom HTML Javascript rendering library.

```
SELECT CG_Extrude(ST_Buffer( ↩
        ST_GeomFromText('POINT(100 90)'),
                                50, ' ↩
        quad_segs=2'),0,0,30) AS geom1,
                                ↩
        CG_Extrude(ST_Buffer(ST_GeomFromText('POINT(80 80)'),
                                50, ' ↩
        quad_segs=1'),0,0,30) AS geom2;
```



Géométries 3D originales superposées. geom2 est la partie qui sera retirée.

```
SELECT CG_3DDifference(geom1,geom2)
                                FROM ( SELECT ↩
         CG_Extrude(ST_Buffer(ST_GeomFromText('POINT
                                50, ' ↩
        quad_segs=2'),0,0,30) AS geom1,
                                CG_Extrude( ↩
                                30),
                                50, ' ↩
        quad_segs=1'),0,0,30) AS geom2 ) As t;
```



Ce qui reste après la suppression de geom2

Voir aussi

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DUnion

8.3.7 CG_Distance

CG_Distance — Calcule la distance minimale entre deux géométries

Synopsis

double precision **CG_Distance**(geometry geomA , geometry geomB);

Description

Calcule la distance minimale entre deux géométries.

Réalisé par le module SFCGAL

Note

NOTE : cette fonction renvoie une valeur double précision représentant la distance.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples de géométrie

```
SELECT CG_Distance('LINESTRING(0.0 0.0,-1.0 -1.0)', 'LINESTRING(3.0 4.0,4.0 5.0)');
        cg_distance
        -------------
        2.0
        (1 row)
```

Voir aussi

CG_3DDistance, CG_Distance

8.3.8 CG_3DDistance

CG_3DDistance — Calcule la distance 3D minimale entre deux géométries

Synopsis

double precision **CG_3DDistance**(geometry geomA , geometry geomB);

Description

Calcule la distance 3D minimale entre deux géométries.

Réalisé par le module SFCGAL

Note
NOTE : cette fonction renvoie une valeur double précision représentant la distance 3D.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples de géométrie

```
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))');
        cg_3ddistance
        ----------------
        1
        (1 row)
```

Voir aussi

CG_Distance, ST_3DDistance

8.3.9 ST_3DConvexHull

ST_3DConvexHull — Calcule l'enveloppe convexe 3D d'une géométrie.

Synopsis

geometry **ST_3DConvexHull**(geometry geom1);

Description

Warning
ST_3DConvexHull is deprecated as of 3.5.0. Use CG_3DConvexHull instead.

Disponibilité : 3.3.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.3.10 CG_3DConvexHull

CG_3DConvexHull — Calcule l'enveloppe convexe 3D d'une géométrie.

Synopsis

geometry **CG_3DConvexHull**(geometry geom1);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

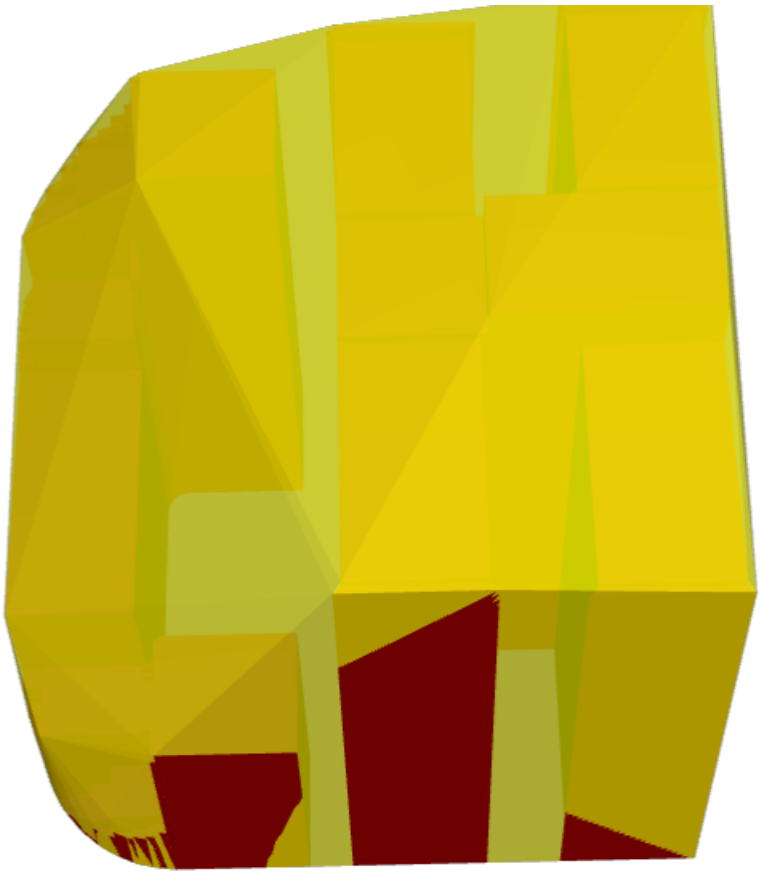
Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_AsText(CG_3DConvexHull('LINESTRING Z(0 0 5, 1 5 3, 5 7 6, 9 5 3 , 5 7 5, 6 3 5) ↩
    '::geometry));
```

```
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)))
```

```
WITH f AS (SELECT i, CG_Extrude(geom, 0,0, i ) AS geom
        FROM ST_Subdivide(ST_Letters('CH'),5) WITH ORDINALITY AS sd(geom,i)
        )
        SELECT CG_3DConvexHull(ST_Collect(f.geom) )
        FROM f;
```



Géométrie originale superposée à l'enveloppe convexe 3D

Voir aussi

ST_Letters, ST_AsX3D

8.3.11 ST_3DIntersection

ST_3DIntersection — Réaliser une intersection 3D

Synopsis

geometry **ST_3DIntersection**(geometry geom1, geometry geom2);

Description

Warning
ST_3DIntersection is deprecated as of 3.5.0. Use CG_3DIntersection instead.

Renvoie une géométrie qui est la partie partagée entre geom1 et geom2.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.3.12 CG_3DIntersection

CG_3DIntersection — Réaliser une intersection 3D

Synopsis

geometry **CG_3DIntersection**(geometry geom1, geometry geom2);

Description

Renvoie une géométrie qui est la partie partagée entre geom1 et geom2.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

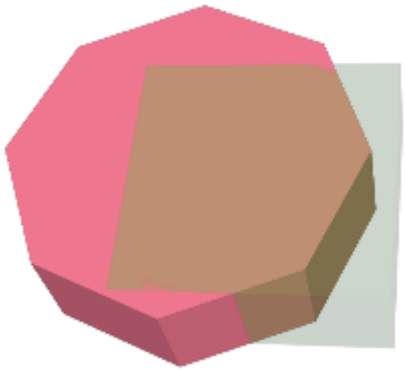
Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

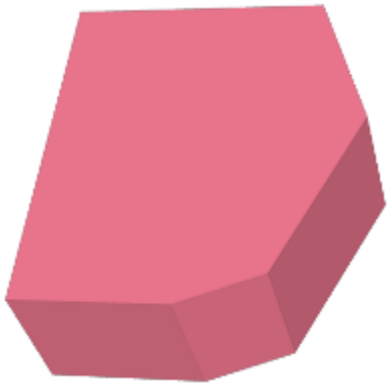
Les images 3D ont été générées à l'aide de PostGIS ST_AsX3D et rendues en HTML à l'aide de X3Dom HTML Javascript rendering library.

```
SELECT CG_Extrude(ST_Buffer( ↩
            ST_GeomFromText('POINT(100 90)'),
                                    50, ' ↩
            quad_segs=2'),0,0,30) AS geom1,
                                    ↩
            CG_Extrude(ST_Buffer(ST_GeomFromText('POINT(80 80)'),
                                    50, ' ↩
            quad_segs=1'),0,0,30) AS geom2;
```



Géométries 3D originales superposées. geom2 est représentée en semi-transparence

```
SELECT CG_3DIntersection(geom1,geom2)
                                    FROM (  ↩
            SELECT CG_Extrude(ST_Buffer(ST_GeomFromT
                                    50, ' ↩
            quad_segs=2'),0,0,30) AS geom1,
                                    ↩
            CG_Extrude(ST_Buffer(ST_GeomFromText('PO
                                    50, ' ↩
            quad_segs=1'),0,0,30) AS geom2 ) As t;
```



Intersection de geom1 et geom2

Lignes et polygones en 3D

```
SELECT ST_AsText(CG_3DIntersection(linestring, polygon)) As wkt
    FROM  ST_GeomFromText('LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)') AS  ↩
        linestring
    CROSS JOIN ST_GeomFromText('POLYGON((0 0 8, 0 1 8, 1 1 8, 1 0 8, 0 0 8))') AS polygon;

    wkt
    --------------------------------
    LINESTRING Z (1 1 8,0.5 0.5 8)
```

Cube (surface polyédrique fermée) et polygone Z

```
SELECT ST_AsText(CG_3DIntersection(
ST_GeomFromText('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))'::geometry))
```

```
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)))
```

L'intersection de 2 solides qui résulte en une intersection volumétrique est également un solide (ST_Dimension renvoie 3)

```
SELECT ST_AsText(CG_3DIntersection( CG_Extrude(ST_Buffer('POINT(10 20)'::geometry,10,1) ↩
    ,0,0,30),
            CG_Extrude(ST_Buffer('POINT(10 20)'::geometry,10,1),2,0,10) ));
```

```
POLYHEDRALSURFACE Z (((13.3333333333333 13.3333333333333 10,20 20 0,20 20  ↩
    10,13.3333333333333 13.3333333333333 10)),
        ((20 20 10,16.6666666666667 23.3333333333333 10,13.3333333333333 13.3333333333333  ↩
           10,20 20 10)),
```

```
        ((20 20 0,16.6666666666667 23.3333333333333 10,20 20 10,20 20 0)),
        ((13.3333333333333 13.3333333333333 10,10 10 0,20 20 0,13.3333333333333  ↩
           13.3333333333333 10)),
        ((16.6666666666667 23.3333333333333 10,12 28 10,13.3333333333333 13.3333333333333  ↩
           10,16.6666666666667 23.3333333333333 10)),
        ((20 20 0,9.99999999999995 30 0,16.6666666666667 23.3333333333333 10,20 20 0)),
        ((10 10 0,9.99999999999995 30 0,20 20 0,10 10 0)),((13.3333333333333  ↩
           13.3333333333333 10,12 12 10,10 10 0,13.3333333333333 13.3333333333333 10)),
        ((12 28 10,12 12 10,13.3333333333333 13.3333333333333 10,12 28 10)),
        ((16.6666666666667 23.3333333333333 10,9.99999999999995 30 0,12 28  ↩
           10,16.6666666666667 23.3333333333333 10)),
        ((10 10 0,0 20 0,9.99999999999995 30 0,10 10 0)),
        ((12 12 10,11 11 10,10 10 0,12 12 10)),((12 28 10,11 11 10,12 12 10,12 28 10)),
        ((9.99999999999995 30 0,11 29 10,12 28 10,9.99999999999995 30 0)),((0 20 0,2 20  ↩
           10,9.99999999999995 30 0,0 20 0)),
        ((10 10 0,2 20 10,0 20 0,10 10 0)),((11 11 10,2 20 10,10 10 0,11 11 10)),((12 28  ↩
           10,11 29 10,11 11 10,12 28 10)),
        ((9.99999999999995 30 0,2 20 10,11 29 10,9.99999999999995 30 0)),((11 11 10,11 29  ↩
           10,2 20 10,11 11 10)))
```

8.3.13 CG_Union

CG_Union — Calcule l'union de deux géométries

Synopsis

geometry **CG_Union**(geometry geomA , geometry geomB);

Description

Calcule l'union de deux géométries.

Réalisé par le module SFCGAL

Note
NOTE : cette fonction renvoie une géométrie représentant l'union.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples de géométrie

```
SELECT CG_Union('POINT(.5 0)', 'LINESTRING(-1 0,1 0)');
               cg_union
               -----------
               LINESTRING(-1 0,0.5 0,1 0)
               (1 row)
```

Voir aussi

ST_3DUnion, ST_Union

8.3.14 ST_3DUnion

ST_3DUnion — Effectuer l'union 3D.

Synopsis

geometry **ST_3DUnion**(geometry geom1, geometry geom2);
geometry **ST_3DUnion**(geometry set g1field);

Description

Warning
ST_3DUnion is deprecated as of 3.5.0. Use CG_3DUnion instead.

Disponibilité : 2.2.0

Disponibilité : 3.3.0 la variante agrégée a été ajoutée

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Variante agrégée: renvoie une géométrie qui est l'union 3D d'un ensemble de géométries. La fonction ST_3DUnion() est une fonction "agrégée" dans la terminologie de PostgreSQL. Cela signifie qu'elle opère sur des lignes de données, de la même manière que les fonctions SUM() et AVG() et, comme la plupart des agrégats, elle ignore les géométries NULL.

8.3.15 CG_3DUnion

CG_3DUnion — Perform 3D union using postgis_sfcgal.

Synopsis

geometry **CG_3DUnion**(geometry geom1, geometry geom2);
geometry **CG_3DUnion**(geometry set g1field);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette méthode implémente la spécification SQL/MM. SQL-MM IEC 13249-3: 5.1

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

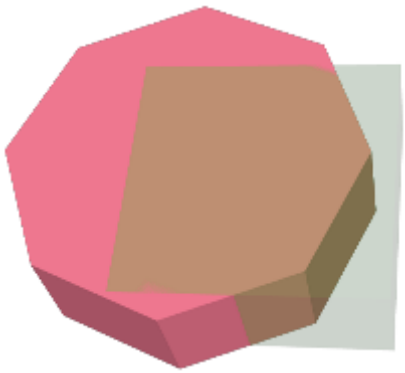
Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Aggregate variant: returns a geometry that is the 3D union of a rowset of geometries. The CG_3DUnion() function is an "aggregate" function in the terminology of PostgreSQL. That means that it operates on rows of data, in the same way the SUM() and AVG() functions do and like most aggregates, it also ignores NULL geometries.

Exemples

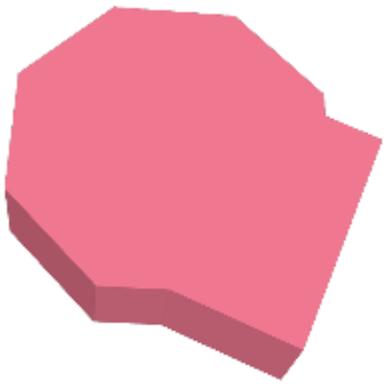
Les images 3D ont été générées à l'aide de PostGIS ST_AsX3D et rendues en HTML à l'aide de X3Dom HTML Javascript rendering library.

```
SELECT CG_Extrude(ST_Buffer( ↩
                    ST_GeomFromText('POINT(100 90)'),
                                50, ' ↩
                    quad_segs=2'),0,0,30) AS geom1,
                                  ↩
                    CG_Extrude(ST_Buffer(ST_GeomFromText('POINT(80 80)'),
                                50, ' ↩
                    quad_segs=1'),0,0,30) AS geom2;
```



Géométries 3D originales superposées. geom2 est celle qui est transparente.

```
SELECT CG_3DUnion(geom1,geom2)
                                    FROM (  ↩
SELECT CG_Extrude(ST_Buffer(ST_G
                                50, ' ↩
quad_segs=2'),0,0,30) AS geom1,
                                  ↩
CG_Extrude(ST_Buffer(ST_GeomFrom
                                50, ' ↩
quad_segs=1'),0,0,30) AS geom2 )
```



Union de geom1 et geom2

Voir aussi

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DDifference

8.3.16 ST_AlphaShape

ST_AlphaShape — Calcul d'une forme Alpha entourant une géométrie

Synopsis

geometry **ST_AlphaShape**(geometry geom, float alpha, boolean allow_holes = false);

Description

Warning
ST_AlphaShape is deprecated as of 3.5.0. Use CG_AlphaShape instead.

Calcule la forme alpha des points d'une géométrie. Une forme alpha est une géométrie polygonale (généralement) concave qui contient tous les sommets de l'entrée et dont les sommets sont un sous-ensemble des sommets de l'entrée. Une forme alpha permet de se rapprocher davantage de la forme de l'entrée que la forme produite par l' enveloppe convexe.

8.3.17 CG_AlphaShape

CG_AlphaShape — Calcul d'une forme Alpha entourant une géométrie

Synopsis

geometry **CG_AlphaShape**(geometry geom, float alpha, boolean allow_holes = false);

Description

Calcule la forme alpha des points d'une géométrie. Une forme alpha est une géométrie polygonale (généralement) concave qui contient tous les sommets de l'entrée et dont les sommets sont un sous-ensemble des sommets de l'entrée. Une forme alpha permet de se rapprocher davantage de la forme de l'entrée que la forme produite par l' enveloppe convexe.

La "proximité de l'ajustement" est contrôlée par le paramètre `alpha`, qui peut prendre des valeurs comprises entre 0 et l'infini. Des valeurs alpha plus petites produisent des résultats plus concaves. Les valeurs alpha supérieures à une certaine valeur dépendant des données produisent l'enveloppe convexe de l'entrée.

Note
Conformément à l'implémentation du CGAL, la valeur alpha est le *carré* du rayon du disque utilisé dans l'algorithme Alpha-Shape pour "éroder" la triangulation de Delaunay des points d'entrée. Voir CGAL Alpha-Shapes pour plus d'informations. Cela diffère de la définition originale des formes alpha, qui définit alpha comme le rayon du disque d'érosion.

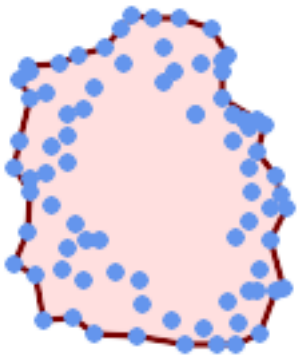
La forme calculée ne contient pas de trous, sauf si l'argument facultatif `allow_holes` est spécifié comme true.

Cette fonction calcule effectivement une enveloppe concave d'une géométrie d'une manière similaire à ST_ConcaveHull, mais utilise CGAL et un algorithme différent.

Availability: 3.5.0 - requires SFCGAL >= 1.4.1.

Cette méthode nécessite le backend SFCGAL.

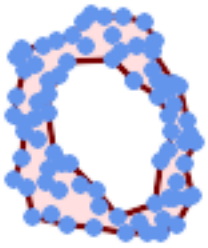
Exemples



Alpha-shape of a MultiPoint (same example As CG_OptimalAlphaShape)

```
SELECT ST_AsText(CG_AlphaShape('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,80.2));
```

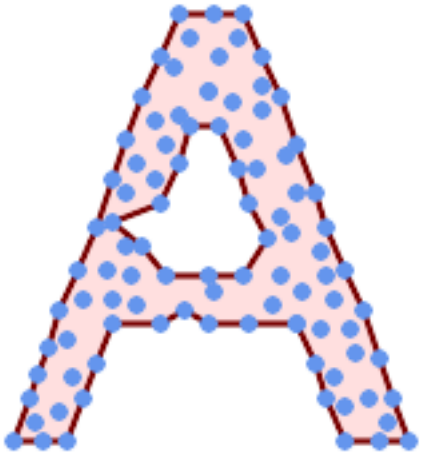
```
POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,
        37 23,30 22,28 33,23 36,26 44,27 54,23 60,24 67,27 77,
        24 82,26 85,34 86,39 88,45 90,49 95,52 98,57 97,
        64 97,72 95,76 88,75 84,83 72,85 71,88 58,89 53))
```



Alpha-shape of a MultiPoint, allowing holes (same example as CG_OptimalAlphaShape)

```
SELECT ST_AsText(CG_AlphaShape('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, 100.1,true))
```

```
POLYGON((89 53,91 50,87 42,90 30,84 19,78 16,73 16,65 16,53 18,43 19,30 22,28 33,23 36,
26 44,27 54,23 60,24 67,27 77,24 82,26 85,34 86,39 88,45 90,49 95,52 98,57 97,64 97,72 95,
76 88,75 84,83 72,85 71,88 58,89 53),(36 61,36 68,40 75,43 80,60 81,68 73,77 67,
81 60,82 54,81 47,78 43,76 27,62 22,54 32,44 42,38 46,36 61))
```



Forme alpha d'un MultiPoint, autorisant les trous (même exemple que ST_ConcaveHull)

```
SELECT ST_AsText(CG_AlphaShape(
                '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,102.2, true));
```

```
POLYGON((26 20,32 36,35 45,39 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 64,114 64,99 64,90 69,81 64,63 64,57  ↩
                49,52 36,46 20,37 20,26 20),
```

```
            (74 93,81 109,88 124,92 138,103 138,110 122,114 109,121 96,112 82,99 82,83 ↩
               82,74 93))
```

Voir aussi

ST_ConcaveHull, CG_OptimalAlphaShape

8.3.18 CG_ApproxConvexPartition

CG_ApproxConvexPartition — Calcul de la partition convexe approximative de la géométrie du polygone

Synopsis

geometry **CG_ApproxConvexPartition**(geometry geom);

Description

Calcule la partition convexe approximative de la géométrie du polygone (à l'aide d'une triangulation).

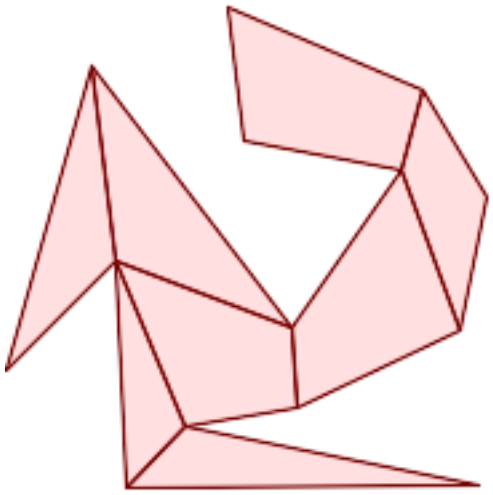
Note
Une partition d'un polygone P est un ensemble de polygones tels que les intérieurs des polygones ne se coupent pas et que l'union des polygones est égale à l'intérieur du polygone original P. Les fonctions CG_ApproxConvexPartition et CG_GreeneApproxConvexPartition produisent des partitions convexes approximativement optimales. Ces deux fonctions produisent des décompositions convexes en décomposant d'abord le polygone en polygones plus simples ; CG_ApproxConvexPartition utilise une triangulation et CG_GreeneApproxConvexPartition une partition monotone. Ces deux fonctions garantissent toutes deux qu'elles ne produiront pas plus de quatre fois le nombre optimal de pièces convexes, mais elles diffèrent par leur complexité d'exécution. Bien que l'algorithme d'approximation basé sur la triangulation produise souvent moins de pièces convexes, ce n'est pas toujours le cas.

Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0.

Nécessite SFCGAL >= 1.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples



Partition convexe approximative (même exemple que CG_YMonotonePartition, CG_GreeneApproxConvexPartition et CG_OptimalConvexPartition)

```
SELECT ST_AsText(CG_ApproxConvexPartition('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));
```

```
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)))
```

Voir aussi

CG_YMonotonePartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

8.3.19 ST_ApproximateMedialAxis

ST_ApproximateMedialAxis — Calculer l'axe médian approximatif d'une géométrie aréolaire.

Synopsis

geometry **ST_ApproximateMedialAxis**(geometry geom);

Description

Warning
ST_ApproximateMedialAxis is deprecated as of 3.5.0. Use CG_ApproximateMedialAxis instead.

Retourne un axe médian approximatif pour l'entrée aréolaire en se basant sur son squelette (straight skeleton). Utilise une API spécifique à SFCGAL lorsque compilée avec une version compatible (1.2.0+). Sinon, la fonction est juste un wrapper de CG_StraightSkeleton (cas le plus lent).

Disponibilité : 2.2.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.3.20 CG_ApproximateMedialAxis

CG_ApproximateMedialAxis — Calculer l'axe médian approximatif d'une géométrie aréolaire.

Synopsis

geometry **CG_ApproximateMedialAxis**(geometry geom);

Description

Retourne un axe médian approximatif pour l'entrée aréolaire en se basant sur son squelette (straight skeleton). Utilise une API spécifique à SFCGAL lorsque compilée avec une version compatible (1.2.0+). Sinon, la fonction est juste un wrapper de CG_StraightSkeleton (cas le plus lent).

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

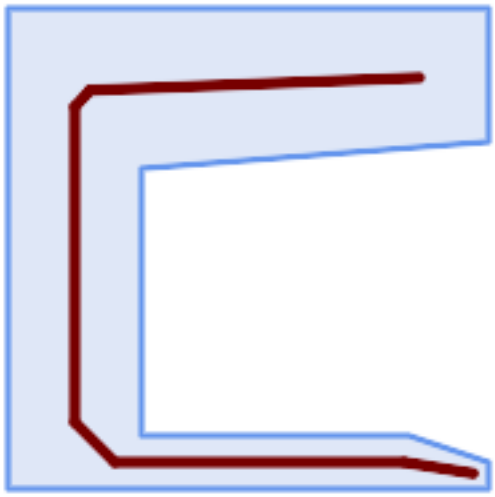
Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT CG_ApproximateMedialAxis(ST_GeomFromText('POLYGON (( 190 190, 10 190, 10 10, 190 10, ↩
     190 20, 160 30, 60 30, 60 130, 190 140, 190 190 ))'));
```



Un polygone et son axe médian approximatif

Voir aussi

CG_StraightSkeleton

8.3.21 ST_ConstrainedDelaunayTriangles

ST_ConstrainedDelaunayTriangles — Renvoie une triangulation de Delaunay contrainte autour de la géométrie d'entrée donnée.

Synopsis

geometry **ST_ConstrainedDelaunayTriangles**(geometry g1);

Description

Warning
ST_ConstrainedDelaunayTriangles is deprecated as of 3.5.0. Use CG_ConstrainedDelaunayTriangles instead.

Renvoie un Constrained Delaunay triangulation autour des sommets de la géométrie d'entrée. Le résultat est un TIN.

Cette méthode nécessite le backend SFCGAL.

Disponibilité : 3.0.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

8.3.22 CG_ConstrainedDelaunayTriangles

CG_ConstrainedDelaunayTriangles — Renvoie une triangulation de Delaunay contrainte autour de la géométrie d'entrée donnée.

Synopsis

geometry **CG_ConstrainedDelaunayTriangles**(geometry g1);

Description

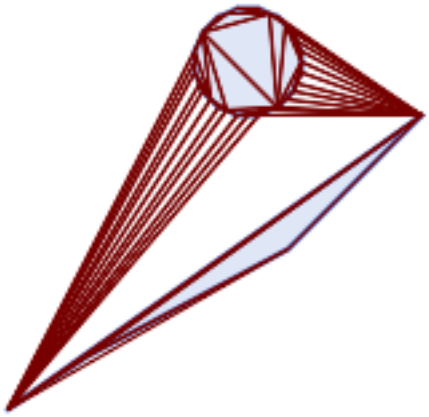
Renvoie un Constrained Delaunay triangulation autour des sommets de la géométrie d'entrée. Le résultat est un TIN.

Cette méthode nécessite le backend SFCGAL.

Disponibilité : 3.0.0

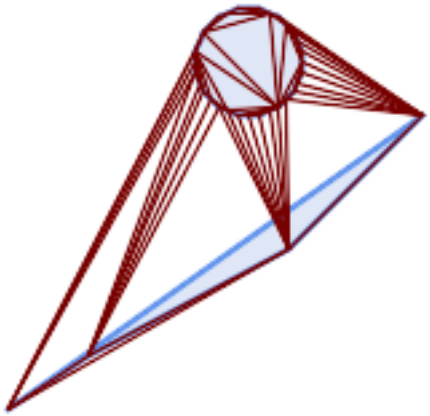
Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples



CG_ConstrainedDelaunayTriangles of 2 polygons

```
select CG_ConstrainedDelaunayTriangles(
    ST_Union(
    POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))'::geometry,
    ST_Buffer('POINT(110 170)'::geometry, 20)
    )
    )
    ;
```



ST_DelaunayTriangles de 2 polygones. Les arêtes des triangles traversent les limites des polygones.

```
select ST_DelaunayTriangles(
    ST_Union(
    POLYGON((175 150, 20 40, 50 60, 125 100, 175 150
    ST_Buffer('POINT(110 170)'::geometry, 20)
    )
    )
    ;
```

Voir aussi

ST_DelaunayTriangles, ST_TriangulatePolygon, CG_Tesselate, ST_ConcaveHull, ST_Dump

8.3.23 ST_Extrude

ST_Extrude — Extruder une surface vers un volume

Synopsis

geometry **ST_Extrude**(geometry geom, float x, float y, float z);

Description

Warning
ST_Extrude is deprecated as of 3.5.0. Use CG_Extrude instead.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.3.24 CG_Extrude

CG_Extrude — Extruder une surface vers un volume

Synopsis

geometry **CG_Extrude**(geometry geom, float x, float y, float z);

Description

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

Les images 3D ont été générées à l'aide de PostGIS ST_AsX3D et rendues en HTML à l'aide de X3Dom HTML Javascript rendering library.

```
SELECT ST_Buffer(ST_GeomFromText('POINT ↩
    (100 90)'),
                                    50, ' ↩
    quad_segs=2'),0,0,30);
```

Octogone original formé à partir du point tampon

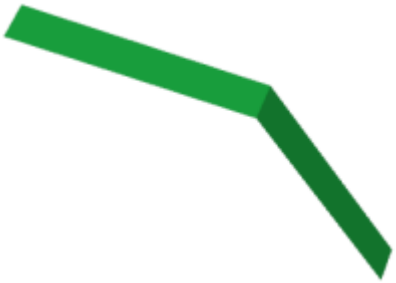
```
CG_Extrude(ST_Buffer(ST_GeomFromText(' ↩
    POINT(100 90)'),
                                    50, ' ↩
    quad_segs=2'),0,0,30);
```

Un hexagone extrudé de 30 unités le long de Z produit une PolyhedralSurfaceZ

```
SELECT ST_GeomFromText('LINESTRING(50 50, ↩
     100 90, 95 150)')
```

Ligne d'origine

```
SELECT CG_Extrude(
                                    ↩
    ST_GeomFromText('LINESTRING(50 50, 100 90, 95 15
```



LineString extrudé le long de Z produit une PolyhedralSurfaceZ

Voir aussi

ST_AsX3D, CG_ExtrudeStraightSkeleton

8.3.25 CG_ExtrudeStraightSkeleton

CG_ExtrudeStraightSkeleton — Extrusion de squelette droit

Synopsis

geometry **CG_ExtrudeStraightSkeleton**(geometry geom, float roof_height, float body_height = 0);

Description

Calcule une extrusion avec une hauteur maximale de la géométrie du polygone.

Note
Perhaps the first (historically) use-case of straight skeletons: given a polygonal roof, the straight skeleton directly gives the layout of each tent. If each skeleton edge is lifted from the plane a height equal to its offset distance, the resulting roof is "correct" in that water will always fall down to the contour edges (the roof's border), regardless of where it falls on the roof. The function computes this extrusion aka "roof" on a polygon. If the argument body_height > 0, so the polygon is extruded like with CG_Extrude(polygon, 0, 0, body_height). The result is an union of these polyhedralsurfaces.

Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0.

Nécessite SFCGAL >= 1.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples

```
SELECT ST_AsText(CG_ExtrudeStraightSkeleton('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))', 3.0, 2.0));
```

```
POLYHEDRALSURFACE Z (((0 0 0,0 4 0,4 4 0,4 5 0,5 5 0,5 0 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2  ↩
    0,1 1 0)),((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 5 2,4 5 0,4 4 0)),((4 5 0,4 5 2,5 5 2,5 5 0,4 5 0)),((5 5 0,5 5 2,5 0 2,5 0 0,5 5  ↩
    0)),((5 0 0,5 0 2,0 0 2,0 0 0,5 0 0)),((1 1 0,1 1 2,2 1 2,2 1 0,1 1 0)),((2 1 0,2 1 2,2  ↩
    2 2,2 2 0,2 1 0)),((2 2 0,2 2 2,1 2 2,1 2 0,2 2 0)),((1 2 0,1 2 2,1 1 2,1 1 0,1 2 0)) ↩
    ,((4 5 2,5 5 2,4 4 2,4 5 2)),((2 1 2,5 0 2,0 0 2,2 1 2)),((5 5 2,5 0 2,4 4 2,5 5 2)),((2 ↩
     1 2,0 0 2,1 1 2,2 1 2)),((1 2 2,1 1 2,0 0 2,1 2 2)),((0 4 2,2 2 2,1 2 2,0 4 2)),((0 4  ↩
    2,1 2 2,0 0 2,0 4 2)),((4 4 2,5 0 2,2 2 2,4 4 2)),((4 4 2,2 2 2,0 4 2,4 4 2)),((2 2 2,5  ↩
    0 2,2 1 2,2 2 2)),((0.5 2.5 2.5,0 0 2,0.5 0.5 2.5,0.5 2.5 2.5)),((1 3 3,0 4 2,0.5 2.5  ↩
    2.5,1 3 3)),((0.5 2.5 2.5,0 4 2,0 0 2,0.5 2.5 2.5)),((2.5 0.5 2.5,5 0 2,3.5 1.5 3.5,2.5  ↩
    0.5 2.5)),((0 0 2,5 0 2,2.5 0.5 2.5,0 0 2)),((0.5 0.5 2.5,0 0 2,2.5 0.5 2.5,0.5 0.5 2.5) ↩
    ),((4.5 3.5 2.5,5 5 2,4.5 4.5 2.5,4.5 3.5 2.5)),((3.5 2.5 3.5,3.5 1.5 3.5,4.5 3.5  ↩
    2.5,3.5 2.5 3.5)),((4.5 3.5 2.5,5 0 2,5 5 2,4.5 3.5 2.5)),((3.5 1.5 3.5,5 0 2,4.5 3.5  ↩
    2.5,3.5 1.5 3.5)),((5 5 2,4 5 2,4.5 4.5 2.5,5 5 2)),((4.5 4.5 2.5,4 4 2,4.5 3.5 2.5,4.5  ↩
    4.5 2.5)),((4.5 4.5 2.5,4 5 2,4 4 2,4.5 4.5 2.5)),((3 3 3,0 4 2,1 3 3,3 3 3)),((3.5 2.5  ↩
    3.5,4.5 3.5 2.5,3 3 3,3.5 2.5 3.5)),((3 3 3,4 4 2,0 4 2,3 3 3)),((4.5 3.5 2.5,4 4 2,3 3  ↩
    3,4.5 3.5 2.5)),((2 1 2,1 1 2,0.5 0.5 2.5,2 1 2)),((2.5 0.5 2.5,2 1 2,0.5 0.5 2.5,2.5  ↩
    0.5 2.5)),((1 1 2,1 2 2,0.5 2.5 2.5,1 1 2)),((0.5 0.5 2.5,1 1 2,0.5 2.5 2.5,0.5 0.5 2.5) ↩
    ),((1 3 3,2 2 2,3 3 3,1 3 3)),((0.5 2.5 2.5,1 2 2,1 3 3,0.5 2.5 2.5)),((1 3 3,1 2 2,2 2  ↩
    2,1 3 3)),((2 2 2,2 1 2,2.5 0.5 2.5,2 2 2)),((3.5 2.5 3.5,3 3 3,3.5 1.5 3.5,3.5 2.5 3.5) ↩
    ),((3.5 1.5 3.5,2 2 2,2.5 0.5 2.5,3.5 1.5 3.5)),((3 3 3,2 2 2,3.5 1.5 3.5,3 3 3)))
```

Voir aussi

ST_Extrude, CG_StraightSkeleton

8.3.26 CG_GreeneApproxConvexPartition

CG_GreeneApproxConvexPartition — Calcul de la partition convexe approximative de la géométrie du polygone

Synopsis

geometry **CG_GreeneApproxConvexPartition**(geometry geom);

Description

Calcule une partition convexe approximative monotone de la géométrie du polygone.

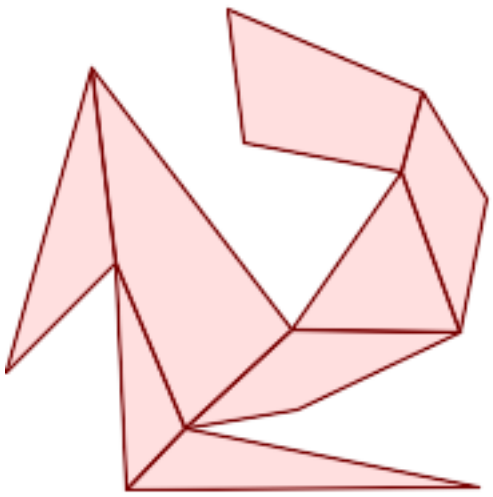
Note
Une partition d'un polygone P est un ensemble de polygones tels que les intérieurs des polygones ne se coupent pas et que l'union des polygones est égale à l'intérieur du polygone original P. Les fonctions CG_ApproxConvexPartition et CG_GreeneApproxConvexPartition produisent des partitions convexes approximativement optimales. Ces deux fonctions produisent des décompositions convexes en décomposant d'abord le polygone en polygones plus simples ; CG_ApproxConvexPartition utilise une triangulation et CG_GreeneApproxConvexPartition une partition monotone. Ces deux fonctions garantissent toutes deux qu'elles ne produiront pas plus de quatre fois le nombre optimal de pièces convexes, mais elles diffèrent par leur complexité d'exécution. Bien que l'algorithme d'approximation basé sur la triangulation produise souvent moins de pièces convexes, ce n'est pas toujours le cas.

Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0.

Nécessite SFCGAL >= 1.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples



Partition convexe approximative de Greene (même exemple que CG_YMonotonePartition, CG_ApproxConvexPartition et CG_OptimalConvexPartition)

```
SELECT ST_AsText(CG_GreeneApproxConvexPartition('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));
```

```
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)))
```

Voir aussi

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_OptimalConvexPartition

8.3.27 ST_MinkowskiSum

ST_MinkowskiSum — Effectue la somme de Minkowski

Synopsis

geometry **ST_MinkowskiSum**(geometry geom1, geometry geom2);

Description

Warning
ST_MinkowskiSum is deprecated as of 3.5.0. Use CG_MinkowskiSum instead.

Cette fonction effectue une somme de minkowski en 2D d'un point, d'une ligne ou d'un polygone avec un polygone.

Une somme de minkowski de deux géométries A et B est l'ensemble de tous les points qui sont la somme de n'importe quel point de A et B. Les sommes de minkowski sont souvent utilisées dans la planification des mouvements et la conception assistée par ordinateur. Plus de détails sur Wikipedia Addition de Minkowski.

Le premier paramètre peut être n'importe quelle géométrie 2D (point, ligne, polygone). Si une géométrie 3D est transmise, elle sera convertie en 2D en forçant Z à 0, ce qui peut entraîner des cas d'invalidité. Le second paramètre doit être un polygone 2D.

La mise en œuvre utilise CGAL 2D Minkowskisum.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

8.3.28 CG_MinkowskiSum

CG_MinkowskiSum — Effectue la somme de Minkowski

Synopsis

geometry **CG_MinkowskiSum**(geometry geom1, geometry geom2);

Description

Cette fonction effectue une somme de minkowski en 2D d'un point, d'une ligne ou d'un polygone avec un polygone.

Une somme de minkowski de deux géométries A et B est l'ensemble de tous les points qui sont la somme de n'importe quel point de A et B. Les sommes de minkowski sont souvent utilisées dans la planification des mouvements et la conception assistée par ordinateur. Plus de détails sur Wikipedia Addition de Minkowski.

Le premier paramètre peut être n'importe quelle géométrie 2D (point, ligne, polygone). Si une géométrie 3D est transmise, elle sera convertie en 2D en forçant Z à 0, ce qui peut entraîner des cas d'invalidité. Le second paramètre doit être un polygone 2D.

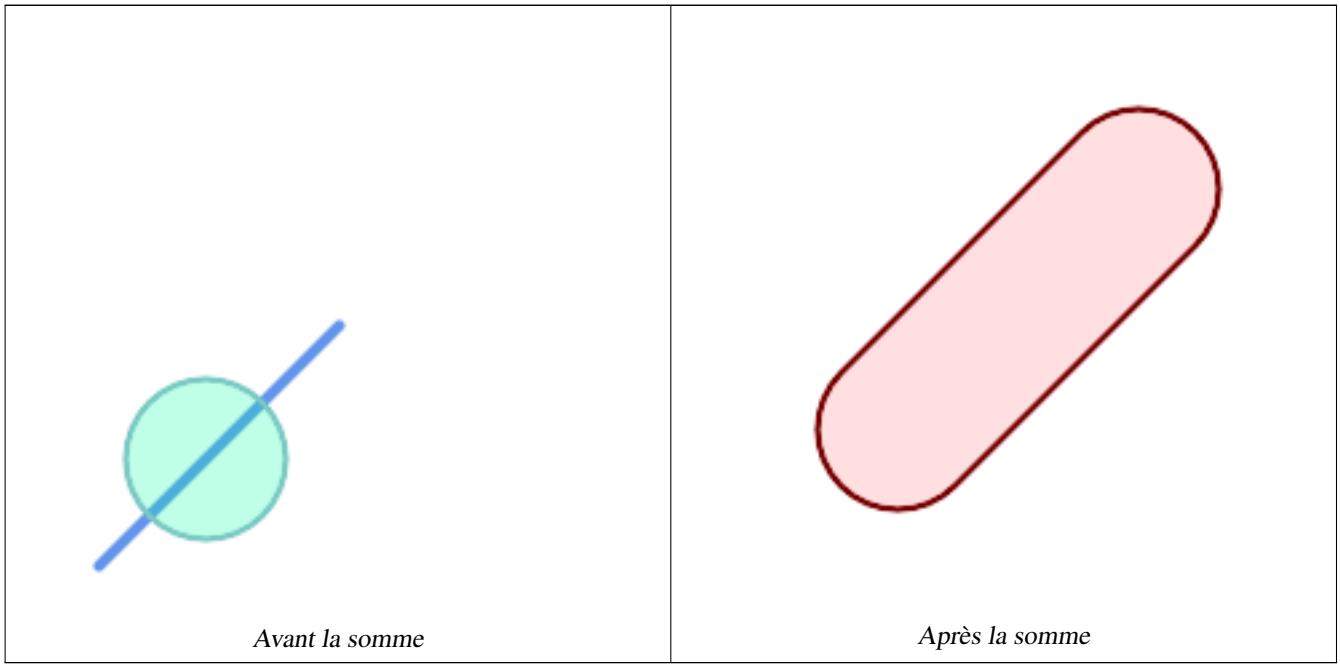
La mise en œuvre utilise CGAL 2D Minkowskisum.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples

Somme de Minkowski d'une ligne et d'un cercle polygonal où la ligne traverse le cercle

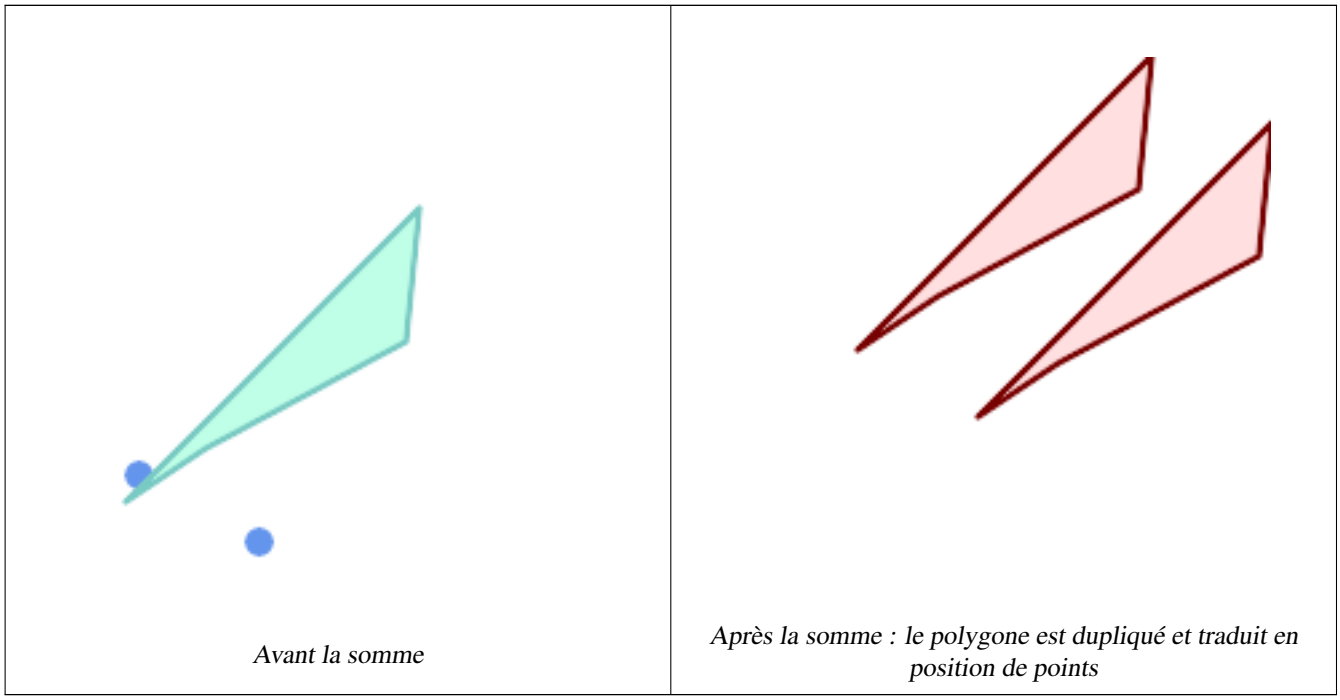


Avant la somme | *Après la somme*

```
SELECT CG_MinkowskiSum(line, circle))
            FROM (SELECT
            ST_MakeLine(ST_Point(10, 10),ST_Point(100, 100)) As line,
            ST_Buffer(ST_GeomFromText('POINT(50 50)'), 30) As circle) As foo;

            -- wkt --
            MULTIPOLYGON(((30 59.9999999999999,30.5764415879031  ↩
                54.1472903395161,32.2836140246614 48.5194970290472,35.0559116309237  ↩
                43.3328930094119,38.7867965644036 38.7867965644035,43.332893009412  ↩
                35.0559116309236,48.5194970290474 32.2836140246614,54.1472903395162  ↩
                30.5764415879031,60.0000000000001 30,65.8527096604839  ↩
                30.5764415879031,71.4805029709527 32.2836140246614,76.6671069905881  ↩
                35.0559116309237,81.2132034355964 38.7867965644036,171.213203435596  ↩
                128.786796564404,174.944088369076 133.332893009412,177.716385975339  ↩
                138.519497029047,179.423558412097 144.147290339516,180 150,179.423558412097  ↩
                155.852709660484,177.716385975339 161.480502970953,174.944088369076  ↩
```

```
                166.667106990588,171.213203435596 171.213203435596,166.667106990588  ↩
                174.944088369076,
            161.480502970953 177.716385975339,155.852709660484 179.423558412097,150  ↩
                180,144.147290339516 179.423558412097,138.519497029047  ↩
                177.716385975339,133.332893009412 174.944088369076,128.786796564403  ↩
                171.213203435596,38.7867965644035 81.2132034355963,35.0559116309236  ↩
                76.667106990588,32.2836140246614 71.4805029709526,30.5764415879031  ↩
                65.8527096604838,30 59.9999999999999)))
```

Somme de Minkowski d'un polygone et d'un multipoint



Avant la somme

Après la somme : le polygone est dupliqué et traduit en position de points

```
SELECT CG_MinkowskiSum(mp, poly)
        FROM (SELECT 'MULTIPOINT(25 50,70 25)'::geometry As mp,
        'POLYGON((130 150, 20 40, 50 60, 125 100, 130 150))'::geometry As poly
        ) As foo


        -- wkt --
        MULTIPOLYGON(
        ((70 115,100 135,175 175,225 225,70 115)),
        ((120 65,150 85,225 125,275 175,120 65))
        )
```

8.3.29 ST_OptimalAlphaShape

ST_OptimalAlphaShape — Calcule une forme alpha entourant une géométrie en utilisant une valeur alpha "optimale".

Synopsis

geometry **ST_OptimalAlphaShape**(geometry geom, boolean allow_holes = false, integer nb_components = 1);

Description

Warning
ST_OptimalAlphaShape is deprecated as of 3.5.0. Use CG_OptimalAlphaShape instead.

Calcule la forme alpha "optimale" des points d'une géométrie. La forme alpha est calculée en utilisant une valeur de α choisie de telle sorte que :

1. le nombre d'éléments du polygone est égal ou inférieur à `nb_components` (qui vaut 1 par défaut)
2. tous les points d'entrée sont contenus dans la forme

Le résultat ne contiendra pas de trous à moins que l'argument facultatif `allow_holes` ne soit spécifié comme true.

Disponibilité : 3.3.0 - nécessite SFCGAL >= 1.4.1.

Cette méthode nécessite le backend SFCGAL.

8.3.30 CG_OptimalAlphaShape

CG_OptimalAlphaShape — Calcule une forme alpha entourant une géométrie en utilisant une valeur alpha "optimale".

Synopsis

geometry **CG_OptimalAlphaShape**(geometry geom, boolean allow_holes = false, integer nb_components = 1);

Description

Calcule la forme alpha "optimale" des points d'une géométrie. La forme alpha est calculée en utilisant une valeur de α choisie de telle sorte que :

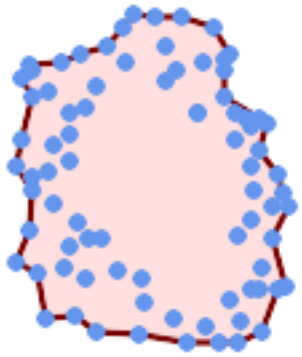
1. le nombre d'éléments du polygone est égal ou inférieur à `nb_components` (qui vaut 1 par défaut)
2. tous les points d'entrée sont contenus dans la forme

Le résultat ne contiendra pas de trous à moins que l'argument facultatif `allow_holes` ne soit spécifié comme true.

Availability: 3.5.0 - requires SFCGAL >= 1.4.1.

Cette méthode nécessite le backend SFCGAL.

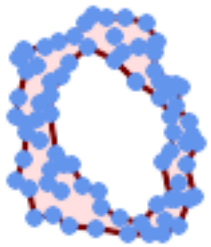
Exemples



Optimal alpha-shape of a MultiPoint (same example as CG_AlphaShape)

```
SELECT ST_AsText(CG_OptimalAlphaShape('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));
```

```
POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,37 23,30 22,28  ↩
    33,23 36,
            26 44,27 54,23 60,24 67,27 77,24 82,26 85,34 86,39 88,45 90,49 95,52 98,57  ↩
                97,64 97,72 95,76 88,75 84,75 77,83 72,85 71,83 64,88 58,89 53))
```



Optimal alpha-shape of a MultiPoint, allowing holes (same example as CG_AlphaShape)

```
SELECT ST_AsText(CG_OptimalAlphaShape('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, allow_holes => true));
```

```
POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,37 23,30 22,28  ↩
    33,23 36,26 44,27 54,23 60,24 67,27 77,24 82,26 85,34 86,39 88,45 90,49 95,52 98,57  ↩
    97,64 97,72 95,76 88,75 84,75 77,83 72,85 71,83 64,88 58,89 53),(36 61,36 68,40 75,43  ↩
    80,50 86,60 81,68 73,77 67,81 60,82 54,81 47,78 43,81 29,76 27,70 20,62 22,55 26,54  ↩
    32,48 34,44 42,38 46,36 61))
```

Voir aussi

ST_ConcaveHull, CG_AlphaShape

8.3.31 CG_OptimalConvexPartition

CG_OptimalConvexPartition — Calcul d'une partition convexe optimale de la géométrie du polygone

Synopsis

geometry **CG_OptimalConvexPartition**(geometry geom);

Description

Calcul d'une partition convexe optimale de la géométrie du polygone.

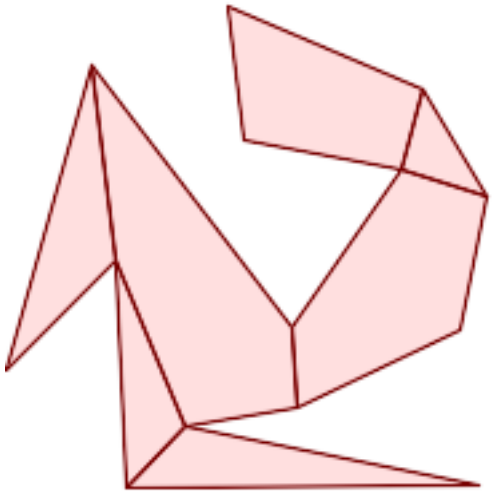
Note
Une partition d'un polygone P est un ensemble de polygones tels que les intérieurs des polygones ne se croisent pas et que l'union des polygones est égale à l'intérieur du polygone original P. CG_OptimalConvexPartition produit une partition optimale en termes de nombre de morceaux.

Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0.

Nécessite SFCGAL >= 1.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples



Partition convexe optimale (même exemple que CG_YMonotonePartition, CG_ApproxConvexPartition et CG_GreeneApproxConvexPartition)

```
SELECT ST_AsText(CG_OptimalConvexPartition('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));
```

```
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)))
```

Voir aussi

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition

8.3.32 CG_StraightSkeleton

CG_StraightSkeleton — Calcule un squelette (straight skeleton) à partir d'une géométrie

Synopsis

geometry **CG_StraightSkeleton**(geometry geom, boolean use_distance_as_m = false);

Description

Disponibilité : 3.5.0

Nécessite SFCGAL >= 1.3.8 pour l'option use_distance_as_m

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

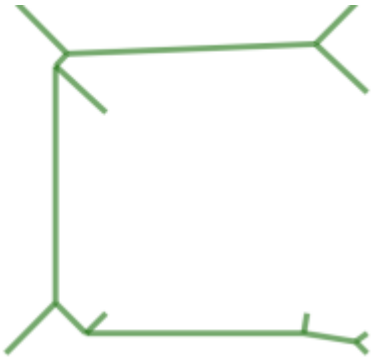
Exemples

```
SELECT CG_StraightSkeleton(ST_GeomFromText('POLYGON (( 190 190, 10 190, 10 10, 190 10, 190 ↩
    20, 160 30, 60 30, 60 130, 190 140, 190 190 ))'));
```

```
ST_AsText(CG_StraightSkeleton('POLYGON((0 0,1 0,1 1,0 1,0 0))', true);
```

```
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))
```

Polygone d'origine



Squelette d'un polygone

Voir aussi

CG_ExtrudeStraightSkeleton

8.3.33 ST_StraightSkeleton

ST_StraightSkeleton — Calcule un squelette (straight skeleton) à partir d'une géométrie

Synopsis

geometry **ST_StraightSkeleton**(geometry geom);

Description

Warning
ST_StraightSkeleton is deprecated as of 3.5.0. Use CG_StraightSkeleton instead.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

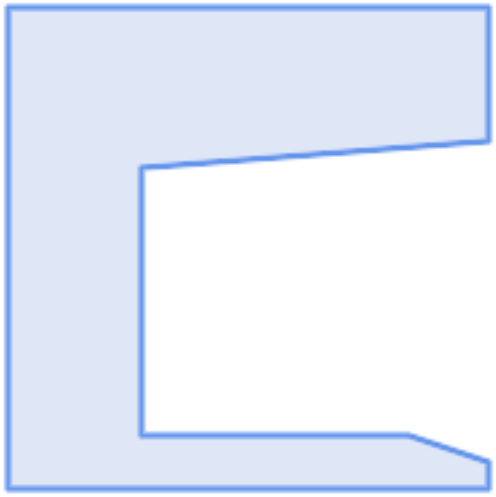
Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

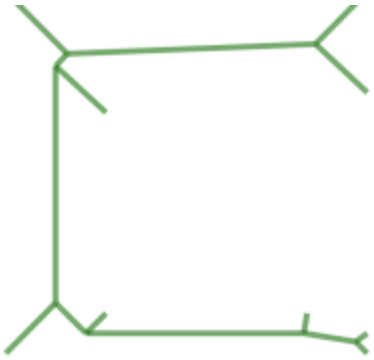
Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_StraightSkeleton(ST_GeomFromText('POLYGON (( 190 190, 10 190, 10 10, 190 10, 190 ↩
    20, 160 30, 60 30, 60 130, 190 140, 190 190 ))'));
```



Polygone d'origine



Squelette d'un polygone

Voir aussi

CG_ExtrudeStraightSkeleton

8.3.34 ST_Tesselate

ST_Tesselate — Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS

Synopsis

geometry **ST_Tesselate**(geometry geom);

Description

Warning
ST_Tesselate is deprecated as of 3.5.0. Use CG_Tesselate instead.

Prend en entrée une surface telle que MULTI(POLYGON) ou POLYHEDRALSURFACE et renvoie une représentation TIN via le processus de tessellation à l'aide de triangles.

Note
ST_TriangulatePolygon est similaire à cette fonction, sauf qu'elle renvoie une collection géométrique de polygones au lieu d'un TIN et qu'elle ne fonctionne qu'avec des géométries 2D.

Disponibilité : 2.1.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

8.3.35 CG_Tesselate

CG_Tesselate — Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS

Synopsis

geometry **CG_Tesselate**(geometry geom);

Description

Prend en entrée une surface telle que MULTI(POLYGON) ou POLYHEDRALSURFACE et renvoie une représentation TIN via le processus de tessellation à l'aide de triangles.

Note
ST_TriangulatePolygon est similaire à cette fonction, sauf qu'elle renvoie une collection géométrique de polygones au lieu d'un TIN et qu'elle ne fonctionne qu'avec des géométries 2D.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

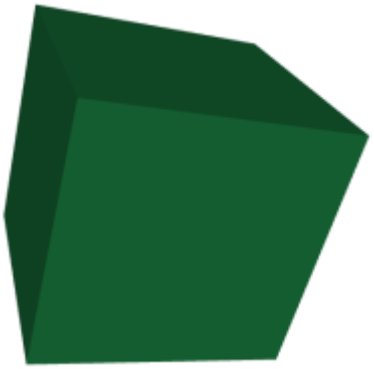
Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT ST_GeomFromText('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
                                             ((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
```



Cube d'origine

```
SELECT CG_Tesselate(ST_GeomFromText('
    POLYHEDRALSURFACE Z( ((0 0 0, 0 0 1, 0 1 1, 0 1
                                             ((0 0 0,
    0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1
                                             ((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
```

Sortie ST_AsText :

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

Cube tessellé avec triangles colorés

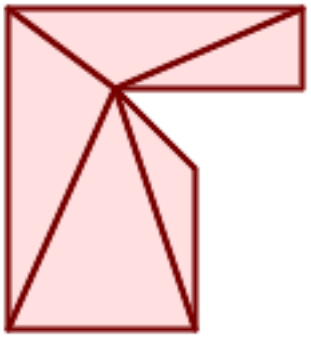
```
SELECT 'POLYGON (( 10 190, 10 70, 80 70,  ↩
    80 130, 50 160, 120 160, 120 190, 10 190 ))'::geometry;
```

Polygone d'origine

```
SELECT
                        CG_Tesselate(' ↩
    POLYGON (( 10 190, 10 70, 80 70, 80 130, 50 160,
    ;
```

Sortie ST_AsText

```
TIN(((80 130,50 160,80 70,80 130)),((50  ↩
    160,10 190,10 70,50 160)),
                ((80 70,50 160,10 70,80  ↩
    70)),((120 160,120 190,50 160,120 160)),
                ((120 190,10 190,50  ↩
    160,120 190)))
```



Polygone tessellé

Voir aussi

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.36 CG_Triangulate

CG_Triangulate — Triangule une géométrie polygonale

Synopsis

geometry **CG_Triangulate**(geometry geom);

Description

Triangule une géométrie polygonale.

Réalisé par le module SFCGAL

Note

NOTE : cette fonction renvoie une géométrie représentant le résultat de la triangulation.

Disponibilité : 3.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples de géométrie

```
SELECT CG_Triangulate('POLYGON((0.0 0.0,1.0 0.0,1.0 1.0,0.0 1.0,0.0 0.0),(0.2 0.2,0.2 ↩
    0.8,0.8 0.8,0.8 0.2,0.2 0.2))');
               cg_triangulate
               ----------------
               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)))

               (1 row)
```

Voir aussi

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.37 CG_Visibility

CG_Visibility — Calculer un polygone de visibilité à partir d'un point ou d'un segment dans une géométrie polygonale

Synopsis

geometry **CG_Visibility**(geometry polygon, geometry point);
geometry **CG_Visibility**(geometry polygon, geometry pointA, geometry pointB);

Description

Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0.

Nécessite SFCGAL >= 1.5.0

Cette méthode nécessite le backend SFCGAL.

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

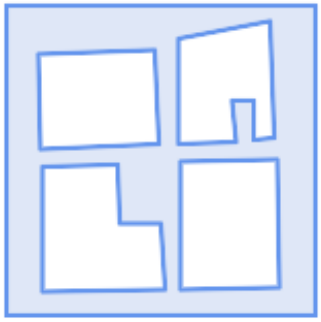
Cette fonction prend en charge les surfaces Polyhedral.

Cette fonction prend en charge les triangles et les réseaux irréguliers triangulés (TIN).

Exemples

```
SELECT CG_Visibility('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, 'POINT(91 ↩
    87)'::geometry);
```

```
SELECT CG_Visibility('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,'POINT(78.5 ↩
     68)'::geometry, 'POINT(98.5 68)'::geometry);
```

		
Polygone d'origine	*Visibilité depuis le point*	*Visibilité depuis le segment*

8.3.38 CG_YMonotonePartition

CG_YMonotonePartition — Calcul de la partition y-monotone de la géométrie du polygone

Synopsis

geometry **CG_YMonotonePartition**(geometry geom);

Description

Calcul de la partition y-monotone de la géométrie du polygone.

Note
Une partition d'un polygone P est un ensemble de polygones tels que les intérieurs des polygones ne se croisent pas et que l'union des polygones est égale à l'intérieur du polygone original P. Un polygone y-monotone est un polygone dont les sommets v1,...,vn peuvent être divisés en deux chaînes v1,...,vk et vk,...,vn,v1, de sorte que toute ligne horizontale croise l'une ou l'autre chaîne au plus une fois. Cet algorithme ne garantit pas de limite au nombre de polygones produits par rapport au nombre optimal.

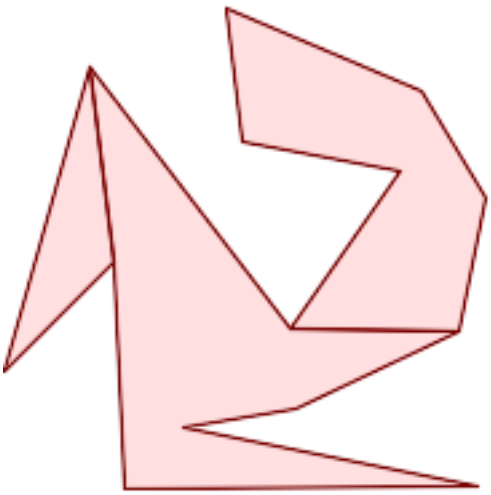
Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0.

Nécessite SFCGAL >= 1.5.0

Cette méthode nécessite le backend SFCGAL.

Exemples

Polygone d'origine



Partition Y-Monotone (même exemple que CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition et CG_OptimalConvexPartition)

```
SELECT ST_AsText(CG_YMonotonePartition('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));
```

```
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)))
```

Voir aussi

CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

Chapter 9

Topologie

Les types et fonctions topologiques de PostGIS sont utilisés pour gérer les objets topologiques tels que les faces, les arêtes et les des noeuds.

La présentation de Sandro Santilli à la conférence PostGIS Day Paris 2011 donne un bon aperçu de PostGIS Topology et de son évolution Topology with PostGIS 2.0 slide deck.

Vincent Picavet fournit un bon résumé et une vue d'ensemble de ce qu'est la topologie, comment elle est utilisée, et divers outils FOSS4G qui la supportent dans PostGIS Topology PGConf EU 2012.

Un exemple de base de données SIG topologique est la base de données US Census Topologically Integrated Geographic Encoding and Referencing System (TIGER). Si vous souhaitez expérimenter la topologie PostGIS et avez besoin de quelques données, consultez Topology_Load_Tiger.

Le module sur la topologie de PostGIS existe dans les versions précédentes de PostGIS mais n'a jamais fait partie de la documentation officielle de PostGIS. Dans la version 2.0.0 de PostGIS, un grand nettoyage est en cours pour en éliminer l'utilisation de toutes les fonctions obsolètes, résoudre les problèmes d'utilisabilité connus, mieux documenter les caractéristiques et les fonctions, ajouter de nouvelles fonctions, et l'améliorer afin de mieux se conformer aux normes SQL-MM.

Les détails sur ce projet peuvent être trouvés à PostGIS Topology Wiki

Toutes les fonctions et toutes les tables associées à ce module sont installées dans un schéma appelé `topology`.

Les fonctions qui sont définies dans le standard SQL/MM sont préfixées par ST_ et les fonctions spécifiques à PostGIS ne sont pas préfixées.

Le support topologique est compilé par défaut à partir de PostGIS 2.0, et peut être désactivé en spécifiant l'option de configuration --without-topology au moment de la construction, comme décrit dans Chapter 2

9.1 Les types associés à "Topology"

9.1.1 getfaceedges_returntype

getfaceedges_returntype — Type composite composé d'un numéro de séquence et d'un numéro d'arête.

Description

Type composite composé d'un numéro de séquence et d'un numéro d'arête. Il s'agit du type de retour pour les fonctions `ST_GetFaceEdges` et `GetNodeEdges`.

1. `sequence` est un entier. Il fait référence à un élément topologique défini dans la table topology.topology qui définit le schéma de la topologie et le srid.
2. `edge` est un entier : c'est l'identifiant d'une arête.

9.1.2 TopoGeometry

TopoGeometry — Un type composite représentant une géométrie topologiquement définie.

Description

Un type composite qui fait référence à une géométrie topologique dans une couche topologique spécifique, ayant un type et un identifiant spécifiques. Les éléments d'une TopoGeometry ont les propriétés suivantes : topology_id, layer_id, id integer, type integer.

1. `topology_id` est un entier : il fait référence à un élément topologique défini dans la table topology.topology qui définit le schéma de la topologie et le srid.
2. `layer_id` est un entier : L'identifiant de la couche dans la table des couches à laquelle la TopoGeometry appartient. La combinaison de topology_id, layer_id fournit une référence unique dans la table topology.layers.
3. `id` est un nombre entier : L'id est le numéro de séquence autogénéré qui définit de manière unique la topogeometry dans la couche topologique correspondante.
4. `type` un entier entre 1 et 4 qui définit le type de la géométrie : 1 : [multi]point, 2 : [multi]line, 3 : [multi]poly, 4 : collection

Transtypages

Cette section liste les transtypages autorisés pour ce type de donnée, qu'ils soient automatiques ou bien explicites

Transtypage vers	Comportement
geometry	automatique

Voir aussi

CreateTopoGeom

9.1.3 validatetopology_returntype

validatetopology_returntype — Un type composite composé d'un message d'erreur et de id1 et id2 pour indiquer l'emplacement de l'erreur. Il s'agit du type de retour pour `ValidateTopology`.

Description

Un type composite composé d'un message d'erreur et de deux nombres entiers. La fonction ValidateTopology renvoie un ensemble d'entiers pour indiquer les erreurs de validation et les id1 et id2 pour indiquer les id des objets topologiques concernés par l'erreur.

1. `error` est une variable : Indique le type d'erreur.

   Les descripteurs d'erreurs actuels sont les suivants : nœuds coïncidents, arête traversant un nœud, arête non simple, mauvaise correspondance de la géométrie de l'extrémité du nœud, mauvaise correspondance de la géométrie du début du nœud, face chevauchant la face, face à l'intérieur de la face,
2. `id1` est un nombre entier : Indique l'identifiant de l'arête / de la face / des nœuds en erreur.
3. `id2` est un nombre entier : Pour les erreurs qui impliquent 2 objets, indique l'arête / ou le nœud secondaire

Voir aussi

ValidateTopology

9.2 Domaines de topologie

9.2.1 TopoElement

TopoElement — Un tableau de 2 entiers généralement utilisé pour identifier un composant TopoGeometry.

Description

Un tableau de 2 entiers utilisé pour représenter une composante d'une TopoGeometry simple ou hiérarchique.

Dans le cas d'une TopoGeometry simple, le premier élément du tableau représente l'identifiant d'une primitive topologique et le deuxième élément représente son type (1:nœud, 2:arête, 3:face). Dans le cas d'une TopoGeometry hiérarchique, le premier élément du tableau représente l'identifiant d'une TopoGeometry enfant et le deuxième élément représente son identifiant de couche.

Note
Pour une TopoGeometry hiérarchique donnée, tous les éléments TopoGeometry enfants proviendront de la même couche enfant, comme spécifié dans l'enregistrement topology.layer pour la couche de la TopoGeometry en cours de définition.

Exemples

```
SELECT te[1] AS id, te[2] AS type FROM
( SELECT ARRAY[1,2]::topology.topoelement AS te ) f;
 id | type
----+------
  1 |    2
```

```
SELECT ARRAY[1,2]::topology.topoelement;
  te
-------
 {1,2}
```

```
--Example of what happens when you try to case a 3 element array to topoelement
-- NOTE: topoement has to be a 2 element array so fails dimension check
SELECT ARRAY[1,2,3]::topology.topoelement;
ERROR:  value for domain topology.topoelement violates check constraint "dimensions"
```

Voir aussi

GetTopoGeomElements, TopoElementArray, TopoGeometry, TopoGeom_addElement, TopoGeom_remElement

9.2.2 TopoElementArray

TopoElementArray — Un tableau d'objets TopoElement.

Description

Un tableau de 1 ou plusieurs objets TopoElement, généralement utilisé pour transmettre des composants d'objets TopoGeometry.

Exemples

```
SELECT '{{1,2},{4,3}}'::topology.topoelementarray As tea;
  tea
-------
{{1,2},{4,3}}

-- more verbose equivalent --
SELECT ARRAY[ARRAY[1,2], ARRAY[4,3]]::topology.topoelementarray As tea;

  tea
-------
{{1,2},{4,3}}

--using the array agg function packaged with topology --
SELECT topology.TopoElementArray_Agg(ARRAY[e,t]) As tea
  FROM generate_series(1,4) As e CROSS JOIN generate_series(1,3) As t;
  tea
---------------------------------------------------------------------------
{{1,1},{1,2},{1,3},{2,1},{2,2},{2,3},{3,1},{3,2},{3,3},{4,1},{4,2},{4,3}}
```

```
SELECT '{{1,2,4},{3,4,5}}'::topology.topoelementarray As tea;
ERROR:  value for domain topology.topoelementarray violates check constraint "dimensions"
```

Voir aussi

TopoElement, GetTopoGeomElementArray, TopoElementArray_Agg

9.3 Gestion de la topologie et de TopoGeometry

9.3.1 AddTopoGeometryColumn

AddTopoGeometryColumn — Ajoute une colonne topogeometry à une table existante, enregistre cette nouvelle colonne en tant que couche dans topology.layer et renvoie le nouveau numéro d'identification de la couche.

Synopsis

integer **AddTopoGeometryColumn**(varchar topology_name, varchar schema_name, varchar table_name, varchar column_name, varchar feature_type);
integer **AddTopoGeometryColumn**(varchar topology_name, varchar schema_name, varchar table_name, varchar column_name, varchar feature_type, integer child_layer);

Description

Chaque objet TopoGeometry appartient à une couche spécifique d'une topologie spécifique. Avant de créer un objet TopoGeometry, il faut créer sa couche topologique. Une couche topologique est une association d'une table de caractéristiques avec la topologie. Elle contient également des informations sur le type et la hiérarchie. Nous créons une couche à l'aide de la fonction AddTopoGeometryColumn() :

Cette fonction ajoutera la colonne demandée à la table et ajoutera un enregistrement à la table topology.layer avec toutes les informations données.

Si vous ne spécifiez pas [child_layer] (ou si vous lui attribuez la valeur NULL), cette couche contiendra des TopoGeometries basiques (composées d'éléments topologiques primitifs). Sinon, cette couche contiendra des TopoGeometries hiérarchiques (composées de TopoGéométries de la couche_enfant).

Une fois la couche créée (son identifiant est renvoyé par la fonction AddTopoGeometryColumn), vous êtes prêt à y construire des objets TopoGeometry

Les `feature_type` valides sont : POINT, MULTIPOINT, LINE, MULTILINE, POLYGON, MULTIPOLYGON, COLLECTION

DIsponibilité : 1.1

Exemples

```
-- Note for this example we created our new table in the ma_topo schema
-- though we could have created it in a different schema -- in which case topology_name and ↩
     schema_name would be different
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');
```

```
CREATE SCHEMA ri;
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);
SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

Voir aussi

DropTopoGeometryColumn, toTopoGeom, CreateTopology, CreateTopoGeom

9.3.2 RenameTopoGeometryColumn

RenameTopoGeometryColumn — Renomme une colonne topogeometry

Synopsis

topology.layer **RenameTopoGeometryColumn**(regclass layer_table, name feature_column, name new_name);

Description

Cette fonction modifie le nom d'une colonne TopoGeometry existante en veillant à ce que les informations de métadonnées la concernant soient mises à jour en conséquence.

Disponibilité : 3.4.0

Exemples

```
SELECT topology.RenameTopoGeometryColumn('public.parcels', 'topogeom', 'tgeom');
```

Voir aussi

AddTopoGeometryColumn, RenameTopology

9.3.3 DropTopology

DropTopology — À utiliser avec précaution : Abandonne un schéma de topologie et supprime sa référence dans la table topology.topology et les références aux tables de ce schéma dans la table geometry_columns.

Synopsis

integer **DropTopology**(varchar topology_schema_name);

Description

Abandonne un schéma topologique et supprime sa référence dans la table topology.topology et les références aux tables de ce schéma dans la table geometry_columns. Cette fonction doit être utilisée avec précaution, car elle peut détruire des données qui vous sont chères. Si le schéma n'existe pas, elle supprime simplement les entrées de référence du schéma nommé.

DIsponibilité : 1.1

Exemples

Cascade abandonne le schéma ma_topo et supprime toutes les références à ce schéma dans topology.topology et geometry_columns.

```
SELECT topology.DropTopology('ma_topo');
```

Voir aussi

DropTopoGeometryColumn

9.3.4 RenameTopology

RenameTopology — Renomme une topologie

Synopsis

varchar **RenameTopology**(varchar old_name, varchar new_name);

Description

Renomme un schéma topologique, en mettant à jour son enregistrement de métadonnées dans la table `topology.topology`.

Disponibilité : 3.4.0

Exemples

Renommer une topologie de `topo_stage` à `topo_prod`.

```
SELECT topology.RenameTopology('topo_stage', 'topo_prod');
```

Voir aussi

CopyTopology, RenameTopoGeometryColumn

9.3.5 DropTopoGeometryColumn

DropTopoGeometryColumn — Supprime la colonne topogeometry de la table nommée `table_name` dans le schéma `schema_name` et désenregistre les colonnes de la table topology.layer.

Synopsis

text **DropTopoGeometryColumn**(varchar schema_name, varchar table_name, varchar column_name);

Description

Supprime la colonne topogeometry de la table nommée `table_name` dans le schéma `schema_name` et désenregistre les colonnes de la table topology.layer. Renvoie un résumé de l'état d'abandon. NOTE : il définit d'abord toutes les valeurs à NULL avant de procéder à l'abandon afin de contourner les contrôles d'intégrité référentielle.

DIsponibilité : 1.1

Exemples

```
SELECT topology.DropTopoGeometryColumn('ma_topo', 'parcel_topo', 'topo');
```

Voir aussi

AddTopoGeometryColumn

9.3.6 Populate_Topology_Layer

Populate_Topology_Layer — Ajoute les entrées manquantes à la table topology.layer en lisant les métadonnées des tables topo.

Synopsis

setof record **Populate_Topology_Layer**();

Description

Ajoute les entrées manquantes à la table `topology.layer` en inspectant les contraintes topologiques sur les tables. Cette fonction est utile pour corriger les entrées dans le catalogue topologique après la restauration de schémas contenant des données topographiques.

Elle renvoie la liste des entrées créées. Les colonnes retournées sont `schema_name`, `table_name`, `feature_column`.

Disponibilité : 2.3.0

Exemples

```
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');
-- this will return no records because this feature is already registered
SELECT *
  FROM topology.Populate_Topology_Layer();
```

```
-- let's rebuild
TRUNCATE TABLE topology.layer;

SELECT *
  FROM topology.Populate_Topology_Layer();

SELECT topology_id,layer_id, schema_name As sn, table_name As tn, feature_column As fc
FROM topology.layer;
```

```
schema_name | table_name | feature_column
-------------+------------+----------------
 strk        | parcels    | topo
(1 row)

 topology_id | layer_id |  sn  |   tn    |  fc
-------------+----------+------+---------+------
           2 |        2 | strk | parcels | topo
(1 row)
```

Voir aussi

AddTopoGeometryColumn

9.3.7 TopologySummary

TopologySummary — Prend un nom de topologie et fournit des totaux récapitulatifs des types d'objets dans la topologie.

Synopsis

text **TopologySummary**(varchar topology_schema_name);

Description

Prend un nom de topologie et fournit des totaux récapitulatifs des types d'objets dans la topologie.

Disponibilité : 2.0.0

Exemples

```
SELECT topology.topologysummary('city_data');
                    topologysummary
-----------------------------------------------------------
 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
```

Voir aussi

Topology_Load_Tiger

9.3.8 ValidateTopology

ValidateTopology — Renvoie un ensemble d'objets validatetopology_returntype détaillant les problèmes liés à la topologie.

Synopsis

setof validatetopology_returntype **ValidateTopology**(varchar toponame, geometry bbox);

Description

Renvoie un ensemble d'objets validatetopology_returntype détaillant les problèmes de topologie, en limitant éventuellement la vérification à la zone spécifiée par le paramètre `bbox`.

La liste des erreurs possibles, ce qu'elles signifient et ce que représentent les identifiants renvoyés est affichée ci-dessous :

Erreur	id1	id2	Signification
nœuds coïncidents	Identifiant du premier nœud.	Identifiant du second nœud.	Deux nœuds ont la même géométrie.
l'arête croise le nœud	Identifiant de l'arête.	Identifiant du nœud.	Une arête a un nœud à l'intérieur. Voir ST_Relate.
arête invalide	Identifiant de l'arête.		La géométrie d'une arête n'est pas valide. Voir ST_IsValid.
l'arête n'est pas simple	Identifiant de l'arête.		Une géométrie d'arête possède des auto-intersections. Voir ST_IsSimple.
une arête croise une arête	Identifiant de la première arête.	Identifiant de la deuxième arête.	Deux arêtes ont une intersection intérieure. Voir ST_Relate.
arête début nœud géométrie inadéquate	Identifiant de l'arête.	Identifiant du nœud de départ indiqué.	La géométrie du nœud indiqué comme nœud de départ d'une arête ne correspond pas au premier point de la géométrie de l'arête. Voir ST_StartPoint.
arête extrémité nœud géométrie non conforme	Identifiant de l'arête.	Identifiant du nœud final indiqué.	La géométrie du nœud indiqué comme nœud final d'une arête ne correspond pas au dernier point de la géométrie de l'arête. Voir ST_EndPoint.
face sans arêtes	Identifiant de la face orpheline.		Aucune arête ne signale une face existante sur l'un ou l'autre de ses côtés (face_gauche, face_droite).
la face n'a pas d'anneaux	Identifiant de la face partiellement définie.		Les arêtes rapportant une face sur leurs côtés ne forment pas un anneau.

Erreur	id1	id2	Signification
la face n'a pas le bon mbr (minimum bounding rectangle)	Identifiant de la face avec un cache mbr (minimum bounding rectangle) erroné.		Le rectangle minimal de délimitation d'une face ne correspond pas au rectangle minimal de délimitation de la collection d'arêtes rapportant la face sur leurs côtés.
le trou n'est pas dans la face annoncée	Identifiant signé d'une arête, identifiant l'anneau. Voir GetRingEdges.		Un anneau d'arêtes rapportant une face sur son extérieur est contenu dans une face différente.
le nœud non isolé a une face non-contenante	Identifiant du nœud mal défini.		Un nœud signalé comme étant à la limite d'une ou plusieurs arêtes indique une face contenante.
le nœud isolé a une face contenante	Identifiant du nœud mal défini.		Un nœud qui n'est pas signalé comme étant à la limite d'une arête n'a pas l'indication d'une face contenante.
Le nœud isolé n'a pas la bonne face contenante	Identifiant du nœud mal représenté.		Un nœud qui n'est pas signalé comme étant à la limite d'une arête indique une face contenante qui n'est pas la face réelle qui le contient. Voir GetFaceContainingPoint.
next_right_edge invalide	Identifiant de l'arête mal représentée.	Identifiant signé de l'arête qui doit être indiqué comme la prochaine arête droite.	L'arête indiquée comme le prochaine arête rencontrée en évoluant sur le côté droit d'une arête est faux.
next_left_edge invalide	Identifiant de l'arête mal représentée.	Identifiant signé de l'arête qui doit être indiquée comme l'arête gauche suivante.	L'arête indiquée comme la prochaine arête rencontrée en évoluant sur le côté gauche d'une arête est faux.
étiquetage mixte des faces dans l'anneau	Identifiant signé d'une arête, identifiant l'anneau. Voir GetRingEdges.		Les arêtes d'un anneau indiquent des faces conflictuelles dans le sens de la progression. C'est ce que l'on appelle un "Side Location Conflict" (Conflit d'emplacement latéral).
anneau non fermé	Identifiant signé d'une arête, identifiant l'anneau. Voir GetRingEdges.		Un anneau d'arêtes formé par les attributs next_left_edge/next_right_edge commence et se termine sur différents nœuds.
la face a plusieurs enveloppes	Identifiant de la face contestée.	Identifiant signé d'une arête, identifiant l'anneau. Voir GetRingEdges.	Plus d'un anneau d'arêtes indique la même face à l'intérieur.

Disponibilité : 1.0.0

Amélioration : 2.0.0 détection plus efficace des croisements d'arêtes et correction des faux positifs qui existaient dans les versions précédentes.

Modifié : 2.2.0 les valeurs pour id1 et id2 ont été échangées pour "edge crosses node" pour être cohérent avec la description de l'erreur.

Modifié : 3.2.0 a ajouté le paramètre optionnel bbox, effectuant des vérifications de l'étiquetage des faces et de la liaison des arêtes.

Exemples

```
SELECT * FROM  topology.ValidateTopology('ma_topo');
      error        | id1 | id2
-------------------+-----+-----
face without edges |   1 |
```

Voir aussi

validatetopology_returntype, Topology_Load_Tiger

9.3.9 ValidateTopologyRelation

ValidateTopologyRelation — Renvoie des informations sur les enregistrements de relations topologiques non valides

Synopsis

setof record **ValidateTopologyRelation**(varchar toponame);

Description

Renvoie un ensemble d'enregistrements donnant des informations sur les invalidités dans la table des relations de la topologie.

Disponibilité : 3.2.0

Voir aussi

ValidateTopology

9.3.10 FindTopology

FindTopology — Renvoie un enregistrement topologique par différents moyens.

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

Prend un identifiant de topologie ou l'identifiant d'un objet lié à la topologie et renvoie un enregistrement topology.topology.

Disponibilité : 3.2.0

Exemples

```
SELECT name(findTopology('features.land_parcels', 'feature'));
   name
-----------
 city_data
(1 row)
```

Voir aussi

FindLayer

9.3.11 FindLayer

FindLayer — Renvoie un enregistrement topology.layer par différents moyens.

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

Prend un identifiant de couche ou l'identifiant d'un objet lié à la topologie et renvoie un enregistrement topology.layer.

Disponibilité : 3.2.0

Exemples

```
SELECT layer_id(findLayer('features.land_parcels', 'feature'));
 layer_id
----------
        1
(1 row)
```

Voir aussi

FindTopology

9.4 Gestion des statistiques de topologie

L'ajout d'éléments à une topologie déclenche de nombreuses requêtes dans la base de données pour trouver les arêtes existantes qui seront divisées, ajouter des nœuds et mettre à jour les arêtes qui formeront un nœud avec le nouveau réseau de lignes. C'est pourquoi il est utile que les statistiques relatives aux données contenues dans les tables de topologie soient à jour.

Les fonctions d'insertion et d'édition de topologie de PostGIS ne mettent pas automatiquement à jour les statistiques, car une mise à jour des statistiques après chaque changement dans une topologie serait exagérée, et c'est donc à l'appelant de s'en charger.

Note
Les statistiques mises à jour par autovacuum ne seront PAS visibles pour les transactions qui ont démarré avant la fin du processus d'autovacuum, de sorte que les transactions de longue durée devront exécuter ANALYZE elles-mêmes, pour utiliser les statistiques mises à jour.

9.5 Constructeurs de topologie

9.5.1 CreateTopology

CreateTopology — Crée un nouveau schéma topologique et l'enregistre dans la table topology.topology.

Synopsis

integer **CreateTopology**(varchar topology_schema_name);
integer **CreateTopology**(varchar topology_schema_name, integer srid);
integer **CreateTopology**(varchar topology_schema_name, integer srid, double precision prec);
integer **CreateTopology**(varchar topology_schema_name, integer srid, double precision prec, boolean hasz);

Description

Crée un nouveau schéma de topologie portant le nom `topology_name` et l'enregistre dans la table `topology.topology`. Les topologies doivent porter un nom unique. Les tables de topologie (`edge_data`, `face`, `node` et `relation`) sont créées dans le schéma. Elle renvoie l'identifiant de la topologie.

Le `srid` est le système de référence spatiale SRID pour la topologie.

La tolérance `prec` est mesurée dans les unités du système de référence spatiale. La tolérance est fixée par défaut à 0.

`hasz` La valeur par défaut est false si elle n'est pas spécifiée.

Cette fonction est similaire à la fonction SQL/MM ST_InitTopoGeo mais possède plus de fonctionnalités.

DIsponibilité : 1.1

Amélioration : 2.0 ajout de la signature acceptant hasZ

Exemples

Créez un schéma topologique appelé `ma_topo` qui stocke les arêtes et les nœuds en mètres de Massachusetts State Plane (SRID = 26986). La tolérance est de 0,5 mètre puisque le système de référence spatiale est basé sur le mètre.

```
SELECT topology.CreateTopology('ma_topo', 26986, 0.5);
```

Créer une topologie pour Rhode Island appelée `ri_topo` dans le système de référence spatiale State Plane-feet (SRID = 3438)

```
SELECT topology.CreateTopology('ri_topo', 3438) AS topoid;
topoid
------
2
```

Voir aussi

Section 4.5, ST_InitTopoGeo, Topology_Load_Tiger

9.5.2 CopyTopology

CopyTopology — Copie une topologie (nœuds, arêtes, faces, couches et TopoGeometries) dans un nouveau schéma

Synopsis

integer **CopyTopology**(varchar existing_topology_name, varchar new_name);

Description

Crée une nouvelle topologie avec le nom `new_name`, avec le SRID et la précision copiés à partir de `existing_topology_name` Les nœuds, les arêtes et les faces dans `existing_topology_name` sont copiés dans la nouvelle topologie, ainsi que les couches et leurs TopoGeometries associées.

Note
Les nouvelles lignes de la table `topology.layer` contiennent des valeurs synthétiques pour schema_name, table_name et feature_column. Ceci est dû au fait que les objets TopoGeometry n'existent qu'en tant que définition et ne sont pas encore disponibles dans une table définie par l'utilisateur.

Disponibilité : 2.0.0

Exemples

Faire une sauvegarde d'une topologie appelée `ma_topo`.

```
SELECT topology.CopyTopology('ma_topo', 'ma_topo_backup');
```

Voir aussi

Section 4.5, CreateTopology, RenameTopology

9.5.3 ST_InitTopoGeo

ST_InitTopoGeo — Crée un nouveau schéma topologique et l'enregistre dans la table topology.topology.

Synopsis

text **ST_InitTopoGeo**(varchar topology_schema_name);

Description

C'est l'équivalent SQL-MM de CreateTopology. Il manque des options pour le système de référence spatial et la tolérance. Il renvoie une description textuelle de la création de la topologie, au lieu de l'identifiant de la topologie.

DIsponibilité : 1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 Topo-Geo et Topo-Net 3 : Routine Details : X.3.17

Exemples

```
SELECT topology.ST_InitTopoGeo('topo_schema_to_create') AS topocreation;
                       astopocreation
-----------------------------------------------------------
 Topology-Geometry 'topo_schema_to_create' (id:7) created.
```

Voir aussi

CreateTopology

9.5.4 ST_CreateTopoGeo

ST_CreateTopoGeo — Ajoute une collection de géométries à une topologie vide donnée et renvoie un message détaillant le succès.

Synopsis

text **ST_CreateTopoGeo**(varchar atopology, geometry acollection);

Description

Ajoute une collection de géométries à une topologie vide donnée et renvoie un message détaillant le succès.

Utile pour remplir une topologie vide.

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details -- X.3.18

Exemples

```
-- Populate topology --
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)
  );

      st_createtopogeo
----------------------------
 Topology ri_topo populated


-- create tables and topo geometries --
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);

SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

Voir aussi

TopoGeo_LoadGeometry, AddTopoGeometryColumn, CreateTopology, DropTopology

9.5.5 TopoGeo_AddPoint

TopoGeo_AddPoint — Ajoute un point à une topologie existante en utilisant une tolérance et en divisant éventuellement une arête existante.

Synopsis

integer **TopoGeo_AddPoint**(varchar atopology, geometry apoint, float8 tolerance);

Description

Ajoute un point à une topologie existante et renvoie son identifiant. Le point donné s'accroche aux nœuds ou aux arêtes existants dans la limite d'une tolérance donnée. Une arête existante peut être divisée par le point accroché.

Disponibilité : 2.0.0

Voir aussi

TopoGeo_AddLineString, TopoGeo_AddPolygon, TopoGeo_LoadGeometry, AddNode, CreateTopology

9.5.6 TopoGeo_AddLineString

TopoGeo_AddLineString — Adds a linestring to an existing topology using a tolerance and possibly splitting existing edges/-faces.

Synopsis

SETOF integer **TopoGeo_AddLineString**(varchar atopology, geometry aline, float8 tolerance);

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.

Note
La mise à jour des statistiques sur les topologies chargées via cette fonction est du ressort de l'appelant, voir maintaining statistics during topology editing and population.

Disponibilité : 2.0.0

Enhanced: 3.2.0 added support for returning signed identifier.

Voir aussi

TopoGeo_AddPoint, TopoGeo_AddPolygon, TopoGeo_LoadGeometry, AddEdge, CreateTopology

9.5.7 TopoGeo_AddPolygon

TopoGeo_AddPolygon — Ajoute un polygone à une topologie existante en utilisant une tolérance et en divisant éventuellement les arêtes/faces existantes. Renvoie les identifiants des faces.

Synopsis

SETOF integer **TopoGeo_AddPolygon**(varchar atopology, geometry apoly, float8 tolerance);

Description

Ajoute un polygone à une topologie existante et renvoie un ensemble d'identifiants de face le constituant. La limite du polygone donné s'accrochera aux nœuds ou aux arêtes existants dans la limite d'une tolérance donnée. Les arêtes et les faces existantes peuvent être divisées par la limite du nouveau polygone.

Note
La mise à jour des statistiques sur les topologies chargées via cette fonction est du ressort de l'appelant, voir maintaining statistics during topology editing and population.

Disponibilité : 2.0.0

Voir aussi

TopoGeo_AddPoint, TopoGeo_AddLineString, TopoGeo_LoadGeometry, AddFace, CreateTopology

9.5.8 TopoGeo_LoadGeometry

TopoGeo_LoadGeometry — Charge une géométrie dans une topologie existante, en effectuant de l'accrochage et des divisions si nécessaire.

Synopsis

void **TopoGeo_LoadGeometry**(varchar atopology, geometry ageom, float8 tolerance);

Description

Charge une géométrie dans une topologie existante. La géométrie donnée s'accrochera aux nœuds ou au arêtes existantes dans la limite d'une tolérance donnée. Les arêtes et les faces existantes peuvent être divisées suite au chargement.

Note
La mise à jour des statistiques sur les topologies chargées via cette fonction est du ressort de l'appelant, voir maintaining statistics during topology editing and population.

Disponibilité : 3.5.0

Voir aussi

TopoGeo_AddPoint, TopoGeo_AddLineString, TopoGeo_AddPolygon, CreateTopology

9.6 Éditeurs de topologie

9.6.1 ST_AddIsoNode

ST_AddIsoNode — Ajoute un noeud isolé à une face dans une topologie et renvoie le nodeid du nouveau noeud. Si la face est nulle, le noeud est quand même créé.

Synopsis

integer **ST_AddIsoNode**(varchar atopology, integer aface, geometry apoint);

Description

Ajoute un noeud isolé avec un point `apoint` à une face existante avec un faceid `aface` à une topologie `atopology` et renvoie le nodeid du nouveau noeud.

Si le système de référence spatiale (srid) de la géométrie du point n'est pas le même que la topologie, si le `apoint` n'est pas une géométrie de point, si le point est nul ou si le point croise une arête existante (même aux limites), une exception est levée. Si le point existe déjà en tant que nœud, une exception est levée.

Si `aface` n'est pas null et que le `apoint` n'est pas à l'intérieur de la face, une exception est levée.

DIsponibilité : 1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Net Routines : X+1.3.1

Exemples

Voir aussi

AddNode, CreateTopology, DropTopology, ST_Intersects

9.6.2 ST_AddIsoEdge

ST_AddIsoEdge — Ajoute une arête isolée définie par la géométrie `alinestring` à une topologie reliant deux nœuds isolés existants `anode` et `anothernode` et renvoie l'identifiant de l'arête de la nouvelle arête.

Synopsis

integer **ST_AddIsoEdge**(varchar atopology, integer anode, integer anothernode, geometry alinestring);

Description

Ajoute une arête isolée définie par la géométrie `alinestring` à une topologie reliant deux nœuds isolés existants `anode` et `anothernode` et renvoie l'identifiant de l'arête de la nouvelle arête.

Si le système de référence spatiale (srid) de la géométrie `alinestring` n'est pas le même que la topologie, si l'un des arguments d'entrée est nul, si les noeuds sont contenus dans plus d'une face, ou si les noeuds sont des noeuds de départ ou d'arrivée d'une arête existante, une exception est levée.

Si le `alinestring` ne se trouve pas à l'intérieur de la face à laquelle appartiennent le `anode` et le `anothernode`, une exception est levée.

Si le `anode` et le `anothernode` ne sont pas les points de départ et d'arrivée du `alinestring`, une exception est levée.

DIsponibilité : 1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.4

Exemples

Voir aussi

ST_AddIsoNode, ST_IsSimple, ST_Within

9.6.3 ST_AddEdgeNewFaces

ST_AddEdgeNewFaces — Ajoutez une nouvelle arête et, si elle divise une face, supprimez la face d'origine et remplacez-la par deux nouvelles faces.

Synopsis

integer **ST_AddEdgeNewFaces**(varchar atopology, integer anode, integer anothernode, geometry acurve);

Description

Ajoutez une nouvelle arête et, si elle divise une face, supprimez la face d'origine et remplacez-la par deux nouvelles faces. Renvoie l'identifiant de l'arête nouvellement ajoutée.

Met à jour toutes les arêtes jointes existantes et les relations en conséquence.

Si l'un des arguments est nul, les nœuds donnés sont inconnus (doivent déjà exister dans la table `node` du schéma topologique), la `acurve` n'est pas une `LINESTRING`, le `anode` et `anothernode` ne sont pas les points de départ et d'arrivée de `acurve`, une erreur est générée.

Si le système de référence spatiale (srid) de la géométrie `acurve` n'est pas le même que la topologie, une exception est levée.

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.12

Exemples

Voir aussi

ST_RemEdgeNewFace

ST_AddEdgeModFace

9.6.4 ST_AddEdgeModFace

ST_AddEdgeModFace — Ajoutez une nouvelle arête et, si elle divise une face, modifiez la face d'origine et ajoutez une nouvelle face.

Synopsis

integer **ST_AddEdgeModFace**(varchar atopology, integer anode, integer anothernode, geometry acurve);

Description

Ajouter une nouvelle arête et, si cela divise une face, modifier la face d'origine et en ajouter une nouvelle.

Note
Si possible, la nouvelle face sera créée sur le côté gauche de la nouvelle arête. Cela ne sera pas possible si la face du côté gauche doit être la face de l'univers (non borné).

Renvoie l'identifiant de l'arête nouvellement ajoutée.

Met à jour toutes les arêtes jointes existantes et les relations en conséquence.

Si l'un des arguments est nul, les nœuds donnés sont inconnus (doivent déjà exister dans la table `node` du schéma topologique), la `acurve` n'est pas une `LINESTRING`, le `anode` et `anothernode` ne sont pas les points de départ et d'arrivée de `acurve`, une erreur est générée.

Si le système de référence spatiale (srid) de la géométrie `acurve` n'est pas le même que la topologie, une exception est levée.

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.13

Exemples

Voir aussi

ST_RemEdgeModFace

ST_AddEdgeNewFaces

9.6.5 ST_RemEdgeNewFace

ST_RemEdgeNewFace — Enlève une arête et, si l'arête enlevée séparait deux faces, supprime les faces originales et les remplace par une nouvelle face.

Synopsis

integer **ST_RemEdgeNewFace**(varchar atopology, integer anedge);

Description

Supprime une arête et, si l'arête supprimée séparait deux faces, supprime les faces originales et les remplace par une nouvelle face.

Renvoie l'identifiant d'une face nouvellement créée ou NULL, si aucune nouvelle face n'est créée. Aucune nouvelle face n'est créée lorsque l'arête supprimée est pendante, isolée ou confinée avec la face de l'univers (ce qui peut entraîner "l'inondation de la face" de l'autre côté de l'univers).

Met à jour toutes les arêtes jointes existantes et les relations en conséquence.

Refuse de supprimer une arête participant à la définition d'une TopoGeometry existante. Refuse de recoller deux faces si une TopoGeometry n'est définie que par l'une d'entre elles (et non par l'autre).

Si l'un des arguments est nul, si l'arête donnée est inconnue (elle doit déjà exister dans la table `edge` du schéma topologique), si le nom de la topologie n'est pas valide, une erreur est générée.

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.14

Exemples

Voir aussi

ST_RemEdgeModFace

ST_AddEdgeNewFaces

9.6.6 ST_RemEdgeModFace

ST_RemEdgeModFace — Supprime une arête et, si l'arête sépare deux faces, supprime une face et modifie l'autre face pour couvrir l'espace des deux.

Synopsis

integer **ST_RemEdgeModFace**(varchar atopology, integer anedge);

Description

Supprime une arête et, si l'arête supprimée sépare deux faces, supprime une face et modifie l'autre face pour couvrir l'espace des deux. La face de droite est conservée de préférence, par souci de cohérence avec ST_AddEdgeModFace. Renvoie l'identifiant de la face conservée.

Met à jour toutes les arêtes jointes existantes et les relations en conséquence.

Refuse de supprimer une arête participant à la définition d'une TopoGeometry existante. Refuse de recoller deux faces si une TopoGeometry n'est définie que par l'une d'entre elles (et non par l'autre).

Si l'un des arguments est nul, si l'arête donnée est inconnue (elle doit déjà exister dans la table `edge` du schéma topologique), si le nom de la topologie n'est pas valide, une erreur est générée.

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.15

Exemples

Voir aussi

ST_AddEdgeModFace

ST_RemEdgeNewFace

9.6.7 ST_ChangeEdgeGeom

ST_ChangeEdgeGeom — Modifie la forme d'une arête sans affecter la structure de la topologie.

Synopsis

text **ST_ChangeEdgeGeom**(varchar atopology, integer anedge, geometry acurve);

Description

Modifie la forme d'une arête sans affecter la structure de la topologie.

Si l'un des arguments est nul, si l'arête donnée n'existe pas dans la table `edge` du schéma topologique, si la `acurve` n'est pas une `LINESTRING`, ou si la modification changerait la topologie sous-jacente, une erreur est déclenchée.

Si le système de référence spatiale (srid) de la géométrie `acurve` n'est pas le même que la topologie, une exception est levée.

Si la nouvelle `acurve` n'est pas simple, une erreur est générée.

Si le déplacement de l'arête de l'ancienne à la nouvelle position heurte un obstacle, une erreur est générée.

Disponibilité : 1.1.0

Amélioration : 2.0.0 ajoute l'application de la cohérence topologique

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details X.3.6

Exemples

```
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) );
 ----
 Edge 1 changed
```

Voir aussi

ST_AddEdgeModFace

ST_RemEdgeModFace

ST_ModEdgeSplit

9.6.8 ST_ModEdgeSplit

ST_ModEdgeSplit — Fractionner une arête en créant un nouveau nœud le long d'une arête existante, en modifiant l'arête d'origine et en ajoutant une nouvelle arête.

Synopsis

integer **ST_ModEdgeSplit**(varchar atopology, integer anedge, geometry apoint);

Description

Fractionner une arête en créant un nouveau nœud le long d'une arête existante, en modifiant l'arête d'origine et en ajoutant une nouvelle arête. Cette opération met à jour toutes les arêtes jointes existantes et les relations en conséquence. Renvoie l'identifiant du nœud nouvellement ajouté.

DIsponibilité : 1.1

Modifié : 2.0 - Dans les versions antérieures, cette fonction était mal nommée ST_ModEdgesSplit

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.9

Exemples

```
-- Add an edge --
 SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227592 893910, 227600  ↩
     893910)', 26986) ) As edgeid;

-- edgeid-
3


-- Split the edge  --
SELECT topology.ST_ModEdgeSplit('ma_topo',  3, ST_SetSRID(ST_Point(227594,893910),26986)  ) ↩
     As node_id;
        node_id
-------------------------
7
```

Voir aussi

ST_NewEdgesSplit, ST_ModEdgeHeal, ST_NewEdgeHeal, AddEdge

9.6.9 ST_ModEdgeHeal

ST_ModEdgeHeal — Répare deux arêtes en supprimant le nœud qui les relie, en modifiant la première arête et en supprimant la seconde. Renvoie l'identifiant du nœud supprimé.

Synopsis

int **ST_ModEdgeHeal**(varchar atopology, integer anedge, integer anotheredge);

Description

Répare deux arêtes en supprimant le nœud qui les relie, en modifiant la première arête et en supprimant la seconde. Renvoie l'identifiant du nœud supprimé. Met à jour toutes les arêtes jointes existantes et les relations en conséquence.

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.9

Voir aussi

ST_ModEdgeSplit ST_NewEdgesSplit

9.6.10 ST_NewEdgeHeal

ST_NewEdgeHeal — Répare deux arêtes en supprimant le nœud qui les relie, en supprimant les deux arêtes et en les remplaçant par une arête dont la direction est la même que la première arête fournie.

Synopsis

int **ST_NewEdgeHeal**(varchar atopology, integer anedge, integer anotheredge);

Description

Répare deux arêtes en supprimant le nœud qui les relie, en supprimant les deux arêtes et en les remplaçant par une arête dont la direction est la même que la première arête fournie. Renvoie l'identifiant de la nouvelle arête remplaçant les arêtes réparées. Met à jour toutes les arêtes jointes existantes et les relations en conséquence.

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X.3.9

Voir aussi

ST_ModEdgeHeal ST_ModEdgeSplit ST_NewEdgesSplit

9.6.11 ST_MoveIsoNode

ST_MoveIsoNode — Déplace un nœud isolé dans une topologie d'un point à un autre. Si la nouvelle géométrie `apoint` existe en tant que noeud, une erreur est générée. Retourne la description du déplacement.

Synopsis

text **ST_MoveIsoNode**(varchar atopology, integer anode, geometry apoint);

Description

Déplace un nœud isolé dans une topologie d'un point à un autre. Si la nouvelle géométrie `apoint` existe en tant que noeud, une erreur est générée.

Si l'un des arguments est nul, si le `apoint` n'est pas un point, si le noeud existant n'est pas isolé (c'est un point de départ ou d'arrivée d'une arête existante), si l'emplacement du nouveau noeud coupe une arête existante (même aux points d'arrivée) ou si le nouvel emplacement se trouve dans une face différente (depuis la version 3.2.0), une exception est levée.

Si le système de référence spatiale (srid) de la géométrie du point n'est pas le même que la topologie, une exception est levée.

Disponibilité : 2.0.0

Amélioration : la version 3.2.0 garantit que le nœud ne peut pas être déplacé vers une autre face

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Net Routines: X.3.2

Exemples

```
-- Add an isolated node with no face  --
SELECT topology.ST_AddIsoNode('ma_topo',  NULL, ST_GeomFromText('POINT(227579 893916)',  ↩
    26986) ) As nodeid;
 nodeid
--------
      7
-- Move the new node --
SELECT topology.ST_MoveIsoNode('ma_topo', 7,  ST_GeomFromText('POINT(227579.5 893916.5)',  ↩
    26986) ) As descrip;
                     descrip
-------------------------------------------------
Isolated Node 7 moved to location 227579.5,893916.5
```

Voir aussi

ST_AddIsoNode

9.6.12 ST_NewEdgesSplit

ST_NewEdgesSplit — Fractionne une arête en créant un nouveau nœud le long d'une arête existante, en supprimant l'arête d'origine et en la remplaçant par deux nouvelles arêtes. Renvoie l'identifiant du nouveau nœud créé qui relie les nouvelles arêtes.

Synopsis

integer **ST_NewEdgesSplit**(varchar atopology, integer anedge, geometry apoint);

Description

Divise une arête avec l'identifiant `anedge` en créant un nouveau noeud avec l'emplacement du point `apoint` le long de l'arête actuelle, en supprimant l'arête d'origine et en la remplaçant par deux nouvelles arêtes. Renvoie l'identifiant du nouveau nœud créé qui relie les nouvelles arêtes. Met à jour toutes les arêtes jointes existantes et les relations en conséquence.

Si le système de référence spatiale (srid) de la géométrie du point n'est pas le même que la topologie, si le `apoint` n'est pas une géométrie de point, si le point est nul, si le point existe déjà en tant que nœud, si l'arête ne correspond pas à une arête existante ou si le point n'est pas à l'intérieur de l'arête, une exception est levée.

DIsponibilité : 1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Net Routines: X.3.8

Exemples

```
-- Add an edge  --
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575 893917,227592 893900) ↩
    ', 26986) ) As edgeid;
-- result-
edgeid
------
        2
-- Split the new edge --
SELECT topology.ST_NewEdgesSplit('ma_topo', 2,  ST_GeomFromText('POINT(227578.5 893913.5)', ↩
     26986) ) As newnodeid;
 newnodeid
---------
       6
```

Voir aussi

ST_ModEdgeSplit ST_ModEdgeHeal ST_NewEdgeHeal AddEdge

9.6.13 ST_RemoveIsoNode

ST_RemoveIsoNode — Supprime un noeud isolé et renvoie la description de l'action. Si le noeud n'est pas isolé (début ou fin d'une arête), une exception est levée.

Synopsis

text **ST_RemoveIsoNode**(varchar atopology, integer anode);

Description

Supprime un noeud isolé et renvoie la description de l'action. Si le nœud n'est pas isolé (est le début ou la fin d'une arête), une exception est levée.

DIsponibilité : 1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X+1.3.3

Exemples

```
-- Remove an isolated node with no face  --
SELECT topology.ST_RemoveIsoNode('ma_topo',  7 ) As result;
         result
-------------------------
 Isolated node 7 removed
```

Voir aussi

ST_AddIsoNode

9.6.14 ST_RemoveIsoEdge

ST_RemoveIsoEdge — Supprime une arête isolée et renvoie la description de l'action. Si l'arête n'est pas isolée, une exception est levée.

Synopsis

text **ST_RemoveIsoEdge**(varchar atopology, integer anedge);

Description

Supprime une arête isolée et renvoie la description de l'action. Si l'arête n'est pas isolée, une exception est levée.

DIsponibilité : 1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM: Topo-Geo et Topo-Net 3: Routine Details: X+1.3.3

Exemples

```
-- Remove an isolated node with no face  --
SELECT topology.ST_RemoveIsoNode('ma_topo',  7 ) As result;
         result
-------------------------
 Isolated node 7 removed
```

Voir aussi

ST_AddIsoNode

9.7 Accès à la topologie

9.7.1 GetEdgeByPoint

GetEdgeByPoint — Trouve l'identifiant d'une arête qui coupe un point donné.

Synopsis

integer **GetEdgeByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

Récupère l'identifiant d'une arête qui coupe un point.

La fonction renvoie un entier (id-edge) à partir d'une topologie, d'un POINT et d'une tolérance. Si la tolérance = 0, le point doit couper l'arête.

Si `apoint` n'intersecte pas une arête, renvoie 0 (zéro).

Si la tolérance d'utilisation > 0 et qu'il y a plus d'une arête près du point, une exception est levée.

Note
Si la tolérance = 0, la fonction utilise ST_Intersects, sinon elle utilise ST_DWithin.

Effectué par le module GEOS.

Disponibilité : 2.0.0

Exemples

Ces exemples utilisent les arêtes que nous avons créées dans AddEdge

```
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;
 with1mtol | withnotol
-----------+-----------
         2 |         0
```

```
SELECT topology.GetEdgeByPoint('ma_topo',geom, 1) As nearnode
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;

-- get error --
ERROR:  Two or more edges found
```

Voir aussi

AddEdge, GetNodeByPoint, GetFaceByPoint

9.7.2 GetFaceByPoint

GetFaceByPoint — Recherche la face intersectant un point donné.

Synopsis

integer **GetFaceByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

Recherche une face référencée par un point, avec une tolérance donnée.

La fonction recherchera effectivement une face coupant un cercle dont le point est le centre et la tolérance le rayon.

Si aucune face n'intersecte l'emplacement de la requête, la valeur 0 est renvoyée (face universelle).

Si plus d'une face croise l'emplacement de la requête, une exception est levée.

Disponibilité : 2.0.0

Amélioration : 3.2.0 mise en œuvre plus efficace et contrat plus clair, arrêt du fonctionnement avec des topologies non valides.

Exemples

```
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;

         with1mtol | withnotol
        -----------+-----------
                         1 |         0
```

```
SELECT topology.GetFaceByPoint('ma_topo',geom, 1) As nearnode
        FROM ST_GeomFromEWKT('POINT(227591.9 893900.4)') As geom;

-- get error --
ERROR:  Two or more faces found
```

Voir aussi

GetFaceContainingPoint, AddFace, GetNodeByPoint, GetEdgeByPoint

9.7.3 GetFaceContainingPoint

GetFaceContainingPoint — Recherche la face contenant un point.

Synopsis

integer **GetFaceContainingPoint**(text atopology, geometry apoint);

Description

Renvoie l'identifiant de la face contenant un point.

Une exception est levée si le point tombe sur la limite d'une face.

Note
La fonction repose sur une topologie valide, utilisant la liaison des arêtes et l'étiquetage des faces.

Disponibilité : 3.2.0

Voir aussi

ST_GetFaceGeometry

9.7.4 GetNodeByPoint

GetNodeByPoint — Recherche l'identifiant d'un nœud à un point donné.

Synopsis

integer **GetNodeByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

Récupère l'identifiant d'un nœud à un point donné.

La fonction renvoie un entier (id-node) à partir d'une topologie, d'un POINT et d'une tolérance. Si tolérance = 0, il s'agit d'une intersection exacte, sinon le nœud est extrait d'un intervalle.

Si `apoint` n'intersecte pas un noeud, renvoie 0 (zéro).

Si la tolérance d'utilisation > 0 et qu'il y a plus d'un nœud près du point, une exception est levée.

Note
Si la tolérance = 0, la fonction utilise ST_Intersects, sinon elle utilise ST_DWithin.

Effectué par le module GEOS.

Disponibilité : 2.0.0

Exemples

Ces exemples utilisent les arêtes que nous avons créées dans AddEdge

```
SELECT topology.GetNodeByPoint('ma_topo',geom, 1) As nearnode
 FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
  nearnode
----------
         2
```

```
SELECT topology.GetNodeByPoint('ma_topo',geom, 1000) As too_much_tolerance
 FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;

 ----get error--
 ERROR:  Two or more nodes found
```

Voir aussi

AddEdge, GetEdgeByPoint, GetFaceByPoint

9.7.5 GetTopologyID

GetTopologyID — Retourne l'identifiant d'une topologie dans la table topology.topology étant donné le nom de la topologie.

Synopsis

integer **GetTopologyID**(varchar toponame);

Description

Retourne l'identifiant d'une topologie dans la table topology.topology en fonction du nom de la topologie.

DIsponibilité : 1.1

Exemples

```
SELECT topology.GetTopologyID('ma_topo') As topo_id;
 topo_id
---------
       1
```

Voir aussi

CreateTopology, DropTopology, GetTopologyName, GetTopologySRID

9.7.6 GetTopologySRID

GetTopologySRID — Renvoie le SRID d'une topologie dans la table topology.topology en fonction du nom de la topologie.

Synopsis

integer **GetTopologyID**(varchar toponame);

Description

Renvoie l'identifiant de référence spatiale d'une topologie dans la table topology.topology en fonction du nom de la topologie.

Disponibilité : 2.0.0

Exemples

```
SELECT topology.GetTopologySRID('ma_topo') As SRID;
 SRID
-------
  4326
```

Voir aussi

CreateTopology, DropTopology, GetTopologyName, GetTopologyID

9.7.7 GetTopologyName

GetTopologyName — Renvoie le nom d'une topologie (schéma) en fonction de l'identifiant de la topologie.

Synopsis

varchar **GetTopologyName**(integer topology_id);

Description

Renvoie le nom de la topologie (schéma) d'une topologie de la table topology.topology en fonction de l'identifiant de la topologie.

DIsponibilité : 1.1

Exemples

```
SELECT topology.GetTopologyName(1) As topo_name;
 topo_name
-----------
 ma_topo
```

Voir aussi

CreateTopology, DropTopology, GetTopologyID, GetTopologySRID

9.7.8 ST_GetFaceEdges

ST_GetFaceEdges — Renvoie un ensemble d'arêtes ordonnées qui délimitent `aface`.

Synopsis

getfaceedges_returntype **ST_GetFaceEdges**(varchar atopology, integer aface);

Description

Renvoie un ensemble d'arêtes ordonnées qui délimitent `aface`. Chaque sortie se compose d'une séquence et d'un numéro d'arête. Les numéros de séquence commencent par la valeur 1.

L'énumération des arêtes de chaque anneau commence par l'arête dont l'identifiant est le plus petit. L'ordre des arêtes suit la règle de la main gauche (la face liée se trouve à gauche de chaque arête dirigée).

Disponibilité : 2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 Topo-Geo et Topo-Net 3: Routine Details: X.3.5

Exemples

```
-- Returns the edges bounding face 1
SELECT (topology.ST_GetFaceEdges('tt', 1)).*;
-- result --
 sequence | edge
----------+------
        1 |   -4
        2 |    5
        3 |    7
        4 |   -6
        5 |    1
        6 |    2
        7 |    3
(7 rows)
```

```
-- Returns the sequence, edge id
-- and geometry of the edges that bound face 1
-- If you just need geom and seq, can use ST_GetFaceGeometry
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;
```

Voir aussi

GetRingEdges, AddFace, ST_GetFaceGeometry

9.7.9 ST_GetFaceGeometry

ST_GetFaceGeometry — Renvoie le polygone dans la topologie donnée avec l'identifiant de face spécifié.

Synopsis

geometry **ST_GetFaceGeometry**(varchar atopology, integer aface);

Description

Renvoie le polygone dans la topologie donnée avec l'identifiant de face spécifié. Construit le polygone à partir des arêtes composant la face.

DIsponibilité : 1.1

Cette méthode implémente la spécification SQL/MM. SQL-MM 3 Topo-Geo et Topo-Net 3: Routine Details: X.3.16

Exemples

```
-- Returns the wkt of the polygon added with AddFace
SELECT ST_AsText(topology.ST_GetFaceGeometry('ma_topo', 1)) As facegeomwkt;
-- result --
               facegeomwkt

------------------------------------------------------------------------------
 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))
```

Voir aussi

AddFace

9.7.10 GetRingEdges

GetRingEdges — Renvoie l'ensemble ordonné des identifiants d'arêtes signés rencontrés en évoluant sur un côté d'arête donné.

Synopsis

getfaceedges_returntype **GetRingEdges**(varchar atopology, integer aring, integer max_edges=null);

Description

Renvoie l'ensemble ordonné des identifiants d'arêtes signés rencontrés en évoluant sur un côté d'arête donné. Chaque sortie se compose d'une séquence et d'un identifiant d'arête signé. Les numéros de séquence commencent par la valeur 1.

Si vous passez un identifiant d'arête positif, la marche commence du côté gauche de l'arête correspondante et suit la direction de l'arête. Si vous passez un identifiant d'arête négatif, la marche commence du côté droit de l'arête correspondante et va vers l'arrière.

Si `max_edges` n'est pas nul, cette fonction ne renvoie pas plus que ces enregistrements. Il s'agit d'un paramètre de sécurité dans le cas de topologies éventuellement non valides.

Note
Cette fonction utilise les métadonnées de liaison des anneaux des arêtes.

Disponibilité : 2.0.0

Voir aussi

ST_GetFaceEdges, GetNodeEdges

9.7.11 GetNodeEdges

GetNodeEdges — Renvoie un ensemble ordonné d'arêtes incidentes au nœud donné.

Synopsis

getfaceedges_returntype **GetNodeEdges**(varchar atopology, integer anode);

Description

Renvoie un ensemble ordonné d'arêtes incidentes au nœud donné. Chaque sortie se compose d'une séquence et d'un numéro d'arête signé. Les numéros de séquence commencent par la valeur 1. Une arête positive commence au nœud donné. Une arête négative se termine au nœud donné. Les arêtes fermées apparaissent deux fois (avec les deux signes). L'ordre est celui des aiguilles d'une montre en partant du nord.

Note
Cette fonction calcule l'ordre plutôt que de dériver des métadonnées et peut donc être utilisée pour établir des liens entre les anneaux des arêtes.

Disponibilité : 2.0

Voir aussi

getfaceedges_returntype, GetRingEdges, ST_Azimuth

9.8 Traitement de la topologie

9.8.1 Polygonize

Polygonize — Recherche et enregistre toutes les faces définies par les arêtes de la topologie.

Synopsis

text **Polygonize**(varchar toponame);

Description

Enregistre toutes les faces qui peuvent être construites à partir d'une primitive d'arête topologique.

La topologie cible est supposée ne pas contenir d'arêtes s'auto-intersectant.

Note
Les faces déjà connues sont reconnues, de sorte qu'il est possible d'appeler Polygonize plusieurs fois sur la même topologie.

Note
Cette fonction n'utilise ni ne définit les champs next_left_edge et next_right_edge de la table des arêtes.

Disponibilité : 2.0.0

Voir aussi

AddFace, ST_Polygonize

9.8.2 AddNode

AddNode — Ajoute un nœud ponctuel à la table des nœuds dans le schéma topologique spécifié et renvoie le nodeid du nouveau nœud. Si le point existe déjà en tant que nœud, l'identifiant du nœud existant est renvoyé.

Synopsis

integer **AddNode**(varchar toponame, geometry apoint, boolean allowEdgeSplitting=false, boolean computeContainingFace=false);

Description

Ajoute un nœud ponctuel à la table des nœuds dans le schéma topologique spécifié. La fonction AddEdge ajoute automatiquement les points de départ et d'arrivée d'une arête lorsqu'elle est appelée, de sorte qu'il n'est pas nécessaire d'ajouter explicitement les nœuds d'une arête.

Si une arête traversant le nœud est trouvée, une exception est levée ou l'arête est scindée, en fonction de la valeur du paramètre `allowEdgeSplitting`.

Si `computeContainingFace` est true, un nœud nouvellement ajouté verra sa face contenante calculée correctement.

Note
Si la géométrie `apoint` existe déjà en tant que noeud, le noeud n'est pas ajouté mais le nodeid existant est renvoyé.

Disponibilité : 2.0.0

Exemples

```
SELECT topology.AddNode('ma_topo', ST_GeomFromText('POINT(227641.6 893816.5)', 26986) ) As ↩
    nodeid;
-- result --
nodeid
--------
 4
```

Voir aussi

AddEdge, CreateTopology

9.8.3 AddEdge

AddEdge — Ajoute une arête linéaire à la table des arêtes et les points de départ et d'arrivée associés à la table des nœuds de points du schéma topologique spécifié en utilisant la géométrie linéaire spécifiée et renvoie l'identifiant de l'arête nouvelle (ou existante).

Synopsis

integer **AddEdge**(varchar toponame, geometry aline);

Description

Ajoute une arête à la table des arêtes et les nœuds associés à la table des nœuds du schéma `toponame` spécifié en utilisant la géométrie de ligne spécifiée et renvoie l'identifiant de l'arête du nouvel enregistrement ou de l'enregistrement existant. L'arête nouvellement ajoutée a une face "univers" des deux côtés et est liée à elle-même.

Note
Si la géométrie `aline` croise, chevauche, contient ou est contenue par un bord de ligne existant, une erreur est générée et l'arête n'est pas ajoutée.

Note
La géométrie de `aline` doit avoir la même `srid` que celle définie pour la topologie, sinon une erreur sys de référence spatiale non valide sera levée.

Effectué par le module GEOS.

Warning
AddEdge is deprecated as of 3.5.0. Use TopoGeo_AddLineString instead.

Disponibilité : 2.0.0

Exemples

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575.8 893917.2,227591.9  ↩
    893900.4)', 26986) ) As edgeid;
-- result-
edgeid
--------
 1

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;
-- result --
edgeid
--------
 2

 SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.2 893900, 227591.9  ↩
     893900.4,
  227704.5 893778.5)', 26986) ) As edgeid;
 -- gives error --
 ERROR:  Edge intersects (not on endpoints) with existing edge 1
```

Voir aussi

TopoGeo_AddLineString, CreateTopology, Section 4.5

9.8.4 AddFace

AddFace — Enregistre une primitive de face dans une topologie et obtient son identifiant.

Synopsis

integer **AddFace**(varchar toponame, geometry apolygon, boolean force_new=false);

Description

Enregistre une primitive de face dans une topologie et obtient son identifiant.

Pour une face nouvellement ajoutée, les arêtes formant ses limites et celles contenues dans la face seront mises à jour pour que les valeurs des champs left_face et right_face soient correctes. Les nœuds isolés contenus dans la face seront également mis à jour pour que la valeur du champ containing_face soit correcte.

Note
Cette fonction n'utilise ni ne définit les champs next_left_edge et next_right_edge de la table des arêtes.

La topologie cible est supposée valide (ne contenant pas d'arêtes se recoupant). Une exception est levée si : La limite du polygone n'est pas entièrement définie par les arêtes existantes ou le polygone chevauche une face existante.

Si la géométrie `apolygon` existe déjà en tant que face, alors : si `force_new` est false (par défaut), l'identifiant de la face existante est renvoyé ; si `force_new` est true, un nouvel identifiant sera attribué à la face nouvellement enregistrée.

Note
Lorsqu'un nouvel enregistrement d'une face existante est effectué (force_new=true), aucune action ne sera entreprise pour résoudre les références pendantes à la face existante dans les tables d'arêtes, de nœuds et de relations, et le champ MBR (Minimum Bounding Rectangle) de l'enregistrement de la face existante ne sera pas mis à jour. C'est à l'appelant de s'en occuper.

Note
La géométrie `apolygon` doit avoir la même `srid` que celle définie pour la topologie, sinon une erreur sys de référence spatiale non valide sera déclenchée.

Disponibilité : 2.0.0

Exemples

```
-- first add the edges we use generate_series as an iterator (the below
-- will only work for polygons with < 10000 points because of our max in gs)
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;
-- result --
 edgeid
--------
      3
      4
      5
      6
      7
```

```
      8
      9
     10
     11
     12
(10 rows)
-- then add the face -

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;
-- result --
faceid
--------
 1
```

Voir aussi

AddEdge, CreateTopology, Section 4.5

9.8.5 ST_Simplify

ST_Simplify — Renvoie une version géométrique "simplifiée" de la TopoGeometry donnée en utilisant l'algorithme de Douglas-Peucker.

Synopsis

geometry **ST_Simplify**(TopoGeometry tg, float8 tolerance);

Description

Renvoie une version géométrique "simplifiée" de la TopoGeometry donnée en utilisant l'algorithme de Douglas-Peucker sur chaque arête de la composante.

Note
La géométrie renvoyée peut être non simple ou non valide.
La séparation des arêtes des composants peut contribuer à préserver la simplicité et la validité.

Effectué par le module GEOS.

Disponibilité : 2.1.0

Voir aussi

Geometry ST_Simplify, ST_IsSimple, ST_IsValid, ST_ModEdgeSplit

9.8.6 RemoveUnusedPrimitives

RemoveUnusedPrimitives — Supprime les primitives topologiques qui ne sont pas nécessaires pour définir les objets TopoGeometry existants.

Synopsis

int **RemoveUnusedPrimitives**(text topology_name, geometry bbox);

Description

Recherche toutes les primitives (nœuds, arêtes, faces) qui ne sont pas strictement nécessaires pour représenter les objets TopoGeometry existants et les supprime, en maintenant la validité de la topologie (liaison des arêtes, étiquetage des faces) et l'occupation de l'espace TopoGeometry.

Aucun nouvel identifiant de primitive n'est créé, mais les primitives existantes sont étendues pour inclure les faces fusionnées (en supprimant les arêtes) ou les arêtes réparées (en supprimant les nœuds).

Disponibilité : 3.3.0

Voir aussi

ST_ModEdgeHeal, ST_RemEdgeModFace

9.9 Constructeurs de TopoGeometry

9.9.1 CreateTopoGeom

CreateTopoGeom — Crée un nouvel objet géométrique topo à partir d'un tableau d'éléments topo - tg_type : 1 :[multi]point, 2 :[multi]ligne, 3 :[multi]poly, 4:collection

Synopsis

topogeometry **CreateTopoGeom**(varchar toponame, integer tg_type, integer layer_id, topoelementarray tg_objs);
topogeometry **CreateTopoGeom**(varchar toponame, integer tg_type, integer layer_id);

Description

Crée un objet topogéométrique pour la couche désignée par `layer_id` et l'enregistre dans la table des relations du schéma `toponame`.

`tg_type` est un nombre entier : 1 : [multi]point (ponctuel), 2 : [multi]ligne (linéaire), 3 : [multi]poly (aréolaire), 4:collection. `layer_id` est l'identifiant de la couche dans la table topology.layer.

Les couches ponctuelles sont formées à partir d'un ensemble de nœuds, les couches linéaires sont formées à partir d'un ensemble d'arêtes, les couches aréales sont formées à partir d'un ensemble de faces, et les collections peuvent être formées à partir d'un mélange de nœuds, d'arêtes et de faces.

L'omission du tableau de composants génère un objet TopoGeometry vide.

DIsponibilité : 1.1

Exemples : Forme à partir d'arêtes existantes

Créer un topogeom dans le schéma ri_topo pour la couche 2 (nos ri_roads), de type (2) LINE, pour la première arête (que nous avons chargée dans `ST_CreateTopoGeo`).

```
INSERT INTO ri.ri_roads(road_name, topo) VALUES('Unknown', topology.CreateTopoGeom('ri_topo ↩
    ',2,2,'{{1,2}}'::topology.topoelementarray);
```

Exemples : Convertir une géométrie aréolaire en une topogeometry de meilleure qualité

Supposons que nous ayons des géométries qui doivent être formées à partir d'une collection de faces. Nous disposons par exemple d'un tableau de groupes de blocs et nous voulons connaître la géométrie topographique de chaque groupe de blocs. Si nos données étaient parfaitement alignées, nous pourrions le faire :

```
-- create our topo geometry column --
SELECT topology.AddTopoGeometryColumn(
        'topo_boston',
        'boston', 'blockgroups', 'topo', 'POLYGON');

-- addtopgeometrycolumn --
1

-- update our column assuming
-- everything is perfectly aligned with our edges
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 allow for some error
--count the face if 50% of it falls
-- within what we think is our blockgroup boundary
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))
        OR
 (  ST_Intersects(b.geom, topology.ST_GetFaceGeometry('topo_boston', f.face_id))
            AND ST_Area(ST_Intersection(b.geom, topology.ST_GetFaceGeometry('topo_boston',  ↩
                f.face_id) ) ) >
                ST_Area(topology.ST_GetFaceGeometry('topo_boston', f.face_id))*0.5
                )
            GROUP BY b.gid) As foo
WHERE foo.gid = bg.gid;

-- and if we wanted to convert our topogeometry back
-- to a denormalized geometry aligned with our faces and edges
-- cast the topo to a geometry
-- The really cool thing is my new geometries
-- are now aligned with my tiger street centerlines
UPDATE boston.blockgroups SET new_geom = topo::geometry;
```

Voir aussi

AddTopoGeometryColumn, toTopoGeom ST_CreateTopoGeo, ST_GetFaceGeometry, TopoElementArray, TopoElementArray_Agg

9.9.2 toTopoGeom

toTopoGeom — Convertit une géométrie simple en une géométrie topographique.

Synopsis

topogeometry **toTopoGeom**(geometry geom, varchar toponame, integer layer_id, float8 tolerance);
topogeometry **toTopoGeom**(geometry geom, topogeometry topogeom, float8 tolerance);

Description

Convertit une géométrie simple en une TopoGeometry.

Les primitives topologiques requises pour représenter la géométrie d'entrée seront ajoutées à la topologie sous-jacente, éventuellement en divisant les primitives existantes, et elles seront associées à la TopoGeometry de sortie dans la table `relation`.

Les objets TopoGeometry existants (à l'exception éventuelle de `topogeom`, s'il en existe) conserveront leur forme.

Lorsque `tolerance` est donné, il sera utilisé pour accrocher la géométrie d'entrée aux primitives existantes.

Dans le premier formulaire, une nouvelle TopoGéométrie sera créée pour la couche donnée (`layer_id`) de la topologie donnée (`toponame`).

Dans la seconde forme, les primitives résultant de la conversion seront ajoutées à la TopoGeometry préexistante (`topogeom`), ajoutant éventuellement de l'espace à sa forme finale. Pour que la nouvelle forme remplace complètement l'ancienne, voir clearTopoGeom.

Disponibilité : 2.0

Amélioré : 2.1.0 ajoute la version d'une TopoGeometry existante.

Exemples

Il s'agit d'un flux de travail complet et autonome

```
-- do this if you don't have a topology setup already
-- creates topology not allowing any tolerance
SELECT topology.CreateTopology('topo_boston_test', 2249);
-- create a new table
CREATE TABLE nei_topo(gid serial primary key, nei varchar(30));
--add a topogeometry column to it
SELECT topology.AddTopoGeometryColumn('topo_boston_test', 'public', 'nei_topo', 'topo', ' ↩
    MULTIPOLYGON') As new_layer_id;
new_layer_id
-----------
1

--use new layer id in populating the new topogeometry column
-- we add the topogeoms to the new layer with 0 tolerance
INSERT INTO nei_topo(nei, topo)
SELECT nei,  topology.toTopoGeom(geom, 'topo_boston_test', 1)
FROM neighborhoods
WHERE gid BETWEEN 1 and 15;

--use to verify what has happened --
SELECT * FROM
    topology.TopologySummary('topo_boston_test');

-- summary--
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
UPDATE nei_topo SET topo = ST_Buffer(clearTopoGeom(topo), -10);

-- Get the no-one-lands left by the above operation
-- I think GRASS calls this "polygon0 layer"
SELECT ST_GetFaceGeometry('topo_boston_test', f.face_id)
  FROM topo_boston_test.face f
  WHERE f.face_id
> 0 -- don't consider the universe face
  AND NOT EXISTS ( -- check that no TopoGeometry references the face
    SELECT * FROM topo_boston_test.relation
    WHERE layer_id = 1 AND element_id = f.face_id
  );
```

Voir aussi

CreateTopology, AddTopoGeometryColumn, CreateTopoGeom, TopologySummary, clearTopoGeom

9.9.3 TopoElementArray_Agg

TopoElementArray_Agg — Renvoie un `topoelementarray` pour un ensemble de tableaux de type, element_id (topoelements).

Synopsis

topoelementarray **TopoElementArray_Agg**(topoelement set tefield);

Description

Utilisé pour créer un TopoElementArray à partir d'un ensemble de TopoElement.

Disponibilité : 2.0.0

Exemples

```
SELECT topology.TopoElementArray_Agg(ARRAY[e,t]) As tea
  FROM generate_series(1,3) As e CROSS JOIN generate_series(1,4) As t;
  tea
---------------------------------------------------------------------------
{{1,1},{1,2},{1,3},{1,4},{2,1},{2,2},{2,3},{2,4},{3,1},{3,2},{3,3},{3,4}}
```

Voir aussi

TopoElement, TopoElementArray

9.9.4 TopoElement

TopoElement — Convertit une topogeometry en un topoelement.

Synopsis

topoelement **TopoElement**(topogeometry topo);

Description

Convertit une TopoGeometry en TopoElement.

Disponibilité : 3.4.0

Exemples

Il s'agit d'un flux de travail complet et autonome

```
-- do this if you don't have a topology setup already
-- Creates topology not allowing any tolerance
SELECT TopoElement(topo)
FROM neighborhoods;
```

```
-- using as cast
SELECT topology.TopoElementArray_Agg(topo::topoelement)
FROM neighborhoods
GROUP BY city;
```

Voir aussi

TopoElementArray_Agg, TopoGeometry, TopoElement

9.10 Editeurs de TopoGeometry

9.10.1 clearTopoGeom

clearTopoGeom — Efface le contenu d'une géométrie topo.

Synopsis

topogeometry **clearTopoGeom**(topogeometry topogeom);

Description

Efface le contenu d'une TopoGeometry en la transformant en une vide. Principalement utile en conjonction avec toTopoGeom pour remplacer la forme des objets existants et de tout objet dépendant dans les niveaux hiérarchiques supérieurs.

Disponibilité : 2.1

Exemples

```
-- Shrink all TopoGeometry polygons by 10 meters
UPDATE nei_topo SET topo = ST_Buffer(clearTopoGeom(topo), -10);
```

Voir aussi

toTopoGeom

9.10.2 TopoGeom_addElement

TopoGeom_addElement — Ajoute un élément à la définition d'une TopoGeometry.

Synopsis

topogeometry **TopoGeom_addElement**(topogeometry tg, topoelement el);

Description

Ajoute un TopoElement à la définition d'un objet TopoGeometry. Il n'y a pas d'erreur si l'élément fait déjà partie de la définition.

Disponibilité : 2.3

Exemples

```
-- Add edge 5 to TopoGeometry tg
UPDATE mylayer SET tg = TopoGeom_addElement(tg, '{5,2}');
```

Voir aussi

TopoGeom_remElement, CreateTopoGeom

9.10.3 TopoGeom_remElement

TopoGeom_remElement — Supprime un élément de la définition d'une TopoGeometry.

Synopsis

topogeometry **TopoGeom_remElement**(topogeometry tg, topoelement el);

Description

Supprime une TopoElement de la définition d'un objet TopoGeometry.

Disponibilité : 2.3

Exemples

```
-- Remove face 43 from TopoGeometry tg
UPDATE mylayer SET tg = TopoGeom_remElement(tg, '{43,3}');
```

Voir aussi

TopoGeom_addElement, CreateTopoGeom

9.10.4 TopoGeom_addTopoGeom

TopoGeom_addTopoGeom — Ajoute un élément d'une TopoGeometry à la définition d'une autre TopoGeometry.

Synopsis

topogeometry **TopoGeom_addTopoGeom**(topogeometry tgt, topogeometry src);

Description

Ajoute les éléments d'une TopoGeometry à la définition d'une autre TopoGeometry, en changeant éventuellement son type mis en cache (attribut type) en une collection, si nécessaire pour contenir tous les éléments de l'objet source.

Les deux objets TopoGeometry doivent être définis par rapport à la même topologie et, s'ils sont définis hiérarchiquement, ils doivent être composés d'éléments de la même couche enfant.

Disponibilité : 3.2

Exemples

```
-- Set an "overall" TopoGeometry value to be composed by all
-- elements of specific TopoGeometry values
UPDATE mylayer SET tg_overall = TopoGeom_addTopogeom(
    TopoGeom_addTopoGeom(
        clearTopoGeom(tg_overall),
        tg_specific1
    ),
    tg_specific2
);
```

Voir aussi

TopoGeom_addElement, clearTopoGeom, CreateTopoGeom

9.10.5 toTopoGeom

toTopoGeom — Ajoute une forme géométrique à une géométrie topographique existante.

Description

Se référer à toTopoGeom.

9.11 Accès aux TopoGeometry

9.11.1 GetTopoGeomElementArray

GetTopoGeomElementArray — Renvoie un `topoelementarray` (un tableau de topoelements) contenant les éléments topologiques et le type de la TopoGeometry donnée (éléments primitifs).

Synopsis

topoelementarray **GetTopoGeomElementArray**(varchar toponame, integer layer_id, integer tg_id);

topoelementarray **GetTopoGeomElementArray**(topogeometry tg);

Description

Renvoie un TopoElementArray contenant les éléments topologiques et le type de la TopoGeometry donnée (éléments primitifs). Cette fonction est similaire à GetTopoGeomElements, sauf qu'elle renvoie les éléments sous la forme d'un tableau plutôt que sous la forme d'un ensemble de données.

tg_id est l'identifiant topogeometry de l'objet topogeometry de la topologie dans la couche désignée par `layer_id` dans la table topology.layer.

DIsponibilité : 1.1

Exemples

Voir aussi

GetTopoGeomElements, TopoElementArray

9.11.2 GetTopoGeomElements

GetTopoGeomElements — Renvoie un ensemble d'objets `topoelement` contenant les éléments topologiques element_id,element_type de la TopoGeometry donnée (éléments primitifs).

Synopsis

setof topoelement **GetTopoGeomElements**(varchar toponame, integer layer_id, integer tg_id);

setof topoelement **GetTopoGeomElements**(topogeometry tg);

Description

Renvoie un ensemble de element_id,element_type (topoelements) correspondant aux éléments topologiques primitifs TopoElement (1 : noeuds, 2 : arêtes, 3 : faces) dont est composé un objet topogeometry donné dans le schéma `toponame`.

tg_id est l'identifiant topogeometry de l'objet topogeometry de la topologie dans la couche désignée par `layer_id` dans la table topology.layer.

Disponibilité : 2.0.0

Exemples

Voir aussi

GetTopoGeomElementArray, TopoElement, TopoGeom_addElement, TopoGeom_remElement

9.11.3 ST_SRID

ST_SRID — Renvoie l'identifiant de référence spatiale d'une topogeometry.

Synopsis

integer **ST_SRID**(topogeometry tg);

Description

Renvoie l'identifiant de référence spatiale pour la ST_Geometry tel que défini dans la table spatial_ref_sys. Section 4.5

Note
La table spatial_ref_sys est une table qui répertorie tous les systèmes de référence spatiale connus de PostGIS et qui est utilisée pour les transformations d'un système de référence spatiale à un autre. Il est donc important de vérifier que vous disposez du bon identifiant de système de référence spatiale si vous envisagez de transformer vos géométries.

Disponibilité : 3.2.0

Cette méthode implémente la spécification SQL/MM. SQL-MM 3: 14.1.5

Exemples

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)',4326));
                --result
                4326
```

Voir aussi

Section 4.5, ST_SetSRID, ST_Transform, ST_SRID

9.12 Sorties TopoGeometry

9.12.1 AsGML

AsGML — Renvoie la représentation GML d'une topogeometry.

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

Renvoie la représentation GML d'une topogeometry au format GML3. Si aucun `nsprefix_in` n'est spécifié, `gml` est utilisé. Fournir une chaîne de caractères vide pour nsprefix afin d'obtenir un espace de noms non qualifié. Les paramètres precision (par défaut : 15) et options (par défaut 1), s'ils sont donnés, sont transmis tels quels à l'appel sous-jacent à ST_AsGML.

Le paramètre `visitedTable`, s'il est donné, est utilisé pour garder une trace des éléments de nœud et d'arêtes visités afin d'utiliser des références croisées (xlink:xref) plutôt que de dupliquer les définitions. La table doit comporter (au moins) deux champs entiers : "element_type" et "element_id". L'utilisateur appelant doit avoir les privilèges de lecture et d'écriture sur la table donnée. Pour de meilleures performances, un index doit être défini sur `element_type` et `element_id`, dans cet ordre. Cet index sera créé automatiquement par l'ajout d'une contrainte d'unicité aux champs. Exemple :

```
CREATE TABLE visited (
  element_type integer, element_id integer,
  unique(element_type, element_id)
);
```

Le paramètre `idprefix`, s'il est indiqué, sera ajouté aux identifiants des balises Edge et Node.

Le paramètre `gmlver`, s'il est donné, sera transmis au ST_AsGML sous-jacent. La valeur par défaut est 3.

Disponibilité : 2.0.0

Exemples

Ceci utilise la géométrie topo que nous avons créée dans CreateTopoGeom

```
SELECT topology.AsGML(topo) As rdgml
  FROM ri.roads
  WHERE road_name = 'Unknown';

-- rdgml--
<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>
```

Même exercice que le précédent sans namespace

```
SELECT topology.AsGML(topo,'') As rdgml
  FROM ri.roads
  WHERE road_name = 'Unknown';

-- rdgml--
<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>
```

Voir aussi

CreateTopoGeom, ST_CreateTopoGeo

9.12.2 AsTopoJSON

AsTopoJSON — Renvoie la représentation TopoJSON d'une topogeometry.

Synopsis

text **AsTopoJSON**(topogeometry tg, regclass edgeMapTable);

Description

Renvoie la représentation TopoJSON d'une topogeometry. Si `edgeMapTable` n'est pas nul, il sera utilisé comme correspondance de consultation/stockage des identifiants d'arêtes aux indices d'arcs. Cela permet d'obtenir un tableau compact des "arcs" dans le document final.

La table, si elle est donnée, doit avoir un champ "arc_id" de type "serial" et un champ "edge_id" de type integer ; le code interrogera la table pour "edge_id" et il est donc recommandé d'ajouter un index sur ce champ.

Note
Les indices d'arc dans la sortie TopoJSON sont basés sur 0 mais ils sont basés sur 1 dans la table "edgeMapTable".

Un document TopoJSON complet devra contenir, en plus des extraits renvoyés par cette fonction, les arcs proprement dits et quelques en-têtes. Voir le TopoJSON specification.

Disponibilité : 2.1.0

Amélioration : 2.2.1 ajout de la prise en charge des entrées "puntal"

Voir aussi

ST_AsGeoJSON

Exemples

```
CREATE TEMP TABLE edgemap(arc_id serial, edge_id int unique);

-- header
SELECT '{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects ↩
    ": {'

-- objects
UNION ALL SELECT '"' || feature_name || '": ' || AsTopoJSON(feature, 'edgemap')
FROM features.big_parcels WHERE feature_name = 'P3P4';

-- arcs
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

-- footer
UNION ALL SELECT ']}'::text as t;

-- Result:
{ "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]]
 ]}
```

9.13 Relations spatiales de topologie

9.13.1 Equals

Equals — Retourne vrai si deux topogeometries sont composées des mêmes primitives topologiques.

Synopsis

boolean **Equals**(topogeometry tg1, topogeometry tg2);

Description

Retourne vrai si deux topogeometries sont composées des mêmes primitives topologiques : faces, arêtes, nœuds.

Note
Cette fonction n'est pas prise en charge pour les topogeometries qui sont des collections de géométries. Elle ne permet pas non plus de comparer des topogeometries issues de topologies différentes.

Disponibilité : 1.1.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Voir aussi

GetTopoGeomElements, ST_Equals

9.13.2 Intersects

Intersects — Retourne true si une paire de primitives des deux topogeometries s'intersectent.

Synopsis

boolean **Intersects**(topogeometry tg1, topogeometry tg2);

Description

Retourne true si une paire de primitives des deux topogeometries s'intersectent.

Note
Cette fonction n'est pas prise en charge pour les topogeometries qui sont des collections de géométries. Elle ne permet pas non plus de comparer des topogeometries issues de topologies différentes. Elle n'est pas non plus compatible avec les topogeometries hiérarchiques (topogeometries composées d'autres topogeometries).

Disponibilité : 1.1.0

Cette fonction prend en charge la 3D et ne supprime pas l'indice z.

Exemples

Voir aussi

ST_Intersects

9.14 Importer et exporter des topologies

Une fois que vous avez créé des topologies, et éventuellement des couches topologiques associées, vous pouvez les exporter dans un format de fichier pour les sauvegarder ou les transférer dans une autre base de données.

L'utilisation des outils standards de dump/restauration de PostgreSQL est problématique car les topologies sont composées d'un ensemble de tables (4 pour les primitives, un nombre arbitraire pour les couches) et d'enregistrements dans des tables de métadonnées (topology.topology et topology.layer). De plus, les identifiants de topologie ne sont pas univoques d'une base de données à l'autre, de sorte que les paramètres de votre topologie devront être modifiés lors de sa restauration.

Afin de simplifier l'exportation/la restauration des topologies, une paire d'exécutables est fournie : `pgtopo_export` et `pgtopo_import`. Exemple d'utilisation :

```
pgtopo_export dev_db topo1 | pgtopo_import topo1 | psql staging_db
```

9.14.1 Utiliser l'exportateur de topologie

Le script `pgtopo_export` prend le nom d'une base de données et d'une topologie et produit un fichier dump qui peut être utilisé pour importer la topologie (et les couches associées) dans une nouvelle base de données.

Par défaut, `pgtopo_export` écrit le fichier dump sur la sortie standard afin qu'il puisse être acheminé vers `pgtopo_import` ou redirigé vers un fichier (refusant d'écrire dans le terminal). Vous pouvez optionnellement spécifier un nom de fichier de sortie en utilisant l'argument `-f` dans la ligne de commandes.

Par défaut `pgtopo_export` inclut un dump de toutes les couches définies par rapport à la topologie donnée. Cela peut être plus de données que vous n'en avez besoin, ou peut ne pas fonctionner (dans le cas où vos tables de couches ont des dépendances complexes), auquel cas vous pouvez demander à ignorer les couches avec l'argument `--skip-layers` et les traiter séparément.

En utilisant `pgtopo_export` avec l'argument `--help` (ou `-h` en abrégé) affichera toujours une courte chaîne de caractères sur l'utilisation.

Le format du fichier dump est une archive tar compressée d'un répertoire `pgtopo_export` contenant au moins un fichier `pgtopo_dump_version` avec des informations sur la version du format. A partir de la version `1`, le répertoire contient des fichiers CSV délimités par des tabulations avec les données des tables primitives de topologie (node, edge_data, face, relation), les enregistrements de topologie et de couche associés et (sauf si `--skip-layers` est donné) un dump PostgreSQL au format personnalisé des tables signalées comme étant des couches de la topologie donnée.

9.14.2 Utiliser l'importateur de topologie

Le script `pgtopo_import` prend un dump topologique au format `pgtopo_export` et un nom à donner à la topologie à créer et produit un script SQL reconstruisant la topologie et les couches associées.

Le fichier SQL généré contiendra des instructions qui créent une topologie avec le nom donné, chargent les données primitives, restaurent et enregistrent toutes les couches de la topologie en liant correctement toutes les valeurs TopoGeometry à leur topologie correcte.

Par défaut, `pgtopo_import` lit le dump depuis l'entrée standard afin qu'il puisse être utilisé en conjonction avec `pgtopo_export` dans une chaîne de traitement. Vous pouvez optionnellement spécifier un nom de fichier d'entrée avec l'argument `-f` en ligne de commande.

Par défaut, `pgtopo_import` inclut dans le fichier SQL de sortie le code permettant de restaurer toutes les couches trouvées dans le dump.

Ceci peut être indésirable ou ne pas fonctionner si votre base de données cible a déjà des tables avec le même nom que celles dans le dump. Dans ce cas, vous pouvez demander à ignorer les couches avec l'argument `--skip-layers` et les traiter séparément (ou plus tard).

SQL pour charger et lier uniquement les couches à une topologie nommée peut être généré en utilisant l'argument `--only-layers`. Cela peut être utile pour charger des couches APRÈS avoir résolu les conflits de noms ou pour lier des couches à une topologie différente (par exemple une version spatialement simplifiée de la topologie de départ).

Chapter 10

Gestion des données raster, requêtes et applications

10.1 Chargement et création de rasters

Dans la plupart des cas, vous créerez des rasters PostGIS en chargeant des fichiers raster existants à l'aide du chargeur de rasters `raster2pgsql`.

10.1.1 Utilisation de raster2pgsql pour charger des rasters

Le `raster2pgsql` est un exécutable de chargement de données raster qui charge les formats de données matricielles pris en charge par GDAL en SQL adapté au chargement dans une table raster PostGIS. Il est capable de charger des dossiers de fichiers raster ainsi que de créer des aperçus de rasters.

Comme raster2pgsql est le plus souvent compilé dans le cadre de PostGIS (à moins que vous ne compiliez votre propre bibliothèque GDAL), les types de données raster pris en charge par l'exécutable seront les mêmes que ceux compilés dans la bibliothèque de dépendance GDAL. Pour obtenir une liste des types de données raster que votre `raster2pgsql` supporte, utilisez l'argument `-G`.

Note
Lors de la création de vues d'ensemble d'un facteur spécifique à partir d'un ensemble de rasters alignés, il est possible que les vues d'ensemble ne soient pas alignées. Consultez http://trac.osgeo.org/postgis/ticket/1764 pour un exemple où les vues d'ensemble ne sont pas alignées.

10.1.1.1 Exemple d'utilisation

Une session d'exemple utilisant le chargeur pour créer un fichier d'entrée et le charger en morceaux de 100 x 100 tuiles peut ressembler à ceci :

```
# -s use srid 4326
# -I create spatial index
# -C use standard raster constraints
# -M vacuum analyze after load
# *.tif load all these files
# -F include a filename column in the raster table
# -t tile the output 100x100
# public.demelevation load into this table
raster2pgsql -s 4326 -I -C -M -F -t 100x100 *.tif public.demelevation
> elev.sql
```

```
# -d connect to this database
# -f read this file after connecting
psql -d gisdb -f elev.sql
```

Note
Si vous ne spécifiez pas le schéma dans le nom de la table cible, la table sera créée dans le schéma par défaut de la base de données ou de l'utilisateur avec lequel vous vous connectez.

Une conversion et un téléchargement peuvent être effectués en une seule étape à l'aide de commandes UNIX :

```
raster2pgsql -s 4326 -I -C -M *.tif -F -t 100x100 public.demelevation | psql -d gisdb
```

Charger les rasters Massachusetts state plane meters aerial tiles dans un schéma appelé `aerial` et créer une vue complète, des tables d'ensemble à 2 et 4 niveaux, utiliser le mode copy pour l'insertion (pas de fichier intermédiaire juste directement dans la base de données), et -e ne pas forcer tout dans une transaction (bien si vous voulez voir les données dans les tables tout de suite sans attendre). Décomposer les rasters en tuiles de 128x128 pixels et appliquer les contraintes des rasters. Utilisez le mode copy au lieu de l'insertion de table. (-F) Inclure un champ appelé filename pour contenir le nom du fichier à partir duquel les tuiles ont été découpées.

```
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
```

```
--get a list of raster types supported:
raster2pgsql -G
```

La commande -G génère une liste du type

```
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 options raster2pgsql

-? Affiche l'écran d'aide. L'aide s'affichera également si vous ne fournissez aucun argument.

-G Affiche les formats raster pris en charge.

(c|a|d|p) Ces options s'excluent mutuellement :

- **-c** Créer une nouvelle table et la remplir de données raster, *c'est le mode par défaut*
- **-a** Ajouter des données raster à une table existante.
- **-d** Supprimer la table, en créer une nouvelle et l'alimenter avec des données raster
- **-p** En mode préparation, ne créez que la table.

Traitement des données raster : Application de contraintes pour un enregistrement correct dans les catalogues de données

-C Appliquer les contraintes du raster -- srid, pixelsize etc. pour s'assurer que le raster est correctement enregistré dans la vue `raster_columns`.

-x Désactive la contrainte d'étendue maximale. Ne s'applique que si l'option -C est également utilisée.

-r Définit les contraintes (espacement unique et tuiles de couverture) pour le blocage régulier. Ne s'applique que si l'option -C est également utilisée.

Traitement des données raster : Paramètres facultatifs utilisés pour manipuler le jeu de données raster d'entrée

-s <SRID> Attribue au raster de sortie le SRID spécifié. S'il n'est pas fourni ou s'il est égal à zéro, les métadonnées du raster seront vérifiées pour déterminer un SRID approprié.

-b BAND Indice (basé sur 1) de la bande à extraire du raster. Pour plus d'un indice de bande, séparez-les par une virgule (,). Si rien n'est spécifié, toutes les bandes de l'image seront extraites.

-t TILE_SIZE Découpe le raster en tuiles qui seront insérées une par une dans la table. `TILE_SIZE` est exprimée en LARGEURxHAUTEUR ou fixée à la valeur "auto" pour permettre au chargeur de calculer une taille de tuile appropriée en utilisant le premier raster et en l'appliquant à tous les rasters.

-P Les tuiles les plus à droite et les plus en bas sont remplacées pour garantir que toutes les tuiles ont la même largeur et la même hauteur.

-R, --register Enregistrer le raster en tant que raster de système de fichiers (out-db).
Seules les métadonnées du raster et le chemin d'accès au raster sont stockés dans la base de données (pas les pixels).

-l OVERVIEW_FACTOR Créer un aperçu du raster. Pour plus d'un facteur, séparez-les par une virgule (,). Le nom de la table de synthèse suit le modèle o_`facteur de synthèse`_`table`, où `facteur de synthèse` est un espace réservé pour le facteur de synthèse numérique et `table` est remplacé par le nom de la table de base. La vue d'ensemble créée est stockée dans la base de données et n'est pas affectée par -R. Notez que le fichier sql généré contiendra à la fois la table principale et les tables de synthèse.

-N NODATA Valeur NODATA à utiliser pour les bandes sans valeur NODATA.

Paramètres facultatifs utilisés pour manipuler les objets de la base de données

-f COLUMN Spécifier le nom de la colonne raster de destination, la valeur par défaut étant "rast"

-F Ajouter une colonne avec le nom du fichier

-n COLUMN Spécifier le nom de la colonne "nom de fichier". Implique -F.

-q Mettre les identifiants PostgreSQL entre guillemets.

-I Créer un index GiST sur la colonne raster.

-M Vacuum analyze la table raster.

-k Conserve les tuiles vides et saute les vérifications des valeurs NODATA pour chaque bande de données matricielles. Vous gagnez du temps en vérifiant, mais vous risquez de vous retrouver avec beaucoup plus de lignes inutiles dans votre base de données et ces lignes inutiles ne sont pas marquées comme des tuiles vides.

-T tablespace Spécifier le tablespace pour la nouvelle table. Notez que les index (y compris la clé primaire) utiliseront toujours le tablespace par défaut à moins que l'option -X ne soit également utilisée.

-X tablespace Spécifier le tablespace pour le nouvel index de la table. Ceci s'applique à la clé primaire et à l'index spatial si l'option -I est utilisée.

-Y max_rows_per_copy=50 Utiliser les instructions de copie au lieu des instructions d'insertion. Spécifier optionnellement `max_rows_per_copy` ; par défaut 50 si non spécifié.

-e Exécutez chaque déclaration individuellement, n'utilisez pas de transaction.

-E ENDIAN Contrôle l'ordre des octets de la sortie binaire générée du raster ; spécifier 0 pour XDR et 1 pour NDR (par défaut) ; seule la sortie NDR est prise en charge actuellement

-V version Indique la version du format de sortie. La valeur par défaut est 0. Seule la valeur 0 est prise en charge pour l'instant.

10.1.2 Création de rasters à l'aide des fonctions raster de PostGIS

Il arrive souvent que vous souhaitiez créer des rasters et des tables de rasters directement dans la base de données. Il existe une multitude de fonctions pour ce faire. Les étapes générales à suivre.

1. Créer une table avec une colonne raster pour contenir les nouveaux enregistrements raster, ce qui peut être fait avec :

   ```
   CREATE TABLE myrasters(rid serial primary key, rast raster);
   ```

2. Il existe de nombreuses fonctions pour vous aider à atteindre cet objectif. Si vous créez des rasters qui ne sont pas des dérivés d'autres rasters, vous devrez commencer par ST_MakeEmptyRaster, suivi de ST_AddBand

   Vous pouvez également créer des rasters à partir de géométries. Pour ce faire, vous devrez utiliser ST_AsRaster éventuellement accompagné d'autres fonctions telles que ST_Union ou ST_MapAlgebraFct ou n'importe quelle autre fonction d'algèbre de carte.

   Il existe encore bien d'autres options pour créer de nouvelles tables raster à partir de tables existantes. Par exemple, vous pouvez créer une table raster dans une projection différente de celle d'une table existante en utilisant ST_Transform

3. Une fois que vous avez rempli votre table initialement, vous devez créer un index spatial sur la colonne des données raster avec quelque chose comme :

   ```
   CREATE INDEX myrasters_rast_st_convexhull_idx ON myrasters USING gist( ST_ConvexHull( ↩
       rast) );
   ```

   Notez l'utilisation de ST_ConvexHull puisque la plupart des opérateurs de données raster sont basés sur l'enveloppe convexe des données raster.

   Note
   Les versions antérieures à la version 2.0 de PostGIS raster étaient basées sur l'enveloppe plutôt que sur l'enveloppe convexe. Pour que les index spatiaux fonctionnent correctement, vous devez les supprimer et les remplacer par des index basés sur l'enveloppe convexe.

4. Appliquer des contraintes raster en utilisant AddRasterConstraints

10.1.3 Utilisation de rasters "out db" stockés sur le cloud

L'outil `raster2pgsql` utilise GDAL pour accéder aux données raster et peut tirer parti d'une fonctionnalité clé de GDAL : la possibilité de lire des données raster qui sont stockées à distance dans des "magasins d'objets" en nuage (par exemple AWS S3, Google Cloud Storage).

L'utilisation efficace des rasters stockés dans le cloud nécessite l'utilisation d'un format "optimisé pour le cloud". Le plus connu et le plus largement utilisé est le format "cloud optimized GeoTIFF". L'utilisation d'un format non optimisé pour le cloud, comme un JPEG ou un TIFF non structuré, se traduira par des performances très médiocres, car le système devra télécharger l'intégralité du raster chaque fois qu'il devra accéder à un sous-ensemble.

Tout d'abord, chargez votre raster dans le service de stockage cloud de votre choix. Une fois qu'il est chargé, vous aurez un URI pour y accéder, soit un URI "http", soit parfois un URI spécifique au service (par exemple, "s3://bucket/object") Pour accéder aux buckets non publics, vous devrez fournir des options de configuration GDAL pour authentifier votre connexion. Notez que cette commande *lit* à partir du raster (stocké dans le cloud) et *écrit* dans la base de données.

```
AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxxxx \
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx \
raster2pgsql \
  -s 990000 \
  -t 256x256 \
  -I \
  -R \
  /vsis3/your.bucket.com/your_file.tif \
  your_table \
  | psql your_db
```

Une fois la table chargée, vous devez donner à la base de données la permission de lire les rasters distants, en définissant deux permissions, postgis.enable_outdb_rasters et postgis.gdal_enabled_drivers.

```
SET postgis.enable_outdb_rasters = true;
SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Pour que les changements soient durables, définissez-les directement dans votre base de données. Vous devrez vous reconnecter pour bénéficier des nouveaux paramètres.

```
ALTER DATABASE your_db SET postgis.enable_outdb_rasters = true;
ALTER DATABASE your_db SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Pour les rasters non publics, il se peut que vous deviez fournir des clés d'accès pour lire les rasters stockés dans le cloud. Les mêmes clés que vous avez utilisées pour écrire l'appel `raster2pgsql` peuvent être définies pour être utilisées à l'intérieur de la base de données, avec la configuration postgis.gdal_vsi_options. Notez que plusieurs options peuvent être définies en séparant les paires `key=value` par des espaces.

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
```

Une fois les données chargées et les autorisations définies, vous pouvez interagir avec la table raster comme avec n'importe quelle autre table raster, en utilisant les mêmes fonctions. La base de données se chargera de tous les mécanismes de connexion aux données stockées dans le cloud lorsqu'elle aura besoin de lire les données des pixels.

10.2 Catalogues Raster

Deux vues de catalogue raster sont fournies avec PostGIS. Ces deux vues utilisent des informations intégrées dans les contraintes des tables raster. Par conséquent, les vues de catalogue sont toujours cohérentes avec les données raster des tables puisque les contraintes sont appliquées.

1. `raster_columns` cette vue répertorie toutes les colonnes des tables raster de votre base de données.
2. `raster_overviews` cette vue répertorie toutes les colonnes des tables raster de votre base de données qui servent d'aperçu pour une table plus finement structurée. Les tables de ce type sont générées lorsque vous utilisez le commutateur `-l` pendant le chargement.

10.2.1 Catalogue des colonnes raster

Le `raster_columns` est un catalogue de toutes les colonnes des tables raster de votre base de données qui sont de type raster. Il s'agit d'une vue qui utilise les contraintes des tables afin que les informations soient toujours cohérentes, même si vous restaurez une table raster à partir d'une sauvegarde d'une autre base de données. Les colonnes suivantes existent dans le catalogue `raster_columns`.

Si vous n'avez pas créé vos tables avec le chargeur ou si vous avez oublié de spécifier l'option `-C` pendant le chargement, vous pouvez appliquer les contraintes après coup en utilisant AddRasterConstraints pour que le catalogue `raster_columns` enregistre les informations communes sur vos tuiles raster.

- `r_table_catalog` La base de données dans laquelle se trouve la table. La lecture se fera toujours dans la base de données actuelle.
- `r_table_schema` Schéma de la base de données auquel appartient la table raster.
- `r_table_name` table raster
- `r_raster_column` la colonne de la table `r_table_name` qui est de type raster. Rien dans PostGIS ne vous empêche d'avoir plusieurs colonnes raster par table. Il est donc possible d'avoir une table raster listée plusieurs fois avec une colonne raster différente pour chacune d'entre elles.

- `srid` L'identifiant de référence spatiale du raster. Il doit s'agir d'une entrée dans le fichier Section 4.5.
- `scale_x` L'échelle entre les coordonnées spatiales géométriques et le pixel. Elle n'est disponible que si toutes les tuiles de la colonne raster ont la même `scale_x` et que cette contrainte est appliquée. Se référer à ST_ScaleX pour plus de détails.
- `scale_y` L'échelle entre les coordonnées spatiales géométriques et le pixel. Elle n'est disponible que si toutes les tuiles de la colonne matricielle ont la même `scale_y` et que la contrainte `scale_y` est appliquée. Voir ST_ScaleY pour plus de détails.
- `blocksize_x` La largeur (nombre de pixels) de chaque tuile raster. Pour plus de détails, voir ST_Width.
- `blocksize_y` La largeur (nombre de pixels vers le bas) de chaque tuile raster. Voir ST_Height pour plus de détails.
- `same_alignment` Un booléen qui est vrai si toutes les tuiles raster ont le même alignement . Pour plus de détails, voir ST_SameAlignment.
- `regular_blocking` Si la colonne de données raster est soumise aux contraintes d'unicité spatiale et de couverture des tuiles, la valeur sera TRUE. Dans le cas contraire, la valeur sera FALSE.
- `num_bands` Le nombre de bandes dans chaque tuile de votre jeu de données raster. Il s'agit de la même information que celle fournie par ST_NumBands
- `pixel_types` Un tableau définissant le type de pixel pour chaque bande. Le nombre d'éléments de ce tableau est identique au nombre de bandes. Les types de pixels sont l'un des suivants, définis dans ST_BandPixelType.
- `nodata_values` Un tableau de nombres en double précision indiquant la `nodata_value` pour chaque bande. Le nombre d'éléments de ce tableau est égal au nombre de bandes. Ces nombres définissent la valeur du pixel pour chaque bande qui doit être ignorée pour la plupart des opérations. Il s'agit d'informations similaires à celles fournies par ST_BandNoDataValue.
- `out_db` Un tableau d'indicateurs booléens indiquant si les données des bandes raster sont conservées en dehors de la base de données. Le nombre d'éléments de ce tableau est égal au nombre de bandes.
- `extent` Il s'agit de l'étendue de toutes les lignes de votre jeu de données raster. Si vous prévoyez de charger d'autres données qui modifieront l'étendue de l'ensemble, vous devrez exécuter la fonction DropRasterConstraints avant le chargement, puis réappliquer les contraintes avec AddRasterConstraints après le chargement.
- `spatial_index` Un booléen qui est vrai si la colonne raster a un index spatial.

10.2.2 Aperçu des données raster

`raster_overviews` catalogue des informations sur les colonnes des tables raster utilisées pour les aperçus et des informations supplémentaires à leur sujet qu'il est utile de connaître lors de l'utilisation des aperçus. Les tableaux de synthèse sont catalogués à la fois dans `raster_columns` et `raster_overviews` parce qu'il s'agit d'images raster à part entière, mais aussi parce qu'elles servent de caricature à plus faible résolution d'un tableau à plus haute résolution. Elles sont générées en même temps que la table raster principale lorsque vous utilisez le commutateur `-l` dans le chargement des données raster ou peuvent être générées manuellement en utilisant AddOverviewConstraints.

Les tableaux d'aperçus contiennent les mêmes contraintes que les autres tableaux raster, ainsi que des contraintes supplémentaires spécifiques aux aperçus.

Note
Les informations contenues dans `raster_overviews` ne font pas double emploi avec les informations contenues dans `raster_columns`. Si vous avez besoin des informations sur un tableau de synthèse présentes dans `raster_columns`, vous pouvez joindre les `raster_overviews` et `raster_columns` pour obtenir l'ensemble des informations dont vous avez besoin.

Les deux principales raisons pour lesquelles les aperçus sont nécessaires sont les suivantes :

1. Représentation à basse résolution des tables centrales, couramment utilisée pour l'agrandissement rapide des cartes.

2. Les calculs y sont généralement plus rapides que sur leurs homologues à plus haute résolution, car il y a moins d'enregistrements et chaque pixel couvre un plus grand territoire. Bien que les calculs ne soient pas aussi précis que les tableaux à haute résolution qu'ils supportent, ils peuvent être suffisants pour de nombreux calculs empiriques.

Le catalogue `raster_overviews` contient les colonnes d'information suivantes.

- `o_table_catalog` La base de données dans laquelle se trouve l'aperçu. Il s'agit toujours de la base de données actuelle.
- `o_table_schema` Schéma de la base de données auquel appartient l'aperçu de la table raster.
- `o_table_name` nom de la table d'aperçu du raster
- `o_raster_column` la colonne raster dans le tableau de synthèse.
- `r_table_catalog` La base de données dans laquelle se trouve la table raster pour laquelle cette vue d'ensemble est utilisée. Il s'agit toujours de la base de données actuelle.
- `r_table_schema` Schéma de la base de données auquel appartient la table raster à laquelle ce service de vue d'ensemble est associé.
- `r_table_name` table raster que cet aperçu dessert.
- `r_raster_column` la colonne raster que cette colonne d'aperçu dessert.
- `overview_factor` - il s'agit du niveau pyramidal de l'aperçu. Plus le nombre est élevé, plus la résolution du tableau est faible. raster2pgsql, s'il reçoit un dossier d'images, calculera la vue d'ensemble de chaque fichier image et le chargera séparément. Le niveau 1 est supposé être le fichier original. Au niveau 2, chaque tuile représente 4 de l'original. Par exemple, si vous avez un dossier de fichiers images de 5000x5000 pixels que vous avez choisi de diviser en 125x125, pour chaque fichier image votre table de base aura (5000*5000)/(125*125) enregistrements = 1600, votre table (l=2) `o_2` aura un plafond(1600/Power(2,2)) = 400 lignes, votre (l=3) `o_3` aura un plafond(1600/Power(2,3)) = 200 lignes. Si vos pixels ne sont pas divisibles par la taille de vos tuiles, vous obtiendrez des bouts de tuiles (tuiles non complètement remplies). Notez que chaque tuile d'aperçu générée par raster2pgsql a le même nombre de pixels que son parent, mais a une résolution inférieure, chaque pixel représentant (Puissance(2,facteur_d'aperçu) pixels de l'original).

10.3 Créer des applications personnalisées avec PostGIS Raster

Le fait que PostGIS raster vous fournisse des fonctions SQL pour rendre les rasters dans les formats d'image connus vous donne beaucoup d'options pour les rendre. Par exemple, vous pouvez utiliser OpenOffice / LibreOffice pour le rendu comme démontré dans Rendering PostGIS Raster graphics with LibreOffice Base Reports. En outre, vous pouvez utiliser une grande variété de langages, comme le montre cette section.

10.3.1 Exemple de sortie PHP utilisant ST_AsPNG avec d'autres fonctions raster

Dans cette section, nous allons montrer comment utiliser le driver PHP PostgreSQL et la famille de fonctions ST_AsGDALRaster pour produire les bandes 1,2,3 d'un raster dans un flux de requête PHP qui peut ensuite être incorporé dans une balise html img src.

L'exemple de requête montre comment combiner tout un ensemble de fonctions raster pour récupérer toutes les tuiles qui intersectent une boîte de délimitation wgs 84 particulière, puis unit avec ST_Union les tuiles intersectées en retournant toutes les bandes, transforme en projection spécifiée par l'utilisateur en utilisant ST_Transform, et sort les résultats au format png en utilisant ST_AsPNG.

Vous appellerez la commande ci-dessous en utilisant

```
http://mywebserver/test_raster.php?srid=2249
```

pour obtenir l'image raster en pieds de plan de l'État du Massachusetts.

```
<?php
/** contents of test_raster.php **/
$conn_str ='dbname=mydb host=localhost port=5432 user=myuser password=mypwd';
$dbconn = pg_connect($conn_str);
header('Content-Type: image/png');
/**If a particular projection was requested use it otherwise use mass state plane meters  ↩
    **/
if (!empty( $_REQUEST['srid'] ) &amp;&amp; is_numeric( $_REQUEST['srid']) ){
                $input_srid = intval($_REQUEST['srid']);
}
else { $input_srid = 26986; }
/** The set bytea_output may be needed for PostgreSQL 9.0+, but not for 8.4 **/
$sql = "set bytea_output='escape';
SELECT ST_AsPNG(ST_Transform(
                        ST_AddBand(ST_Union(rast,1), ARRAY[ST_Union(rast,2),ST_Union(rast ↩
                            ,3)])
                                ,$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]);
?>
```

10.3.2 Exemple ASP.NET C# Sortie utilisant ST_AsPNG en conjonction avec d'autres fonctions raster

Dans cette section, nous allons montrer comment utiliser le driver Npgsql PostgreSQL .NET et la famille de fonctions ST_AsGDALRaste pour produire les bandes 1,2,3 d'un raster dans un flux de requête PHP qui peut ensuite être incorporé dans une balise html img src.

Pour cet exercice, vous aurez besoin du driver PostgreSQL npgsql .NET dont vous pouvez obtenir la dernière version sur http://npgsql.projects.postgresql.org/. Il vous suffit de télécharger la dernière version et de la placer dans votre dossier ASP.NET bin.

L'exemple de requête montre comment combiner tout un ensemble de fonctions raster pour récupérer toutes les tuiles qui intersectent une boîte de délimitation wgs 84 particulière, puis unit avec ST_Union les tuiles intersectées en retournant toutes les bandes, transforme en projection spécifiée par l'utilisateur en utilisant ST_Transform, et sort les résultats au format png en utilisant ST_AsPNG.

Il s'agit du même exemple que Section 10.3.1 sauf qu'il est implémenté en C#.

Vous appellerez la commande ci-dessous en utilisant

```
http://mywebserver/TestRaster.ashx?srid=2249
```

pour obtenir l'image raster en pieds de plan de l'État du Massachusetts.

```
 -- web.config connection string section --
<connectionStrings>
    <add name="DSN"
        connectionString="server=localhost;database=mydb;Port=5432;User Id=myuser;password= ↩
            mypwd"/>
</connectionStrings>
```

```
// Code for TestRaster.ashx
<%@ 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 {
                    using (NpgsqlConnection conn = new NpgsqlConnection(System. ↩
                        Configuration.ConfigurationManager.ConnectionStrings["DSN"]. ↩
                        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;
        }
}
```

10.3.3 Application console Java qui produit une requête raster sous forme de fichier image

Il s'agit d'une simple application console Java qui prend une requête renvoyant une image et l'envoie dans le fichier spécifié.

Vous pouvez télécharger les derniers drivers JDBC de PostgreSQL à partir de http://jdbc.postgresql.org/download.html

Vous pouvez compiler le code suivant à l'aide d'une commande du type :

```
set env CLASSPATH .:..\postgresql-9.0-801.jdbc4.jar
javac SaveQueryImage.java
jar cfm SaveQueryImage.jar Manifest.txt *.class
```

Et l'appeler à partir de la ligne de commande avec quelque chose comme

```
java -jar SaveQueryImage.jar "SELECT ST_AsPNG(ST_AsRaster(ST_Buffer(ST_Point(1,5),10, ' ↩
    quad_segs=2'),150, 150, '8BUI',100));" "test.png"
```

```
-- Manifest.txt --
Class-Path: postgresql-9.0-801.jdbc4.jar
Main-Class: SaveQueryImage
```

```
// Code for SaveQueryImage.java
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);
      }
  }
}
```

10.3.4 Utiliser PLPython pour extraire des images via SQL

Il s'agit d'une fonction stockée en plpython qui crée un fichier dans le répertoire du serveur pour chaque enregistrement. Nécessite l'installation de plpython. Elle devrait fonctionner correctement avec plpythonu et plpython3u.

```
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
-- note the postgresql daemon account needs to have write access to folder
-- this echos back the file names created;
 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;

     write_file
---------------------
 C:/temp/slices1.png
 C:/temp/slices2.png
 C:/temp/slices3.png
 C:/temp/slices4.png
 C:/temp/slices5.png
```

10.3.5 Sortie de données raster avec PSQL

Malheureusement, PSQL n'a pas de fonctionnalité intégrée facile à utiliser pour produire des binaires. Il s'agit d'une astuce qui s'appuie sur le support des gros objets de PostgreSQL. Pour l'utiliser, lancez d'abord votre ligne de commande psql connectée à votre base de données.

Contrairement à l'approche python, cette approche crée le fichier sur votre ordinateur local.

```
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);
-- you'll get an output something like --
   oid   | num_bytes
---------+-----------
 2630819 |     74860

-- next note the oid and do this replacing the c:/test.png to file path location
-- on your local computer
 \lo_export 2630819 'C:/temp/aerial_samp.png'

-- this deletes the file from large object storage on db
SELECT lo_unlink(2630819);
```

Chapter 11

Référence Raster

Les fonctions mentionnées ci-dessous sont celles qu'un utilisateur typique de PostGIS Raster sera amené à utiliser et qui sont disponibles dans PostGIS Raster. D'autres fonctions sont également définies, requises en interne pour le support des objets rasters, mais ne sont pas destinées à être utilisées par le grand public.

`raster` est un nouveau type PostGIS pour stocker et analyser les données raster.

Pour charger des rasters depuis des fichiers, voir Section 10.1

Pour les exemples dans cette section, nous utiliserons une table de rasters fictifs, créée via la requête suivante

```
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' ||
'41000000077195641000000000000000000000000000000000 ↩
    FFFFFFFF050005000400FDFEFDFEFEFDFEFEFDF9FAFEF' ||
' ↩
    EFCF9FBFDFEFEFDFCFAFEFEFE04004E627AADD16076B4F9FE6370A9F5FE59637AB0E54F5861708704004656648 7A1506
    ')::raster);
```

11.1 Types de données pour la prise en charge raster

11.1.1 geomval

geomval — Un type spatial comportant deux champs : geom (stockant un objet géométrique) et val (stockant la valeur du pixel en double précision depuis une bande du raster).

Description

geomval est un type composite, comprenant un objet géométrique dans le champ .geom et val, une valeur en double précision qui représente la valeur du pixel à un emplacement spécifique d'une bande raster. Ce type est utilisé par la fonction ST_DumpAsPolygon et la famille de fonctions d'intersection Raster, en tant que type de sortie pour éclater une bande raster en polygones.

Voir aussi

Section 13.6

11.1.2 addbandarg

addbandarg — Un type composite utilisé comme entrée pour la fonction ST_AddBand pour définir les attributs et la valeur initiale d'une nouvelle bande.

Description

Un type composite utilisé comme entrée pour la fonction ST_AddBand pour définir les attributs et la valeur initiale d'une nouvelle bande.

`index integer` Indice (basé sur 1) où insérer la nouvelle bande dans la liste des bandes du raster. Si NULL, la nouvelle bande sera ajoutée en dernière position.

`pixeltype text` Type de pixel pour la nouvelle bande. Doit être un des types de pixels définis, tels que décrits dans ST_BandPixelType.

`initialvalue double precision` Valeur initiale pour tous les pixels de la nouvelle bande.

`nodataval double precision` Valeur NODATA pour la nouvelle bande. Si NULL, la nouvelle bande n'aura pas de valeur NODATA attribuée.

Voir aussi

ST_AddBand

11.1.3 rastbandarg

rastbandarg — Un type composite pour représenter à la fois un raster et un indice d'une bande de ce raster.

Description

Un type composite pour représenter à la fois un raster et un indice d'une bande de ce raster.

`rast raster` Le raster en question

`nband integer` Indice (basé sur 1) de la bande du raster

Voir aussi

ST_MapAlgebra (callback function version)

11.1.4 raster

raster — Type de données raster spatial.

Description

raster est le type de données spatial pour représenter les données rasters, telles qu'importées depuis des fichiers JPEGs, TIFFs, PNGs, modèles d'élévation numérique. Chaque raster a une ou plusieurs bandes, chacune correspondant à un ensemble de valeurs de pixels. Les rasters peuvent être géo-référencés.

Note
PostGIS doit être compilé avec le support GDAL. Actuellement, les rasters peuvent être implicitement convertis en type geometry, mais la conversion retourne un ST_ConvexHull du raster. Ce transtypage implicite peut être supprimé dans le futur proche et ne devrait pas être utilisé.

Transtypages

Cette section liste les transtypages autorisés pour ce type de donnée, qu'ils soient automatiques ou bien explicites

Transtypage vers	Comportement
geometry	automatique

Voir aussi

Chapter 11

11.1.5 reclassarg

reclassarg — Un type composite utilisé comme entrée pour la fonction ST_Reclass pour définir le comportement de la reclassification.

Description

Un type composite utilisé comme entrée pour la fonction ST_Reclass pour définir le comportement de la reclassification.

`nband integer` L'indice de bande à reclassifier.

`reclassexpr text` Expression des classes, sous forme de correspondance range:map_range séparés par des virgules pour définir les correspondances entre les anciennes valeurs de la bande et la nouvelle valeur. (signifie plus grand que,) signifie plus petit que,] signifie inférieur ou égal, [signifie supérieur ou égal

```
1. [a-b] = a <= x <= b

2. (a-b] = a < x <= b

3. [a-b) = a <= x < b

4. (a-b) = a < x < b
```

La notation (est facultative : a-b est équivalent à (a-b)

pixeltype text Un des types de pixels définis, tels que décrits dans ST_BandPixelType

nodataval double precision Valeur à utiliser comme NODATA. Pour les formats d'images supportant la transparence, ces pixels seront transparents.

Exemple : Reclassifier la bande 2 en 8BUI avec 255 comme valeur NODATA

```
SELECT ROW(2, '0-100:1-10, 101-500:11-150,501 - 10000: 151-254', '8BUI', 255)::reclassarg;
```

Exemple : Reclassifier la bande 1 en 1BB sans valeur NODATA

```
SELECT ROW(1, '0-100]:0, (100-255:1', '1BB', NULL)::reclassarg;
```

Voir aussi

ST_Reclass

11.1.6 summarystats

summarystats — Un type composite retournée par les fonctions ST_SummaryStats et ST_SummaryStatsAgg.

Description

Un type composite retournée par les fonctions ST_SummaryStats et ST_SummaryStatsAgg.

count integer Nombre de pixels comptabilisés pour les résumés statistiques.

sum double precision Somme des valeurs de tous les pixels comptabilisés.

mean double precision Moyenne arithmétique des valeurs de tous les pixels comptabilisés.

stddev double precision Écart type des valeurs de tous les pixels comptabilisés.

min double precision Minimum des valeurs de tous les pixels comptabilisés.

max double precision Maximum des valeurs de tous les pixels comptabilisés.

Voir aussi

ST_SummaryStats, ST_SummaryStatsAgg

11.1.7 unionarg

unionarg — Un type composite utilisé comme entrée de la fonction ST_Union pour définir les bandes à traiter et le comportement de l'opération d'union.

Description

Un type composite utilisé comme entrée de la fonction ST_Union pour définir les bandes à traiter et le comportement de l'opération d'union.

`nband integer` Indice (basé sur 1) indiquant la bande de chaque raster d'entrée à traiter.

`uniontype text` Type de l'opération d'union. Doit être un des types définis, tels que décrits dans ST_Union.

Voir aussi

ST_Union

11.2 Gestion raster

11.2.1 AddRasterConstraints

AddRasterConstraints — Ajoute des contraintes raster à une table pour une colonne spécifique pour contraindre le système de référence spatial, l'échelle, la taille des blocs, l'alignement, le nombre de bandes, le type de pixels, la contrainte d'unicité spatiale. La table doit être chargée avec des données pour que les contraintes puissent être inférées. Retourne true si les contraintes ont été ajoutées, ou émet une notice sinon.

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=fa
boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true , boolean out_db=true , boolean extent=true);

Description

Génère des contraintes pour une colonne raster, qui peuvent être utilisées pour afficher des informations depuis le catalogue raster `raster_columns`. `rastschema` est le nom du schéma de la table. `srid` doit être un entier défini dans la table SPATIAL_REF_SYS.

Le chargeur de rasters `raster2pgsql` utilise cette fonction pour enregistrer les tables raster

Noms de contraintes à passer en arguments : se référer à Section 10.2.1 pour plus de détails.

- `blocksize` défini la largeur en X et en Y
- `blocksize_x` défini la largeur (nombre de pixels) de chaque tuile raster
- `blocksize_y` défini la hauteur (nombre de pixels vers le bas) de chaque tuile raster
- `extent` calcule l'étendue de l'ensemble de la table et contraint tous les rasters à être inclus dans cette étendue
- `num_bands` nombre de bandes
- `pixel_types` tableau de types de pixels pour chaque bande - contraint chaque bande n au même type de pixels
- `regular_blocking` défini les contraintes d'unité spatiale (deux rasters ne peuvent pas être spatialement identiques) et de couverture de tuiles (raster est aligné sur une couverture)

- `same_alignment` garantie que toutes les tuiles ont le même alignement. Comparer n'importe quel couple de tuiles retournera true. Voir ST_SameAlignment pour plus de détails.
- `srid` garantie que tous les rasters ont le même srid
- Plus -- n'importe quelle contrainte listée parmi les arguments des fonctions décrites ci-dessus

Note
Cette fonction infère les contraintes depuis les données déjà présentes dans la table. Ainsi, vous devez d'abord créer la colonne raster et charger des données dedans avant de pouvoir utiliser cette fonction.

Note
Si vous avez besoin de charger plus de données dans la table après avoir défini les contraintes, vous pourrez avoir besoin d'exécuter la fonction DropRasterConstraints avant le chargement, en particulier si l'étendue des données est modifiée.

Disponibilité : 2.0.0

Exemples : Appliquer toutes les contraintes possibles à partir des données présentes

```
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);


-- verify if registered correctly in the raster_columns view --
SELECT srid, scale_x, scale_y, blocksize_x, blocksize_y, num_bands, pixel_types,  ↩
    nodata_values
    FROM raster_columns
    WHERE r_table_name = 'myrasters';

 srid | scale_x | scale_y | blocksize_x | blocksize_y | num_bands | pixel_types|  ↩
     nodata_values
------+---------+---------+-------------+-------------+-----------+-------------+--------------- ↩

 4326 |       2 |       2 |        1000 |        1000 |         1 | {8BSI}      | {0}
```

Exemples : Appliquer une seule contrainte

```
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');
-- get notice--
NOTICE:  Adding regular blocking constraint
NOTICE:  Adding blocksize-X constraint
NOTICE:  Adding blocksize-Y constraint
```

Voir aussi

Section 10.2.1, ST_AddBand, ST_MakeEmptyRaster, DropRasterConstraints, ST_BandPixelType, ST_SRID

11.2.2 DropRasterConstraints

DropRasterConstraints — Supprime des contraintes raster PostGIS raster sur une colonne raster d'une table. Utile pour recharger des données ou mettre à jour les données d'une colonne raster.

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=t
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=fa
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

Supprime des contraintes raster PostGIS ajoutées via AddRasterConstraints sur une colonne raster d'une table. Utile pour recharger des données ou mettre à jour les données d'une colonne raster. Vous n'avez pas besoin d'appeler cette fonction si vous voulez supprimer une table raster ou une colonne raster.

Pour supprimer une table raster, utilisez la requête classique

```
DROP TABLE mytable
```

Pour supprimer une colonne raster et garder le reste de la table intacte, utilisez la requête classique

```
ALTER TABLE mytable DROP COLUMN rast
```

La table disparaîtra du catalogue `raster_columns` si la colonne ou la table est supprimée. Si seulement les contraintes ont été supprimées, la colonne raster sera toujours listée dans le catalogue `raster_columns`, mais il n'y aura pas d'autre information que les noms de la colonne et de la table.

Disponibilité : 2.0.0

Exemples

```
SELECT DropRasterConstraints ('myrasters','rast');
----RESULT output ---
t

-- verify change in raster_columns --
SELECT srid, scale_x, scale_y, blocksize_x, blocksize_y, num_bands, pixel_types,  ↩
    nodata_values
    FROM raster_columns
    WHERE r_table_name = 'myrasters';

 srid | scale_x | scale_y | blocksize_x | blocksize_y | num_bands | pixel_types|  ↩
     nodata_values
------+---------+---------+-------------+-------------+-----------+-------------+--------------- ↩

    0 |         |         |             |             |           |             |
```

Voir aussi

AddRasterConstraints

11.2.3 AddOverviewConstraints

AddOverviewConstraints — Marque une colonne raster comme étant un aperçu d'une autre colonne.

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

Ajoute des contraintes sur une colonne raster, qui peuvent être utilisées pour afficher des informations depuis le catalogue raster `raster_overviews`.

La paramètre `ovfactor` représente le facture de multiplication d'échelle pour la colonne d'aperçu : plus le facteur d'aperçu est élevé, plus résolution est basse.

Si les paramètres `ovschema` et `refschema` sont omis, la première table trouvée lors du scan de `search_path` sera utilisée.

Disponibilité : 2.0.0

Exemples

```
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);

-- verify if registered correctly in the raster_overviews view --
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';
  ot  | oc |  rt  | rc | f
------+----+------+----+---
 res2 | r2 | res1 | r1 | 2
(1 row)
```

Voir aussi

Section 10.2.2, DropOverviewConstraints, ST_CreateOverview, AddRasterConstraints

11.2.4 DropOverviewConstraints

DropOverviewConstraints — Supprime les contraintes d'aperçu de la colonne.

Synopsis

boolean **DropOverviewConstraints**(name ovschema, name ovtable, name ovcolumn);
boolean **DropOverviewConstraints**(name ovtable, name ovcolumn);

Description

Supprime dans le catalogue raster `raster_overviews` les contraintes raster de la colonne indiquant que c'est un aperçu une autre colonne.

Si le paramètre `ovschema` est omis, la première table trouvée lors du scan de `search_path` sera utilisée.

Disponibilité : 2.0.0

Voir aussi

Section 10.2.2, AddOverviewConstraints, DropRasterConstraints

11.2.5 PostGIS_GDAL_Version

PostGIS_GDAL_Version — Retourne la version de la bibliothèque GDAL utilisée par PostGIS.

Synopsis

text **PostGIS_GDAL_Version**();

Description

Retourne la version de la bibliothèque GDAL utilisée par PostGIS. Vérifie également si GDAL peut accéder à ses fichiers de données et reportera une erreur si ce n'est pas le cas.

Exemples

```
SELECT PostGIS_GDAL_Version();
       postgis_gdal_version
-----------------------------------
 GDAL 1.11dev, released 2013/04/13
```

Voir aussi

postgis.gdal_datapath

11.2.6 PostGIS_Raster_Lib_Build_Date

PostGIS_Raster_Lib_Build_Date — Retourne la date de compilation de la bibliothèque raster.

Synopsis

text **PostGIS_Raster_Lib_Build_Date**();

Description

Retourne la date de compilation de la bibliothèque raster

Exemples

```
SELECT PostGIS_Raster_Lib_Build_Date();
postgis_raster_lib_build_date
------------------------------
2010-04-28 21:15:10
```

Voir aussi

PostGIS_Raster_Lib_Version

11.2.7 PostGIS_Raster_Lib_Version

PostGIS_Raster_Lib_Version — Retourne la version complète de la bibliothèque raster et les informations sur la configuration de la compilation.

Synopsis

text **PostGIS_Raster_Lib_Version**();

Description

Retourne la version complète de la bibliothèque raster et les informations sur la configuration de la compilation.

Exemples

```
SELECT PostGIS_Raster_Lib_Version();
postgis_raster_lib_version
------------------------------
 2.0.0
```

Voir aussi

PostGIS_Lib_Version

11.2.8 ST_GDALDrivers

ST_GDALDrivers — Retourne la liste des formats raster supportés par PostGIS via GDAL. Seuls les formats avec can_write=True peuvent être utilisés par 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

Retourne la liste des formats raster supportés par GDAL, sous forme short_name, long_name et creator. Vous pouvez utiliser short_name comme paramètre `format` pour ST_AsGDALRaster. Les options peuvent varier selon les drivers compilés dans votre bibliothèque libgdal. `create_options` retourne un XML contenant un ensemble de CreationOptionList/Option avec comme attribut name et comme attributs optionnels `type` et `description`, ainsi qu'un ensemble de `VALUE` pour chaque option de création pour ce driver.

Changement : 2.5.0 - ajout des colonnes can_read et can_write.

Changement : 2.0.6, 2.1.3 - aucun driver n'est activé par défaut, sauf si la variable GUC ou d'environnement gdal_enabled_drivers est définie.

Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0.

Exemples : Liste des drivers

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SELECT short_name, long_name, can_write
FROM st_gdaldrivers()
ORDER BY short_name;

   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
```

Exemple : Liste des options pour chaque driver

```
-- Output the create options XML column of JPEG as a table  --
-- Note you can use these creator options in ST_AsGDALRaster options argument
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;

       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)
```

```
-- raw xml output for creator options for GeoTiff --
SELECT create_options
FROM st_gdaldrivers()
WHERE short_name = 'GTiff';

<CreationOptionList>
    <Option name="COMPRESS" type="string-select">
        <Value
>NONE</Value>
        <Value
>LZW</Value>
```

```
        <Value
>PACKBITS</Value>
        <Value
>JPEG</Value>
        <Value
>CCITTRLE</Value>
        <Value
>CCITTFAX3</Value>
        <Value
>CCITTFAX4</Value>
        <Value
>DEFLATE</Value>
    </Option>
    <Option name="PREDICTOR" type="int" description="Predictor Type"/>
    <Option name="JPEG_QUALITY" type="int" description="JPEG quality 1-100" default="75"/>
    <Option name="ZLEVEL" type="int" description="DEFLATE compression level 1-9" default ↩
        ="6"/>
    <Option name="NBITS" type="int" description="BITS for sub-byte files (1-7), sub-uint16  ↩
        (9-15), sub-uint32 (17-31)"/>
    <Option name="INTERLEAVE" type="string-select" default="PIXEL">
        <Value
>BAND</Value>
        <Value
>PIXEL</Value>
    </Option>
    <Option name="TILED" type="boolean" description="Switch to tiled format"/>
    <Option name="TFW" type="boolean" description="Write out world file"/>
    <Option name="RPB" type="boolean" description="Write out .RPB (RPC) file"/>
    <Option name="BLOCKXSIZE" type="int" description="Tile Width"/>
    <Option name="BLOCKYSIZE" type="int" description="Tile/Strip Height"/>
    <Option name="PHOTOMETRIC" type="string-select">
        <Value
>MINISBLACK</Value>
        <Value
>MINISWHITE</Value>
        <Value
>PALETTE</Value>
        <Value
>RGB</Value>
        <Value
>CMYK</Value>
        <Value
>YCBCR</Value>
        <Value
>CIELAB</Value>
        <Value
>ICCLAB</Value>
        <Value
>ITULAB</Value>
    </Option>
    <Option name="SPARSE_OK" type="boolean" description="Can newly created files have  ↩
        missing blocks?" default="FALSE"/>
    <Option name="ALPHA" type="boolean" description="Mark first extrasample as being alpha ↩
        "/>
    <Option name="PROFILE" type="string-select" default="GDALGeoTIFF">
        <Value
>GDALGeoTIFF</Value>
        <Value
>GeoTIFF</Value>
        <Value
>BASELINE</Value>
    </Option>
```

```
    <Option name="PIXELTYPE" type="string-select">
        <Value
>DEFAULT</Value>
        <Value
>SIGNEDBYTE</Value>
    </Option>
    <Option name="BIGTIFF" type="string-select" description="Force creation of BigTIFF file ↩
        ">
        <Value
>YES</Value>
        <Value
>NO</Value>
        <Value
>IF_NEEDED</Value>
        <Value
>IF_SAFER</Value>
    </Option>
    <Option name="ENDIANNESS" type="string-select" default="NATIVE" description="Force  ↩
        endianness of created file. For DEBUG purpose mostly">
        <Value
>NATIVE</Value>
        <Value
>INVERTED</Value>
        <Value
>LITTLE</Value>
        <Value
>BIG</Value>
    </Option>
    <Option name="COPY_SRC_OVERVIEWS" type="boolean" default="NO" description="Force copy  ↩
        of overviews of source dataset (CreateCopy())"/>
</CreationOptionList>

-- Output the create options XML column for GTiff as a table  --
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,
       array_to_string(xpath('Value/text()', g.opt),', ') As vals
FROM (SELECT unnest(xpath('/CreationOptionList/Option', create_options::xml)) As opt
FROM  st_gdaldrivers()
WHERE short_name = 'GTiff') As g;

       oname        |     otype     |                              descrip  ↩
                                          |                                   vals
--------------------+---------------+-----------------------------------------------------------------

 COMPRESS           | string-select |  ↩
                                                                          | NONE, LZW,  ↩
     PACKBITS, JPEG, CCITTRLE, CCITTFAX3, CCITTFAX4, DEFLATE
 PREDICTOR          | int           | Predictor Type  ↩
                                                          |
 JPEG_QUALITY       | int           | JPEG quality 1-100  ↩
                                                       |
 ZLEVEL             | int           | DEFLATE compression level 1-9  ↩
                                            |
 NBITS              | int           | BITS for sub-byte files (1-7), sub-uint16 (9-15), sub ↩
     -uint32 (17-31) |
 INTERLEAVE         | string-select |  ↩
                                                                          | BAND, PIXEL
 TILED              | boolean       | Switch to tiled format  ↩
                                                   |
 TFW                | boolean       | Write out world file  ↩
                                                     |
```

```
 RPB                | boolean       | Write out .RPB (RPC) file  ↩
                                                                 |
 BLOCKXSIZE         | int           | Tile Width  ↩
                                                                     |
 BLOCKYSIZE         | int           | Tile/Strip Height  ↩
                                                                    |
 PHOTOMETRIC        | string-select |  ↩
                                                                                | MINISBLACK,  ↩
    MINISWHITE, PALETTE, RGB, CMYK, YCBCR, CIELAB, ICCLAB, ITULAB
 SPARSE_OK          | boolean       | Can newly created files have missing blocks?  ↩
                                |
 ALPHA              | boolean       | Mark first extrasample as being alpha  ↩
                                   |
 PROFILE            | string-select |  ↩
                                                                                | GDALGeoTIFF,  ↩
    GeoTIFF, BASELINE
 PIXELTYPE          | string-select |  ↩
                                                                                | DEFAULT,  ↩
    SIGNEDBYTE
 BIGTIFF            | string-select | Force creation of BigTIFF file  ↩
                                             | YES, NO, IF_NEEDED, IF_SAFER
 ENDIANNESS         | string-select | Force endianness of created file. For DEBUG purpose  ↩
    mostly            | NATIVE, INVERTED, LITTLE, BIG
 COPY_SRC_OVERVIEWS | boolean       | Force copy of overviews of source dataset (CreateCopy ↩
    ())             |
(19 rows)
```

Voir aussi

ST_AsGDALRaster, ST_SRID, postgis.gdal_enabled_drivers

11.2.9 ST_Contour

ST_Contour — Génère un ensemble de courbes de niveau vectorielles depuis la bande raster spécifiée, en utilisant l'algorithme de contour GDAL.

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

Génère un ensemble de courbes de niveau vectorielles depuis la bande raster spécifiée, en utilisant l'algorithme de contour GDAL.

Si le paramètre `fixed_levels` est un tableau non vide, les paramètres `level_interval` et `level_base` sont ignorés.

Les paramètres d'entrée sont :

`rast` Le raster pour générer le contour de

`bandnumber` La bande pour générer le contour de

`level_interval` Intervalle d'élévation entre les courbes de niveau générées

level_base La "base" par rapport à laquelle les intervalles de contour sont appliqués ; elle est normalement égale à zéro, mais peut être différente. Pour générer des courbes de niveau de 10 m à 5, 15, 25, ... le LEVEL_BASE serait de 5.

fixed_levels Intervalle d'élévation entre les courbes de niveau générées

polygonize Si `true`, des polygones de contour seront créés, plutôt que des lignes de polygone.

La valeur de retour est un ensemble de résultats avec les attributs suivants :

geom La géométrie de la ligne de contour.

id Un identifiant unique de la ligne de contour, déterminée par GDAL.

value La valeur raster représentée par la ligne. Pour un modèle numérique de terrain, la valeur correspond à l'élévation du contour.

Disponibilité : 3.2.0

Exemple

```
WITH c AS (
SELECT (ST_Contour(rast, 1, fixed_levels => ARRAY[100.0, 200.0, 300.0])).*
FROM dem_grid WHERE rid = 1
)
SELECT st_astext(geom), id, value
FROM c;
```

Voir aussi

ST_InterpolateRaster

11.2.10 ST_InterpolateRaster

ST_InterpolateRaster — Interpole une surface quadrillée à partir d'un ensemble de points 3-d, en utilisant les coordonnées X et Y des points sur la grille et la coordonnée Z des points pour l'élévation des points.

Synopsis

raster **ST_InterpolateRaster**(geometry input_points, text algorithm_options, raster template, integer template_band_num=1);

Description

Interpole une surface quadrillée à partir d'un ensemble de points 3-d, en utilisant les coordonnées X et Y des points sur la grille et la coordonnée Z des points pour l'élévation des points. 5 algorithmes d'interpolations sont disponibles : inverse de la distance, inverse de la distance avec plus proche voisin, moyenne mobile, plus proche voisin, et interpolation linéaire. Voir gdal_grid documentation pour plus de détails sur ces algorithmes et leurs paramètres. Pour plus d'informations sur le calcul des interpolations, voir le tutorial GDAL grid.

Les paramètres d'entrée sont :

input_points Les points pour effectuer l'interpolation. Toute géometrie avec une dimension Z est acceptée ; tous les points de la géométrie seront utilisés.

algorithm_options Une chaîne définissant l'algorithme et ses options, au format utilisable par gdal_grid. Par exemple, pour une interpolation en inverse de la distance avec un lissage de 2, la valeur sera "invdist:smoothing=2.0"

template Un gabarit raster pour la création du raster de sortie. Les largeur, hauteur, taille des pixels, étendue spatiale et type des pixels seront définis à partir de ce gabarit.

template_band_num Indice de la bande raster du gabarit raster à utiliser ; par défaut, la première bande sera utilisée.

Disponibilité : 3.2.0

Exemple

```
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')
)
```

Voir aussi

ST_Contour

11.2.11 UpdateRasterSRID

UpdateRasterSRID — Change le SRID de tous les rasters dans la table et colonne en paramètres.

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

Change le SRID de tous les rasters dans la table et colonne en paramètres. Cette fonction supprimera toutes les contraintes nécessaires (étendue, alignement et SRID) avant de changer le SRID de la colonne raster.

Note
Les données (les valeurs des pixels) des rasters ne sont pas changées par cette fonction. Seules les métadonnées du rasters sont modifiées.

Disponibilité : 2.1.0

Voir aussi

UpdateGeometrySRID

11.2.12 ST_CreateOverview

ST_CreateOverview — Créé une version à plus faible résolution d'une colonne raster.

Synopsis

regclass **ST_CreateOverview**(regclass tab, name col, int factor, text algo='NearestNeighbor');

Description

Créé une table d'aperçu avec les tuiles rééchantillonnées depuis la table source. Les tuiles de sortie auront la même taille que les tuiles d'entrées et couvriront la même étendue, mais avec une résolution plus faible (la taille d'un pixel sera 1/`factor` dans les deux directions).

La table d'aperçu sera disponible dans le catalogue `raster_overviews` avec les contraintes rasters ajoutées.

Les algorithmes disponibles sont : 'NearestNeighbor' (plus proche voisin), 'Bilinear' (Bilinéaire), 'Cubic' (Cubique), 'CubicSpline' (Cubique Spline) et 'Lanczos'. Voir GDAL Warp resampling methods pour plus de détails.

Disponibilité : 2.2.0

Exemple

Sortie de meilleure qualité en général, mais plus lent à générer

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2, 'Lanczos');
```

Sortie plus rapide, via la méthode de plus proche voisin

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2);
```

Voir aussi

ST_Retile, AddOverviewConstraints, AddRasterConstraints, Section 10.2.2

11.3 Constructeurs de raster

11.3.1 ST_AddBand

ST_AddBand — Retourne un raster avec la/les nouvelle(s) bande(s) ajouté(s) à un index donné, de type et valeur initiale donnés. Si aucun index n'est spécifié, la bande est ajoutée à la fin.

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

Renvoie un raster avec une nouvelle bande ajoutée à la position donnée (index), du type donné, de la valeur initiale donnée et de la valeur de nodata donnée. Si aucun index n'est spécifié, la bande est ajoutée à la fin. Si aucun `fromband` n'est spécifié, la bande 1 est supposée. Le type de pixel est une représentation sous forme de chaîne de l'un des types de pixels spécifiés dans ST_BandPixelType. Si un index existant est spécifié, toutes les bandes suivantes >= cet index sont incrémentées de 1. Si une valeur initiale supérieure à la valeur maximale du type de pixel est spécifiée, la valeur initiale est fixée à la valeur la plus élevée autorisée par le type de pixel.

Pour la variante acceptant un tableau de addbandarg en paramètre (Variante 1), un index spécifique au paramètre addbandarg est relative au raster au moment où la bande décrite par ce paramètre addbandarg est ajoutée au raster. Voir l'exemple Multiples nouvelles bandes ci-dessous.

Pour la variante acceptant un tableau de rasters (Variante 5), si `torast` est NULL, alors la bande `fromband` de chaque raster est accumulée dans le nouveau raster.

Pour les variantes acceptant un paramètre `outdbfile` (Variantes 6 et 7), la valeur doit correspondre au chemin complet vers le fichier raster. Ce fichier doit être accessible par le processus du serveur postgres.

Amélioration : 2.1.0 ajout du paramètre addbandarg.

Amélioration : 2.1.0 introduction du support des bandes out-db.

Exemples : Nouvelle bande

```
-- Add another band of type 8 bit unsigned integer with pixels initialized to 200
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
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[]
     )
    );

-- output meta data of raster bands to verify all is right --
SELECT  (bmd).*
FROM (SELECT ST_BandMetaData(rast,generate_series(1,2)) As bmd
    FROM dummy_rast WHERE rid = 10) AS foo;
 --result --
 pixeltype | nodatavalue | isoutdb | path
-----------+-------------+---------+------
 1BB       |             | f       |
 4BUI      |             | f       |


-- output meta data of raster -
SELECT  (rmd).width, (rmd).height, (rmd).numbands
FROM (SELECT ST_MetaData(rast) As rmd
    FROM dummy_rast WHERE rid = 10) AS foo;
-- result --
 upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↩
     numbands
------------+------------+-------+--------+--------+--------+-------+-------+------+--------

          0 |          0 |   100 |    100 |      1 |     -1 |     0 |     0 |   0 |         ↩
              2
```

Exemples : Multiples nouvelles bandes

```
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[]
);

 bandnum | pixeltype | nodatavalue | isoutdb | path
---------+-----------+-------------+---------+------
       1 | 8BUI      |           0 | f       |
       2 | 32BF      |          -1 | f       |
       3 | 32BUI     |          12 | f       |
       4 | 16BUI     |           2 | f       |
```

```
-- 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: The ORDER BY test_type is only supported in PostgreSQL 9.0+
-- for 8.4 and below it usually works to order your data in a subselect (but not guaranteed ↩
    )
-- 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
SELECT
    mouse,
    ST_AddBand(NULL, array_agg(rast ORDER BY test_type), 1) As rast
FROM mice_studies
GROUP BY mouse;
```

Exemples : Nouvelle bande Out-db

```
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[]
);

 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
```

Voir aussi

ST_BandMetaData, ST_BandPixelType, ST_MakeEmptyRaster, ST_MetaData, ST_NumBands, ST_Reclass

11.3.2 ST_AsRaster

ST_AsRaster — Convertit une géométrie PostGIS en un raster PostGIS.

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

Convertit une géométrie PostGIS en un raster PostGIS. Les nombreuses variantes permettent trois groupes de possibilités pour définir l'alignement et la taille des pixels du raster en sortie.

Le premier groupe, constitué des deux premières variantes, produit un raster avec le même alignement (`scalex`, `scaley`, `gridx` et `gridy`), le même type de pixels et la même valeur nodata que le raster donné en référence. En général, vous passerez cette référence en faisant une jointure entre la table contenant la géométrie avec la table contenant le raster de référence.

Le second groupe, composé de quatre variantes, vous permettent de spécifier les dimensions du raster via les paramètres de taille de pixel (`scalex` & `scaley` et `skewx` & `skewy`). Les dimensions `width` & `height` du raster résultant seront ajustés pour s'adapter à l'étendu de la géométrie. Dans la plupart des cas, vous devrez transtyper les paramètres `scalex` & `scaley` de integer à double precision pour que PostgreSQL choisisse la variante correcte.

Le troisième groupe, composé de quatre variantes, vous permettent de spécifier les dimensions du raster directement (`width` & `height`). Les paramètres de taille de pixel (`scalex` & `scaley` et `skewx` & `skewy`) du raster résultant seront ajustés pour s'adapter à l'étendue de la géométrie.

Les deux premières variantes des deux derniers groupes vous permettent de spécifier l'alignement avec un coin arbitraire de la grille d'alignement (`gridx` & `gridy`) et les deux dernières variantes via le coin haut-gauche (`upperleftx` & `upperlefty`).

Chaque groupe de variantes permet de produire un raster à une ou plusieurs bandes. Pour plusieurs bandes, vous devez fournir un tableau de types de pixels (`pixeltype[]`), un tableau de valeurs initiales (`value`) et un tableau de valeurs nodata (`nodataval`). Si non fournis, pixeltype est par défaut à 8BUI, value à 1 et nodataval à 0.

Le raster en sortie aura le même système de référence spatial que la géométrie source, sauf pour les variantes utilisant un raster de référence. Dans ce cas, le raster en sortie aura le même SRID que le raster de référence.

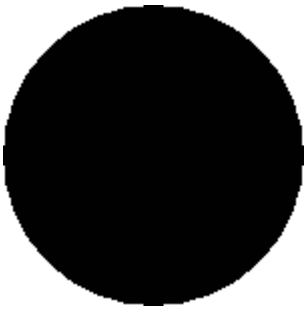
Le paramètre optionnel `touched` est par défaut à false et correspond à l'option GDAL ALL_TOUCHED, qui détermine si les pixels touchés par des lignes ou polygones seront mis à jour, et pas seulement ceux sur le chemin de la ligne ou dont le point central est dans le polygone.

Ceci est particulièrement utile pour le rendu de JPEGs et de PNGs directement depuis la base de données en utilisant une combinaison de ST_AsPNG et des fonctions de la famille de ST_AsGDALRaster.

Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0.

Note
Pas encore en mesure de faire le rendu de géométries complexes comme les CURVES, TIN et PolyhedralSurfaces, mais cela devrait être le cas lorsque GDAL le supportera.

Exemples : Rendu de géométries en fichiers PNG



cercle noir

```
-- this will output a black circle taking up 150 x 150 pixels --
SELECT ST_AsPNG(ST_AsRaster(ST_Buffer(ST_Point(1,5),10),150, 150));
```

exemple d'un buffer rendu via PostGIS

```
-- the bands map to RGB bands - the value (118,154,118) - teal  --
SELECT ST_AsPNG(
    ST_AsRaster(
        ST_Buffer(
            ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'), 10,'join=bevel'),
            200,200,ARRAY['8BUI', '8BUI', '8BUI'], ARRAY[118,154,118], ARRAY[0,0,0]));
```

Voir aussi

ST_BandPixelType, ST_Buffer, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsJPEG, ST_SRID

11.3.3 ST_Band

ST_Band — Retourne une ou plusieurs bandes d'un raster existant en tant que nouveau raster. Utile pour construire de nouveaux rasters à partir de rasters existants.

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

Retourne une ou plusieurs bandes d'un raster existant en tant que nouveau raster. Utile pour construire de nouveaux rasters à partir de rasters existants ou pour exporter certaines bandes d'un raster, ou pour réarranger l'ordre des bandes d'un raster. Si aucune bande n'est spécifiée, ou si aucune des bandes spécifiées n'existent dans le raster, alors toutes les bandes sont retournées. Cette fonction est utilisée en interne par diverses fonctions, comme pour supprimer une bande.

Warning
Pour les variantes avec le paramètre `nbands` en texte, le délimiteur par défaut est `,`, il faut donc passer par exemple `'1,2,3'`. Pour utiliser un autre délimiteur, utilisez le paramètre delimiter : `ST_Band(rast, '1@2@3', '@')`. Néanmoins, pour utiliser plusieurs bandes, nous encourageons fortement d'utiliser la variante avec le paramètre de type tableau (e.g. `ST_Band(rast, '{1,2,3}'::int[]);`); la variante avec la liste de bandes en `text` peut être supprimée des prochaines versions de PostGIS.

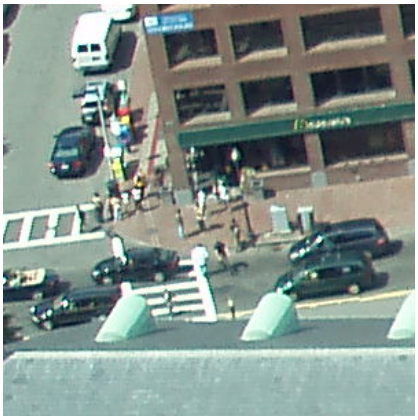
Disponibilité : 2.0.0

Exemples

```
-- Make 2 new rasters: 1 containing band 1 of dummy, second containing band 2 of dummy and ↩
    then reclassified as a 2BUI
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', '2 ↩
        BUI') As rast2
        FROM dummy_rast
        WHERE rid = 2) As foo;

 numb1 | pix1 | numb2 | pix2
-------+------+-------+------
     1 | 8BUI |     1 | 2BUI
```

```
-- Return bands 2 and 3. Using array cast syntax
SELECT ST_NumBands(ST_Band(rast, '{2,3}'::int[])) As num_bands
    FROM dummy_rast WHERE rid=2;

num_bands
----------
2
```

```
-- Return bands 2 and 3. Use array to define bands
SELECT ST_NumBands(ST_Band(rast, ARRAY[2,3])) As num_bands
    FROM dummy_rast
WHERE rid=2;
```

		
originale (colonne rast)	*dupe_band*	*sing_band*

```
--Make a new raster with 2nd band of original and 1st band repeated twice,
and another with just the third band
SELECT rast, ST_Band(rast, ARRAY[2,1,1]) As dupe_band,
    ST_Band(rast, 3) As sing_band
FROM samples.than_chunked
WHERE rid=35;
```

Voir aussi

ST_AddBand, ST_NumBands, ST_Reclass, Chapter 11

11.3.4 ST_MakeEmptyCoverage

ST_MakeEmptyCoverage — Couvre une zone géo-référencée avec une grille raster de tuiles vides.

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

Créé un ensemble de tuiles rasters via ST_MakeEmptyRaster. La dimension de la grille est définie par `width` & `height`. La dimension des tuiles est définie par `tilewidth` & `tileheight`. La zone géo-référencée couverte démarre au coin supérieur gauche (`upperleftx`, `upperlefty`) jusqu'au coin inférieur droit (`upperleftx` + `width` * `scalex`, `upperlefty` + `height` * `scaley`).

Note
A noter que scaley est généralement négatif pour les rasters, alors que scalex est généralement positif. Le coin inférieur droit aura une valeur Y inférieure et une valeur X supérieure au coin supérieur gauche.

Disponibilité : 2.4.0

Exemples de base

Créer 16 tuiles de 1x1px dans une grille 4x4 pour couvrir une zone WGS84 du coin supérieur gauche (22, 77) jusqu'au coin inférieur droit (55, 33).

```
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;

 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
```

Voir aussi

ST_MakeEmptyRaster

11.3.5 ST_MakeEmptyRaster

ST_MakeEmptyRaster — Retourne un raster vide (sans aucune bande) de dimension donnée (width & height), de coin supérieur gauche à X et Y, de paramètres de taille de pixel données (scalex, scaley, skewx & skewy) et de système de référence spatial (srid) donné. Si un raster est spécifié, retourne un nouveau raster de même taille, alignement et SRID. Si srid n'est pas spécifié, le système de référence spatial est défini à inconnu (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

Retourne un raster vide (sans aucune bande) de dimension donnée (width & height), géo-référencé via le coin supérieur gauche (upperleftx & upperlefty), via les paramètres de taille de pixel données (scalex, scaley, skewx & skewy) et via le système de référence spatial (srid) donné.

La dernière variante utilise un seul paramètre pour spécifier la taille des pixels (pixelsize). scalex est défini à la valeur de ce paramètre et scaley est défini à la valeur négative de ce paramètre. skewx et skewy sont définis à 0.

Si un raster existant est spécifié, retourne un nouveau raster avec les mêmes métadonnées (sans aucune bande).

Si aucun srid n'est spécifié, la valeur 0 est utilisée. Après avoir créé un raster vide, vous voudrez probablement ajouter des bandes ou l'éditer. Voir ST_AddBand pour définir des bandes et ST_SetValue pour initialiser des valeurs.

Exemples

```
INSERT INTO dummy_rast(rid,rast)
VALUES(3, ST_MakeEmptyRaster( 100, 100, 0.0005, 0.0005, 1, 1, 0, 0, 4326) );

--use an existing raster as template for new raster
INSERT INTO dummy_rast(rid,rast)
SELECT 4, ST_MakeEmptyRaster(rast)
FROM dummy_rast WHERE rid = 3;

-- output meta data of rasters we just added
SELECT rid, (md).*
FROM (SELECT rid, ST_MetaData(rast) As md
    FROM dummy_rast
    WHERE rid IN(3,4)) As foo;

-- output --
 rid | upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↩
     numbands
-----+------------+------------+-------+--------+------------+------------+-------+-------+------+--

   3 |     0.0005 |     0.0005 |   100 |    100 |          1 |          1 |    0  |     0 | ↩
       4326 |        0
   4 |     0.0005 |     0.0005 |   100 |    100 |          1 |          1 |    0  |     0 | ↩
       4326 |        0
```

Voir aussi

ST_AddBand, ST_MetaData, ST_ScaleX, ST_ScaleY, ST_SetValue, ST_SkewX, , ST_SkewY

11.3.6 ST_Tile

ST_Tile — Retourne un ensemble de rasters issus de la division d'un raster d'entrée selon les dimensions spécifiées pour les rasters de sortie.

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

Retourne un ensemble de rasters issus de la division d'un raster d'entrée selon les dimensions spécifiées pour les rasters de sortie.

Si `padwithnodata` est FALSE, les tuiles des bords droits et bas peuvent avoir des dimensions différentes du reste des tuiles. Si `padwithnodata` est TRUE, toutes les tuiles auront les mêmes dimensions, potentiellement avec les tuiles des bords complétées par la valeur NODATA. Si la bande raster n'a pas de valeur NODATA définie, cette valeur peut être spécifiée via le paramètre `nodataval`.

Note
Si une bande spécifiée du raster d'entrée est out-of-db, la bande correspondant dans les rasters de sortie sera aussi out-of-db.

Disponibilité : 2.1.0

Exemples

```
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 ST_Tile(rast, 3, 3, TRUE) AS rast FROM bar
```

```
)
SELECT
    ST_DumpValues(rast)
FROM baz;

              st_dumpvalues
----------------------------------------
 (1,"{{1,1,1},{1,1,1},{1,1,1}}")
 (2,"{{10,10,10},{10,10,10},{10,10,10}}")
 (1,"{{2,2,2},{2,2,2},{2,2,2}}")
 (2,"{{20,20,20},{20,20,20},{20,20,20}}")
 (1,"{{3,3,3},{3,3,3},{3,3,3}}")
 (2,"{{30,30,30},{30,30,30},{30,30,30}}")
 (1,"{{4,4,4},{4,4,4},{4,4,4}}")
 (2,"{{40,40,40},{40,40,40},{40,40,40}}")
 (1,"{{5,5,5},{5,5,5},{5,5,5}}")
 (2,"{{50,50,50},{50,50,50},{50,50,50}}")
 (1,"{{6,6,6},{6,6,6},{6,6,6}}")
 (2,"{{60,60,60},{60,60,60},{60,60,60}}")
 (1,"{{7,7,7},{7,7,7},{7,7,7}}")
 (2,"{{70,70,70},{70,70,70},{70,70,70}}")
 (1,"{{8,8,8},{8,8,8},{8,8,8}}")
 (2,"{{80,80,80},{80,80,80},{80,80,80}}")
 (1,"{{9,9,9},{9,9,9},{9,9,9}}")
 (2,"{{90,90,90},{90,90,90},{90,90,90}}")
(18 rows)
```

```
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 ST_Tile(rast, 3, 3, 2) AS rast FROM bar
)
SELECT
    ST_DumpValues(rast)
FROM baz;

              st_dumpvalues
----------------------------------------
 (1,"{{10,10,10},{10,10,10},{10,10,10}}")
 (1,"{{20,20,20},{20,20,20},{20,20,20}}")
 (1,"{{30,30,30},{30,30,30},{30,30,30}}")
```

```
 (1,"{{40,40,40},{40,40,40},{40,40,40}}")
 (1,"{{50,50,50},{50,50,50},{50,50,50}}")
 (1,"{{60,60,60},{60,60,60},{60,60,60}}")
 (1,"{{70,70,70},{70,70,70},{70,70,70}}")
 (1,"{{80,80,80},{80,80,80},{80,80,80}}")
 (1,"{{90,90,90},{90,90,90},{90,90,90}}")
(9 rows)
```

Voir aussi

ST_Union, ST_Retile

11.3.7 ST_Retile

ST_Retile — Retourne un ensemble de tuiles configurées à partir d'une couverture raster composée de tuiles arbitraires.

Synopsis

SETOF raster **ST_Retile**(regclass tab, name col, geometry ext, float8 sfx, float8 sfy, int tw, int th, text algo='NearestNeighbor');

Description

Retourne un ensemble de tuiles ayant l'échelle spécifiée (`sfx`, `sfy`), de taille maximale (`tw`, `th`) et couvrant l'étendue spécifiée (`ext`), et avec les données provenant de la couverture raster spécifiée (`tab`, `col`).

Les algorithmes disponibles sont : 'NearestNeighbor' (plus proche voisin), 'Bilinear' (Bilinéaire), 'Cubic' (Cubique), 'CubicSpline' (Cubique Spline) et 'Lanczos'. Voir GDAL Warp resampling methods pour plus de détails.

Disponibilité : 2.2.0

Voir aussi

ST_CreateOverview

11.3.8 ST_FromGDALRaster

ST_FromGDALRaster — Retourne un raster depuis un fichier raster supporté par GDAL.

Synopsis

raster **ST_FromGDALRaster**(bytea gdaldata, integer srid=NULL);

Description

Retourne un raster depuis un fichier raster supporté par GDAL. Le paramètre `gdaldata` de type bytea est le contenu du fichier raster GDAL.

Si `srid` est NULL, la fonction essaye de déterminer le SRID depuis le raster GDAL. Si `srid` est spécifié, la valeur spécifiée écrase tout SRID déterminé automatiquement.

Disponibilité : 2.1.0

Exemples

```
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;

 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)
```

Voir aussi

ST_AsGDALRaster

11.4 Fonctions d'accès aux rasters

11.4.1 ST_GeoReference

ST_GeoReference — Retourne les méta-données de géo-référencement au format GDAL (par défaut) ou ESRI, tel qu'utilisé généralement dans les fichiers world file.

Synopsis

text **ST_GeoReference**(raster rast, text format=GDAL);

Description

Retourne les méta-données de géo-référencement au format GDAL ou ESRI, tel qu'utilisé généralement dans les fichiers world file. Si aucun type n'est spécifié, utilise le format GDAL par défaut. type est une chaîne 'GDAL' ou 'ESRI'.

La différence entre les représentations de format est la suivante :

GDAL :

```
scalex
skewy
skewx
scaley
upperleftx
upperlefty
```

ESRI :

```
scalex
skewy
skewx
scaley
upperleftx + scalex*0.5
upperlefty + scaley*0.5
```

Exemples

```
SELECT ST_GeoReference(rast, 'ESRI') As esri_ref, ST_GeoReference(rast, 'GDAL') As gdal_ref
 FROM dummy_rast WHERE rid=1;

   esri_ref    |   gdal_ref
---------------+--------------
 2.0000000000 | 2.0000000000
 0.0000000000 : 0.0000000000
 0.0000000000 : 0.0000000000
 3.0000000000 : 3.0000000000
 1.5000000000 : 0.5000000000
 2.0000000000 : 0.5000000000
```

Voir aussi

ST_SetGeoReference, ST_ScaleX, ST_ScaleY

11.4.2 ST_Height

ST_Height — Retourne la hauteur du raster en pixels.

Synopsis

integer **ST_Height**(raster rast);

Description

Retourne la hauteur du raster en pixels.

Exemples

```
SELECT rid, ST_Height(rast) As rastheight
FROM dummy_rast;

 rid | rastheight
-----+------------
   1 |         20
   2 |          5
```

Voir aussi

ST_Width

11.4.3 ST_IsEmpty

ST_IsEmpty — Retourne true si le raster est vide (largeur = 0 et hauteur = 0). Sinon, retourne false.

Synopsis

boolean **ST_IsEmpty**(raster rast);

Description

Retourne true si le raster est vide (largeur = 0 et hauteur = 0). Sinon, retourne false.

Disponibilité : 2.0.0

Exemples

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(100, 100, 0, 0, 0, 0, 0, 0))
st_isempty |
-----------+
f          |


SELECT ST_IsEmpty(ST_MakeEmptyRaster(0, 0, 0, 0, 0, 0, 0, 0))
st_isempty |
-----------+
t          |
```

Voir aussi

ST_HasNoBand

11.4.4 ST_MemSize

ST_MemSize — Retourne l'espace utilisé par le raster (en octets).

Synopsis

integer **ST_MemSize**(raster rast);

Description

Retourne l'espace utilisé par le raster (en octets).

Cette fonction est un bon complément aux fonctions PostgreSQL natives pg_column_size, pg_size_pretty, pg_relation_size, pg_total_relation_size.

Note

pg_relation_size qui donne la taille en octets d'une table peut renvoyer une taille en octets inférieure à ST_MemSize. Cela est dû au fait que pg_relation_size n'ajoute pas la contribution des tables toasted et que les grandes géométries sont stockées dans les tables TOAST. pg_column_size peut retourner une valeur inférieure parce qu'elle retourne la taille compressée.
pg_total_relation_size - comprend la table, les tables toasted et les index.

Disponibilité : 2.2.0

Exemples

```
SELECT ST_MemSize(ST_AsRaster(ST_Buffer(ST_Point(1,5),10,1000),150, 150, '8BUI')) As  ↩
    rast_mem;

        rast_mem
        --------
        22568
```

Voir aussi

11.4.5 ST_MetaData

ST_MetaData — Retourne les méta-données de base de l'objet raster : taille des pixels, rotation, coin haut/bas gauche, etc.

Synopsis

record **ST_MetaData**(raster rast);

Description

Retourne les méta-données de base de l'objet raster : taille des pixels, rotation, coin haut/bas gauche, etc. Les colonnes retournées sont : upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands

Exemples

```
SELECT rid, (foo.md).*
 FROM (SELECT rid, ST_MetaData(rast) As md
FROM dummy_rast) As foo;

 rid | upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↩
     numbands
 ----+------------+------------+-------+--------+--------+-----------+-------+-------+------+-------

   1 |        0.5 |        0.5 |    10 |     20 |      2 |       3 |     0 |     0 |    0 |  ↩
             0
   2 | 3427927.75 |    5793244 |     5 |      5 |   0.05 |  -0.05 |     0 |     0 |    0 |  ↩
             3
```

Voir aussi

ST_BandMetaData, ST_NumBands

11.4.6 ST_NumBands

ST_NumBands — Retourne le nombre de bandes de l'objet raster.

Synopsis

integer **ST_NumBands**(raster rast);

Description

Retourne le nombre de bandes de l'objet raster.

Exemples

```
SELECT rid, ST_NumBands(rast) As numbands
FROM dummy_rast;

rid | numbands
----+----------
  1 |        0
  2 |        3
```

Voir aussi

ST_Value

11.4.7 ST_PixelHeight

ST_PixelHeight — Retourne la hauteur d'un pixel, dans l'unité du système de référence spatial.

Synopsis

double precision **ST_PixelHeight**(raster rast);

Description

Retourne la hauteur d'un pixel, dans l'unité du système de référence spatial. Dans le cas général où il n'y a pas d'obliquité (skew), la hauteur d'un pixel est simplement l'échelle entre les coordonnées géométriques et les pixels rasters.

Voir ST_PixelWidth pour une représentation schématique de la relation.

Exemples : Rasters sans obliquité (skew)

```
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;

 rastheight | pixheight | scalex | scaley | skewx | skewy
------------+-----------+--------+--------+-------+----------
         20 |         3 |      2 |      3 |     0 |        0
          5 |      0.05 |   0.05 |  -0.05 |     0 |        0
```

Exemples : Rasters avec obliquité (skew) différent de 0

```
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;
```

```
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
```

Voir aussi

ST_PixelWidth, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.4.8 ST_PixelWidth

ST_PixelWidth — Retourne la largeur d'un pixel, dans l'unité du système de référence spatial.

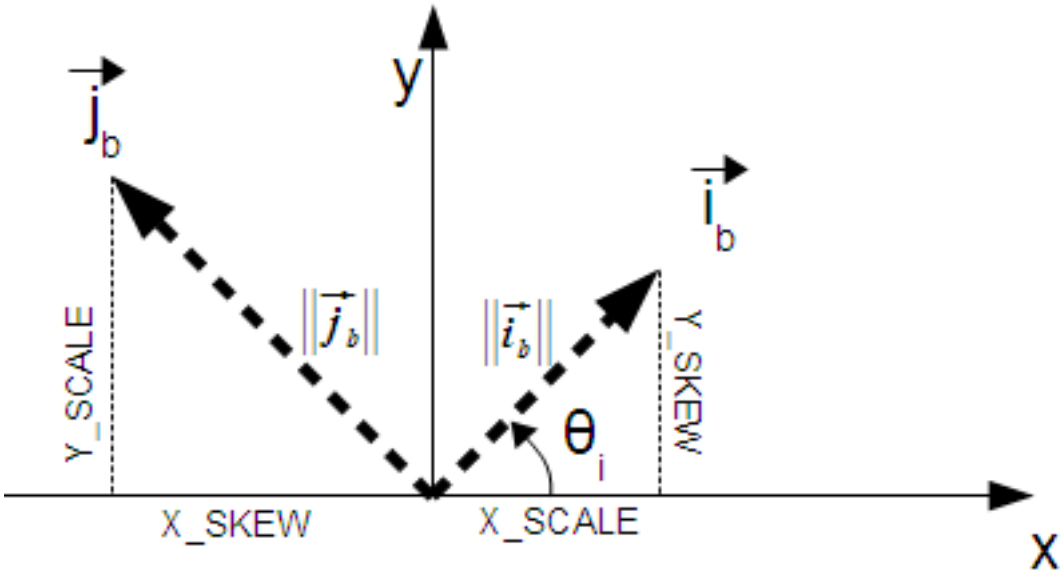
Synopsis

double precision **ST_PixelWidth**(raster rast);

Description

Retourne la largeur d'un pixel, dans l'unité du système de référence spatial. Dans le cas général où il n'y a pas d'obliquité (skew), la largeur d'un pixel est simplement l'échelle entre les coordonnées géométriques et les pixels rasters.

Le diagramme suivant illustre cette relation :



Largeur du pixel : taille du pixel dans la direction i
Hauteur du pixel : taille du pixel dans la direction j

Exemples : Rasters sans obliquité (skew)

```
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;
```

```
    rastwidth | pixwidth | scalex | scaley | skewx | skewy
    ----------+----------+--------+--------+-------+----------
    10 |        2 |      2 |      3 |     0 |        0
     5 |     0.05 |   0.05 |  -0.05 |     0 |        0
```

Exemples : Rasters avec obliquité (skew) différent de 0

```
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;

    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
```

Voir aussi

ST_PixelHeight, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.4.9 ST_ScaleX

ST_ScaleX — Renvoie la composante X de la largeur du pixel dans l'unité du système de référence spatial.

Synopsis

float8 **ST_ScaleX**(raster rast);

Description

Renvoie la composante X de la largeur du pixel dans l'unité du système de référence spatial. Voir World File pour plus de détails.

Changement : 2.0.0. Dans les versions WKTRaster, cette fonction était appelée ST_PixelSizeX.

Exemples

```
SELECT rid, ST_ScaleX(rast) As rastpixwidth
FROM dummy_rast;

 rid | rastpixwidth
-----+--------------
   1 |            2
   2 |         0.05
```

Voir aussi

ST_Width

11.4.10 ST_ScaleY

ST_ScaleY — Renvoie la composante Y de la hauteur du pixel dans l'unité du système de référence spatial.

Synopsis

float8 **ST_ScaleY**(raster rast);

Description

Renvoie la composante Y de la hauteur du pixel dans l'unité du système de référence spatial. Peut être négative. Voir World File pour plus de détails.

Changement : 2.0.0. Dans les versions WKTRaster, cette fonction était appelée ST_PixelSizeY.

Exemples

```
SELECT rid, ST_ScaleY(rast) As rastpixheight
FROM dummy_rast;

 rid | rastpixheight
-----+---------------
   1 |             3
   2 |         -0.05
```

Voir aussi

ST_Height

11.4.11 ST_RasterToWorldCoord

ST_RasterToWorldCoord — Retourne le coin supérieur gauche du raster, sous forme de coordonnées X et Y (longitude et latitude) d'un colonne et d'une ligne. Les numéros de colonne et de ligne commencent à 1.

Synopsis

record **ST_RasterToWorldCoord**(raster rast, integer xcolumn, integer yrow);

Description

Retourne le coin supérieur gauche du raster, sous forme de coordonnées X et Y (longitude et latitude) d'un colonne et d'une ligne. Les valeurs X et Y retournées sont dans l'unité du système de référence spatial du raster géo-référencé. Les numéros de colonne et de ligne commencent à 1, mais si l'un des deux paramètres est 0, un nombre négatif ou supérieur à la dimension du raster, la fonction retourne des coordonnées en dehors du raster, en supposant que la grille du raster est applicable en dehors de son étendue.

Disponibilité : 2.1.0

Exemples

```
-- non-skewed raster
SELECT
    rid,
    (ST_RasterToWorldCoord(rast,1, 1)).*,
    (ST_RasterToWorldCoord(rast,2, 2)).*
FROM dummy_rast

 rid | longitude  | latitude | longitude |  latitude
-----+------------+----------+-----------+------------
   1 |        0.5 |      0.5 |       2.5 |        3.5
   2 | 3427927.75 |  5793244 | 3427927.8 | 5793243.95
```

```
-- skewed raster
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

 rid | longitude  | latitude | longitude | latitude
-----+------------+----------+-----------+-----------
   1 |        0.5 |      0.5 |     203.5 |       6.5
   2 | 3427927.75 |  5793244 | 3428128.8 | 5793243.9
```

Voir aussi

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SetSkew

11.4.12 ST_RasterToWorldCoordX

ST_RasterToWorldCoordX — Retourne la coordonnée X du coin supérieur gauche du raster à column et row. Les numéros de colonne et de ligne commencent à 1.

Synopsis

float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn);
float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn, integer yrow);

Description

Retourne la coordonnée X du coin supérieur gauche du raster à column et row dans l'unité du système de référence spatial du raster géo-référencé. Les numéros de colonne et de ligne commencent à 1, mais si l'un des deux paramètres est 0, un nombre négatif ou supérieur à la dimension du raster, la fonction retourne des coordonnées en dehors du raster, en supposant que la grille du raster est applicable en dehors de son étendue (même taille de pixel et skew).

Note
Pour des rasters sans obliquité (skew), fournir le numéro de colonne X est suffisant. Pour des rasters avec obliquité, les coordonnées géo-référencées sont fonction de ST_ScaleX et ST_SkewX et de la ligne et colonne. Une erreur sera émise si seulement la colonne X est fournie pour une raster avec obliquité.

Changement : 2.1.0 Jusqu'à la version 2.0.x, cette fonction était appelée ST_Raster2WorldCoordX

Exemples

```
-- non-skewed raster providing column is sufficient
SELECT rid, ST_RasterToWorldCoordX(rast,1) As x1coord,
    ST_RasterToWorldCoordX(rast,2) As x2coord,
    ST_ScaleX(rast) As pixelx
FROM dummy_rast;

 rid |  x1coord   |  x2coord  | pixelx
-----+------------+-----------+--------
   1 |        0.5 |       2.5 |      2
   2 | 3427927.75 | 3427927.8 |   0.05
```

```
-- for fun lets skew it
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;

 rid |  x1coord   |  x2coord  | pixelx
-----+------------+-----------+--------
   1 |        0.5 |     203.5 |      2
   2 | 3427927.75 | 3428128.8 |   0.05
```

Voir aussi

ST_ScaleX, ST_RasterToWorldCoordY, ST_SetSkew, ST_SkewX

11.4.13 ST_RasterToWorldCoordY

ST_RasterToWorldCoordY — Retourne la coordonnée Y du coin supérieur gauche du raster à column et row. Les numéros de colonne et de ligne commencent à 1.

Synopsis

float8 **ST_RasterToWorldCoordY**(raster rast, integer yrow);
float8 **ST_RasterToWorldCoordY**(raster rast, integer xcolumn, integer yrow);

Description

Retourne la coordonnée Y du coin supérieur gauche du raster à column et row dans l'unité du système de référence spatial du raster géo-référencé. Les numéros de colonne et de ligne commencent à 1, mais si l'un des deux paramètres est 0, un nombre négatif ou supérieur à la dimension du raster, la fonction retourne des coordonnées en dehors du raster, en supposant que la grille du raster est applicable en dehors de son étendue (même taille de pixel et skew).

Note
Pour des rasters sans obliquité (skew), fournir le numéro de ligne Y est suffisant. Pour des rasters avec obliquité, les coordonnées géo-référencées sont fonction de ST_ScaleY et ST_SkewY et de la ligne et colonne. Une erreur sera émise si seulement la ligne Y est fournie pour une raster avec obliquité.

Changement : 2.1.0 Jusqu'à la version 2.0.x, cette fonction était appelée ST_Raster2WorldCoordY

Exemples

```
-- non-skewed raster providing row is sufficient
SELECT rid, ST_RasterToWorldCoordY(rast,1) As y1coord,
    ST_RasterToWorldCoordY(rast,3) As y2coord,
    ST_ScaleY(rast) As pixely
FROM dummy_rast;

 rid | y1coord |  y2coord  | pixely
-----+---------+-----------+--------
   1 |     0.5 |       6.5 |      3
   2 | 5793244 | 5793243.9 |  -0.05
```

```
-- for fun lets skew it
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;

 rid | y1coord |  y2coord  | pixely
-----+---------+-----------+--------
   1 |     0.5 |       107 |      3
   2 | 5793244 | 5793344.4 |  -0.05
```

Voir aussi

ST_ScaleY, ST_RasterToWorldCoordX, ST_SetSkew, ST_SkewY

11.4.14 ST_Rotation

ST_Rotation — Retourne l'angle de rotation du raster en radians.

Synopsis

float8 **ST_Rotation**(raster rast);

Description

Retourne la rotation uniforme du raster en radians. Si le raster n'a pas de rotation uniforme, retourne NaN. Voir World File pour plus de détails.

Exemples

```
SELECT rid, ST_Rotation(ST_SetScale(ST_SetSkew(rast, sqrt(2)), sqrt(2))) as rot FROM  ↩
    dummy_rast;

 rid |        rot
-----+-------------------
   1 | 0.785398163397448
   2 | 0.785398163397448
```

Voir aussi

ST_SetRotation, ST_SetScale, ST_SetSkew

11.4.15 ST_SkewX

ST_SkewX — Retourne l'obliquité géo-référencée X (paramètre de rotation).

Synopsis

float8 **ST_SkewX**(raster rast);

Description

Retourne l'obliquité géo-référencée X (paramètre de rotation). Voir World File pour plus de détails.

Exemples

```
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
    ST_GeoReference(rast) as georef
FROM dummy_rast;

 rid | skewx | skewy |       georef
-----+-------+-------+--------------------
   1 |     0 |     0 | 2.0000000000
                     : 0.0000000000
                     : 0.0000000000
                     : 3.0000000000
                     : 0.5000000000
                     : 0.5000000000
                     :
   2 |     0 |     0 | 0.0500000000
                     : 0.0000000000
                     : 0.0000000000
                     : -0.0500000000
                     : 3427927.7500000000
                     : 5793244.0000000000
```

Voir aussi

ST_GeoReference, ST_SkewY, ST_SetSkew

11.4.16 ST_SkewY

ST_SkewY — Retourne l'obliquité géo-référencée Y (paramètre de rotation).

Synopsis

float8 **ST_SkewY**(raster rast);

Description

Retourne l'obliquité géo-référencée Y (paramètre de rotation). Voir World File pour plus de détails.

Exemples

```
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
    ST_GeoReference(rast) as georef
FROM dummy_rast;

 rid | skewx | skewy |       georef
-----+-------+-------+--------------------
   1 |     0 |     0 | 2.0000000000
                     : 0.0000000000
                     : 0.0000000000
                     : 3.0000000000
                     : 0.5000000000
                     : 0.5000000000
                     :
   2 |     0 |     0 | 0.0500000000
                     : 0.0000000000
                     : 0.0000000000
                     : -0.0500000000
                     : 3427927.7500000000
                     : 5793244.0000000000
```

Voir aussi

ST_GeoReference, ST_SkewX, ST_SetSkew

11.4.17 ST_SRID

ST_SRID — Retourne l'identifiant du système de référence spatial du raster, tel que défini dans la table spatial_ref_sys.

Synopsis

integer **ST_SRID**(raster rast);

Description

Retourne l'identifiant du système de référence spatial du raster, tel que défini dans la table spatial_ref_sys.

Note
À partir de PostGIS 2.0+, le SRID d'un raster ou d'une géométrie non-géo-référencée est 0 au lieu de -1 précédemment.

Exemples

```
SELECT ST_SRID(rast) As srid
FROM dummy_rast WHERE rid=1;

srid
----------------
0
```

Voir aussi

Section 4.5, ST_SRID

11.4.18 ST_Summary

ST_Summary — Retourne un résumé du contenu du raster sous forme de texte.

Synopsis

text **ST_Summary**(raster rast);

Description

Retourne un résumé du contenu du raster sous forme de texte.

Disponibilité : 2.1.0

Exemples

```
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
    )
);

                            st_summary
------------------------------------------------------------------
 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)
```

Voir aussi

ST_MetaData, ST_BandMetaData, ST_Summary ST_Extent

11.4.19 ST_UpperLeftX

ST_UpperLeftX — Retourne la coordonnée X du coin supérieur gauche du raster projeté dans le système de référence spatial.

Synopsis

float8 **ST_UpperLeftX**(raster rast);

Description

Retourne la coordonnée X du coin supérieur gauche du raster projeté dans le système de référence spatial.

Exemples

```
SELECt rid, ST_UpperLeftX(rast) As ulx
FROM dummy_rast;

 rid |    ulx
-----+------------
   1 |        0.5
   2 | 3427927.75
```

Voir aussi

ST_UpperLeftY, ST_GeoReference, Box3D

11.4.20 ST_UpperLeftY

ST_UpperLeftY — Retourne la coordonnée Y du coin supérieur gauche du raster projeté dans le système de référence spatial.

Synopsis

float8 **ST_UpperLeftY**(raster rast);

Description

Retourne la coordonnée Y du coin supérieur gauche du raster projeté dans le système de référence spatial.

Exemples

```
SELECT rid, ST_UpperLeftY(rast) As uly
FROM dummy_rast;

 rid |   uly
-----+---------
   1 |     0.5
   2 | 5793244
```

Voir aussi

ST_UpperLeftX, ST_GeoReference, Box3D

11.4.21 ST_Width

ST_Width — Retourne la largeur du raster en pixels.

Synopsis

integer **ST_Width**(raster rast);

Description

Retourne la largeur du raster en pixels.

Exemples

```
SELECT ST_Width(rast) As rastwidth
FROM dummy_rast WHERE rid=1;

rastwidth
----------------
10
```

Voir aussi

ST_Height

11.4.22 ST_WorldToRasterCoord

ST_WorldToRasterCoord — Retourne le coin supérieur gauche comme colonne et ligne, en fonction de coordonnées géométriques X et Y (longitude et latitude) ou un point géométrique dans le système de référence spatial du raster.

Synopsis

record **ST_WorldToRasterCoord**(raster rast, geometry pt);
record **ST_WorldToRasterCoord**(raster rast, double precision longitude, double precision latitude);

Description

Retourne le coin supérieur gauche comme colonne et ligne, en fonction de coordonnées géométriques X et Y (longitude et latitude) ou un point géométrique. Cette fonction fonctionne aussi bien si X et Y ou le point sont dans l'étendue du raster ou non. Les coordonnées géométriques X et Y ou les coordonnées du point doivent être dans le système de référence spatial du raster.

Disponibilité : 2.1.0

Exemples

```
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;

 rid | columnx |   rowy    | columnx |   rowy
-----+---------+-----------+---------+-----------
   1 | 1713964 |         7 | 1713964 |         7
   2 |       2 | 115864471 |       2 | 115864471
```

Voir aussi

ST_WorldToRasterCoordX, ST_WorldToRasterCoordY, ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

11.4.23 ST_WorldToRasterCoordX

ST_WorldToRasterCoordX — Retourne la colonne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw) exprimés dans le système de référence spatial du raster.

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

Retourne la colonne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw). Un point, ou le couple (xw, yw) sont requis pour un raster avec obliquité (skew). Sinon, xw est suffisant. Les coordonnées sont exprimées dans le système de référence spatial du raster.

Changement : 2.1.0 Dans les versions précédentes, cette fonctionne était appelée ST_World2RasterCoordX

Exemples

```
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;

 rid | xcoord  |  xcoord_xwyw   | ptxcoord
-----+---------+---------+----------
   1 | 1713964 | 1713964 |  1713964
   2 |       1 |       1 |        1
```

Voir aussi

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

11.4.24 ST_WorldToRasterCoordY

ST_WorldToRasterCoordY — Retourne la ligne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw) exprimés dans le système de référence spatial du raster.

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

Retourne la ligne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw). Un point, ou le couple (xw, yw) sont requis pour un raster avec obliquité (skew). Sinon, xw est suffisant. Les coordonnées sont exprimées dans le système de référence spatial du raster.

Changement : 2.1.0 Dans les versions précédentes, cette fonctionne était appelée ST_World2RasterCoordY

Exemples

```
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;

 rid |  ycoord   | ycoord_xwyw | ptycoord
-----+-----------+-------------+-----------
   1 |         7 |           7 |         7
   2 | 115864471 |   115864471 | 115864471
```

Voir aussi

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

11.5 Fonctions d'accès aux bandes raster

11.5.1 ST_BandMetaData

ST_BandMetaData — Retourne les méta-données de base d'une bande raster spécifique. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

(1) record **ST_BandMetaData**(raster rast, integer band=1);
(2) record **ST_BandMetaData**(raster rast, integer[] band);

Description

Retourne les méta-données de base d'une bande raster spécifique. Les colonnes retournées sont : pixeltype, nodatavalue, isoutdb, path, outdbbandnum, filesize, filetimestamp.

Note
Si le raster ne contient aucune bande, une erreur est émise.

Note
Si la bande n'a pas de valeur NODATA, nodatavalue est NULL.

Note
Si isoutdb est False, alors path, outdbbandnum, filesize et filetimestamp sont NULL. Si l'accès outdb est désactivé, filesize et filetimestamp sont également NULL.

Amélioration : 2.5.0 ajout de *outdbbandnum*, *filesize* et *filetimestamp* pour les rasters outdb.

Exemples : Variante 1

```
SELECT
    rid,
    (foo.md).*
FROM (
    SELECT
        rid,
        ST_BandMetaData(rast, 1) AS md
    FROM dummy_rast
    WHERE rid=2
) As foo;

 rid | pixeltype | nodatavalue | isoutdb | path | outdbbandnum
-----+-----------+---- --------+---------+------+--------------
   2 | 8BUI      |           0 | f       |      |
```

Exemples : Variante 2

```
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
    *
FROM ST_BandMetadata(
    (SELECT rast FROM foo),
    ARRAY[1,3,2]::int[]
);

 bandnum | pixeltype | nodatavalue | isoutdb |                                        path  ↩
                                          | outdbbandnum  | filesize | filetimestamp |
---------+-----------+-------------+---------+-----------------------------------------------------------

       1 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/raster/test ↩
           /regress/loader/Projected.tif      |            1  |    12345 |    1521807257 |
       3 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/raster/test ↩
           /regress/loader/Projected.tif      |            3  |    12345 |    1521807257 |
       2 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/raster/test ↩
           /regress/loader/Projected.tif      |            2  |    12345 |    1521807257 |
```

Voir aussi

ST_MetaData, ST_BandPixelType

11.5.2 ST_BandNoDataValue

ST_BandNoDataValue — Retourne la valeur qui représente l'absence de valeur (nodata) pour cette bande. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

double precision **ST_BandNoDataValue**(raster rast, integer bandnum=1);

Description

Retourne la valeur qui représente l'absence de valeur (nodata) pour cette bande

Exemples

```
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;

 bnval1 | bnval2 | bnval3
--------+--------+--------
      0 |      0 |      0
```

Voir aussi

ST_NumBands

11.5.3 ST_BandIsNoData

ST_BandIsNoData — Retourne true si la bande ne contient que des valeurs nodata.

Synopsis

boolean **ST_BandIsNoData**(raster rast, integer band, boolean forceChecking=true);
boolean **ST_BandIsNoData**(raster rast, boolean forceChecking=true);

Description

Retourne true si la bande ne contient que des valeurs nodata. Si aucune bande n'est spécifiée, la bande 1 est utilisée. Si le dernier paramètre est TRUE, la bande entière est vérifiée pixel par pixel. Sinon, la fonction retourne simplement la valeur du flag isnodata pour cette bande. La valeur par défaut pour ce paramètre est FALSE.

Disponibilité : 2.0.0

Note
Si le flag est obsolète (c'est à dire si le résultat est différent selon qu'on utilise TRUE ou non comme dernier paramètre), vous devriez mettre à jour le raster pour mettre à jour ce flag, en utilisant ST_SetBandIsNodata(), ou ST_SetBandNodataValue() avec TRUE en dernier paramètre. Voir ST_SetBandIsNoData.

Exemples

```
-- Create dummy table with one raster column
create table dummy_rast (rid integer, rast raster);

-- Add raster with two bands, one pixel/band. In the first band, nodatavalue = pixel value ↩
    = 3.
-- In the second band, nodatavalue = 13, pixel value = 4
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
```

Voir aussi

ST_BandNoDataValue, ST_NumBands, ST_SetBandNoDataValue, ST_SetBandIsNoData

11.5.4 ST_BandPath

ST_BandPath — Retourne le chemin système du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

text **ST_BandPath**(raster rast, integer bandnum=1);

Description

Retourne le chemin système du fichier d'une bande. Renvoie une erreur si la bande n'est pas outofdb.

Exemples

Voir aussi

11.5.5 ST_BandFileSize

ST_BandFileSize — Retourne la taille du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

bigint **ST_BandFileSize**(raster rast, integer bandnum=1);

Description

Retourne la taille du fichier d'une bande stockée sur le système de fichier. Renvoie une erreur si la bande n'est pas outofdb, ou si l'accès outdb est désactivé.

Cette fonction est généralement utilisée conjointement à ST_BandPath() et ST_BandFileTimestamp(), pour qu'un client puisse déterminer le nom du fichier d'un raster outdb, tel que vu par le serveur.

Disponibilité : 2.5.0

Exemples

```
SELECT ST_BandFileSize(rast,1) FROM dummy_rast WHERE rid = 1;

 st_bandfilesize
-----------------
          240574
```

11.5.6 ST_BandFileTimestamp

ST_BandFileTimestamp — Retourne le timestamp du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

bigint **ST_BandFileTimestamp**(raster rast, integer bandnum=1);

Description

Retourne le timestamp (nombre de secondes depuis Jan 1st 1970 00:00:00 UTC) du fichier d'une bande stockée sur le système de fichier. Renvoie une erreur si la bande n'est pas outofdb, ou si l'accès outdb est désactivé.

Cette fonction est généralement utilisée conjointement à ST_BandPath() et ST_BandFileSize(), pour qu'un client puisse déterminer le nom du fichier d'un raster outdb, tel que vu par le serveur.

Disponibilité : 2.5.0

Exemples

```
SELECT ST_BandFileTimestamp(rast,1) FROM dummy_rast WHERE rid = 1;

 st_bandfiletimestamp
----------------------
           1521807257
```

11.5.7 ST_BandPixelType

ST_BandPixelType — Retourne le type de pixel d'une bande. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

text **ST_BandPixelType**(raster rast, integer bandnum=1);

Description

Retourne le nom décrivant le type de données et la taille des valeurs stockées pour chaque pixel de la bande spécifiée.

Il existe 11 types de pixels. Les types de pixels supportés sont :

- 1BB - booléen 1-bit
- 2BUI - entier non signé 2-bits
- 4BUI - entier non signé 4-bits
- 8BSI - entier signé 8-bits
- 8BUI - entier non signé 8-bits
- 16BSI - entier signé 16-bits
- 16BUI - entier non signé 16-bits
- 32BSI - entier signé 32-bits
- 32BUI - entier non signé 32-bits
- 32BF - flottant 32-bits
- 64BF - flottant 64-bits

Exemples

```
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;

 btype1 | btype2 | btype3
--------+--------+--------
 8BUI   | 8BUI   | 8BUI
```

Voir aussi

ST_NumBands

11.5.8 ST_MinPossibleValue

ST_MinPossibleValue — Retourne la valeur minimale supportée par le type de pixel pixeltype.

Synopsis

integer **ST_MinPossibleValue**(text pixeltype);

Description

Retourne la valeur minimale supportée par le type de pixel pixeltype.

Exemples

```
SELECT ST_MinPossibleValue('16BSI');

 st_minpossiblevalue
---------------------
              -32768


SELECT ST_MinPossibleValue('8BUI');

 st_minpossiblevalue
---------------------
                   0
```

Voir aussi

ST_BandPixelType

11.5.9 ST_HasNoBand

ST_HasNoBand — Retourne true si le raster n'a pas la bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

boolean **ST_HasNoBand**(raster rast, integer bandnum=1);

Description

Retourne true si le raster n'a pas la bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
Disponibilité : 2.0.0

Exemples

```
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;

rid | hb1 | hb2 | hb4 | numbands
-----+-----+-----+-----+----------
1 | t   | t   | t   |        0
2 | f   | f   | t   |        3
```

Voir aussi

ST_NumBands

11.6 Fonctions d'accès et de modifications des pixels raster

11.6.1 ST_PixelAsPolygon

ST_PixelAsPolygon — Retourne la géométrie polygonale qui délimite le pixel pour une ligne et colonne spécifiées.

Synopsis

geometry **ST_PixelAsPolygon**(raster rast, integer columnx, integer rowy);

Description

Retourne la géométrie polygonale qui délimite le pixel pour une ligne et colonne spécifiées.

Disponibilité : 2.0.0

Exemples

```
-- get raster pixel polygon
SELECT i,j, ST_AsText(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;

 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, ..
```

Voir aussi

ST_DumpAsPolygons, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids, ST_Intersection, ST_AsText

11.6.2 ST_PixelAsPolygons

ST_PixelAsPolygons — Retourne la géométrie polygonale qui délimite chaque pixel de la bande raster, avec la valeur et les coordonnées raster X et Y de chaque pixel.

Synopsis

setof record **ST_PixelAsPolygons**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

Description

Retourne la géométrie polygonale qui délimite chaque pixel de la bande raster, avec la valeur (double précision) et les coordonnées raster X et Y (entiers) de chaque pixel.

Format de l'enregistrement de retour : *geom* geometry, *val* double precision, *x* entier, *y* entier.

Note
Si *exclude_nodata_value* = TRUE, seuls les pixels qui ne sont pas NODATA sont retournés.

Note
ST_PixelAsPolygons retourne un polygone pour chaque pixel. Ce comportement est différent de ST_DumpAsPolygons, où chaque géométrie représente un ou plusieurs pixels de même valeur.

Disponibilité : 2.0.0

Amélioration : 2.1.0 ajout du paramètre optionnel exclude_nodata_value.

Changement : 2.1.1 Changement du comportement de behavior of exclude_nodata_value.

Exemples

```
-- get raster pixel polygon
SELECT (gv).x, (gv).y, (gv).val, ST_AsText((gv).geom) geom
FROM (SELECT ST_PixelAsPolygons(
                 ST_SetValue(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 0.001,  ↩
                     -0.001, 0.001, 0.001, 4269),
                                                 '8BUI'::text, 1, 0),
                                      2, 2, 10),
                            1, 1, NULL)
) gv
) foo;

 x | y | val |                geom
---+---+-----+-------------------------------------------------------------------------
 1 | 1 |     | POLYGON((0 0,0.001 0.001,0.002 0,0.001 -0.001,0 0))
 1 | 2 |   1 | POLYGON((0.001 -0.001,0.002 0,0.003 -0.001,0.002 -0.002,0.001 -0.001))
 2 | 1 |   1 | POLYGON((0.001 0.001,0.002 0.002,0.003 0.001,0.002 0,0.001 0.001))
 2 | 2 |  10 | POLYGON((0.002 0,0.003 0.001,0.004 0,0.003 -0.001,0.002 0))
```

Voir aussi

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids, ST_AsText

11.6.3 ST_PixelAsPoint

ST_PixelAsPoint — Retourne le point géométrique du coin supérieur gauche du pixel.

Synopsis

geometry **ST_PixelAsPoint**(raster rast, integer columnx, integer rowy);

Description

Retourne le point géométrique du coin supérieur gauche du pixel.

Disponibilité : 2.1.0

Exemples

```
SELECT ST_AsText(ST_PixelAsPoint(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;

   st_astext
----------------
 POINT(0.5 0.5)
```

Voir aussi

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids

11.6.4 ST_PixelAsPoints

ST_PixelAsPoints — Retourne un point géométrique pour chaque pixel de la bande raster, avec sa valeur et ses coordonnées raster X et Y. Les coordonnées du points sont ceux du coin supérieur gauche du pixel.

Synopsis

setof record **ST_PixelAsPoints**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

Description

Retourne un point géométrique pour chaque pixel de la bande raster, avec sa valeur et ses coordonnées raster X et Y. Les coordonnées du points sont ceux du coin supérieur gauche du pixel.

Format de l'enregistrement de retour : *geom* geometry, *val* double precision, *x* entier, *y* entier.

Note

Si *exclude_nodata_value* = TRUE, seuls les pixels qui ne sont pas NODATA sont retournés.

Disponibilité : 2.1.0

Changement : 2.1.1 Changement du comportement de behavior of exclude_nodata_value.

Exemples

```
SELECT x, y, val, ST_AsText(geom) FROM (SELECT (ST_PixelAsPoints(rast, 1)).* FROM  ↩
    dummy_rast WHERE rid = 2) foo;

 x | y | val |          st_astext
---+---+-----+-----------------------------
 1 | 1 | 253 | POINT(3427927.75 5793244)
 2 | 1 | 254 | POINT(3427927.8 5793244)
 3 | 1 | 253 | POINT(3427927.85 5793244)
 4 | 1 | 254 | POINT(3427927.9 5793244)
```

```
 5 | 1 | 254 | POINT(3427927.95 5793244)
 1 | 2 | 253 | POINT(3427927.75 5793243.95)
 2 | 2 | 254 | POINT(3427927.8 5793243.95)
 3 | 2 | 254 | POINT(3427927.85 5793243.95)
 4 | 2 | 253 | POINT(3427927.9 5793243.95)
 5 | 2 | 249 | POINT(3427927.95 5793243.95)
 1 | 3 | 250 | POINT(3427927.75 5793243.9)
 2 | 3 | 254 | POINT(3427927.8 5793243.9)
 3 | 3 | 254 | POINT(3427927.85 5793243.9)
 4 | 3 | 252 | POINT(3427927.9 5793243.9)
 5 | 3 | 249 | POINT(3427927.95 5793243.9)
 1 | 4 | 251 | POINT(3427927.75 5793243.85)
 2 | 4 | 253 | POINT(3427927.8 5793243.85)
 3 | 4 | 254 | POINT(3427927.85 5793243.85)
 4 | 4 | 254 | POINT(3427927.9 5793243.85)
 5 | 4 | 253 | POINT(3427927.95 5793243.85)
 1 | 5 | 252 | POINT(3427927.75 5793243.8)
 2 | 5 | 250 | POINT(3427927.8 5793243.8)
 3 | 5 | 254 | POINT(3427927.85 5793243.8)
 4 | 5 | 254 | POINT(3427927.9 5793243.8)
 5 | 5 | 254 | POINT(3427927.95 5793243.8)
```

Voir aussi

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsCentroid, ST_PixelAsCentroids

11.6.5 ST_PixelAsCentroid

ST_PixelAsCentroid — Retourne le centroïde (point géométrique) de la zone représentée par un pixel.

Synopsis

geometry **ST_PixelAsCentroid**(raster rast, integer x, integer y);

Description

Retourne le centroïde (point géométrique) de la zone représentée par un pixel.

Amélioration : 3.2.0 Plus rapide, désormais implémenté en C.

Disponibilité : 2.1.0

Exemples

```
SELECT ST_AsText(ST_PixelAsCentroid(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;

  st_astext
--------------
 POINT(1.5 2)
```

Voir aussi

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroids

11.6.6 ST_PixelAsCentroids

ST_PixelAsCentroids — Retourne le centroïde (point géométrique) pour chaque pixel de la bande raster, avec sa valeur et les coordonnées raster X et Y. Le point géométrique est le centroïde de la zone représentée par un pixel.

Synopsis

setof record **ST_PixelAsCentroids**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

Description

Retourne le centroïde (point géométrique) pour chaque pixel de la bande raster, avec sa valeur et les coordonnées raster X et Y. Le point géométrique est le centroïde de la zone représentée par un pixel.

Format de l'enregistrement de retour : *geom* geometry, *val* double precision, *x* entier, *y* entier.

Note

Si *exclude_nodata_value* = TRUE, seuls les pixels qui ne sont pas NODATA sont retournés.

Amélioration : 3.2.0 Plus rapide, désormais implémenté en C.

Changement : 2.1.1 Changement du comportement de behavior of exclude_nodata_value.

Disponibilité : 2.1.0

Exemples

```
--LATERAL syntax requires PostgreSQL 9.3+
SELECT x, y, val, ST_AsText(geom)
    FROM (SELECT dp.* FROM dummy_rast, LATERAL ST_PixelAsCentroids(rast, 1) AS dp WHERE rid ↩
        = 2) foo;
 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)
```

Voir aussi

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid

11.6.7 ST_Value

ST_Value — Retourne la valeur d'une bande raster spécifiée au pixel donné par columnx, rowy, ou à un point géométrique spécifié. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié. Si `exclude_nodata_value` vaut false, tous les pixels y compris ceux ayant la valeur `nodata` sont considérés comme intersectés et leur valeur sera retournée. Si `exclude_nodata_value` n'est pas spécifié, la valeur est lue depuis les méta-données du raster.

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=true);

Description

Retourne la valeur d'une bande raster spécifiée au pixel donné par columnx, rowy, ou à un point géométrique spécifié. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié.

Si `exclude_nodata_value` vaut true, seulement les pixels différents de la valeur `nodata` seront utilisés. Si `exclude_nodata_v` vaut false, tous les pixels seront utilisés.

Les valeurs possibles pour le paramètre `resample` sont : "nearest" (interpolation par plus proche voisin) et "bilinear" (interpolation bilinéaire). Ce paramètre est utilisé pour estimer la valeur entre les centres des pixels.

Amélioration : 3.2.0 ajout du paramètre optionnel resample.

Amélioration : 2.0.0 ajout du paramètre optionnel exclude_nodata_value.

Exemples

```
-- get raster values at particular postgis geometry points
-- the srid of your geometry should be same as for your raster
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;

 rid | b1pval | b2pval
-----+--------+--------
   2 |    252 |     79


-- general fictitious example using a real table
SELECT rid, ST_Value(rast, 3, sometable.geom) As b3pval
FROM sometable
WHERE ST_Intersects(rast,sometable.geom);
```

```
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;

 rid | b1pval | b2pval | b3pval
-----+--------+--------+--------
   2 |    253 |     78 |     70
```

```
--- Get all values in bands 1,2,3 of each pixel --
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);

 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 --
SELECT ST_AsText(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);

            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 particular polygon --
SELECT ST_AsText(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;


        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

SELECT ST_AsText(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;

        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)))
```

Voir aussi

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_Intersects, ST_Intersection

11.6.8 ST_NearestValue

ST_NearestValue — Retourne la valeur la plus proche différent de `NODATA` pour une bande raster spécifiée au pixel donné par columnx et rowy, ou à un point géométrique spécifié dans le même système de référence spatial que le raster.

Synopsis

double precision **ST_NearestValue**(raster rast, integer bandnum, geometry pt, boolean exclude_nodata_value=true);
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_value=true);

Description

Retourne la valeur la plus proche différent de `NODATA` pour une bande raster spécifiée au pixel donné par columnx et rowy, ou à un point géométrique spécifié. Si le pixel donné par columnx, rowy ou par le point géométrique est de valeur `NODATA`, la fonction trouve le plus proche pixel dont la valeur est différente de `NODATA`.

Le numéro de bande démarre à 1, et la bande 1 est utilisée si `bandnum` non spécifié. Si `exclude_nodata_value` vaut false, tous les pixels y compris ceux ayant la valeur `nodata` sont considérés comme intersectés et leur valeur sera retournée. Si `exclude_nodata_value` n'est pas spécifié, la valeur est lue depuis les méta-données du raster.

Disponibilité : 2.1.0

Note
ST_NearestValue remplace directement ST_Value.

Exemples

```
-- pixel 2x2 has value
SELECT
    ST_Value(rast, 2, 2) AS value,
    ST_NearestValue(rast, 2, 2) AS nearestvalue
FROM (
    SELECT
        ST_SetValue(
            ST_SetValue(
                ST_SetValue(
```

```
                    ST_SetValue(
                        ST_SetValue(
                            ST_AddBand(
                                ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                                '8BUI'::text, 1, 0
                            ),
                            1, 1, 0.
                        ),
                        2, 3, 0.
                    ),
                    3, 5, 0.
                ),
                4, 2, 0.
            ),
            5, 4, 0.
        ) AS rast
) AS foo

 value | nearestvalue
-------+--------------
     1 |            1
```

```
-- pixel 2x3 is NODATA
SELECT
    ST_Value(rast, 2, 3) AS value,
    ST_NearestValue(rast, 2, 3) AS nearestvalue
FROM (
    SELECT
        ST_SetValue(
            ST_SetValue(
                ST_SetValue(
                    ST_SetValue(
                        ST_SetValue(
                            ST_AddBand(
                                ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                                '8BUI'::text, 1, 0
                            ),
                            1, 1, 0.
                        ),
                        2, 3, 0.
                    ),
                    3, 5, 0.
                ),
                4, 2, 0.
            ),
            5, 4, 0.
        ) AS rast
) AS foo

 value | nearestvalue
-------+--------------
       |            1
```

Voir aussi

ST_Neighborhood, ST_Value

11.6.9 ST_SetZ

ST_SetZ — Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée Z copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.

Synopsis

geometry **ST_SetZ**(raster rast, geometry geom, text resample=nearest, integer band=1);

Description

Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée Z copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.

Le paramètre `resample` peut être "nearest" (plus proche) pour copier les valeurs de chaque pixel auquel le vertex correspond, ou "bilinear" pour utiliser une interpolation bilinéaire pour calculer une valeur qui prend aussi en compte les pixels voisins.

Disponibilité : 3.2.0

Exemples

```
--
-- 2x2 test raster with values
--
-- 10 50
-- 40 20
--
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_AsText(
  ST_SetZ(
    rast,
    band => 1,
    geom => 'SRID=4326;LINESTRING(1.0 1.9, 1.0 0.2)'::geometry,
    resample => 'bilinear'
))
FROM test_raster

            st_astext
---------------------------------
 LINESTRING Z (1 1.9 38,1 0.2 27)
```

Voir aussi

ST_Value, ST_SetM

11.6.10 ST_SetM

ST_SetM — Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée M copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.

Synopsis

geometry **ST_SetM**(raster rast, geometry geom, text resample=nearest, integer band=1);

Description

Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée M copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.

Le paramètre `resample` peut être "nearest" (plus proche) pour copier les valeurs de chaque pixel auquel le vertex correspond, ou "bilinear" pour utiliser une interpolation bilinéaire pour calculer une valeur qui prend aussi en compte les pixels voisins.

Disponibilité : 3.2.0

Exemples

```
--
-- 2x2 test raster with values
--
-- 10 50
-- 40 20
--
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_AsText(
  ST_SetM(
    rast,
    band => 1,
    geom => 'SRID=4326;LINESTRING(1.0 1.9, 1.0 0.2)'::geometry,
    resample => 'bilinear'
))
FROM test_raster

            st_astext
---------------------------------
 LINESTRING M (1 1.9 38,1 0.2 27)
```

Voir aussi

ST_Value, ST_SetZ

11.6.11 ST_Neighborhood

ST_Neighborhood — Retourne un tableau 2-D de double avec les valeurs non NODATA autour du pixel de la bande spécifiée, aux coordonnées spécifiées par columnX & rowY ou par un point géométrique dans le même système de référence spatial que le raster.

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=tru

Description

Retourne un tableau 2-D de double avec les valeurs non NODATA autour du pixel de la bande spécifiée, aux coordonnées spécifiées par columnX & rowY ou par un point géométrique dans le même système de référence spatial que le raster. Les paramètres distanceX et distanceY définissent le nombre de pixels autour du pixel spécifié sur les axes X et Y. Exemple : vous pouvez récupérer toutes les valeurs autour du pixel souhaité, à une distance de 3 pixels selon l'axe X et à une distance de 2 pixels selon l'axe Y. La valeur centrale du tableau 2-D sera la valeur du pixel spécifié par columnX et rowY ou par le point géométrique.

Le numéro de bande démarre à 1, et la bande 1 est utilisée si bandnum non spécifié. Si exclude_nodata_value vaut false, tous les pixels y compris ceux ayant la valeur nodata sont considérés comme intersectés et leur valeur sera retournée. Si exclude_nodata_value n'est pas spécifié, la valeur est lue depuis les méta-données du raster.

Note
Le nombre d'éléments retournés dans le tableau 2-D sur chaque axe sera 2 * (distanceX|distanceY) + 1. Pour distanceX et distanceY de 1, le tableau sera de taille 3x3.

Note
Le tableau 2-D en sortie peut être passé à n'importe quelle fonction de traitement raster, comme ST_Min4ma, ST_Sum4ma, ST_Mean4ma.

Disponibilité : 2.1.0

Exemples

```
-- pixel 2x2 has value
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

         st_neighborhood
---------------------------------
{{NULL,1,1},{1,1,1},{1,NULL,1}}
```

```
-- pixel 2x3 is NODATA
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

       st_neighborhood
------------------------------
 {{1,1,1},{1,NULL,1},{1,1,1}}
```

```
-- pixel 3x3 has value
-- exclude_nodata_value = FALSE
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

      st_neighborhood
---------------------------
{{1,1,0},{0,1,1},{1,1,1}}
```

Voir aussi

ST_NearestValue, ST_Min4ma, ST_Max4ma, ST_Sum4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.6.12 ST_SetValue

ST_SetValue — Retourne un nouveau raster en modifiant la valeur du pixel pour la bande spécifiée et aux coordonnées columnx, rowy, ou pour tous les pixels qui intersectent une géométrie spécifiée. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié.

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

Retourne un nouveau raster en modifiant la valeur du pixel pour la bande spécifiée et aux coordonnées columnx, rowy, ou pour tous les pixels qui intersectent une géométrie spécifiée. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié.

Amélioration : 2.1.0 La variante de ST_SetValue() avec la géométrie supporte n'importe quelle type de géométrie, pas uniquement des points. La variante avec la géométrie est une enveloppe autour de la variante de ST_SetValues() avec geomval[]

Exemples

```
                -- Geometry example
SELECT (foo.geomval).val, ST_AsText(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;

 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 ...
```

```
-- Store the changed raster --
    UPDATE dummy_rast SET rast = ST_SetValue(rast,1, ST_Point(3427927.75, 5793243.95),100)
        WHERE rid = 2   ;
```

Voir aussi

ST_Value, ST_DumpAsPolygons

11.6.13 ST_SetValues

ST_SetValues — Retourne un nouveau raster en modifiant les valeurs de certains pixels d'une bande spécifiée.

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

Retourne un nouveau raster en modifiant les valeurs de certains pixels d'une bande spécifiée. Les paramètres `columnx` et `rowy` démarrent à 1.

Si `keepnodata` est TRUE, les pixels qui sont NODATA ne seront pas mis à jour avec la valeur donnée par `newvalueset`.

Pour la variante 1, les pixels sont déterminés par les coordonnées du pixel `columnx` et `rowy`, ainsi que la dimension du tableau `newvalueset`. `noset` peut être utilisé pour s'assurer qu'aucun pixel présent dans `newvalueset` et ayant déjà une valeur soient modifiés (PostgreSQL ne supportant pas les tableaux irréguliers/agglomérés). Voir l'exemple Variante 1.

La variante 2 est similaire à la variante 1, mais avec une valeur unique en double precision `nosetvalue` au lieu du tableau de booléens `noset`. Les éléments dans `newvalueset` avec la valeur `nosetvalue` seront ignorés. Voir l'exemple Variante 2.

Pour la variante 3, les pixels à modifier doivent être explicitement spécifiés par les coordonnées `columnx`, `rowy`, ainsi que les tailles `width` (largeur) et `height` (hauteur). Voir l'exemple Variante 3.

La variante 4 est identique à la variante 3, en modifiant la bande raster 1 du raster `rast`.

Pour la variante 5, un tableau de geomval est utilisé pour déterminer les pixels à modifier. Si toutes les géométries du tableau sont de type POINT ou MULTIPOINT, la fonction utilise un raccourci où la longitude et la latitude du chaque point sont utilisés pour modifier directement le pixel. Sinon, les géométries sont converties en raster, puis itérés en une passe. Voir l'exemple Variante 5.

Disponibilité : 2.1.0

Exemples : Variante 1

```
/*
The ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 1 | 1 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |    =
>    | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
*/
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[][]
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 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 ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 9 | 9 | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |    =
>    | 9 |   | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 9 | 9 | 9 |
+ - + - + - +          + - + - + - +
*/
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[][]
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 x | y | val
---+---+-----
 1 | 1 |   9
 1 | 2 |   9
```

```
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

```
/*
The ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 9 | 9 | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |    =>    | 1 |   | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 9 | 9 | 9 |
+ - + - + - +          + - + - + - +
*/
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[][]
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 x | y | val
---+---+-----
 1 | 1 |   9
 1 | 2 |   1
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

```
/*
The ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
|   | 1 | 1 |          |   | 9 | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |    =>    | 1 |   | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 9 | 9 | 9 |
```

```
+ - + - + - +          + - + - + - +
*/
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
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 x | y | val
---+---+-----
 1 | 1 |
 1 | 2 |   1
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |
 2 | 3 |   9
 3 | 1 |   9
 3 | 3 |   9
 3 | 3 |   9
```

Exemples : Variante 2

```
/*
The ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 1 | 1 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |    =>    | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
*/
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
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 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
```

```
/*
This example is like the previous one.  Instead of nosetvalue = -1, nosetvalue = NULL

The ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 1 | 1 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |    =>    | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
*/
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
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 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
```

Exemples : Variante 3

```
/*
The ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 1 | 1 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |    =>    | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
*/
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
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 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 ST_SetValues() does the following...

+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 1 | 1 |
+ - + - + - +          + - + - + - +
| 1 |   | 1 |    =>    | 1 |   | 9 |
+ - + - + - +          + - + - + - +
| 1 | 1 | 1 |          | 1 | 9 | 9 |
+ - + - + - +          + - + - + - +
*/
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
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

Exemples : Variante 5

```
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
)
SELECT
    rid, gid, ST_DumpValues(ST_SetValue(rast, 1, geom, gid))
FROM foo t1
CROSS JOIN bar t2
ORDER BY rid, gid;

 rid | gid |                                                                st_dumpvalues
-----+-----+-----------------------------------------------------------------------------------------------

   1 |   1 | (1,"{{NULL,NULL,NULL,NULL,NULL},{NULL,NULL,NULL,NULL,NULL},{NULL,NULL,1,NULL, ↩
       NULL},{NULL,NULL,NULL,NULL,NULL},{NULL,NULL,NULL,NULL,NULL}}")
   1 |   2 | (1,"{{NULL,NULL,NULL,NULL,NULL},{NULL,2,2,2,NULL},{NULL,2,2,2,NULL},{NULL ↩
       ,2,2,2,NULL},{NULL,NULL,NULL,NULL,NULL}}")
   1 |   3 | (1,"{{3,3,3,3,3},{3,NULL,NULL,NULL,NULL},{3,NULL,NULL,NULL,NULL},{3,NULL,NULL, ↩
       NULL,NULL},{NULL,NULL,NULL,NULL,NULL}}")
   1 |   4 | (1,"{{4,NULL,NULL,NULL,NULL},{NULL,NULL,NULL,NULL,NULL},{NULL,NULL,NULL,NULL, ↩
       NULL},{NULL,NULL,NULL,NULL,NULL},{NULL,NULL,NULL,NULL,4}}")
```

```
(4 rows)
```

L'exemple suivant montre qu'une géométrie située plus loin dans le tableau peut écraser une valeur issue d'une géométrie précédente

```
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
)
SELECT
    t1.rid, t2.gid, t3.gid, ST_DumpValues(ST_SetValues(rast, 1, ARRAY[ROW(t2.geom, t2.gid), ↩
         ROW(t3.geom, t3.gid)]::geomval[]))
FROM foo t1
CROSS JOIN bar t2
CROSS JOIN bar t3
WHERE t2.gid = 1
    AND t3.gid = 2
ORDER BY t1.rid, t2.gid, t3.gid;

 rid | gid | gid |                                                    st_dumpvalues
-----+-----+-----+-----------------------------------------------------------------------------------------

   1 |   1 |   2 | (1,"{{NULL,NULL,NULL,NULL,NULL},{NULL,2,2,2,NULL},{NULL,2,2,2,NULL},{ ↩
       NULL,2,2,2,NULL},{NULL,NULL,NULL,NULL,NULL}}")
(1 row)
```

Cet exemple est le contraire de l'exemple précédent

```
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
)
SELECT
    t1.rid, t2.gid, t3.gid, ST_DumpValues(ST_SetValues(rast, 1, ARRAY[ROW(t2.geom, t2.gid), ↩
         ROW(t3.geom, t3.gid)]::geomval[]))
FROM foo t1
CROSS JOIN bar t2
CROSS JOIN bar t3
WHERE t2.gid = 2
    AND t3.gid = 1
ORDER BY t1.rid, t2.gid, t3.gid;

 rid | gid | gid |                                                    st_dumpvalues
-----+-----+-----+-----------------------------------------------------------------------------------------

   1 |   2 |   1 | (1,"{{NULL,NULL,NULL,NULL,NULL},{NULL,2,2,2,NULL},{NULL,2,1,2,NULL},{ ↩
       NULL,2,2,2,NULL},{NULL,NULL,NULL,NULL,NULL}}")
(1 row)
```

Voir aussi

ST_Value, ST_SetValue, ST_PixelAsPolygons

11.6.14 ST_DumpValues

ST_DumpValues — Retourne les valeurs d'une bande raster spécifiée, sous forme d'un tableau à deux dimensions.

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

Retour les valeurs d'une bande, sous forme d'un tableau à 2 dimensions (le premier index étant la ligne, le second étant la colonne). Si `nband` est NULL ou non spécifié, toutes les bandes raster sont traitées.

Disponibilité : 2.1.0

Exemples

```
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;

 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)
```

```
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;

 nband |                        valarray
-------+-----------------------------------------------------
     3 | {{NULL,NULL,NULL},{NULL,NULL,NULL},{NULL,NULL,NULL}}
     1 | {{1,1,1},{1,1,1},{1,1,1}}
(2 rows)
```

```
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;

 st_dumpvalues
---------------
             5
(1 row)
```

Voir aussi

ST_Value, ST_SetValue, ST_SetValues

11.6.15 ST_PixelOfValue

ST_PixelOfValue — Retourne les coordonnées columnx, rowy du pixel dont la valeur est égale à la valeur recherchée.

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

Retourne les coordonnées columnx, rowy du pixel dont la valeur est égale à la valeur recherchée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Disponibilité : 2.1.0

Exemples

```
SELECT
    (pixels).*
FROM (
    SELECT
        ST_PixelOfValue(
            ST_SetValue(
                ST_SetValue(
                    ST_SetValue(
                        ST_SetValue(
                            ST_SetValue(
                                ST_AddBand(
                                    ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                                    '8BUI'::text, 1, 0
                                ),
                                1, 1, 0
                            ),
                            2, 3, 0
                        ),
                        3, 5, 0
                    ),
                    4, 2, 0
                ),
                5, 4, 255
            )
```

```
        , 1, ARRAY[1, 255]) AS pixels
) AS foo

 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 Éditeurs de raster

11.7.1 ST_SetGeoReference

ST_SetGeoReference — Définit les 6 paramètres de géo-référencement en un seul appel. Les nombres doivent être séparés par un espace. Accepte les formats GDAL (par défaut) ou ESRI.

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

Définit les 6 paramètres de géo-référencement en un seul appel. Accepte les formats 'GDAL' (par défaut) ou 'ESRI'. Si les 6 coordonnées ne sont pas spécifiées, retourne null.

La différence entre les représentations de format est la suivante :

GDAL :

```
scalex skewy skewx scaley upperleftx upperlefty
```

ESRI :

```
scalex skewy skewx scaley upperleftx + scalex*0.5 upperlefty + scaley*0.5
```

Note
Si le raster a des bandes out-db, modifier le géoréférencement peut entraîner un accès incorrect aux données de la bande out-db.

Amélioration : 2.1.0 Ajout de la variante ST_SetGeoReference(raster, double precision, ...)

Exemples

```
WITH foo AS (
    SELECT ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0) AS rast
)
SELECT
    0 AS rid, (ST_Metadata(rast)).*
FROM foo
UNION ALL
SELECT
    1, (ST_Metadata(ST_SetGeoReference(rast, '10 0 0 -10 0.1 0.1', 'GDAL'))).*
FROM foo
UNION ALL
SELECT
    2, (ST_Metadata(ST_SetGeoReference(rast, '10 0 0 -10 5.1 -4.9', 'ESRI'))).*
FROM foo
UNION ALL
SELECT
    3, (ST_Metadata(ST_SetGeoReference(rast, 1, 1, 10, -10, 0.001, 0.001))).*
FROM foo

 rid |     upperleftx     |     upperlefty     | width | height | scalex | scaley | skewx | ↩
     skewy | srid | numbands
-----+--------------------+--------------------+-------+--------+--------+--------+-------+-------+-
   0 |                  0 |                  0 |     5 |      5 |      1 |     -1 |     0 | ↩
         0 |    0 |        0
   1 |                0.1 |                0.1 |     5 |      5 |     10 |    -10 |     0 | ↩
         0 |    0 |        0
   2 | 0.0999999999999996 | 0.0999999999999996 |     5 |      5 |     10 |    -10 |     0 | ↩
         0 |    0 |        0
   3 |                  1 |                  1 |     5 |      5 |     10 |    -10 | 0.001 | ↩
     0.001 |    0 |        0
```

Voir aussi

ST_GeoReference, ST_ScaleX, ST_ScaleY, ST_UpperLeftX, ST_UpperLeftY

11.7.2 ST_SetRotation

ST_SetRotation — Définit la rotation du raster en radians.

Synopsis

raster **ST_SetRotation**(raster rast, float8 rotation);

Description

Applique une rotation uniforme du raster. L'angle de rotation est en radians. Voir World File pour plus de détails.

Exemples

```
SELECT
  ST_ScaleX(rast1), ST_ScaleY(rast1), ST_SkewX(rast1), ST_SkewY(rast1),
  ST_ScaleX(rast2), ST_ScaleY(rast2), ST_SkewX(rast2), ST_SkewY(rast2)
FROM (
  SELECT ST_SetRotation(rast, 15) AS rast1, rast as rast2 FROM dummy_rast
) AS foo;
      st_scalex       |      st_scaley      |      st_skewx       |      st_skewy       |  ↩
         st_scalex | st_scaley | st_skewx | st_skewy
--------------------+---------------------+---------------------+---------------------+-----------+--

   -1.51937582571764 |   -2.27906373857646 |   1.95086352047135 |   1.30057568031423 |  ↩
                2 |          3 |         0 |         0
 -0.0379843956429411 | -0.0379843956429411 | 0.0325143920078558 | 0.0325143920078558 |  ↩
          0.05 |      -0.05 |         0 |         0
```

Voir aussi

ST_Rotation, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.7.3 ST_SetScale

ST_SetScale — Définit la résolution des pixels en X et Y en unité du système de référence spatial : nombre d'unités/pixel en largeur/hauteur.

Synopsis

raster **ST_SetScale**(raster rast, float8 xy);
raster **ST_SetScale**(raster rast, float8 x, float8 y);

Description

Définit la résolution des pixels en X et Y en unité du système de référence spatial : nombre d'unités/pixel en largeur/hauteur. Si une seule unité est spécifiée, la même résolution est utilisée pour X et pour Y.

Note
ST_SetScale est différent de ST_Rescale : ST_SetScale ne rééchantillonne pas le raster pour correspondre à l'étendue du raster. La fonction modifie seulement les méta-données (ou la géo-référence) d'un raster pour corriger une échelle incorrecte. ST_Rescale produit un raster de taille différente pour correspondre à l'étendue spatiale du raster d'entrée. ST_SetScale ne modifie pas la largeur ou hauteur du raster.

Changement : 2.0.0 Dans les versions WKTRaster, cette fonction était appelée ST_SetPixelSize.

Exemples

```
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;
```

```
 pixx | pixy |                    newbox
------+------+-----------------------------------------------
  1.5 |  1.5 | BOX(3427927.75 5793244 0, 3427935.25 5793251.5 0)
```

```
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;

 pixx | pixy |                   newbox
------+------+-------------------------------------------
  1.5 | 0.55 | BOX(3427927.75 5793244 0,3427935.25 5793247 0)
```

Voir aussi

ST_ScaleX, ST_ScaleY, Box3D

11.7.4 ST_SetSkew

ST_SetSkew — Définit l'obliquité X et Y (skew, ou paramètre de rotation). Si une seule valeur est spécifiée, la même valeur est utilisée pour X et pour Y.

Synopsis

raster **ST_SetSkew**(raster rast, float8 skewxy);
raster **ST_SetSkew**(raster rast, float8 skewx, float8 skewy);

Description

Définit l'obliquité X et Y (skew, ou paramètre de rotation). Si une seule valeur est spécifiée, la même valeur est utilisée pour X et pour Y. Voir World File pour plus de détails.

Exemples

```
-- Example 1
UPDATE dummy_rast SET rast = ST_SetSkew(rast,1,2) WHERE rid = 1;
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
    ST_GeoReference(rast) as georef
FROM dummy_rast WHERE rid = 1;

rid | skewx | skewy |    georef
----+-------+-------+--------------
  1 |     1 |     2 | 2.0000000000
                    : 2.0000000000
                    : 1.0000000000
                    : 3.0000000000
                    : 0.5000000000
                    : 0.5000000000
```

```
-- Example 2 set both to same number:
UPDATE dummy_rast SET rast = ST_SetSkew(rast,0) WHERE rid = 1;
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
    ST_GeoReference(rast) as georef
FROM dummy_rast WHERE rid = 1;

 rid | skewx | skewy |    georef
-----+-------+-------+--------------
   1 |     0 |     0 | 2.0000000000
                     : 0.0000000000
                     : 0.0000000000
                     : 3.0000000000
                     : 0.5000000000
                     : 0.5000000000
```

Voir aussi

ST_GeoReference, ST_SetGeoReference, ST_SkewX, ST_SkewY

11.7.5 ST_SetSRID

ST_SetSRID — Modifie le SRID d'un raster à une valeur définie dans la table spatial_ref_sys.

Synopsis

raster **ST_SetSRID**(raster rast, integer srid);

Description

Modifie le SRID d'un raster à une valeur spécifiée.

Note
Cette fonction ne modifie pas le raster en lui-même : elle définit juste la méta-données définissant le système de référence spatial. Ceci est utile pour ensuite effectuer des transformations.

Voir aussi

Section 4.5, ST_SRID

11.7.6 ST_SetUpperLeft

ST_SetUpperLeft — Modifie les coordonnées du pixel du coin supérieur gauche du raster, selon les coordonnées X et Y projetées.

Synopsis

raster **ST_SetUpperLeft**(raster rast, double precision x, double precision y);

Description

Modifie les coordonnées du pixel du coin supérieur gauche du raster aux coordonnées X et Y projetées

Exemples

```
SELECT ST_SetUpperLeft(rast,-71.01,42.37)
FROM dummy_rast
WHERE rid = 2;
```

Voir aussi

ST_UpperLeftX, ST_UpperLeftY

11.7.7 ST_Resample

ST_Resample — Rééchantillonne un raster, en utilisant l'algorithme spécifié, les nouvelles dimensions, un coin arbitraire de la grille et un ensemble de paramètres de géo-référencement définis ou empruntés à un autre 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=NearestNeighbor, 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

Rééchantillonne un raster, en utilisant l'algorithme spécifié, les nouvelles dimensions (width & height), un coin de la grille (gridx & gridy) et un ensemble de paramètres de géo-référencement (scalex, scaley, skewx & skewy) définis ou empruntés à un autre raster. Si un raster de référence est spécifié, les deux rasters doivent avoir le même SRID.

Les nouvelles valeurs des pixels sont calculées par un des algorithmes de rééchantillonnage suivants :

- NearestNeighbor (plus proche voisin)
- Bilinear (Bilinéaire)
- Cubic (Cubique)
- CubicSpline (Cubique Spline)
- Lanczos
- Max
- Min

L'algorithme par défaut est NearestNeighbor, qui est le plus rapide mais donne le moins bon résultat.

Une erreur maximale de 0.125 est utilisée si le paramètre `maxerr` n'est pas spécifié.

Note
Voir GDAL Warp resampling methods pour plus de détails.

Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+

Amélioration : 3.4.0 ajout des options de rééchantillonnage max et min

Exemples

```
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;

 orig_width | new_width
------------+-------------
        200 |         100
```

Voir aussi

ST_Rescale, ST_Resize, ST_Transform

11.7.8 ST_Rescale

ST_Rescale — Rééchantillonne un raster en ajustant juste son échelle (ou la taille des pixels). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline), Lanczos, Max ou Min. La valeur par défaut est 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=NearestNeighbor, double precision maxerr=0.125);

Description

Rééchantillonne un raster en ajustant juste son échelle (ou la taille des pixels). Les nouvelles valeurs des pixels sont calculées en utilisant un des algorithmes de rééchantillonnage suivants :

- NearestNeighbor (plus proche voisin)
- Bilinear (Bilinéaire)
- Cubic (Cubique)
- CubicSpline (Cubique Spline)
- Lanczos
- Max
- Min

L'algorithme par défaut est NearestNeighbor, qui est le plus rapide mais donne le moins bon résultat.

`scalex` et `scaley` définissent la nouvelle taille des pixels. scaley est souvent négatif pour obtenir un raster correctement orienté.

Quand la nouvelle valeur de scalex (respectivement scaley) n'est pas un diviseur de la hauteur (respectivement largeur) du raster, l'étendue du raster résultant est agrandie pour englober l'étendue du raster d'entrée. Si vous voulez conserver l'étendue exacte du raster d'entrée, utilisez ST_Resize

`maxerr` est le seuil pour l'approximation de l'algorithme de rééchantillonnage (en pixels). La valeur de 0.125 est utilisée si le paramètre `maxerr` n'est pas spécifié, qui est la même valeur que celle utilisée par l'utilitaire GDAL gdalwarp. Si définie à 0, aucune approximation n'est effectuée.

Note
Voir GDAL Warp resampling methods pour plus de détails.

Note
ST_Rescale est différent de ST_SetScale : ST_SetScale ne rééchantillonne pas le raster pour correspondre à l'étendue du raster. ST_SetScale modifie seulement les méta-données (ou lagéo-référence) d'un raster pour corriger une échelle incorrecte. ST_Rescale produit un raster de taille différente pour correspondre à l'étendue spatiale du raster d'entrée. ST_SetScale ne modifie pas la largeur ou hauteur du raster.

Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+

Amélioration : 3.4.0 ajout des options de rééchantillonnage max et min

Changement : 2.1.0 Fonctionne avec les rasters sans SRID

Exemples

Exemple simple de rééchantillonnage d'un raster d'une taille de pixel de 0.001 degrés vers une taille de pixel de 0.0015 degrés.

```
-- the original raster pixel size
SELECT ST_PixelWidth(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0,  ↩
    4269), '8BUI'::text, 1, 0)) width

   width
----------
0.001

-- the rescaled raster raster pixel size
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

   width
----------
0.0015
```

Voir aussi

ST_Resize, ST_Resample, ST_SetScale, ST_ScaleX, ST_ScaleY, ST_Transform

11.7.9 ST_Reskew

ST_Reskew — Rééchantillonne un raster en ajustant simplement son obliquité (skew, ou paramètre de rotation). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est 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=NearestNeighbor, double precision maxerr=0.125);

Description

Rééchantillonne un raster en ajustant simplement son obliquité (skew, ou paramètre de rotation). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor, qui est le plus rapide mais donne le moins bon résultat.

`skewx` et `skewy` définissent la nouvelle obliquité.

L'étendue du nouveau raster englobe l'étendu du raster en entrée.

Un pourcentage maximal d'erreur de 0.125 % est utilisé si le paramètre `maxerr` n'est pas spécifié.

Note
Voir GDAL Warp resampling methods pour plus de détails.

Note
ST_Reskew est différent de ST_SetSkew : ST_SetSkew ne rééchantillonne pas le raster pour correspondre à l'étendue du raster. ST_SetSkew modifie seulement les méta-données (ou la géo-référence) du raster pour corriger une obliquité incorrecte. ST_Reskew produit un raster de taille différente pour correspondre à l'étendue spatiale du raster d'entrée. ST_SetSkew ne modifie pas la largeur ou hauteur du raster.

Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+

Changement : 2.1.0 Fonctionne avec les rasters sans SRID

Exemples

Exemple simple de rééchantillonnage d'un raster d'une obliquité de 0.0 à 0.0015.

```
-- the original raster non-rotated
SELECT ST_Rotation(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0, 4269) ↩
    , '8BUI'::text, 1, 0));

-- result
0

-- the reskewed raster raster rotation
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));

-- result
-0.982793723247329
```

Voir aussi

ST_Resample, ST_Rescale, ST_SetSkew, ST_SetRotation, ST_SkewX, ST_SkewY, ST_Transform

11.7.10 ST_SnapToGrid

ST_SnapToGrid — Rééchantillonne un raster en l'accrochant sur une grille. Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.

Synopsis

raster **ST_SnapToGrid**(raster rast, double precision gridx, double precision gridy, text algorithm=NearestNeighbor, 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=NearestNeigh
double precision maxerr=0.125);

Description

Rééchantillonne un raster en l'accrochant sur une grille définie par un coin arbitraire (gridx & gridy) et une taille de pixel optionnelle (scalex & scaley). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor, qui est le plus rapide mais donne le moins bon résultat.

`gridx` et `gridy` définissent le pixel d'un coin arbitraire de la nouvelle grille. Ce n'est pas nécessairement le coin supérieur gauche du nouveau raster, et n'a pas besoin d'être à l'intérieur ou sur les bords de l'emprise du nouveau raster.

Il est possible de définir une taille de pixel pour la nouvelle grille avec les paramètres optionnels `scalex` et `scaley`.

L'étendue du nouveau raster englobe l'étendu du raster en entrée.

Un pourcentage maximal d'erreur de 0.125 % est utilisé si le paramètre `maxerr` n'est pas spécifié.

Note
Voir GDAL Warp resampling methods pour plus de détails.

Note
Voir ST_Resample pour plus de contrôle sur les paramètres de la grille.

Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+

Changement : 2.1.0 Fonctionne avec les rasters sans SRID

Exemples

Exemple simple d'accrochage d'un raster sur une grille légèrement différente.

```
-- the original raster upper left X
SELECT ST_UpperLeftX(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 0.001, -0.001, 0, 0, 4269) ↩
    , '8BUI'::text, 1, 0));
-- result
0

-- the upper left of raster after snapping
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));

--result
-0.0008
```

Voir aussi

ST_Resample, ST_Rescale, ST_UpperLeftX, ST_UpperLeftY

11.7.11 ST_Resize

ST_Resize — Redimensionne un raster à une nouvelle largeur/hauteur

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

Redimensionne un raster à une nouvelle largeur/hauteur. La nouvelle largeur/hauteur peut être spécifiée comme un nombre exact de pixels ou un pourcentage de la taille du raster. L'étendue du nouveau raster est la même que le raster d'entrée.

Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor, qui est le plus rapide mais donne le moins bon résultat.

La variante 1 attend la largeur/hauteur réelle du raster de sortie.

La variante 2 attend des valeurs décimales entre zéro (0) et un (1) indiquant le pourcentage de la largeur/hauteur du raster d'entrée.

La variante 3 prend en compte soit la largeur/hauteur réelle du raster de sortie ou un pourcentage sous forme de texte ("20%") indiquant le pourcentage de la largeur/hauteur du raster d'entrée.

Disponibilité : 2.1.0 Nécessite GDAL 1.6.1+

Exemples

```
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
), bar AS (
    SELECT rid, ST_Metadata(rast) AS meta, rast FROM foo
)
SELECT rid, (meta).* FROM bar

 rid | upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↩
     numbands
-----+------------+------------+-------+--------+--------+--------+-------+-------+------+----------

   1 |          0 |          0 |   500 |    500 |      1 |     -1 |     0 |     0 |    0 |  ↩
             1
   2 |          0 |          0 |   500 |    100 |      1 |     -1 |     0 |     0 |    0 |  ↩
             1
   3 |          0 |          0 |   250 |    900 |      1 |     -1 |     0 |     0 |    0 |  ↩
             1
(3 rows)
```

Voir aussi

ST_Resample, ST_Rescale, ST_Reskew, ST_SnapToGrid

11.7.12 ST_Transform

ST_Transform — Reprojette un raster depuis un système de référence spatial vers un autre, en utilisant l'algorithme de rééchantillonnage spécifié. Les algorithmes possibles sont NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.

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

Reprojette un raster depuis un système de référence spatial vers un autre, en utilisant l'algorithme de rééchantillonnage spécifié. L'algorithme utilisé par défaut est NearestNeighbor si aucun n'est spécifié. Le pourcentage d'erreur maximal est 0.125 si maxerr n'est pas spécifié.

Les algorithmes disponibles sont : 'NearestNeighbor' (plus proche voisin), 'Bilinear' (Bilinéaire), 'Cubic' (Cubique), 'Cubic-Spline' (Cubique Spline) et 'Lanczos'. Voir GDAL Warp resampling methods pour plus de détails.

ST_Transform est souvent confondu avec ST_SetSRID(). ST_Transform modifie réellement les coordonnées du raster (et rééchantillonne les valeurs des pixels) d'un système de référence spatial vers un autre, alors que ST_SetSRID() change uniquement l'identifiant SRID du raster.

Contrairement aux autres variantes, la variante 3 nécessite un raster de référence `alignto`. Le raster de sortie sera transformé dans le système de référence spatial (SRID) du raster de référence et sera aligné (ST_SameAlignment = TRUE) avec le raster de référence.

Note

Si vous constatez que votre support de transformation ne fonctionne pas correctement, il se peut que vous deviez définir la variable d'environnement PROJSO avec la bibliothèque de projection .so ou .dll utilisée par PostGIS. Il suffit que le nom du fichier soit indiqué. Par exemple, sous Windows, dans Panneau de configuration -> Système -> Variables d'environnement, ajoutez une variable système appelée `PROJSO` et définissez-la sur `libproj.dll` (si vous utilisez proj 4.6.1). Vous devrez redémarrer votre service/daemon PostgreSQL après ce changement.

Warning

Lors de la transformation d'une couverture tuilée, vous voudrez en général utiliser un raster de référence pour garantir le même alignement et sans lacunes dans les tuiles, comme le montre l'exemple Variante 3.

Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+

Amélioration : 2.1.0 Ajout de la variante ST_Transform(rast, alignto)

Exemples

```
SELECT ST_Width(mass_stm) As w_before, ST_Width(wgs_84) As w_after,
  ST_Height(mass_stm) As h_before, ST_Height(wgs_84) As h_after
    FROM
    ( SELECT rast As mass_stm, ST_Transform(rast,4326) As wgs_84
  ,  ST_Transform(rast,4326, 'Bilinear') AS wgs_84_bilin
       FROM aerials.o_2_boston
           WHERE ST_Intersects(rast,
               ST_Transform(ST_MakeEnvelope(-71.128, 42.2392,-71.1277, 42.2397, 4326) ↩
                   ,26986) )
       LIMIT 1) As foo;

 w_before | w_after | h_before | h_after
----------+---------+----------+---------
      200 |     228 |      200 |     170
```

mass_stm original

Après transformation en WGS84 long lat (wgs_84)

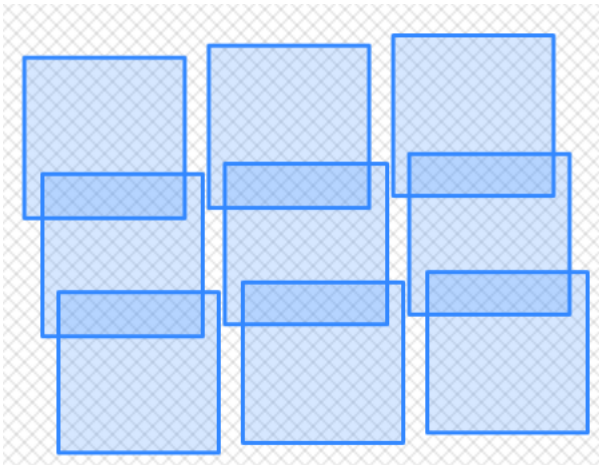
Après transformation en WGS84 long lat avec interpolation bilinéaire au lieu de NN par défaut (wgs_84_bilin)

Exemples : Variante 3

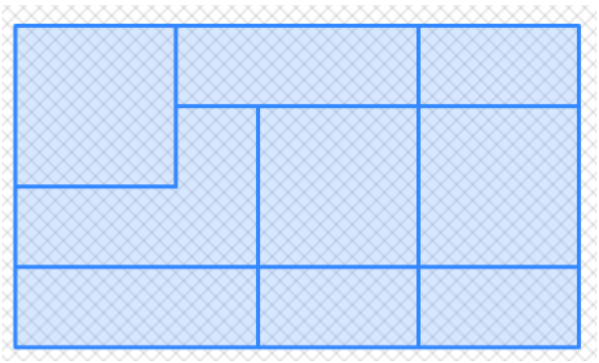
L'exemple suivant montre la différence entre ST_Transform(raster, srid) et ST_Transform(raster, alignto)

```
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_SameAlignment(rast) AS rast,
```

```
    ST_SameAlignment(not_aligned) AS not_aligned,
    ST_SameAlignment(aligned) AS aligned
FROM baz

 rast | not_aligned | aligned
------+-------------+---------
 t    | f           | t
```



Non aligné (not_aligned)



Aligné (aligned)

Voir aussi

ST_Transform, ST_SetSRID

11.8 Éditeurs de bandes raster

11.8.1 ST_SetBandNoDataValue

ST_SetBandNoDataValue — Définit la valeur pour l'absence de données (nodata) pour la bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée. Pour indiquer qu'une bande n'a pas de valeur nodata, définir la valeur nodata = NULL.

Synopsis

raster **ST_SetBandNoDataValue**(raster rast, double precision nodatavalue);
raster **ST_SetBandNoDataValue**(raster rast, integer band, double precision nodatavalue, boolean forcechecking=false);

Description

Définit la valeur pour l'absence de données (nodata) pour la bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée. Cette valeur affecte les résultats de ST_Polygon, ST_DumpAsPolygons, ainsi que les fonctions ST_PixelAs...().

Exemples

```
-- change just first band no data value
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast,1, 254)
WHERE rid = 2;

-- change no data band value of bands 1,2,3
UPDATE dummy_rast
    SET rast =
        ST_SetBandNoDataValue(
            ST_SetBandNoDataValue(
                ST_SetBandNoDataValue(
                    rast,1, 254)
                ,2,99),
                3,108)
        WHERE rid = 2;

-- wipe out the nodata value this will ensure all pixels are considered for all processing functions
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast,1, NULL)
WHERE rid = 2;
```

Voir aussi

ST_BandNoDataValue, ST_NumBands

11.8.2 ST_SetBandIsNoData

ST_SetBandIsNoData — Définit la valeur du flag isnodata de la bande à TRUE.

Synopsis

raster **ST_SetBandIsNoData**(raster rast, integer band=1);

Description

Définit la valeur du flag isnodata de la bande à TRUE. Si aucune bande n'est spécifiée, la bande 1 est utilisée. Cette fonction ne devrait être appelée que si le flag est considéré comme obsolète, i.e. si le résultat de ST_BandIsNoData est différent selon qu'on utilise TRUE comme dernier argument ou non

Disponibilité : 2.0.0

Exemples

```
-- Create dummy table with one raster column
create table dummy_rast (rid integer, rast raster);

-- Add raster with two bands, one pixel/band. In the first band, nodatavalue = pixel value = 3.
-- In the second band, nodatavalue = 13, pixel value = 4
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. We are going to set it to true
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
```

Voir aussi

ST_BandNoDataValue, ST_NumBands, ST_SetBandNoDataValue, ST_BandIsNoData

11.8.3 ST_SetBandPath

ST_SetBandPath — Met à jour le chemin externe et le numéro de bande d'une bande out-db

Synopsis

raster **ST_SetBandPath**(raster rast, integer band, text outdbpath, integer outdbindex, boolean force=false);

Description

Met à jour le chemin externe et le numéro de bande d'une bande externe out-db.

Note
Si `force` est true, aucun test n'est effectué pour garantir la compatibilité (e.g. alignement, support des pixels) entre le fichier raster externe et le raster PostGIS. Ce mode est prévu pour les changements sur le système de fichiers où réside le raster externe.

Disponibilité : 2.5.0

Exemples

```
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,
    *
FROM ST_BandMetadata(
    (SELECT rast FROM foo),
    ARRAY[1,3,2]::int[]
)
UNION ALL
SELECT
    2,
    *
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[]
)
ORDER BY 1, 2;

 query | bandnum | pixeltype | nodatavalue | isoutdb |  ↩
                                        path                                         |  ↩
     outdbbandnum
-------+---------+-----------+-------------+---------+--------------------------------------------
     1 |       1 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
         raster/test/regress/loader/Projected.tif       |            1
     1 |       2 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
         raster/test/regress/loader/Projected.tif       |            2
```

```
 1 |       3 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
     raster/test/regress/loader/Projected.tif       |            3
 2 |       1 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
     raster/test/regress/loader/Projected.tif       |            1
 2 |       2 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
raster/test/regress/loader/Projected2.tif      |            1
 2 |       3 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
     raster/test/regress/loader/Projected.tif       |            3
```

Voir aussi

ST_BandMetaData, ST_SetBandIndex

11.8.4 ST_SetBandIndex

ST_SetBandIndex — Met à jour le numéro de bande externe d'une bande out-db

Synopsis

raster **ST_SetBandIndex**(raster rast, integer band, integer outdbindex, boolean force=false);

Description

Met à jour le numéro de bande externe d'une bande out-db. Cette opération n'affecte pas le fichier raster externe associé à la bande out-db

Note
Si `force` est true, aucun test n'est effectué pour assurer la compatibilité (e.g., l'alignement, le support des pixels) entre le fichier raster externe et le raster PostGIS. Ce mode est destiné aux cas où les bandes sont déplacées dans le fichier raster externe.

Note
En interne, cette méthode remplacement la bande du raster PostGIS à l'index `band` avec la nouvelle bande, au lieu de mettre à jour les informations de chemin existantes.

Disponibilité : 2.5.0

Exemples

```
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,
    *
FROM ST_BandMetadata(
    (SELECT rast FROM foo),
    ARRAY[1,3,2]::int[]
)
```

```
UNION ALL
SELECT
    2,
    *
FROM ST_BandMetadata(
    (
        SELECT
            ST_SetBandIndex(
                rast,
                2,
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1,3,2]::int[]
)
ORDER BY 1, 2;

 query | bandnum | pixeltype | nodatavalue | isoutdb |  ↩
                                          path                                          |  ↩
     outdbbandnum
-------+---------+-----------+-------------+---------+----------------------------------------------

     1 |       1 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
         raster/test/regress/loader/Projected.tif       |            1
     1 |       2 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
         raster/test/regress/loader/Projected.tif       |            2
     1 |       3 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
         raster/test/regress/loader/Projected.tif       |            3
     2 |       1 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
         raster/test/regress/loader/Projected.tif       |            1
     2 |       2 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
    raster/test/regress/loader/Projected.tif       |            1
     2 |       3 | 8BUI      |             | t       | /home/pele/devel/geo/postgis-git/ ↩
         raster/test/regress/loader/Projected.tif       |            3
```

Voir aussi

ST_BandMetaData, ST_SetBandPath

11.9 Statistiques et analyses des bandes raster

11.9.1 ST_Count

ST_Count — Renvoie le nombre de pixels dans une bande donnée d'un raster ou d'une couverture raster. Si aucune bande n'est spécifiée, la valeur par défaut est la bande 1. Si exclude_nodata_value est true, seuls les pixels dont la valeur est différente de la valeur nodata seront comptés.

Synopsis

bigint **ST_Count**(raster rast, integer nband=1, boolean exclude_nodata_value=true);
bigint **ST_Count**(raster rast, boolean exclude_nodata_value);

Description

Renvoie le nombre de pixels dans une bande donnée d'un raster ou d'une couverture raster. Si aucune bande `nband` n'est spécifiée, la valeur par défaut est 1.

Note
Si `exclude_nodata_value` est true, seuls les pixels dont la valeur est différente de la valeur `nodata` du raster seront comptés. Définir `exclude_nodata_value` à false pour compter tous les pixels

Changement : 3.1.0 - Suppression des variantes ST_Count(rastertable, rastercolumn, ...). Utiliser ST_CountAgg à la place.

Disponibilité : 2.0.0

Exemples

```
--example will count all pixels not 249 and one will count all pixels.  --
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;

rid | exclude_nodata | include_nodata
-----+----------------+----------------
   2 |             23 |             25
```

Voir aussi

ST_CountAgg, ST_SummaryStats, ST_SetBandNoDataValue

11.9.2 ST_CountAgg

ST_CountAgg — Agrégat. Renvoie le nombre de pixels dans une bande donnée d'un ensemble de rasters. Si aucune bande n'est spécifiée, la valeur par défaut est la bande 1. Si exclude_nodata_value est true, seuls les pixels différents de la valeur NODATA seront comptés.

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

Renvoie le nombre de pixels dans une bande donnée d'un ensemble de rasters. Si aucune bande `nband` n'est spécifiée, la valeur par défaut est 1.

Si `exclude_nodata_value` est true, seuls les pixels dont la valeur est différente de la valeur `nodata` du raster seront comptés. Définir `exclude_nodata_value` à false pour compter tous les pixels

Par défaut, tous les pixels sont échantillonnés. Pour obtenir une réponse plus rapide, définir `sample_percent` à une valeur comprise entre zéro (0) et un (1)

Disponibilité : 2.2.0

Exemples

```
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)
FROM foo;

 st_countagg
-------------
          20
(1 row)
```

Voir aussi

ST_Count, ST_SummaryStats, ST_SetBandNoDataValue

11.9.3 ST_Histogram

ST_Histogram — Retourne un ensemble d'enregistrements résumant une distribution de données raster ou de couverture raster, dans des classes distinctes. Le nombre de classes est calculé automatiquement s'il n'est pas spécifié.

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

Retourne un ensemble d'enregistrements composés de min, max, count, percent pour une bande raster donnée pour chaque classe. Si aucune bande `nband` n'est spécifiée, la bande 1 est utilisée.

Note
Par défaut, seules les valeurs de pixels différentes de la valeur `nodata` sont prises en compte. Définir `exclude_nodata_value` à false pour obtenir le comptage de tous les pixels.

`width` width : un tableau indiquant la largeur de chaque classe. Si le nombre de classes est supérieur au nombre de largeurs, les largeurs sont répétées.

Exemple : 9 classes, les largeurs étant [a, b, c], le résultat sera [a, b, c, a, b, c, a, b, c]

`bins` Nombre de classes -- c'est le nombre d'enregistrements retournés par la fonction si spécifié. Si non spécifié, le nombre de classes est calculé automatiquement.

`right` calcule l'histogramme à partir de la droite plutôt que de la gauche (par défaut). Cela modifie les critères d'évaluation d'une valeur x de [a, b) à (a, b]

Changement : 3.1.0 Suppression de la variante ST_Histogram(table_name, column_name).

Disponibilité : 2.0.0

Exemple : Une seule tuile raster - calcul des histogrames pour les bandes 1, 2, 3 avec calcul automatique des classes

```
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;

 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
```

Exemple : Seulement la bande 2, mais avec 6 classes

```
SELECT (stats).*
FROM (SELECT rid, ST_Histogram(rast, 2,6) As stats
    FROM dummy_rast
     WHERE rid=2) As foo;

    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.
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;

  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)
```

Voir aussi

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg

11.9.4 ST_Quantile

ST_Quantile — Calcule les quantiles d'un raster ou d'une couverture raster, dans le contexte de l'échantillon ou de la population. Ainsi, une valeur peut être examinée pour se situer au percentile de 25%, 50% ou 75% du raster.

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

Calcule les quantiles d'un raster ou d'une couverture raster, dans le contexte de l'échantillon ou de la population. Ainsi, une valeur peut être examinée pour se situer au percentile de 25%, 50% ou 75% du raster.

Note
Si `exclude_nodata_value` est false, les pixels de valeur nodata seront également pris en compte.

Changement : 3.1.0 Suppression de la variante ST_Quantile(table_name, column_name).

Disponibilité : 2.0.0

Exemples

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast,249) WHERE rid=2;
--Example will consider only pixels of band 1 that are not 249 and in named quantiles --

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;

 quantile | value
----------+-------
     0.25 |   253
     0.75 |   254

SELECT ST_Quantile(rast, 0.75) As value
    FROM dummy_rast WHERE rid=2;

value
------
  254
```

```
--real live example.  Quantile of all pixels in band 2 intersecting a geometry
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
;


 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
```

Voir aussi

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg, ST_SetBandNoDataValue

11.9.5 ST_SummaryStats

ST_SummaryStats — Retourne des résumés statistiques (count, sum, mean, stddev, min, max) pour une bande raster ou une couverture raster spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

Synopsis

summarystats **ST_SummaryStats**(raster rast, boolean exclude_nodata_value);
summarystats **ST_SummaryStats**(raster rast, integer nband, boolean exclude_nodata_value);

Description

Retourne un résumé statistique summarystats composé de count, sum, mean, stddev, min, max pour une bande raster ou une couverture raster spécifiée. Si aucune bande `nband` n'est spécifiée, la bande 1 est utilisée.

Note
Par défaut, seules les valeurs de pixels différentes de la valeur `nodata` sont prises en compte. Définir `exclude_nodata_value` à false pour obtenir le comptage de tous les pixels.

Note
Par défaut, tous les pixels sont échantillonnés. Pour obtenir une réponse plus rapide, définir `sample_percent` à une valeur inférieure à 1

Changement : 3.1.0 Suppression des variantes ST_SummaryStats(rastertable, rastercolumn, ...). Utiliser ST_SummaryStatsAgg à la place.

Disponibilité : 2.0.0

Exemple : Une seule tuile raster

```
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;

 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
```

Exemple : Statistiques des pixels qui intersectent des bâtiments d'intérêt

Cet exemple a pris 574 ms sur PostGIS Windows 64-bit avec tous les bâtiments de Boston et les tuiles aériennes (tuiles de 150x150 pixels chacune ~ 134 000 tuiles), ~102 000 enregistrements de bâtiments

```
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;
 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
```

Exemple : Couverture raster

```
-- stats for each band --
SELECT band, (stats).*
FROM (SELECT band, ST_SummaryStats('o_4_boston','rast', band) As stats
    FROM generate_series(1,3) As band) As foo;

 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 -- will get better speed if set sampling to less than 100%
-- Here we set to 25% and get a much faster answer
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;

 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
```

Voir aussi

summarystats, ST_SummaryStatsAgg, ST_Count, ST_Clip

11.9.6 ST_SummaryStatsAgg

ST_SummaryStatsAgg — Agrégat. Retourne des résumés statistiques (count, sum, mean, stddev, min, max) pour une bande raster spécifiée pour une ensemble de rasters. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

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

Retourne un résumé statistique summarystats composé de count, sum, mean, stddev, min, max pour une bande raster ou une couverture raster spécifiée. Si aucune bande `nband` n'est spécifiée, la bande 1 est utilisée.

Note
Par défaut, seules les valeurs de pixels différentes de la valeur `NODATA` sont prises en compte. Définir `exclude_nodata_value` à false pour obtenir le comptage de tous les pixels.

Note
Par défaut, tous les pixels sont échantillonnés. Pour obtenir une réponse plus rapide, définir `sample_percent` à une valeur comprise entre zéro (0) et un (1)

Disponibilité : 2.2.0

Exemples

```
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,
    round((stats).sum::numeric, 3),
    round((stats).mean::numeric, 3),
    round((stats).stddev::numeric, 3),
    round((stats).min::numeric, 3),
    round((stats).max::numeric, 3)
FROM (
    SELECT
        ST_SummaryStatsAgg(rast, 1, TRUE, 1) AS stats
    FROM foo
) bar;

 count |  round   | round  | round |  round   | round
-------+---------+--------+-------+---------+-------
    20 | -68.584 | -3.429 | 6.571 | -10.000 | 3.142
(1 row)
```

Voir aussi

summarystats, ST_SummaryStats, ST_Count, ST_Clip

11.9.7 ST_ValueCount

ST_ValueCount — Retourne un ensemble d'enregistrements contenant une valeur de pixels et le nombre de pixels ayant cette valeur dans la bande du raster spécifié (ou de la couverture raster). Si aucune bande n'est spécifiée, la bande 1 est utilisée. Par défaut, les pixels de valeur nodata ne sont pas comptés, et toutes les autres valeurs sont retournées, avec leur valeur arrondies à l'entier le plus proche.

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

Retourne un ensemble d'enregistrements contenant les colonnes `value` `count`, correspondant à la valeur de pixel dans la bande et le nombre de pixels ayant cette valeur dans la bande du raster spécifié (ou de la couverture raster).

Si aucune bande `nband` n'est spécifiée, la bande 1 est utilisée. Si aucune valeur `searchvalues` n'est spécifiée, retourne toutes les valeurs de pixel trouvées dans le raster ou dans la couverture raster. Si une valeur searchvalue est spécifiée, retourne un entier au lieu d'enregistrements, correspondant au nombre de pixels ayant la valeur recherchée

Note
Si `exclude_nodata_value` est false, les pixels de valeur nodata seront également pris en compte.

Disponibilité : 2.0.0

Exemples

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast,249) WHERE rid=2;
--Example will count only pixels of band 1 that are not 249. --

SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;

 value | count
-------+-------
   250 |     2
   251 |     1
   252 |     2
   253 |     6
   254 |    12

-- Example will coount all pixels of band 1 including 249 --
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast,1,false) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;

 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
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast,2) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;
 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
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;

 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 --
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)
             ) ;

 rid | count
-----+-------
   1 |    56
   2 |    95
  14 |    37
  15 |    64
```

Voir aussi

ST_Count, ST_SetBandNoDataValue

11.10 Import de raster

11.10.1 ST_RastFromWKB

ST_RastFromWKB — Retourne un raster à partir d'un raster Well-Known Binary (WKB).

Synopsis

raster **ST_RastFromWKB**(bytea wkb);

Description

Construit un raster PostGIS à partir d'une représentation raster Well-Known Binary (WKB).

Disponibilité : 2.5.0

Exemples

```
SELECT (ST_Metadata(
    ST_RastFromWKB(
        '\001\000\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\0
            bytea
    )
)).* AS metadata;

 upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↩
     numbands
------------+------------+-------+--------+--------+--------+-------+-------+------+---------- ↩

        0.5 |        0.5 |    10 |     20 |      2 |      3 |     0 |     0 |   10 |  ↩
                  0
```

Voir aussi

ST_MetaData, ST_RastFromHexWKB, ST_AsBinary/ST_AsWKB, ST_AsHexWKB

11.10.2 ST_RastFromHexWKB

ST_RastFromHexWKB — Retourne un raster à partir d'un raster Well-Known Binary (WKB) en hexadécimal.

Synopsis

raster **ST_RastFromHexWKB**(text wkb);

Description

Construit un raster PostGIS à partir d'une représentation hexadécimale raster Well-Known Binary (WKB).

Disponibilité : 2.5.0

Exemples

```
SELECT (ST_Metadata(
    ST_RastFromHexWKB(
        '0100000000000000000000004000000000000008400000000000000 ↩
            E03F000000000000E03F0000000000000000000000000000000A0000000A001400'
    )
)).* AS metadata;

 upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↩
     numbands
------------+------------+-------+--------+--------+--------+-------+-------+------+---------- ↩

        0.5 |        0.5 |    10 |     20 |      2 |      3 |     0 |     0 |   10 |  ↩
                  0
```

Voir aussi

ST_MetaData, ST_RastFromWKB, ST_AsBinary/ST_AsWKB, ST_AsHexWKB

11.11 Export de raster

11.11.1 ST_AsBinary/ST_AsWKB

ST_AsBinary/ST_AsWKB — Retourne la représentation Well-Known Binary (WKB) de ce raster.

Synopsis

bytea **ST_AsBinary**(raster rast, boolean outasin=FALSE);
bytea **ST_AsWKB**(raster rast, boolean outasin=FALSE);

Description

Retourne la représentation binaire du raster. Si `outasin` est TRUE, les bandes out-db sont traitées comme in-db. Voir raster/doc/RFC2-WellKnownBinaryFormat situé dans le dossier source de PostGIS pour plus de détails sur la représentation.

Cette fonction est utile dans les curseurs binaires pour extraire des données de la base de données sans les convertir en une représentation sous forme de chaîne de caractères.

Note
Par défaut, la sortie WKB contient le chemin d'accès au fichier externe pour les bandes out-db. Si le client n'a pas accès au fichier raster sous-jacent d'une bande out-db, définir `outasin` à TRUE.

Amélioration : 2.1.0 Ajout de `outasin`

Amélioration : 2.5.0 Ajout de `ST_AsWKB`

Exemples

```
SELECT ST_AsBinary(rast) As rastbin FROM dummy_rast WHERE rid=1;

                     rastbin
---------------------------------------------------------------------------
\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\00
```

Voir aussi

ST_RastFromWKB, ST_AsHexWKB

11.11.2 ST_AsHexWKB

ST_AsHexWKB — Retourne la représentation hexadécimale Well-Known Binary (WKB) de ce raster.

Synopsis

bytea **ST_AsHexWKB**(raster rast, boolean outasin=FALSE);

Description

Retourne la représentation binaire en hexadécimal du raster. Si `outasin` est TRUE, les bandes out-db sont traitées comme in-db. Voir raster/doc/RFC2-WellKnownBinaryFormat situé dans le dossier source de PostGIS pour les détails de la représentation.

> **Note**
> Par défaut, la sortie Hex WKB contient le chemin d'accès au fichier externe pour les bandes out-db. Si le client n'a pas accès au fichier raster sous-jacent de la bande out-db, définir `outasin` à TRUE.

Disponibilité : 2.5.0

Exemples

```
SELECT ST_AsHexWKB(rast) As rastbin FROM dummy_rast WHERE rid=1;

                                                        st_ashexwkb
-----------------------------------------------------------------------------------------------------

 0100000000000000000000004000000000000000840000000000000 ↩
    E03F000000000000E03F00000000000000000000000000000000A0000000A001400
```

Voir aussi

ST_RastFromHexWKB, ST_AsBinary/ST_AsWKB

11.11.3 ST_AsGDALRaster

ST_AsGDALRaster — Retourne la tuile raster dans le format GDAL raster spécifié. Les formats raster disponibles sont ceux supportés par votre bibliothèque compilée. Utilisez ST_GDALDrivers() pour obtenir la liste des formats supportés par votre bibliothèque.

Synopsis

bytea **ST_AsGDALRaster**(raster rast, text format, text[] options=NULL, integer srid=sameassource);

Description

Retourne la tuile raster dans le format spécifié. Les paramètres sont détaillés ci-dessous :

- `format` format de sortie. Cela dépend des drivers compilés dans votre bibliothèque libgdal. Les formats généralement disponibles sont 'JPEG', 'GTIff', 'PNG'. Utilisez ST_GDALDrivers pour obtenir une liste des formats pris en charge par votre bibliothèque.
- `options` tableau textuel d'options GDAL. Les options valides dépendent du format. Pour plus de détails, voir GDAL Raster format options.
- `srs` La chaîne proj4text ou srtext (issue de spatial_ref_sys) à incorporer dans l'image

Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0.

Exemple de sortie JPEG, plusieurs tuiles en un seul raster

```
SELECT ST_AsGDALRaster(ST_Union(rast), 'JPEG', ARRAY['QUALITY=50']) As rastjpg
FROM dummy_rast
WHERE rast && ST_MakeEnvelope(10, 10, 11, 11);
```

Utilisation de PostgreSQL Large Object Support pour exporter des données raster

Une façon d'exporter des données raster vers un autre format est d'utiliser les fonctions d'exportation de grands objets de PostgreSQL. Nous allons reprendre l'exemple précédent en ajoutant l'export. Notez que vous devrez avoir un accès super utilisateur à la base de données car elle utilise les fonctions lo côté serveur. L'exportation se fera également vers un chemin d'accès sur le réseau du serveur. Si vous avez besoin d'exporter localement, utilisez les fonctions lo_ équivalentes de psql qui exportent vers le système de fichiers local au lieu du système de fichiers du serveur.

```
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;
```

Exemples de sorties GTIFF

```
SELECT ST_AsGDALRaster(rast, 'GTiff') As rastjpg
FROM dummy_rast WHERE rid=2;

-- Out GeoTiff with jpeg compression, 90% quality
SELECT ST_AsGDALRaster(rast, 'GTiff',
  ARRAY['COMPRESS=JPEG', 'JPEG_QUALITY=90'],
  4269) As rasttiff
FROM dummy_rast WHERE rid=2;
```

Voir aussi

Section 10.3, ST_GDALDrivers, ST_SRID

11.11.4 ST_AsJPEG

ST_AsJPEG — Retourne les bandes sélectionnées du raster sous la forme d'une image JPEG (sous forme de tableau d'octets). Si aucune bande n'est spécifiée, et que le raster a 1 ou plus de 3 bandes, seule la première bande est utilisée. Si le raster a exactement 3 bandes, les 3 bandes sont utilisées et mappées en 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

Retourne les bandes sélectionnées du raster sous la forme d'une seule image JPEG (Joint Photographic Exports Group Image). Utilisez ST_AsGDALRaster si vous avez besoin d'exporter vers des types de raster moins courants. Si aucune bande n'est spécifiée, et que le raster a 1 ou plus de 3 bandes, seule la première bande est utilisée. Si le raster a exactement 3 bandes, les 3 bandes sont utilisées. Il existe de nombreuses variantes de la fonction avec de nombreuses options. Celles-ci sont détaillées ci-dessous :

- `nband` pour exporter une seule bande.
- `nbands` spécifie un tableau des bandes à exporter (à noter que JPEG ne supporte que 3 bandes aux maximum). L'ordre des bandes est RGB. Par exemple, ARRAY[3,2,1] signifie que la bande 3 est la rouge, la bande 2 est la verte et la bande 1 est la bleue
- `quality` nombre de 0 à 100. Plus le chiffre est élevé, plus l'image est nette.
- `options` tableau textuel d'options GDAL, parmi les options définies pour JPEG (voir create_options pour JPEG de ST_GDALDrivers Pour JPEG, les options disponibles sont `PROGRESSIVE` ON ou OFF et `QUALITY` une valeur de 0 à 100 et par défaut 75. Voir GDAL Raster format options pour plus de détails.

Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0.

Exemples : Export

```
-- output first 3 bands 75% quality
SELECT ST_AsJPEG(rast) As rastjpg
    FROM dummy_rast WHERE rid=2;

-- output only first band as 90% quality
SELECT ST_AsJPEG(rast,1,90) As rastjpg
    FROM dummy_rast WHERE rid=2;

-- output first 3 bands (but make band 2 Red, band 1 green, and band 3 blue, progressive ↩
    and 90% quality
SELECT ST_AsJPEG(rast,ARRAY[2,1,3],ARRAY['QUALITY=90','PROGRESSIVE=ON']) As rastjpg
    FROM dummy_rast WHERE rid=2;
```

Voir aussi

Section 10.3, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsTIFF

11.11.5 ST_AsPNG

ST_AsPNG — Retourne les bandes sélectionnées du raster sous la forme d'une image PNG (sous forme de tableau d'octets). Si aucune bande n'est spécifiée et que le raster a 1, 3 ou 4 bandes, toutes les bandes sont utilisées. Si aucune bande n'est spécifiée et que le raster a 2 ou plus de 4 bandes, seule la bande 1 est utilisée. Les bandes sont mappées en RGB ou RGBA.

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

Retourne les bandes sélectionnées du raster sous la forme d'une seule PNG (image graphique portable). Utilisez ST_AsGDALRaster si vous avez besoin d'exporter vers des types de raster moins courants. Si aucune bande n'est spécifiée, les 3 premières bandes sont exportées. Il existe de nombreuses variantes de cette fonction avec de nombreuses options. Si aucun `srid` n'est spécifié, c'est le srid du raster qui est utilisé. Ces options sont détaillées ci-dessous :

- `nband` pour exporter une seule bande.
- `nbands` spécifie un tableau des bandes à exporter (à noter que PNG ne supporte que 4 bandes aux maximum). L'ordre des bandes est RGBA. Par exemple, ARRAY[3,2,1] signifie que la bande 3 est la rouge, la bande 2 est la verte et la bande 1 est la bleue
- `compression` nombre de 1 à 9. Plus le chiffre est élevé, plus la compression est importante.
- `options` tableau textuel d'options GDAL, parmi les options définies pour PNG (voir create_options pour PNG de ST_GDALDrivers) Pour PNG, seule l'option ZLEVEL est valide (temps à consacrer à la compression -- par défaut 6), par exemple ARRAY['ZLEVEL=9'] WORLDFILE n'est pas autorisé car la fonction devrait produire deux sorties. Voir GDAL Raster format options pour plus de détails.

Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0.

Exemples

```
SELECT ST_AsPNG(rast) As rastpng
FROM dummy_rast WHERE rid=2;

-- export the first 3 bands and map band 3 to Red, band 1 to Green, band 2 to blue
SELECT ST_AsPNG(rast, ARRAY[3,1,2]) As rastpng
FROM dummy_rast WHERE rid=2;
```

Voir aussi

ST_AsGDALRaster, ST_ColorMap, ST_GDALDrivers, Section 10.3

11.11.6 ST_AsTIFF

ST_AsTIFF — Retourne les bandes sélectionnées du raster sous la forme d'une seule image TIFF (sous forme de tableau d'octets). Si aucune bande n'est spécifiée ou si l'une des bandes spécifiées n'existe pas dans le raster, toutes les bandes sont utilisées.

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

Retourne les bandes sélectionnées du raster sous la forme d'un fichier TIFF (Tagged Image File Format). Si aucune bande n'est spécifiée, toutes les bandes seront utilisées. Cette fonction est une enveloppe autour de ST_AsGDALRaster. Utilisez ST_AsGDALRaster si vous avez besoin d'exporter vers des types de raster moins courants. Il existe de nombreuses variantes de la fonction avec de nombreuses options. Si aucun texte SRS de référence spatiale n'est présent, la référence spatiale du raster est utilisée. Ces options sont détaillées ci-dessous :

- `nbands` spécifie un tableau des bandes à exporter (à noter que PNG ne supporte que 3 bandes aux maximum). L'ordre des bandes est RGB. Par exemple, ARRAY[3,2,1] signifie que la bande 3 est la rouge, la bande 2 est la verte et la bande 1 est la bleue
- `compression` Expression de compression -- JPEG90 (ou un autre pourcentage), LZW, JPEG, DEFLATE9.
- `options` tableau textuel d'options GDAL, parmi les options définies pour GTiff (voir create_options pour GTiff de ST_GDALDrivers Voir GDAL Raster format options pour plus de détails.
- `srid` srid du système de référence spatial du raster. Il est utilisé pour remplir les informations de géo-référence

Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0.

Exemple : Utiliser la compression jpeg à 90%

```
SELECT ST_AsTIFF(rast, 'JPEG90') As rasttiff
FROM dummy_rast WHERE rid=2;
```

Voir aussi

ST_GDALDrivers, ST_AsGDALRaster, ST_SRID

11.12 Traitement des données raster : algèbre cartographique

11.12.1 ST_Clip

ST_Clip — Returns the raster clipped by the input geometry. If band number is not specified, all bands are processed. If `crop` is not specified or TRUE, the output raster is cropped. If `touched` is set to TRUE, then touched pixels are included, otherwise only if the center of the pixel is in the geometry it is included.

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

Retourne un raster coupé par la géométrie d'entrée `geom`. Si l'index des bandes n'est pas spécifié, toutes les bandes sont traitées.

Les rasters résultant de ST_Clip doivent avoir une valeur de nodata assignée pour les zones découpées, une pour chaque bande. Si aucune valeur n'est fournie et que le raster d'entrée n'a pas de valeur de nodata définie, les valeurs de nodata du raster résultant sont fixées à ST_MinPossibleValue(ST_BandPixelType(rast, band)). Lorsque la taille du tableau nodata inférieure au nombre de bandes, la dernière valeur du tableau est utilisée pour les bandes restantes. Si la taille du tableau nodata est supérieure au nombre de bandes, les valeurs de nodata supplémentaires sont ignorées. Toutes les variantes acceptant un tableau de valeurs de nodata acceptent également une valeur unique qui sera assignée à chaque bande.

Si `crop` n'est pas spécifié ou si défini à true, le raster de sortie est recadré à l'intersection des extensions `geom`et `rast`. Si `crop` est défini à false, le nouveau raster a la même étendue que `rast`. Si `touched` est défini comme vrai, tous les pixels de `rast` qui intersectent la géométrie sont sélectionnés.

Note
Le comportement par défaut est touched=false, ce qui sélectionnera uniquement les pixels dont le centre est couvert par la géométrie.

Amélioration : 3.5.0 - ajout de l'argument touched.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Réécrit en C

Les exemples présentés ici utilisent les données aériennes du Massachusetts disponibles sur le site de MassGIS MassGIS Aerial Orthos.

Exemples : Comparaison entre la sélection des pixels intersectés et non

```
SELECT ST_Count(rast) AS count_pixels_in_orig, ST_Count(rast_touched) AS all_touched_pixels ↩
    , ST_Count(rast_not_touched) AS default_clip
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)
 , ST_Clip(r.rast, g.geom, touched =
> true) AS rast_touched
 , ST_Clip(r.rast, g.geom, touched =
> false) AS rast_not_touched;

 count_pixels_in_orig | all_touched_pixels | default_clip
----------------------+--------------------+--------------
                 2605 |                 16 |           10
(1 row)
```

Exemples : Coupe d'une bande (non modifiée)

```
-- Clip the first band of an aerial tile by a 20 meter buffer.
SELECT ST_Clip(rast, 1,
        ST_Buffer(ST_Centroid(ST_Envelope(rast)),20)
    ) from aerials.boston
WHERE rid = 4;
```

```
-- Demonstrate effect of crop on final dimensions of raster
-- Note how final extent is clipped to that of the geometry
-- if crop = true
SELECT ST_XMax(ST_Envelope(ST_Clip(rast, 1, clipper, true))) As xmax_w_trim,
    ST_XMax(clipper) As xmax_clipper,
    ST_XMax(ST_Envelope(ST_Clip(rast, 1, clipper, false))) As xmax_wo_trim,
    ST_XMax(ST_Envelope(rast)) As xmax_rast_orig
FROM (SELECT rast, ST_Buffer(ST_Centroid(ST_Envelope(rast)),6) As clipper
    FROM aerials.boston
WHERE rid = 6) As foo;

   xmax_w_trim    |   xmax_clipper   |   xmax_wo_trim   |  xmax_rast_orig
------------------+------------------+------------------+------------------
 230657.436173996 | 230657.436173996 | 230666.436173996 | 230666.436173996
```

Tuile raster complète avant découpe

Après découpe

Exemples : Coupe d'une bande sans recadrage et ajout d'autres bandes inchangées

```
-- Same example as before, but we need to set crop to false to be able to use ST_AddBand
-- because ST_AddBand requires all bands be the same Width and height
SELECT ST_AddBand(ST_Clip(rast, 1,
        ST_Buffer(ST_Centroid(ST_Envelope(rast)),20),false
    ), ARRAY[ST_Band(rast,2),ST_Band(rast,3)] ) from aerials.boston
WHERE rid = 6;
```

Tuile raster complète avant découpe

Après découpe - surréaliste

Exemples : Coupe de toutes les bandes

```
-- Clip all bands of an aerial tile by a 20 meter buffer.
-- Only difference is we don't specify a specific band to clip
-- so all bands are clipped
SELECT ST_Clip(rast,
      ST_Buffer(ST_Centroid(ST_Envelope(rast)), 20),
      false
    ) from aerials.boston
WHERE rid = 4;
```

Tuile raster complète avant découpe

Après découpe

Voir aussi

ST_AddBand, ST_Count, ST_MapAlgebra (callback function version), ST_Intersection

11.12.2 ST_ColorMap

ST_ColorMap — Crée un nouveau raster comprenant jusqu'à quatre bandes 8BUI (niveaux de gris, RGB, RGBA) à partir du raster source et d'une bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

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

Appliquer une palette de couleurs `colormap` à la bande `nband` du raster `rast`, pour obtenir un nouveau raster comprenant jusqu'à quatre bandes 8BUI. Le nombre de bandes 8BUI dans le nouveau raster est déterminé par le nombre de composantes de couleur définies dans `colormap`.

Si `nband` n'est pas spécifié, la bande 1 est utilisée.

`colormap` peut être un mot-clé d'une palette de couleurs prédéfinie, ou un ensemble de lignes définissant la valeur et les composants de couleur.

Valeurs prédéfinies disponibles pour `colormap` :

- `grayscale` ou `greyscale` pour un raster à 1 bande 8BUI en nuances de gris.
- `pseudocolor` pour un raster à 4 bandes 8BUI (RGBA) avec des couleurs allant du bleu au rouge en passant par le vert.
- `fire` pour un raster à 4 bandes 8BUI (RGBA) avec des couleurs allant du noir au rouge en passant par le jaune pâle.
- `bluered` pour un raster à 4 bandes 8BUI (RGBA) avec des couleurs allant du bleu au rouge en passant par le blanc pâle.

Vous pouvez passer un ensemble d'entrées (une par ligne) à `colormap` pour spécifier des palettes de couleurs personnalisées. Chaque entrée se compose généralement de cinq valeurs : la valeur du pixel et les composantes rouge, verte, bleue et alpha correspondantes (composantes de couleur comprises entre 0 et 255). Des valeurs en pourcentage peuvent être utilisées à la place des valeurs en pixels, où 0% et 100% sont les valeurs minimales et maximales trouvées dans la bande raster. Les valeurs peuvent être séparées par des virgules (','), des tabulations, des deux-points (':') et/ou des espaces. La valeur du pixel peut être fixée à *nv*, *null* ou *nodata* pour la valeur NODATA. Un exemple est fourni ci-dessous.

```
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
```

La syntaxe de `colormap` est similaire à celle du mode couleur du relief de GDAL gdaldem.

Valeurs disponibles pour `method` :

- `INTERPOLATE` pour utiliser l'interpolation linéaire afin de mélanger les couleurs entre les valeurs de pixels données
- `EXACT` pour correspondre strictement aux valeurs des pixels trouvés dans la palette de couleurs. Les pixels dont la valeur ne correspond pas à une entrée de la palette de couleurs seront mis à 0 0 0 0 (RGBA)
- `NEAREST` pour utiliser l'entrée de la palette de couleurs dont la valeur est la plus proche de la valeur du pixel

Note
Une excellente référence pour les palette de couleurs est ColorBrewer.

Warning
Les bandes résultantes du nouveau raster n'auront pas de valeur NODATA définie. Utilisez ST_SetBandNoDataValue pour définir une valeur NODATA si nécessaire.

Disponibilité : 2.1.0

Exemples

Définit une table de bric-à-brac pour jouer

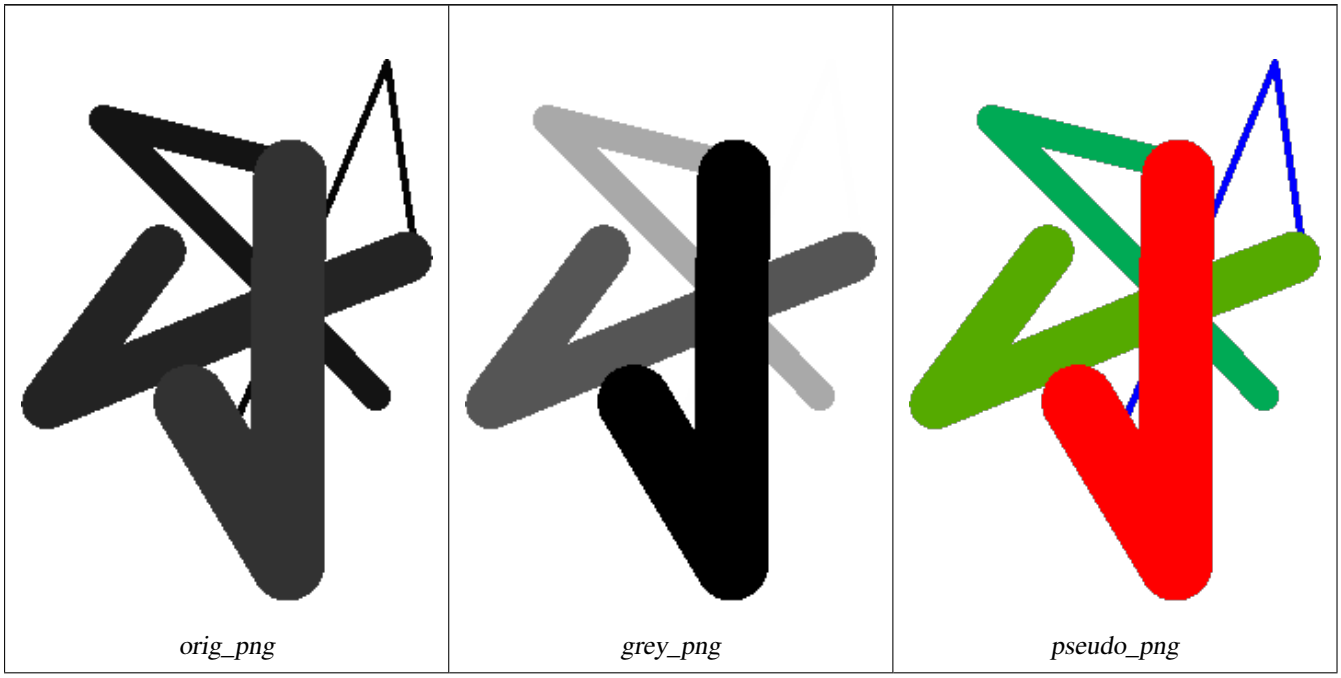
```
-- setup test raster table --
DROP TABLE IF EXISTS funky_shapes;
CREATE TABLE funky_shapes(rast raster);

INSERT INTO funky_shapes(rast)
WITH ref AS (
    SELECT ST_MakeEmptyRaster( 200, 200, 0, 200, 1, -1, 0, 0) AS rast
)
SELECT
    ST_Union(rast)
FROM (
    SELECT
        ST_AsRaster(
            ST_Rotate(
                ST_Buffer(
                    ST_GeomFromText('LINESTRING(0 2,50 50,150 150,125 50)'),
                    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
    ST_NumBands(rast) As n_orig,
    ST_NumBands(ST_ColorMap(rast,1, 'greyscale')) As ngrey,
    ST_NumBands(ST_ColorMap(rast,1, 'pseudocolor')) As npseudo,
    ST_NumBands(ST_ColorMap(rast,1, 'fire')) As nfire,
    ST_NumBands(ST_ColorMap(rast,1, 'bluered')) As nbluered,
    ST_NumBands(ST_ColorMap(rast,1, '
100% 255   0   0
 80% 160   0   0
 50% 130   0   0
 30%  30   0   0
 20%  60   0   0
  0%   0   0   0
  nv 255 255 255
    ')) As nred
FROM funky_shapes;
```

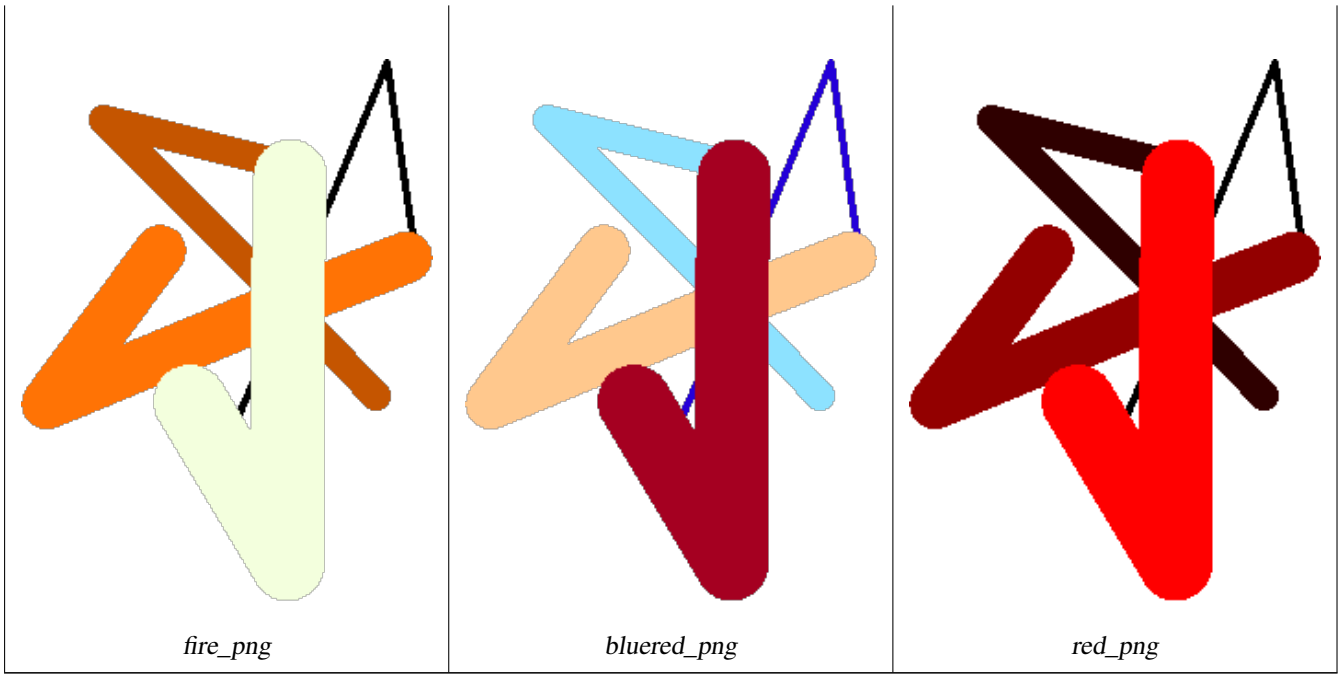
```
n_orig | ngrey | npseudo | nfire | nbluered | nred
--------+-------+---------+-------+----------+------
      1 |     1 |       4 |     4 |        4 |    3
```

Exemples : Comparaison de différentes palettes de couleurs via ST_AsPNG

```
SELECT
    ST_AsPNG(rast) As orig_png,
    ST_AsPNG(ST_ColorMap(rast,1,'greyscale')) As grey_png,
    ST_AsPNG(ST_ColorMap(rast,1, 'pseudocolor')) As pseudo_png,
    ST_AsPNG(ST_ColorMap(rast,1, 'nfire')) As fire_png,
    ST_AsPNG(ST_ColorMap(rast,1, 'bluered')) As bluered_png,
    ST_AsPNG(ST_ColorMap(rast,1, '
100% 255   0   0
 80% 160   0   0
 50% 130   0   0
 30%  30   0   0
 20%  60   0   0
  0%   0   0   0
  nv 255 255 255
    ')) As red_png
FROM funky_shapes;
```



orig_png | *grey_png* | *pseudo_png*



fire_png | *bluered_png* | *red_png*

Voir aussi

ST_AsPNG, ST_AsRaster ST_MapAlgebra (callback function version), ST_Grayscale ST_NumBands, ST_Reclass, ST_SetBandNoDat
ST_Union

11.12.3 ST_Grayscale

ST_Grayscale — Crée un nouveau raster à 1 bande 8BUI à partir du raster source et des bandes spécifiées représentant les composantes rouge, vert et bleu

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

Crée un raster à 1 bande 8BUI à partir de trois bandes d'entrée (provenant d'un ou plusieurs rasters). Toute bande d'entrée dont le type de pixel n'est pas 8BUI sera reclassée avec ST_Reclass.

Note
Cette fonction est différente de ST_ColorMap avec le mot-clé `grayscale`, car ST_ColorMap n'opère que sur une seule bande alors que cette fonction s'attend à trois bandes pour RGB. Cette fonction applique l'équation suivante pour convertir les composantes RGB en niveaux de gris : 0,2989 * ROUGE + 0,5870 * VERT + 0,1140 * BLEU

Disponibilité : 2.5.0

Exemples : Variante 1

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.enable_outdb_rasters = True;

WITH apple AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(350, 246, 0, 0, 1, -1, 0, 0, 0),
        '/tmp/apple.png'::text,
        NULL::int[]
    ) AS rast
)
SELECT
    ST_AsPNG(rast) AS original_png,
    ST_AsPNG(ST_Grayscale(rast)) AS grayscale_png
FROM apple;
```

original_png

grayscale_png

Exemples : Variante 2

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.enable_outdb_rasters = True;

WITH apple AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(350, 246, 0, 0, 1, -1, 0, 0, 0),
        '/tmp/apple.png'::text,
        NULL::int[]
    ) AS rast
)
SELECT
    ST_AsPNG(rast) AS original_png,
    ST_AsPNG(ST_Grayscale(
        ARRAY[
            ROW(rast, 1)::rastbandarg, -- red
            ROW(rast, 2)::rastbandarg, -- green
            ROW(rast, 3)::rastbandarg, -- blue
        ]::rastbandarg[]
    )) AS grayscale_png
FROM apple;
```

Voir aussi

ST_AsPNG, ST_Reclass, ST_ColorMap

11.12.4 ST_Intersection

ST_Intersection — Retourne un raster ou un ensemble de paires (géométrie, valeur de pixel) représentant la partie partagée de deux rasters ou l'intersection géométrique d'une vectorisation du raster et d'une géométrie.

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

Retourne un raster ou un ensemble de paires (géométrie, valeur de pixel) représentant la partie partagée de deux rasters ou l'intersection géométrique d'une vectorisation du raster et d'une géométrie.

Les trois premières variantes, qui retournent un ensemble de géométries, fonctionnent dans l'espace vectoriel. Le raster est d'abord vectorisé (via ST_DumpAsPolygons) en un ensemble de lignes geomval, et ces lignes sont ensuite intersectées avec la géométrie via la fonction PostGIS ST_Intersection (geometry, geometry). Les géométries intersectant uniquement une zone de valeur nodata d'un raster renvoient une géométrie vide. Elles sont normalement exclues des résultats par l'utilisation correcte de ST_Intersects dans la clause WHERE.

Vous pouvez accéder aux parties géométrie et valeur de l'ensemble résultant de geomval en les entourant de parenthèses et en ajoutant '.geom' ou '.val' à la fin de l'expression. par exemple (ST_Intersection(rast, geom)).geom

Les autres variantes, qui retournent un raster, fonctionnent dans l'espace raster. Elles utilisent la variante de ST_MapAlgebraExpr utilisant deux rasters pour réaliser l'intersection.

L'étendue du raster résultant correspond à l'intersection géométrique des deux étendues des rasters. Le raster résultant inclut les bandes 'BAND1', 'BAND2' ou 'BOTH', en fonction du paramètre `returnband`. Les zones de valeurs nodata présentes dans n'importe quelle bande se traduisent par des zones de valeurs nodata dans toutes les bandes du résultat. En d'autres termes, tout pixel croisant un pixel à valeur nodata devient un pixel à valeur nodata dans le résultat.

Les rasters résultant de ST_Intersection doivent avoir une valeur nodata assignée pour les zones non intersectées. Vous pouvez définir ou remplacer la valeur nodata pour toute bande résultante en fournissant un tableau `nodataval[]` d'une ou deux valeurs nodata selon que vous demandez les bandes 'BAND1', 'BAND2' ou 'BOTH'. La première valeur du tableau remplace la valeur nodata dans la première bande et la deuxième valeur remplace la valeur nodata dans la deuxième bande. Si une bande d'entrée n'a pas de valeur nodata définie et qu'aucune n'est fournie sous forme de tableau, une valeur est choisie via la fonction ST_MinPossibleValue. Toutes les variantes acceptant un tableau de valeurs nodata peuvent également accepter une valeur unique qui sera assignée à chaque bande demandée.

Dans toutes les variantes, si aucune bande n'est spécifiée, la bande 1 est utilisée. Si vous avez besoin d'une intersection entre un raster et une géométrie qui retourne un raster, utilisez ST_Clip.

Note
Pour mieux contrôler l'étendue résultante, ou ce qu'il faut retourner lorsqu'une valeur nodata est rencontrée, utilisez la variante à deux rasters de ST_MapAlgebraExpr.

Note
Pour calculer l'intersection d'une bande raster avec une géométrie dans l'espace raster, utilisez ST_Clip. ST_Clip travaille sur des bandes rasters multiples et ne retourne pas de bande correspondant à la géométrie rasterisée.

Note
ST_Intersection devrait être utilisée conjointement avec ST_Intersects et un index sur la colonne raster et/ou la colonne géométrique.

Amélioration : 2.0.0 - Ajout de l'intersection dans l'espace raster. Dans les versions antérieures à la version 2.0.0, seules les intersections réalisées dans l'espace vectoriel étaient prises en charge.

Exemples : Géométrie, Raster -- avec pour résultat des geomval

```
SELECT
    foo.rid,
    foo.gid,
    ST_AsText((foo.geomval).geom) As geomwkt,
    (foo.geomval).val
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, ST_GeomFromText('LINESTRING(3427927.85 5793243.75,3427927.8  ↩
                5793243.75,3427927.8 5793243.8)')),
            (3, ST_GeomFromText('LINESTRING(1 2, 3 4)'))
    ) As g(gid,geom)
    WHERE A.rid = 2
) As foo;

 rid | gid |      geomwkt                                                  | val
-----+-----+-----------------------------------------------------------------------------------------

   2 |   1 | POINT(3427928 5793243.85)                                     | 249
   2 |   1 | POINT(3427928 5793243.85)                                     | 253
   2 |   2 | POINT(3427927.85 5793243.75)                                  | 254
   2 |   2 | POINT(3427927.8 5793243.8)                                    | 251
   2 |   2 | POINT(3427927.8 5793243.8)                                    | 253
   2 |   2 | LINESTRING(3427927.8 5793243.75,3427927.8 5793243.8)    | 252
   2 |   2 | MULTILINESTRING((3427927.8 5793243.8,3427927.8 5793243.75),...) | 250
   2 |   3 | GEOMETRYCOLLECTION EMPTY
```

Voir aussi

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText

11.12.5 ST_MapAlgebra (callback function version)

ST_MapAlgebra (callback function version) — Version avec fonction de rappel - Retourne un raster à une bande à partir d'un ou plusieurs rasters d'entrée, d'index de bandes et d'une fonction de rappel spécifiée par l'utilisateur.

Synopsis

raster **ST_MapAlgebra**(rastbandarg[] rastbandargset, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=INTERSECTIO
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 user-
args=NULL);
raster **ST_MapAlgebra**(raster rast, integer nband, regprocedure callbackfunc, float8[] mask, boolean weighted, text pixel-
type=NULL, text extenttype=INTERSECTION, raster customextent=NULL, text[] VARIADIC userargs=NULL);

Description

Retourne un raster à une bande à partir d'un ou plusieurs rasters d'entrée, d'index de bandes et d'une fonction de rappel spécifiée par l'utilisateur.

rast,rast1,rast2, rastbandargset Rasters sur lesquels le traitement d'algèbre cartographique est évalué.

`rastbandargset` permet d'utiliser une opération d'algèbre cartographique sur plusieurs rasters et/ou plusieurs bandes. Voir l'exemple Variante 1.

nband, nband1, nband2 Numéros de bande du raster à évaluer. nband peut être un entier ou un tableau d'entiers désignant les bandes. nband1 est la bande sur rast1 et nband2 est la bande sur rast2 pour le cas de 2 rasters/2 bandes.

callbackfunc Le paramètre `callbackfunc` doit être le nom et la signature d'une fonction SQL ou PL/pgSQL, transformée en une regprocedure. Un exemple de fonction PL/pgSQL est le suivant :

```
CREATE OR REPLACE FUNCTION sample_callbackfunc(value double precision[][][], position ↩
    integer[][], VARIADIC userargs text[])
    RETURNS double precision
    AS $$
    BEGIN
        RETURN 0;
    END;
    $$ LANGUAGE 'plpgsql' IMMUTABLE;
```

Le `callbackfunc` doit avoir trois arguments : un tableau de double précision à 3 dimensions, un tableau d'entiers à 2 dimensions et un tableau de texte variadique à 1 dimension. Le premier argument `valeur` est l'ensemble des valeurs (en double précision) de toutes les matrices d'entrée. Les trois dimensions (où les index sont basés sur 1) sont : trame #, ligne y, colonne x. Le deuxième argument `position` est l'ensemble des positions des pixels de la trame de sortie et des trames d'entrée. La dimension extérieure (où les indices sont basés sur 0) est la trame #. La position à l'indice 0 de la dimension extérieure est la position en pixels de la trame de sortie. Pour chaque dimension extérieure, il y a deux éléments dans la dimension intérieure pour X et Y. Le troisième argument `userargs` sert à transmettre les arguments spécifiés par l'utilisateur.

Passer un argument de type regprocedure à une fonction SQL nécessite de passer la signature complète de la fonction, puis de la convertir en un type regprocedure. Pour passer la fonction PL/pgSQL ci-dessus en tant qu'argument, le code SQL de l'argument est le suivant :

```
'sample_callbackfunc(double precision[], integer[], text[])'::regprocedure
```

Notez que l'argument contient le nom de la fonction, les types des arguments de la fonction, des guillemets autour du nom et des types d'arguments, ainsi qu'un cast vers un regprocedure.

mask Un tableau à n dimensions (matrice) de nombres utilisés pour filtrer les cellules transmises à la fonction de rappel de l'algèbre cartographique. 0 signifie que la valeur d'une cellule voisine doit être traitée comme une absence de données et 1 signifie que la valeur doit être traitée comme une donnée. Si weight est définit à true, les valeurs sont utilisées comme multiplicateurs pour multiplier la valeur du pixel par cette valeur dans la position de voisinage.

weighted booléen (true/false) indiquant si la valeur d'un masque doit être pondérée (multipliée par la valeur originale) ou non (ne s'applique qu'aux variantes qui prennent un masque).

pixeltype Si `pixeltype` est spécifié, l'unique bande du nouveau raster sera de ce type de pixel. Si pixeltype est NULL ou omis, la nouvelle bande du raster aura le même pixeltype que la bande spécifiée du premier raster (pour les types d'étendue : INTERSECTION, UNION, FIRST, CUSTOM) ou que la bande spécifiée du raster approprié (pour les types d'étendue : SECOND, LAST). En cas de doute, spécifiez toujours `pixeltype`.

Le type de pixel résultant du raster de sortie doit correspondre à l'un des types énumérés dans ST_BandPixelType, ou être omis ou défini à NULL.

extenttype Les valeurs possibles sont INTERSECTION (par défaut), UNION, FIRST (premier, par défaut pour les variantes à un seul raster), SECOND, LAST (dernier), CUSTOM (personnalisé).

customextent Si `extentype` est CUSTOM, un raster doit être spécifié par le paramètre `customextent`. Voir l'exemple 4 de la variante 1.

distancex La distance en pixels de la cellule de référence dans la direction x. La largeur de la matrice résultante sera donc `2*distancex + 1`. Si elle n'est pas spécifiée, seule la cellule de référence est prise en compte (voisinage de 0).

distancey La distance en pixels de la cellule de référence dans la direction y. La hauteur de la matrice résultante sera `2*distancey + 1` Si elle n'est pas spécifiée, seule la cellule de référence est prise en compte (voisinage de 0).

userargs Le troisième paramètre de la fonction `callbackfunc` est un tableau variadic text . Tous les arguments textuels supplémentaires sont transmis à `callbackfunc`, et sont contenus dans l'argument `userargs`.

Note
Pour plus d'informations sur le mot-clé VARIADIC, veuillez vous référer à la documentation PostgreSQL et à la section "SQL Functions with Variable Numbers of Arguments" de Query Language (SQL) Functions.

Note
Le paramètre text[] de la fonction `callbackfunc` est obligatoire, que vous choisissiez ou non de transmettre des arguments à la fonction de rappel.

La variante 1 accepte un tableau de `rastbandarg` permettant l'utilisation d'une opération d'algèbre cartographique sur de nombreux masters et/ou de nombreuses bandes. Voir l'exemple de la variante 1.

Les variantes 2 et 3 agissent sur une ou plusieurs bandes d'un même raster. Voir l'exemple des variantes 2 et 3.

La variante 4 utilise deux rasters avec une bande par raster. Voir l'exemple de la variante 4.

Disponibilité : 2.2.0 : Possibilité d'ajouter un masque

Disponibilité : 2.1.0

Exemples : Variante 1

Un raster, une bande

```
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
```

Un raster, plusieurs bandes

```
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
```

Plusieurs rasters, plusieurs bandes

```
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
```

Exemple complet avec les tuiles d'une couverture avec voisinage. Cette requête ne fonctionne qu'avec PostgreSQL 9.1 ou plus.

```
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
```

Exemple identique au précédent pour les tuiles d'une couverture avec voisinage, mais fonctionne avec PostgreSQL 9.0.

```
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
)
WITH 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[],
            'raster_nmapalgebra_test(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;
```

Exemples : Variantes 2 et 3

Un raster, plusieurs bandes

```
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
```

Un raster, une bande

```
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
```

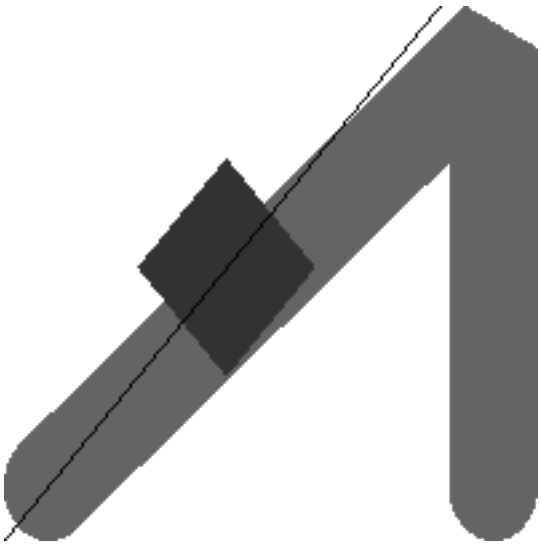
Exemples : Variante 4

Deux rasters, deux bandes

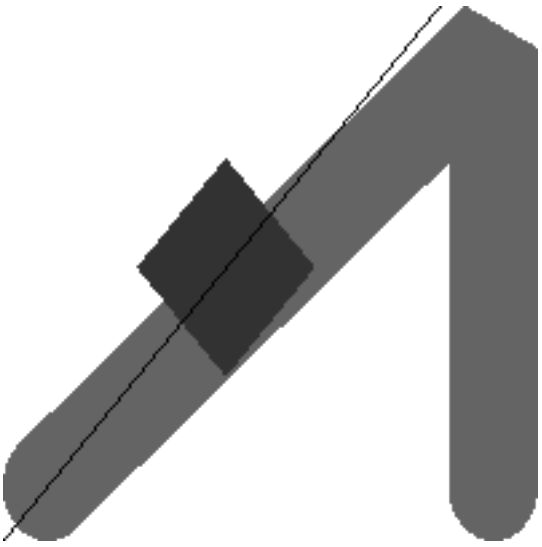
```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
         0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast UNION  ↩
        ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 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
```

Exemples : Utilisation de masques

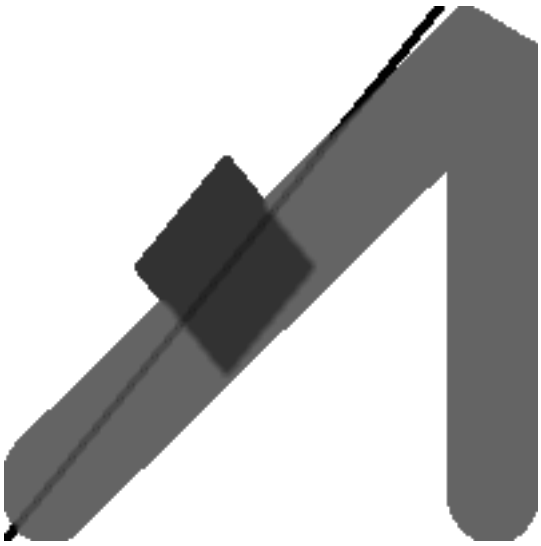
```
WITH foo AS (SELECT
   ST_SetBandNoDataValue(
ST_SetValue(ST_SetValue(ST_AsRaster(
        ST_Buffer(
            ST_GeomFromText('LINESTRING(50 50,100 90,100 50)'), 5,'join=bevel'),
            200,200,ARRAY['8BUI'], ARRAY[100], ARRAY[0]), ST_Buffer('POINT(70 70)':: ↩
                geometry,10,'quad_segs=1') ,50),
  'LINESTRING(20 20, 100 100, 150 98)'::geometry,1),0)  AS rast )
SELECT 'original' AS title, rast
FROM foo
UNION ALL
SELECT 'no mask mean value' AS title, ST_MapAlgebra(rast,1,'ST_mean4ma(double precision[],  ↩
    int[], text[])'::regprocedure) AS rast
FROM foo
UNION ALL
SELECT 'mask only consider neighbors, exclude center' 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 'mask weighted only consider neighbors, exclude center multi other pixel values by  ↩
    2' 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;
```



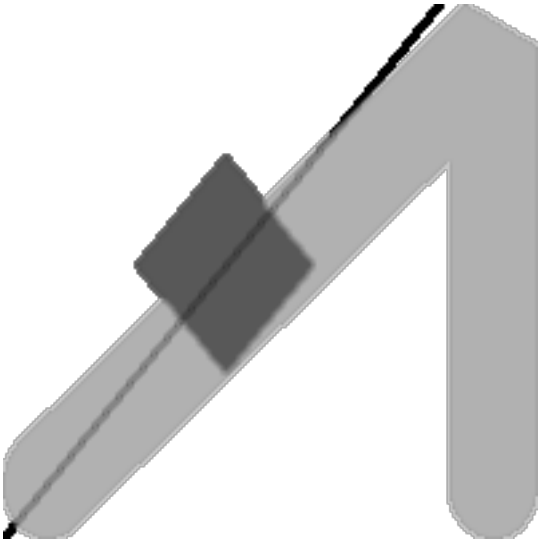
original



valeur moyenne, sans masque (ce qui revient à avoir que des 1 dans la matrice du masque)



masque uniquement sur les voisins, en excluant le centre



masque pondéré, uniquement sur les voisins en excluant le centre, multiplier les autres pixels par 2

Voir aussi

rastbandarg, ST_Union, ST_MapAlgebra (expression version)

11.12.6 ST_MapAlgebra (expression version)

ST_MapAlgebra (expression version) — Version avec expression - Retourne un raster à une bande à partir d'un ou deux rasters d'entrée, d'index de bandes et d'une ou plusieurs expressions SQL spécifiées par l'utilisateur.

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

Version avec expression - Retourne un raster à une bande à partir d'un ou deux rasters d'entrée, d'index de bandes et d'une ou plusieurs expressions SQL spécifiées par l'utilisateur.

Disponibilité : 2.1.0

Description : Variantes 1 et 2 (un raster)

Crée un nouveau raster à une bande formé en appliquant une opération algébrique PostgreSQL valide définie par `expression` sur le raster d'entrée (`rast`). Si `nband` n'est pas fourni, la bande 1 est utilisée. Le nouveau raster aura la même géoréférence, la même largeur et la même hauteur que le raster original mais n'aura qu'une seule bande.

Si `pixeltype` est fourni, le nouveau raster aura une bande de ce type de pixel. Si pixeltype est NULL, la nouvelle bande raster aura le même type de pixel que la bande d'entrée de `rast`.

- Mots clés possibles pour `expression`

  1. `[rast]` - Valeur du pixel d'intérêt
  2. `[rast.val]` - Valeur du pixel d'intérêt
  3. `[rast.x]` - colonne du pixel d'intérêt (démarrant à 1)
  4. `[rast.y]` - ligne du pixel d'intérêt (démarrant à 1)

Description : Variantes 3 et 4 (deux rasters)

Crée un nouveau raster à une bande formé en appliquant une opération algébrique PostgreSQL valide aux deux bandes définies par `expression` sur les deux bandes d'entrée `rast1` (et optionnellement `rast2`). Si aucune bande `band1`, `band2` n'est spécifiée, la bande 1 est utilisée. Le raster résultant sera aligné (échelle, obliquité et coins) sur la grille définie par le premier raster. Le raster résultant aura l'étendue définie par le paramètre `extenttype`.

`expression` Une expression algébrique PostgreSQL impliquant les deux rasters et les fonctions/opérateurs définis par PostgreSQL qui définiront la valeur du pixel lorsque les pixels intersectent, par exemple (([rast1] + [rast2])/2.0)::integer

`pixeltype` Le type de pixel résultant du raster de sortie. Doit être l'un de ceux listés dans ST_BandPixelType, omis ou NULL. S'il n'est pas fourni ou NULL, le type de pixel du premier raster sera utilisé par défaut.

`extenttype` Contrôle l'étendue du raster de sortie

  1. `INTERSECTION` - L'étendue du nouveau raster est l'intersection des deux rasters. Il s'agit de la valeur par défaut.
  2. `UNION` - L'étendue du nouveau raster est l'union des deux rasters.
  3. `FIRST` - L'étendue du nouveau raster est la même que celle du premier raster.
  4. `SECOND` - L'étendue du nouveau raster est la même que celle du second raster.

`nodata1expr` Une expression algébrique impliquant uniquement `rast2`, ou une constante qui définit la valeur à retourner lorsque les pixels de `rast1` sont des valeurs nodata et que les pixels correspondants dans rast2 ont des valeurs.

`nodata2expr` Une expression algébrique impliquant uniquement `rast1`, ou une constante qui définit la valeur à retourner lorsque les pixels de `rast2` sont des valeurs nodata et que les pixels correspondants dans rast1 ont des valeurs.

`nodatanodataval` Constante numérique à retourner lorsque les pixels de rast1 et de rast2 sont tous deux des valeurs nodata.

- Mots clés possibles dans `expression`, `nodata1expr` et `nodata2expr`

  1. `[rast1]` - Valeur du pixel d'intérêt sur `rast1`
  2. `[rast1.val]` - Valeur du pixel d'intérêt de `rast1`
  3. `[rast1.x]` - Colonne du pixel d'intérêt de `rast1` (démarrant à 1)
  4. `[rast1.y]` - Ligne du pixel d'intérêt de `rast1` (démarrant à 1)
  5. `[rast2]` - Valeur du pixel d'intérêt sur `rast2`
  6. `[rast2.val]` - Valeur du pixel d'intérêt de `rast2`
  7. `[rast2.x]` - Colonne du pixel d'intérêt de `rast2` (démarrant à 1)
  8. `[rast2.y]` - Ligne du pixel d'intérêt de `rast2` (démarrant à 1)

Exemples : Variantes 1 et 2

```
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;
```

Exemples : Variantes 3 et 4

```
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;
```

Voir aussi

rastbandarg, ST_Union, ST_MapAlgebra (callback function version)

11.12.7 ST_MapAlgebraExpr

ST_MapAlgebraExpr — Version avec raster à 1 bande : Crée un nouveau raster à une bande formé par l'application d'une opération algébrique PostgreSQL valide sur la bande d'entrée du raster et du type de pixel fourni. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

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

Warning
ST_MapAlgebraExpr est obsolète depuis 2.1.0. Utilisez plutôt ST_MapAlgebra (expression version).

Crée un nouveau raster à une bande formé par l'application d'une opération algébrique PostgreSQL valide définie par `expression` sur le raster d'entrée (`rast`). Si aucune `band` n'est spécifiée, la bande 1 est utilisée. Le nouveau raster aura la même géoréférence, la même largeur et la même hauteur que le raster original mais n'aura qu'une seule bande.

Si `pixeltype` est fourni, le nouveau raster aura une bande de ce type de pixel. Si pixeltype est NULL, la nouvelle bande raster aura le même type de pixel que la bande d'entrée de `rast`.

Dans l'expression, vous pouvez utiliser le mot clé `[rast]` pour faire référence à la valeur en pixels de la bande originale, `[rast.x]` pour faire référence à l'indice de colonne de pixels (basé sur 1), `[rast.y]` pour faire référence à l'indice de ligne de pixels (basé sur 1).

Disponibilité : 2.0.0

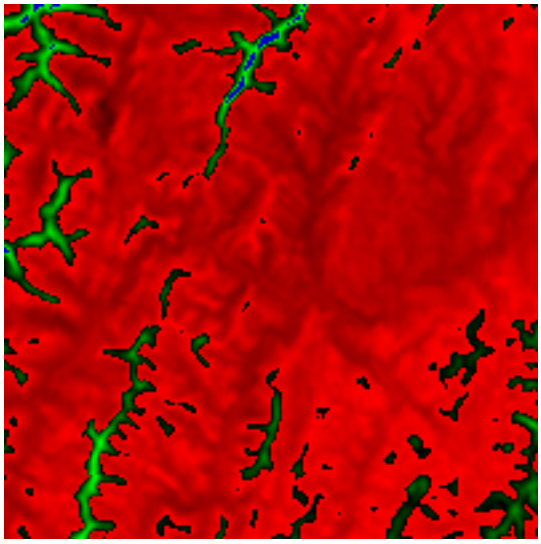
Exemples

Crée une nouveau raster à 1 bande qui est une fonction modulo 2 de la bande du raster original.

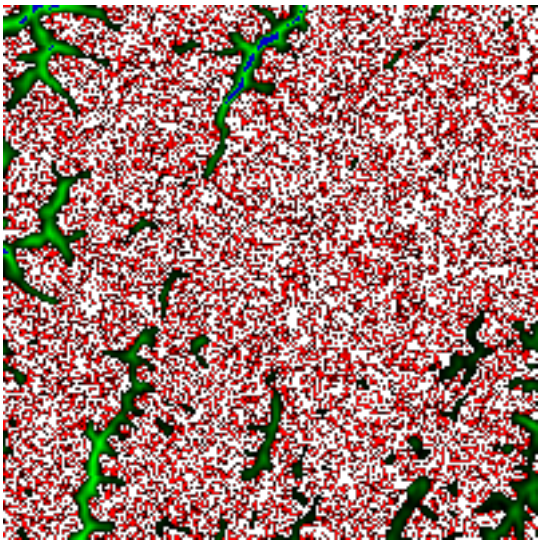
```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
UPDATE dummy_rast SET map_rast = ST_MapAlgebraExpr(rast,NULL,'mod([rast]::numeric,2)')  ↩
    WHERE rid = 2;

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;

 origval | mapval
---------+--------
     253 |      1
     254 |      0
     253 |      1
     253 |      1
     254 |      0
     254 |      0
     250 |      0
     254 |      0
     254 |      0
```

Crée un nouveau raster à 1 bande de type de pixel 2BUI à partir d'un raster original reclassifié et avec la valeur nodata à 0.

```
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;

 origval | mapval
---------+--------
     249 |      1
     250 |      1
     251 |
     252 |      2
     253 |      3
     254 |      3

SELECT
    ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast
WHERE rid = 2;

 b1pixtyp
----------
 2BUI
```



original (colonne rast_view)



rast_view_ma

Crée un nouveau raster à 3 bandes de même type de pixel que le raster original à 3 bandes, la première bande étant modifiée par l'algèbre cartographique et les 2 bandes restantes n'étant pas modifiées.

```
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;
```

Voir aussi

ST_MapAlgebraExpr, ST_MapAlgebraFct, ST_BandPixelType, ST_GeoReference, ST_Value

11.12.8 ST_MapAlgebraExpr

ST_MapAlgebraExpr — Version avec 2 bandes : Crée un nouveau raster à une bande formé en appliquant une opération algébrique PostgreSQL valide sur les deux bandes du raster d'entrée et du type de pixel fourni. La bande 1 de chaque raster est utilisée si aucun numéro de bande n'est spécifié. Le raster résultant sera aligné (échelle, obliquité et coins) sur la grille définie par le premier raster et aura son étendue définie par le paramètre "extenttype". Les valeurs de "extenttype" peuvent être : INTERSECTION, UNION, FIRST, SECOND.

Synopsis

raster **ST_MapAlgebraExpr**(raster rast1, raster rast2, text expression, text pixeltype=same_as_rast1_band, text extenttype=INTERSEC
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

Warning
ST_MapAlgebraExpr est obsolète depuis 2.1.0. Utilisez plutôt ST_MapAlgebra (expression version).

Crée un nouveau raster à une bande formé en appliquant une opération algébrique PostgreSQL valide aux deux bandes définies par `expression` sur les deux bandes d'entrée `rast1` (et optionnellement `rast2`). Si aucune bande `band1`, `band2` n'est spécifiée, la bande 1 est utilisée. Le raster résultant sera aligné (échelle, obliquité et coins) sur la grille définie par le premier raster. Le raster résultant aura l'étendue définie par le paramètre `extenttype`.

`expression` Une expression algébrique PostgreSQL impliquant les deux rasters et les fonctions/opérateurs définis par PostgreSQL qui définiront la valeur du pixel lorsque les pixels intersectent, par exemple (([rast1] + [rast2])/2.0)::integer

`pixeltype` Le type de pixel résultant du raster de sortie. Doit être l'un de ceux listés dans ST_BandPixelType, omis ou NULL. S'il n'est pas fourni ou NULL, le type de pixel du premier raster sera utilisé par défaut.

`extenttype` Contrôle l'étendue du raster de sortie

1. `INTERSECTION` - L'étendue du nouveau raster est l'intersection des deux rasters. Il s'agit de la valeur par défaut.
2. `UNION` - L'étendue du nouveau raster est l'union des deux rasters.
3. `FIRST` - L'étendue du nouveau raster est la même que celle du premier raster.
4. `SECOND` - L'étendue du nouveau raster est la même que celle du second raster.

`nodata1expr` Une expression algébrique impliquant uniquement `rast2`, ou une constante qui définit la valeur à retourner lorsque les pixels de `rast1` sont des valeurs nodata et que les pixels correspondants dans rast2 ont des valeurs.

`nodata2expr` Une expression algébrique impliquant uniquement `rast1`, ou une constante qui définit la valeur à retourner lorsque les pixels de `rast2` sont des valeurs nodata et que les pixels correspondants dans rast1 ont des valeurs.

`nodatanodataval` Constante numérique à retourner lorsque les pixels de rast1 et de rast2 sont tous deux des valeurs nodata.

Si le `pixeltype` est fourni, le nouveau raster aura une bande de ce type de pixel. Si le type de pixel est NULL ou si aucun type de pixel n'est spécifié, la nouvelle bande raster aura le même type de pixel que la bande d'entrée de `rast1`.

Utilisez le mot clé `[rast1.val] [rast2.val]` pour désigner la valeur en pixels des bandes raster d'origine et `[rast1.x]`, `[rast1.y]` etc. pour désigner la position des pixels dans les colonnes et les lignes.

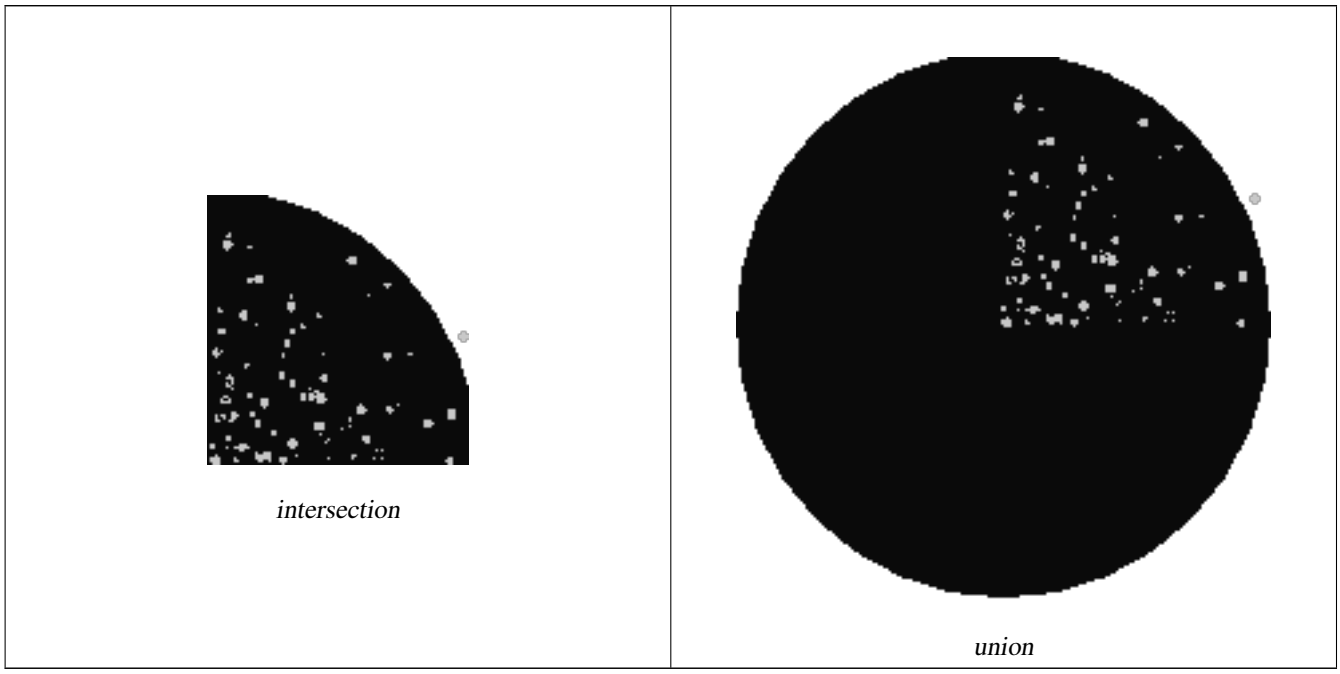
Disponibilité : 2.0.0

Exemple : Intersection et Union avec 2 bandes

Crée une nouveau raster à 1 bande qui est une fonction modulo 2 de la bande du raster original.

```
--Create a cool set of rasters --
DROP TABLE IF EXISTS fun_shapes;
CREATE TABLE fun_shapes(rid serial PRIMARY KEY, fun_name text, rast raster);

-- Insert some cool shapes around Boston in Massachusetts state plane meters --
INSERT INTO fun_shapes(fun_name, rast)
VALUES ('ref', ST_AsRaster(ST_MakeEnvelope(235229, 899970, 237229, 901930,26986),200,200,'8 ↩
    BUI',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;

--map them -
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
```



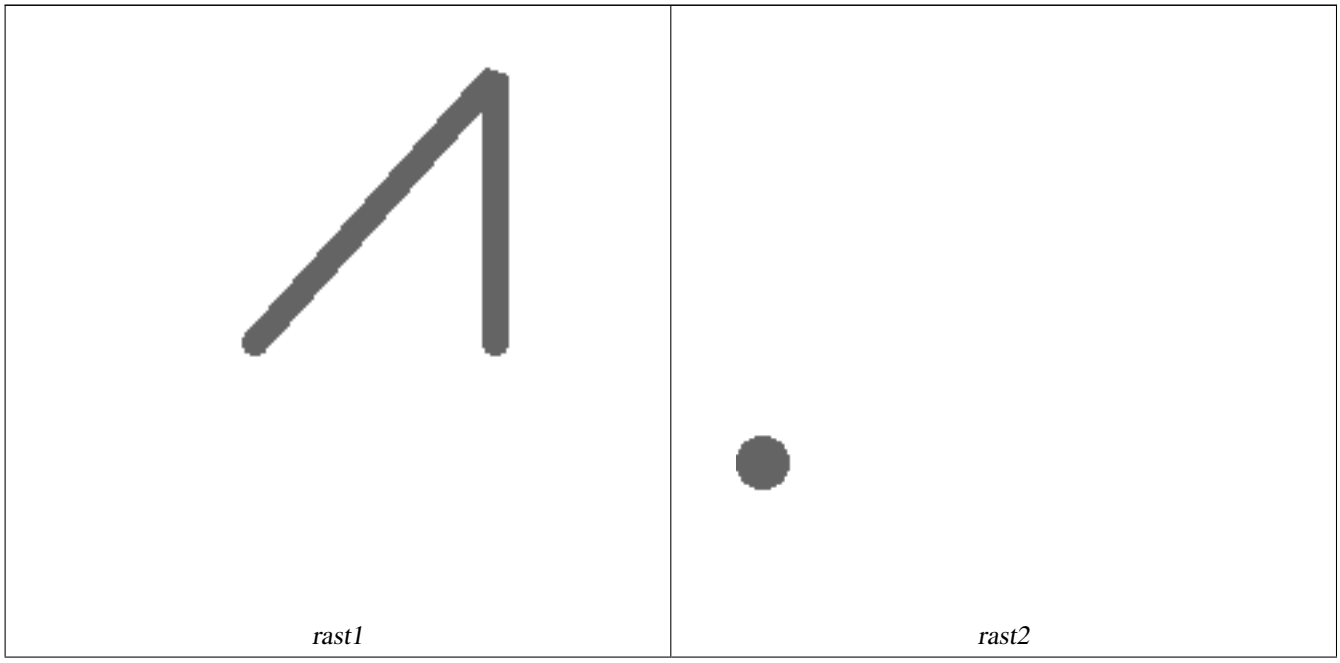
intersection

union

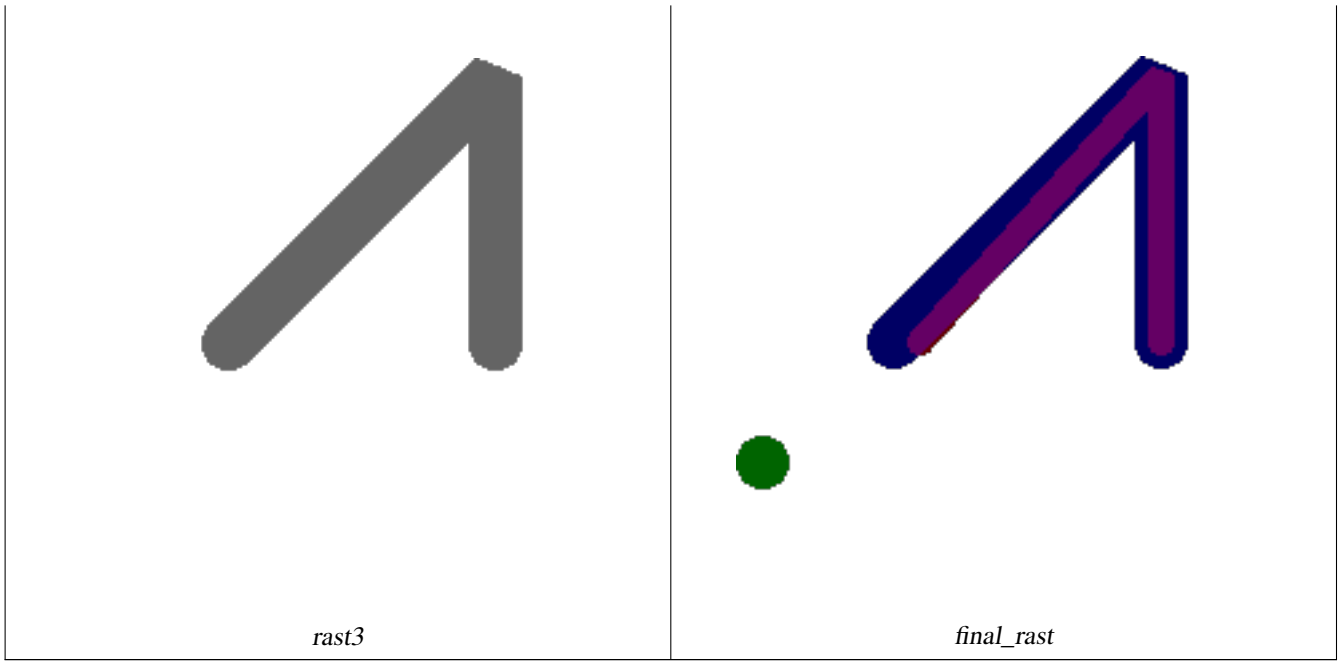
Exemple : Superposition de rasters sur un canevas sous forme de bandes distinctes

```
-- we use ST_AsPNG to render the image so all single band ones look grey --
WITH mygeoms
    AS ( SELECT 2 As bnum, ST_Buffer(ST_Point(1,5),10) As geom
            UNION ALL
```

```
          SELECT 3 AS bnum,
              ST_Buffer(ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'), 10,'join= ↩
                  bevel') As geom
          UNION ALL
          SELECT 1 As bnum,
              ST_Buffer(ST_GeomFromText('LINESTRING(60 50,150 150,150 50)'), 5,'join= ↩
                  bevel') As geom
          ),
-- define our canvas to be 1 to 1 pixel to geometry
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,
                 Max(ST_SRID(geom)) As srid
                 from mygeoms
                 ) As foo
          ),
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]') As rast
            FROM mygeoms AS m CROSS JOIN canvas
            ORDER BY m.bnum) As rasts
            )
       SELECT rasts[1] As rast1 , rasts[2] As rast2, rasts[3] As rast3, ST_AddBand(
                 ST_AddBand(rasts[1],rasts[2]), rasts[3]) As final_rast
         FROM rbands;
```



rast1 *rast2*



rast3 | *final_rast*

Exemple : Superposition d'une délimitation de 2 mètres de certaines parcelles sur une image aérienne

```
-- Create new 3 band raster composed of first 2 clipped bands, and overlay of 3rd band with ↩
     our geometry
-- This query took 3.6 seconds on PostGIS windows 64-bit install
WITH pr AS
-- Note the order of operation: we clip all the rasters to dimensions of our region
(SELECT ST_Clip(rast,ST_Expand(geom,50) ) As rast, g.geom
    FROM aerials.o_2_boston AS r INNER JOIN
-- union our parcels of interest so they form a single geometry we can later intersect with
        (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))
),
-- we then union the raster shards together
-- ST_Union on raster is kinda of slow but much faster the smaller you can get the rasters
-- therefore we want to clip first and then union
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)
-- return our final raster which is the unioned shard with
-- with the overlay of our parcel boundaries
-- add first 2 bands, then mapalgebra of 3rd band + geometry
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;
```

Les lignes bleues représentent les limites des parcelles sélectionnées

Voir aussi

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 — Version à 1 bande - Crée un nouveau raster à une bande formé par l'application d'une fonction PostgreSQL valide sur la bande d'entrée du raster et sur le type de pixel fourni. La bande 1 est utilisée si aucune bande n'est spécifiée.

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

Warning
ST_MapAlgebraFct est obsolète depuis 2.1.0. Utilisez plutôt ST_MapAlgebra (callback function version).

Crée un nouveau raster à une bande formé en appliquant une fonction PostgreSQL valide spécifiée par `onerasteruserfunc` sur le raster d'entrée (`rast`). Si aucune bande `band` n'est spécifiée, la bande 1 est utilisée. Le nouveau raster aura la même géoréférence, la même largeur et la même hauteur que le raster original mais n'aura qu'une seule bande.

Si `pixeltype` est fourni, le nouveau raster aura une bande de ce type de pixel. Si pixeltype est NULL, la nouvelle bande raster aura le même type de pixel que la bande d'entrée de `rast`.

Le paramètre `onerasteruserfunc` doit être le nom et la signature d'une fonction SQL ou PL/pgSQL, transformée en regprocedure. Un exemple de fonction PL/pgSQL très simple et assez inutile est le suivant :

```
CREATE OR REPLACE FUNCTION simple_function(pixel FLOAT, pos INTEGER[], VARIADIC args TEXT ↩
    [])
    RETURNS FLOAT
    AS $$ BEGIN
        RETURN 0.0;
    END; $$
    LANGUAGE 'plpgsql' IMMUTABLE;
```

La `fonction utilisateur` peut accepter deux ou trois arguments : une valeur flottante, un tableau d'entiers facultatif et un tableau de texte variable. Le premier argument est la valeur d'une cellule matricielle individuelle (quel que soit le type de données matricielles). Le deuxième argument est la position de la cellule de traitement actuelle sous la forme '{x,y}'. Le troisième argument indique que tous les paramètres restants de ST_MapAlgebraFct doivent être transmis à la `fonction utilisateur`.

Passer un argument de type regprodedure à une fonction SQL nécessite de passer la signature complète de la fonction, puis de la convertir en un type regprocedure. Pour passer la fonction PL/pgSQL ci-dessus en tant qu'argument, le code SQL de l'argument est le suivant :

```
'simple_function(float,integer[],text[])'::regprocedure
```

Notez que l'argument contient le nom de la fonction, les types des arguments de la fonction, des guillemets autour du nom et des types d'arguments, ainsi qu'un cast vers un regprocedure.

Le troisième paramètre de la fonction `userfunction` est un tableau variadic text . Tous les arguments textuels supplémentaires de l'appel à ST_MapAlgebraFct sont transmis à `userfunction`, et sont contenus dans l'argument `args`.

Note
Pour plus d'informations sur le mot-clé VARIADIC, veuillez vous référer à la documentation PostgreSQL et à la section "SQL Functions with Variable Numbers of Arguments" de Query Language (SQL) Functions.

Note
Le paramètre text[] de la fonction `userfunction` est obligatoire, que vous choisissiez ou non de transmettre des arguments à la fonction de rappel.

Disponibilité : 2.0.0

Exemples

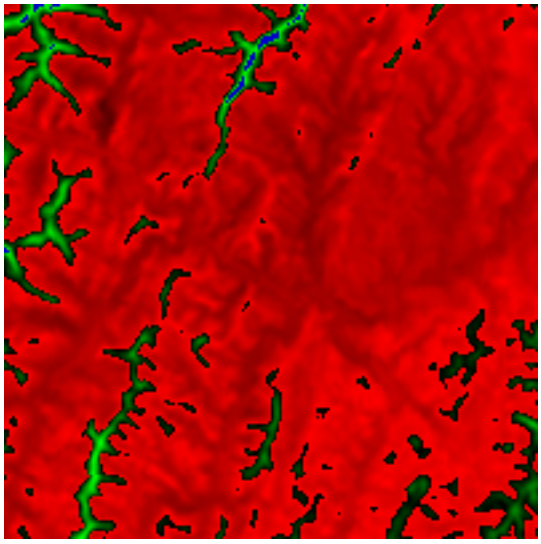
Crée une nouveau raster à 1 bande qui est une fonction modulo 2 de la bande du raster original.

```
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;

 origval | mapval
---------+--------
     253 |      1
     254 |      0
     253 |      1
     253 |      1
     254 |      0
     254 |      0
     250 |      0
     254 |      0
     254 |      0
```

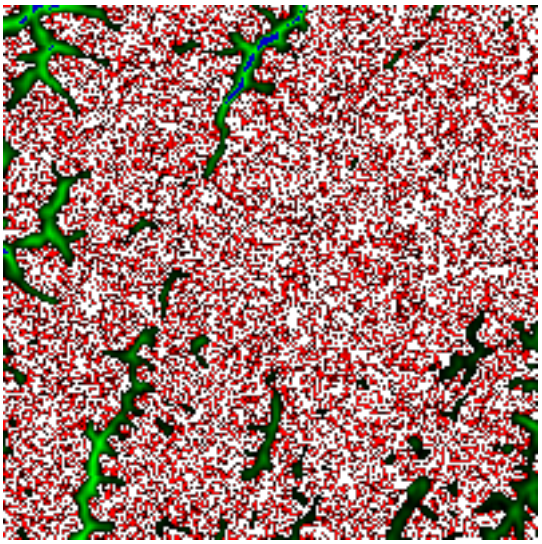
Crée un nouveau raster à 1 bande de type de pixel 2BUI à partir d'un raster reclassifié avec la valeur nodata fixée via un paramètre passé à la fonction utilisateur (0).

```
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;

 origval | mapval
---------+--------
     249 |      1
     250 |      1
     251 |
     252 |      2
     253 |      3
     254 |      3

SELECT ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast WHERE rid = 2;

 b1pixtyp
----------
 2BUI
```



original (colonne rast-view)



rast_view_ma

Crée un nouveau raster à 3 bandes de même type de pixel que le raster original à 3 bandes, la première bande étant modifiée par l'algèbre cartographique et les 2 bandes restantes n'étant pas modifiées.

```
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;
```

Voir aussi

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

11.12.10 ST_MapAlgebraFct

ST_MapAlgebraFct — Version à 2 bandes - Crée un nouveau raster à une bande formé par l'application d'une fonction PostgreSQL valide sur les 2 bandes d'entrée du raster et sur le type de pixel fourni. La bande 1 est utilisée si aucune bande n'est spécifiée. Le type d'étendue INTERSECTION est utilisé si non spécifié.

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

Warning
ST_MapAlgebraFct est obsolète depuis 2.1.0. Utilisez plutôt ST_MapAlgebra (callback function version).

Crée un nouveau raster à une bande formé par l'application d'une fonction PostgreSQL valide spécifiée par `tworastuserfunc` sur les rasters d'entrée `rast1`, `rast2`. Si aucune bande `band1` ou `band2` n'est spécifiée, la bande 1 est utilisée. Le nouveau raster aura la même géoréférence, la même largeur et la même hauteur que les rasters originaux, mais il n'aura qu'une seule bande.

Si `pixeltype` est fourni, le nouveau raster aura une bande de ce type de pixel. Si pixeltype est NULL ou omis, la nouvelle bande raster aura le même type de pixel que la bande d'entrée de `rast1`.

Le paramètre `tworastuserfunc` doit être le nom et la signature d'une fonction SQL ou PL/pgSQL, transformée en une regprocedure. Un exemple de fonction PL/pgSQL est le suivant :

```
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;
```

La `tworastuserfunc` peut accepter trois ou quatre arguments : une valeur de double précision, une valeur de double précision, un tableau d'entiers facultatif et un tableau de texte variable. Le premier argument est la valeur d'une cellule matricielle individuelle dans `rast1` (quel que soit le type de données matricielles). Le deuxième argument est la valeur d'une cellule de trame individuelle dans `rast2`. Le troisième argument est la position de la cellule de traitement actuelle sous la forme '{x,y}'. Le quatrième argument indique que tous les paramètres restants de ST_MapAlgebraFct doivent être transmis au `tworastuserfunc`.

Passer un argument de type regprodedure à une fonction SQL nécessite de passer la signature complète de la fonction, puis de la convertir en un type regprocedure. Pour passer la fonction PL/pgSQL ci-dessus en tant qu'argument, le code SQL de l'argument est le suivant :

```
'simple_function(double precision, double precision, integer[], text[])'::regprocedure
```

Notez que l'argument contient le nom de la fonction, les types des arguments de la fonction, des guillemets autour du nom et des types d'arguments, ainsi qu'un cast vers un regprocedure.

Le quatrième paramètre de la fonction `tworastuserfunc` est un tableau variadic text . Tous les arguments textuels supplémentaires de l'appel à ST_MapAlgebraFct sont transmis à `tworastuserfunc`, et sont contenus dans l'argument `userargs`.

Note
Pour plus d'informations sur le mot-clé VARIADIC, veuillez vous référer à la documentation PostgreSQL et à la section "SQL Functions with Variable Numbers of Arguments" de Query Language (SQL) Functions.

Note
Le paramètre text[] de la fonction `tworastuserfunc` est obligatoire, que vous choisissiez ou non de transmettre des arguments à la fonction de rappel.

Disponibilité : 2.0.0

Exemple : Superposition de rasters sur un canevas sous forme de bandes distinctes

```
-- define our user defined function --
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;

-- prep our test table of rasters
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 with ST_AddBand
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;
```

superposition des bandes de la carte (canevas) (R : small road, G : circle, B : big road)

Fonction définie par l'utilisateur avec arguments supplémentaires

```
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)';
```

Fonction définie par l'utilisateur avec arguments supplémentaires et plusieurs bandes issus du même raster

Voir aussi

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

11.12.11 ST_MapAlgebraFctNgb

ST_MapAlgebraFctNgb — Version à 1 bande : Algèbre cartographique Plus proche voisin en utilisant une fonction PostgreSQL définie par l'utilisateur. Retourne un raster dont les valeurs sont le résultat d'une fonction utilisateur PLPGSQL prenant un voisinage des valeurs de la bande raster d'entrée.

Synopsis

raster **ST_MapAlgebraFctNgb**(raster rast, integer band, text pixeltype, integer ngbwidth, integer ngbheight, regprocedure onerastngbuserfunc, text nodatamode, text[] VARIADIC args);

Description

Warning
ST_MapAlgebraFctNgb est obsolète depuis 2.1.0. Utilisez plutôt ST_MapAlgebra (callback function version).

(version à un raster) Retourne un raster dont les valeurs sont le résultat d'une fonction utilisateur PLPGSQL prenant un voisinage de valeurs de la bande du raster d'entrée. La fonction utilisateur prend le voisinage des valeurs des pixels sous la forme d'un tableau de nombres. Pour chaque pixel, elle renvoie le résultat de la fonction utilisateur, en remplaçant la valeur du pixel actuellement traité par le résultat de la fonction utilisateur.

rast Raster sur lequel la fonction utilisateur est évaluée.

band Numéro de bande du raster à évaluer. La valeur par défaut est 1.

pixeltype Le type de pixel résultant du raster de sortie. Doit être l'un de ceux listés dans ST_BandPixelType, ou être omis ou défini à NULL. S'il n'est pas spécifié ou s'il est défini à NULL, le type de pixel du raster `rast` sera utilisé par défaut. Les résultats sont tronqués s'ils sont plus grands que ce qui est autorisé pour le type de pixel.

ngbwidth La largeur du voisinage, en nombre de cellules.

ngbheight La hauteur du voisinage, en nombre de cellules.

onerastngbuserfunc Fonction utilisateur PLPGSQL/psql à appliquer aux pixels du voisinage de la bande du raster. Le premier élément est un tableau bidimensionnel de nombres, représentant le voisinage rectangulaire du pixel

nodatamode Définit la valeur à transmettre à la fonction pour un pixel de voisinage qui est nodata ou NULL

'ignore' : toutes les valeurs NODATA rencontrées dans le voisinage sont ignorées par le calcul -- ce drapeau doit être envoyé à la fonction de rappel de l'utilisateur, et c'est la fonction de l'utilisateur qui décide comment l'ignorer.

'NULL' : toute valeur NODATA rencontrée dans le voisinage aura pour conséquence que le pixel résultant sera NULL -- la fonction de rappel de l'utilisateur est ignorée dans ce cas.

'value' : toutes les valeurs NODATA rencontrées dans le voisinage sont remplacées par le pixel de référence (celui qui se trouve au centre du voisinage). Notez que si cette valeur est NODATA, le comportement est le même que 'NULL' (pour le voisinage concerné)

args Arguments à transmettre à la fonction utilisateur.

Disponibilité : 2.0.0

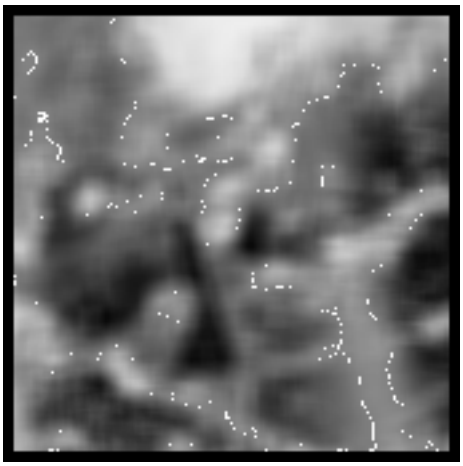
Exemples

Les exemples utilisent le raster katrina chargé comme une seule tuile, tel que décrit dans http://trac.osgeo.org/gdal/wiki/frmts_wtkraster.h et ensuite traitée dans les exemples ST_Rescale

```
--
-- A simple 'callback' user function that averages up all the values in a neighborhood.
--
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;
```

```
-- now we apply to our raster averaging pixels within 2 pixels of each other in X and Y  ↩
    direction --
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;
```

Première bande du raster



nouveau raster après calcul de la moyenne des pixels situés à moins de 4x4 pixels les uns des autres

Voir aussi

ST_MapAlgebraFct, ST_MapAlgebraExpr, ST_Rescale

11.12.12 ST_Reclass

ST_Reclass — Crée un nouveau raster composé de types de bandes reclassifiés par rapport à l'original. La bande n est la bande à modifier. Si nband n'est pas spécifié, la bande 1 est utilisée. Toutes les autres bandes sont retournées inchangées. Cas d'utilisation : convertir une bande 16BUI en 8BUI et ainsi de suite pour un rendu plus simple en tant que formats visualisables.

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

Crée un nouveau raster formé par l'application d'une opération algébrique PostgreSQL valide définie par `reclassexpr` sur le raster d'entrée (`rast`). Si aucune bande `band` n'est spécifiée, la bande 1 est utilisée. Le nouveau raster aura la même géoréférence, la même largeur et la même hauteur que le raster original. Les bandes non désignées seront retournées inchangées. Voir reclassarg pour la description des expressions de reclassification valides.

Les bandes du nouveau raster auront le type de pixel `pixeltype`. Si `reclassargset` est spécifié, chaque reclassarg définit le comportement de chaque bande générée.

Disponibilité : 2.0.0

Exemples de base

Crée un nouveau raster à partir de l'original où la bande 2 est convertie de 8BUI à 4BUI et où toutes les valeurs entre 101 et 254 sont fixées à nodata.

```
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;

 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
```

Exemple avancé : utiliser plusieurs reclassargs

Crée un nouveau raster à partir de l'original où les bandes 1, 2, 3 sont respectivement converties en 1BB, 4BUI, 4BUI et reclassifiées. Cette opération utilise l'argument variadique `reclassarg` qui peut prendre en entrée un nombre variable de reclassargs (théoriquement autant de bandes que vous avez)

```
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;

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
```

Exemple avancé : Mapping d'un raster 32BF à bande unique en plusieurs bandes visualisables

Crée un nouveau raster à 3 bandes (8BUI, 8BUI, 8BUI visualisable) à partir d'un raster qui n'a qu'une seule bande de 32bf

```
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)
    ]
    );
```

Voir aussi

ST_AddBand, ST_Band, ST_BandPixelType, ST_MakeEmptyRaster, reclassarg, ST_Value

11.12.13 ST_Union

ST_Union — Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.

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

Retourne l'union d'un ensemble de tuiles raster en un seul raster, composé d'au moins une bande. L'étendue du raster résultant est l'étendue de l'ensemble. Dans le cas d'une intersection, la valeur résultante est définie par `uniontype` parmi : LAST (par défaut), FIRST, MIN, MAX, COUNT, SUM, MEAN, RANGE.

Note
Pour pouvoir calculer l'union de raster, tous les rasters doivent avoir le même alignement. Voir ST_SameAlignment et ST_NotSameAlignmentReason pour plus de détails. Une façon de résoudre les problèmes d'alignement est d'utiliser ST_Resample et d'utiliser le même raster de référence pour l'alignement.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Vitesse améliorée (entièrement en C).

Disponibilité : 2.1.0 Ajout de la variante ST_Union(rast, unionarg).

Amélioration : 2.1.0 ST_Union(rast) (variante 1) permet l'union sur toutes les bandes de tous les rasters d'entrée. Les versions précédentes de PostGIS ne prenaient en compte que la première bande.

Amélioration : 2.1.0 ST_Union(rast, uniontype) (variante 4) permet l'union de toutes les bandes de tous les rasters d'entrée.

Exemples : Reconstituer une tuile raster à partir d'une tuile raster fragmentée d'une seule bande

```
-- this creates a single band from first band of raster tiles
-- that form the original file system tile
SELECT filename, ST_Union(rast,1) As file_rast
FROM sometable WHERE filename IN('dem01', 'dem02') GROUP BY filename;
```

Exemples : Retourne un raster multi-bandes qui est l'union de tuiles intersectant une géométrie

```
-- this creates a multi band raster collecting all the tiles that intersect a line
-- Note: 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[]
SELECT ST_Union(rast)
FROM aerials.boston
WHERE ST_Intersects(rast,  ST_GeomFromText('LINESTRING(230486 887771, 230500 88772)',26986) ↩
    );
```

Exemples : Retourne un raster multi-bandes qui est l'union de tuiles intersectant une géométrie

Nous utilisons ici la syntaxe plus longue si nous ne voulons qu'un sous-ensemble de bandes ou si nous voulons changer l'ordre des bandes

```
-- this creates a multi band raster collecting all the tiles that intersect a line
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) ↩
    );
```

Voir aussi

unionarg, ST_Envelope, ST_ConvexHull, ST_Clip, ST_Union

11.13 Fonctions de rappel intégrées d'algèbre cartographique

11.13.1 ST_Distinct4ma

ST_Distinct4ma — Fonction de traitement des données raster qui calcule le nombre de valeurs de pixels uniques dans un voisinage.

Synopsis

float8 **ST_Distinct4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Distinct4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule le nombre de valeurs de pixels uniques dans un voisinage de pixels.

Note
La variante 1 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebraFctNgb.

Note
La variante 2 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Warning
L'utilisation de la variante 1 est non recommandée, ST_MapAlgebraFctNgb étant dépréciée depuis 2.1.0.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Ajout de la variante 2

Exemples

```
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;
 rid | st_value
-----+----------
   2 |        3
(1 row)
```

Voir aussi

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 — Fonction de traitement des données raster qui interpole la valeur d'un pixel à partir de son voisinage.

Synopsis

double precision **ST_InvDistWeight4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule une valeur interpolée pour un pixel en utilisant la méthode de la distance inverse pondérée.

Deux paramètres facultatifs peuvent être fournis par l'intermédiaire de `userargs`. Le premier paramètre est le facteur de puissance (variable k dans l'équation ci-dessous) entre 0 et 1, utilisé dans l'équation de la pondération de la distance inverse. S'il n'est pas spécifié, la valeur par défaut est 1. Le second paramètre est le pourcentage de pondération, appliqué uniquement lorsque la valeur du pixel d'intérêt est incluse dans la valeur interpolée du voisinage. S'il n'est pas spécifié et que le pixel d'intérêt a une valeur, cette valeur est retournée.

L'équation de base de la distance inverse pondérée est :

$$\hat{z}(x_o) = \frac{\sum_{j=1}^{m} z(x_j) d_{ij}^{-k}}{\sum_{j=1}^{m} d_{ij}^{-k}}$$

k = facteur de puissance, un nombre réel entre 0 et 1

Note
Cette fonction est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Disponibilité : 2.1.0

Exemples

```
-- NEEDS EXAMPLE
```

Voir aussi

ST_MapAlgebra (callback function version), ST_MinDist4ma

11.13.3 ST_Max4ma

ST_Max4ma — Fonction de traitement des données raster qui calcule la valeur maximale d'un pixel dans un voisinage.

Synopsis

float8 **ST_Max4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Max4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule la valeur maximale d'un pixel dans un voisinage de pixels.

Pour la variante 2, une valeur de substitution pour les pixels NODATA peut être spécifiée en passant cette valeur à userargs.

Note
La variante 1 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebraFctNgb.

Note
La variante 2 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Warning
L'utilisation de la variante 1 est non recommandée, ST_MapAlgebraFctNgb étant dépréciée depuis 2.1.0.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Ajout de la variante 2

Exemples

```
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;
 rid | st_value
-----+----------
   2 |      254
(1 row)
```

Voir aussi

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Sum4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.4 ST_Mean4ma

ST_Mean4ma — Fonction de traitement des données raster qui calcule la valeur moyenne d'un pixel dans un voisinage.

Synopsis

float8 **ST_Mean4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Mean4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule la valeur moyenne d'un pixel dans un voisinage de pixels.

Pour la variante 2, une valeur de substitution pour les pixels NODATA peut être spécifiée en passant cette valeur à userargs.

Note
La variante 1 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebraFctNgb.

Note
La variante 2 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Warning
L'utilisation de la variante 1 est non recommandée, ST_MapAlgebraFctNgb étant dépréciée depuis 2.1.0.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Ajout de la variante 2

Exemples : Variante 1

```
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;
 rid |     st_value
-----+------------------
   2 | 253.222229003906
(1 row)
```

Exemples : Variante 2

```
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;
 rid |     st_value
-----+------------------
   2 | 253.222229003906
(1 row)
```

Voir aussi

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Sum4ma, ST_Range4ma, ST_StdDev4ma

11.13.5 ST_Min4ma

ST_Min4ma — Fonction de traitement des données raster qui calcule la valeur minimale d'un pixel dans un voisinage.

Synopsis

float8 **ST_Min4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Min4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule la valeur minimale d'un pixel dans un voisinage de pixels.

Pour la variante 2, une valeur de substitution pour les pixels NODATA peut être spécifiée en passant cette valeur à userargs.

Note
La variante 1 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebraFctNgb.

Note
La variante 2 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Warning
L'utilisation de la variante 1 est non recommandée, ST_MapAlgebraFctNgb étant dépréciée depuis 2.1.0.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Ajout de la variante 2

Exemples

```
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;
 rid | st_value
-----+----------
   2 |      250
(1 row)
```

Voir aussi

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Max4ma, ST_Sum4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.6 ST_MinDist4ma

ST_MinDist4ma — Fonction de traitement des données raster qui renvoie la distance minimale (en nombre de pixels) entre le pixel d'intérêt et un pixel voisin ayant une valeur.

Synopsis

double precision **ST_MinDist4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Retourne la plus petite distance (en nombre de pixels) entre le pixel d'intérêt et le pixel le plus proche ayant une valeur dans le voisinage.

Note
Le but de cette fonction est de fournir un point de données informatif qui aide à déduire l'utilité de la valeur interpolée du pixel d'intérêt à partir de ST_InvDistWeight4ma. Cette fonction est particulièrement utile lorsque le voisinage est peu peuplé.

Note
Cette fonction est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Disponibilité : 2.1.0

Exemples

```
-- NEEDS EXAMPLE
```

Voir aussi

ST_MapAlgebra (callback function version), ST_InvDistWeight4ma

11.13.7 ST_Range4ma

ST_Range4ma — Fonction de traitement des données raster qui calcule la plage de valeurs des pixels dans un voisinage.

Synopsis

float8 **ST_Range4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Range4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule la plage de valeurs des pixels dans un voisinage de pixels.

Pour la variante 2, une valeur de substitution pour les pixels NODATA peut être spécifiée en passant cette valeur à userargs.

Note
La variante 1 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebraFctNgb.

Note
La variante 2 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Warning
L'utilisation de la variante 1 est non recommandée, ST_MapAlgebraFctNgb étant dépréciée depuis 2.1.0.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Ajout de la variante 2

Exemples

```
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;
 rid | st_value
-----+----------
   2 |        4
(1 row)
```

Voir aussi

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 — Fonction de traitement des données raster qui calcule l'écart type des valeurs des pixels dans un voisinage.

Synopsis

float8 **ST_StdDev4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_StdDev4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule l'écart type des valeurs des pixels dans un voisinage de pixels.

Note
La variante 1 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebraFctNgb.

Note
La variante 2 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Warning
L'utilisation de la variante 1 est non recommandée, ST_MapAlgebraFctNgb étant dépréciée depuis 2.1.0.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Ajout de la variante 2

Exemples

```
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;
 rid |     st_value
-----+------------------
   2 | 1.30170822143555
(1 row)
```

Voir aussi

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 — Fonction de traitement des données raster qui calcule la somme de toutes les valeurs de pixels dans un voisinage.

Synopsis

float8 **ST_Sum4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Sum4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

Calcule la somme de toutes les valeurs des pixels dans un voisinage de pixels.

Pour la variante 2, une valeur de substitution pour les pixels NODATA peut être spécifiée en passant cette valeur à userargs.

Note
La variante 1 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebraFctNgb.

Note
La variante 2 est une fonction de rappel spécialisée à utiliser comme paramètre de rappel pour ST_MapAlgebra (callback function version).

Warning
L'utilisation de la variante 1 est non recommandée, ST_MapAlgebraFctNgb étant dépréciée depuis 2.1.0.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Ajout de la variante 2

Exemples

```
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;
 rid | st_value
-----+----------
   2 |     2279
(1 row)
```

Voir aussi

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.14 Traitement des données raster : MNT (élévation)

11.14.1 ST_Aspect

ST_Aspect — Retourne l'exposition (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.

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

Retourne l'exposition (par défaut, en degrés) d'une bande raster d'élévation. Utilise l'algèbre cartographique et applique l'équation d'exposition aux pixels voisins.

`units` indique l'unité de l'exposition. Les valeurs possibles sont : RADIANS, DEGREES (par défaut).

Lorsque `units` = RADIANS, les valeurs sont comprises entre 0 et 2 * pi radians, mesurés dans le sens des aiguilles d'une montre à partir du nord.

Lorsque `units` = DEGREES, les valeurs sont comprises entre 0 et 360 degrés, mesurés dans le sens des aiguilles d'une montre à partir du nord.

Si la pente du pixel est nulle, l'exposition du pixel est -1.

Note
Pour plus d'informations sur la pente, l'exposition et l'ombrage, veuillez consulter ESRI - How hillshade works et ERDAS Field Guide - Aspect Images.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Utilise ST_MapAlgebra() et ajout du paramètre optionnel `interpolate_nodata`

Changement : 2.1.0 Dans les versions précédentes, les valeurs retournées étaient en radians. Désormais, les valeurs retournées sont par défaut en degrés

Exemples : Variante 1

```
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_Aspect(rast, 1, '32BF'))
FROM foo




----------------------------------------------------------------------------------------------------------------

-----------------------------------
 (1,"{{315,341.565063476562,0,18.4349479675293,45},{288.434936523438,315,0,45,71.5650482177734},{270

2227,180,161.565048217773,135}}")
(1 row)
```

Exemples : Variante 2

Exemple complet avec les tuiles d'une couverture. Cette requête ne fonctionne qu'avec PostgreSQL 9.1 ou plus.

```
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;
```

Voir aussi

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_HillShade, ST_Slope

11.14.2 ST_HillShade

ST_HillShade — Retourne l'hypothétique éclairage d'une bande raster d'élévation en utilisant les valeurs d'azimut, d'altitude, de luminosité et d'échelle fournies.

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

Retourne l'hypothétique éclairage d'une bande raster d'élévation en utilisant les données d'entrée d'azimut, d'altitude, de luminosité et d'échelle. Utilise l'algèbre cartographique et applique l'équation d'ombrage aux pixels voisins. Les valeurs des pixels retournées sont comprises entre 0 et 255.

`azimuth` est une valeur comprise entre 0 et 360 degrés, mesurée dans le sens des aiguilles d'une montre à partir du nord.

`altitude` est une valeur comprise entre 0 et 90 degrés, où 0 degré correspond à l'horizon et 90 degrés à la verticale.

`max_bright` est une valeur comprise entre 0 et 255, 0 correspondant à une absence de luminosité et 255 à une luminosité maximale.

`scale` est le rapport entre les unités verticales et les unités horizontales. Pour Feet:LatLon, utiliser scale=370400, pour Meters:LatLon utiliser scale=111120.

Si `interpolate_nodata` est TRUE, les valeurs des pixels NODATA du raster d'entrée seront interpolées à l'aide de ST_InvDistWeight4ma avant de calculer l'éclairage.

Note
Pour plus d'informations sur l'ombrage, veuillez consulter How hillshade works.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Utilise ST_MapAlgebra() et ajout du paramètre optionnel `interpolate_nodata`

Changement : 2.1.0 Dans les versions précédentes, l'azimut et l'altitude étaient exprimés en radians. Désormais, l'azimut et l'altitude sont exprimés en degrés

Exemples : Variante 1

```
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




----------------------------------------------------------------------------------------------------------------------------------


-------------------------------------------------------------------------
 (1,"{{NULL,NULL,NULL,NULL,NULL},{NULL,251.32763671875,220.749786376953,147.224319458008, ↩
     NULL},{NULL,220.749786376953,180.312225341797,67.7497863769531,NULL},{NULL ↩
     ,147.224319458008
,67.7497863769531,43.1210060119629,NULL},{NULL,NULL,NULL,NULL,NULL}}")
(1 row)
```

Exemples : Variante 2

Exemple complet avec les tuiles d'une couverture. Cette requête ne fonctionne qu'avec PostgreSQL 9.1 ou plus.

```
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;
```

Voir aussi

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_Aspect, ST_Slope

11.14.3 ST_Roughness

ST_Roughness — Retourne un raster avec la rugosité d'un MNT.

Synopsis

raster **ST_Roughness**(raster rast, integer nband, raster customextent, text pixeltype="32BF" , boolean interpolate_nodata=FALSE);

Description

Calcule la rugosité d'un MNT en soustrayant le maximum du minimum pour une zone donnée.

Disponibilité : 2.1.0

Exemples

```
-- needs examples
```

Voir aussi

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Slope, ST_HillShade, ST_Aspect

11.14.4 ST_Slope

ST_Slope — Retourne la pente (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.

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

Retourne la pente (par défaut, en degrés) d'une bande raster d'élévation. Utilise l'algèbre cartographique et applique l'équation de pente aux pixels voisins.

`units` indique l'unité de la pente. Les valeurs possibles sont : RADIANS, DEGREES (par défaut), PERCENT.

`scale` est le rapport entre les unités verticales et les unités horizontales. Pour Feet:LatLon, utiliser scale=370400, pour Meters:LatLon utiliser scale=111120.

Si `interpolate_nodata` est TRUE, les valeurs des pixels NODATA du raster d'entrée seront interpolées à l'aide de ST_InvDistWeight4ma avant de calculer la pente de la surface.

Note
Pour plus d'informations sur la pente, l'exposition et l'ombrage, veuillez consulter ESRI - How hillshade works et ERDAS Field Guide - Slope Images.

Disponibilité : 2.0.0

Amélioration : 2.1.0 Utilise ST_MapAlgebra() et ajout des paramètres optionnels `units`, `scale`, `interpolate_nodata`

Changement : 2.1.0 Dans les versions précédentes, les valeurs retournées étaient en radians. Désormais, les valeurs retournées sont par défaut en degrés

Exemples : Variante 1

```
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

------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------
 (1,"{{10.0249881744385,21.5681285858154,26.5650520324707,21.5681285858154,10.0249881744385},{21.568
```

```
{26.5650520324707,36.8698959350586,0,36.8698959350586,26.5650520324707},{21.5681285858154,35.2643890

5681285858154,26.5650520324707,21.5681285858154,10.0249881744385}}")
(1 row)
```

Exemples : Variante 2

Exemple complet avec les tuiles d'une couverture. Cette requête ne fonctionne qu'avec PostgreSQL 9.1 ou plus.

```
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;
```

Voir aussi

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_HillShade, ST_Aspect

11.14.5 ST_TPI

ST_TPI — Retourne un raster avec l'index de position topographique (TPI) calculé.

Synopsis

raster **ST_TPI**(raster rast, integer nband, raster customextent, text pixeltype="32BF" , boolean interpolate_nodata=FALSE);

Description

Calcule l'index de position topographique (TPI), qui est défini comme la moyenne focale avec un rayon de un moins la cellule centrale.

Note
Cette fonction ne prend en charge qu'un rayon focal de un.

Disponibilité : 2.1.0

Exemples

```
-- needs examples
```

Voir aussi

ST_MapAlgebra (callback function version), ST_TRI, ST_Roughness, ST_Slope, ST_HillShade, ST_Aspect

11.14.6 ST_TRI

ST_TRI — Retourne un raster avec l'indice de rugosité du terrain (TRI) calculé.

Synopsis

raster **ST_TRI**(raster rast, integer nband, raster customextent, text pixeltype="32BF" , boolean interpolate_nodata=FALSE);

Description

L'indice de rugosité du terrain est calculé en comparant le pixel central à ses voisins, en prenant les valeurs absolues des différences et en faisant la moyenne du résultat.

Note
Cette fonction ne prend en charge qu'un rayon focal de un.

Disponibilité : 2.1.0

Exemples

```
-- needs examples
```

Voir aussi

ST_MapAlgebra (callback function version), ST_Roughness, ST_TPI, ST_Slope, ST_HillShade, ST_Aspect

11.15 Traitement des données raster : raster vers géométrie

11.15.1 Box3D

Box3D — Retourne la représentation 3d de la boîte englobante du raster.

Synopsis

box3d **Box3D**(raster rast);

Description

Retourne la boîte représentant l'étendue du raster.

Le polygone est défini par les points des coins de la boîte englobante ((`MINX`, `MINY`), (`MAXX`, `MAXY`))

Changement : 2.0.0 Dans les versions antérieures à la version 2.0, utilisait le type box2d au lieu de box3d. Comme box2d est un type obsolète, il a été remplacé par box3d.

Exemples

```
SELECT
    rid,
    Box3D(rast) AS rastbox
FROM dummy_rast;

rid |        rastbox
----+------------------------------------------------
1   | BOX3D(0.5 0.5 0,20.5 60.5 0)
2   | BOX3D(3427927.75 5793243.5 0,3427928 5793244 0)
```

Voir aussi

ST_Envelope

11.15.2 ST_ConvexHull

ST_ConvexHull — Retourne l'enveloppe convexe du raster, en incluant les valeurs de pixels égales à BandNoDataValue. Pour les données raster de forme régulière et sans obliquité, cette fonction donne le même résultat que ST_Envelope ; elle n'est donc utile que pour les données raster de forme irrégulière ou inclinée.

Synopsis

geometry **ST_ConvexHull**(raster rast);

Description

Retourne l'enveloppe convexe du raster, en incluant les valeurs de pixels égales à BandNoDataValue. Pour les données raster de forme régulière et sans obliquité, cette fonction donne plus ou moins le même résultat que ST_Envelope ; elle n'est donc utile que pour les données raster de forme irrégulière ou inclinée.

Note
ST_Envelope arrondit les coordonnées à l'inférieur, et ajoute donc un petit tampon autour du raster. La réponse est donc subtilement différente de ST_ConvexHull qui n'arrondit pas les coordonnées.

Exemples

Voir PostGIS Raster Specification pour un diagramme de ceci.

```
-- Note envelope and convexhull are more or less the same
SELECT ST_AsText(ST_ConvexHull(rast)) As convhull,
    ST_AsText(ST_Envelope(rast)) As env
FROM dummy_rast WHERE rid=1;

                        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
SELECT ST_AsText(ST_ConvexHull(rast)) As convhull,
    ST_AsText(ST_Envelope(rast)) As env
FROM (SELECT ST_SetRotation(rast, 0.1, 0.1) As rast
    FROM dummy_rast WHERE rid=1) As foo;

                        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) ↩
    )
```

Voir aussi

ST_Envelope, ST_MinConvexHull, ST_ConvexHull, ST_AsText

11.15.3 ST_DumpAsPolygons

ST_DumpAsPolygons — Retourne un ensemble d'enregistrements de type geomval (geom, val), à partir d'une bande raster donnée. Si aucune bande bandnum n'est spécifiée, la bande 1 est utilisée.

Synopsis

setof geomval **ST_DumpAsPolygons**(raster rast, integer band_num=1, boolean exclude_nodata_value=TRUE);

Description

Il s'agit d'une fonction de retour d'ensemble (SRF). Elle retourne un ensemble d'enregistrements geomval, composés par une géométrie (geom) et une valeur de bande de pixels (val). Chaque polygone est l'union de tous les pixels de cette bande qui ont la même valeur de pixel, indiquée par val.

ST_DumpAsPolygon est utile pour polygoniser les données raster. C'est l'inverse d'un GROUP BY, en ce sens qu'elle crée de nouvelles lignes. Par exemple, cette fonction peut être utilisée pour décomposer un seul raster en plusieurs POLYGONS/MULTIPOLYGONS.

Changement 3.3.0, la validation et la correction sont désactivées pour améliorer les performances. Peut donc retourner des géométries non valides.

Disponibilité : Nécessite GDAL 1.7 ou plus.

Note
Si une valeur nodata est définie pour une bande, les pixels ayant cette valeur ne seront pas retournés, sauf si exclude_nodata_value=false.

Note
Si vous ne voulez que le nombre de pixels ayant une valeur donnée dans un raster, il est plus rapide d'utiliser ST_ValueCount.

Note
Cette fonction est différente de ST_PixelAsPolygons, où une géométrie est retournée pour chaque pixel, quelle que soit la valeur du pixel.

Exemples

```
-- this syntax requires PostgreSQL 9.3+
SELECT val, ST_AsText(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;

 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))
```

Voir aussi

geomval, ST_Value, ST_Polygon, ST_ValueCount

11.15.4 ST_Envelope

ST_Envelope — Retourne la représentation polygonale de l'étendue du raster.

Synopsis

geometry **ST_Envelope**(raster rast);

Description

Retourne la représentation polygonale de l'étendue du raster, dans les unités du système de référence spatial défini par srid. Il s'agit d'une boîte de délimitation minimale float8 représentée sous la forme d'un polygone.

Le polygone est défini par les points des coins de la boîte englobante ((MINX, MINY), (MINX, MAXY), (MAXX, MAXY), (MAXX, MINY), (MINX, MINY))

Exemples

```
SELECT rid, ST_AsText(ST_Envelope(rast)) As envgeomwkt
FROM dummy_rast;

 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))
```

Voir aussi

ST_Envelope, ST_AsText, ST_SRID

11.15.5 ST_MinConvexHull

ST_MinConvexHull — Retourne la géométrie de l'enveloppe convexe du raster en excluant les pixels NODATA.

Synopsis

geometry **ST_MinConvexHull**(raster rast, integer nband=NULL);

Description

Retourne la géométrie de l'enveloppe convexe du raster en excluant les pixels NODATA. Si nband est NULL, toutes les bandes du raster sont prises en compte.

Disponibilité : 2.1.0

Exemples

```
WITH foo AS (
    SELECT
        ST_SetValues(
            ST_SetValues(
                ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(9, 9, 0, 0, 1, -1, 0, 0, 0), 1, '8 ↩
                    BUI', 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
    ST_AsText(ST_ConvexHull(rast)) AS hull,
    ST_AsText(ST_MinConvexHull(rast)) AS mhull,
    ST_AsText(ST_MinConvexHull(rast, 1)) AS mhull_1,
    ST_AsText(ST_MinConvexHull(rast, 2)) AS mhull_2
FROM foo

               hull                |                mhull                |                ↩
                   mhull_1                |                mhull_2
-----------------------------------+-----------------------------------+----------------------------

 POLYGON((0 0,9 0,9 -9,0 -9,0 0)) | POLYGON((0 -3,9 -3,9 -9,0 -9,0 -3)) | POLYGON((3 -3,9  ↩
    -3,9 -6,3 -6,3 -3)) | POLYGON((0 -3,6 -3,6 -9,0 -9,0 -3))
```

Voir aussi

ST_Envelope, ST_ConvexHull, ST_ConvexHull, ST_AsText

11.15.6 ST_Polygon

ST_Polygon — Retourne une géométrie multipolygonale formée par l'union des pixels dont la valeur est différente de nodata. Si aucune bande bandnum n'est spécifiée, la bande 1 est utilisée.

Synopsis

geometry **ST_Polygon**(raster rast, integer band_num=1);

Description

Changement 3.3.0, la validation et la correction sont désactivées pour améliorer les performances. Peut donc retourner des géométries non valides.

Disponibilité : 0.1.6 Nécessite GDAL 1.7 ou plus.

Amélioration : 2.1.0 Vitesse améliorée (entièrement en C) et le multipolygone retourné est garanti d'être valide.

Changement : 2.1.0 Dans les versions précédentes, retournait parfois un polygone, retourne désormais toujours un multipolygon.

Exemples

```
-- by default no data band value is 0 or not set, so polygon will return a square polygon
SELECT ST_AsText(ST_Polygon(rast)) As geomwkt
FROM dummy_rast
WHERE rid = 2;

geomwkt
-----------------------------------------
MULTIPOLYGON(((3427927.75 5793244,3427928 5793244,3427928 5793243.75,3427927.75  ↩
    5793243.75,3427927.75 5793244)))


-- now we change the no data value of first band
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast,1,254)
WHERE rid = 2;
SELECt rid, ST_BandNoDataValue(rast)
from dummy_rast where rid = 2;

-- ST_Polygon excludes the pixel value 254 and returns a multipolygon
SELECT ST_AsText(ST_Polygon(rast)) As geomwkt
FROM dummy_rast
WHERE rid = 2;

geomwkt
-----------------------------------------------------------
MULTIPOLYGON(((3427927.9 5793243.95,3427927.85 5793243.95,3427927.85 5793244,3427927.9  ↩
    5793244,3427927.9 5793243.95)),((3427928 5793243.85,3427928 5793243.8,3427927.95  ↩
    5793243.8,3427927.95 5793243.85,3427927.9 5793243.85,3427927.9 5793243.9,3427927.9  ↩
    5793243.95,3427927.95 5793243.95,3427928 5793243.95,3427928 5793243.85)),((3427927.8  ↩
    5793243.75,3427927.75 5793243.75,3427927.75 5793243.8,3427927.75 5793243.85,3427927.75  ↩
    5793243.9,3427927.75 5793244,3427927.8 5793244,3427927.8 5793243.9,3427927.8  ↩
    5793243.85,3427927.85 5793243.85,3427927.85 5793243.8,3427927.85 5793243.75,3427927.8  ↩
    5793243.75)))

-- Or if you want the no data value different for just one time

SELECT ST_AsText(
    ST_Polygon(
        ST_SetBandNoDataValue(rast,1,252)
        )
    ) As geomwkt
FROM dummy_rast
WHERE rid =2;

geomwkt
---------------------------------
MULTIPOLYGON(((3427928 5793243.85,3427928 5793243.8,3427928 5793243.75,3427927.85  ↩
    5793243.75,3427927.8 5793243.75,3427927.8 5793243.8,3427927.75 5793243.8,3427927.75  ↩
    5793243.85,3427927.75 5793243.9,3427927.75 5793244,3427927.8 5793244,3427927.85  ↩
    5793244,3427927.9 5793244,3427928 5793244,3427928 5793243.95,3427928 5793243.85) ↩
    ,(3427927.9 5793243.9,3427927.9 5793243.85,3427927.95 5793243.85,3427927.95  ↩
    5793243.9,3427927.9 5793243.9)))
```

Voir aussi

ST_Value, ST_DumpAsPolygons

11.16 Opérateurs raster

11.16.1 &&

&& — Retourne TRUE si la boite englobante de A intersecte la boite englobante de B.

Synopsis

boolean **&&**(raster A , raster B);
boolean **&&**(raster A , geometry B);
boolean **&&**(geometry B , raster A);

Description

L'opérateur && retourne TRUE si la boîte englobante du raster/géométrie A intersecte la boîte englobante du raster/géométrie B.

Note
Cet opérande utilisera tous les index qui peuvent être disponibles sur les données raster.

Disponibilité : 2.0.0

Exemples

```
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;

 a_rid | b_rid | intersect
-------+-------+---------
     2 |     2 | t
     2 |     3 | f
     2 |     1 | f
```

11.16.2 &<

&< — Retourne TRUE si la boîte englobante de A est à gauche de celle de B.

Synopsis

boolean **&<**(raster A , raster B);

Description

L'opérateur &< retourne TRUE si la boîte englobante du raster A chevauche ou est à gauche de la boîte englobante de B ou, plus précisément, si elle chevauche ou n'est PAS à droite de la boîte englobante de B.

Note
Cet opérande utilisera tous les index qui peuvent être disponibles sur les données raster.

Exemples

```
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;

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 &>

&> — Retourne TRUE si la boîte englobante de A est à droite de celle de B.

Synopsis

boolean **&>**(raster A , raster B);

Description

L'opérateur &> retourne TRUE si la boîte englobante du raster A chevauche ou est à droite de la boîte englobante du raster B ou, plus précisément, si elle chevauche ou n'est PAS à gauche de la boîte englobante du raster B.

Note
Cette opérande utilisera tous les index qui peuvent être disponibles sur les géométries.

Exemples

```
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;

 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 =

= — Retourne TRUE si la boîte englobante de A est la même que celle de B. Utilise des boîtes englobantes en double précision.

Synopsis

boolean =(raster A , raster B);

Description

L'opérateur = retourne TRUE si la boîte englobante du raster A est la même que celle du raster B. PostgreSQL utilise les opérateurs =, <, et > définis pour les rasters pour effectuer des tris et comparaisons internes sur les rasters (par exemple, dans une clause GROUP BY ou ORDER BY).

Caution
Cet opérande n'utilisera PAS les index éventuellement disponibles sur les données raster. Utilisez plutôt ~=. Cet opérateur existe principalement pour permettre le regroupement par colonne raster.

Disponibilité : 2.1.0

Voir aussi

~=

11.16.5 @

@ — Retourne TRUE si la boîte englobante de A est contenue dans celle de B. Utilise des boîtes englobantes en double précision.

Synopsis

boolean @(raster A , raster B);
boolean @(geometry A , raster B);
boolean @(raster B , geometry A);

Description

L'opérateur @ retourne TRUE si la boîte englobante du raster/géométrie A est contenue dans la boîte englobante du raster/géométrie B.

Note
Cet opérande utilisera des index spatiaux sur les données raster.

Disponibilité : 2.0.0 ajout de raster @ raster, raster @ geometry

Disponibilité : 2.0.5 ajout de geometry @ raster

Voir aussi

~

11.16.6 ~=

~= — Renvoie TRUE si la boîte de délimitation de A est la même que celle de B.

Synopsis

boolean ~=(raster A , raster B);

Description

L'opérateur ~= retourne TRUE si la boîte englobante du raster A est la même que la boîte englobante du raster B.

Note
Cet opérande utilisera tous les index qui peuvent être disponibles sur les données raster.

Disponibilité : 2.0.0

Exemples

Un cas d'utilisation très utile est celui où l'on prend deux ensembles de rasters à bande unique qui sont du même morceau mais représentent des thèmes différents, et où l'on crée un raster multi-bandes

```
SELECT ST_AddBand(prec.rast, alt.rast) As new_rast
    FROM prec INNER JOIN alt ON (prec.rast ~= alt.rast);
```

Voir aussi

ST_AddBand, =

11.16.7 ~

~ — Retourne TRUE si la boîte englobante de A contient celle de B. Utilise des boîtes englobantes en double précision.

Synopsis

boolean ~(raster A , raster B);
boolean ~(geometry A , raster B);
boolean ~(raster B , geometry A);

Description

L'opérateur ~ retourne TRUE si la boîte englobante du raster/géométrie A contient la boîte englobante du raster/géométrie B.

Note
Cet opérande utilisera des index spatiaux sur les données raster.

Disponibilité : 2.0.0

Voir aussi

@

11.17 Relations spatiales entre raster et entre bandes raster

11.17.1 ST_Contains

ST_Contains — Retourne true si aucun point du raster rastB ne se trouve à l'extérieur du raster rastA et si au moins un point de l'intérieur du raster rastB se trouve à l'intérieur du raster rastA.

Synopsis

boolean **ST_Contains**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Contains**(raster rastA , raster rastB);

Description

Le raster rastA contient rastB si et seulement si aucun point de rastB ne se trouve à l'extérieur de rastA et si au moins un point de l'intérieur de rastB se trouve à l'intérieur de rastA. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si un numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cette fonction utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_Contains(ST_Polygon(raster), geometry) ou ST_Contains(geometry, ST_Polygon(raster)).

Note
ST_Contains() est l'inverse de ST_Within(). Ainsi, ST_Contains(rastA, rastB) implique ST_Within(rastB, rastA).

Disponibilité : 2.1.0

Exemples

```
-- specified band numbers
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;

NOTICE:  The first raster provided has no bands
 rid | rid | st_contains
-----+-----+-------------
   1 |   1 |
   1 |   2 | f
```

```
-- no band numbers specified
SELECT r1.rid, r2.rid, ST_Contains(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 1;
 rid | rid | st_contains
-----+-----+-------------
   1 |   1 | t
   1 |   2 | f
```

Voir aussi

ST_Intersects, ST_Within

11.17.2 ST_ContainsProperly

ST_ContainsProperly — Retourne true si rastB intersecte l'intérieur de rastA, mais pas la frontière ou l'extérieur de rastA.

Synopsis

boolean **ST_ContainsProperly**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_ContainsProperly**(raster rastA , raster rastB);

Description

Le raster rastA contient proprement rastB si rastB intersecte l'intérieur de rastA mais pas la limite ou l'extérieur de rastA. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Le raster rastA ne contient pas proprement lui-même, mais il se contient lui-même.

Note
Cette fonction utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_ContainsProperly(ST_Polygon(raster), geometry) ou ST_ContainsProperly(geometry, ST_Polygon(raster)).

Disponibilité : 2.1.0

Exemples

```
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;

 rid | rid | st_containsproperly
-----+-----+---------------------
   2 |   1 | f
   2 |   2 | f
```

Voir aussi

ST_Intersects, ST_Contains

11.17.3 ST_Covers

ST_Covers — Retourne true si aucun point du raster rastB ne se trouve à l'extérieur du raster rastA.

Synopsis

boolean **ST_Covers**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Covers**(raster rastA , raster rastB);

Description

Le raster rastA recouvre le raster rastB si et seulement si aucun point du raster rastB ne se trouve à l'extérieur du raster rastA. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cette fonction utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_Covers(ST_Polygon(raster), geometry) ou ST_Covers(geometry, ST_Polygon(raster)).

Disponibilité : 2.1.0

Exemples

```
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;

 rid | rid | st_covers
-----+-----+-----------
   2 |   1 | f
   2 |   2 | t
```

Voir aussi

ST_Intersects, ST_CoveredBy

11.17.4 ST_CoveredBy

ST_CoveredBy — Retourne true si aucun point du raster rastA ne se trouve à l'extérieur du raster rastB.

Synopsis

boolean **ST_CoveredBy**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_CoveredBy**(raster rastA , raster rastB);

Description

Le raster rastA est couvert par rastB si et seulement si aucun point de rastA ne se trouve à l'extérieur de rastB. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cette fonction utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_CoveredBy(ST_Polygon(raster), geometry) ou ST_CoveredBy(geometry, ST_Polygon(raster)).

Disponibilité : 2.1.0

Exemples

```
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;

 rid | rid | st_coveredby
-----+-----+--------------
   2 |   1 | f
   2 |   2 | t
```

Voir aussi

ST_Intersects, ST_Covers

11.17.5 ST_Disjoint

ST_Disjoint — Retourne true si le raster rastA n'intersecte pas spatialement le rastB.

Synopsis

boolean **ST_Disjoint**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Disjoint**(raster rastA , raster rastB);

Description

Les rasters rastA et rastB sont disjoints s'ils ne partagent aucun espace. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cette fonction n'utilise PAS d'index.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_Disjoint(ST_Polygon(raster), geometry).

Disponibilité : 2.1.0

Exemples

```
-- rid = 1 has no bands, hence the NOTICE and the NULL value for st_disjoint
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;

NOTICE:  The second raster provided has no bands
 rid | rid | st_disjoint
-----+-----+-------------
   2 |   1 |
   2 |   2 | f
```

```
-- this time, without specifying band numbers
SELECT r1.rid, r2.rid, ST_Disjoint(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN  ↩
    dummy_rast r2 WHERE r1.rid = 2;

 rid | rid | st_disjoint
-----+-----+-------------
   2 |   1 | t
   2 |   2 | f
```

Voir aussi

ST_Intersects

11.17.6 ST_Intersects

ST_Intersects — Retourne vrai si le raster rastA intersecte spatialement le 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

Retourne vrai si le raster rastA intersecte spatialement le raster rastB. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cette fonction utilisera tous les index qui peuvent être disponibles sur les données raster.

Amélioration : 2.0.0 ajout de l'intersection raster/raster.

Warning
Changement : 2.1.0 Le fonctionnement des variantes ST_Intersects(raster, geometry) a changé pour correspondre au fonctionnement de ST_Intersects(geometry, raster).

Exemples

```
-- different bands of same raster
SELECT ST_Intersects(rast, 2, rast, 3) FROM dummy_rast WHERE rid = 2;

 st_intersects
---------------
 t
```

Voir aussi

ST_Intersection, ST_Disjoint

11.17.7 ST_Overlaps

ST_Overlaps — Retourne true si les raster rastA et rastB intersectent mais que l'un ne contient pas complètement l'autre.

Synopsis

boolean **ST_Overlaps**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Overlaps**(raster rastA , raster rastB);

Description

Retourne true si le raster rastA recouvre spatialement le raster rastB. Cela signifie que rastA et rastB intersectent mais que l'un ne contient pas complètement l'autre. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cette fonction utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_Overlaps(ST_Polygon(raster), geometry).

Disponibilité : 2.1.0

Exemples

```
-- comparing different bands of same raster
SELECT ST_Overlaps(rast, 1, rast, 2) FROM dummy_rast WHERE rid = 2;

 st_overlaps
-------------
 f
```

Voir aussi

ST_Intersects

11.17.8 ST_Touches

ST_Touches — Retourne true si les raster rastA et rastB ont au moins un point en commun mais que leurs intérieurs n'intersectent pas.

Synopsis

boolean **ST_Touches**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Touches**(raster rastA , raster rastB);

Description

Retourne true si le raster rastA touche spatialement le raster rastB. Cela signifie que rastA et rastB ont au moins un point en commun mais que leurs intérieurs n'intersectent pas. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cette fonction utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_Touches(ST_Polygon(raster), geometry).

Disponibilité : 2.1.0

Exemples

```
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;

 rid | rid | st_touches
-----+-----+------------
   2 |   1 | f
   2 |   2 | f
```

Voir aussi

ST_Intersects

11.17.9 ST_SameAlignment

ST_SameAlignment — Retourne true si les rasters ont les mêmes skew, scale, spatial ref, et offset (les pixels peuvent être placés sur la même grille sans être coupés) et false si ce n'est pas le cas, avec une NOTICE détaillant le problème.

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

Version non agrégée (variantes 1 et 2) : Retourne true si les deux rasters (fournis directement ou créés en utilisant les valeurs pour upperleft, scale, skew et srid) ont la même échelle, le même skew, le même srid et si au moins un des quatre coins de n'importe quel pixel d'un raster tombe sur n'importe quel coin de la grille de l'autre raster. Retourne false si ce n'est pas le cas et un NOTICE détaillant le problème d'alignement.

Version agrégée (variante 3) : A partir d'un ensemble de rasters, retourne vrai si tous les rasters de l'ensemble sont alignés. La fonction ST_SameAlignment() est une fonction "agrégée" dans la terminologie de PostgreSQL. Cela signifie qu'elle opère sur des lignes de données, de la même manière que les fonctions SUM() et AVG().

Disponibilité : 2.0.0

Amélioration : 2.1.0 ajout de la variante agrégée

Exemples : Rasters

```
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;

sm
----
t
```

```
SELECT ST_SameAlignment(A.rast,b.rast)
 FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;

 NOTICE:  The two rasters provided have different SRIDs
NOTICE:  The two rasters provided have different SRIDs
 st_samealignment
------------------
 t
 f
 f
 f
```

Voir aussi

Section 10.1, ST_NotSameAlignmentReason, ST_MakeEmptyRaster

11.17.10 ST_NotSameAlignmentReason

ST_NotSameAlignmentReason — Retourne un texte indiquant si les rasters sont alignés et, s'ils ne le sont pas, la raison du problème.

Synopsis

text **ST_NotSameAlignmentReason**(raster rastA, raster rastB);

Description

Retourne un texte indiquant si les rasters sont alignés et, s'ils ne le sont pas, la raison du problème.

Note
S'il y a plusieurs raisons pour lesquelles les rasters ne sont pas alignés, une seule raison (le premier test qui a échoué) sera retournée.

Disponibilité : 2.1.0

Exemples

```
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)
    )
;

 st_samealignment |            st_notsamealignmentreason
------------------+-------------------------------------------------
 f                | The rasters have different scales on the X axis
(1 row)
```

Voir aussi

Section 10.1, ST_SameAlignment

11.17.11 ST_Within

ST_Within — Retourne true si aucun point du raster rastA ne se trouve à l'extérieur du raster rastB et si au moins un point de l'intérieur du raster rastA se trouve à l'intérieur du raster rastB.

Synopsis

boolean **ST_Within**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Within**(raster rastA , raster rastB);

Description

Le raster rastA est compris dans rastB si et seulement si aucun point de rastA ne se trouve à l'extérieur de rastB et si au moins un point de l'intérieur de rastA se trouve à l'intérieur de rastB. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

Note
Cet opérande utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_Within(ST_Polygon(raster), geometry) ou ST_Within(geometry, ST_Polygon(raster)).

Note
ST_Within() est l'inverse de ST_Contains(). Ainsi, ST_Within(rastA, rastB) implique ST_Contains(rastB, rastA).

Disponibilité : 2.1.0

Exemples

```
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;

 rid | rid | st_within
-----+-----+-----------
   2 |   1 | f
   2 |   2 | t
```

Voir aussi

ST_Intersects, ST_Contains, ST_DWithin, ST_DFullyWithin

11.17.12 ST_DWithin

ST_DWithin — Retourne true si les rasters rastA et rastB se trouvent à une distance donnée l'un de l'autre.

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

Retourne true si les rasters rastA et rastB sont à la distance spécifiée l'un de l'autre. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

La distance est spécifiée en unités définies par le système de référence spatiale des rasters. Pour que cette fonction ait un sens, les rasters sources doivent avoir la même projection de coordonnées et le même SRID.

Note
Cet opérande utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_DWithin(ST_Polygon(raster), geometry).

Disponibilité : 2.1.0

Exemples

```
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;

 rid | rid | st_dwithin
-----+-----+------------
   2 |   1 | f
   2 |   2 | t
```

Voir aussi

ST_Within, ST_DFullyWithin

11.17.13 ST_DFullyWithin

ST_DFullyWithin — Retourne true si les rasters rastA et rastB se trouvent entièrement à une distance donnée l'un de l'autre.

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

Retourne true si les rasters rastA et rastB sont entièrement à une distance donnée l'un de l'autre. Si le numéro de bande n'est pas spécifié (ou si NULL), seule l'enveloppe convexe du raster est prise en compte dans le test. Si le numéro de bande est spécifié, seuls les pixels ayant une valeur (différente de NODATA) sont pris en compte dans le test.

La distance est spécifiée en unités définies par le système de référence spatiale des rasters. Pour que cette fonction ait un sens, les rasters sources doivent avoir la même projection de coordonnées et le même SRID.

Note
Cet opérande utilisera tous les index qui peuvent être disponibles sur les données raster.

Note
Pour tester la relation spatiale entre un raster et une géométrie, utilisez ST_Polygon sur le raster, par exemple ST_DFullyWithin(ST_Polygon(raster), geometry).

Disponibilité : 2.1.0

Exemples

```
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;

 rid | rid | st_dfullywithin
-----+-----+-----------------
   2 |   1 | f
   2 |   2 | t
```

Voir aussi

ST_Within, ST_DWithin

11.18 Astuces raster

11.18.1 Rasters out-DB

11.18.1.1 Répertoire contenant de nombreux fichiers

Lorsque GDAL ouvre un fichier, il parcourt l'entièreté du répertoire du fichier pour construire un catalogue d'autres fichiers. Si ce répertoire contient de nombreux fichiers (par exemple des milliers, des millions), l'ouverture de ce fichier devient extrêmement lente (en particulier si ce fichier se trouve sur un lecteur réseau tel que NFS).

Pour contrôler ce comportement, GDAL fournit la variable d'environnement suivante : GDAL_DISABLE_READDIR_ON_OPEN. Définissez `GDAL_DISABLE_READDIR_ON_OPEN` à `TRUE` pour désactiver l'analyse des répertoires.

Sous Ubuntu (et en supposant que vous utilisez les paquets PostgreSQL pour Ubuntu), `GDAL_DISABLE_READDIR_ON_OPEN` peut être défini dans */etc/postgresql/POSTGRESQL_VERSION/CLUSTER_NAME/environment* (où POSTGRESQL_VERSION est la version de PostgreSQL, par ex.9.6 et CLUSTER_NAME est le nom du cluster, par exemple maindb). Vous pouvez également définir les variables d'environnement PostGIS ici.

```
# 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 Nombre maximum de fichiers ouverts

Le nombre maximum de fichiers ouverts autorisé par Linux et PostgreSQL est généralement prudent (typiquement 1024 fichiers ouverts par processus), étant donné l'hypothèse que le système est utilisé par des utilisateurs humains. Pour les rasters Out-DB, une seule requête valide peut facilement dépasser cette limite (par exemple, un jeu de données de 10 ans avec un raster pour chaque jour contenant les températures minimales et maximales et nous voulons connaître les valeurs min et max absolues pour un pixel dans ce jeu de données).

Le changement le plus simple à effectuer est le paramètre suivant de PostgreSQL : max_files_per_process. La valeur par défaut est de 1000, ce qui est beaucoup trop faible pour les données raster Out-DB. Une valeur de départ sûre pourrait être 65536, mais cela dépend vraiment de vos ensembles de données et des requêtes exécutées sur ces ensembles de données. Ce paramètre ne peut être défini qu'au démarrage du serveur et probablement uniquement dans le fichier de configuration de PostgreSQL (par exemple */etc/postgresql/POSTGRESQL_VERSION/CLUSTER_NAME/postgresql.conf* dans les environnements Ubuntu).

```
...
# - Kernel Resource Usage -

max_files_per_process = 65536           # min 25
                                        # (change requires restart)
...
```

La principale modification à apporter concerne les limites des fichiers ouverts du noyau Linux. Il y a deux parties à cela :

- Nombre maximal de fichiers ouverts pour l'ensemble du système
- Nombre maximal de fichiers ouverts par processus

11.18.1.2.1 Nombre maximal de fichiers ouverts pour l'ensemble du système

L'exemple suivant vous permet de connaître le nombre maximum de fichiers ouverts sur l'ensemble du système :

```
$ sysctl -a | grep fs.file-max
fs.file-max = 131072
```

Si la valeur retournée n'est pas assez importante, ajoutez un fichier dans */etc/sysctl.d/* comme dans l'exemple suivant :

```
$ 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 Nombre maximal de fichiers ouverts par processus

Nous devons augmenter le nombre maximum de fichiers ouverts par processus pour les processus du serveur PostgreSQL.

Pour connaître le nombre maximal de fichiers ouverts utilisé par les processus de service PostgreSQL, suivez l'exemple suivant (assurez-vous que PostgreSQL est en cours d'exécution) :

```
$ ps aux | grep postgres
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

$ cat /proc/31718/limits
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
```

Dans l'exemple ci-dessus, nous avons inspecté la limite des fichiers ouverts pour le processus 31718. Peu importe le processus PostgreSQL, n'importe lequel fera l'affaire. La réponse qui nous intéresse est *Max open files*.

Nous voulons augmenter *Soft Limit* et *Hard Limit* de *Max open files* pour qu'il soit supérieur à la valeur que nous avons spécifiée pour le paramètre PostgreSQL `max_files_per_process`. Dans notre exemple, nous avons fixé `max_files_per_process` à 65536.

Sous Ubuntu (et en supposant que vous utilisez les paquets PostgreSQL pour Ubuntu), la façon la plus simple de changer les paramètres *Soft Limit* et *Hard Limit* est d'éditer **/etc/init.d/postgresql** (SysV) ou **/lib/systemd/system/postgresql*.service** (systemd).

Commençons par le cas de SysV Ubuntu où nous ajoutons **ulimit -H -n 262144** et **ulimit -n 131072** à **/etc/init.d/postgresql**.

```
...
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)
...
```

Passons maintenant au cas de systemd Ubuntu. Nous ajouterons **LimitNOFILE=131072** à chaque fichier **/lib/systemd/system/-postgresql*.service** dans la section **[Service]**.

```
...
[Service]

LimitNOFILE=131072

...

[Install]
WantedBy=multi-user.target
...
```

Après avoir effectué les changements nécessaires à systemd, assurez-vous de recharger le démon

```
systemctl daemon-reload
```

Chapter 12

PostGIS Extras

Ce chapitre documente les fonctionnalités trouvées dans le dossier extras des fichiers tarballs et du dépôt de sources de PostGIS. Celles-ci ne sont pas toujours packagées avec les versions binaires de PostGIS, mais sont généralement basées sur PL/pgSQL ou des scripts shell standard qui peuvent être exécutés tels quels.

12.1 Address Standardizer

Il s'agit d'un fork du PAGC standardizer (le code original de cette partie était PAGC PostgreSQL Address Standardizer).

Address standardizer est un analyseur d'adresses qui prend une adresse en entrée et la normalise sur la base d'un ensemble de règles stockées dans une table et des tables d'aide lex et gaz.

Le code est intégré dans une seule bibliothèque d'extension PostgreSQL appelée `address_standardizer` qui peut être installée avec `CREATE EXTENSION address_standardizer;`. En plus de l'extension address_standardizer, une extension de données d'exemple appelée `address_standardizer_data_us` extensions est construite, qui contient des tables de gaz, de lex et de règles pour les données américaines. Cette extension peut être installée via : `CREATE EXTENSION address_standardizer_data_us;`

Le code de cette extension se trouve dans le `extensions/address_standardizer` de PostGIS et est actuellement indépendant.

Pour les instructions d'installation, voir : Section 2.3.

12.1.1 Fonctionnement de l'analyseur

L'analyseur fonctionne de droite à gauche en examinant d'abord les macroéléments (code postal, état/province, ville), puis les microéléments afin de déterminer s'il s'agit d'un numéro de maison, d'une rue, d'une intersection ou d'un point de repère. Pour l'instant, il ne recherche pas le code ou le nom du pays, mais cela pourrait être introduit à l'avenir.

Code pays On suppose qu'il s'agit des États-Unis ou du Canada sur la base des éléments suivants : code postal (États-Unis ou Canada), état/province (États-Unis ou Canada), autre (États-Unis)

Code postal Ils sont reconnus à l'aide d'expressions régulières compatibles avec Perl. Ces expressions régulières se trouvent actuellement dans le fichier parseaddress-api.c et sont relativement simples à modifier si nécessaire.

État/province Ils sont reconnus à l'aide d'expressions régulières compatibles avec Perl. Ces expressions régulières sont actuellement dans le fichier parseaddress-api.c mais pourraient être déplacées dans includes dans le futur pour faciliter la maintenance.

12.1.2 Types de normalisateurs d'adresses

12.1.2.1 stdaddr

stdaddr — Type composite composé des éléments d'une adresse. Il s'agit du type de retour de la fonction `standardize_address`.

Description

Type composite composé d'éléments d'une adresse. Il s'agit du type de retour pour la fonction standardize_address. Certaines descriptions d'éléments sont empruntées à PAGC Postal Attributes.

Les numéros de jetons indiquent le numéro de référence de la sortie dans le rules table.

Cette méthode nécessite l'extension address_standardizer.

building est un texte (jeton numéro 0) : Fait référence au numéro ou au nom du bâtiment. Identifiants et types de bâtiments non analysés. Généralement vide pour la plupart des adresses.

house_num est un texte (jeton numéro 1) : Il s'agit du numéro d'une rue. Exemple *75* dans `75 State Street`.

predir est un texte (jeton numéro 2) : NOM DE RUE PRÉDIRECTIONNEL tel que Nord, Sud, Est, Ouest, etc.

qual est un texte (jeton numéro 3) : NOM DE RUE PRE-MODIFIER Exemple *OLD* dans `3715 OLD HIGHWAY 99`.

pretype est un texte (jeton numéro 4) : TYPE DE PRÉFIXE DE RUE

name est un texte (jeton numéro 5) : NOM DE RUE

suftype est un texte (jeton numéro 6) : STREET POST TYPE e.g. St, Ave, Cir. Un type de rue suivant le nom de la rue racine. Exemple *STREET* dans `75 State Street`.

sufdir est un texte (jeton numéro 7) : STREET POST-DIRECTIONAL Un modificateur directionnel qui suit le nom de la rue. Exemple *WEST* dans `3715 TENTH AVENUE WEST`.

ruralroute est un texte (jeton numéro 8) : RURAL ROUTE . Exemple *7* dans `RR 7`.

extra est un texte : Informations supplémentaires telles que le numéro d'étage.

city est un texte (jeton numéro 10) : Exemple Boston.

state est un texte (jeton numéro 11) : Exemple `MASSACHUSETTS`

country est un texte (jeton numéro 12) : Exemple `USA`

postcode est le texte POSTAL CODE (ZIP CODE) (numéro de jeton 13) : Exemple `02109`

box est le texte POSTAL BOX NUMBER (numéro de jeton 14 et 15) : Exemple `02109`

unit est le texte Numéro d'appartement ou numéro de suite (numéro de jeton 17) : Exemple *3B* dans `APT 3B`.

12.1.3 Tables Address Standardizer

12.1.3.1 rules table

rules table — La table rules contient un ensemble de règles qui établit une correspondance entre les jetons de la séquence d'entrée de l'adresse et la séquence de sortie normalisée. Une règle est définie comme un ensemble de jetons d'entrée suivi de -1 (terminateur) suivi d'un ensemble de jetons de sortie suivi de -1 suivi d'un nombre indiquant le type de règle suivi du classement de la règle.

Description

La table rules doit comporter au moins les colonnes suivantes, mais vous pouvez en ajouter d'autres pour vos propres besoins.

id Clé primaire de la table

rule champ de texte indiquant la règle. Pour plus de détails, voir PAGC Address Standardizer Rule records.

Une règle se compose d'un ensemble d'entiers non négatifs représentant les jetons d'entrée, terminés par un -1, suivis d'un nombre égal d'entiers non négatifs représentant les attributs postaux, terminés par un -1, suivis d'un entier représentant un type de règle, suivi d'un entier représentant le rang de la règle. Les règles sont classées de 0 (la plus basse) à 17 (la plus haute).

Ainsi, par exemple, la règle `2 0 2 22 3 -1 5 5 6 7 3 -1 2 6` associe la séquence de jetons de sortie *TYPE NUMBER TYPE DIRECT QUALIF* à la séquence de sortie *STREET STREET SUFTYP SUFDIR QUALIF*. La règle est une règle ARC_C de rang 6.

Les numéros des jetons de sortie correspondants sont énumérés dans stdaddr.

Jetons d'entrée

Chaque règle commence par un ensemble de jetons d'entrée suivi d'un terminateur `-1`. Les jetons d'entrée valides extraits de Tokens d'entrée du PAGC sont les suivants :

Jetons de saisie basés sur la forme

AMPERS (13). L'esperluette (&) est fréquemment utilisée pour abréger le mot "et".

DASH (9). Un caractère de ponctuation.

DOUBLE (21). Séquence de deux lettres. Souvent utilisé comme identifiant.

FRACT (25). Les fractions sont parfois utilisées dans les nombres civiques ou les nombres unitaires.

MIXED (23). Chaîne alphanumérique contenant à la fois des lettres et des chiffres. Utilisée pour les identifiants.

NUMBER (0). Une chaîne de chiffres.

ORD (15). Représentations telles que First or 1st. Souvent utilisé dans les noms de rue.

ORD (18). Une seule lettre.

WORD (1). Un mot est une chaîne de lettres de longueur arbitraire. Une même lettre peut être à la fois SINGLE et WORD.

Jetons d'entrée basés sur des fonctions

BOXH (14). Mots utilisés pour désigner les boîtes postales. Par exemple *Box* ou *PO Box*.

BUILDH (19). Mots utilisés pour désigner des bâtiments ou des ensembles de bâtiments, généralement sous forme de préfixe. Par exemple : *Tower* dans *Tower 7A*.

BUILDT (24). Mots et abréviations utilisés pour désigner des bâtiments ou des ensembles de bâtiments, généralement sous forme de suffixe. Par exemple : *Shopping Centre*.

DIRECT (22). Mots utilisés pour indiquer des directions, par exemple *Nord*.

MILE (20). Mots utilisés pour indiquer les adresses des points kilométriques.

ROAD (6). Mots et abréviations utilisés pour désigner les autoroutes et les routes. Par exemple : l'*Interstate* dans *Interstate 5*

RR (8). Mots et abréviations utilisés pour désigner les routes rurales. *RR*.

TYPE (2). Mots et abréviations utilisés pour désigner les types de rues. Par exemple : *ST* ou *AVE*.

UNITH (16). Mots et abréviations utilisés pour désigner les sous-adresses internes. Par exemple, *APT* ou *UNIT*.

Type de poste Jetons d'entrée

QUINT (28). Un nombre à 5 chiffres. Identifie un code postal

QUAD (29). Un numéro à 4 chiffres. Identifie le ZIP4.

PCH (27). Séquence de 3 caractères composée d'une lettre, d'un numéro et d'une lettre. Identifie un FSA, les 3 premiers caractères d'un code postal canadien.

PCT (26). Séquence de 3 caractères composée d'un numéro, d'une lettre et d'un chiffre. Identifie un LDU, les 3 derniers caractères d'un code postal canadien.

Stopwords

STOPWORDS se combinent avec WORDS. Dans les règles, une chaîne de plusieurs WORDs et STOPWORDs sera représentée par un seul mot-clé WORD.

STOPWORD (7). Un mot de faible importance lexicale, qui peut être omis dans l'analyse syntaxique. Par exemple : *THE*.

Jetons de sortie

Après le premier -1 (terminateur), suivent les jetons de sortie et leur ordre, suivis d'un terminateur `-1`. Les numéros des jetons de sortie correspondants sont énumérés dans stdaddr. Les éléments autorisés dépendent du type de règle. Les jetons de sortie valables pour chaque type de règle sont énumérés dans the section called "Types de règles et rangs".

Types de règles et rangs

La dernière partie de la règle est le type de règle qui est désigné par l'un des éléments suivants, suivi du rang de la règle. Les règles sont classées de 0 (la plus faible) à 17 (la plus élevée).

MACRO_C

(numéro du jeton = "**0**"). La classe de règles pour l'analyse des clauses MACRO telles que *PLACE STATE ZIP*

MACRO_C jetons de sortie ("output tokens") (excepté pour http://www.pagcgeo.org/docs/html/pagc-12.html#--r-typ--.

CITY (jeton numéro "10"). Exemple "Albany"

STATE (jeton numéro "11"). Exemple "NY"

NATION (numéro de jeton "12"). Cet attribut n'est pas utilisé dans la plupart des fichiers de référence. Exemple "USA"

POSTAL (nombre de jetons "13"). (éléments SADS "ZIP CODE" , "PLUS 4"). Cet attribut est utilisé à la fois pour le code postal américain et le code postal canadien.

MICRO_C

(numéro du jeton = "**1**"). La classe de règles pour l'analyse des clauses MICRO complètes (telles que House, street, sufdir, predir, pretyp, suftype, qualif) (c'est-à-dire ARC_C plus CIVIC_C). Ces règles ne sont pas utilisées dans la phase de construction.

MICRO_C jetons de sortie ("output tokens") (exceptés pour http://www.pagcgeo.org/docs/html/pagc-12.html#--r-typ--.

HOUSE est un texte (jeton numéro `1`) : Il s'agit du numéro d'une rue. Exemple *75* dans `75 State Street`.

predir est un texte (jeton numéro `2`) : NOM DE RUE PRÉDIRECTIONNEL tel que Nord, Sud, Est, Ouest, etc.

qual est un texte (jeton numéro `3`) : NOM DE RUE PRE-MODIFIER Exemple *OLD* dans `3715 OLD HIGHWAY 99`.

pretype est un texte (jeton numéro `4`) : TYPE DE PRÉFIXE DE RUE

street est un texte (jeton numéro `5`) : NOM DE RUE

suftype est un texte (jeton numéro 6) : STREET POST TYPE e.g. St, Ave, Cir. Un type de rue suivant le nom de la rue racine. Exemple *STREET* dans `75 State Street`.

sufdir est un texte (jeton numéro 7) : STREET POST-DIRECTIONAL Un modificateur directionnel qui suit le nom de la rue. Exemple *WEST* dans `3715 TENTH AVENUE WEST`.

ARC_C

(numéro de jeton = "**2**"). Classe de règles pour l'analyse des clauses MICRO, à l'exclusion de l'attribut HOUSE. En tant que telle, elle utilise le même ensemble de jetons de sortie que MICRO_C, à l'exception du jeton HOUSE.

CIVIC_C

(numéro du jeton = "**3**"). La classe de règles pour l'analyse de l'attribut HOUSE.

EXTRA_C

(numéro du jeton = "**4**"). La classe de règles pour l'analyse des attributs EXTRA - attributs exclus du géocodage. Ces règles ne sont pas utilisées dans la phase de construction.

EXTRA_C jetons de sortie ("output tokens") (excepté pour http://www.pagcgeo.org/docs/html/pagc-12.html#--r-typ--.

BLDNG (numéro de jeton 0) : Identifiants et types de bâtiments non analysés.

BOXH (numéro de jeton 14) : Le **BOX** dans `BOX 3B`

BOXT (token number 15) : La **3B** dans `BOX 3B`

RR (token number 8) : Le **RR** dans `RR 7`

UNITH (numéro de jeton 16) : L'**APT** dans `APT 3B`

UNITT (numéro de jeton 17) : La **3B** dans `APT 3B`

UNKNWN (numéro de jeton 9) : Une sortie non classée par ailleurs.

12.1.3.2 lex table

lex table — La table lex est utilisée pour classer les entrées alphanumériques et les associer (a) à des jetons d'entrée (voir the section called "Jetons d'entrée") et (b) à des représentations normalisées.

Description

La table lex (abréviation de lexicon) est utilisée pour classer les entrées alphanumériques et les associer à des the section called "Jetons d'entrée" et (b) à des représentations normalisées. Les éléments que vous trouverez dans ces tables sont `ONE` mappé à stdword : `1`.

La table lex comporte au moins les colonnes suivantes. Vous pouvez ajouter

id Clé primaire de la table

seq entier : définition du nombre ?

word texte : le mot en entrée

stdword texte : le mot de remplacement normalisé

token entier : le type de mot dont il s'agit. Il ne sera remplacé que s'il est utilisé dans ce contexte. Voir Tokens PAGC.

12.1.3.3 gaz table

gaz table — La table gaz est utilisée pour normaliser les noms de lieux et associer cette entrée avec (a) des tokens d'entrée (Voir the section called "Jetons d'entrée") et (b) des représentations normalisées.

Description

La table gaz (abréviation de gazeteer) est utilisée pour normaliser les noms de lieux et associer cette entrée à des the section called "Jetons d'entrée" et (b) à des représentations normalisées. Par exemple, si vous êtes aux États-Unis, vous pouvez charger ces tableaux avec les noms des États et les abréviations associées.

La table gaz comporte au moins les colonnes suivantes. Vous pouvez ajouter d'autres colonnes si vous le souhaitez pour vos propres besoins.

id Clé primaire de la table

seq entier : définition nombre ? - identificateur utilisé pour cette instance du mot

word texte : le mot en entrée

stdword texte : le mot de remplacement normalisé

token entier : le type de mot dont il s'agit. Il ne sera remplacé que s'il est utilisé dans ce contexte. Voir Tokens PAGC.

12.1.4 Fonctions Address Standardizer

12.1.4.1 debug_standardize_address

debug_standardize_address — Retourne une chaîne au format json avec les jetons d'entrée et les normalisations

Synopsis

text **debug_standardize_address**(text lextab, text gaztab, text rultab, text micro, text macro=NULL);

Description

Ceci est une fonction pour déboguer les règles de normalisation des adresses et les correspondances lex/gaz. Cette fonction retourne une chaîne au format json qui inclut les règles correspondantes, la correspondance des jetons d'entrées et la meilleure adresse normalisée selon stdaddr, en utilisant les tables lex table, gaz table et rules table, ainsi qu'une adresse.

Pour une adresse sur une seule ligne, utiliser `micro`

Pour une adresse sur deux lignes, utiliser `micro` pour la première ligne standard d'une adresse postale par exemple `house_num street`, et macro pour la deuxième ligne postale standard de l'adresse, par exemple `city, state postal_code country`.

Les éléments retournées dans le document json sont

input_tokens Pour chaque mot de l'adresse d'entrée, retourne la position du mot, la catégorisation du jeton correspondant au mot, et le mot standard correspondant. A noter que pour certains mots d'entrée, vous pouvez obtenir plusieurs enregistrements car certaines entrées peuvent correspondre à plusieurs catégories à la fois.

rules L'ensemble des règles correspondant à l'entrée, et le score de correspondance pour chacune. La première règle (avec le plus haut score) est celle utilisée pour la normalisation

stdaddr Les éléments de l'adresse normalisée stdaddr, qui seraient retournés par standardize_address

Disponibilité : 3.4.0

Cette méthode nécessite l'extension address_standardizer.

Exemples

Utilisation de l'extension address_standardizer_data_us

```
CREATE EXTENSION address_standardizer_data_us; -- only needs to be done once
```

Variante 1 : prend une adresse sur une seule ligne et retourne les jetons d'entrée

```
SELECT it->>'pos' AS position, it->>'word' AS word, it->>'stdword' AS standardized_word,
            it->>'token' AS token, it->>'token-code' AS token_code
    FROM jsonb(
            debug_standardize_address('us_lex',
                'us_gaz', 'us_rules', 'One Devonshire Place, PH 301, Boston, MA 02109')
                 ) AS s, jsonb_array_elements(s->'input_tokens') AS it;
```

```
position |    word    | standardized_word | token   | token_code
---------+------------+-------------------+---------+------------
 0       | ONE        | 1                 | NUMBER | 0
 0       | ONE        | 1                 | WORD    | 1
 1       | DEVONSHIRE | DEVONSHIRE        | WORD    | 1
 2       | PLACE      | PLACE             | TYPE    | 2
 3       | PH         | PATH              | TYPE    | 2
 3       | PH         | PENTHOUSE         | UNITT   | 17
 4       | 301        | 301               | NUMBER | 0
(7 rows)
```

Variante 2 : Adresse en deux parties et retourne la première règle de correspondance et son score

```
SELECT (s->'rules'->0->>'score')::numeric AS score, it->>'pos' AS position,
        it->>'input-word' AS word, it->>'input-token' AS input_token, it->>'mapped-word' AS ↩
             standardized_word,
            it->>'output-token' AS output_token
    FROM jsonb(
            debug_standardize_address('us_lex',
                'us_gaz', 'us_rules', 'One Devonshire Place, PH 301', 'Boston, MA 02109')
                 ) AS s, jsonb_array_elements(s->'rules'->0->'rule_tokens') AS it;
```

```
score    | position |    word    | input_token | standardized_word | output_token
---------+----------+------------+-------------+-------------------+--------------
 0.876250 | 0        | ONE        | NUMBER      | 1                 | HOUSE
 0.876250 | 1        | DEVONSHIRE | WORD        | DEVONSHIRE        | STREET
 0.876250 | 2        | PLACE      | TYPE        | PLACE             | SUFTYP
 0.876250 | 3        | PH         | UNITT       | PENTHOUSE         | UNITT
 0.876250 | 4        | 301        | NUMBER      | 301               | UNITT
(5 rows)
```

Voir aussi

stdaddr, rules table, lex table, gaz table, Pagc_Normalize_Address

12.1.4.2 parse_address

parse_address — Prend une adresse d'une ligne et la décompose en plusieurs parties

Synopsis

record **parse_address**(text address);

Description

Prend une adresse en entrée et renvoie un enregistrement composé des champs *num*, *street*, *street2*, *address1*, *city*, *state*, *zip*, *zipplus*, *country*.

Disponibilité : 2.2.0

Cette méthode nécessite l'extension address_standardizer.

Exemples

Une seule adresse

```
SELECT num, street, city, zip, zipplus
        FROM parse_address('1 Devonshire Place, Boston, MA 02109-1234') AS a;
```

```
num |      street      |  city  |  zip  | zipplus
-----+------------------+--------+-------+---------
 1   | Devonshire Place | Boston | 02109 | 1234
```

Table des adresses

```
-- basic table
CREATE TABLE places(addid serial PRIMARY KEY, address text);

INSERT INTO places(address)
VALUES ('529 Main Street, Boston MA, 02129'),
 ('77 Massachusetts Avenue, Cambridge, MA 02139'),
 ('25 Wizard of Oz, Walaford, KS 99912323'),
 ('26 Capen Street, Medford, MA'),
 ('124 Mount Auburn St, Cambridge, Massachusetts 02138'),
 ('950 Main Street, Worcester, MA 01610');

 -- parse the addresses
 -- if you want all fields you can use (a).*
SELECT addid, (a).num, (a).street, (a).city, (a).state, (a).zip, (a).zipplus
FROM (SELECT addid, parse_address(address) As a
 FROM places) AS p;
```

```
addid | num |        street        |   city    | state |  zip  | zipplus
-------+-----+----------------------+-----------+-------+-------+---------
     1 | 529 | Main Street          | Boston    | MA    | 02129 |
     2 | 77  | Massachusetts Avenue | Cambridge | MA    | 02139 |
     3 | 25  | Wizard of Oz         | Walaford  | KS    | 99912 | 323
     4 | 26  | Capen Street         | Medford   | MA    |       |
     5 | 124 | Mount Auburn St      | Cambridge | MA    | 02138 |
     6 | 950 | Main Street          | Worcester | MA    | 01610 |
(6 rows)
```

Voir aussi

12.1.4.3 standardize_address

standardize_address — Renvoie une forme stdaddr d'une adresse d'entrée en utilisant les tables lex, gaz et rule.

Synopsis

stdaddr **standardize_address**(text lextab, text gaztab, text rultab, text address);
stdaddr **standardize_address**(text lextab, text gaztab, text rultab, text micro, text macro);

Description

Renvoie une forme stdaddr d'une adresse d'entrée utilisant des lex table noms de tables, gaz table, et rules table noms de tables et une adresse.

Variante 1 : prend une adresse sur une seule ligne.

Variante 2 : prend une adresse en deux parties. Une `micro` constituée de la première ligne standard d'une adresse postale, par exemple `house_num street`, et une macro constituée de la deuxième ligne postale standard d'une adresse, par exemple `city, state postal_code country`.

Disponibilité : 2.2.0

Cette méthode nécessite l'extension address_standardizer.

Exemples

Utilisation de l'extension address_standardizer_data_us

```
CREATE EXTENSION address_standardizer_data_us; -- only needs to be done once
```

Variante 1 : adresse sur une seule ligne. Cela ne fonctionne pas bien avec les adresses non américaines

```
SELECT house_num, name, suftype, city, country, state, unit  FROM standardize_address(' ↩
    us_lex',
                            'us_gaz', 'us_rules', 'One Devonshire Place, PH 301, Boston, MA  ↩
                                02109');
```

```
house_num |     name     | suftype |  city   | country |     state      |      unit
----------+--------------+---------+---------+---------+----------------+-----------------
1         | DEVONSHIRE | PLACE   | BOSTON | USA     | MASSACHUSETTS | # PENTHOUSE 301
```

Utilisation des tables fournies avec tiger geocoder. Cet exemple ne fonctionne que si vous avez installé `postgis_tiger_geocoder`.

```
SELECT *  FROM standardize_address('tiger.pagc_lex',
          'tiger.pagc_gaz', 'tiger.pagc_rules', 'One Devonshire Place, PH 301, Boston, MA  ↩
             02109-1234');
```

Pour faciliter la lecture, nous utiliserons l'extension hstore CREATE EXTENSION hstore ; vous devez installer

```
SELECT (each(hstore(p))).*
 FROM standardize_address('tiger.pagc_lex', 'tiger.pagc_gaz',
   'tiger.pagc_rules', 'One Devonshire Place, PH 301, Boston, MA 02109') As p;
```

```
key     |      value
------------+-----------------
 box        |
 city       | BOSTON
 name       | DEVONSHIRE
 qual       |
 unit       | # PENTHOUSE 301
 extra      |
 state      | MA
 predir     |
 sufdir     |
 country    | USA
 pretype    |
 suftype    | PL
 building   |
 postcode   | 02109
 house_num  | 1
 ruralroute |
(16 rows)
```

Variante 2 : adresse en deux parties

```
SELECT (each(hstore(p))).*
 FROM standardize_address('tiger.pagc_lex', 'tiger.pagc_gaz',
   'tiger.pagc_rules', 'One Devonshire Place, PH 301', 'Boston, MA 02109, US') As p;
```

```
key     |      value
------------+-----------------
 box        |
 city       | BOSTON
 name       | DEVONSHIRE
 qual       |
 unit       | # PENTHOUSE 301
 extra      |
 state      | MA
 predir     |
 sufdir     |
 country    | USA
 pretype    |
 suftype    | PL
 building   |
 postcode   | 02109
 house_num  | 1
 ruralroute |
(16 rows)
```

Voir aussi

stdaddr, rules table, lex table, gaz table, Pagc_Normalize_Address

12.2 Géocodeur Tiger

Il existe quelques autres géocodeurs open source pour PostGIS qui, contrairement au géocodeur Tiger, présentent l'avantage de prendre en charge le géocodage multi-pays

- Nominatim utilise les données formatées de OpenStreetMap gazeteer. Il nécessite osm2pgsql pour charger les données, PostgreSQL 8.4+ et PostGIS 1.5+ pour fonctionner. Il est présenté sous la forme d'une interface webservice et semble conçu pour être appelé en tant que webservice. Tout comme le tiger geocoder, il possède à la fois un géocodeur et un géocodeur inversé. La documentation n'indique pas clairement s'il dispose d'une interface SQL pure comme le géocodeur tiger, ou si une grande partie de la logique est implémentée dans l'interface web.
- GIS Graphy utilise également PostGIS et, comme Nominatim, fonctionne avec des données OpenStreetMap (OSM). Il est livré avec un chargeur pour charger les données OSM et, comme Nominatim, il est capable de géocoder d'autres pays que les États-Unis. Tout comme Nominatim, il fonctionne comme un service web et s'appuie sur Java 1.5, Servlet apps, Solr. GisGraphy est multiplateforme et dispose également d'un géocodeur inversé parmi d'autres fonctionnalités intéressantes.

12.2.1 Drop_Indexes_Generate_Script

Drop_Indexes_Generate_Script — Génère un script qui supprime toutes les clés non primaires et les index non uniques sur le schéma tiger et le schéma spécifié par l'utilisateur. Le schéma par défaut est `tiger_data` si aucun schéma n'est spécifié.

Synopsis

text **Drop_Indexes_Generate_Script**(text param_schema=tiger_data);

Description

Génère un script qui supprime toutes les clés non primaires et les index non uniques sur le schéma tiger et le schéma spécifié par l'utilisateur. Le schéma par défaut est `tiger_data` si aucun schéma n'est spécifié.

Ceci est utile pour minimiser le gonflement de l'index qui peut perturber le planificateur de requêtes ou prendre de l'espace inutilement. A utiliser en combinaison avec Install_Missing_Indexes pour ajouter uniquement les index utilisés par le géocodeur.

Disponibilité : 2.0.0

Exemples

```
SELECT drop_indexes_generate_script() As actionsql;
actionsql
-----------------------------------------------------------
DROP INDEX tiger.idx_tiger_countysub_lookup_lower_name;
DROP INDEX tiger.idx_tiger_edges_countyfp;
DROP INDEX tiger.idx_tiger_faces_countyfp;
DROP INDEX tiger.tiger_place_the_geom_gist;
DROP INDEX tiger.tiger_edges_the_geom_gist;
DROP INDEX tiger.tiger_state_the_geom_gist;
DROP INDEX tiger.idx_tiger_addr_least_address;
DROP INDEX tiger.idx_tiger_addr_tlid;
DROP INDEX tiger.idx_tiger_addr_zip;
DROP INDEX tiger.idx_tiger_county_countyfp;
DROP INDEX tiger.idx_tiger_county_lookup_lower_name;
DROP INDEX tiger.idx_tiger_county_lookup_snd_name;
DROP INDEX tiger.idx_tiger_county_lower_name;
DROP INDEX tiger.idx_tiger_county_snd_name;
DROP INDEX tiger.idx_tiger_county_the_geom_gist;
DROP INDEX tiger.idx_tiger_countysub_lookup_snd_name;
DROP INDEX tiger.idx_tiger_cousub_countyfp;
DROP INDEX tiger.idx_tiger_cousub_cousubfp;
DROP INDEX tiger.idx_tiger_cousub_lower_name;
DROP INDEX tiger.idx_tiger_cousub_snd_name;
DROP INDEX tiger.idx_tiger_cousub_the_geom_gist;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_least_address;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_tlid;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_zip;
DROP INDEX tiger_data.idx_tiger_data_ma_county_countyfp;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lookup_lower_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lookup_snd_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lower_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_snd_name;
:
:
```

Voir aussi

Install_Missing_Indexes, Missing_Indexes_Generate_Script

12.2.2 Drop_Nation_Tables_Generate_Script

Drop_Nation_Tables_Generate_Script — Génère un script qui supprime toutes les tables du schéma spécifié qui commencent par `county_all`, `state_all` ou code d'état suivi de `county` ou `state`.

Synopsis

text **Drop_Nation_Tables_Generate_Script**(text param_schema=tiger_data);

Description

Génère un script qui supprime toutes les tables du schéma spécifié qui commencent par `county_all`, `state_all` ou code d'état suivi de `county` ou `state`. Ceci est nécessaire si vous mettez à jour les données de `tiger_2010` vers `tiger_2011`.

Disponibilité : 2.1.0

Exemples

```
SELECT drop_nation_tables_generate_script();
DROP TABLE tiger_data.county_all;
DROP TABLE tiger_data.county_all_lookup;
DROP TABLE tiger_data.state_all;
DROP TABLE tiger_data.ma_county;
DROP TABLE tiger_data.ma_state;
```

Voir aussi

Loader_Generate_Nation_Script

12.2.3 Drop_State_Tables_Generate_Script

Drop_State_Tables_Generate_Script — Génère un script qui supprime toutes les tables du schéma spécifié qui sont préfixées par l'abréviation de l'état. La valeur par défaut du schéma est `tiger_data` si aucun schéma n'est spécifié.

Synopsis

text **Drop_State_Tables_Generate_Script**(text param_state, text param_schema=tiger_data);

Description

Génère un script qui supprime toutes les tables du schéma spécifié qui sont préfixées par l'abréviation de l'état. Le schéma par défaut est `tiger_data` si aucun schéma n'est spécifié. Cette fonction est utile pour supprimer les tables d'un état juste avant de le recharger, au cas où quelque chose se serait mal passé lors du chargement précédent.

Disponibilité : 2.0.0

Exemples

```
SELECT drop_state_tables_generate_script('PA');
DROP TABLE tiger_data.pa_addr;
DROP TABLE tiger_data.pa_county;
DROP TABLE tiger_data.pa_county_lookup;
DROP TABLE tiger_data.pa_cousub;
DROP TABLE tiger_data.pa_edges;
DROP TABLE tiger_data.pa_faces;
DROP TABLE tiger_data.pa_featnames;
DROP TABLE tiger_data.pa_place;
DROP TABLE tiger_data.pa_state;
DROP TABLE tiger_data.pa_zip_lookup_base;
DROP TABLE tiger_data.pa_zip_state;
DROP TABLE tiger_data.pa_zip_state_loc;
```

Voir aussi

Loader_Generate_Script

12.2.4 Geocode

Geocode — Prend une adresse sous forme de chaîne de caractères (ou autre adresse normalisée) et produit un ensemble de lieux possibles comprenant une géométrie de point en NAD 83 long lat, une adresse normalisée pour chacun d'entre eux et l'évaluation. Plus la note est basse, plus la correspondance est probable. Les résultats sont triés par ordre décroissant. Il est possible d'indiquer en option le nombre maximum de résultats (10 par défaut) et la restriction de la région (NULL par défaut)

Synopsis

setof record **geocode**(varchar address, integer max_results=10, geometry restrict_region=NULL, norm_addy OUT addy, geometry OUT geomout, integer OUT rating);
setof record **geocode**(norm_addy in_addy, integer max_results=10, geometry restrict_region=NULL, norm_addy OUT addy, geometry OUT geomout, integer OUT rating);

Description

Prend une adresse sous forme de chaîne de caractères (ou une adresse déjà normalisée) et produit un ensemble de lieux possibles comprenant une géométrie de point en NAD 83 long lat, une `adresse_normalisée` (addy) pour chacun d'entre eux, et l'évaluation. Plus la note est basse, plus la correspondance est probable. Les résultats sont triés en fonction de la note la plus basse. Utilise les données du tiger (edges, faces, addr), la correspondance floue de PostgreSQL (soundex, levenshtein) et les fonctions d'interpolation de lignes de PostGIS pour interpoler l'adresse le long des bords du tigre. Plus la note est élevée, moins le géocodage a de chances d'être correct. Par défaut, le point géocodé est décalé de 10 mètres par rapport à la ligne centrale du côté (L/R) de la rue où se trouve l'adresse.

Amélioration : 2.0.0 pour prendre en charge les données structurées Tiger 2010 et révision de certaines logiques pour améliorer la vitesse et la précision du géocodage, et pour décaler le point de la ligne centrale vers le côté de la rue où se trouve l'adresse. Le nouveau paramètre `max_results` permet de spécifier le nombre de meilleurs résultats ou de renvoyer uniquement le meilleur résultat.

Exemples : De base

Les exemples de temps ci-dessous ont été calculés sur une machine Windows 7 à un processeur 3.0 GHZ avec 2 Go de mémoire vive, PostgreSQL 9.1rc1/PostGIS 2.0 avec toutes les données de MA, MN, CA, RI State Tiger.

Les correspondances exactes sont calculées plus rapidement (61 ms)

```
SELECT g.rating, ST_X(g.geomout) As lon, ST_Y(g.geomout) As lat,
    (addy).address As stno, (addy).streetname As street,
    (addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st,( ↩
        addy).zip
    FROM geocode('75 State Street, Boston MA 02109', 1) As g;
 rating |        lon         |      lat       | stno | street | styp |  city  | st |  zip
--------+--------------------+----------------+------+--------+------+--------+----+-------
      0 | -71.0557505845646 | 42.35897920691 |   75 | State  | St   | Boston | MA | 02109
```

Même si le code postal n'est pas transmis, le géocodeur peut deviner (cela a pris environ 122-150 ms)

```
SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktlonlat,
    (addy).address As stno, (addy).streetname As street,
    (addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st,( ↩
        addy).zip
    FROM geocode('226 Hanover Street, Boston, MA',1) As g;
 rating |         wktlonlat         | stno | street  | styp |  city  | st |  zip
```

```
--------+--------------------------+------+---------+------+--------+----+-------
      1 | POINT(-71.05528 42.36316) |  226 | Hanover | St   | Boston | MA | 02113
```

Peut gérer les fautes d'orthographe et propose plusieurs solutions possibles avec des évaluations, mais prend plus de temps (500 ms).

```
SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktlonlat,
    (addy).address As stno, (addy).streetname As street,
    (addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st,( ↩
       addy).zip
    FROM geocode('31 - 37 Stewart Street, Boston, MA 02116',1) As g;
 rating |         wktlonlat         | stno | street | styp |  city  | st |  zip
--------+---------------------------+------+--------+------+--------+----+-------
     70 | POINT(-71.06466 42.35114) |   31 | Stuart | St   | Boston | MA | 02116
```

Utilisation pour effectuer un géocodage par lots d'adresses. Le plus simple est de mettre `max_results=1`. Ne traiter que les adresses qui n'ont pas encore été géocodées (qui n'ont pas d'évaluation).

```
CREATE TABLE addresses_to_geocode(addid serial PRIMARY KEY, address text,
        lon numeric, lat numeric, new_address text, rating integer);

INSERT INTO addresses_to_geocode(address)
VALUES ('529 Main Street, Boston MA, 02129'),
 ('77 Massachusetts Avenue, Cambridge, MA 02139'),
 ('25 Wizard of Oz, Walaford, KS 99912323'),
 ('26 Capen Street, Medford, MA'),
 ('124 Mount Auburn St, Cambridge, Massachusetts 02138'),
 ('950 Main Street, Worcester, MA 01610');

-- only update the first 3 addresses (323-704 ms -  there are caching and shared memory  ↩
    effects so first geocode you do is always slower) --
-- for large numbers of addresses you don't want to update all at once
-- since the whole geocode must commit at once
-- For this example we rejoin with LEFT JOIN
-- and set to rating to -1 rating if no match
-- to ensure we don't regeocode a bad address
UPDATE addresses_to_geocode
  SET  (rating, new_address, lon, lat)
    = ( COALESCE(g.rating,-1), pprint_addy(g.addy),
       ST_X(g.geomout)::numeric(8,5), ST_Y(g.geomout)::numeric(8,5) )
FROM (SELECT addid, address
    FROM addresses_to_geocode
    WHERE rating IS NULL ORDER BY addid LIMIT 3) As a
    LEFT JOIN LATERAL geocode(a.address,1) As g ON true
WHERE a.addid = addresses_to_geocode.addid;

result
-----
Query returned successfully: 3 rows affected, 480 ms execution time.

SELECT * FROM addresses_to_geocode WHERE rating is not null;

 addid |                   address                    |    lon    |   lat    |  ↩
                  new_address                | rating
-------+----------------------------------------------+-----------+----------+-----------------------

     1 | 529 Main Street, Boston MA, 02129            | -71.07177 | 42.38357 | 529 Main St, ↩
         Boston, MA 02129             |      0
     2 | 77 Massachusetts Avenue, Cambridge, MA 02139 | -71.09396 | 42.35961 | 77  ↩
        Massachusetts Ave, Cambridge, MA 02139 |      0
     3 | 25 Wizard of Oz, Walaford, KS 99912323       | -97.92913 | 38.12717 | Willowbrook, ↩
         KS 67502                     |    108
```

```
(3 rows)
```

Exemples : Utilisation du filtre de géométrie

```
SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktlonlat,
    (addy).address As stno, (addy).streetname As street,
    (addy).streettypeabbrev As styp,
    (addy).location As city, (addy).stateabbrev As st,(addy).zip
  FROM geocode('100 Federal Street, MA',
        3,
        (SELECT ST_Union(the_geom)
            FROM place WHERE statefp = '25' AND name = 'Lynn')::geometry
        ) As g;

 rating |         wktlonlat         | stno | street  | styp | city | st |  zip
--------+---------------------------+------+---------+------+------+----+-------
      7 | POINT(-70.96796 42.4659)  |  100 | Federal | St   | Lynn | MA | 01905
     16 | POINT(-70.96786 42.46853) | NULL | Federal | St   | Lynn | MA | 01905
(2 rows)

Time: 622.939 ms
```

Voir aussi

Normalize_Address, Pprint_Addy, ST_AsText, ST_SnapToGrid, ST_X, ST_Y

12.2.5 Geocode_Intersection

Geocode_Intersection — Prend 2 rues qui s'intersectent et un état, une ville, un code postal, et produit un ensemble d'emplacements possibles sur la première rue croisée qui est à l'intersection, comprend également un "geomout" comme emplacement du point en NAD 83 long lat, une `adresse_normalisée` (addy) pour chaque emplacement, et l'évaluation. Plus la note est basse, plus la correspondance est probable. Les résultats sont triés en fonction de la note la plus basse. Il est possible d'indiquer le nombre maximum de résultats, la valeur par défaut étant de 10. Utilise les données Tiger (edges, faces, addr), la correspondance floue de PostgreSQL (soundex, levenshtein).

Synopsis

setof record **geocode_intersection**(text roadway1, text roadway2, text in_state, text in_city, text in_zip, integer max_results=10, norm_addy OUT addy, geometry OUT geomout, integer OUT rating);

Description

Prend 2 rues qui s'intersectent et un état, une ville, un code postal, et produit un ensemble d'emplacements possibles sur la première rue croisée qui est à l'intersection, comprend également une géométrie de point dans NAD 83 long lat, une adresse normalisée pour chaque emplacement, et la cote. Plus la note est basse, plus la correspondance est probable. Les résultats sont triés par ordre décroissant. Il est possible d'indiquer le nombre maximum de résultats, la valeur par défaut étant de 10. Retourne `adresse_normalisée` (addy) pour chaque, "geomout" comme emplacement du point en nad 83 long lat, et la note. Plus la note est basse, plus la correspondance est probable. Les résultats sont triés en fonction de la note la plus basse. Utilise les données Tiger (bords, visages, adresses), la correspondance floue de PostgreSQL (soundex, levenshtein)

Disponibilité : 2.0.0

Exemples : De base

Les exemples de temps ci-dessous sont basés sur une machine Windows 7 à un processeur 3.0 GHZ avec 2 Go de mémoire vive, avec PostgreSQL 9.0/PostGIS 1.5 et toutes les données de MA State Tiger chargées. Actuellement un peu lent (3000 ms)

Test sur Windows 2003 64-bit 8GB sur PostGIS 2.0 PostgreSQL 64-bit Tiger 2011 données chargées -- (41ms)

```
SELECT pprint_addy(addy), st_astext(geomout),rating
            FROM geocode_intersection( 'Haverford St','Germania St', 'MA', 'Boston',  ↩
                '02130',1);
           pprint_addy            |         st_astext          | rating
----------------------------------+----------------------------+--------
98 Haverford St, Boston, MA 02130 | POINT(-71.101375 42.31376) |      0
```

Même si le code postal n'est pas transmis, le géocodeur peut deviner (cela a pris environ 3500 ms sur la boîte Windows 7), sur la boîte Windows 2003 64-bit 741 ms

```
SELECT pprint_addy(addy), st_astext(geomout),rating
                FROM geocode_intersection('Weld', 'School', 'MA', 'Boston');
          pprint_addy          |        st_astext         | rating
-------------------------------+--------------------------+--------
 98 Weld Ave, Boston, MA 02119 | POINT(-71.099 42.314234) |      3
 99 Weld Ave, Boston, MA 02119 | POINT(-71.099 42.314234) |      3
```

Voir aussi

Geocode, Pprint_Addy, ST_AsText

12.2.6 Get_Geocode_Setting

Get_Geocode_Setting — Renvoie la valeur d'un paramètre spécifique stocké dans la table tiger.geocode_settings.

Synopsis

text **Get_Geocode_Setting**(text setting_name);

Description

Renvoie la valeur d'un paramètre spécifique stocké dans la table tiger.geocode_settings. Les paramètres permettent d'activer le débogage des fonctions. Plus tard, il sera possible de contrôler la notation à l'aide des paramètres. La liste actuelle des paramètres est la suivante :

```
name              | setting |  unit   | category  |  ↩
                                                                  short_desc
--------------------------------+---------+---------+-----------+-----------------------------------

 debug_geocode_address          | false   | boolean | debug     | outputs debug information ↩
      in notice log such as queries when geocode_address is called if true
 debug_geocode_intersection     | false   | boolean | debug     | outputs debug information ↩
      in notice log such as queries when geocode_intersection is called if true
 debug_normalize_address        | false   | boolean | debug     | outputs debug information ↩
      in notice log such as queries and intermediate expressions when normalize_address is  ↩
     called if true
 debug_reverse_geocode          | false   | boolean | debug     | if true, outputs debug  ↩
     information in notice log such as queries and intermediate expressions when  ↩
     reverse_geocode
```

```
reverse_geocode_numbered_roads | 0       | integer | rating    | For state and county  ↩
    highways, 0 - no preference in name,
                                                                  1 - prefer the numbered  ↩
                                                                      highway name, 2 -  ↩
                                                                      prefer local state/ ↩
                                                                      county name
use_pagc_address_parser        | false   | boolean | normalize | If set to true, will try  ↩
    to use the address_standardizer extension (via pagc_normalize_address)
                                                                 instead of tiger  ↩
                                                                    normalize_address built ↩
                                                                     one
```

Modifié : 2.2.0 : les paramètres par défaut sont désormais conservés dans une table appelée geocode_settings_default. Les paramètres personnalisés sont dans geocode_settings et ne contiennent que ceux qui ont été définis par l'utilisateur.

Disponibilité : 2.1.0

Exemple de paramétrage du débogage de retour

```
SELECT get_geocode_setting('debug_geocode_address) As result;
result
---------
false
```

Voir aussi

Set_Geocode_Setting

12.2.7 Get_Tract

Get_Tract — Renvoie le secteur de recensement ou le champ de la table des secteurs où se trouve la géométrie. Par défaut, le nom court du secteur est renvoyé.

Synopsis

text **get_tract**(geometry loc_geom, text output_field=name);

Description

Une géométrie donnée renvoie la localisation du secteur de recensement de cette géométrie. Le NAD 83 long lat est supposé si aucun système de référence spatiale n'est spécifié.

Note
Cette fonction utilise le recensement `tract` qui n'est pas chargé par défaut. Si vous avez déjà chargé votre table d'état, vous pouvez charger tract ainsi que bg, et tabblock en utilisant le script Loader_Generate_Census_Script.
Si vous n'avez pas encore chargé les données de votre État et que vous souhaitez charger ces tableaux supplémentaires, procédez comme suit

```
UPDATE tiger.loader_lookuptables SET load = true WHERE load = false AND lookup_name  ↩
    IN('tract', 'bg', 'tabblock');
```

alors ils seront inclus par le Loader_Generate_Script.

Disponibilité : 2.0.0

Exemples : De base

```
SELECT get_tract(ST_Point(-71.101375, 42.31376) ) As tract_name;
tract_name
---------
1203.01
```

```
--this one returns the tiger geoid
SELECT get_tract(ST_Point(-71.101375, 42.31376), 'tract_id' ) As tract_id;
tract_id
---------
25025120301
```

Voir aussi

Geocode>

12.2.8 Install_Missing_Indexes

Install_Missing_Indexes — Recherche toutes les tables dont les colonnes clés sont utilisées dans les jointures du géocodeur et les conditions de filtrage qui n'ont pas d'index utilisés sur ces colonnes et les ajoute.

Synopsis

boolean **Install_Missing_Indexes**();

Description

Recherche toutes les tables dans les schémas `tiger` et `tiger_data` avec des colonnes clés utilisées dans les jointures et les filtres du géocodeur qui n'ont pas d'index sur ces colonnes et produira le DDL SQL pour définir l'index pour ces tables et exécutera ensuite le script généré. Il s'agit d'une fonction d'aide qui ajoute de nouveaux index nécessaires pour rendre les requêtes plus rapides et qui peuvent avoir été manquants au cours du processus de chargement. Cette fonction accompagne Missing_Indexes_Generate_Script qui, en plus de générer le script de création d'index, l'exécute également. Elle est appelée dans le cadre du script de mise à niveau `update_geocode.sql`.

Disponibilité : 2.0.0

Exemples

```
SELECT install_missing_indexes();
         install_missing_indexes
-------------------------
 t
```

Voir aussi

Loader_Generate_Script, Missing_Indexes_Generate_Script

12.2.9 Loader_Generate_Census_Script

Loader_Generate_Census_Script — Génère un script shell pour la plate-forme spécifiée et les états spécifiés qui téléchargera les tables de données Tiger census state tract, bg et tabblocks, les structurera et les chargera dans le schéma `tiger_data`. Chaque script d'état est renvoyé sous la forme d'un enregistrement distinct.

Synopsis

setof text **loader_generate_census_script**(text[] param_states, text os);

Description

Génère un script shell pour la plate-forme spécifiée et les états spécifiés, qui téléchargera les données de recensement Tiger `tract`, les tables de données `bg` et `tabblocks`, les étapes et le chargement dans le schéma `tiger_data`. Chaque script d'état est renvoyé sous la forme d'un enregistrement distinct.

Il utilise unzip sous Linux (7-zip sous Windows par défaut) et wget pour effectuer le téléchargement. Il utilise Section 4.7.2 pour charger les données. Notez que la plus petite unité qu'il traite est un état entier. Il ne traitera que les fichiers des dossiers staging et temp.

Il utilise les tables de contrôle suivantes pour contrôler le processus et les différentes variations syntaxiques de l'interpréteur de commandes du système d'exploitation.

1. `loader_variables` garde la trace de diverses variables telles que le site de recensement, l'année, les données et les schémas d'étape
2. `loader_platform` profils des différentes plates-formes et de l'emplacement des différents exécutables. Livré avec windows et linux. D'autres peuvent être ajoutés.
3. `loader_lookuptables` chaque enregistrement définit un type de table (state, county), s'il faut y traiter les enregistrements et comment les charger. Définit les étapes d'importation des données, de structuration des données, d'ajout et de suppression de colonnes, d'index et de contraintes pour chaque table. Chaque table est préfixée par l'état et hérite d'une table du schéma tigre. Par exemple, crée `tiger_data.ma_faces` qui hérite de `tiger.faces`

Disponibilité : 2.0.0

Note
Loader_Generate_Script inclut cette logique, mais si vous avez installé tiger geocoder avant PostGIS 2.0.0 alpha5, vous devrez l'exécuter sur les états que vous avez déjà faits pour obtenir ces tables supplémentaires.

Exemples

Générer un script pour charger des données pour des états sélectionnés dans le format d'un script shell Windows.

```
SELECT loader_generate_census_script(ARRAY['MA'], 'windows');
-- result --
set STATEDIR="\gisdata\www2.census.gov\geo\pvs\tiger2010st\25_Massachusetts"
set TMPDIR=\gisdata\temp\
set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"
set WGETTOOL="C:\wget\wget.exe"
set PGBIN=C:\projects\pg\pg91win\bin\
set PGPORT=5432
set PGHOST=localhost
set PGUSER=postgres
set PGPASSWORD=yourpasswordhere
set PGDATABASE=tiger_postgis20
set PSQL="%PGBIN%psql"
set SHP2PGSQL="%PGBIN%shp2pgsql"
cd \gisdata

%WGETTOOL% http://www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts/25/ --no-parent -- ↩
    relative --accept=*bg10.zip,*tract10.zip,*tabblock10.zip --mirror --reject=html
del %TMPDIR%\*.* /Q
```

```
%PSQL% -c "DROP SCHEMA tiger_staging CASCADE;"
%PSQL% -c "CREATE SCHEMA tiger_staging;"
cd %STATEDIR%
for /r %%z in (*.zip) do %UNZIPTOOL% e %%z  -o%TMPDIR%
cd %TMPDIR%
%PSQL% -c "CREATE TABLE tiger_data.MA_tract(CONSTRAINT pk_MA_tract PRIMARY KEY (tract_id) ) ↩
     INHERITS(tiger.tract); "
%SHP2PGSQL% -c -s 4269 -g the_geom   -W "latin1" tl_2010_25_tract10.dbf tiger_staging. ↩
    ma_tract10 | %PSQL%
%PSQL% -c "ALTER TABLE tiger_staging.MA_tract10 RENAME geoid10 TO tract_id;  SELECT  ↩
    loader_load_staged_data(lower('MA_tract10'), lower('MA_tract')); "
%PSQL% -c "CREATE INDEX tiger_data_MA_tract_the_geom_gist ON tiger_data.MA_tract USING gist ↩
    (the_geom);"
%PSQL% -c "VACUUM ANALYZE tiger_data.MA_tract;"
%PSQL% -c "ALTER TABLE tiger_data.MA_tract ADD CONSTRAINT chk_statefp CHECK (statefp =  ↩
    '25');"
:
```

Générer le script sh

```
STATEDIR="/gisdata/www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts"
TMPDIR="/gisdata/temp/"
UNZIPTOOL=unzip
WGETTOOL="/usr/bin/wget"
export PGBIN=/usr/pgsql-9.0/bin
export PGPORT=5432
export PGHOST=localhost
export PGUSER=postgres
export PGPASSWORD=yourpasswordhere
export PGDATABASE=geocoder
PSQL=${PGBIN}/psql
SHP2PGSQL=${PGBIN}/shp2pgsql
cd /gisdata

wget http://www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts/25/ --no-parent --relative ↩
     --accept=*bg10.zip,*tract10.zip,*tabblock10.zip --mirror --reject=html
rm -f ${TMPDIR}/*.*
${PSQL} -c "DROP SCHEMA tiger_staging CASCADE;"
${PSQL} -c "CREATE SCHEMA tiger_staging;"
cd $STATEDIR
for z in *.zip; do $UNZIPTOOL -o -d $TMPDIR $z; done
:
:
```

Voir aussi

Loader_Generate_Script

12.2.10 Loader_Generate_Script

Loader_Generate_Script — Génère un script shell pour la plateforme spécifiée et les états spécifiés qui téléchargera les données Tiger, les structurera et les chargera dans le schéma `tiger_data`. Chaque script d'état est renvoyé sous la forme d'un enregistrement séparé. La dernière version prend en charge les modifications structurelles de Tiger 2010 et charge également les tableaux de secteurs de recensement, de groupes d'îlots et d'îlots.

Synopsis

setof text **loader_generate_script**(text[] param_states, text os);

Description

Génère un script shell pour la plateforme spécifiée et les états spécifiés qui téléchargera les données Tiger, les structurera et les chargera dans le schéma `tiger_data`. Chaque script d'état est renvoyé sous la forme d'un enregistrement séparé.

Il utilise unzip sous Linux (7-zip sous Windows par défaut) et wget pour effectuer le téléchargement. Il utilise Section 4.7.2 pour charger les données. Notez que la plus petite unité qu'il traite est un état entier, mais vous pouvez l'écraser en téléchargeant les fichiers vous-même. Il ne traitera que les fichiers des dossiers staging et temp.

Il utilise les tables de contrôle suivantes pour contrôler le processus et les différentes variations syntaxiques de l'interpréteur de commandes du système d'exploitation.

1. `loader_variables` garde la trace de diverses variables telles que le site de recensement, l'année, les données et les schémas d'étape
2. `loader_platform` profils des différentes plates-formes et de l'emplacement des différents exécutables. Livré avec windows et linux. D'autres peuvent être ajoutés.
3. `loader_lookuptables` chaque enregistrement définit un type de table (state, county), s'il faut y traiter les enregistrements et comment les charger. Définit les étapes d'importation des données, de structuration des données, d'ajout et de suppression de colonnes, d'index et de contraintes pour chaque table. Chaque table est préfixée par l'état et hérite d'une table du schéma tigre. Par exemple, crée `tiger_data.ma_faces` qui hérite de `tiger.faces`

Disponibilité : 2.0.0 pour prendre en charge les données structurées de Tiger 2010 et charger les tableaux de secteurs de recensement (tract), de groupes d'îlots (bg) et d'îlots (tabblocks).

Note
Si vous utilisez pgAdmin 3, sachez que par défaut pgAdmin 3 tronque les textes longs. Pour corriger cela, modifiez *Fichier -> Options -> Outil de requête -> Editeur de requête -> Caractères max. par colonne* à plus de 50000 caractères.

Exemples

En utilisant psql où gistest est votre base de données et `/gisdata/data_load.sh` est le fichier à créer avec les commandes shell à exécuter.

```
psql -U postgres -h localhost -d gistest -A -t \
 -c "SELECT Loader_Generate_Script(ARRAY['MA'], 'gistest')" > /gisdata/data_load.sh;
```

Générer un script pour charger des données pour 2 états dans le format d'un script shell Windows.

```
SELECT loader_generate_script(ARRAY['MA','RI'], 'windows') AS result;
-- result --
set TMPDIR=\gisdata\temp\
set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"
set WGETTOOL="C:\wget\wget.exe"
set PGBIN=C:\Program Files\PostgreSQL\9.4\bin\
set PGPORT=5432
set PGHOST=localhost
set PGUSER=postgres
set PGPASSWORD=yourpasswordhere
set PGDATABASE=geocoder
set PSQL="%PGBIN%psql"
set SHP2PGSQL="%PGBIN%shp2pgsql"
cd \gisdata

cd \gisdata
%WGETTOOL% ftp://ftp2.census.gov/geo/tiger/TIGER2015/PLACE/tl_*_25_* --no-parent --relative ↩
     --recursive --level=2 --accept=zip --mirror --reject=html
```

```
cd \gisdata/ftp2.census.gov/geo/tiger/TIGER2015/PLACE
:
:
```

Générer le script sh

```
SELECT loader_generate_script(ARRAY['MA','RI'], 'sh') AS result;
-- result --
TMPDIR="/gisdata/temp/"
UNZIPTOOL=unzip
WGETTOOL="/usr/bin/wget"
export PGBIN=/usr/lib/postgresql/9.4/bin
-- variables used by psql: https://www.postgresql.org/docs/current/static/libpq-envars.html
export PGPORT=5432
export PGHOST=localhost
export PGUSER=postgres
export PGPASSWORD=yourpasswordhere
export PGDATABASE=geocoder
PSQL=${PGBIN}/psql
SHP2PGSQL=${PGBIN}/shp2pgsql
cd /gisdata

cd /gisdata
wget ftp://ftp2.census.gov/geo/tiger/TIGER2015/PLACE/tl_*_25_* --no-parent --relative -- ↩
    recursive --level=2 --accept=zip --mirror --reject=html
cd /gisdata/ftp2.census.gov/geo/tiger/TIGER2015/PLACE
rm -f ${TMPDIR}/*.*
:
:
```

Voir aussi

Section 2.4.1, Loader_Generate_Nation_Script, Drop_State_Tables_Generate_Script

12.2.11 Loader_Generate_Nation_Script

Loader_Generate_Nation_Script — Génère un script shell pour la plate-forme spécifiée qui charge les données dans les tables county et state.

Synopsis

text **loader_generate_nation_script**(text os);

Description

Génère un script shell pour la plate-forme spécifiée qui charge les tables `county_all`, `county_all_lookup`, `state_all` dans le schéma `tiger_data`. Elles héritent respectivement des tables `county`, `county_lookup`, `state` dans le schéma `tiger`.

Il utilise unzip sous Linux (7-zip sous Windows par défaut) et wget pour effectuer le téléchargement. Il utilise Section 4.7.2 pour charger les données.

Il utilise les tables de contrôle suivantes `tiger.loader_platform`, `tiger.loader_variables` et `tiger.loader_lookup` pour contrôler le processus et les différentes variations syntaxiques de l'interpréteur de commandes du système d'exploitation.

1. `loader_variables` garde la trace de diverses variables telles que le site de recensement, l'année, les données et les schémas d'étape

2. `loader_platform` profils des différentes plates-formes et de l'emplacement des différents exécutables. Fourni avec windows et linux/unix. D'autres peuvent être ajoutés.

3. `loader_lookuptables` chaque enregistrement définit un type de table (state, county), s'il faut y traiter les enregistrements et comment les charger. Définit les étapes d'importation des données, de structuration des données, d'ajout et de suppression de colonnes, d'index et de contraintes pour chaque table. Chaque table est préfixée par l'état et hérite d'une table du schéma tigre. Par exemple, crée `tiger_data.ma_faces` qui hérite de `tiger.faces`

Amélioré : 2.4.1 L'étape de chargement de la zone de tabulation du code postal 5 (zcta5) a été corrigée et, lorsqu'elle est activée, les données zcta5 sont chargées sous la forme d'une table unique appelée zcta5_all dans le cadre du chargement du script de la nation.

Disponibilité : 2.1.0

Note
Si vous souhaitez que la zone de tabulation du code postal 5 (zcta5) soit incluse dans le chargement de votre script nation, procédez comme suit :

```
UPDATE tiger.loader_lookuptables SET load = true WHERE table_name = 'zcta510';
```

Note
Si vous utilisiez la version `tiger_2010` et que vous souhaitez recharger en tant qu'état avec des données tiger plus récentes, vous devrez, pour le tout premier chargement, générer et exécuter des instructions de dépôt Drop_Nation_Tables_Generate_Script avant d'exécuter ce script.

Exemples

Générer un script pour charger les données de la nation sur Windows.

```
SELECT loader_generate_nation_script('windows');
```

Générer un script pour charger les données pour les systèmes Linux/Unix.

```
SELECT loader_generate_nation_script('sh');
```

Voir aussi

Loader_Generate_Script, Drop_Nation_Tables_Generate_Script

12.2.12 Missing_Indexes_Generate_Script

Missing_Indexes_Generate_Script — Recherche toutes les tables dont les colonnes clés sont utilisées dans les jointures du géocodeur et qui n'ont pas d'index sur ces colonnes, et génère le DDL SQL permettant de définir l'index pour ces tables.

Synopsis

text **Missing_Indexes_Generate_Script**();

Description

Trouve toutes les tables dans les schémas `tiger` et `tiger_data` avec des colonnes clés utilisées dans les jointures du géocodeur qui n'ont pas d'index sur ces colonnes et produira le DDL SQL pour définir l'index pour ces tables. Il s'agit d'une fonction d'aide qui ajoute de nouveaux index nécessaires pour rendre les requêtes plus rapides et qui peuvent avoir été manquants au cours du processus de chargement. Au fur et à mesure de l'amélioration du géocodeur, cette fonction sera mise à jour pour prendre en compte les nouveaux index utilisés. Si cette fonction ne produit rien, cela signifie que toutes vos tables ont ce que nous pensons être les index clés déjà en place.

Disponibilité : 2.0.0

Exemples

```
SELECT missing_indexes_generate_script();
-- output: This was run on a database that was created before many corrections were made to ↩
     the loading script ---
CREATE INDEX idx_tiger_county_countyfp ON tiger.county USING btree(countyfp);
CREATE INDEX idx_tiger_cousub_countyfp ON tiger.cousub USING btree(countyfp);
CREATE INDEX idx_tiger_edges_tfidr ON tiger.edges USING btree(tfidr);
CREATE INDEX idx_tiger_edges_tfidl ON tiger.edges USING btree(tfidl);
CREATE INDEX idx_tiger_zip_lookup_all_zip ON tiger.zip_lookup_all USING btree(zip);
CREATE INDEX idx_tiger_data_ma_county_countyfp ON tiger_data.ma_county USING btree(countyfp ↩
    );
CREATE INDEX idx_tiger_data_ma_cousub_countyfp ON tiger_data.ma_cousub USING btree(countyfp ↩
    );
CREATE INDEX idx_tiger_data_ma_edges_countyfp ON tiger_data.ma_edges USING btree(countyfp);
CREATE INDEX idx_tiger_data_ma_faces_countyfp ON tiger_data.ma_faces USING btree(countyfp);
```

Voir aussi

Loader_Generate_Script, Install_Missing_Indexes

12.2.13 Normalize_Address

Normalize_Address — Étant donné une adresse textuelle, cette fonction renvoie un type composite `norm_addy` qui contient le suffixe de la route, le préfixe et le type normalisé, la rue, le nom de la rue, etc. divisés en champs distincts. Cette fonction fonctionne uniquement avec les données de recherche fournies avec le géocodeur tiger (pas besoin pour les données de recensement tiger).

Synopsis

norm_addy **normalize_address**(varchar in_address);

Description

Étant donné une adresse textuelle, cette fonction renvoie un type composite `norm_addy` qui contient le suffixe de la route, le préfixe et le type normalisé, la rue, le nom de la rue, etc. divisés en champs distincts. Il s'agit de la première étape du processus de géocodage visant à normaliser toutes les adresses sous forme postale. Aucune autre donnée n'est nécessaire en dehors de celles fournies avec le géocodeur.

Cette fonction utilise simplement les différentes tables de recherche direction/état/suffixe préchargées avec le géocodeur tiger et situées dans le schéma `tiger`, de sorte qu'il n'est pas nécessaire de télécharger les données de recensement tiger ou d'autres données supplémentaires pour l'utiliser. Il se peut que vous ayez besoin d'ajouter des abréviations ou des noms alternatifs aux différentes tables de recherche du schéma `tiger`.

Il utilise diverses tables de contrôle situées dans le schéma `tiger` pour normaliser l'adresse d'entrée.

Les champs de l'objet de type `norm_addy` renvoyés par cette fonction dans cet ordre où () indique un champ requis par le géocodeur, [] indique un champ optionnel :

(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip] [parsed] [zip4] [address_alphanumeric]

Amélioré : 2.4.0 L'objet norm_addy comprend des champs supplémentaires zip4 et address_alphanumeric.

1. `address` est un nombre entier : Le numéro de la rue
2. `predirAbbrev` est varchar : Préfixe directionnel de la route tel que N, S, E, W, etc. Ces préfixes sont contrôlés à l'aide de la table `direction_lookup`.
3. `streetName` varchar
4. `streetTypeAbbrev` varchar version abrégée du type de rue : par exemple St, Ave, Cir. Ceux-ci sont contrôlés à l'aide de la table `street_type_lookup`.
5. `postdirAbbrev` varchar abrégé directionnel suffixe de la route N, S, E, W etc. Elles sont contrôlées à l'aide de la table `direction_lookup`.
6. `internal` varchar adresse interne telle qu'un numéro d'appartement ou de suite.
7. `location` varchar généralement une ville ou une province.
8. `stateAbbrev` varchar deux caractères État américain, par exemple MA, NY, MI. Ceux-ci sont contrôlés par la table `state_lookup`.
9. `zip` varchar Code postal à 5 chiffres, par exemple 02109.
10. `parsed` booléen - indique si l'adresse a été formée à partir du processus de normalisation. La fonction normalize_address fixe cette valeur à true avant de renvoyer l'adresse.
11. `zip4` 4 derniers chiffres d'un code postal à 9 chiffres. Disponibilité : PostGIS 2.4.0.
12. `address_alphanumeric` Numéro de rue complet, même s'il contient des caractères alpha comme 17R. Il est préférable d'utiliser la fonction Pagc_Normalize_Address. Disponibilité : PostGIS 2.4.0.

Exemples

Sortie des champs de sélection. Utilisez Pprint_Addy si vous souhaitez une sortie textuelle.

```
SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev
 FROM (SELECT address, normalize_address(address) As na
        FROM addresses_to_geocode) As g;

                        orig                          |  streetname   | streettypeabbrev
------------------------------------------------------+---------------+------------------
 28 Capen Street, Medford, MA                         | Capen         | St
 124 Mount Auburn St, Cambridge, Massachusetts 02138  | Mount Auburn  | St
 950 Main Street, Worcester, MA 01610                 | Main          | St
 529 Main Street, Boston MA, 02129                    | Main          | St
 77 Massachusetts Avenue, Cambridge, MA 02139         | Massachusetts | Ave
 25 Wizard of Oz, Walaford, KS 99912323               | Wizard of Oz  |
```

Voir aussi

Geocode, Pprint_Addy

12.2.14 Pagc_Normalize_Address

Pagc_Normalize_Address — Étant donné une adresse textuelle, cette fonction renvoie un type composite `norm_addy` qui contient le suffixe de la route, le préfixe et le type normalisé, la rue, le nom de la rue, etc. divisés en champs distincts. Cette fonction fonctionne uniquement avec les données de recherche fournies avec le géocodeur tiger (pas besoin pour les données de recensement tiger). Nécessite l'extension address_standardizer.

Synopsis

norm_addy **pagc_normalize_address**(varchar in_address);

Description

Étant donné une adresse textuelle, cette fonction renvoie un type composite `norm_addy` qui contient le suffixe de la route, le préfixe et le type normalisé, la rue, le nom de la rue, etc. divisés en champs distincts. Il s'agit de la première étape du processus de géocodage visant à normaliser toutes les adresses sous forme postale. Aucune autre donnée n'est nécessaire en dehors de celles fournies avec le géocodeur.

Cette fonction utilise simplement les diverses tables de recherche pagc_* préchargées avec le géocodeur tiger et situées dans le schéma `tiger`, de sorte qu'il n'est pas nécessaire de télécharger les données de recensement tiger ou d'autres données supplémentaires pour l'utiliser. Il se peut que vous ayez besoin d'ajouter des abréviations ou des noms alternatifs aux différentes tables de recherche du schéma `tiger`.

Il utilise diverses tables de contrôle situées dans le schéma `tiger` pour normaliser l'adresse d'entrée.

Les champs de l'objet de type `norm_addy` renvoyés par cette fonction dans cet ordre où () indique un champ requis par le géocodeur, [] indique un champ optionnel :

Il existe de légères variations dans la casse et le formatage des Normalize_Address.

Disponibilité : 2.1.0

Cette méthode nécessite l'extension address_standardizer.

(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip]

Le standardaddr natif de l'extension address_standardizer est pour l'instant un peu plus riche que norm_addy puisqu'il est conçu pour prendre en charge les adresses internationales (y compris le pays). Les champs équivalents au standardaddr sont :

house_num,predir, name, suftype, sufdir, unit, city, state, postcode

Amélioré : 2.4.0 L'objet norm_addy comprend des champs supplémentaires zip4 et address_alphanumeric.

1. `address` est un nombre entier : Le numéro de la rue
2. `predirAbbrev` est varchar : Préfixe directionnel de la route tel que N, S, E, W, etc. Ces préfixes sont contrôlés à l'aide de la table `direction_lookup`.
3. `streetName` varchar
4. `streetTypeAbbrev` varchar version abrégée du type de rue : par exemple St, Ave, Cir. Ceux-ci sont contrôlés à l'aide de la table `street_type_lookup`.
5. `postdirAbbrev` varchar abrégé directionnel suffixe de la route N, S, E, W etc. Elles sont contrôlées à l'aide de la table `direction_lookup`.
6. `internal` varchar adresse interne telle qu'un numéro d'appartement ou de suite.
7. `location` varchar généralement une ville ou une province.
8. `stateAbbrev` varchar deux caractères État américain, par exemple MA, NY, MI. Ceux-ci sont contrôlés par la table `state_lookup`.

9. `zip` varchar Code postal à 5 chiffres, par exemple 02109.
10. `parsed` booléen - indique si l'adresse a été formée à partir du processus de normalisation. La fonction normalize_address fixe cette valeur à true avant de renvoyer l'adresse.
11. `zip4` 4 derniers chiffres d'un code postal à 9 chiffres. Disponibilité : PostGIS 2.4.0.
12. `address_alphanumeric` Numéro de rue complet, même s'il contient des caractères alpha comme 17R. Il est préférable d'utiliser la fonction Pagc_Normalize_Address. Disponibilité : PostGIS 2.4.0.

Exemples

Exemple d'appel unique

```
SELECT addy.*
FROM pagc_normalize_address('9000 E ROO ST STE 999, Springfield, CO') AS addy;


 address | predirabbrev | streetname | streettypeabbrev | postdirabbrev | internal  |   ↩
    location    | stateabbrev | zip | parsed
---------+--------------+------------+------------------+---------------+-----------+-------------+-

    9000 | E            | ROO        | ST               |               | SUITE 999 |  ↩
       SPRINGFIELD | CO          |     | t
```

Appel par lots. Il y a actuellement des problèmes de vitesse avec la façon dont postgis_tiger_geocoder mobilise l'address_standardizer. Ces problèmes seront résolus dans les prochaines éditions. Pour les contourner, si vous avez besoin de rapidité pour le géocodage par lots pour appeler la génération d'un normaddy en mode batch, nous vous encourageons à appeler directement la fonction address_standardizer standardize_address comme indiqué ci-dessous qui est un exercice similaire à ce que nous avons fait dans Normalize_Address qui utilise les données créées dans Geocode.

```
WITH g AS (SELECT address, ROW((sa).house_num, (sa).predir, (sa).name
  , (sa).suftype, (sa).sufdir, (sa).unit , (sa).city, (sa).state, (sa).postcode, true):: ↩
     norm_addy As na
 FROM (SELECT address, standardize_address('tiger.pagc_lex'
       , 'tiger.pagc_gaz'
       , 'tiger.pagc_rules', address) As sa
        FROM addresses_to_geocode) As g)
SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev
 FROM  g;

 orig                                                 |  streetname   | streettypeabbrev
------------------------------------------------------+---------------+------------------
 529 Main Street, Boston MA, 02129                    | MAIN          | ST
 77 Massachusetts Avenue, Cambridge, MA 02139         | MASSACHUSETTS | AVE
 25 Wizard of Oz, Walaford, KS 99912323               | WIZARD OF     |
 26 Capen Street, Medford, MA                         | CAPEN         | ST
 124 Mount Auburn St, Cambridge, Massachusetts 02138  | MOUNT AUBURN  | ST
 950 Main Street, Worcester, MA 01610                 | MAIN          | ST
```

Voir aussi

Normalize_Address, Geocode

12.2.15 Pprint_Addy

Pprint_Addy — Étant donné un objet de type composite `norm_addy`, renvoie une jolie représentation de celui-ci. Généralement utilisé en conjonction avec normalize_address.

Synopsis

varchar **pprint_addy**(norm_addy in_addy);

Description

Étant donné un objet de type composite `norm_addy`, il renvoie une jolie représentation de cet objet. Aucune autre donnée n'est nécessaire en dehors de celles fournies avec le géocodeur.

Généralement utilisé en conjonction avec Normalize_Address.

Exemples

Jolie représentation d'une adresse unique

```
SELECT pprint_addy(normalize_address('202 East Fremont Street, Las Vegas, Nevada 89101'))  ↩
    As pretty_address;
           pretty_address
----------------------------------------
 202 E Fremont St, Las Vegas, NV 89101
```

Jolie représentation d'adresses à partir d'une table d'adresses

```
SELECT address As orig, pprint_addy(normalize_address(address)) As pretty_address
        FROM addresses_to_geocode;

                        orig                         |              pretty_address
-----------------------------------------------------+------------------------------------------- ↩

 529 Main Street, Boston MA, 02129                   | 529 Main St, Boston MA, 02129
 77 Massachusetts Avenue, Cambridge, MA 02139        | 77 Massachusetts Ave, Cambridge, MA  ↩
    02139
 28 Capen Street, Medford, MA                        | 28 Capen St, Medford, MA
 124 Mount Auburn St, Cambridge, Massachusetts 02138 | 124 Mount Auburn St, Cambridge, MA  ↩
    02138
 950 Main Street, Worcester, MA 01610                | 950 Main St, Worcester, MA 01610
```

Voir aussi

Normalize_Address

12.2.16 Reverse_Geocode

Reverse_Geocode — Prend un point géométrique dans un système spatial connu et renvoie un enregistrement contenant un tableau d'adresses théoriquement possibles et un tableau de rues transversales. Si include_strnum_range = true, la plage de rues est incluse dans les rues transversales.

Synopsis

record **Reverse_Geocode**(geometry pt, boolean include_strnum_range=false, geometry[] OUT intpt, norm_addy[] OUT addy, varchar[] OUT street);

Description

Prend un point géométrique dans un réf spatial connu et renvoie un enregistrement contenant un tableau d'adresses théoriquement possibles et un tableau de rues transversales. Si include_strnum_range = true, la plage de rues est incluse dans les rues transversales. include_strnum_range prend par défaut la valeur false s'il n'est pas fourni. Les adresses sont triées en fonction de la route dont le point est le plus proche, de sorte que la première adresse est très probablement la bonne.

Pourquoi parle-t-on d'adresses théoriques plutôt que d'adresses réelles ? Les données Tiger ne contiennent pas d'adresses réelles, mais seulement des tranches de rues. L'adresse théorique est donc une adresse interpolée sur la base de l'étendue des rues. Par exemple, l'interpolation d'une de mes adresses donne 26 Court St. et 26 Court Sq. alors que le 26 Court Sq. n'existe pas. Cela s'explique par le fait qu'un point peut se trouver à l'angle de deux rues et que la logique interpole donc le long des deux rues. La logique suppose également que les adresses sont également espacées le long d'une rue, ce qui est bien sûr faux puisqu'un bâtiment municipal peut occuper une bonne partie de la rue et que le reste des bâtiments est regroupé à l'extrémité.

Note : Hmm cette fonction repose sur les données Tiger. Si vous n'avez pas chargé de données couvrant la région de ce point, vous obtiendrez un enregistrement rempli de NULLS.

Les éléments renvoyés de l'enregistrement sont les suivants :

1. `intpt` est un tableau de points : Il s'agit des points de la ligne centrale de la rue les plus proches du point d'entrée. Il y a autant de points que d'adresses.
2. `addy` est un tableau de norm_addy (adresses normalisées) : Il s'agit d'un tableau d'adresses possibles correspondant au point d'entrée. La première adresse du tableau est la plus probable. En général, il ne devrait y en avoir qu'une seule, sauf dans le cas où un point se trouve à l'angle de deux ou trois rues, ou si le point se trouve quelque part sur la route et non sur le côté.
3. `street` un tableau de varchar : Il s'agit des rues transversales (ou de la rue) (rues qui croisent ou sont la rue sur laquelle le point est projeté).

Amélioration : 2.4.1 si le jeu de données optionnel zcta5 est chargé, la fonction reverse_geocode peut résoudre l'état et le code postal même si les données spécifiques à l'état ne sont pas chargées. Voir Loader_Generate_Nation_Script pour plus de détails sur le chargement des données zcta5.

Disponibilité : 2.0.0

Exemples

Exemple d'un point situé à l'angle de deux rues, mais le plus proche d'une seule. C'est l'emplacement approximatif du MIT : 77 Massachusetts Ave, Cambridge, MA 02139 Notez que bien que nous n'ayons pas 3 rues, PostgreSQL retournera simplement null pour les entrées supérieures à notre limite supérieure, ce qui permet de l'utiliser en toute sécurité. Cela inclut les plages de rues

```
SELECT pprint_addy(r.addy[1]) As st1, pprint_addy(r.addy[2]) As st2, pprint_addy(r.addy[3]) ↩
    As st3,
           array_to_string(r.street, ',') As cross_streets
       FROM reverse_geocode(ST_GeomFromText('POINT(-71.093902 42.359446)',4269),true) As r ↩
           ;

 result
 ------
      st1                                    | st2 | st3 |                cross_streets
-----------------------------------------+-----+-----+-------------------------------------------

 67 Massachusetts Ave, Cambridge, MA 02139 |     |     | 67 - 127 Massachusetts Ave,32 - 88 ↩
     Vassar St
```

Ici, nous avons choisi de ne pas inclure les plages d'adresses pour les rues transversales et nous avons choisi un emplacement vraiment très proche d'un coin de deux rues, qui pourrait donc être connu sous deux adresses différentes.

```
SELECT pprint_addy(r.addy[1]) As st1, pprint_addy(r.addy[2]) As st2,
pprint_addy(r.addy[3]) As st3, array_to_string(r.street, ',') As cross_str
FROM reverse_geocode(ST_GeomFromText('POINT(-71.06941 42.34225)',4269)) As r;

result
--------
               st1               |               st2               | st3 | cross_str
---------------------------------+---------------------------------+-----+------------------------ ←

 5 Bradford St, Boston, MA 02118 | 49 Waltham St, Boston, MA 02118 |     | Waltham St
```

Pour celui-ci, nous réutilisons notre exemple géocodé de Geocode et nous ne voulons que l'adresse principale et au plus deux rues transversales.

```
SELECT actual_addr, lon, lat, pprint_addy((rg).addy[1]) As int_addr1,
    (rg).street[1] As cross1, (rg).street[2] As cross2
FROM (SELECT address As actual_addr, lon, lat,
    reverse_geocode( ST_SetSRID(ST_Point(lon,lat),4326) ) As rg
    FROM addresses_to_geocode WHERE rating
> -1) As foo;

                     actual_addr                     |    lon    |   lat    |  ↩
                                        int_addr1                 |     cross1      |    ↩
                         cross2
-----------------------------------------------------+-----------+----------+-----------------------

 529 Main Street, Boston MA, 02129                   | -71.07181 | 42.38359 | 527 Main St,  ↩
     Boston, MA 02129             | Medford St      |
 77 Massachusetts Avenue, Cambridge, MA 02139        | -71.09428 | 42.35988 | 77  ↩
     Massachusetts Ave, Cambridge, MA 02139 | Vassar St       |
 26 Capen Street, Medford, MA                        | -71.12377 | 42.41101 | 9 Edison Ave, ↩
      Medford, MA 02155           | Capen St        | Tesla Ave
 124 Mount Auburn St, Cambridge, Massachusetts 02138 | -71.12304 | 42.37328 | 3 University  ↩
     Rd, Cambridge, MA 02138      | Mount Auburn St |
 950 Main Street, Worcester, MA 01610                | -71.82368 | 42.24956 | 3 Maywood St, ↩
      Worcester, MA 01603         | Main St         | Maywood Pl
```

Voir aussi

Pprint_Addy, Geocode, Loader_Generate_Nation_Script

12.2.17 Topology_Load_Tiger

Topology_Load_Tiger — Charge une région définie de données tiger dans une topologie PostGIS et transforme les données tiger en référence spatiale de la topologie et en s'adaptant à la tolérance de précision de la topologie.

Synopsis

text **Topology_Load_Tiger**(varchar topo_name, varchar region_type, varchar region_id);

Description

Charge une région définie de données tigrées dans une topologie PostGIS. Les faces, les nœuds et les arêtes sont transformés dans le système de référence spatiale de la topologie cible et les points sont adaptés à la tolérance de la topologie cible. Les faces,

les nœuds et les arêtes créés conservent les mêmes identifiants que les faces, les nœuds et les arêtes des données Tiger d'origine, de sorte que les ensembles de données puissent à l'avenir être plus facilement réconciliés avec les données Tiger. Renvoie un résumé des détails du processus.

Cela peut s'avérer utile, par exemple, pour les données de redécoupage, lorsque les polygones nouvellement formés doivent suivre les lignes centrales des rues et que les polygones résultants ne doivent pas se chevaucher.

Note
Cette fonction repose sur les données Tiger ainsi que sur l'installation du module topologique PostGIS. Pour plus d'informations, voir Chapter 9 et Section 2.2.3. Si vous n'avez pas chargé de données couvrant la région concernée, aucun enregistrement topologique ne sera créé. Cette fonction échouera également si vous n'avez pas créé de topologie à l'aide des fonctions de topologie.

Note
La plupart des erreurs de validation de la topologie sont dues à des problèmes de tolérance : après la transformation, les points des arêtes ne s'alignent pas tout à fait ou ne se chevauchent pas. Pour remédier à cette situation, vous pouvez augmenter ou réduire la précision si vous obtenez des échecs de validation de la topologie.

Arguments requis :

1. `topo_name` Le nom d'une topologie PostGIS existante dans laquelle charger les données.
2. `region_type` Type de région délimitée. Actuellement, seuls `place` et `county` sont pris en charge. Il est prévu d'en avoir plusieurs autres. Il s'agit de la table à consulter pour définir les limites de la région, par exemple `tiger.place`, `tiger.county`
3. `region_id` C'est ce que TIGER appelle le geoid. Il s'agit de l'identifiant unique de la région dans la table. Pour la table place, il s'agit de la colonne `plcidfp` dans `tiger.place`. Pour la table county, il s'agit de la colonne `cntyidfp` dans `tiger.county`

Disponibilité : 2.0.0

Exemple : Boston, Massachusetts Topologie

Créez une topologie pour Boston, Massachusetts dans Mass State Plane Feet (2249) avec une tolérance de 0,25 pied, puis chargez dans Boston city tiger faces, edges, nodes.

```
SELECT topology.CreateTopology('topo_boston', 2249, 0.25);
createtopology
--------------
   15
-- 60,902 ms ~ 1 minute on windows 7 desktop running 9.1 (with 5 states tiger data loaded)
SELECT tiger.topology_load_tiger('topo_boston', 'place', '2507000');
-- topology_loader_tiger --
29722 edges holding in temporary. 11108 faces added. 1875 edges of faces added.  20576 ↩
    nodes added.
19962 nodes contained in a face.  0 edge start end corrected.  31597 edges added.

-- 41 ms --
SELECT topology.TopologySummary('topo_boston');
 -- topologysummary--
Topology topo_boston (15), SRID 2249, precision 0.25
20576 nodes, 31597 edges, 11109 faces, 0 topogeoms in 0 layers

-- 28,797 ms to validate yeh returned no errors --
SELECT * FROM
```

```
    topology.ValidateTopology('topo_boston');

       error       |   id1    |    id2
-------------------+----------+-----------
```

Exemple : Suffolk, Massachusetts Topologie

Créez une topologie pour Suffolk, Mass State Plane Meters (26986) avec une tolérance de 0,25 mètre, puis chargez dans le schéma tiger du comté de Suffolk les faces, edges, nodes.

```
SELECT topology.CreateTopology('topo_suffolk', 26986, 0.25);
-- this took 56,275 ms ~ 1 minute on Windows 7 32-bit with 5 states of tiger loaded
-- must have been warmed up after loading boston
SELECT tiger.topology_load_tiger('topo_suffolk', 'county', '25025');
-- topology_loader_tiger --
 36003 edges holding in temporary. 13518 faces added. 2172 edges of faces added.
 24761 nodes added.  24075 nodes contained in a face.  0 edge start end corrected.  38175  ↩
     edges added.
-- 31 ms --
SELECT topology.TopologySummary('topo_suffolk');
 -- topologysummary--
 Topology topo_suffolk (14), SRID 26986, precision 0.25
24761 nodes, 38175 edges, 13519 faces, 0 topogeoms in 0 layers

-- 33,606 ms to validate --
SELECT * FROM
    topology.ValidateTopology('topo_suffolk');

       error       |   id1    |    id2
-------------------+----------+-----------
 coincident nodes  | 81045651 |  81064553
 edge crosses node | 81045651 |  85737793
 edge crosses node | 81045651 |  85742215
 edge crosses node | 81045651 | 620628939
 edge crosses node | 81064553 |  85697815
 edge crosses node | 81064553 |  85728168
 edge crosses node | 81064553 |  85733413
```

Voir aussi

CreateTopology, CreateTopoGeom, TopologySummary, ValidateTopology

12.2.18 Set_Geocode_Setting

Set_Geocode_Setting — Définit un paramètre qui affecte le comportement des fonctions du géocodeur.

Synopsis

text **Set_Geocode_Setting**(text setting_name, text setting_value);

Description

Définit la valeur d'un paramètre spécifique stocké dans la table `tiger.geocode_settings`. Les paramètres permettent d'activer le débogage des fonctions. Plus tard, il sera envisagé de contrôler la notation à l'aide de paramètres. La liste actuelle des paramètres est listée dans Get_Geocode_Setting.

Disponibilité : 2.1.0

Exemple de paramétrage du débogage de retour

Si vous exécutez Geocode alors que cette fonction est vraie, le journal NOTICE indiquera le temps et les requêtes.

```
SELECT set_geocode_setting('debug_geocode_address', 'true') As result;
result
---------
true
```

Voir aussi

Get_Geocode_Setting

Chapter 13

Index des fonctions spéciales de PostGIS

13.1 Fonctions d'agrégation de PostGIS

Les fonctions ci-dessous sont des fonctions d'agrégation spatiale qui sont utilisées de la même manière que les fonctions d'agrégation SQL telles que `sum` et `average`.

- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- ST_3DExtent - Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.
- ST_3DUnion - Effectuer l'union 3D.
- ST_AsFlatGeobuf - Renvoie une représentation FlatGeobuf d'un ensemble de lignes.
- ST_AsGeobuf - Retourne une représentation Geobuf d'un ensemble de lignes.
- ST_AsMVT - Fonction d'agrégation renvoyant une représentation MVT d'un ensemble de lignes.
- ST_ClusterDBSCAN - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie d'entrée en utilisant l'algorithme DBSCAN.
- ST_ClusterIntersecting - Fonction d'agrégation qui regroupe les géométries en entrée en ensembles connectés.
- ST_ClusterIntersectingWin - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, en regroupant les géométries en entrée en ensembles connectés.
- ST_ClusterKMeans - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée en utilisant l'algorithme K-means.
- ST_ClusterWithin - Fonction agrégée qui regroupe les géométries en fonction de la distance de séparation.
- ST_ClusterWithinWin - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, regroupement en utilisant la distance de séparation.
- ST_Collect - Crée une géométrie GeometryCollection ou Multi à partir d'un ensemble de géométries.
- ST_CoverageInvalidEdges - Fonction window qui trouve les endroits où les polygones ne forment pas une couverture valide.
- ST_CoverageSimplify - Fonction window qui simplifie les bords d'une couverture polygonale.
- ST_CoverageUnion - Calcule l'union d'un ensemble de polygones formant une couverture en supprimant les arêtes communes.
- ST_Extent - Fonction agrégée qui renvoie la boîte de délimitation des géométries.
- ST_MakeLine - Crée une LineString à partir de géométries Point, MultiPoint ou LineString.

- ST_MemUnion - Fonction d'agrégation qui fusionne les géométries d'une manière efficace sur le plan de la mémoire mais plus lente
- ST_Polygonize - Calcule une collection de polygones formés à partir du tracé d'un ensemble de géométries.
- ST_SameAlignment - Retourne true si les rasters ont les mêmes skew, scale, spatial ref, et offset (les pixels peuvent être placés sur la même grille sans être coupés) et false si ce n'est pas le cas, avec une NOTICE détaillant le problème.
- ST_Union - Calcule une géométrie représentant l'union des ensembles de points des géométries d'entrée.
- ST_Union - Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.
- TopoElementArray_Agg - Renvoie un topoelementarray pour un ensemble de tableaux de type, element_id (topoelements).

13.2 Fonctions Window PostGIS

Les fonctions ci-dessous sont des fonctions window spatiale qui sont utilisées de la même manière que les fonctions de fenêtre SQL telles que `row_number()`, `lead()`, et `lag()`. Elles doivent être suivies d'une clause `OVER()`.

- ST_ClusterDBSCAN - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie d'entrée en utilisant l'algorithme DBSCAN.
- ST_ClusterIntersectingWin - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, en regroupant les géométries en entrée en ensembles connectés.
- ST_ClusterKMeans - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée en utilisant l'algorithme K-means.
- ST_ClusterWithinWin - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, regroupement en utilisant la distance de séparation.
- ST_CoverageInvalidEdges - Fonction window qui trouve les endroits où les polygones ne forment pas une couverture valide.
- ST_CoverageSimplify - Fonction window qui simplifie les bords d'une couverture polygonale.

13.3 Fonctions de PostGIS compatibles avec SQL-MM

Les fonctions ci-dessous sont des fonctions PostGIS conformes à la norme SQL/MM 3

- CG_3DArea - Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- CG_3DDifference - Effectuer une différence 3D
- CG_3DIntersection - Réaliser une intersection 3D
- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- CG_Volume - Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- ST_3DArea - Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- ST_3DDWithin - Teste si deux géométries 3D se trouvent à une distance 3D donnée
- ST_3DDifference - Effectuer une différence 3D
- ST_3DDistance - Renvoie la distance cartésienne minimale en 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DIntersection - Réaliser une intersection 3D

- ST_3DIntersects - Teste si deux géométries se croisent dans l'espace en 3D - uniquement pour les points, les lignes, les polygones, les surfaces polyédriques (aire)
- ST_3DLength - Renvoie la longueur 3D d'une géométrie linéaire.
- ST_3DPerimeter - Renvoie le périmètre 3D d'une géométrie polygonale.
- ST_3DUnion - Effectuer l'union 3D.
- ST_AddEdgeModFace - Ajoutez une nouvelle arête et, si elle divise une face, modifiez la face d'origine et ajoutez une nouvelle face.
- ST_AddEdgeNewFaces - Ajoutez une nouvelle arête et, si elle divise une face, supprimez la face d'origine et remplacez-la par deux nouvelles faces.
- ST_AddIsoEdge - Ajoute une arête isolée définie par la géométrie alinestring à une topologie reliant deux nœuds isolés existants anode et anothernode et renvoie l'identifiant de l'arête de la nouvelle arête.
- ST_AddIsoNode - Ajoute un noeud isolé à une face dans une topologie et renvoie le nodeid du nouveau noeud. Si la face est nulle, le noeud est quand même créé.
- ST_Area - Renvoie l'aire d'une géométrie polygonale.
- ST_AsBinary - Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsGML - Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsText - Renvoie la représentation Well-Known Text (WKT) de la géométrie/geography sans métadonnées SRID.
- ST_Boundary - Renvoie la limite d'une géométrie.
- ST_Buffer - Calcule une géométrie couvrant tous les points situés à une distance donnée d'une géométrie.
- ST_Centroid - Renvoie le centre géométrique d'une géométrie.
- ST_ChangeEdgeGeom - Modifie la forme d'une arête sans affecter la structure de la topologie.
- ST_Contains - Tests si chaque point de B est situé dans A, et que leurs intérieurs ont un point commun
- ST_ConvexHull - Calcule l'enveloppe convexe d'une géométrie.
- ST_CoordDim - Renvoie la dimension des coordonnées d'une géométrie.
- ST_CreateTopoGeo - Ajoute une collection de géométries à une topologie vide donnée et renvoie un message détaillant le succès.
- ST_Crosses - Teste si deux géométries ont en commun certains points intérieurs, mais pas tous
- ST_CurveN - Renvoie la Nième courbe composante d'une CompoundCurve.
- ST_CurveToLine - Convertit une géométrie contenant des courbes en une géométrie linéaire.
- ST_Difference - Calcule une géométrie représentant la partie de la géométrie A qui n'intersecte pas la géométrie B.
- ST_Dimension - Renvoie la dimension topologique d'une géométrie.
- ST_Disjoint - Teste si deux géométries n'ont pas de points communs
- ST_Distance - Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_EndPoint - Renvoie le dernier point d'une LineString ou CircularLineString.
- ST_Envelope - Renvoie une géométrie représentant la boîte de délimitation d'une géométrie.
- ST_Equals - Teste si deux géométries comprennent le même ensemble de points

- ST_ExteriorRing - Renvoie une ligne représentant l'anneau extérieur d'un polygone.
- ST_GMLToSQL - Retourne un objet de type ST_Geometry à partir de sa représentation GML. Alias pour ST_GeomFromGML
- ST_GeomCollFromText - Crée une collection Geometry à partir de la collection WKT avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0.
- ST_GeomFromText - Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT).
- ST_GeomFromWKB - Retourne un objet de type geometry à partir de sa représentation binaire Well-Know Binary (WKB) et d'un SRID optionnel.
- ST_GeometryFromText - Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT). Alias pour ST_GeomFromText
- ST_GeometryN - Renvoie un élément d'une collection de géométries.
- ST_GeometryType - Renvoie le type SQL-MM d'une géométrie sous forme de texte.
- ST_GetFaceEdges - Renvoie un ensemble d'arêtes ordonnées qui délimitent aface.
- ST_GetFaceGeometry - Renvoie le polygone dans la topologie donnée avec l'identifiant de face spécifié.
- ST_InitTopoGeo - Crée un nouveau schéma topologique et l'enregistre dans la table topology.topology.
- ST_InteriorRingN - Renvoie le Nième anneau intérieur (trou) d'un polygone.
- ST_Intersection - Calcule une géométrie représentant la partie partagée des géométries A et B.
- ST_Intersects - Teste si deux géométries se croisent (elles ont au moins un point en commun)
- ST_IsClosed - Teste si les points de départ et d'arrivée d'une LineString coïncident. Pour une PolyhedralSurface, teste si elle est fermée (volumétrique).
- ST_IsEmpty - Teste si une géométrie est vide.
- ST_IsRing - Teste si une ligne est fermée et simple.
- ST_IsSimple - Teste si une géométrie n'a pas de points d'auto-intersection ou d'auto-tangente.
- ST_IsValid - Teste si une géométrie est bien formée en 2D.
- ST_Length - Renvoie la longueur 2D d'une géométrie linéaire.
- ST_LineFromText - Construit une géométrie à partir d'une représentation WKT avec le SRID donné. Si aucun SRID n'est donné, la valeur par défaut est 0.
- ST_LineFromWKB - Construit une LINESTRING depuis la représentation binaire WKB et le srid donné
- ST_LinestringFromWKB - Construit une géométrie depuis la représentation binaire WKB et le SRID donné.
- ST_LocateAlong - Renvoie le(s) point(s) d'une géométrie qui correspond(ent) à une valeur de mesure.
- ST_LocateBetween - Renvoie les parties d'une géométrie qui correspondent à un intervalle de mesure.
- ST_M - Renvoie la coordonnée M d'un point.
- ST_MLineFromText - Retourne un objet de type ST_MultiLineString à partir de sa représentation WKT.
- ST_MPointFromText - Créé une Geometry depuis un WKT avec le SRID donné. Si le SRID n'est pas fourni, il sera défini par défaut à 0.
- ST_MPolyFromText - Crée une géométrie multi-polygone à partir de WKT avec le SRID donné. Si le SRID n'est pas donné, la valeur par défaut est 0.
- ST_ModEdgeHeal - Répare deux arêtes en supprimant le nœud qui les relie, en modifiant la première arête et en supprimant la seconde. Renvoie l'identifiant du nœud supprimé.

- ST_ModEdgeSplit - Fractionner une arête en créant un nouveau nœud le long d'une arête existante, en modifiant l'arête d'origine et en ajoutant une nouvelle arête.
- ST_MoveIsoNode - Déplace un nœud isolé dans une topologie d'un point à un autre. Si la nouvelle géométrie apoint existe en tant que noeud, une erreur est générée. Retourne la description du déplacement.
- ST_NewEdgeHeal - Répare deux arêtes en supprimant le nœud qui les relie, en supprimant les deux arêtes et en les remplaçant par une arête dont la direction est la même que la première arête fournie.
- ST_NewEdgesSplit - Fractionne une arête en créant un nouveau nœud le long d'une arête existante, en supprimant l'arête d'origine et en la remplaçant par deux nouvelles arêtes. Renvoie l'identifiant du nouveau nœud créé qui relie les nouvelles arêtes.
- ST_NumCurves - Renvoie le nombre de courbes composantes d'une CompoundCurve.
- ST_NumGeometries - Renvoie le nombre d'éléments dans une collection de géométrie.
- ST_NumInteriorRings - Renvoie le nombre d'anneaux intérieurs (trous) d'un polygone.
- ST_NumPatches - Renvoie le nombre de faces d'une surface polyédrique. Retourne null pour les géométries non polyédriques.
- ST_NumPoints - Renvoie le nombre de points dans une LineString ou une CircularString.
- ST_OrderingEquals - Teste si deux géométries représentent la même géométrie et ont des points dans le même ordre directionnel
- ST_Overlaps - Teste si deux géométries ont la même dimension et se croisent, mais si chacune a au moins un point qui n'est pas dans l'autre
- ST_PatchN - Renvoie la Nième géométrie (face) d'une PolyhedralSurface.
- ST_Perimeter - Renvoie la longueur de la limite d'une géométrie polygonale ou d'une géographie.
- ST_Point - Crée un point avec des valeurs X, Y et SRID.
- ST_PointFromText - Construit une géométrie point à partir d'une représentation WKT et le SRID donné. Si aucun SRID n'est donné, la valeur par défaut est 0.
- ST_PointFromWKB - Construit une géométrie depuis la représentation binaire WKB et le SRID donné
- ST_PointN - Renvoie le Nième point de la première LineString ou LineString circulaire d'une géométrie.
- ST_PointOnSurface - Calcule un point dont on garantit qu'il se trouve dans un polygone ou sur une géométrie.
- ST_Polygon - Crée un polygone à partir d'une LineString avec un SRID spécifié.
- ST_PolygonFromText - Créé une Geometry depuis un WKT avec le SRID donné. Si le SRID n'est pas fourni, il sera défini par défaut à 0.
- ST_Relate - Teste si deux géométries ont une relation topologique correspondant à un modèle de matrice d'intersection, ou calcule leur matrice d'intersection
- ST_RemEdgeModFace - Supprime une arête et, si l'arête sépare deux faces, supprime une face et modifie l'autre face pour couvrir l'espace des deux.
- ST_RemEdgeNewFace - Enlève une arête et, si l'arête enlevée séparait deux faces, supprime les faces originales et les remplace par une nouvelle face.
- ST_RemoveIsoEdge - Supprime une arête isolée et renvoie la description de l'action. Si l'arête n'est pas isolée, une exception est levée.
- ST_RemoveIsoNode - Supprime un noeud isolé et renvoie la description de l'action. Si le noeud n'est pas isolé (début ou fin d'une arête), une exception est levée.
- ST_SRID - Renvoie l'identifiant de référence spatiale d'une géométrie.

- ST_StartPoint - Renvoie le premier point d'une LineString.
- ST_SymDifference - Calcule une géométrie représentant les parties des géométries A et B qui ne s'intersectent pas.
- ST_Touches - Teste si deux géométries ont au moins un point en commun, mais que leurs intérieurs ne se croisent pas
- ST_Transform - Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent.
- ST_Union - Calcule une géométrie représentant l'union des ensembles de points des géométries d'entrée.
- ST_Volume - Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- ST_WKBToSQL - Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Binary (WKB). Alias pour ST_GeomFromWKB sans SRID
- ST_WKTToSQL - Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT). Alias pour ST_GeomFromText
- ST_Within - Tests si chaque point de A se trouve dans B, et que leurs intérieurs ont un point commun
- ST_X - Renvoie la coordonnée X d'un point.
- ST_Y - Renvoie la coordonnée Y d'un point.
- ST_Z - Renvoie la coordonnée Z d'un point.
- ST_SRID - Renvoie l'identifiant de référence spatiale d'une topogeometry.

13.4 Fonctions d'aide au type geography de PostGIS

Les fonctions et opérateurs ci-dessous sont des fonctions/opérateurs PostGIS qui prennent en entrée ou renvoient en sortie un objet de type geography.

Note
Les fonctions avec un (T) ne sont pas des fonctions géodésiques natives, et utilisent un appel ST_Transform vers et depuis la géométrie pour effectuer l'opération. Par conséquent, elles peuvent ne pas se comporter comme prévu lorsque l'on passe au-dessus de la ligne de base, des pôles, et pour les grandes géométries ou les paires de géométries qui couvrent plus d'une zone UTM. Transformation de base - (favorisant l'UTM, l'azimut Lambert (Nord/Sud), et se rabattant sur le mercator dans le pire des cas)

- ST_Area - Renvoie l'aire d'une géométrie polygonale.
- ST_AsBinary - Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsEWKT - Renvoie la représentation Well-Known Text (WKT) de la géométrie avec les métadonnées SRID.
- ST_AsGML - Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsGeoJSON - Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_AsKML - Renvoyer la géométrie sous forme d'élément KML.
- ST_AsSVG - Renvoie les données de chemin SVG pour une géométrie.
- ST_AsText - Renvoie la représentation Well-Known Text (WKT) de la géométrie/geography sans métadonnées SRID.
- ST_Azimuth - Renvoie l'azimut basé sur le nord d'une ligne entre deux points.
- ST_Buffer - Calcule une géométrie couvrant tous les points situés à une distance donnée d'une géométrie.

- ST_Centroid - Renvoie le centre géométrique d'une géométrie.
- ST_ClosestPoint - Renvoie le point 2D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne la plus courte d'une géométrie à l'autre.
- ST_CoveredBy - Tests si chaque point de A se trouve dans B
- ST_Covers - Tests si chaque point de B est situé dans A
- ST_DWithin - Teste si deux géométries se trouvent à une distance donnée
- ST_Distance - Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_GeogFromText - Retourne un objet de type geography à partir de sa représentation Well-Know Text (WKT ou EWKT).
- ST_GeogFromWKB - Retourne un objet de type geography à partir de sa représentation binaire Well-Know Binary (WKB ou EWKB).
- ST_GeographyFromText - Retourne un objet de type geography à partir de sa représentation Well-Know Text (WKT ou EWKT).
- = - Renvoie TRUE si les coordonnées et l'ordre des coordonnées de la géométrie/géographie A sont les mêmes que les coordonnées et l'ordre des coordonnées de la géométrie/géographie B.
- ST_Intersection - Calcule une géométrie représentant la partie partagée des géométries A et B.
- ST_Intersects - Teste si deux géométries se croisent (elles ont au moins un point en commun)
- ST_Length - Renvoie la longueur 2D d'une géométrie linéaire.
- ST_LineInterpolatePoint - Renvoie un point interpolé le long d'une ligne à un emplacement fractionnaire.
- ST_LineInterpolatePoints - Renvoie des points interpolés le long d'une ligne à un intervalle fractionnaire.
- ST_LineLocatePoint - Renvoie l'emplacement fractionnaire du point le plus proche d'un point sur une ligne.
- ST_LineSubstring - Renvoie la partie d'une ligne située entre deux emplacements fractionnaires.
- ST_Perimeter - Renvoie la longueur de la limite d'une géométrie polygonale ou d'une géographie.
- ST_Project - Renvoie un point projeté à partir d'un point de départ en fonction d'une distance et d'un azimut.
- ST_Segmentize - Renvoie une geometry/geography modifiée dont aucun segment ne dépasse une distance donnée.
- ST_ShortestLine - Renvoie la ligne 2D la plus courte entre deux géométries
- ST_Summary - Renvoie un résumé textuel du contenu d'une géométrie.
- <-> - Renvoie la distance en 2D entre A et B.
- && - Renvoi VRAI si la boite englobante 2D de A intersecte la boite englobante 2D de B.

13.5 Fonctions de support des données raster de PostGIS

Les fonctions et opérateurs ci-dessous sont des fonctions/opérateurs PostGIS qui prennent en entrée ou renvoient en sortie un objet de type raster. Ils sont classés par ordre alphabétique.

- Box3D - Retourne la représentation 3d de la boîte englobante du raster.
- @ - Retourne TRUE si la boîte englobante de A est contenue dans celle de B. Utilise des boîtes englobantes en double précision.
- ~ - Retourne TRUE si la boîte englobante de A contient celle de B. Utilise des boîtes englobantes en double précision.
- = - Retourne TRUE si la boîte englobante de A est la même que celle de B. Utilise des boîtes englobantes en double précision.

- && - Retourne TRUE si la boite englobante de A intersecte la boite englobante de B.
- &< - Retourne TRUE si la boîte englobante de A est à gauche de celle de B.
- &> - Retourne TRUE si la boîte englobante de A est à droite de celle de B.
- ~= - Renvoie TRUE si la boîte de délimitation de A est la même que celle de B.
- ST_Retile - Retourne un ensemble de tuiles configurées à partir d'une couverture raster composée de tuiles arbitraires.
- ST_AddBand - Retourne un raster avec la/les nouvelle(s) bande(s) ajouté(s) à un index donné, de type et valeur initiale donnés. Si aucun index n'est spécifié, la bande est ajoutée à la fin.
- ST_AsBinary/ST_AsWKB - Retourne la représentation Well-Known Binary (WKB) de ce raster.
- ST_AsGDALRaster - Retourne la tuile raster dans le format GDAL raster spécifié. Les formats raster disponibles sont ceux supportés par votre bibliothèque compilée. Utilisez ST_GDALDrivers() pour obtenir la liste des formats supportés par votre bibliothèque.
- ST_AsHexWKB - Retourne la représentation hexadécimale Well-Known Binary (WKB) de ce raster.
- ST_AsJPEG - Retourne les bandes sélectionnées du raster sous la forme d'une image JPEG (sous forme de tableau d'octets). Si aucune bande n'est spécifiée, et que le raster a 1 ou plus de 3 bandes, seule la première bande est utilisée. Si le raster a exactement 3 bandes, les 3 bandes sont utilisées et mappées en RGB.
- ST_AsPNG - Retourne les bandes sélectionnées du raster sous la forme d'une image PNG (sous forme de tableau d'octets). Si aucune bande n'est spécifiée et que le raster a 1, 3 ou 4 bandes, toutes les bandes sont utilisées. Si aucune bande n'est spécifiée et que le raster a 2 ou plus de 4 bandes, seule la bande 1 est utilisée. Les bandes sont mappées en RGB ou RGBA.
- ST_AsRaster - Convertit une géométrie PostGIS en un raster PostGIS.
- ST_AsTIFF - Retourne les bandes sélectionnées du raster sous la forme d'une seule image TIFF (sous forme de tableau d'octets). Si aucune bande n'est spécifiée ou si l'une des bandes spécifiées n'existe pas dans le raster, toutes les bandes sont utilisées.
- ST_Aspect - Retourne l'exposition (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_Band - Retourne une ou plusieurs bandes d'un raster existant en tant que nouveau raster. Utile pour construire de nouveaux rasters à partir de rasters existants.
- ST_BandFileSize - Retourne la taille du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_BandFileTimestamp - Retourne le timestamp du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_BandIsNoData - Retourne true si la bande ne contient que des valeurs nodata.
- ST_BandMetaData - Retourne les méta-données de base d'une bande raster spécifique. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_BandNoDataValue - Retourne la valeur qui représente l'absence de valeur (nodata) pour cette bande. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_BandPath - Retourne le chemin système du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_BandPixelType - Retourne le type de pixel d'une bande. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_Clip - Returns the raster clipped by the input geometry. If band number is not specified, all bands are processed. If crop is not specified or TRUE, the output raster is cropped. If touched is set to TRUE, then touched pixels are included, otherwise only if the center of the pixel is in the geometry it is included.
- ST_ColorMap - Crée un nouveau raster comprenant jusqu'à quatre bandes 8BUI (niveaux de gris, RGB, RGBA) à partir du raster source et d'une bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

- ST_Contains - Retourne true si aucun point du raster rastB ne se trouve à l'extérieur du raster rastA et si au moins un point de l'intérieur du raster rastB se trouve à l'intérieur du raster rastA.
- ST_ContainsProperly - Retourne true si rastB intersecte l'intérieur de rastA, mais pas la frontière ou l'extérieur de rastA.
- ST_Contour - Génère un ensemble de courbes de niveau vectorielles depuis la bande raster spécifiée, en utilisant l'algorithme de contour GDAL.
- ST_ConvexHull - Retourne l'enveloppe convexe du raster, en incluant les valeurs de pixels égales à BandNoDataValue. Pour les données raster de forme régulière et sans obliquité, cette fonction donne le même résultat que ST_Envelope ; elle n'est donc utile que pour les données raster de forme irrégulière ou inclinée.
- ST_Count - Renvoie le nombre de pixels dans une bande donnée d'un raster ou d'une couverture raster. Si aucune bande n'est spécifiée, la valeur par défaut est la bande 1. Si exclude_nodata_value est true, seuls les pixels dont la valeur est différente de la valeur nodata seront comptés.
- ST_CountAgg - Agrégat. Renvoie le nombre de pixels dans une bande donnée d'un ensemble de rasters. Si aucune bande n'est spécifiée, la valeur par défaut est la bande 1. Si exclude_nodata_value est true, seuls les pixels différents de la valeur NODATA seront comptés.
- ST_CoveredBy - Retourne true si aucun point du raster rastA ne se trouve à l'extérieur du raster rastB.
- ST_Covers - Retourne true si aucun point du raster rastB ne se trouve à l'extérieur du raster rastA.
- ST_DFullyWithin - Retourne true si les rasters rastA et rastB se trouvent entièrement à une distance donnée l'un de l'autre.
- ST_DWithin - Retourne true si les rasters rastA et rastB se trouvent à une distance donnée l'un de l'autre.
- ST_Disjoint - Retourne true si le raster rastA n'intersecte pas spatialement le rastB.
- ST_DumpAsPolygons - Retourne un ensemble d'enregistrements de type geomval (geom, val), à partir d'une bande raster donnée. Si aucune bande bandnum n'est spécifiée, la bande 1 est utilisée.
- ST_DumpValues - Retourne les valeurs d'une bande raster spécifiée, sous forme d'un tableau à deux dimensions.
- ST_Envelope - Retourne la représentation polygonale de l'étendue du raster.
- ST_FromGDALRaster - Retourne un raster depuis un fichier raster supporté par GDAL.
- ST_GeoReference - Retourne les méta-données de géo-référencement au format GDAL (par défaut) ou ESRI, tel qu'utilisé généralement dans les fichiers world file.
- ST_Grayscale - Crée un nouveau raster à 1 bande 8BUI à partir du raster source et des bandes spécifiées représentant les composantes rouge, vert et bleu
- ST_HasNoBand - Retourne true si le raster n'a pas la bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_Height - Retourne la hauteur du raster en pixels.
- ST_HillShade - Retourne l'hypothétique éclairage d'une bande raster d'élévation en utilisant les valeurs d'azimut, d'altitude, de luminosité et d'échelle fournies.
- ST_Histogram - Retourne un ensemble d'enregistrements résumant une distribution de données raster ou de couverture raster, dans des classes distinctes. Le nombre de classes est calculé automatiquement s'il n'est pas spécifié.
- ST_InterpolateRaster - Interpole une surface quadrillée à partir d'un ensemble de points 3-d, en utilisant les coordonnées X et Y des points sur la grille et la coordonnée Z des points pour l'élévation des points.
- ST_Intersection - Retourne un raster ou un ensemble de paires (géométrie, valeur de pixel) représentant la partie partagée de deux rasters ou l'intersection géométrique d'une vectorisation du raster et d'une géométrie.
- ST_Intersects - Retourne vrai si le raster rastA intersecte spatialement le raster rastB.
- ST_IsEmpty - Retourne true si le raster est vide (largeur = 0 et hauteur = 0). Sinon, retourne false.
- ST_MakeEmptyCoverage - Couvre une zone géo-référencée avec une grille raster de tuiles vides.

- ST_MakeEmptyRaster - Retourne un raster vide (sans aucune bande) de dimension donnée (width & height), de coin supérieur gauche à X et Y, de paramètres de taille de pixel données (scalex, scaley, skewx & skewy) et de système de référence spatial (srid) donné. Si un raster est spécifié, retourne un nouveau raster de même taille, alignement et SRID. Si srid n'est pas spécifié, le système de référence spatial est défini à inconnu (0).
- ST_MapAlgebra (callback function version) - Version avec fonction de rappel - Retourne un raster à une bande à partir d'un ou plusieurs rasters d'entrée, d'index de bandes et d'une fonction de rappel spécifiée par l'utilisateur.
- ST_MapAlgebraExpr - Version avec raster à 1 bande : Crée un nouveau raster à une bande formé par l'application d'une opération algébrique PostgreSQL valide sur la bande d'entrée du raster et du type de pixel fourni. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_MapAlgebraExpr - Version avec 2 bandes : Crée un nouveau raster à une bande formé en appliquant une opération algébrique PostgreSQL valide sur les deux bandes du raster d'entrée et du type de pixel fourni. La bande 1 de chaque raster est utilisée si aucun numéro de bande n'est spécifié. Le raster résultant sera aligné (échelle, obliquité et coins) sur la grille définie par le premier raster et aura son étendue définie par le paramètre "extenttype". Les valeurs de "extenttype" peuvent être : INTERSECTION, UNION, FIRST, SECOND.
- ST_MapAlgebraFct - Version à 1 bande - Crée un nouveau raster à une bande formé par l'application d'une fonction PostgreSQL valide sur la bande d'entrée du raster et sur le type de pixel fourni. La bande 1 est utilisée si aucune bande n'est spécifiée.
- ST_MapAlgebraFct - Version à 2 bandes - Crée un nouveau raster à une bande formé par l'application d'une fonction PostgreSQL valide sur les 2 bandes d'entrée du raster et sur le type de pixel fourni. La bande 1 est utilisée si aucune bande n'est spécifiée. Le type d'étendue INTERSECTION est utilisé si non spécifié.
- ST_MapAlgebraFctNgb - Version à 1 bande : Algèbre cartographique Plus proche voisin en utilisant une fonction PostgreSQL définie par l'utilisateur. Retourne un raster dont les valeurs sont le résultat d'une fonction utilisateur PLPGSQL prenant un voisinage des valeurs de la bande raster d'entrée.
- ST_MapAlgebra (expression version) - Version avec expression - Retourne un raster à une bande à partir d'un ou deux rasters d'entrée, d'index de bandes et d'une ou plusieurs expressions SQL spécifiées par l'utilisateur.
- ST_MemSize - Retourne l'espace utilisé par le raster (en octets).
- ST_MetaData - Retourne les méta-données de base de l'objet raster : taille des pixels, rotation, coin haut/bas gauche, etc.
- ST_MinConvexHull - Retourne la géométrie de l'enveloppe convexe du raster en excluant les pixels NODATA.
- ST_NearestValue - Retourne la valeur la plus proche différent de NODATA pour une bande raster spécifiée au pixel donné par columnx et rowy, ou à un point géométrique spécifié dans le même système de référence spatial que le raster.
- ST_Neighborhood - Retourne un tableau 2-D de double avec les valeurs non NODATA autour du pixel de la bande spécifiée, aux coordonnées spécifiées par columnX & rowY ou par un point géométrique dans le même système de référence spatial que le raster.
- ST_NotSameAlignmentReason - Retourne un texte indiquant si les rasters sont alignés et, s'ils ne le sont pas, la raison du problème.
- ST_NumBands - Retourne le nombre de bandes de l'objet raster.
- ST_Overlaps - Retourne true si les raster rastA et rastB intersectent mais que l'un ne contient pas complètement l'autre.
- ST_PixelAsCentroid - Retourne le centroïde (point géométrique) de la zone représentée par un pixel.
- ST_PixelAsCentroids - Retourne le centroïde (point géométrique) pour chaque pixel de la bande raster, avec sa valeur et les coordonnées raster X et Y. Le point géométrique est le centroïde de la zone représentée par un pixel.
- ST_PixelAsPoint - Retourne le point géométrique du coin supérieur gauche du pixel.
- ST_PixelAsPoints - Retourne un point géométrique pour chaque pixel de la bande raster, avec sa valeur et ses coordonnées raster X et Y. Les coordonnées du points sont ceux du coin supérieur gauche du pixel.
- ST_PixelAsPolygon - Retourne la géométrie polygonale qui délimite le pixel pour une ligne et colonne spécifiées.

- ST_PixelAsPolygons - Retourne la géométrie polygonale qui délimite chaque pixel de la bande raster, avec la valeur et les coordonnées raster X et Y de chaque pixel.
- ST_PixelHeight - Retourne la hauteur d'un pixel, dans l'unité du système de référence spatial.
- ST_PixelOfValue - Retourne les coordonnées columnx, rowy du pixel dont la valeur est égale à la valeur recherchée.
- ST_PixelWidth - Retourne la largeur d'un pixel, dans l'unité du système de référence spatial.
- ST_Polygon - Retourne une géométrie multipolygonale formée par l'union des pixels dont la valeur est différente de nodata. Si aucune bande bandnum n'est spécifiée, la bande 1 est utilisée.
- ST_Quantile - Calcule les quantiles d'un raster ou d'une couverture raster, dans le contexte de l'échantillon ou de la population. Ainsi, une valeur peut être examinée pour se situer au percentile de 25%, 50% ou 75% du raster.
- ST_RastFromHexWKB - Retourne un raster à partir d'un raster Well-Known Binary (WKB) en hexadécimal.
- ST_RastFromWKB - Retourne un raster à partir d'un raster Well-Known Binary (WKB).
- ST_RasterToWorldCoord - Retourne le coin supérieur gauche du raster, sous forme de coordonnées X et Y (longitude et latitude) d'un colonne et d'une ligne. Les numéros de colonne et de ligne commencent à 1.
- ST_RasterToWorldCoordX - Retourne la coordonnée X du coin supérieur gauche du raster à column et row. Les numéros de colonne et de ligne commencent à 1.
- ST_RasterToWorldCoordY - Retourne la coordonnée Y du coin supérieur gauche du raster à column et row. Les numéros de colonne et de ligne commencent à 1.
- ST_Reclass - Crée un nouveau raster composé de types de bandes reclassifiés par rapport à l'original. La bande n est la bande à modifier. Si nband n'est pas spécifié, la bande 1 est utilisée. Toutes les autres bandes sont retournées inchangées. Cas d'utilisation : convertir une bande 16BUI en 8BUI et ainsi de suite pour un rendu plus simple en tant que formats visualisables.
- ST_Resample - Rééchantillonne un raster, en utilisant l'algorithme spécifié, les nouvelles dimensions, un coin arbitraire de la grille et un ensemble de paramètres de géo-référencement définis ou empruntés à un autre raster.
- ST_Rescale - Rééchantillonne un raster en ajustant juste son échelle (ou la taille des pixels). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline), Lanczos, Max ou Min. La valeur par défaut est NearestNeighbor.
- ST_Resize - Redimensionne un raster à une nouvelle largeur/hauteur
- ST_Reskew - Rééchantillonne un raster en ajustant simplement son obliquité (skew, ou paramètre de rotation). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.
- ST_Rotation - Retourne l'angle de rotation du raster en radians.
- ST_Roughness - Retourne un raster avec la rugosité d'un MNT.
- ST_SRID - Retourne l'identifiant du système de référence spatial du raster, tel que défini dans la table spatial_ref_sys.
- ST_SameAlignment - Retourne true si les rasters ont les mêmes skew, scale, spatial ref, et offset (les pixels peuvent être placés sur la même grille sans être coupés) et false si ce n'est pas le cas, avec une NOTICE détaillant le problème.
- ST_ScaleX - Renvoie la composante X de la largeur du pixel dans l'unité du système de référence spatial.
- ST_ScaleY - Renvoie la composante Y de la hauteur du pixel dans l'unité du système de référence spatial.
- ST_SetBandIndex - Met à jour le numéro de bande externe d'une bande out-db
- ST_SetBandIsNoData - Définit la valeur du flag isnodata de la bande à TRUE.
- ST_SetBandNoDataValue - Définit la valeur pour l'absence de données (nodata) pour la bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée. Pour indiquer qu'une bande n'a pas de valeur nodata, définir la valeur nodata = NULL.
- ST_SetBandPath - Met à jour le chemin externe et le numéro de bande d'une bande out-db

- ST_SetGeoReference - Définit les 6 paramètres de géo-référencement en un seul appel. Les nombres doivent être séparés par un espace. Accepte les formats GDAL (par défaut) ou ESRI.
- ST_SetM - Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée M copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.
- ST_SetRotation - Définit la rotation du raster en radians.
- ST_SetSRID - Modifie le SRID d'un raster à une valeur définie dans la table spatial_ref_sys.
- ST_SetScale - Définit la résolution des pixels en X et Y en unité du système de référence spatial : nombre d'unités/pixel en largeur/hauteur.
- ST_SetSkew - Définit l'obliquité X et Y (skew, ou paramètre de rotation). Si une seule valeur est spécifiée, la même valeur est utilisée pour X et pour Y.
- ST_SetUpperLeft - Modifie les coordonnées du pixel du coin supérieur gauche du raster, selon les coordonnées X et Y projetées.
- ST_SetValue - Retourne un nouveau raster en modifiant la valeur du pixel pour la bande spécifiée et aux coordonnées columnx, rowy, ou pour tous les pixels qui intersectent une géométrie spécifiée. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié.
- ST_SetValues - Retourne un nouveau raster en modifiant les valeurs de certains pixels d'une bande spécifiée.
- ST_SetZ - Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée Z copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.
- ST_SkewX - Retourne l'obliquité géo-référencée X (paramètre de rotation).
- ST_SkewY - Retourne l'obliquité géo-référencée Y (paramètre de rotation).
- ST_Slope - Retourne la pente (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_SnapToGrid - Rééchantillonne un raster en l'accrochant sur une grille. Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.
- ST_Summary - Retourne un résumé du contenu du raster sous forme de texte.
- ST_SummaryStats - Retourne des résumés statistiques (count, sum, mean, stddev, min, max) pour une bande raster ou une couverture raster spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_SummaryStatsAgg - Agrégat. Retourne des résumés statistiques (count, sum, mean, stddev, min, max) pour une bande raster spécifiée pour une ensemble de rasters. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_TPI - Retourne un raster avec l'index de position topographique (TPI) calculé.
- ST_TRI - Retourne un raster avec l'indice de rugosité du terrain (TRI) calculé.
- ST_Tile - Retourne un ensemble de rasters issus de la division d'un raster d'entrée selon les dimensions spécifiées pour les rasters de sortie.
- ST_Touches - Retourne true si les raster rastA et rastB ont au moins un point en commun mais que leurs intérieurs n'intersectent pas.
- ST_Transform - Reprojette un raster depuis un système de référence spatial vers un autre, en utilisant l'algorithme de rééchantillonnage spécifié. Les algorithmes possibles sont NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.
- ST_Union - Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.
- ST_UpperLeftX - Retourne la coordonnée X du coin supérieur gauche du raster projeté dans le système de référence spatial.
- ST_UpperLeftY - Retourne la coordonnée Y du coin supérieur gauche du raster projeté dans le système de référence spatial.

- ST_Value - Retourne la valeur d'une bande raster spécifiée au pixel donné par columnx, rowy, ou à un point géométrique spécifié. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié. Si exclude_nodata_value vaut false, tous les pixels y compris ceux ayant la valeur nodata sont considérés comme intersectés et leur valeur sera retournée. Si exclude_nodata_value n'est pas spécifié, la valeur est lue depuis les méta-données du raster.
- ST_ValueCount - Retourne un ensemble d'enregistrements contenant une valeur de pixels et le nombre de pixels ayant cette valeur dans la bande du raster spécifié (ou de la couverture raster). Si aucune bande n'est spécifiée, la bande 1 est utilisée. Par défaut, les pixels de valeur nodata ne sont pas comptés, et toutes les autres valeurs sont retournées, avec leur valeur arrondies à l'entier le plus proche.
- ST_Width - Retourne la largeur du raster en pixels.
- ST_Within - Retourne true si aucun point du raster rastA ne se trouve à l'extérieur du raster rastB et si au moins un point de l'intérieur du raster rastA se trouve à l'intérieur du raster rastB.
- ST_WorldToRasterCoord - Retourne le coin supérieur gauche comme colonne et ligne, en fonction de coordonnées géométriques X et Y (longitude et latitude) ou un point géométrique dans le système de référence spatial du raster.
- ST_WorldToRasterCoordX - Retourne la colonne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw) exprimés dans le système de référence spatial du raster.
- ST_WorldToRasterCoordY - Retourne la ligne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw) exprimés dans le système de référence spatial du raster.
- UpdateRasterSRID - Change le SRID de tous les rasters dans la table et colonne en paramètres.

13.6 Fonctions PostGIS de dump Geometry / Geography / Raster

Les fonctions ci-dessous sont des fonctions PostGIS qui prennent en entrée ou renvoient en sortie un ensemble ou un seul objet de type geometry_dump ou geomval.

- ST_DumpAsPolygons - Retourne un ensemble d'enregistrements de type geomval (geom, val), à partir d'une bande raster donnée. Si aucune bande bandnum n'est spécifiée, la bande 1 est utilisée.
- ST_Intersection - Retourne un raster ou un ensemble de paires (géométrie, valeur de pixel) représentant la partie partagée de deux rasters ou l'intersection géométrique d'une vectorisation du raster et d'une géométrie.
- ST_Dump - Renvoie un ensemble de lignes geometry_dump pour les composants d'une géométrie.
- ST_DumpPoints - Renvoie un ensemble de lignes geometry_dump pour les coordonnées dans une géométrie.
- ST_DumpRings - Renvoie un ensemble de lignes geometry_dump pour les anneaux extérieurs et intérieurs d'un polygone.
- ST_DumpSegments - Renvoie un ensemble de lignes geometry_dump pour les segments d'une géométrie.

13.7 Fonctions Box de PostGIS

Les fonctions ci-dessous sont des fonctions PostGIS qui prennent en entrée ou renvoient en sortie la famille box* des types spatiaux PostGIS. La famille de types box comprend box2d, et box3d

- Box2D - Renvoie une BOX2D représentant l'étendue 2D d'une géométrie.
- Box3D - Renvoie une BOX3D représentant l'étendue 3D d'une géométrie.
- Box3D - Retourne la représentation 3d de la boîte englobante du raster.
- ST_3DExtent - Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.
- ST_3DMakeBox - Crée un BOX3D défini par deux géométries de points 3D.

- ST_AsMVTGeom - Transforme une géométrie dans l'espace de coordonnées d'une tuile MVT.
- ST_AsTWKB - Renvoie la géométrie sous forme de TWKB, diminutif de "Tiny Well-Known Binary"
- ST_Box2dFromGeoHash - Retourne une BOX2D à partir d'une chaîne GeoHash.
- ST_ClipByBox2D - Calcule la partie d'une géométrie située à l'intérieur d'un rectangle.
- ST_EstimatedExtent - Renvoie l'étendue estimée d'une table spatiale.
- ST_Expand - Renvoie une boîte de délimitation développée à partir d'une autre boîte de délimitation ou d'une géométrie.
- ST_Extent - Fonction agrégée qui renvoie la boîte de délimitation des géométries.
- ST_MakeBox2D - Crée un BOX2D défini par deux géométries de points 2D.
- ST_RemoveIrrelevantPointsForView - Removes points that are irrelevant for rendering a specific rectangluar view of a geometry.
- ST_XMax - Retourne les maxima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_XMin - Retourne les minima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_YMax - Retourne les maxima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_YMin - Retourne les minima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_ZMax - Retourne les maxima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_ZMin - Retourne les minima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- RemoveUnusedPrimitives - Supprime les primitives topologiques qui ne sont pas nécessaires pour définir les objets TopoGeometry existants.
- ValidateTopology - Renvoie un ensemble d'objets validatetopology_returntype détaillant les problèmes liés à la topologie.
- ~(box2df,box2df) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient une autre boîte de délimitation de précision flottante 2D (BOX2DF).
- ~(box2df,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient la boîte de délimitation 2D d'une géométrie.
- ~(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D d'une géométrie contient une boîte de délimitation de précision flottante 2D (GIDX).
- @(box2df,box2df) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans une autre boîte de délimitation de précision flottante 2D.
- @(box2df,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans la boîte de délimitation 2D d'une géométrie.
- @(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D d'une géométrie est contenue dans une boîte de délimitation 2D à précision flottante (BOX2DF).
- &&(box2df,box2df) - Renvoie TRUE si deux boîtes de délimitation 2D à précision flottante (BOX2DF) se croisent.
- &&(box2df,geometry) - Renvoie TRUE si une boîte de délimitation 2D de précision flottante (BOX2DF) intersecte la boîte de délimitation 2D (mise en cache) d'une géométrie.
- &&(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D (en cache) d'une géométrie intersecte une boîte de délimitation 2D de précision flottante (BOX2DF).

13.8 Fonctions PostGIS supportant la 3D

Les fonctions ci-dessous sont des fonctions PostGIS qui n'éliminent pas l'indice Z.

- AddGeometryColumn - Ajoute une colonne de géométrie à une table existante.
- Box3D - Renvoie une BOX3D représentant l'étendue 3D d'une géométrie.
- CG_3DArea - Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- CG_3DConvexHull - Calcule l'enveloppe convexe 3D d'une géométrie.
- CG_3DDifference - Effectuer une différence 3D
- CG_3DIntersection - Réaliser une intersection 3D
- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- CG_ApproximateMedialAxis - Calculer l'axe médian approximatif d'une géométrie aréolaire.
- CG_ConstrainedDelaunayTriangles - Renvoie une triangulation de Delaunay contrainte autour de la géométrie d'entrée donnée.
- CG_Extrude - Extruder une surface vers un volume
- CG_ForceLHR - Force l'orientation LHR d'un objet
- CG_IsPlanar - Vérifie si une surface est planaire ou non
- CG_IsSolid - Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.
- CG_MakeSolid - Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.
- CG_Orientation - Détermine l'orientation d'une surface
- CG_StraightSkeleton - Calcule un squelette (straight skeleton) à partir d'une géométrie
- CG_Tesselate - Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS
- CG_Visibility - Calculer un polygone de visibilité à partir d'un point ou d'un segment dans une géométrie polygonale
- CG_Volume - Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- DropGeometryColumn - Supprime une colonne géométrique d'une table spatiale.
- GeometryType - Renvoie le type d'une géométrie sous forme de texte.
- ST_3DArea - Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- ST_3DClosestPoint - Renvoie le point 3D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne 3D la plus courte.
- ST_3DConvexHull - Calcule l'enveloppe convexe 3D d'une géométrie.
- ST_3DDFullyWithin - Teste si deux géométries 3D sont entièrement comprises dans une distance 3D donnée
- ST_3DDWithin - Teste si deux géométries 3D se trouvent à une distance 3D donnée
- ST_3DDifference - Effectuer une différence 3D
- ST_3DDistance - Renvoie la distance cartésienne minimale en 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DExtent - Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.

- ST_3DIntersection - Réaliser une intersection 3D
- ST_3DIntersects - Teste si deux géométries se croisent dans l'espace en 3D - uniquement pour les points, les lignes, les polygones, les surfaces polyédriques (aire)
- ST_3DLength - Renvoie la longueur 3D d'une géométrie linéaire.
- ST_3DLineInterpolatePoint - Renvoie un point interpolé le long d'une ligne 3D à un emplacement fractionnaire.
- ST_3DLongestLine - Renvoie la ligne 3D la plus longue entre deux géométries
- ST_3DMaxDistance - Renvoie la distance maximale cartésienne 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DPerimeter - Renvoie le périmètre 3D d'une géométrie polygonale.
- ST_3DShortestLine - Renvoie la ligne 3D la plus courte entre deux géométries
- ST_3DUnion - Effectuer l'union 3D.
- ST_AddMeasure - Interpole les mesures le long d'une géométrie linéaire.
- ST_AddPoint - Ajoute un point à une LineString.
- ST_Affine - Appliquer une transformation affine 3D à une géométrie.
- ST_ApproximateMedialAxis - Calculer l'axe médian approximatif d'une géométrie aréolaire.
- ST_AsBinary - Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsEWKB - Renvoie la représentation Extended Well-Known Binary (EWKB) de la géométrie avec les métadonnées SRID.
- ST_AsEWKT - Renvoie la représentation Well-Known Text (WKT) de la géométrie avec les métadonnées SRID.
- ST_AsGML - Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsGeoJSON - Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_AsHEXEWKB - Renvoie une géométrie au format HEXEWKB (en tant que texte) en utilisant l'encodage little-endian (NDR) ou big-endian (XDR).
- ST_AsKML - Renvoyer la géométrie sous forme d'élément KML.
- ST_AsX3D - Renvoie une géométrie au format X3D xml node element : ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Boundary - Renvoie la limite d'une géométrie.
- ST_BoundingDiagonal - Retourne la diagonale de la boîte englobante pour la géométrie en argument.
- ST_CPAWithin - Teste si le point d'approche le plus proche de deux trajectoires se trouve dans la distance spécifiée.
- ST_ChaikinSmoothing - Renvoie une version lissée d'une géométrie, en utilisant l'algorithme Chaikin
- ST_ClosestPointOfApproach - Renvoie une mesure au point d'approche le plus proche de deux trajectoires.
- ST_Collect - Crée une géométrie GeometryCollection ou Multi à partir d'un ensemble de géométries.
- ST_ConstrainedDelaunayTriangles - Renvoie une triangulation de Delaunay contrainte autour de la géométrie d'entrée donnée.
- ST_ConvexHull - Calcule l'enveloppe convexe d'une géométrie.
- ST_CoordDim - Renvoie la dimension des coordonnées d'une géométrie.
- ST_CurveN - Renvoie la Nième courbe composante d'une CompoundCurve.
- ST_CurveToLine - Convertit une géométrie contenant des courbes en une géométrie linéaire.

- ST_DelaunayTriangles - Renvoie la triangulation de Delaunay des sommets d'une géométrie.
- ST_Difference - Calcule une géométrie représentant la partie de la géométrie A qui n'intersecte pas la géométrie B.
- ST_DistanceCPA - Renvoie la distance entre le point d'approche le plus proche de deux trajectoires.
- ST_Dump - Renvoie un ensemble de lignes geometry_dump pour les composants d'une géométrie.
- ST_DumpPoints - Renvoie un ensemble de lignes geometry_dump pour les coordonnées dans une géométrie.
- ST_DumpRings - Renvoie un ensemble de lignes geometry_dump pour les anneaux extérieurs et intérieurs d'un polygone.
- ST_DumpSegments - Renvoie un ensemble de lignes geometry_dump pour les segments d'une géométrie.
- ST_EndPoint - Renvoie le dernier point d'une LineString ou CircularLineString.
- ST_ExteriorRing - Renvoie une ligne représentant l'anneau extérieur d'un polygone.
- ST_Extrude - Extruder une surface vers un volume
- ST_FlipCoordinates - Renvoie une version d'une géométrie dont les axes X et Y sont inversés.
- ST_Force2D - Forcer les géométries à passer en "mode bidimensionnel".
- ST_ForceCurve - Retransformation d'une géométrie dans son type de courbure, le cas échéant.
- ST_ForceLHR - Force l'orientation LHR d'un objet
- ST_ForcePolygonCCW - Oriente tous les anneaux extérieurs dans le sens inverse des aiguilles d'une montre et tous les anneaux intérieurs dans le sens des aiguilles d'une montre.
- ST_ForcePolygonCW - Oriente tous les anneaux extérieurs dans le sens des aiguilles d'une montre et tous les anneaux intérieurs dans le sens inverse des aiguilles d'une montre.
- ST_ForceRHR - Force l'orientation des sommets d'un polygone à suivre la règle de la main droite.
- ST_ForceSFS - Forcer les géométries à utiliser uniquement les types de géométrie SFS 1.1.
- ST_Force3D - Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ.
- ST_Force3DZ - Forcer les géométries en mode XYZ.
- ST_Force4D - Forcer les géométries en mode XYZM.
- ST_ForceCollection - Convertir la géométrie en une GEOMETRYCOLLECTION.
- ST_GeomFromEWKB - Retourne un objet ST_Geometry à partir de sa représentation binaire étendue (Extended Well-Known Binary representation, EWKB).
- ST_GeomFromEWKT - Retourne un objet ST_Geometry à partir de sa représentation textuelle étendue (Extended Well-Known Text representation, EWKT).
- ST_GeomFromGML - Prend en paramètre une représentation GML d'une géométrie et renvoie un objet PostGIS de type geometry
- ST_GeomFromGeoJSON - Prend en entrée une géométrie au format geojson et renvoie un objet Postgis de type geometry
- ST_GeomFromKML - Prend en entrée une géométrie au format KML et renvoie un objet Postgis de type geometry
- ST_GeometricMedian - Renvoie la médiane géométrique d'un MultiPoint.
- ST_GeometryN - Renvoie un élément d'une collection de géométries.
- ST_GeometryType - Renvoie le type SQL-MM d'une géométrie sous forme de texte.
- ST_HasArc - Teste si une géométrie contient un arc de cercle
- ST_HasM - Vérifie si une géométrie a une dimension M (mesure).

- ST_HasZ - Vérifie si une géométrie possède une dimension Z.
- ST_InteriorRingN - Renvoie le Nième anneau intérieur (trou) d'un polygone.
- ST_InterpolatePoint - Renvoie la mesure interpolée d'une géométrie la plus proche d'un point.
- ST_Intersection - Calcule une géométrie représentant la partie partagée des géométries A et B.
- ST_IsClosed - Teste si les points de départ et d'arrivée d'une LineString coïncident. Pour une PolyhedralSurface, teste si elle est fermée (volumétrique).
- ST_IsCollection - Teste si une géométrie est un type de collection de géométrie.
- ST_IsPlanar - Vérifie si une surface est planaire ou non
- ST_IsPolygonCCW - Teste si les polygones ont des anneaux extérieurs orientés dans le sens inverse des aiguilles d'une montre et des anneaux intérieurs orientés dans le sens des aiguilles d'une montre.
- ST_IsPolygonCW - Teste si les polygones ont des anneaux extérieurs orientés dans le sens des aiguilles d'une montre et des anneaux intérieurs orientés dans le sens inverse des aiguilles d'une montre.
- ST_IsSimple - Teste si une géométrie n'a pas de points d'auto-intersection ou d'auto-tangente.
- ST_IsSolid - Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.
- ST_IsValidTrajectory - Teste si la géométrie est une trajectoire valide.
- ST_LengthSpheroid - Renvoie la longueur/périmètre 2D ou 3D d'une géométrie lon/lat sur un sphéroïde.
- ST_LineFromMultiPoint - Crée une LineString à partir d'une géométrie MultiPoint.
- ST_LineInterpolatePoint - Renvoie un point interpolé le long d'une ligne à un emplacement fractionnaire.
- ST_LineInterpolatePoints - Renvoie des points interpolés le long d'une ligne à un intervalle fractionnaire.
- ST_LineSubstring - Renvoie la partie d'une ligne située entre deux emplacements fractionnaires.
- ST_LineToCurve - Convertit une géométrie linéaire en une géométrie courbe.
- ST_LocateBetweenElevations - Renvoie les parties d'une géométrie qui se trouvent dans un intervalle d'élévation (Z).
- ST_M - Renvoie la coordonnée M d'un point.
- ST_MakeLine - Crée une LineString à partir de géométries Point, MultiPoint ou LineString.
- ST_MakePoint - Crée un point 2D, 3DZ ou 4D.
- ST_MakePolygon - Crée un polygone à partir d'une collection et d'une liste facultative de trous.
- ST_MakeSolid - Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.
- ST_MakeValid - Tente de rendre valide une géométrie invalide sans perdre de sommets.
- ST_MemSize - Renvoie la quantité d'espace mémoire que prend une géométrie.
- ST_MemUnion - Fonction d'agrégation qui fusionne les géométries d'une manière efficace sur le plan de la mémoire mais plus lente
- ST_NDims - Renvoie la dimension des coordonnées d'une géométrie.
- ST_NPoints - Retourne le nombre de points (vertex) d'un objet géométrique.
- ST_NRings - Renvoie le nombre d'anneaux dans une géométrie polygonale.
- ST_Node - Nœuds d'une collection de lignes.
- ST_NumCurves - Renvoie le nombre de courbes composantes d'une CompoundCurve.

- ST_NumGeometries - Renvoie le nombre d'éléments dans une collection de géométrie.
- ST_NumPatches - Renvoie le nombre de faces d'une surface polyédrique. Retourne null pour les géométries non polyédriques.
- ST_Orientation - Détermine l'orientation d'une surface
- ST_PatchN - Renvoie la Nième géométrie (face) d'une PolyhedralSurface.
- ST_PointFromWKB - Construit une géométrie depuis la représentation binaire WKB et le SRID donné
- ST_PointN - Renvoie le Nième point de la première LineString ou LineString circulaire d'une géométrie.
- ST_PointOnSurface - Calcule un point dont on garantit qu'il se trouve dans un polygone ou sur une géométrie.
- ST_Points - Renvoie un MultiPoint contenant les coordonnées d'une géométrie.
- ST_Polygon - Crée un polygone à partir d'une LineString avec un SRID spécifié.
- ST_RemovePoint - Supprime un point d'une ligne.
- ST_RemoveRepeatedPoints - Renvoie une version d'une géométrie dont les points en double ont été supprimés.
- ST_Reverse - Retourne la géométrie avec l'ordre des sommets inversé.
- ST_Rotate - Fait pivoter une géométrie autour d'un point d'origine.
- ST_RotateX - Fait pivoter une géométrie autour de l'axe X.
- ST_RotateY - Fait pivoter une géométrie autour de l'axe Y.
- ST_RotateZ - Fait pivoter une géométrie autour de l'axe Z.
- ST_Scale - Met à l'échelle une géométrie en fonction de facteurs donnés.
- ST_Scroll - Modifier le point de départ d'une LineString fermée.
- ST_SetPoint - Remplacer le point d'une ligne par un point donné.
- ST_ShiftLongitude - Décale les coordonnées de longitude d'une géométrie entre -180..180 et 0..360.
- ST_SnapToGrid - Accrocher tous les points de la géométrie d'entrée à une grille régulière.
- ST_StartPoint - Renvoie le premier point d'une LineString.
- ST_StraightSkeleton - Calcule un squelette (straight skeleton) à partir d'une géométrie
- ST_SwapOrdinates - Renvoie une version de la géométrie donnée avec les valeurs d'ordonnées permutées.
- ST_SymDifference - Calcule une géométrie représentant les parties des géométries A et B qui ne s'intersectent pas.
- ST_Tesselate - Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS
- ST_TransScale - Traduit et met à l'échelle une géométrie en fonction des paramètres offset et factor spécifiés.
- ST_Translate - Traduit une géométrie en fonction de décalages donnés.
- ST_UnaryUnion - Calcule l'union des composantes d'une seule géométrie.
- ST_Union - Calcule une géométrie représentant l'union des ensembles de points des géométries d'entrée.
- ST_Volume - Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- ST_WrapX - Enveloppe une géométrie autour d'une valeur X.
- ST_X - Renvoie la coordonnée X d'un point.
- ST_XMax - Retourne les maxima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.

- ST_XMin - Retourne les minima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_Y - Renvoie la coordonnée Y d'un point.
- ST_YMax - Retourne les maxima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_YMin - Retourne les minima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_Z - Renvoie la coordonnée Z d'un point.
- ST_ZMax - Retourne les maxima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_ZMin - Retourne les minima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_Zmflag - Retourne un code indiquant la dimension des coordonnées ZM d'une géométrie.
- Equals - Retourne vrai si deux topogeometries sont composées des mêmes primitives topologiques.
- Intersects - Retourne true si une paire de primitives des deux topogeometries s'intersectent.
- UpdateGeometrySRID - Met à jour le SRID de tous les éléments d'une colonne géométrique et les métadonnées de la table.
- &&& - Renvoie TRUE si la boîte de délimitation n-D de A intersecte la boîte de délimitation n-D de B.
- &&&(geometry,gidx) - Renvoie TRUE si la boîte de délimitation n-D (en cache) d'une géométrie intersecte une boîte de délimitation de précision flottante n-D (GIDX).
- &&&(gidx,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante n-D (GIDX) intersecte la boîte de délimitation n-D (mise en cache) d'une géométrie.
- &&&(gidx,gidx) - Renvoie TRUE si deux boîtes de délimitation (GIDX) de précision flottante n-D se croisent.

13.9 Fonctions d'aide aux géométries courbes de PostGIS

Les fonctions ci-dessous sont des fonctions PostGIS qui peuvent utiliser CIRCULARSTRING, CURVEPOLYGON et d'autres types de géométrie courbe

- AddGeometryColumn - Ajoute une colonne de géométrie à une table existante.
- Box2D - Renvoie une BOX2D représentant l'étendue 2D d'une géométrie.
- Box3D - Renvoie une BOX3D représentant l'étendue 3D d'une géométrie.
- DropGeometryColumn - Supprime une colonne géométrique d'une table spatiale.
- GeometryType - Renvoie le type d'une géométrie sous forme de texte.
- PostGIS_AddBBox - Ajoute une bounding box à la géométrie.
- PostGIS_DropBBox - Supprime le cache de la boîte de délimitation de la géométrie.
- PostGIS_HasBBox - Renvoie TRUE si la bbox de cette géométrie est en cache, sinon FALSE.
- ST_3DExtent - Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.
- ST_Affine - Appliquer une transformation affine 3D à une géométrie.
- ST_AsBinary - Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsEWKB - Renvoie la représentation Extended Well-Known Binary (EWKB) de la géométrie avec les métadonnées SRID.
- ST_AsEWKT - Renvoie la représentation Well-Known Text (WKT) de la géométrie avec les métadonnées SRID.

- ST_AsHEXEWKB - Renvoie une géométrie au format HEXEWKB (en tant que texte) en utilisant l'encodage little-endian (NDR) ou big-endian (XDR).
- ST_AsSVG - Renvoie les données de chemin SVG pour une géométrie.
- ST_AsText - Renvoie la représentation Well-Known Text (WKT) de la géométrie/geography sans métadonnées SRID.
- ST_ClusterDBSCAN - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie d'entrée en utilisant l'algorithme DBSCAN.
- ST_ClusterWithin - Fonction agrégée qui regroupe les géométries en fonction de la distance de séparation.
- ST_ClusterWithinWin - Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, regroupement en utilisant la distance de séparation.
- ST_Collect - Crée une géométrie GeometryCollection ou Multi à partir d'un ensemble de géométries.
- ST_CoordDim - Renvoie la dimension des coordonnées d'une géométrie.
- ST_CurveToLine - Convertit une géométrie contenant des courbes en une géométrie linéaire.
- ST_Distance - Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_Dump - Renvoie un ensemble de lignes geometry_dump pour les composants d'une géométrie.
- ST_DumpPoints - Renvoie un ensemble de lignes geometry_dump pour les coordonnées dans une géométrie.
- ST_EndPoint - Renvoie le dernier point d'une LineString ou CircularLineString.
- ST_EstimatedExtent - Renvoie l'étendue estimée d'une table spatiale.
- ST_FlipCoordinates - Renvoie une version d'une géométrie dont les axes X et Y sont inversés.
- ST_Force2D - Forcer les géométries à passer en "mode bidimensionnel".
- ST_ForceCurve - Retransformation d'une géométrie dans son type de courbure, le cas échéant.
- ST_ForceSFS - Forcer les géométries à utiliser uniquement les types de géométrie SFS 1.1.
- ST_Force3D - Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ.
- ST_Force3DM - Forcer les géométries en mode XYM.
- ST_Force3DZ - Forcer les géométries en mode XYZ.
- ST_Force4D - Forcer les géométries en mode XYZM.
- ST_ForceCollection - Convertir la géométrie en une GEOMETRYCOLLECTION.
- ST_GeoHash - Retourne une représentation GeoHash de la géométrie.
- ST_GeogFromWKB - Retourne un objet de type geography à partir de sa représentation binaire Well-Know Binary (WKB ou EWKB).
- ST_GeomFromEWKB - Retourne un objet ST_Geometry à partir de sa représentation binaire étendue (Extended Well-Known Binary representation, EWKB).
- ST_GeomFromEWKT - Retourne un objet ST_Geometry à partir de sa représentation textuelle étendue (Extended Well-Known Text representation, EWKT).
- ST_GeomFromText - Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT).
- ST_GeomFromWKB - Retourne un objet de type geometry à partir de sa représentation binaire Well-Know Binary (WKB) et d'un SRID optionnel.
- ST_GeometryN - Renvoie un élément d'une collection de géométries.

- = - Renvoie TRUE si les coordonnées et l'ordre des coordonnées de la géométrie/géographie A sont les mêmes que les coordonnées et l'ordre des coordonnées de la géométrie/géographie B.
- &<| - Renvoie TRUE si la boîte englobante de A chevauche ou est inférieure à celle de B.
- ST_HasArc - Teste si une géométrie contient un arc de cercle
- ST_Intersects - Teste si deux géométries se croisent (elles ont au moins un point en commun)
- ST_IsClosed - Teste si les points de départ et d'arrivée d'une LineString coïncident. Pour une PolyhedralSurface, teste si elle est fermée (volumétrique).
- ST_IsCollection - Teste si une géométrie est un type de collection de géométrie.
- ST_IsEmpty - Teste si une géométrie est vide.
- ST_LineToCurve - Convertit une géométrie linéaire en une géométrie courbe.
- ST_MemSize - Renvoie la quantité d'espace mémoire que prend une géométrie.
- ST_NPoints - Retourne le nombre de points (vertex) d'un objet géométrique.
- ST_NRings - Renvoie le nombre d'anneaux dans une géométrie polygonale.
- ST_PointFromWKB - Construit une géométrie depuis la représentation binaire WKB et le SRID donné
- ST_PointN - Renvoie le Nième point de la première LineString ou LineString circulaire d'une géométrie.
- ST_Points - Renvoie un MultiPoint contenant les coordonnées d'une géométrie.
- ST_Rotate - Fait pivoter une géométrie autour d'un point d'origine.
- ST_RotateZ - Fait pivoter une géométrie autour de l'axe Z.
- ST_SRID - Renvoie l'identifiant de référence spatiale d'une géométrie.
- ST_Scale - Met à l'échelle une géométrie en fonction de facteurs donnés.
- ST_SetSRID - Définir le SRID d'une géométrie.
- ST_StartPoint - Renvoie le premier point d'une LineString.
- ST_Summary - Renvoie un résumé textuel du contenu d'une géométrie.
- ST_SwapOrdinates - Renvoie une version de la géométrie donnée avec les valeurs d'ordonnées permutées.
- ST_TransScale - Traduit et met à l'échelle une géométrie en fonction des paramètres offset et factor spécifiés.
- ST_Transform - Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent.
- ST_Translate - Traduit une géométrie en fonction de décalages donnés.
- ST_XMax - Retourne les maxima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_XMin - Retourne les minima X d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_YMax - Retourne les maxima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_YMin - Retourne les minima Y d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_ZMax - Retourne les maxima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_ZMin - Retourne les minima Z d'une boîte de délimitation 2D ou 3D ou d'une géométrie.
- ST_Zmflag - Retourne un code indiquant la dimension des coordonnées ZM d'une géométrie.
- UpdateGeometrySRID - Met à jour le SRID de tous les éléments d'une colonne géométrique et les métadonnées de la table.

- ~(box2df,box2df) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient une autre boîte de délimitation de précision flottante 2D (BOX2DF).
- ~(box2df,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient la boîte de délimitation 2D d'une géométrie.
- ~(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D d'une géométrie contient une boîte de délimitation de précision flottante 2D (GIDX).
- && - Renvoi VRAI si la boite englobante 2D de A intersecte la boite englobante 2D de B.
- &&& - Renvoie TRUE si la boîte de délimitation n-D de A intersecte la boîte de délimitation n-D de B.
- @(box2df,box2df) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans une autre boîte de délimitation de précision flottante 2D.
- @(box2df,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans la boîte de délimitation 2D d'une géométrie.
- @(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D d'une géométrie est contenue dans une boîte de délimitation 2D à précision flottante (BOX2DF).
- &&(box2df,box2df) - Renvoie TRUE si deux boîtes de délimitation 2D à précision flottante (BOX2DF) se croisent.
- &&(box2df,geometry) - Renvoie TRUE si une boîte de délimitation 2D de précision flottante (BOX2DF) intersecte la boîte de délimitation 2D (mise en cache) d'une géométrie.
- &&(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D (en cache) d'une géométrie intersecte une boîte de délimitation 2D de précision flottante (BOX2DF).
- &&&(geometry,gidx) - Renvoie TRUE si la boîte de délimitation n-D (en cache) d'une géométrie intersecte une boîte de délimitation de précision flottante n-D (GIDX).
- &&&(gidx,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante n-D (GIDX) intersecte la boîte de délimitation n-D (mise en cache) d'une géométrie.
- &&&(gidx,gidx) - Renvoie TRUE si deux boîtes de délimitation (GIDX) de précision flottante n-D se croisent.

13.10 Fonctions de support des surfaces polyédriques de PostGIS

Les fonctions ci-dessous sont des fonctions PostGIS qui peuvent utiliser les géométries POLYHEDRALSURFACE, POLYHEDRALSURFACEM

- Box2D - Renvoie une BOX2D représentant l'étendue 2D d'une géométrie.
- Box3D - Renvoie une BOX3D représentant l'étendue 3D d'une géométrie.
- CG_3DArea - Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- CG_3DConvexHull - Calcule l'enveloppe convexe 3D d'une géométrie.
- CG_3DDifference - Effectuer une différence 3D
- CG_3DIntersection - Réaliser une intersection 3D
- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- CG_ApproximateMedialAxis - Calculer l'axe médian approximatif d'une géométrie aréolaire.
- CG_Extrude - Extruder une surface vers un volume
- CG_ForceLHR - Force l'orientation LHR d'un objet

- CG_IsPlanar - Vérifie si une surface est planaire ou non
- CG_IsSolid - Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.
- CG_MakeSolid - Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.
- CG_StraightSkeleton - Calcule un squelette (straight skeleton) à partir d'une géométrie
- CG_Tesselate - Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS
- CG_Visibility - Calculer un polygone de visibilité à partir d'un point ou d'un segment dans une géométrie polygonale
- CG_Volume - Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- GeometryType - Renvoie le type d'une géométrie sous forme de texte.
- ST_3DArea - Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- ST_3DClosestPoint - Renvoie le point 3D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne 3D la plus courte.
- ST_3DConvexHull - Calcule l'enveloppe convexe 3D d'une géométrie.
- ST_3DDFullyWithin - Teste si deux géométries 3D sont entièrement comprises dans une distance 3D donnée
- ST_3DDWithin - Teste si deux géométries 3D se trouvent à une distance 3D donnée
- ST_3DDifference - Effectuer une différence 3D
- ST_3DDistance - Renvoie la distance cartésienne minimale en 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DExtent - Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.
- ST_3DIntersection - Réaliser une intersection 3D
- ST_3DIntersects - Teste si deux géométries se croisent dans l'espace en 3D - uniquement pour les points, les lignes, les polygones, les surfaces polyédriques (aire)
- ST_3DLongestLine - Renvoie la ligne 3D la plus longue entre deux géométries
- ST_3DMaxDistance - Renvoie la distance maximale cartésienne 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DShortestLine - Renvoie la ligne 3D la plus courte entre deux géométries
- ST_3DUnion - Effectuer l'union 3D.
- ST_Affine - Appliquer une transformation affine 3D à une géométrie.
- ST_ApproximateMedialAxis - Calculer l'axe médian approximatif d'une géométrie aréolaire.
- ST_Area - Renvoie l'aire d'une géométrie polygonale.
- ST_AsBinary - Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsEWKB - Renvoie la représentation Extended Well-Known Binary (EWKB) de la géométrie avec les métadonnées SRID.
- ST_AsEWKT - Renvoie la représentation Well-Known Text (WKT) de la géométrie avec les métadonnées SRID.
- ST_AsGML - Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsX3D - Renvoie une géométrie au format X3D xml node element : ISO-IEC-19776-1.2-X3DEncodings-XML

- ST_CoordDim - Renvoie la dimension des coordonnées d'une géométrie.
- ST_Dimension - Renvoie la dimension topologique d'une géométrie.
- ST_Dump - Renvoie un ensemble de lignes geometry_dump pour les composants d'une géométrie.
- ST_DumpPoints - Renvoie un ensemble de lignes geometry_dump pour les coordonnées dans une géométrie.
- ST_Expand - Renvoie une boîte de délimitation développée à partir d'une autre boîte de délimitation ou d'une géométrie.
- ST_Extent - Fonction agrégée qui renvoie la boîte de délimitation des géométries.
- ST_Extrude - Extruder une surface vers un volume
- ST_FlipCoordinates - Renvoie une version d'une géométrie dont les axes X et Y sont inversés.
- ST_Force2D - Forcer les géométries à passer en "mode bidimensionnel".
- ST_ForceLHR - Force l'orientation LHR d'un objet
- ST_ForceRHR - Force l'orientation des sommets d'un polygone à suivre la règle de la main droite.
- ST_ForceSFS - Forcer les géométries à utiliser uniquement les types de géométrie SFS 1.1.
- ST_Force3D - Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ.
- ST_Force3DZ - Forcer les géométries en mode XYZ.
- ST_ForceCollection - Convertir la géométrie en une GEOMETRYCOLLECTION.
- ST_GeomFromEWKB - Retourne un objet ST_Geometry à partir de sa représentation binaire étendue (Extended Well-Known Binary representation, EWKB).
- ST_GeomFromEWKT - Retourne un objet ST_Geometry à partir de sa représentation textuelle étendue (Extended Well-Known Text representation, EWKT).
- ST_GeomFromGML - Prend en paramètre une représentation GML d'une géométrie et renvoie un objet PostGIS de type geometry
- ST_GeometryN - Renvoie un élément d'une collection de géométries.
- ST_GeometryType - Renvoie le type SQL-MM d'une géométrie sous forme de texte.
- = - Renvoie TRUE si les coordonnées et l'ordre des coordonnées de la géométrie/géographie A sont les mêmes que les coordonnées et l'ordre des coordonnées de la géométrie/géographie B.
- &<| - Renvoie TRUE si la boîte englobante de A chevauche ou est inférieure à celle de B.
- ~= - Renvoie TRUE si la boîte de délimitation de A est la même que celle de B.
- ST_IsClosed - Teste si les points de départ et d'arrivée d'une LineString coïncident. Pour une PolyhedralSurface, teste si elle est fermée (volumétrique).
- ST_IsPlanar - Vérifie si une surface est planaire ou non
- ST_IsSolid - Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.
- ST_MakeSolid - Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.
- ST_MemSize - Renvoie la quantité d'espace mémoire que prend une géométrie.
- ST_NPoints - Retourne le nombre de points (vertex) d'un objet géométrique.
- ST_NumGeometries - Renvoie le nombre d'éléments dans une collection de géométrie.
- ST_NumPatches - Renvoie le nombre de faces d'une surface polyédrique. Retourne null pour les géométries non polyédriques.

- ST_PatchN - Renvoie la Nième géométrie (face) d'une PolyhedralSurface.
- ST_RemoveRepeatedPoints - Renvoie une version d'une géométrie dont les points en double ont été supprimés.
- ST_Reverse - Retourne la géométrie avec l'ordre des sommets inversé.
- ST_Rotate - Fait pivoter une géométrie autour d'un point d'origine.
- ST_RotateX - Fait pivoter une géométrie autour de l'axe X.
- ST_RotateY - Fait pivoter une géométrie autour de l'axe Y.
- ST_RotateZ - Fait pivoter une géométrie autour de l'axe Z.
- ST_Scale - Met à l'échelle une géométrie en fonction de facteurs donnés.
- ST_ShiftLongitude - Décale les coordonnées de longitude d'une géométrie entre -180..180 et 0..360.
- ST_StraightSkeleton - Calcule un squelette (straight skeleton) à partir d'une géométrie
- ST_Summary - Renvoie un résumé textuel du contenu d'une géométrie.
- ST_SwapOrdinates - Renvoie une version de la géométrie donnée avec les valeurs d'ordonnées permutées.
- ST_Tesselate - Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS
- ST_Transform - Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent.
- ST_Volume - Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- ~(box2df,box2df) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient une autre boîte de délimitation de précision flottante 2D (BOX2DF).
- ~(box2df,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient la boîte de délimitation 2D d'une géométrie.
- ~(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D d'une géométrie contient une boîte de délimitation de précision flottante 2D (GIDX).
- && - Renvoi VRAI si la boite englobante 2D de A intersecte la boite englobante 2D de B.
- &&& - Renvoie TRUE si la boîte de délimitation n-D de A intersecte la boîte de délimitation n-D de B.
- @(box2df,box2df) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans une autre boîte de délimitation de précision flottante 2D.
- @(box2df,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans la boîte de délimitation 2D d'une géométrie.
- @(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D d'une géométrie est contenue dans une boîte de délimitation 2D à précision flottante (BOX2DF).
- &&(box2df,box2df) - Renvoie TRUE si deux boîtes de délimitation 2D à précision flottante (BOX2DF) se croisent.
- &&(box2df,geometry) - Renvoie TRUE si une boîte de délimitation 2D de précision flottante (BOX2DF) intersecte la boîte de délimitation 2D (mise en cache) d'une géométrie.
- &&(geometry,box2df) - Renvoie TRUE si la boîte de délimitation 2D (en cache) d'une géométrie intersecte une boîte de délimitation 2D de précision flottante (BOX2DF).
- &&&(geometry,gidx) - Renvoie TRUE si la boîte de délimitation n-D (en cache) d'une géométrie intersecte une boîte de délimitation de précision flottante n-D (GIDX).
- &&&(gidx,geometry) - Renvoie TRUE si une boîte de délimitation de précision flottante n-D (GIDX) intersecte la boîte de délimitation n-D (mise en cache) d'une géométrie.
- &&&(gidx,gidx) - Renvoie TRUE si deux boîtes de délimitation (GIDX) de précision flottante n-D se croisent.

13.11 Matrice d'aide aux fonctions de PostGIS

Vous trouverez ci-dessous une liste alphabétique des fonctions spatiales spécifiques de PostGIS et des types de données spatiales qu'elles utilisent ou de la conformité OGC/SQL à laquelle elles tentent de se conformer.

- Un ✓ signifie que la fonction fonctionne nativement avec le type ou le sous-type.
- Un signifie qu'il fonctionne mais avec une transformation intégrée utilisant la géométrie, la transformation vers une référence spatiale "meilleur srid", puis la transformation à nouveau. Les résultats peuvent ne pas être conformes aux attentes pour les grandes zones ou les zones aux pôles et peuvent accumuler des erreurs en virgule flottante.
- Un signifie que la fonction fonctionne avec le type en raison d'un transfert automatique vers un autre type, tel que box3d, plutôt que d'une prise en charge directe du type.
- Un signifie que la fonction n'est disponible que si PostGIS a été compilé avec le support SFCGAL.
- geom - Support géométrique 2D de base (x,y).
- geog - Support géographique de base en 2D (x,y).
- 2.5D - géométries de base en 2D dans un espace 3 D/4D (avec coordonnée Z ou M).
- PS - Surfaces polyédriques
- T - Triangles et surfaces irrégulières triangulées (TIN)

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_Collect	✓		✓	✓			
ST_LineFromMulti	✓		✓				
ST_MakeEnvelope	✓						
ST_MakeLine	✓		✓				
ST_MakePoint	✓		✓				
ST_MakePointM	✓						
ST_MakePolygon	✓		✓				
ST_Point	✓				✓		
ST_PointZ	✓						
ST_PointM	✓						
ST_PointZM	✓						
ST_Polygon	✓		✓		✓		
ST_TileEnvelope	✓						
ST_HexagonGrid	✓						
ST_Hexagon	✓						
ST_SquareGrid	✓						
ST_Square	✓						

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_Letters	✓						
GeometryType	✓		✓	✓		✓	✓
ST_Boundary	✓		✓		✓		
ST_BoundingDiago	✓		✓				
ST_CoordDim	✓		✓	✓	✓	✓	✓
ST_Dimension	✓				✓	✓	✓
ST_Dump	✓		✓	✓		✓	✓
ST_DumpPoints	✓		✓	✓		✓	✓
ST_DumpSegments	✓		✓				✓
ST_DumpRings	✓		✓				
ST_EndPoint	✓		✓	✓	✓		
ST_Envelope	✓				✓		
ST_ExteriorRing	✓		✓		✓		
ST_GeometryN	✓		✓	✓	✓	✓	✓
ST_GeometryType	✓		✓		✓	✓	
ST_HasArc	✓		✓	✓			
ST_InteriorRingN	✓		✓		✓		
ST_NumCurves	✓		✓		✓		
ST_CurveN	✓		✓		✓		
ST_IsClosed	✓		✓	✓	✓	✓	
ST_IsCollection	✓		✓	✓			
ST_IsEmpty	✓			✓	✓		
ST_IsPolygonCCW	✓		✓				
ST_IsPolygonCW	✓		✓				
ST_IsRing	✓				✓		
ST_IsSimple	✓		✓		✓		
ST_M	✓		✓		✓		
ST_MemSize	✓		✓	✓		✓	✓
ST_NDims	✓		✓				
ST_NPoints	✓		✓	✓		✓	
ST_NRings	✓		✓	✓			
ST_NumGeometrie	✓		✓		✓	✓	✓
ST_NumInteriorRin	✓				✓		
ST_NumInteriorRin	✓						

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_NumPatches	✓		✓		✓	✓	
ST_NumPoints	✓				✓		
ST_PatchN	✓		✓		✓	✓	
ST_PointN	✓		✓	✓	✓		
ST_Points	✓		✓	✓			
ST_StartPoint	✓		✓	✓	✓		
ST_Summary	✓	✓		✓		✓	✓
ST_X	✓		✓		✓		
ST_Y	✓		✓		✓		
ST_Z	✓		✓		✓		
ST_Zmflag	✓		✓	✓			
ST_HasZ	✓		✓				
ST_HasM	✓		✓				
ST_AddPoint	✓		✓				
ST_CollectionExtra	✓						
ST_CollectionHom	✓ ize						
ST_CurveToLine	✓		✓	✓	✓		
ST_Scroll	✓		✓				
ST_FlipCoordinates	✓		✓	✓		✓	✓
ST_Force2D	✓		✓	✓		✓	
ST_Force3D	✓		✓	✓		✓	
ST_Force3DZ	✓		✓	✓		✓	
ST_Force3DM	✓			✓			
ST_Force4D	✓		✓	✓			
ST_ForceCollection	✓		✓	✓		✓	
ST_ForceCurve	✓		✓	✓			
ST_ForcePolygonC	✓		✓				
ST_ForcePolygonC	✓		✓				
ST_ForceSFS	✓		✓	✓		✓	✓
ST_ForceRHR	✓		✓			✓	
ST_LineExtend	✓						
ST_LineToCurve	✓		✓	✓			
ST_Multi	✓						
ST_Normalize	✓						

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_Project	✓	✓					
ST_QuantizeCoordi	✓	s					
ST_RemovePoint	✓		✓				
ST_RemoveRepeate	✓	ints	✓			✓	
ST_RemoveIrreleva	✓	intsForView					
ST_RemoveSmallPa	✓						
ST_Reverse	✓		✓			✓	
ST_Segmentize	✓	✓					
ST_SetPoint	✓		✓				
ST_ShiftLongitude	✓		✓			✓	✓
ST_WrapX	✓		✓				
ST_SnapToGrid	✓		✓				
ST_Snap	✓						
ST_SwapOrdinates	✓		✓	✓		✓	✓
ST_IsValid	✓				✓		
ST_IsValidDetail	✓						
ST_IsValidReason	✓						
ST_MakeValid	✓		✓				
ST_InverseTransfor	✓	peline					
ST_SetSRID	✓			✓			
ST_SRID	✓			✓	✓		
ST_Transform	✓			✓	✓	✓	
ST_TransformPipel	✓						
postgis_srs_codes							
postgis_srs							
postgis_srs_all							
postgis_srs_search	✓						
ST_BdPolyFromTe	✓						
ST_BdMPolyFromT	✓						
ST_GeogFromText		✓					
ST_GeographyFromText		✓					
ST_GeomCollFrom	✓				✓		
ST_GeomFromEWK	✓		✓	✓		✓	✓
ST_GeomFromMA	✓	1					
ST_GeometryFrom	✓				✓		

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_GeomFromText	✓			✓	✓		
ST_LineFromText	✓				✓		
ST_MLineFromTex	✓				✓		
ST_MPointFromTe	✓				✓		
ST_MPolyFromTex	✓				✓		
ST_PointFromText	✓				✓		
ST_PolygonFromTe	✓				✓		
ST_WKTToSQL	✓				✓		
ST_GeogFromWKB		✓		✓			
ST_GeomFromEWI	✓		✓	✓		✓	✓
ST_GeomFromWK	✓			✓	✓		
ST_LineFromWKB	✓				✓		
ST_LinestringFrom	✓ B				✓		
ST_PointFromWKI	✓		✓	✓	✓		
ST_WKBToSQL	✓				✓		
ST_Box2dFromGeo	✗ h						
ST_GeomFromGeo	✓ ı						
ST_GeomFromGM	✓		✓			✓	✓
ST_GeomFromGeo	✓ N		✓				
ST_GeomFromKM	✓		✓				
ST_GeomFromTWI	✓						
ST_GMLToSQL	✓				✓		
ST_LineFromEncoc	✓ olyline						
ST_PointFromGeoHash							
ST_FromFlatGeobufToTable							
ST_FromFlatGeobuf							
ST_AsEWKT	✓	✓	✓	✓		✓	✓
ST_AsText	✓	✓		✓	✓		
ST_AsBinary	✓	✓	✓	✓	✓	✓	✓
ST_AsEWKB	✓		✓	✓		✓	✓
ST_AsHEXEWKB	✓		✓	✓			
ST_AsEncodedPoly	✓						
ST_AsFlatGeobuf	✗						
ST_AsGeobuf	✗						
ST_AsGeoJSON	✓	✓	✓				

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_AsGML	✓	✓	✓		✓	✓	✓
ST_AsKML	✓	✓	✓				
ST_AsLatLonText	✓						
ST_AsMARC21	✓						
ST_AsMVTGeom	✓						
ST_AsMVT	✓						
ST_AsSVG	✓	✓		✓			
ST_AsTWKB	✓						
ST_AsX3D	✓		✓			✓	✓
ST_GeoHash	✓			✓			
&&	✓	✓		✓		✓	
&&(geometry,box2d	✓			✓		✓	
&&(box2df,geomet	✓			✓		✓	
&&(box2df,box2df)	✓			✓		✓	
&&&	✓		✓	✓		✓	✓
&&&(geometry,gid	✓		✓	✓		✓	✓
&&&(gidx,geometr	✓		✓	✓		✓	✓
&&&(gidx,gidx)			✓	✓		✓	✓
&<	✓						
&<|	✓			✓		✓	
&>	✓						
<<	✓						
<<|	✓						
=	✓	✓		✓		✓	
>>	✓						
@	✓						
@(geometry,box2df	✓			✓		✓	
@(box2df,geometry	✓			✓		✓	
@(box2df,box2df)	✓			✓		✓	
|&>	✓						
|>>	✓						
~	✓						
~(geometry,box2df)	✓			✓		✓	
~(box2df,geometry)	✓			✓		✓	

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
~(box2df,box2df)	✓			✓		✓	
~=	✓					✓	
<->	✓	✓					
|=|	✓						
<#>	✓						
<<->>	✓						
ST_3DIntersects	✓		✓		✓	✓	✓
ST_Contains	✓				✓		
ST_ContainsProper	✓						
ST_CoveredBy	✓	✓					
ST_Covers	✓	✓					
ST_Crosses	✓				✓		
ST_Disjoint	✓				✓		
ST_Equals	✓				✓		
ST_Intersects	✓	✓		✓	✓		✓
ST_LineCrossingDi ion	✓						
ST_OrderingEquals	✓				✓		
ST_Overlaps	✓				✓		
ST_Relate	✓				✓		
ST_RelateMatch							
ST_Touches	✓				✓		
ST_Within	✓				✓		
ST_3DDWithin	✓		✓		✓	✓	
ST_3DDFullyWithi	✓		✓			✓	
ST_DFullyWithin	✓						
ST_DWithin	✓	✓					
ST_PointInsideCirc	✓						
ST_Area	✓	✓			✓	✓	
ST_Azimuth	✓	✓					
ST_Angle	✓						
ST_ClosestPoint	✓	✓					
ST_3DClosestPoint	✓		✓			✓	
ST_Distance	✓	✓		✓	✓		
ST_3DDistance	✓		✓		✓	✓	

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_DistanceSphere	✓						
ST_DistanceSphero	✓						
ST_FrechetDistance	✓						
ST_HausdorffDistar	✓						
ST_Length	✓	✓			✓		
ST_Length2D	✓						
ST_3DLength	✓		✓		✓		
ST_LengthSpheroid	✓		✓				
ST_LongestLine	✓						
ST_3DLongestLine	✓		✓			✓	
ST_MaxDistance	✓						
ST_3DMaxDistance	✓		✓			✓	
ST_MinimumClear	✓						
ST_MinimumClear	✓ Line						
ST_Perimeter	✓	✓			✓		
ST_Perimeter2D	✓						
ST_3DPerimeter	✓		✓		✓		
ST_ShortestLine	✓	✓					
ST_3DShortestLine	✓		✓			✓	
ST_ClipByBox2D	✓						
ST_Difference	✓		✓		✓		
ST_Intersection	✓		✓		✓		
ST_MemUnion	✓		✓				
ST_Node	✓		✓				
ST_Split	✓						
ST_Subdivide	✓						
ST_SymDifference	✓		✓		✓		
ST_UnaryUnion	✓		✓				
ST_Union	✓		✓		✓		
ST_Buffer	✓				✓		
ST_BuildArea	✓						
ST_Centroid	✓	✓			✓		
ST_ChaikinSmooth	✓		✓				
ST_ConcaveHull	✓						

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_ConvexHull	✓		✓		✓		
ST_DelaunayTriang	✓		✓				✓
ST_FilterByM	✓						
ST_GeneratePoints	✓						
ST_GeometricMedi	✓		✓				
ST_LineMerge	✓						
ST_MaximumInscri	✓ Circle						
ST_LargestEmptyC	✓						
ST_MinimumBoun	✓ Circle						
ST_MinimumBoun	✓ Radius						
ST_OrientedEnvelo	✓						
ST_OffsetCurve	✓						
ST_PointOnSurface	✓		✓		✓		
ST_Polygonize	✓						
ST_ReducePrecisio	✓						
ST_SharedPaths	✓						
ST_Simplify	✓						
ST_SimplifyPreserv	✓ pology						
ST_SimplifyPolygo	✓ ll						
ST_SimplifyVW	✓						
ST_SetEffectiveAre	✓						
ST_TriangulatePoly	✓						
ST_VoronoiLines	✓						
ST_VoronoiPolygor	✓						
ST_CoverageInvalic	✓ es						
ST_CoverageSimpli	✓						
ST_CoverageUnion	✓						
ST_Affine	✓		✓	✓		✓	✓
ST_Rotate	✓		✓	✓		✓	✓
ST_RotateX	✓		✓			✓	✓
ST_RotateY	✓		✓			✓	✓
ST_RotateZ	✓		✓	✓		✓	✓
ST_Scale	✓		✓	✓		✓	✓
ST_Translate	✓		✓	✓			

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_TransScale	✓		✓	✓			
ST_ClusterDBSCA	✓			✓			
ST_ClusterIntersect	✓						
ST_ClusterIntersect	✓ Vin						
ST_ClusterKMeans	✓						
ST_ClusterWithin	✓			✓			
ST_ClusterWithinW	✓			✓			
Box2D	✓			✓		✓	✓
Box3D	✓		✓	✓		✓	✓
ST_EstimatedExten	☑			✓			
ST_Expand	✓					✓	✓
ST_Extent	✓					✓	✓
ST_3DExtent	✓		✓	✓		✓	✓
ST_MakeBox2D	✓						
ST_3DMakeBox	✓						
ST_XMax	☑		✓	✓			
ST_XMin	☑		✓	✓			
ST_YMax	☑		✓	✓			
ST_YMin	☑		✓	✓			
ST_ZMax	☑		✓	✓			
ST_ZMin	☑		✓	✓			
ST_LineInterpolatel	✓ t	✓	✓				
ST_3DLineInterpol	✓ oint		✓				
ST_LineInterpolatel	✓ ts	✓	✓				
ST_LineLocatePoin	✓	✓					
ST_LineSubstring	✓	✓	✓				
ST_LocateAlong	✓				✓		
ST_LocateBetween	✓				✓		
ST_LocateBetweenI	✓ ations		✓				
ST_InterpolatePoint	✓		✓				
ST_AddMeasure	✓		✓				
ST_IsValidTrajecto	✓		✓				
ST_ClosestPointOf	✓ oach		✓				
ST_DistanceCPA	✓		✓				

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_CPAWithin	✓		✓				
postgis.gdal_datapath							
postgis.gdal_enabled_drivers							
postgis.enable_outdb_rasters							
postgis.gdal_vsi_options							
PostGIS_AddBBox	✓			✓			
PostGIS_DropBBox	✓			✓			
PostGIS_HasBBox	✓			✓			
postgis_sfcgal_version							
postgis_sfcgal_full_version							
CG_ForceLHR							
CG_IsPlanar							
CG_IsSolid							
CG_MakeSolid							
CG_Orientation							
CG_Area							
CG_3DArea							
CG_Volume							
ST_ForceLHR							
ST_IsPlanar							
ST_IsSolid							
ST_MakeSolid							
ST_Orientation							
ST_3DArea							
ST_Volume							
CG_Intersection							
CG_Intersects							
CG_3DIntersects							

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
CG_Difference							
ST_3DDifference							
CG_3DDifference							
CG_Distance							
CG_3DDistance							
ST_3DConvexHull							
CG_3DConvexHull							
ST_3DIntersection							
CG_3DIntersection							
CG_Union							
ST_3DUnion							
CG_3DUnion							
ST_AlphaShape	✓						
CG_AlphaShape							
CG_ApproxConvex	tion						
ST_ApproximateM	Axis						
CG_ApproximateM	lAxis						
ST_ConstrainedDe	yTriangles						
CG_ConstrainedDe	ayTriangles						
ST_Extrude							
CG_Extrude							
CG_ExtrudeStraigh	leton						
CG_GreeneApprox	exPartition						
ST_MinkowskiSum							

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
CG_MinkowskiSur							
ST_OptimalAlphaS							
CG_OptimalAlpha							
CG_OptimalConve ition							
CG_StraightSkelet							
ST_StraightSkeleto							
ST_Tesselate							
CG_Tesselate							
CG_Triangulate							
CG_Visibility							
CG_YMonotonePa n							
getfaceedges_returntype							
TopoGeometry							
validatetopology_returntype							
TopoElement							
TopoElementArray							
AddTopoGeometryColumn							
RenameTopoGeometryColumn							
DropTopology							
RenameTopology							
DropTopoGeometryColumn							
Populate_Topology_Layer							
TopologySummary							
ValidateTopology ✓							
ValidateTopologyRelation							
FindTopology							
FindLayer							
CreateTopology							
CopyTopology							
ST_InitTopoGeo					✓		
ST_CreateTopoGeo ✓					✓		
TopoGeo_AddPoint ✓							
TopoGeo_AddLine ✓ g							
TopoGeo_AddPoly ✓							
TopoGeo_LoadGeo ✓ y							
ST_AddIsoNode ✓					✓		

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_AddIsoEdge	✓				✓		
ST_AddEdgeNewFa	✓				✓		
ST_AddEdgeModFa	✓				✓		
ST_RemEdgeNewFace					✓		
ST_RemEdgeModFace					✓		
ST_ChangeEdgeGe	✓				✓		
ST_ModEdgeSplit	✓				✓		
ST_ModEdgeHeal					✓		
ST_NewEdgeHeal					✓		
ST_MoveIsoNode	✓				✓		
ST_NewEdgesSplit	✓				✓		
ST_RemoveIsoNode					✓		
ST_RemoveIsoEdge					✓		
GetEdgeByPoint	✓						
GetFaceByPoint	✓						
GetFaceContainingI	✓ t						
GetNodeByPoint	✓						
GetTopologyID							
GetTopologySRID							
GetTopologyName							
ST_GetFaceEdges					✓		
ST_GetFaceGeomet	✓				✓		
GetRingEdges							
GetNodeEdges							
Polygonize							
AddNode	✓						
AddEdge	✓						
AddFace	✓						
ST_Simplify	✓						
RemoveUnusedPrin	✓ es						
CreateTopoGeom	✓						
toTopoGeom	✓						
TopoElementArray_Agg							
TopoElement	✓						
clearTopoGeom	✓						
TopoGeom_addEle	✓						
TopoGeom_remEle	✓ t						

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
TopoGeom_addTopoGeom	✓						
toTopoGeom							
GetTopoGeomElementArray							
GetTopoGeomElements							
ST_SRID	✓				✓		
AsGML	✓						
AsTopoJSON	✓						
Equals	✓		✓				
Intersects	✓		✓				
geomval							
addbandarg							
rastbandarg							
raster							
reclassarg							
summarystats							
unionarg							
AddRasterConstraints							
DropRasterConstraints							
AddOverviewConstraints							
DropOverviewConstraints							
PostGIS_GDAL_Version							
PostGIS_Raster_Lib_Build_Date							
PostGIS_Raster_Lib_Version							
ST_GDALDrivers							
ST_Contour							
ST_InterpolateRaster	✓						
UpdateRasterSRID							
ST_CreateOverview							
ST_AddBand							
ST_AsRaster	✓						
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							

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_SkewX							
ST_SkewY							
ST_SRID							
ST_Summary							
ST_UpperLeftX							
ST_UpperLeftY							
ST_Width							
ST_WorldToRasterCoord	✓						
ST_WorldToRasterCoordX	✓						
ST_WorldToRasterCoordY	✓						
ST_BandMetaData							
ST_BandNoDataValue							
ST_BandIsNoData							
ST_BandPath							
ST_BandFileSize							
ST_BandFileTimestamp							
ST_BandPixelType							
ST_MinPossibleValue							
ST_HasNoBand							
ST_PixelAsPolygon	✓						
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							

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
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	✓						
ST_MapAlgebra (callback function version)							
ST_MapAlgebra (expression version)							
ST_MapAlgebraExpr							
ST_MapAlgebraExpr							
ST_MapAlgebraFct							
ST_MapAlgebraFct							
ST_MapAlgebraFctNgb							
ST_Reclass							
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							
Box3D							
ST_ConvexHull	✓						
ST_DumpAsPolygons							

Fonction	geom	geog	2.5D	Courbes	SQL MM	PS	T
ST_Envelope	✓						
ST_MinConvexHull	✓						
ST_Polygon	✓						
&&	✓						
&<							
&>							
=							
@	✓						
~=							
~	✓						
ST_Contains							
ST_ContainsProperly							
ST_Covers							
ST_CoveredBy							
ST_Disjoint							
ST_Intersects	✓						
ST_Overlaps							
ST_Touches							
ST_SameAlignment							
ST_NotSameAlignmentReason							
ST_Within							
ST_DWithin							
ST_DFullyWithin							
stdaddr							
rules table							
lex table							
gaz table							
debug_standardize_address							
parse_address							
standardize_address							
Drop_Indexes_Generate_Script							
Drop_Nation_Tables_Generate_Script							
Drop_State_Tables_Generate_Script							
Geocode	✓						
Geocode_Intersecti	✓						
Get_Geocode_Setting							
Get_Tract	✓						
Install_Missing_Indexes							
Loader_Generate_Census_Script							
Loader_Generate_Script							
Loader_Generate_Nation_Script							
Missing_Indexes_Generate_Script							
Normalize_Address							
Pagc_Normalize_Address							
Pprint_Addy							
Reverse_Geocode	✓						
Topology_Load_Tiger							
Set_Geocode_Setting							

13.12 Fonctions PostGIS nouvelles, améliorées ou modifiées

13.12.1 Fonctions PostGIS nouvelles ou améliorées en 3.5

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 3.5

- CG_3DArea - Disponibilité : 3.5.0 Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- CG_3DConvexHull - Disponibilité : 3.5.0 Calcule l'enveloppe convexe 3D d'une géométrie.
- CG_3DDifference - Disponibilité : 3.5.0 Effectuer une différence 3D
- CG_3DDistance - Disponibilité : 3.5.0 Calcule la distance 3D minimale entre deux géométries
- CG_3DIntersection - Disponibilité : 3.5.0 Réaliser une intersection 3D
- CG_3DIntersects - Disponibilité : 3.5.0 Teste si deux géométries 3D intersectent
- CG_3DUnion - Disponibilité : 3.5.0 Perform 3D union using postgis_sfcgal.
- CG_AlphaShape - Availability: 3.5.0 - requires SFCGAL >= 1.4.1. Calcul d'une forme Alpha entourant une géométrie
- CG_ApproxConvexPartition - Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0. Calcul de la partition convexe approximative de la géométrie du polygone
- CG_ApproximateMedialAxis - Disponibilité : 3.5.0 Calculer l'axe médian approximatif d'une géométrie aréolaire.
- CG_Area - Disponibilité : 3.5.0 Calcule la surface d'une géométrie
- CG_Difference - Disponibilité : 3.5.0 Calcul la différence géométrique entre deux géométries
- CG_Distance - Disponibilité : 3.5.0 Calcule la distance minimale entre deux géométries
- CG_Extrude - Disponibilité : 3.5.0 Extruder une surface vers un volume
- CG_ExtrudeStraightSkeleton - Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0. Extrusion de squelette droit
- CG_ForceLHR - Disponibilité : 3.5.0 Force l'orientation LHR d'un objet
- CG_GreeneApproxConvexPartition - Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0. Calcul de la partition convexe approximative de la géométrie du polygone
- CG_Intersection - Disponibilité : 3.5.0 Calcul l'intersection de deux géométries
- CG_Intersects - Disponibilité : 3.5.0 Teste si deux géométries se croisent (elles ont au moins un point en commun)
- CG_IsPlanar - Disponibilité : 3.5.0 Vérifie si une surface est planaire ou non
- CG_IsSolid - Disponibilité : 3.5.0 Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.
- CG_MakeSolid - Disponibilité : 3.5.0 Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.
- CG_MinkowskiSum - Disponibilité : 3.5.0 Effectue la somme de Minkowski
- CG_OptimalAlphaShape - Availability: 3.5.0 - requires SFCGAL >= 1.4.1. Calcule une forme alpha entourant une géométrie en utilisant une valeur alpha "optimale".
- CG_OptimalConvexPartition - Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0. Calcul d'une partition convexe optimale de la géométrie du polygone
- CG_Orientation - Disponibilité : 3.5.0 Détermine l'orientation d'une surface
- CG_StraightSkeleton - Disponibilité : 3.5.0 Calcule un squelette (straight skeleton) à partir d'une géométrie

- CG_Tesselate - Disponibilité : 3.5.0 Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS
- CG_Triangulate - Disponibilité : 3.5.0 Triangule une géométrie polygonale
- CG_Union - Disponibilité : 3.5.0 Calcule l'union de deux géométries
- CG_Visibility - Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0. Calculer un polygone de visibilité à partir d'un point ou d'un segment dans une géométrie polygonale
- CG_Volume - Disponibilité : 3.5.0 Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- CG_YMonotonePartition - Disponibilité : 3.5.0 - nécessite SFCGAL >= 1.5.0. Calcul de la partition y-monotone de la géométrie du polygone
- ST_HasM - Disponibilité : 3.5.0 Vérifie si une géométrie a une dimension M (mesure).
- ST_HasZ - Disponibilité : 3.5.0 Vérifie si une géométrie possède une dimension Z.
- ST_RemoveIrrelevantPointsForView - Disponibilité : 3.5.0 Removes points that are irrelevant for rendering a specific rectangluar view of a geometry.
- ST_RemoveSmallParts - Disponibilité : 3.5.0 Removes small parts (polygon rings or linestrings) of a geometry.
- TopoGeo_LoadGeometry - Disponibilité : 3.5.0 Charge une géométrie dans une topologie existante, en effectuant de l'accrochage et des divisions si nécessaire.

Fonctions améliorées dans PostGIS 3.5

- ST_Clip - Amélioration : 3.5.0 - ajout de l'argument touched. Returns the raster clipped by the input geometry. If band number is not specified, all bands are processed. If crop is not specified or TRUE, the output raster is cropped. If touched is set to TRUE, then touched pixels are included, otherwise only if the center of the pixel is in the geometry it is included.

Fonctions modifiées dans PostGIS 3.5

- ST_AsGeoJSON - Modifié : la version 3.5.0 permet de spécifier la colonne contenant l'identifiant de l'élément Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_DFullyWithin - Modifié : 3.5.0 : la logique implémentée utilise désormais un test de confinement dans un tampon, plutôt que l'algorithme ST_MaxDistance. Les résultats seront différents de ceux des versions précédentes, mais devraient être plus proches des attentes de l'utilisateur. Teste si une géométrie se trouve entièrement à une distance d'une autre géométrie

13.12.2 Fonctions PostGIS nouvelles ou améliorées en 3.4

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 3.4

- PostGIS_GEOS_Compiled_Version - Disponibilité : 3.4.0 Renvoie le numéro de version de la librairie GEOS avec laquelle PostGIS a été construit.
- PostGIS_PROJ_Compiled_Version - Disponibilité : 3.5.0 Returns the version number of the PROJ library against which PostGIS was built.
- RenameTopoGeometryColumn - Disponibilité : 3.4.0 Renomme une colonne topogeometry
- RenameTopology - Disponibilité : 3.4.0 Renomme une topologie
- ST_ClusterIntersectingWin - Disponibilité : 3.4.0 Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, en regroupant les géométries en entrée en ensembles connectés.

- ST_ClusterWithinWin - Disponibilité : 3.4.0 Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée, regroupement en utilisant la distance de séparation.
- ST_CoverageInvalidEdges - Disponibilité : 3.4.0 Fonction window qui trouve les endroits où les polygones ne forment pas une couverture valide.
- ST_CoverageSimplify - Disponibilité : 3.4.0 Fonction window qui simplifie les bords d'une couverture polygonale.
- ST_CoverageUnion - Disponibilité : 3.4.0 - nécessite GEOS >= 3.8.0 Calcule l'union d'un ensemble de polygones formant une couverture en supprimant les arêtes communes.
- ST_InverseTransformPipeline - Disponibilité : 3.4.0 Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent en utilisant l'inverse d'un pipeline de transformation de coordonnées défini.
- ST_LargestEmptyCircle - Disponibilité : 3.4.0. Calcule le plus grand cercle ne recouvrant pas une géométrie.
- ST_LineExtend - Disponibilité : 3.4.0 Renvoie une ligne prolongée vers l'avant et vers l'arrière selon les distances spécifiées.
- ST_TransformPipeline - Disponibilité : 3.4.0 Retourner une nouvelle géométrie avec des coordonnées transformées dans un système de référence spatial différent à l'aide d'un pipeline de transformation de coordonnées défini.
- TopoElement - Disponibilité : 3.4.0 Convertit une topogeometry en un topoelement.
- debug_standardize_address - Disponibilité : 3.4.0 Retourne une chaîne au format json avec les jetons d'entrée et les normalisations
- postgis_srs - Disponibilité : 3.4.0 Renvoyer une fiche de métadonnées pour l'autorité et le srid demandés.
- postgis_srs_all - Disponibilité : 3.4.0 Renvoie des enregistrements de métadonnées pour chaque système de référence spatiale dans la base de données Proj sous-jacente.
- postgis_srs_codes - Disponibilité : 3.4.0 Renvoie la liste des codes SRS associés à l'autorité donnée.
- postgis_srs_search - Disponibilité : 3.4.0 Renvoyer les enregistrements de métadonnées pour les systèmes de coordonnées projetées dont les zones d'utilisation contiennent entièrement le paramètre bounds.

Fonctions améliorées dans PostGIS 3.4

- PostGIS_Full_Version - Amélioration : 3.4.0 inclut désormais les configurations supplémentaires PROJ NETWORK_ENABLED, URL_ENDPOINT et DATABASE_PATH pour l'emplacement proj.db Donne des informations complètes sur la version de PostGIS et la configuration du packaging.
- PostGIS_PROJ_Version - Amélioration : 3.4.0 inclut désormais PROJ NETWORK_ENABLED, URL_ENDPOINT et DATABASE_PA pour l'emplacement proj.db Renvoie le numéro de version de la librairie PROJ4.
- ST_AsSVG - Amélioration : 3.4.0 pour prendre en charge tous les types de courbes Renvoie les données de chemin SVG pour une géométrie.
- ST_ClosestPoint - Amélioré : 3.4.0 - Prise en charge de la geography. Renvoie le point 2D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne la plus courte d'une géométrie à l'autre.
- ST_LineSubstring - Amélioration : 3.4.0 - La prise en charge de la géographie a été introduite. Renvoie la partie d'une ligne située entre deux emplacements fractionnaires.
- ST_Project - Amélioration : 3.4.0 Autorise les arguments géométriques et la forme en deux points omettant l'azimut. Renvoie un point projeté à partir d'un point de départ en fonction d'une distance et d'un azimut.
- ST_Resample - Amélioration : 3.4.0 ajout des options de rééchantillonnage max et min Rééchantillonne un raster, en utilisant l'algorithme spécifié, les nouvelles dimensions, un coin arbitraire de la grille et un ensemble de paramètres de géoréférencement définis ou empruntés à un autre raster.
- ST_Rescale - Amélioration : 3.4.0 ajout des options de rééchantillonnage max et min Rééchantillonne un raster en ajustant juste son échelle (ou la taille des pixels). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline), Lanczos, Max ou Min. La valeur par défaut est NearestNeighbor.

- ST_ShortestLine - Amélioré : 3.4.0 - Prise en charge de la geography. Renvoie la ligne 2D la plus courte entre deux géométries

Fonctions modifiées dans PostGIS 3.4

- PostGIS_Extensions_Upgrade - Modifié : 3.4.0 pour ajouter l'argument target_version. Packages et mises à jour des extensions PostGIS (par exemple postgis_raster, postgis_topology, postgis_sfcgal) vers la version donnée ou la plus récente.

13.12.3 Fonctions PostGIS nouvelles ou améliorées en 3.3

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 3.3

- RemoveUnusedPrimitives - Disponibilité : 3.3.0 Supprime les primitives topologiques qui ne sont pas nécessaires pour définir les objets TopoGeometry existants.
- ST_3DConvexHull - Disponibilité : 3.3.0 Calcule l'enveloppe convexe 3D d'une géométrie.
- ST_3DUnion - Disponibilité : 3.3.0 la variante agrégée a été ajoutée Effectuer l'union 3D.
- ST_AsMARC21 - Disponibilité : 3.3.0 Renvoie la géométrie sous forme d'enregistrement MARC21/XML avec un champ de données géographiques (034).
- ST_GeomFromMARC21 - Disponibilité : 3.3.0, nécessite libxml2 2.6+ Prend les données géographiques MARC21/XML en entrée et renvoie un objet géométrique PostGIS.
- ST_Letters - Disponibilité : 3.3.0 Renvoie les lettres d'entrée rendues sous forme de géométrie avec une position de départ par défaut à l'origine et une hauteur de texte par défaut de 100.
- ST_OptimalAlphaShape - Disponibilité : 3.3.0 - nécessite SFCGAL >= 1.4.1. Calcule une forme alpha entourant une géométrie en utilisant une valeur alpha "optimale".
- ST_SimplifyPolygonHull - Disponibilité : 3.3.0. Calcule une enveloppe extérieure ou intérieure simplifiée préservant la topologie d'une géométrie polygonale.
- ST_TriangulatePolygon - Disponibilité : 3.3.0. Calcule la triangulation de Delaunay contrainte des polygones
- postgis_sfcgal_full_version - Disponibilité : 3.3.0 Retourne la version complète de SFCGAL en cours d'utilisation, y compris les versions CGAL et Boost

Fonctions améliorées dans PostGIS 3.3

- ST_ConcaveHull - Amélioré : 3.3.0, implémentation native de GEOS activée pour GEOS 3.11+ Calcule une géométrie éventuellement concave qui contient tous les sommets de la géométrie d'entrée
- ST_LineMerge - Amélioration : 3.3.0 accepte un paramètre direct. Renvoie les lignes formées par la couture d'une MultiLineString.

Fonctions modifiées dans PostGIS 3.3

- PostGIS_Extensions_Upgrade - Modifié : 3.3.0 support pour les mises à jour à partir de n'importe quelle version de PostGIS. Ne fonctionne pas sur tous les systèmes. Packages et mises à jour des extensions PostGIS (par exemple postgis_raster, postgis_topology, postgis_sfcgal) vers la version donnée ou la plus récente.

13.12.4 Fonctions PostGIS nouvelles ou améliorées en 3.2

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 3.2

- FindLayer - Disponibilité : 3.2.0 Renvoie un enregistrement topology.layer par différents moyens.
- FindTopology - Disponibilité : 3.2.0 Renvoie un enregistrement topologique par différents moyens.
- GetFaceContainingPoint - Disponibilité : 3.2.0 Recherche la face contenant un point.
- ST_AsFlatGeobuf - Disponibilité : 3.2.0 Renvoie une représentation FlatGeobuf d'un ensemble de lignes.
- ST_Contour - Disponibilité : 3.2.0 Génère un ensemble de courbes de niveau vectorielles depuis la bande raster spécifiée, en utilisant l'algorithme de contour GDAL.
- ST_DumpSegments - Disponibilité : 3.2.0 Renvoie un ensemble de lignes geometry_dump pour les segments d'une géométrie.
- ST_FromFlatGeobuf - Disponibilité : 3.2.0 Lit les données FlatGeobuf.
- ST_FromFlatGeobufToTable - Disponibilité : 3.2.0 Crée une table basée sur la structure des données FlatGeobuf.
- ST_InterpolateRaster - Disponibilité : 3.2.0 Interpole une surface quadrillée à partir d'un ensemble de points 3-d, en utilisant les coordonnées X et Y des points sur la grille et la coordonnée Z des points pour l'élévation des points.
- ST_SRID - Disponibilité : 3.2.0 Renvoie l'identifiant de référence spatiale d'une topogeometry.
- ST_Scroll - Disponibilité : 3.2.0 Modifier le point de départ d'une LineString fermée.
- ST_SetM - Disponibilité : 3.2.0 Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée M copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.
- ST_SetZ - Disponibilité : 3.2.0 Retourne une géométrie avec les mêmes coordonnées X/Y que la géométrie d'entrée, et avec la coordonnée Z copiée depuis les valeurs du raster selon l'algorithme d'interpolation en paramètre.
- TopoGeom_addTopoGeom - Disponibilité : 3.2 Ajoute un élément d'une TopoGeometry à la définition d'une autre TopoGeometry.
- ValidateTopologyRelation - Disponibilité : 3.2.0 Renvoie des informations sur les enregistrements de relations topologiques non valides
- postgis.gdal_vsi_options - Disponibilité : 3.2.0 Une chaîne de configuration pour définir les options utilisées lors de l'utilisation d'un raster out-db.

Fonctions améliorées dans PostGIS 3.2

- GetFaceByPoint - Amélioration : 3.2.0 mise en œuvre plus efficace et contrat plus clair, arrêt du fonctionnement avec des topologies non valides. Recherche la face intersectant un point donné.
- ST_ClusterKMeans - Amélioré : 3.2.0 Support pour max_radius Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée en utilisant l'algorithme K-means.
- ST_MakeValid - Amélioration : 3.2.0, ajout d'options d'algorithme, 'linework' et 'structure' qui nécessite GEOS >= 3.10.0. Tente de rendre valide une géométrie invalide sans perdre de sommets.
- ST_PixelAsCentroid - Amélioration : 3.2.0 Plus rapide, désormais implémenté en C. Retourne le centroïde (point géométrique) de la zone représentée par un pixel.
- ST_PixelAsCentroids - Amélioration : 3.2.0 Plus rapide, désormais implémenté en C. Retourne le centroïde (point géométrique) pour chaque pixel de la bande raster, avec sa valeur et les coordonnées raster X et Y. Le point géométrique est le centroïde de la zone représentée par un pixel.

- ST_Point - Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie. Crée un point avec des valeurs X, Y et SRID.
- ST_PointM - Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie. Crée un point avec des valeurs X, Y, M et SRID.
- ST_PointZ - Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie. Crée un point avec des valeurs X, Y, Z et SRID.
- ST_PointZM - Amélioration : 3.2.0 srid a été ajouté comme argument optionnel supplémentaire. Les anciennes installations nécessitent une combinaison avec ST_SetSRID pour marquer le srid sur la géométrie. Crée un point avec des valeurs X, Y, Z, M et SRID.
- ST_RemovePoint - Amélioration : 3.2.0 Supprime un point d'une ligne.
- ST_RemoveRepeatedPoints - Amélioration : 3.2.0 Renvoie une version d'une géométrie dont les points en double ont été supprimés.
- ST_StartPoint - Amélioré : 3.2.0 renvoie un point pour toutes les géométries. Le comportement précédent renvoyait NULL si l'entrée n'était pas une LineString. Renvoie le premier point d'une LineString.
- ST_Value - Amélioration : 3.2.0 ajout du paramètre optionnel resample. Retourne la valeur d'une bande raster spécifiée au pixel donné par columnx, rowy, ou à un point géométrique spécifié. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié. Si exclude_nodata_value vaut false, tous les pixels y compris ceux ayant la valeur nodata sont considérés comme intersectés et leur valeur sera retournée. Si exclude_nodata_value n'est pas spécifié, la valeur est lue depuis les méta-données du raster.
- TopoGeo_AddLineString - Enhanced: 3.2.0 added support for returning signed identifier. Adds a linestring to an existing topology using a tolerance and possibly splitting existing edges/faces.

Fonctions modifiées dans PostGIS 3.2

- ST_Boundary - Modifié : 3.2.0 support pour TIN, n'utilise pas geos, ne linéarise pas les courbes Renvoie la limite d'une géométrie.
- ValidateTopology - Modifié : 3.2.0 a ajouté le paramètre optionnel bbox, effectuant des vérifications de l'étiquetage des faces et de la liaison des arêtes. Renvoie un ensemble d'objets validatetopology_returntype détaillant les problèmes liés à la topologie.

13.12.5 Fonctions PostGIS nouvelles ou améliorées en 3.1

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 3.1

- ST_Hexagon - Disponibilité : 3.1.0 Renvoie un seul hexagone, en utilisant la taille du bord et les coordonnées de la cellule fournies dans l'espace de la grille de l'hexagone.
- ST_HexagonGrid - Disponibilité : 3.1.0 Renvoie un ensemble d'hexagones et d'indices de cellules qui couvrent complètement les limites de l'argument géométrie.
- ST_MaximumInscribedCircle - Disponibilité : 3.1.0. Calcule le plus grand cercle contenu dans une géométrie.
- ST_ReducePrecision - Disponibilité : 3.1.0. Renvoie une géométrie valide dont les points sont arrondis en fonction de la tolérance de la grille.
- ST_Square - Disponibilité : 3.1.0 Renvoie un seul carré, en utilisant la taille du bord et la coordonnée de la cellule fournies dans l'espace de la grille du carré.

- ST_SquareGrid - Disponibilité : 3.1.0 Renvoie un ensemble de carrés de grille et d'indices de cellules qui couvrent complètement les limites de l'argument géométrie.

Fonctions améliorées dans PostGIS 3.1

- ST_AsEWKT - Amélioré : support du paramètre optionnel de précision dans la version 3.1.0. Renvoie la représentation Well-Known Text (WKT) de la géométrie avec les métadonnées SRID.
- ST_ClusterKMeans - Amélioration : 3.1.0 Prise en charge des géométries et des poids en 3D Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée en utilisant l'algorithme K-means.
- ST_Difference - Amélioration : 3.1.0 accepte un paramètre gridSize. Calcule une géométrie représentant la partie de la géométrie A qui n'intersecte pas la géométrie B.
- ST_Intersection - Amélioration : 3.1.0 accepte un paramètre gridSize Calcule une géométrie représentant la partie partagée des géométries A et B.
- ST_MakeValid - Amélioration : 3.1.0, suppression des coordonnées avec des valeurs NaN. Tente de rendre valide une géométrie invalide sans perdre de sommets.
- ST_Subdivide - Amélioration : 3.1.0 accepte un paramètre gridSize. Calcule une subdivision rectiligne d'une géométrie.
- ST_SymDifference - Amélioration : 3.1.0 accepte un paramètre gridSize. Calcule une géométrie représentant les parties des géométries A et B qui ne s'intersectent pas.
- ST_TileEnvelope - Amélioré : 3.1.0 Ajout d'un paramètre de marge. Crée un polygone rectangulaire dans Web Mercator (SRID:3857) en utilisant le système de tuiles XYZ.
- ST_UnaryUnion - Amélioration : 3.1.0 accepte un paramètre gridSize. Calcule l'union des composantes d'une seule géométrie.
- ST_Union - Amélioration : 3.1.0 accepte un paramètre gridSize. Calcule une géométrie représentant l'union des ensembles de points des géométries d'entrée.

Fonctions modifiées dans PostGIS 3.1

- ST_Count - Changement : 3.1.0 - Suppression des variantes ST_Count(rastertable, rastercolumn, ...). Utiliser à la place. Renvoie le nombre de pixels dans une bande donnée d'un raster ou d'une couverture raster. Si aucune bande n'est spécifiée, la valeur par défaut est la bande 1. Si exclude_nodata_value est true, seuls les pixels dont la valeur est différente de la valeur nodata seront comptés.
- ST_Force3D - Modifié : 3.1.0. Ajout de la prise en charge pour pouvoir passer une valeur Z non nulle. Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ.
- ST_Force3DM - Modifié : 3.1.0. Ajout de la prise en charge de pouvoir passer une valeur M non nulle. Forcer les géométries en mode XYM.
- ST_Force3DZ - Modifié : 3.1.0. Ajout de la prise en charge pour pouvoir passer une valeur Z non nulle. Forcer les géométries en mode XYZ.
- ST_Force4D - Modifié : 3.1.0. Ajout de la prise en charge de pouvoir passer des valeurs Z et M non nulles. Forcer les géométries en mode XYZM.
- ST_Histogram - Changement : 3.1.0 Suppression de la variante ST_Histogram(table_name, column_name). Retourne un ensemble d'enregistrements résumant une distribution de données raster ou de couverture raster, dans des classes distinctes. Le nombre de classes est calculé automatiquement s'il n'est pas spécifié.
- ST_Quantile - Changement : 3.1.0 Suppression de la variante ST_Quantile(table_name, column_name). Calcule les quantiles d'un raster ou d'une couverture raster, dans le contexte de l'échantillon ou de la population. Ainsi, une valeur peut être examinée pour se situer au percentile de 25%, 50% ou 75% du raster.
- ST_SummaryStats - Changement : 3.1.0 Suppression des variantes ST_SummaryStats(rastertable, rastercolumn, ...). Utiliser à la place. Retourne des résumés statistiques (count, sum, mean, stddev, min, max) pour une bande raster ou une couverture raster spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

13.12.6 Fonctions PostGIS nouvelles ou améliorées en 3.0

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 3.0

- CG_ConstrainedDelaunayTriangles - Disponibilité : 3.0.0 Renvoie une triangulation de Delaunay contrainte autour de la géométrie d'entrée donnée.
- ST_3DLineInterpolatePoint - Disponibilité : 3.0.0 Renvoie un point interpolé le long d'une ligne 3D à un emplacement fractionnaire.
- ST_ConstrainedDelaunayTriangles - Disponibilité : 3.0.0 Renvoie une triangulation de Delaunay contrainte autour de la géométrie d'entrée donnée.
- ST_TileEnvelope - Disponibilité : 3.0.0 Crée un polygone rectangulaire dans Web Mercator (SRID:3857) en utilisant le système de tuiles XYZ.

Fonctions améliorées dans PostGIS 3.0

- ST_AsMVT - Amélioration : 3.0 - ajout de la prise en charge du Feature ID. Fonction d'agrégation renvoyant une représentation MVT d'un ensemble de lignes.
- ST_Contains - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Tests si chaque point de B est situé dans A, et que leurs intérieurs ont un point commun
- ST_ContainsProperly - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Tests si chaque point de B se trouve à l'intérieur de A
- ST_CoveredBy - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Tests si chaque point de A se trouve dans B
- ST_Covers - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Tests si chaque point de B est situé dans A
- ST_Crosses - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Teste si deux géométries ont en commun certains points intérieurs, mais pas tous
- ST_CurveToLine - Amélioration : la version 3.0.0 a mis en place un nombre minimum de segments par arc linéarisé afin d'éviter une rupture topologique. Convertit une géométrie contenant des courbes en une géométrie linéaire.
- ST_Disjoint - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Teste si deux géométries n'ont pas de points communs
- ST_Equals - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Teste si deux géométries comprennent le même ensemble de points
- ST_GeneratePoints - Amélioration : 3.0.0, ajout du paramètre seed Génère un multipoint de points aléatoires contenus dans un polygone ou un multipolygone.
- ST_GeomFromGeoJSON - Amélioré : 3.0.0 La géométrie parsée prend par défaut la valeur SRID=4326 si elle n'est pas spécifiée autrement. Prend en entrée une géométrie au format geojson et renvoie un objet Postgis de type geometry
- ST_LocateBetween - Amélioration : 3.0.0 - ajout de la prise en charge du POLYGONE, du TIN et du TRIANGLE. Renvoie les parties d'une géométrie qui correspondent à un intervalle de mesure.
- ST_LocateBetweenElevations - Amélioration : 3.0.0 - ajout de la prise en charge du POLYGONE, du TIN et du TRIANGLE. Renvoie les parties d'une géométrie qui se trouvent dans un intervalle d'élévation (Z).
- ST_Overlaps - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Teste si deux géométries ont la même dimension et se croisent, mais si chacune a au moins un point qui n'est pas dans l'autre

- ST_Relate - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Teste si deux géométries ont une relation topologique correspondant à un modèle de matrice d'intersection, ou calcule leur matrice d'intersection
- ST_Segmentize - Amélioration : 3.0.0 La segmentation de géométrie produit désormais des sous-segments de longueur égale Renvoie une geometry/geography modifiée dont aucun segment ne dépasse une distance donnée.
- ST_Touches - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Teste si deux géométries ont au moins un point en commun, mais que leurs intérieurs ne se croisent pas
- ST_Within - Amélioration : 3.0.0 a permis la prise en charge de GEOMETRYCOLLECTION Tests si chaque point de A se trouve dans B, et que leurs intérieurs ont un point commun

Fonctions modifiées dans PostGIS 3.0

- PostGIS_Extensions_Upgrade - Modifié : 3.0.0 pour repackager les extensions libres et supporter postgis_raster. Packages et mises à jour des extensions PostGIS (par exemple postgis_raster, postgis_topology, postgis_sfcgal) vers la version donnée ou la plus récente.
- ST_3DDistance - Modifié : 3.0.0 - Version SFCGAL supprimée Renvoie la distance cartésienne minimale en 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DIntersects - Modifié : 3.0.0 SFCGAL backend supprimé, GEOS backend supporte les TINs. Teste si deux géométries se croisent dans l'espace en 3D - uniquement pour les points, les lignes, les polygones, les surfaces polyédriques (aire)
- ST_Area - Modifié : 3.0.0 - ne dépend plus de SFCGAL. Renvoie l'aire d'une géométrie polygonale.
- ST_AsGeoJSON - Modifié : la version 3.0.0 prend en charge les enregistrements en tant que données d'entrée Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_AsGeoJSON - Modifié : 3.0.0 SRID de sortie si ce n'est pas EPSG:4326. Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_AsKML - Modifié : 3.0.0 - Suppression de la signature de la variante "versioned" Renvoyer la géométrie sous forme d'élément KML.
- ST_Distance - Modifié : 3.0.0 - ne dépend plus de SFCGAL. Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_Intersection - Modifié : 3.0.0 ne dépend pas de SFCGAL. Calcule une géométrie représentant la partie partagée des géométries A et B.
- ST_Intersects - Modifié : 3.0.0 La version SFCGAL a été supprimée et la prise en charge native des TINS 2D a été ajoutée. Teste si deux géométries se croisent (elles ont au moins un point en commun)
- ST_Union - Modifié : 3.0.0 ne dépend pas de SFCGAL. Calcule une géométrie représentant l'union des ensembles de points des géométries d'entrée.

13.12.7 Fonctions PostGIS nouvelles ou améliorées en 2.5

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 2.5

- PostGIS_Extensions_Upgrade - Disponibilité : 2.5.0 Packages et mises à jour des extensions PostGIS (par exemple postgis_raster, postgis_topology, postgis_sfcgal) vers la version donnée ou la plus récente.
- ST_Angle - Disponibilité : 2.5.0 Renvoie l'angle entre deux vecteurs définis par 3 ou 4 points, ou 2 lignes.
- ST_AsHexWKB - Disponibilité : 2.5.0 Retourne la représentation hexadécimale Well-Known Binary (WKB) de ce raster.
- ST_BandFileSize - Disponibilité : 2.5.0 Retourne la taille du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.

- ST_BandFileTimestamp - Disponibilité : 2.5.0 Retourne le timestamp du fichier d'une bande stockée sur le système de fichier. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_ChaikinSmoothing - Disponibilité : 2.5.0 Renvoie une version lissée d'une géométrie, en utilisant l'algorithme Chaikin
- ST_FilterByM - Disponibilité : 2.5.0 Supprime les vertices en fonction de leur valeur M
- ST_Grayscale - Disponibilité : 2.5.0 Crée un nouveau raster à 1 bande 8BUI à partir du raster source et des bandes spécifiées représentant les composantes rouge, vert et bleu
- ST_LineInterpolatePoints - Disponibilité : 2.5.0 Renvoie des points interpolés le long d'une ligne à un intervalle fractionnaire.
- ST_OrientedEnvelope - Disponibilité : 2.5.0. Renvoie un rectangle de surface minimale contenant une géométrie.
- ST_QuantizeCoordinates - Disponibilité : 2.5.0 Met à zéro les bits de poids faible des coordonnées
- ST_RastFromHexWKB - Disponibilité : 2.5.0 Retourne un raster à partir d'un raster Well-Known Binary (WKB) en hexadécimal.
- ST_RastFromWKB - Disponibilité : 2.5.0 Retourne un raster à partir d'un raster Well-Known Binary (WKB).
- ST_SetBandIndex - Disponibilité : 2.5.0 Met à jour le numéro de bande externe d'une bande out-db
- ST_SetBandPath - Disponibilité : 2.5.0 Met à jour le chemin externe et le numéro de bande d'une bande out-db

Fonctions améliorées dans PostGIS 2.5

- ST_AsBinary/ST_AsWKB - Amélioration : 2.5.0 Ajout de ST_AsWKB Retourne la représentation Well-Known Binary (WKB) de ce raster.
- ST_AsMVT - Amélioration : 2.5.0 - ajout de la prise en charge des requêtes parallèles. Fonction d'agrégation renvoyant une représentation MVT d'un ensemble de lignes.
- ST_AsText - Amélioration : 2.5 - introduction de la précision des paramètres optionnels. Renvoie la représentation Well-Known Text (WKT) de la géométrie/geography sans métadonnées SRID.
- ST_BandMetaData - Amélioration : 2.5.0 ajout de outdbbandnum, filesize et filetimestamp pour les rasters outdb. Retourne les méta-données de base d'une bande raster spécifique. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_Buffer - Amélioration : 2.5.0 - La prise en charge de la géométrie ST_Buffer a été améliorée pour permettre la spécification de la mise en mémoire tampon latérale side=both|left|right. Calcule une géométrie couvrant tous les points situés à une distance donnée d'une géométrie.
- ST_GeomFromGeoJSON - Amélioration : 2.5.0 peut maintenant accepter json et jsonb comme entrées. Prend en entrée une géométrie au format geojson et renvoie un objet Postgis de type geometry
- ST_GeometricMedian - Amélioré : 2.5.0 Ajout de la prise en charge de M comme poids des points. Renvoie la médiane géométrique d'un MultiPoint.
- ST_Intersects - Amélioré : 2.5.0 Supporte GEOMETRYCOLLECTION. Teste si deux géométries se croisent (elles ont au moins un point en commun)
- ST_OffsetCurve - Amélioration : 2.5 - ajout de la prise en charge de GEOMETRYCOLLECTION et MULTILINESTRING Renvoie une ligne décalée par rapport à une distance et un côté donnés à partir d'une ligne en entrée.
- ST_Scale - Amélioration : la prise en charge de la mise à l'échelle par rapport à une origine locale (paramètre origin) a été introduite dans la version 2.5.0. Met à l'échelle une géométrie en fonction de facteurs donnés.
- ST_Split - Amélioration : la prise en charge de la division d'un polygone par une ligne multiple a été introduite dans la version 2.5.0. Renvoie une collection de géométries créées en divisant une géométrie par une autre géométrie.
- ST_Subdivide - Amélioration : 2.5.0 réutilise les points existants lors de la division d'un polygone, le nombre de vertex est réduit de 8 à 5. Calcule une subdivision rectiligne d'une géométrie.

Fonctions modifiées dans PostGIS 2.5

- ST_GDALDrivers - Changement : 2.5.0 - ajout des colonnes can_read et can_write. Retourne la liste des formats raster supportés par PostGIS via GDAL. Seuls les formats avec can_write=True peuvent être utilisés par ST_AsGDALRaster

13.12.8 Fonctions PostGIS nouvelles ou améliorées en 2.4

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 2.4

- ST_AsGeobuf - Disponibilité : 2.4.0 Retourne une représentation Geobuf d'un ensemble de lignes.
- ST_AsMVT - Disponibilité : 2.4.0 Fonction d'agrégation renvoyant une représentation MVT d'un ensemble de lignes.
- ST_AsMVTGeom - Disponibilité : 2.4.0 Transforme une géométrie dans l'espace de coordonnées d'une tuile MVT.
- ST_Centroid - Disponibilité : la prise en charge du type geography a été introduite dans la version 2.4.0. Renvoie le centre géométrique d'une géométrie.
- ST_ForcePolygonCCW - Disponibilité : 2.4.0 Oriente tous les anneaux extérieurs dans le sens inverse des aiguilles d'une montre et tous les anneaux intérieurs dans le sens des aiguilles d'une montre.
- ST_ForcePolygonCW - Disponibilité : 2.4.0 Oriente tous les anneaux extérieurs dans le sens des aiguilles d'une montre et tous les anneaux intérieurs dans le sens inverse des aiguilles d'une montre.
- ST_FrechetDistance - Disponibilité : 2.4.0 - nécessite GEOS >= 3.7.0 Renvoie la distance de Fréchet entre deux géométries.
- ST_IsPolygonCCW - Disponibilité : 2.4.0 Teste si les polygones ont des anneaux extérieurs orientés dans le sens inverse des aiguilles d'une montre et des anneaux intérieurs orientés dans le sens des aiguilles d'une montre.
- ST_IsPolygonCW - Disponibilité : 2.4.0 Teste si les polygones ont des anneaux extérieurs orientés dans le sens des aiguilles d'une montre et des anneaux intérieurs orientés dans le sens inverse des aiguilles d'une montre.
- ST_MakeEmptyCoverage - Disponibilité : 2.4.0 Couvre une zone géo-référencée avec une grille raster de tuiles vides.

Fonctions améliorées dans PostGIS 2.4

- Loader_Generate_Nation_Script - Amélioré : 2.4.1 L'étape de chargement de la zone de tabulation du code postal 5 (zcta5) a été corrigée et, lorsqu'elle est activée, les données zcta5 sont chargées sous la forme d'une table unique appelée zcta5_all dans le cadre du chargement du script de la nation. Génère un script shell pour la plate-forme spécifiée qui charge les données dans les tables county et state.
- Normalize_Address - Amélioré : 2.4.0 L'objet norm_addy comprend des champs supplémentaires zip4 et address_alphanumeric. Étant donné une adresse textuelle, cette fonction renvoie un type composite norm_addy qui contient le suffixe de la route, le préfixe et le type normalisé, la rue, le nom de la rue, etc. divisés en champs distincts. Cette fonction fonctionne uniquement avec les données de recherche fournies avec le géocodeur tiger (pas besoin pour les données de recensement tiger).
- Pagc_Normalize_Address - Amélioré : 2.4.0 L'objet norm_addy comprend des champs supplémentaires zip4 et address_alphanumeric. Étant donné une adresse textuelle, cette fonction renvoie un type composite norm_addy qui contient le suffixe de la route, le préfixe et le type normalisé, la rue, le nom de la rue, etc. divisés en champs distincts. Cette fonction fonctionne uniquement avec les données de recherche fournies avec le géocodeur tiger (pas besoin pour les données de recensement tiger). Nécessite l'extension address_standardizer.
- Reverse_Geocode - Amélioration : 2.4.1 si le jeu de données optionnel zcta5 est chargé, la fonction reverse_geocode peut résoudre l'état et le code postal même si les données spécifiques à l'état ne sont pas chargées. Voir pour plus de détails sur le chargement des données zcta5. Prend un point géométrique dans un système spatial connu et renvoie un enregistrement contenant un tableau d'adresses théoriquement possibles et un tableau de rues transversales. Si include_strnum_range = true, la plage de rues est incluse dans les rues transversales.
- ST_AsTWKB - Amélioration : 2.4.0 amélioration de la mémoire et de la vitesse. Renvoie la géométrie sous forme de TWKB, diminutif de "Tiny Well-Known Binary"
- ST_Covers - Amélioration : 2.4.0 Ajout de la prise en charge des polygones dans les polygones et des lignes dans les polygones pour le type geography Tests si chaque point de B est situé dans A
- ST_CurveToLine - Amélioration : 2.4.0 a ajouté la prise en charge de la tolérance de l'écart maximal et de l'angle maximal, ainsi que de la sortie symétrique. Convertit une géométrie contenant des courbes en une géométrie linéaire.

- ST_Project - Amélioration : 2.4.0 Autorise les distances négatives et les azimuts non normalisés. Renvoie un point projeté à partir d'un point de départ en fonction d'une distance et d'un azimut.
- ST_Reverse - Amélioration : la prise en charge des courbes a été introduite dans la version 2.4.0. Retourne la géométrie avec l'ordre des sommets inversé.

Fonctions modifiées dans PostGIS 2.4

- = - Modifié : 2.4.0, dans les versions précédentes, il s'agissait d'une égalité de boîte de délimitation et non d'une égalité géométrique. Si vous avez besoin d'une égalité de boîte de délimitation, utilisez à la place. Renvoie TRUE si les coordonnées et l'ordre des coordonnées de la géométrie/géographie A sont les mêmes que les coordonnées et l'ordre des coordonnées de la géométrie/géographie B.
- ST_Node - Modifié : 2.4.0 cette fonction utilise GEOSNode en interne au lieu de GEOSUnaryUnion. Cela peut entraîner un ordre et une direction différents des lignes résultantes par rapport à PostGIS < 2.4. Nœuds d'une collection de lignes.

13.12.9 Fonctions PostGIS nouvelles ou améliorées en 2.3

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 2.3

- &&&(geometry,gidx) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si la boîte de délimitation n-D (en cache) d'une géométrie intersecte une boîte de délimitation de précision flottante n-D (GIDX).
- &&&(gidx,geometry) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si une boîte de délimitation de précision flottante n-D (GIDX) intersecte la boîte de délimitation n-D (mise en cache) d'une géométrie.
- &&&(gidx,gidx) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si deux boîtes de délimitation (GIDX) de précision flottante n-D se croisent.
- &&(box2df,box2df) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si deux boîtes de délimitation 2D à précision flottante (BOX2DF) se croisent.
- &&(box2df,geometry) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si une boîte de délimitation 2D de précision flottante (BOX2DF) intersecte la boîte de délimitation 2D (mise en cache) d'une géométrie.
- &&(geometry,box2df) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si la boîte de délimitation 2D (en cache) d'une géométrie intersecte une boîte de délimitation 2D de précision flottante (BOX2DF).
- @(box2df,box2df) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans une autre boîte de délimitation de précision flottante 2D.
- @(box2df,geometry) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) est contenue dans la boîte de délimitation 2D d'une géométrie.
- @(geometry,box2df) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si la boîte de délimitation 2D d'une géométrie est contenue dans une boîte de délimitation 2D à précision flottante (BOX2DF).
- Populate_Topology_Layer - Disponibilité : 2.3.0 Ajoute les entrées manquantes à la table topology.layer en lisant les métadonnées des tables topo.
- ST_ClusterDBSCAN - Disponibilité : 2.3.0 Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie d'entrée en utilisant l'algorithme DBSCAN.

- ST_ClusterKMeans - Disponibilité : 2.3.0 Fonction Window qui renvoie un identifiant de cluster pour chaque géométrie en entrée en utilisant l'algorithme K-means.
- ST_GeneratePoints - Disponibilité : 2.3.0 Génère un multipoint de points aléatoires contenus dans un polygone ou un multipolygone.
- ST_GeometricMedian - Disponibilité : 2.3.0 Renvoie la médiane géométrique d'un MultiPoint.
- ST_MakeLine - Disponibilité : 2.3.0 - La prise en charge des éléments d'entrée MultiPoint a été introduite Crée une LineString à partir de géométries Point, MultiPoint ou LineString.
- ST_MinimumBoundingRadius - Disponibilité - 2.3.0 Renvoie le point central et le rayon du plus petit cercle contenant une géométrie.
- ST_MinimumClearance - Disponibilité : 2.3.0 Renvoie la clearance (le dégagement) d'une géométrie, une mesure de la robustesse d'une géométrie.
- ST_MinimumClearanceLine - Disponibilité : 2.3.0 - nécessite GEOS >= 3.6.0 Renvoie la chaîne de lignes à deux points couvrant le dégagement (clearance) minimum d'une géométrie.
- ST_Normalize - Disponibilité : 2.3.0 Renvoie la géométrie sous sa forme canonique.
- ST_Points - Disponibilité : 2.3.0 Renvoie un MultiPoint contenant les coordonnées d'une géométrie.
- ST_VoronoiLines - Disponibilité : 2.3.0 Renvoie les limites des polygones de Voronoï des sommets d'une géométrie.
- ST_VoronoiPolygons - Disponibilité : 2.3.0 Renvoie les cellules de la représentation de Voronoï des sommets d'une géométrie.
- ST_WrapX - Disponibilité : 2.3.0 nécessite GEOS Enveloppe une géométrie autour d'une valeur X.
- TopoGeom_addElement - Disponibilité : 2.3 Ajoute un élément à la définition d'une TopoGeometry.
- TopoGeom_remElement - Disponibilité : 2.3 Supprime un élément de la définition d'une TopoGeometry.
- ~(box2df,box2df) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient une autre boîte de délimitation de précision flottante 2D (BOX2DF).
- ~(box2df,geometry) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si une boîte de délimitation de précision flottante 2D (BOX2DF) contient la boîte de délimitation 2D d'une géométrie.
- ~(geometry,box2df) - Disponibilité : 2.3.0 le support des Block Range INdexes (BRIN) a été introduit. Nécessite PostgreSQL 9.5+. Renvoie TRUE si la boîte de délimitation 2D d'une géométrie contient une boîte de délimitation de précision flottante 2D (GIDX).

Fonctions améliorées dans PostGIS 2.3

- ST_Contains - Amélioré : 2.3.0 Amélioration du court-circuit PIP étendu à la prise en charge des multipoints avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones. Tests si chaque point de B est situé dans A, et que leurs intérieurs ont un point commun
- ST_Covers - Amélioration : 2.3.0 Amélioration du court-circuit PIP pour la géométrie étendue à la prise en charge des multipoints avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones. Tests si chaque point de B est situé dans A
- ST_Expand - Amélioration : 2.3.0 : prise en charge de l'expansion d'une boîte par différentes quantités dans différentes dimensions. Renvoie une boîte de délimitation développée à partir d'une autre boîte de délimitation ou d'une géométrie.
- ST_Intersects - Amélioré : 2.3.0 Amélioration du court-circuit PIP étendu à la prise en charge des multipoints avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones. Teste si deux géométries se croisent (elles ont au moins un point en commun)

- ST_Segmentize - Amélioration : 2.3.0 La segmentation d'objets geography produit désormais des sous-segments de longueur égale Renvoie une geometry/geography modifiée dont aucun segment ne dépasse une distance donnée.
- ST_Transform - Amélioration : la version 2.3.0 a introduit la prise en charge du texte PROJ.4 direct. Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent.
- ST_Within - Amélioration : 2.3.0 Amélioration du court-circuit PIP pour la géométrie étendue à la prise en charge des multi-points avec peu de points. Les versions précédentes ne prenaient en charge que les points dans les polygones. Tests si chaque point de A se trouve dans B, et que leurs intérieurs ont un point commun

Fonctions modifiées dans PostGIS 2.3

- ST_PointN - Modifié : 2.3.0 : indexation négative disponible (-1 est le dernier point) Renvoie le Nième point de la première LineString ou LineString circulaire d'une géométrie.

13.12.10 Fonctions PostGIS nouvelles ou améliorées en 2.2

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 2.2

- <<->> - Disponibilité : 2.2.0 -- KNN disponible uniquement pour PostgreSQL 9.1+ Renvoie la distance n-D entre les géométries A et B ou les boîtes englobantes
- ST_3DDifference - Disponibilité : 2.2.0 Effectuer une différence 3D
- ST_3DUnion - Disponibilité : 2.2.0 Effectuer l'union 3D.
- ST_ApproximateMedialAxis - Disponibilité : 2.2.0 Calculer l'axe médian approximatif d'une géométrie aréolaire.
- ST_AsEncodedPolyline - Disponibilité : 2.2.0 Renvoie une polyligne encodée à partir d'une géométrie LineString.
- ST_AsTWKB - Disponibilité : 2.2.0 Renvoie la géométrie sous forme de TWKB, diminutif de "Tiny Well-Known Binary"
- ST_BoundingDiagonal - Disponibilité : 2.2.0 Retourne la diagonale de la boîte englobante pour la géométrie en argument.
- ST_CPAWithin - Disponibilité : 2.2.0 Teste si le point d'approche le plus proche de deux trajectoires se trouve dans la distance spécifiée.
- ST_ClipByBox2D - Disponibilité : 2.2.0 Calcule la partie d'une géométrie située à l'intérieur d'un rectangle.
- ST_ClosestPointOfApproach - Disponibilité : 2.2.0 Renvoie une mesure au point d'approche le plus proche de deux trajectoires.
- ST_ClusterIntersecting - Disponibilité : 2.2.0 Fonction d'agrégation qui regroupe les géométries en entrée en ensembles connectés.
- ST_ClusterWithin - Disponibilité : 2.2.0 Fonction agrégée qui regroupe les géométries en fonction de la distance de séparation.
- ST_CountAgg - Disponibilité : 2.2.0 Agrégat. Renvoie le nombre de pixels dans une bande donnée d'un ensemble de rasters. Si aucune bande n'est spécifiée, la valeur par défaut est la bande 1. Si exclude_nodata_value est true, seuls les pixels différents de la valeur NODATA seront comptés.
- ST_CreateOverview - Disponibilité : 2.2.0 Créé une version à plus faible résolution d'une colonne raster.
- ST_DistanceCPA - Disponibilité : 2.2.0 Renvoie la distance entre le point d'approche le plus proche de deux trajectoires.
- ST_ForceCurve - Disponibilité : 2.2.0 Retransformation d'une géométrie dans son type de courbure, le cas échéant.
- ST_IsPlanar - Disponibilité : 2.2.0 : Ce point était documenté dans la version 2.1.0 mais a été accidentellement omis dans la version 2.1. Vérifie si une surface est planaire ou non
- ST_IsSolid - Disponibilité : 2.2.0 Teste si la géométrie est un solide. Aucun contrôle de validité n'est effectué.

- ST_IsValidTrajectory - Disponibilité : 2.2.0 Teste si la géométrie est une trajectoire valide.
- ST_LineFromEncodedPolyline - Disponibilité : 2.2.0 Crée une LineString depuis une polyligne encodée ("Encoded Polyline").
- ST_MakeSolid - Disponibilité : 2.2.0 Transformer la géométrie dans un solide. Aucune vérification n'est effectuée. Pour obtenir un solide valide, la géométrie d'entrée doit être une surface polyédrique fermée ou un TIN fermé.
- ST_MapAlgebra (callback function version) - Disponibilité : 2.2.0 : Possibilité d'ajouter un masque Version avec fonction de rappel - Retourne un raster à une bande à partir d'un ou plusieurs rasters d'entrée, d'index de bandes et d'une fonction de rappel spécifiée par l'utilisateur.
- ST_MemSize - Disponibilité : 2.2.0 Retourne l'espace utilisé par le raster (en octets).
- ST_RemoveRepeatedPoints - Disponibilité : 2.2.0 Renvoie une version d'une géométrie dont les points en double ont été supprimés.
- ST_Retile - Disponibilité : 2.2.0 Retourne un ensemble de tuiles configurées à partir d'une couverture raster composée de tuiles arbitraires.
- ST_SetEffectiveArea - Disponibilité : 2.2.0 Définit la surface effective de chaque sommet, en utilisant l'algorithme Visvalingam-Whyatt.
- ST_SimplifyVW - Disponibilité : 2.2.0 Renvoie une représentation simplifiée d'une géométrie, en utilisant l'algorithme de Visvalingam-Whyatt
- ST_Subdivide - Disponibilité : 2.2.0 Calcule une subdivision rectiligne d'une géométrie.
- ST_SummaryStatsAgg - Disponibilité : 2.2.0 Agrégat. Retourne des résumés statistiques (count, sum, mean, stddev, min, max) pour une bande raster spécifiée pour une ensemble de rasters. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_SwapOrdinates - Disponibilité : 2.2.0 Renvoie une version de la géométrie donnée avec les valeurs d'ordonnées permutées.
- ST_Volume - Disponibilité : 2.2.0 Calcule le volume d'un solide 3D. S'il est appliqué à des géométries de surface (même fermées), il renvoie 0.
- parse_address - Disponibilité : 2.2.0 Prend une adresse d'une ligne et la décompose en plusieurs parties
- postgis.enable_outdb_rasters - Disponibilité : 2.2.0 Une option de configuration booléenne pour permettre l'accès aux bandes matricielles de out-db.
- postgis.gdal_datapath - Disponibilité : 2.2.0 Une option de configuration pour régler la valeur de l'option GDAL_DATA de GDAL. Si elle n'est pas assignée, la valeur de la variable d'environnement GDAL_DATA est utilisée.
- postgis.gdal_enabled_drivers - Disponibilité : 2.2.0 Option de configuration permettant de définir les drivers GDAL activés dans l'environnement PostGIS. Affecte la variable de configuration GDAL GDAL_SKIP.
- standardize_address - Disponibilité : 2.2.0 Renvoie une forme stdaddr d'une adresse d'entrée en utilisant les tables lex, gaz et rule.
- |=| - Disponibilité : 2.2.0. La prise en charge des index est disponible uniquement pour PostgreSQL 9.5+ Renvoie la distance entre les trajectoires A et B à leur point d'approche le plus proche.

Fonctions améliorées dans PostGIS 2.2

- <-> - Amélioré : 2.2.0 -- Comportement KNN ("K nearest neighbor") réel pour la géométrie et la géographie pour PostgreSQL 9.5+. Note : pour la géographie, KNN est basé sur la sphère plutôt que sur le sphéroïde. Pour PostgreSQL 9.4 et moins, le support de la géographie est nouveau mais ne supporte que le centroïde de la boîte de délimitation. Renvoie la distance en 2D entre A et B.
- AsTopoJSON - Amélioration : 2.2.1 ajout de la prise en charge des entrées "puntal" Renvoie la représentation TopoJSON d'une topogeometry.

- ST_Area - Amélioration : 2.2.0 - mesure sur sphéroïde effectuée avec GeographicLib pour une meilleure précision et robustesse. Nécessite PROJ >= 4.9.0 pour profiter de la nouvelle fonctionnalité. Renvoie l'aire d'une géométrie polygonale.
- ST_AsX3D - Amélioration : 2.2.0 : Prise en charge des coordonnées géographiques et de l'inversion des axes (x/y, long/lat). Voir les options pour plus de détails. Renvoie une géométrie au format X3D xml node element : ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Azimuth - Amélioration : 2.2.0 mesure sur sphéroïde effectuée avec GeographicLib pour améliorer la précision et la robustesse. Nécessite PROJ >= 4.9.0 pour profiter de la nouvelle fonctionnalité. Renvoie l'azimut basé sur le nord d'une ligne entre deux points.
- ST_Distance - Amélioration : 2.2.0 - mesure sur sphéroïde effectuée avec GeographicLib pour une meilleure précision et robustesse. Nécessite PROJ >= 4.9.0 pour profiter de la nouvelle fonctionnalité. Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_Scale - Amélioration : La prise en charge de la mise à l'échelle de toutes les dimensions (paramètre factor) a été introduite dans la version 2.2.0. Met à l'échelle une géométrie en fonction de facteurs donnés.
- ST_Split - Amélioration : la version 2.2.0 prend en charge la division d'une ligne par une limite multiligne, multipoint ou (multi)polygone. Renvoie une collection de géométries créées en divisant une géométrie par une autre géométrie.
- ST_Summary - Amélioré : 2.2.0 Ajout de la prise en charge des TIN et des courbes Renvoie un résumé textuel du contenu d'une géométrie.

Fonctions modifiées dans PostGIS 2.2

- <-> - Modifié : 2.2.0 -- Pour les utilisateurs de PostgreSQL 9.5, l'ancienne syntaxe Hybrid peut être plus lente, donc vous voudrez vous débarrasser de ce hack si vous exécutez votre code uniquement sur PostGIS 2.2+ 9.5+. Voir les exemples ci-dessous. Renvoie la distance en 2D entre A et B.
- Get_Geocode_Setting - Modifié : 2.2.0 : les paramètres par défaut sont désormais conservés dans une table appelée geocode_settings_d Les paramètres personnalisés sont dans geocode_settings et ne contiennent que ceux qui ont été définis par l'utilisateur. Renvoie la valeur d'un paramètre spécifique stocké dans la table tiger.geocode_settings.
- ST_3DClosestPoint - Modifié : 2.2.0 - si 2 géométries 2D sont saisies, un point 2D est renvoyé (au lieu de l'ancien comportement supposant 0 pour Z manquant). Dans le cas de 2D et 3D, Z n'est plus supposé être 0 pour Z manquant. Renvoie le point 3D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne 3D la plus courte.
- ST_3DDistance - Modifié : 2.2.0 - Dans le cas de la 2D et de la 3D, Z n'est plus considéré comme égal à 0 en cas de Z manquant. Renvoie la distance cartésienne minimale en 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DLongestLine - Modifié : 2.2.0 - si 2 géométries 2D sont saisies, un point 2D est renvoyé (au lieu de l'ancien comportement supposant 0 pour Z manquant). Dans le cas de 2D et 3D, Z n'est plus supposé être 0 pour Z manquant. Renvoie la ligne 3D la plus longue entre deux géométries
- ST_3DMaxDistance - Modifié : 2.2.0 - Dans le cas de la 2D et de la 3D, Z n'est plus considéré comme égal à 0 en cas de Z manquant. Renvoie la distance maximale cartésienne 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DShortestLine - Modifié : 2.2.0 - si 2 géométries 2D sont saisies, un point 2D est renvoyé (au lieu de l'ancien comportement supposant 0 pour Z manquant). Dans le cas de 2D et 3D, Z n'est plus supposé être 0 pour Z manquant. Renvoie la ligne 3D la plus courte entre deux géométries
- ST_DistanceSphere - Modifié : 2.2.0 Dans les versions antérieures, cette fonction s'appelait ST_Distance_Sphere Renvoie la distance minimale en mètres entre deux géométries lon/lat en utilisant un modèle de terre sphérique.
- ST_DistanceSpheroid - Modifié : 2.2.0 Dans les versions précédentes, cette fonction était appelée ST_Distance_Sphéroïde Renvoie la distance minimale entre deux géométries lon/lat en utilisant un modèle de terre sphéroïdale.
- ST_Equals - Modifié : 2.2.0 Retourne vrai même pour les géométries invalides si elles sont binairement égales Teste si deux géométries comprennent le même ensemble de points

- ST_LengthSpheroid - Modifié : 2.2.0 Dans les versions précédentes, cette fonction s'appelait ST_Length_Spheroid et avait l'alias ST_3DLength_Spheroid Renvoie la longueur/périmètre 2D ou 3D d'une géométrie lon/lat sur un sphéroïde.
- ST_MemSize - Modifié : 2.2.0 nom modifié en ST_MemSize pour respecter la convention de nommage. Renvoie la quantité d'espace mémoire que prend une géométrie.
- ST_PointInsideCircle - Modifié : 2.2.0 Dans les versions précédentes, cette fonction était appelée ST_Point_Inside_Circle Teste si un point géométrique se trouve à l'intérieur d'un cercle défini par un centre et un rayon
- ValidateTopology - Modifié : 2.2.0 les valeurs pour id1 et id2 ont été échangées pour "edge crosses node" pour être cohérent avec la description de l'erreur. Renvoie un ensemble d'objets validatetopology_returntype détaillant les problèmes liés à la topologie.

13.12.11 Fonctions PostGIS nouvelles ou améliorées en 2.1

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 2.1

- = - Disponibilité : 2.1.0 Retourne TRUE si la boîte englobante de A est la même que celle de B. Utilise des boîtes englobantes en double précision.
- AsTopoJSON - Disponibilité : 2.1.0 Renvoie la représentation TopoJSON d'une topogeometry.
- Drop_Nation_Tables_Generate_Script - Disponibilité : 2.1.0 Génère un script qui supprime toutes les tables du schéma spécifié qui commencent par county_all, state_all ou code d'état suivi de county ou state.
- Get_Geocode_Setting - Disponibilité : 2.1.0 Renvoie la valeur d'un paramètre spécifique stocké dans la table tiger.geocode_se
- Loader_Generate_Nation_Script - Disponibilité : 2.1.0 Génère un script shell pour la plate-forme spécifiée qui charge les données dans les tables county et state.
- Pagc_Normalize_Address - Disponibilité : 2.1.0 Étant donné une adresse textuelle, cette fonction renvoie un type composite norm_addy qui contient le suffixe de la route, le préfixe et le type normalisé, la rue, le nom de la rue, etc. divisés en champs distincts. Cette fonction fonctionne uniquement avec les données de recherche fournies avec le géocodeur tiger (pas besoin pour les données de recensement tiger). Nécessite l'extension address_standardizer.
- ST_3DArea - Disponibilité : 2.1.0 Calcule la surface des géométries de surface 3D. Retourne 0 pour les solides.
- ST_3DIntersection - Disponibilité : 2.1.0 Réaliser une intersection 3D
- ST_Box2dFromGeoHash - Disponibilité : 2.1.0 Retourne une BOX2D à partir d'une chaîne GeoHash.
- ST_ColorMap - Disponibilité : 2.1.0 Crée un nouveau raster comprenant jusqu'à quatre bandes 8BUI (niveaux de gris, RGB, RGBA) à partir du raster source et d'une bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_Contains - Disponibilité : 2.1.0 Retourne true si aucun point du raster rastB ne se trouve à l'extérieur du raster rastA et si au moins un point de l'intérieur du raster rastB se trouve à l'intérieur du raster rastA.
- ST_ContainsProperly - Disponibilité : 2.1.0 Retourne true si rastB intersecte l'intérieur de rastA, mais pas la frontière ou l'extérieur de rastA.
- ST_CoveredBy - Disponibilité : 2.1.0 Retourne true si aucun point du raster rastA ne se trouve à l'extérieur du raster rastB.
- ST_Covers - Disponibilité : 2.1.0 Retourne true si aucun point du raster rastB ne se trouve à l'extérieur du raster rastA.
- ST_DFullyWithin - Disponibilité : 2.1.0 Retourne true si les rasters rastA et rastB se trouvent entièrement à une distance donnée l'un de l'autre.
- ST_DWithin - Disponibilité : 2.1.0 Retourne true si les rasters rastA et rastB se trouvent à une distance donnée l'un de l'autre.

- ST_DelaunayTriangles - Disponibilité : 2.1.0 Renvoie la triangulation de Delaunay des sommets d'une géométrie.
- ST_Disjoint - Disponibilité : 2.1.0 Retourne true si le raster rastA n'intersecte pas spatialement le rastB.
- ST_DumpValues - Disponibilité : 2.1.0 Retourne les valeurs d'une bande raster spécifiée, sous forme d'un tableau à deux dimensions.
- ST_Extrude - Disponibilité : 2.1.0 Extruder une surface vers un volume
- ST_ForceLHR - Disponibilité : 2.1.0 Force l'orientation LHR d'un objet
- ST_FromGDALRaster - Disponibilité : 2.1.0 Retourne un raster depuis un fichier raster supporté par GDAL.
- ST_GeomFromGeoHash - Disponibilité : 2.1.0 Retourne une geometry depuis une chaîne GeoHash.
- ST_InvDistWeight4ma - Disponibilité : 2.1.0 Fonction de traitement des données raster qui interpole la valeur d'un pixel à partir de son voisinage.
- ST_MapAlgebra (callback function version) - Disponibilité : 2.1.0 Version avec fonction de rappel - Retourne un raster à une bande à partir d'un ou plusieurs rasters d'entrée, d'index de bandes et d'une fonction de rappel spécifiée par l'utilisateur.
- ST_MapAlgebra (expression version) - Disponibilité : 2.1.0 Version avec expression - Retourne un raster à une bande à partir d'un ou deux rasters d'entrée, d'index de bandes et d'une ou plusieurs expressions SQL spécifiées par l'utilisateur.
- ST_MinConvexHull - Disponibilité : 2.1.0 Retourne la géométrie de l'enveloppe convexe du raster en excluant les pixels NODATA.
- ST_MinDist4ma - Disponibilité : 2.1.0 Fonction de traitement des données raster qui renvoie la distance minimale (en nombre de pixels) entre le pixel d'intérêt et un pixel voisin ayant une valeur.
- ST_MinkowskiSum - Disponibilité : 2.1.0 Effectue la somme de Minkowski
- ST_NearestValue - Disponibilité : 2.1.0 Retourne la valeur la plus proche différent de NODATA pour une bande raster spécifiée au pixel donné par columnx et rowy, ou à un point géométrique spécifié dans le même système de référence spatial que le raster.
- ST_Neighborhood - Disponibilité : 2.1.0 Retourne un tableau 2-D de double avec les valeurs non NODATA autour du pixel de la bande spécifiée, aux coordonnées spécifiées par columnX & rowY ou par un point géométrique dans le même système de référence spatial que le raster.
- ST_NotSameAlignmentReason - Disponibilité : 2.1.0 Retourne un texte indiquant si les rasters sont alignés et, s'ils ne le sont pas, la raison du problème.
- ST_Orientation - Disponibilité : 2.1.0 Détermine l'orientation d'une surface
- ST_Overlaps - Disponibilité : 2.1.0 Retourne true si les raster rastA et rastB intersectent mais que l'un ne contient pas complètement l'autre.
- ST_PixelAsCentroid - Disponibilité : 2.1.0 Retourne le centroïde (point géométrique) de la zone représentée par un pixel.
- ST_PixelAsCentroids - Disponibilité : 2.1.0 Retourne le centroïde (point géométrique) pour chaque pixel de la bande raster, avec sa valeur et les coordonnées raster X et Y. Le point géométrique est le centroïde de la zone représentée par un pixel.
- ST_PixelAsPoint - Disponibilité : 2.1.0 Retourne le point géométrique du coin supérieur gauche du pixel.
- ST_PixelAsPoints - Disponibilité : 2.1.0 Retourne un point géométrique pour chaque pixel de la bande raster, avec sa valeur et ses coordonnées raster X et Y. Les coordonnées du points sont ceux du coin supérieur gauche du pixel.
- ST_PixelOfValue - Disponibilité : 2.1.0 Retourne les coordonnées columnx, rowy du pixel dont la valeur est égale à la valeur recherchée.
- ST_PointFromGeoHash - Disponibilité : 2.1.0 Retourne un point à partir d'une chaîne GeoHash.

- ST_RasterToWorldCoord - Disponibilité : 2.1.0 Retourne le coin supérieur gauche du raster, sous forme de coordonnées X et Y (longitude et latitude) d'un colonne et d'une ligne. Les numéros de colonne et de ligne commencent à 1.
- ST_Resize - Disponibilité : 2.1.0 Nécessite GDAL 1.6.1+ Redimensionne un raster à une nouvelle largeur/hauteur
- ST_Roughness - Disponibilité : 2.1.0 Retourne un raster avec la rugosité d'un MNT.
- ST_SetValues - Disponibilité : 2.1.0 Retourne un nouveau raster en modifiant les valeurs de certains pixels d'une bande spécifiée.
- ST_Simplify - Disponibilité : 2.1.0 Renvoie une version géométrique "simplifiée" de la TopoGeometry donnée en utilisant l'algorithme de Douglas-Peucker.
- ST_StraightSkeleton - Disponibilité : 2.1.0 Calcule un squelette (straight skeleton) à partir d'une géométrie
- ST_Summary - Disponibilité : 2.1.0 Retourne un résumé du contenu du raster sous forme de texte.
- ST_TPI - Disponibilité : 2.1.0 Retourne un raster avec l'index de position topographique (TPI) calculé.
- ST_TRI - Disponibilité : 2.1.0 Retourne un raster avec l'indice de rugosité du terrain (TRI) calculé.
- ST_Tesselate - Disponibilité : 2.1.0 Effectue la tessellation de la surface d'un polygone ou d'une surface polyédrique et renvoie un TIN ou une collection de TINS
- ST_Tile - Disponibilité : 2.1.0 Retourne un ensemble de rasters issus de la division d'un raster d'entrée selon les dimensions spécifiées pour les rasters de sortie.
- ST_Touches - Disponibilité : 2.1.0 Retourne true si les raster rastA et rastB ont au moins un point en commun mais que leurs intérieurs n'intersectent pas.
- ST_Union - Disponibilité : 2.1.0 Ajout de la variante ST_Union(rast, unionarg). Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.
- ST_Within - Disponibilité : 2.1.0 Retourne true si aucun point du raster rastA ne se trouve à l'extérieur du raster rastB et si au moins un point de l'intérieur du raster rastA se trouve à l'intérieur du raster rastB.
- ST_WorldToRasterCoord - Disponibilité : 2.1.0 Retourne le coin supérieur gauche comme colonne et ligne, en fonction de coordonnées géométriques X et Y (longitude et latitude) ou un point géométrique dans le système de référence spatial du raster.
- Set_Geocode_Setting - Disponibilité : 2.1.0 Définit un paramètre qui affecte le comportement des fonctions du géocodeur.
- UpdateRasterSRID - Disponibilité : 2.1.0 Change le SRID de tous les rasters dans la table et colonne en paramètres.
- clearTopoGeom - Disponibilité : 2.1 Efface le contenu d'une géométrie topo.
- postgis_sfcgal_version - Disponibilité : 2.1.0 Retourne la version de SFCGAL utilisée

Fonctions améliorées dans PostGIS 2.1

- ST_AddBand - Amélioration : 2.1.0 ajout du paramètre addbandarg. Retourne un raster avec la/les nouvelle(s) bande(s) ajouté(s) à un index donné, de type et valeur initiale donnés. Si aucun index n'est spécifié, la bande est ajoutée à la fin.
- ST_AddBand - Amélioration : 2.1.0 introduction du support des bandes out-db. Retourne un raster avec la/les nouvelle(s) bande(s) ajouté(s) à un index donné, de type et valeur initiale donnés. Si aucun index n'est spécifié, la bande est ajoutée à la fin.
- ST_AsBinary/ST_AsWKB - Amélioration : 2.1.0 Ajout de outasin Retourne la représentation Well-Known Binary (WKB) de ce raster.
- ST_AsGML - Amélioration : 2.1.0 La prise en charge des identifiants a été introduite pour GML 3. Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.

- ST_Aspect - Amélioration : 2.1.0 Utilise ST_MapAlgebra() et ajout du paramètre optionnel interpolate_nodata Retourne l'exposition (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_Boundary - Amélioration : 2.1.0 introduction du support pour Triangle Renvoie la limite d'une géométrie.
- ST_Clip - Amélioration : 2.1.0 Réécrit en C Returns the raster clipped by the input geometry. If band number is not specified, all bands are processed. If crop is not specified or TRUE, the output raster is cropped. If touched is set to TRUE, then touched pixels are included, otherwise only if the center of the pixel is in the geometry it is included.
- ST_DWithin - Amélioration : la version 2.1.0 a amélioré la vitesse de la géographie. Voir Making Geography faster pour plus de détails. Teste si deux géométries se trouvent à une distance donnée
- ST_DWithin - Amélioration : la prise en charge des géométries courbes a été introduite dans la version 2.1.0. Teste si deux géométries se trouvent à une distance donnée
- ST_Distance - Amélioration : la version 2.1.0 a amélioré la vitesse pour le type geography. Voir Making Geography faster pour plus de détails. Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_Distance - Amélioration : 2.1.0 - la prise en charge des géométries courbes a été introduite. Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_Distinct4ma - Amélioration : 2.1.0 Ajout de la variante 2 Fonction de traitement des données raster qui calcule le nombre de valeurs de pixels uniques dans un voisinage.
- ST_DumpPoints - Amélioré : 2.1.0 Vitesse plus rapide. Réimplémentation en C natif. Renvoie un ensemble de lignes geometry_dump pour les coordonnées dans une géométrie.
- ST_HillShade - Amélioration : 2.1.0 Utilise ST_MapAlgebra() et ajout du paramètre optionnel interpolate_nodata Retourne l'hypothétique éclairage d'une bande raster d'élévation en utilisant les valeurs d'azimut, d'altitude, de luminosité et d'échelle fournies.
- ST_MakeValid - Amélioration : 2.1.0, ajout du support pour GEOMETRYCOLLECTION et MULTIPOINT. Tente de rendre valide une géométrie invalide sans perdre de sommets.
- ST_Max4ma - Amélioration : 2.1.0 Ajout de la variante 2 Fonction de traitement des données raster qui calcule la valeur maximale d'un pixel dans un voisinage.
- ST_Mean4ma - Amélioration : 2.1.0 Ajout de la variante 2 Fonction de traitement des données raster qui calcule la valeur moyenne d'un pixel dans un voisinage.
- ST_Min4ma - Amélioration : 2.1.0 Ajout de la variante 2 Fonction de traitement des données raster qui calcule la valeur minimale d'un pixel dans un voisinage.
- ST_PixelAsPolygons - Amélioration : 2.1.0 ajout du paramètre optionnel exclude_nodata_value. Retourne la géométrie polygonale qui délimite chaque pixel de la bande raster, avec la valeur et les coordonnées raster X et Y de chaque pixel.
- ST_Polygon - Amélioration : 2.1.0 Vitesse améliorée (entièrement en C) et le multipolygone retourné est garanti d'être valide. Retourne une géométrie multipolygonale formée par l'union des pixels dont la valeur est différente de nodata. Si aucune bande bandnum n'est spécifiée, la bande 1 est utilisée.
- ST_Range4ma - Amélioration : 2.1.0 Ajout de la variante 2 Fonction de traitement des données raster qui calcule la plage de valeurs des pixels dans un voisinage.
- ST_SameAlignment - Amélioration : 2.1.0 ajout de la variante agrégée Retourne true si les rasters ont les mêmes skew, scale, spatial ref, et offset (les pixels peuvent être placés sur la même grille sans être coupés) et false si ce n'est pas le cas, avec une NOTICE détaillant le problème.
- ST_Segmentize - Amélioration : la prise en charge des objets de type geography a été introduite dans la version 2.1.0. Renvoie une geometry/geography modifiée dont aucun segment ne dépasse une distance donnée.
- ST_SetGeoReference - Amélioration : 2.1.0 Ajout de la variante ST_SetGeoReference(raster, double precision, ...) Définit les 6 paramètres de géo-référencement en un seul appel. Les nombres doivent être séparés par un espace. Accepte les formats GDAL (par défaut) ou ESRI.

- ST_SetValue - Amélioration : 2.1.0 La variante de ST_SetValue() avec la géométrie supporte n'importe quelle type de géométrie, pas uniquement des points. La variante avec la géométrie est une enveloppe autour de la variante de ST_SetValues() avec geomval[] Retourne un nouveau raster en modifiant la valeur du pixel pour la bande spécifiée et aux coordonnées columnx, rowy, ou pour tous les pixels qui intersectent une géométrie spécifiée. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié.
- ST_Slope - Amélioration : 2.1.0 Utilise ST_MapAlgebra() et ajout des paramètres optionnels units, scale, interpolate_nodata Retourne la pente (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_StdDev4ma - Amélioration : 2.1.0 Ajout de la variante 2 Fonction de traitement des données raster qui calcule l'écart type des valeurs des pixels dans un voisinage.
- ST_Sum4ma - Amélioration : 2.1.0 Ajout de la variante 2 Fonction de traitement des données raster qui calcule la somme de toutes les valeurs de pixels dans un voisinage.
- ST_Summary - Amélioré : 2.1.0 Indicateur S pour indiquer si le système de référence spatiale est connu Renvoie un résumé textuel du contenu d'une géométrie.
- ST_Transform - Amélioration : 2.1.0 Ajout de la variante ST_Transform(rast, alignto) Reprojette un raster depuis un système de référence spatial vers un autre, en utilisant l'algorithme de rééchantillonnage spécifié. Les algorithmes possibles sont NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.
- ST_Union - Amélioration : 2.1.0 Vitesse améliorée (entièrement en C). Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.
- ST_Union - Amélioration : 2.1.0 ST_Union(rast) (variante 1) permet l'union sur toutes les bandes de tous les rasters d'entrée. Les versions précédentes de PostGIS ne prenaient en compte que la première bande. Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.
- ST_Union - Amélioration : 2.1.0 ST_Union(rast, uniontype) (variante 4) permet l'union de toutes les bandes de tous les rasters d'entrée. Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.
- toTopoGeom - Amélioré : 2.1.0 ajoute la version d'une TopoGeometry existante. Convertit une géométrie simple en une géométrie topographique.

Fonctions modifiées dans PostGIS 2.1

- ST_Aspect - Changement : 2.1.0 Dans les versions précédentes, les valeurs retournées étaient en radians. Désormais, les valeurs retournées sont par défaut en degrés Retourne l'exposition (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_EstimatedExtent - Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Estimated_Extent. Renvoie l'étendue estimée d'une table spatiale.
- ST_Force2D - Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_2D. Forcer les géométries à passer en "mode bidimensionnel".
- ST_Force3D - Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_3D. Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ.
- ST_Force3DM - Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_3DM. Forcer les géométries en mode XYM.
- ST_Force3DZ - Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_3DZ. Forcer les géométries en mode XYZ.
- ST_Force4D - Modifié : 2.1.0. Jusqu'à la version 2.0.x, elle s'appelait ST_Force_4D. Forcer les géométries en mode XYZM.
- ST_ForceCollection - Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Force_Collection. Convertir la géométrie en une GEOMETRYCOLLECTION.

- ST_HillShade - Changement : 2.1.0 Dans les versions précédentes, l'azimut et l'altitude étaient exprimés en radians. Désormais, l'azimut et l'altitude sont exprimés en degrés Retourne l'hypothétique éclairage d'une bande raster d'élévation en utilisant les valeurs d'azimut, d'altitude, de luminosité et d'échelle fournies.
- ST_LineInterpolatePoint - Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Line_Interpolate_Point. Renvoie un point interpolé le long d'une ligne à un emplacement fractionnaire.
- ST_LineLocatePoint - Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Line_Locate_Point. Renvoie l'emplacement fractionnaire du point le plus proche d'un point sur une ligne.
- ST_LineSubstring - Modifié : 2.1.0. Jusqu'à la version 2.0.x, cette fonction était appelée ST_Line_Substring. Renvoie la partie d'une ligne située entre deux emplacements fractionnaires.
- ST_PixelAsCentroids - Changement : 2.1.1 Changement du comportement de behavior of exclude_nodata_value. Retourne le centroïde (point géométrique) pour chaque pixel de la bande raster, avec sa valeur et les coordonnées raster X et Y. Le point géométrique est le centroïde de la zone représentée par un pixel.
- ST_PixelAsPoints - Changement : 2.1.1 Changement du comportement de behavior of exclude_nodata_value. Retourne un point géométrique pour chaque pixel de la bande raster, avec sa valeur et ses coordonnées raster X et Y. Les coordonnées du points sont ceux du coin supérieur gauche du pixel.
- ST_PixelAsPolygons - Changement : 2.1.1 Changement du comportement de behavior of exclude_nodata_value. Retourne la géométrie polygonale qui délimite chaque pixel de la bande raster, avec la valeur et les coordonnées raster X et Y de chaque pixel.
- ST_Polygon - Changement : 2.1.0 Dans les versions précédentes, retournait parfois un polygone, retourne désormais toujours un multipolygon. Retourne une géométrie multipolygonale formée par l'union des pixels dont la valeur est différente de nodata. Si aucune bande bandnum n'est spécifiée, la bande 1 est utilisée.
- ST_RasterToWorldCoordX - Changement : 2.1.0 Jusqu'à la version 2.0.x, cette fonction était appelée ST_Raster2WorldCoordX Retourne la coordonnée X du coin supérieur gauche du raster à column et row. Les numéros de colonne et de ligne commencent à 1.
- ST_RasterToWorldCoordY - Changement : 2.1.0 Jusqu'à la version 2.0.x, cette fonction était appelée ST_Raster2WorldCoordY Retourne la coordonnée Y du coin supérieur gauche du raster à column et row. Les numéros de colonne et de ligne commencent à 1.
- ST_Rescale - Changement : 2.1.0 Fonctionne avec les rasters sans SRID Rééchantillonne un raster en ajustant juste son échelle (ou la taille des pixels). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline), Lanczos, Max ou Min. La valeur par défaut est NearestNeighbor.
- ST_Reskew - Changement : 2.1.0 Fonctionne avec les rasters sans SRID Rééchantillonne un raster en ajustant simplement son obliquité (skew, ou paramètre de rotation). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.
- ST_Segmentize - Modifié : 2.1.0 Suite à l'introduction de la prise en charge du type geography, l'utilisation ST_Segmentize('LINESTR 2, 3 4)', 0.5) provoque une erreur de fonction ambiguë. L'entrée doit être correctement typée en tant que geometry ou geography. Utilisez ST_GeomFromText, ST_GeogFromText ou un cast vers le type requis (par exemple, ST_Segmentize('LINESTRING(1 2, 3 4)'::geometry, 0.5)) Renvoie une geometry/geography modifiée dont aucun segment ne dépasse une distance donnée.
- ST_Slope - Changement : 2.1.0 Dans les versions précédentes, les valeurs retournées étaient en radians. Désormais, les valeurs retournées sont par défaut en degrés Retourne la pente (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_SnapToGrid - Changement : 2.1.0 Fonctionne avec les rasters sans SRID Rééchantillonne un raster en l'accrochant sur une grille. Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.

- ST_WorldToRasterCoordX - Changement : 2.1.0 Dans les versions précédentes, cette fonctionne était appelée ST_World2RasterCoord Retourne la colonne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw) exprimés dans le système de référence spatial du raster.
- ST_WorldToRasterCoordY - Changement : 2.1.0 Dans les versions précédentes, cette fonctionne était appelée ST_World2RasterCoord Retourne la ligne dans le raster du point géométrique (pt) ou des coordonnées X et Y (xw, yw) exprimés dans le système de référence spatial du raster.

13.12.12 Fonctions PostGIS nouvelles ou améliorées en 2.0

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 2.0

- && - Disponibilité : 2.0.0 Retourne TRUE si la boite englobante de A intersecte la boite englobante de B.
- &&& - Disponibilité : 2.0.0 Renvoie TRUE si la boîte de délimitation n-D de A intersecte la boîte de délimitation n-D de B.
- <#> - Disponibilité : 2.0.0 -- KNN disponible uniquement pour PostgreSQL 9.1+ Renvoie la distance 2D entre les boîtes de délimitation A et B.
- <-> - Disponibilité : 2.0.0 -- Le KNN fournit des voisins les plus proches basés sur les distances entre les centroïdes géométriques au lieu des distances réelles. Résultats exacts pour les points, inexacts pour tous les autres types. Disponible pour PostgreSQL 9.1+ Renvoie la distance en 2D entre A et B.
- @ - Disponibilité : 2.0.0 ajout de raster @ raster, raster @ geometry Retourne TRUE si la boîte englobante de A est contenue dans celle de B. Utilise des boîtes englobantes en double précision.
- @ - Disponibilité : 2.0.5 ajout de geometry @ raster Retourne TRUE si la boîte englobante de A est contenue dans celle de B. Utilise des boîtes englobantes en double précision.
- AddEdge - Disponibilité : 2.0.0 Ajoute une arête linéaire à la table des arêtes et les points de départ et d'arrivée associés à la table des nœuds de points du schéma topologique spécifié en utilisant la géométrie linéaire spécifiée et renvoie l'identifiant de l'arête nouvelle (ou existante).
- AddFace - Disponibilité : 2.0.0 Enregistre une primitive de face dans une topologie et obtient son identifiant.
- AddNode - Disponibilité : 2.0.0 Ajoute un nœud ponctuel à la table des nœuds dans le schéma topologique spécifié et renvoie le nodeid du nouveau nœud. Si le point existe déjà en tant que nœud, l'identifiant du nœud existant est renvoyé.
- AddOverviewConstraints - Disponibilité : 2.0.0 Marque une colonne raster comme étant un aperçu d'une autre colonne.
- AddRasterConstraints - Disponibilité : 2.0.0 Ajoute des contraintes raster à une table pour une colonne spécifique pour contraindre le système de référence spatial, l'échelle, la taille des blocs, l'alignement, le nombre de bandes, le type de pixels, la contrainte d'unicité spatiale. La table doit être chargée avec des données pour que les contraintes puissent être inférées. Retourne true si les contraintes ont été ajoutées, ou émet une notice sinon.
- AsGML - Disponibilité : 2.0.0 Renvoie la représentation GML d'une topogeometry.
- CopyTopology - Disponibilité : 2.0.0 Copie une topologie (nœuds, arêtes, faces, couches et TopoGeometries) dans un nouveau schéma
- DropOverviewConstraints - Disponibilité : 2.0.0 Supprime les contraintes d'aperçu de la colonne.
- DropRasterConstraints - Disponibilité : 2.0.0 Supprime des contraintes raster PostGIS raster sur une colonne raster d'une table. Utile pour recharger des données ou mettre à jour les données d'une colonne raster.
- Drop_Indexes_Generate_Script - Disponibilité : 2.0.0 Génère un script qui supprime toutes les clés non primaires et les index non uniques sur le schéma tiger et le schéma spécifié par l'utilisateur. Le schéma par défaut est tiger_data si aucun schéma n'est spécifié.
- Drop_State_Tables_Generate_Script - Disponibilité : 2.0.0 Génère un script qui supprime toutes les tables du schéma spécifié qui sont préfixées par l'abréviation de l'état. La valeur par défaut du schéma est tiger_data si aucun schéma n'est spécifié.

- Geocode_Intersection - Disponibilité : 2.0.0 Prend 2 rues qui s'intersectent et un état, une ville, un code postal, et produit un ensemble d'emplacements possibles sur la première rue croisée qui est à l'intersection, comprend également un "geomout" comme emplacement du point en NAD 83 long lat, une adresse_normalisée (addy) pour chaque emplacement, et l'évaluation. Plus la note est basse, plus la correspondance est probable. Les résultats sont triés en fonction de la note la plus basse. Il est possible d'indiquer le nombre maximum de résultats, la valeur par défaut étant de 10. Utilise les données Tiger (edges, faces, addr), la correspondance floue de PostgreSQL (soundex, levenshtein).
- GetEdgeByPoint - Disponibilité : 2.0.0 Trouve l'identifiant d'une arête qui coupe un point donné.
- GetFaceByPoint - Disponibilité : 2.0.0 Recherche la face intersectant un point donné.
- GetNodeByPoint - Disponibilité : 2.0.0 Recherche l'identifiant d'un nœud à un point donné.
- GetNodeEdges - Disponibilité : 2.0 Renvoie un ensemble ordonné d'arêtes incidentes au nœud donné.
- GetRingEdges - Disponibilité : 2.0.0 Renvoie l'ensemble ordonné des identifiants d'arêtes signés rencontrés en évoluant sur un côté d'arête donné.
- GetTopoGeomElements - Disponibilité : 2.0.0 Renvoie un ensemble d'objets topoelement contenant les éléments topologiques element_id,element_type de la TopoGeometry donnée (éléments primitifs).
- GetTopologySRID - Disponibilité : 2.0.0 Renvoie le SRID d'une topologie dans la table topology.topology en fonction du nom de la topologie.
- Get_Tract - Disponibilité : 2.0.0 Renvoie le secteur de recensement ou le champ de la table des secteurs où se trouve la géométrie. Par défaut, le nom court du secteur est renvoyé.
- Install_Missing_Indexes - Disponibilité : 2.0.0 Recherche toutes les tables dont les colonnes clés sont utilisées dans les jointures du géocodeur et les conditions de filtrage qui n'ont pas d'index utilisés sur ces colonnes et les ajoute.
- Loader_Generate_Census_Script - Disponibilité : 2.0.0 Génère un script shell pour la plate-forme spécifiée et les états spécifiés qui téléchargera les tables de données Tiger census state tract, bg et tabblocks, les structurera et les chargera dans le schéma tiger_data. Chaque script d'état est renvoyé sous la forme d'un enregistrement distinct.
- Loader_Generate_Script - Disponibilité : 2.0.0 pour prendre en charge les données structurées de Tiger 2010 et charger les tableaux de secteurs de recensement (tract), de groupes d'îlots (bg) et d'îlots (tabblocks). Génère un script shell pour la plateforme spécifiée et les états spécifiés qui téléchargera les données Tiger, les structurera et les chargera dans le schéma tiger_data. Chaque script d'état est renvoyé sous la forme d'un enregistrement séparé. La dernière version prend en charge les modifications structurelles de Tiger 2010 et charge également les tableaux de secteurs de recensement, de groupes d'îlots et d'îlots.
- Missing_Indexes_Generate_Script - Disponibilité : 2.0.0 Recherche toutes les tables dont les colonnes clés sont utilisées dans les jointures du géocodeur et qui n'ont pas d'index sur ces colonnes, et génère le DDL SQL permettant de définir l'index pour ces tables.
- Polygonize - Disponibilité : 2.0.0 Recherche et enregistre toutes les faces définies par les arêtes de la topologie.
- Reverse_Geocode - Disponibilité : 2.0.0 Prend un point géométrique dans un système spatial connu et renvoie un enregistrement contenant un tableau d'adresses théoriquement possibles et un tableau de rues transversales. Si include_strnum_range = true, la plage de rues est incluse dans les rues transversales.
- ST_3DClosestPoint - Disponibilité : 2.0.0 Renvoie le point 3D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne 3D la plus courte.
- ST_3DDFullyWithin - Disponibilité : 2.0.0 Teste si deux géométries 3D sont entièrement comprises dans une distance 3D donnée
- ST_3DDWithin - Disponibilité : 2.0.0 Teste si deux géométries 3D se trouvent à une distance 3D donnée
- ST_3DDistance - Disponibilité : 2.0.0 Renvoie la distance cartésienne minimale en 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DIntersects - Disponibilité : 2.0.0 Teste si deux géométries se croisent dans l'espace en 3D - uniquement pour les points, les lignes, les polygones, les surfaces polyédriques (aire)

- ST_3DLongestLine - Disponibilité : 2.0.0 Renvoie la ligne 3D la plus longue entre deux géométries
- ST_3DMaxDistance - Disponibilité : 2.0.0 Renvoie la distance maximale cartésienne 3D (basée sur la référence spatiale) entre deux géométries en unités projetées.
- ST_3DShortestLine - Disponibilité : 2.0.0 Renvoie la ligne 3D la plus courte entre deux géométries
- ST_AddEdgeModFace - Disponibilité : 2.0 Ajoutez une nouvelle arête et, si elle divise une face, modifiez la face d'origine et ajoutez une nouvelle face.
- ST_AddEdgeNewFaces - Disponibilité : 2.0 Ajoutez une nouvelle arête et, si elle divise une face, supprimez la face d'origine et remplacez-la par deux nouvelles faces.
- ST_AsGDALRaster - Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0. Retourne la tuile raster dans le format GDAL raster spécifié. Les formats raster disponibles sont ceux supportés par votre bibliothèque compilée. Utilisez ST_GDALDrivers() pour obtenir la liste des formats supportés par votre bibliothèque.
- ST_AsJPEG - Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0. Retourne les bandes sélectionnées du raster sous la forme d'une image JPEG (sous forme de tableau d'octets). Si aucune bande n'est spécifiée, et que le raster a 1 ou plus de 3 bandes, seule la première bande est utilisée. Si le raster a exactement 3 bandes, les 3 bandes sont utilisées et mappées en RGB.
- ST_AsLatLonText - Disponibilité : 2.0 Renvoie la représentation en degrés, minutes et secondes du point donné.
- ST_AsPNG - Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0. Retourne les bandes sélectionnées du raster sous la forme d'une image PNG (sous forme de tableau d'octets). Si aucune bande n'est spécifiée et que le raster a 1, 3 ou 4 bandes, toutes les bandes sont utilisées. Si aucune bande n'est spécifiée et que le raster a 2 ou plus de 4 bandes, seule la bande 1 est utilisée. Les bandes sont mappées en RGB ou RGBA.
- ST_AsRaster - Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0. Convertit une géométrie PostGIS en un raster PostGIS.
- ST_AsTIFF - Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0. Retourne les bandes sélectionnées du raster sous la forme d'une seule image TIFF (sous forme de tableau d'octets). Si aucune bande n'est spécifiée ou si l'une des bandes spécifiées n'existe pas dans le raster, toutes les bandes sont utilisées.
- ST_AsX3D - Disponibilité : 2.0.0: ISO-IEC-19776-1.2-X3DEncodings-XML Renvoie une géométrie au format X3D xml node element : ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Aspect - Disponibilité : 2.0.0 Retourne l'exposition (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_Band - Disponibilité : 2.0.0 Retourne une ou plusieurs bandes d'un raster existant en tant que nouveau raster. Utile pour construire de nouveaux rasters à partir de rasters existants.
- ST_BandIsNoData - Disponibilité : 2.0.0 Retourne true si la bande ne contient que des valeurs nodata.
- ST_Clip - Disponibilité : 2.0.0 Returns the raster clipped by the input geometry. If band number is not specified, all bands are processed. If crop is not specified or TRUE, the output raster is cropped. If touched is set to TRUE, then touched pixels are included, otherwise only if the center of the pixel is in the geometry it is included.
- ST_CollectionHomogenize - Disponibilité : 2.0.0 Renvoie la représentation la plus simple d'une collection de géométries.
- ST_ConcaveHull - Disponibilité : 2.0.0 Calcule une géométrie éventuellement concave qui contient tous les sommets de la géométrie d'entrée
- ST_Count - Disponibilité : 2.0.0 Renvoie le nombre de pixels dans une bande donnée d'un raster ou d'une couverture raster. Si aucune bande n'est spécifiée, la valeur par défaut est la bande 1. Si exclude_nodata_value est true, seuls les pixels dont la valeur est différente de la valeur nodata seront comptés.
- ST_CreateTopoGeo - Disponibilité : 2.0 Ajoute une collection de géométries à une topologie vide donnée et renvoie un message détaillant le succès.
- ST_Distinct4ma - Disponibilité : 2.0.0 Fonction de traitement des données raster qui calcule le nombre de valeurs de pixels uniques dans un voisinage.

- ST_FlipCoordinates - Disponibilité : 2.0.0 Renvoie une version d'une géométrie dont les axes X et Y sont inversés.
- ST_GDALDrivers - Disponibilité : 2.0.0 - nécessite GDAL >= 1.6.0. Retourne la liste des formats raster supportés par PostGIS via GDAL. Seuls les formats avec can_write=True peuvent être utilisés par ST_AsGDALRaster
- ST_GeomFromGeoJSON - Disponibilité : 2.0.0 nécessite JSON-C >= 0.9 Prend en entrée une géométrie au format geojson et renvoie un objet Postgis de type geometry
- ST_GetFaceEdges - Disponibilité : 2.0 Renvoie un ensemble d'arêtes ordonnées qui délimitent aface.
- ST_HasNoBand - Disponibilité : 2.0.0 Retourne true si le raster n'a pas la bande spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_HillShade - Disponibilité : 2.0.0 Retourne l'hypothétique éclairage d'une bande raster d'élévation en utilisant les valeurs d'azimut, d'altitude, de luminosité et d'échelle fournies.
- ST_Histogram - Disponibilité : 2.0.0 Retourne un ensemble d'enregistrements résumant une distribution de données raster ou de couverture raster, dans des classes distinctes. Le nombre de classes est calculé automatiquement s'il n'est pas spécifié.
- ST_InterpolatePoint - Disponibilité : 2.0.0 Renvoie la mesure interpolée d'une géométrie la plus proche d'un point.
- ST_IsEmpty - Disponibilité : 2.0.0 Retourne true si le raster est vide (largeur = 0 et hauteur = 0). Sinon, retourne false.
- ST_IsValidDetail - Disponibilité : 2.0.0 Renvoie une ligne valid_detail indiquant si une géométrie est valide ou sinon une raison et une localisation.
- ST_IsValidReason - Disponibilité : la version 2.0 prend des flags. Renvoie un texte indiquant si une géométrie est valide, ou la raison de son invalidité.
- ST_MakeLine - Disponibilité : 2.0.0 - La prise en charge des éléments d'entrée LineString a été introduite Crée une LineString à partir de géométries Point, MultiPoint ou LineString.
- ST_MakeValid - Disponibilité : 2.0.0 Tente de rendre valide une géométrie invalide sans perdre de sommets.
- ST_MapAlgebraExpr - Disponibilité : 2.0.0 Version avec raster à 1 bande : Crée un nouveau raster à une bande formé par l'application d'une opération algébrique PostgreSQL valide sur la bande d'entrée du raster et du type de pixel fourni. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_MapAlgebraExpr - Disponibilité : 2.0.0 Version avec 2 bandes : Crée un nouveau raster à une bande formé en appliquant une opération algébrique PostgreSQL valide sur les deux bandes du raster d'entrée et du type de pixel fourni. La bande 1 de chaque raster est utilisée si aucun numéro de bande n'est spécifié. Le raster résultant sera aligné (échelle, obliquité et coins) sur la grille définie par le premier raster et aura son étendue définie par le paramètre "extenttype". Les valeurs de "extenttype" peuvent être : INTERSECTION, UNION, FIRST, SECOND.
- ST_MapAlgebraFct - Disponibilité : 2.0.0 Version à 1 bande - Crée un nouveau raster à une bande formé par l'application d'une fonction PostgreSQL valide sur la bande d'entrée du raster et sur le type de pixel fourni. La bande 1 est utilisée si aucune bande n'est spécifiée.
- ST_MapAlgebraFct - Disponibilité : 2.0.0 Version à 2 bandes - Crée un nouveau raster à une bande formé par l'application d'une fonction PostgreSQL valide sur les 2 bandes d'entrée du raster et sur le type de pixel fourni. La bande 1 est utilisée si aucune bande n'est spécifiée. Le type d'étendue INTERSECTION est utilisé si non spécifié.
- ST_MapAlgebraFctNgb - Disponibilité : 2.0.0 Version à 1 bande : Algèbre cartographique Plus proche voisin en utilisant une fonction PostgreSQL définie par l'utilisateur. Retourne un raster dont les valeurs sont le résultat d'une fonction utilisateur PLPGSQL prenant un voisinage des valeurs de la bande raster d'entrée.
- ST_Max4ma - Disponibilité : 2.0.0 Fonction de traitement des données raster qui calcule la valeur maximale d'un pixel dans un voisinage.
- ST_Mean4ma - Disponibilité : 2.0.0 Fonction de traitement des données raster qui calcule la valeur moyenne d'un pixel dans un voisinage.
- ST_Min4ma - Disponibilité : 2.0.0 Fonction de traitement des données raster qui calcule la valeur minimale d'un pixel dans un voisinage.

- ST_ModEdgeHeal - Disponibilité : 2.0 Répare deux arêtes en supprimant le nœud qui les relie, en modifiant la première arête et en supprimant la seconde. Renvoie l'identifiant du nœud supprimé.
- ST_MoveIsoNode - Disponibilité : 2.0.0 Déplace un nœud isolé dans une topologie d'un point à un autre. Si la nouvelle géométrie apoint existe en tant que noeud, une erreur est générée. Retourne la description du déplacement.
- ST_NewEdgeHeal - Disponibilité : 2.0 Répare deux arêtes en supprimant le nœud qui les relie, en supprimant les deux arêtes et en les remplaçant par une arête dont la direction est la même que la première arête fournie.
- ST_Node - Disponibilité : 2.0.0 Nœuds d'une collection de lignes.
- ST_NumPatches - Disponibilité : 2.0.0 Renvoie le nombre de faces d'une surface polyédrique. Retourne null pour les géométries non polyédriques.
- ST_OffsetCurve - Disponibilité : 2.0 Renvoie une ligne décalée par rapport à une distance et un côté donnés à partir d'une ligne en entrée.
- ST_PatchN - Disponibilité : 2.0.0 Renvoie la Nième géométrie (face) d'une PolyhedralSurface.
- ST_Perimeter - Disponibilité 2.0.0 : La prise en charge du type geography a été introduite Renvoie la longueur de la limite d'une géométrie polygonale ou d'une géographie.
- ST_PixelAsPolygon - Disponibilité : 2.0.0 Retourne la géométrie polygonale qui délimite le pixel pour une ligne et colonne spécifiées.
- ST_PixelAsPolygons - Disponibilité : 2.0.0 Retourne la géométrie polygonale qui délimite chaque pixel de la bande raster, avec la valeur et les coordonnées raster X et Y de chaque pixel.
- ST_Project - Disponibilité : 2.0.0 Renvoie un point projeté à partir d'un point de départ en fonction d'une distance et d'un azimut.
- ST_Quantile - Disponibilité : 2.0.0 Calcule les quantiles d'un raster ou d'une couverture raster, dans le contexte de l'échantillon ou de la population. Ainsi, une valeur peut être examinée pour se situer au percentile de 25%, 50% ou 75% du raster.
- ST_Range4ma - Disponibilité : 2.0.0 Fonction de traitement des données raster qui calcule la plage de valeurs des pixels dans un voisinage.
- ST_Reclass - Disponibilité : 2.0.0 Crée un nouveau raster composé de types de bandes reclassifiés par rapport à l'original. La bande n est la bande à modifier. Si nband n'est pas spécifié, la bande 1 est utilisée. Toutes les autres bandes sont retournées inchangées. Cas d'utilisation : convertir une bande 16BUI en 8BUI et ainsi de suite pour un rendu plus simple en tant que formats visualisables.
- ST_RelateMatch - Disponibilité : 2.0.0 Teste si une matrice d'intersection DE-9IM correspond à un modèle de matrice d'intersection
- ST_RemEdgeModFace - Disponibilité : 2.0 Supprime une arête et, si l'arête sépare deux faces, supprime une face et modifie l'autre face pour couvrir l'espace des deux.
- ST_RemEdgeNewFace - Disponibilité : 2.0 Enlève une arête et, si l'arête enlevée séparait deux faces, supprime les faces originales et les remplace par une nouvelle face.
- ST_Resample - Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+ Rééchantillonne un raster, en utilisant l'algorithme spécifié, les nouvelles dimensions, un coin arbitraire de la grille et un ensemble de paramètres de géo-référencement définis ou empruntés à un autre raster.
- ST_Rescale - Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+ Rééchantillonne un raster en ajustant juste son échelle (ou la taille des pixels). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline), Lanczos, Max ou Min. La valeur par défaut est NearestNeighbor.
- ST_Reskew - Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+ Rééchantillonne un raster en ajustant simplement son obliquité (skew, ou paramètre de rotation). Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.

- ST_SameAlignment - Disponibilité : 2.0.0 Retourne true si les rasters ont les mêmes skew, scale, spatial ref, et offset (les pixels peuvent être placés sur la même grille sans être coupés) et false si ce n'est pas le cas, avec une NOTICE détaillant le problème.
- ST_SetBandIsNoData - Disponibilité : 2.0.0 Définit la valeur du flag isnodata de la bande à TRUE.
- ST_SharedPaths - Disponibilité : 2.0.0 Renvoie une collection contenant les chemins partagés par les deux lignes/multilignes en entrée.
- ST_Slope - Disponibilité : 2.0.0 Retourne la pente (par défaut, en degrés) d'une bande raster d'élévation. Utile pour l'analyse de terrain.
- ST_Snap - Disponibilité : 2.0.0 Accrocher les segments et les sommets de la géométrie d'entrée aux sommets d'une géométrie de référence.
- ST_SnapToGrid - Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+ Rééchantillonne un raster en l'accrochant sur une grille. Les nouvelles valeurs des pixels sont calculées en utilisant l'algorithme de rééchantillonnage NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.
- ST_Split - Disponibilité : 2.0.0 nécessite GEOS Renvoie une collection de géométries créées en divisant une géométrie par une autre géométrie.
- ST_StdDev4ma - Disponibilité : 2.0.0 Fonction de traitement des données raster qui calcule l'écart type des valeurs des pixels dans un voisinage.
- ST_Sum4ma - Disponibilité : 2.0.0 Fonction de traitement des données raster qui calcule la somme de toutes les valeurs de pixels dans un voisinage.
- ST_SummaryStats - Disponibilité : 2.0.0 Retourne des résumés statistiques (count, sum, mean, stddev, min, max) pour une bande raster ou une couverture raster spécifiée. Si aucune bande n'est spécifiée, la bande 1 est utilisée.
- ST_Transform - Disponibilité : 2.0.0 Nécessite GDAL 1.6.1+ Reprojette un raster depuis un système de référence spatial vers un autre, en utilisant l'algorithme de rééchantillonnage spécifié. Les algorithmes possibles sont NearestNeighbor (plus proche voisin), Bilinear (Bilinéaire), Cubic (Cubique), CubicSpline (Cubique Spline) ou Lanczos. La valeur par défaut est NearestNeighbor.
- ST_UnaryUnion - Disponibilité : 2.0.0 Calcule l'union des composantes d'une seule géométrie.
- ST_Union - Disponibilité : 2.0.0 Retourne l'union d'un ensemble de tuiles raster, en un seul raster composé de 1 ou plusieurs bandes.
- ST_ValueCount - Disponibilité : 2.0.0 Retourne un ensemble d'enregistrements contenant une valeur de pixels et le nombre de pixels ayant cette valeur dans la bande du raster spécifié (ou de la couverture raster). Si aucune bande n'est spécifiée, la bande 1 est utilisée. Par défaut, les pixels de valeur nodata ne sont pas comptés, et toutes les autres valeurs sont retournées, avec leur valeur arrondies à l'entier le plus proche.
- TopoElementArray_Agg - Disponibilité : 2.0.0 Renvoie un topoelementarray pour un ensemble de tableaux de type, element_id (topoelements).
- TopoGeo_AddLineString - Disponibilité : 2.0.0 Adds a linestring to an existing topology using a tolerance and possibly splitting existing edges/faces.
- TopoGeo_AddPoint - Disponibilité : 2.0.0 Ajoute un point à une topologie existante en utilisant une tolérance et en divisant éventuellement une arête existante.
- TopoGeo_AddPolygon - Disponibilité : 2.0.0 Ajoute un polygone à une topologie existante en utilisant une tolérance et en divisant éventuellement les arêtes/faces existantes. Renvoie les identifiants des faces.
- TopologySummary - Disponibilité : 2.0.0 Prend un nom de topologie et fournit des totaux récapitulatifs des types d'objets dans la topologie.
- Topology_Load_Tiger - Disponibilité : 2.0.0 Charge une région définie de données tiger dans une topologie PostGIS et transforme les données tiger en référence spatiale de la topologie et en s'adaptant à la tolérance de précision de la topologie.

- toTopoGeom - Disponibilité : 2.0 Convertit une géométrie simple en une géométrie topographique.
- ~ - Disponibilité : 2.0.0 Retourne TRUE si la boîte englobante de A contient celle de B. Utilise des boîtes englobantes en double précision.
- ~= - Disponibilité : 2.0.0 Renvoie TRUE si la boîte de délimitation de A est la même que celle de B.

Fonctions améliorées dans PostGIS 2.0

- && - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Renvoi VRAI si la boite englobante 2D de A intersecte la boite englobante 2D de B.
- AddGeometryColumn - Amélioration : 2.0.0 introduction du paramètre use_typmod. Le comportement par défaut est de créer une colonne géométrique avec modificateur de type au lieu de contraintes sur la colonne. Ajoute une colonne de géométrie à une table existante.
- Box2D - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie une BOX2D représentant l'étendue 2D d'une géométrie.
- Box3D - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie une BOX3D représentant l'étendue 3D d'une géométrie.
- CreateTopology - Amélioration : 2.0 ajout de la signature acceptant hasZ Crée un nouveau schéma topologique et l'enregistre dans la table topology.topology.
- Geocode - Amélioration : 2.0.0 pour prendre en charge les données structurées Tiger 2010 et révision de certaines logiques pour améliorer la vitesse et la précision du géocodage, et pour décaler le point de la ligne centrale vers le côté de la rue où se trouve l'adresse. Le nouveau paramètre max_results permet de spécifier le nombre de meilleurs résultats ou de renvoyer uniquement le meilleur résultat. Prend une adresse sous forme de chaîne de caractères (ou autre adresse normalisée) et produit un ensemble de lieux possibles comprenant une géométrie de point en NAD 83 long lat, une adresse normalisée pour chacun d'entre eux et l'évaluation. Plus la note est basse, plus la correspondance est probable. Les résultats sont triés par ordre décroissant. Il est possible d'indiquer en option le nombre maximum de résultats (10 par défaut) et la restriction de la région (NULL par défaut)
- GeometryType - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie le type d'une géométrie sous forme de texte.
- Populate_Geometry_Columns - Amélioration : 2.0.0 L'argument optionnel use_typmod a été introduit pour controler si les colonnes sont créés avec des modificateurs de type ou des contraintes de vérification. Assure que les colonnes géométriques sont définies avec des modificateurs de type ou qu'elles sont soumises à des contraintes spatiales appropriées.
- ST_3DExtent - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.
- ST_Affine - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Appliquer une transformation affine 3D à une géométrie.
- ST_Area - Amélioration : 2.0.0 - la prise en charge des surfaces polyédriques 2D a été introduite. Renvoie l'aire d'une géométrie polygonale.
- ST_AsBinary - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsBinary - Amélioration : 2.0.0 le support pour des dimensions de coordonnées plus élevées a été introduit. Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsBinary - Amélioration : 2.0.0 le support pour spécifier endian avec geography a été introduit. Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsEWKB - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie la représentation Extended Well-Known Binary (EWKB) de la géométrie avec les métadonnées SRID.

- ST_AsEWKT - Amélioration : la version 2.0.0 a introduit la prise en charge de la geography, des surfaces polyédriques, des triangles et des TIN. Renvoie la représentation Well-Known Text (WKT) de la géométrie avec les métadonnées SRID.
- ST_AsGML - Amélioration : la prise en charge du préfixe 2.0.0 a été introduite. L'option 4 pour GML3 a été introduite pour permettre l'utilisation de LineString au lieu de Curve tag pour les lignes. La prise en charge GML3 des surfaces polyédriques et des TINS a été introduite. L'option 32 a été introduite pour produire la boîte. Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsKML - Amélioré : 2.0.0 - Ajout d'un préfixe namespace, utilisation d'arguments par défaut et d'arguments nommés Renvoyer la géométrie sous forme d'élément KML.
- ST_Azimuth - Amélioration : la prise en charge du type geography a été introduite dans la version 2.0.0. Renvoie l'azimut basé sur le nord d'une ligne entre deux points.
- ST_Dimension - Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques. Ne renvoie plus une exception si une GEOMETRY EMPTY est passée. Renvoie la dimension topologique d'une géométrie.
- ST_Dump - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie un ensemble de lignes geometry_dump pour les composants d'une géométrie.
- ST_DumpPoints - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie un ensemble de lignes geometry_dump pour les coordonnées dans une géométrie.
- ST_Expand - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie une boîte de délimitation développée à partir d'une autre boîte de délimitation ou d'une géométrie.
- ST_Extent - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Fonction agrégée qui renvoie la boîte de délimitation des géométries.
- ST_Force2D - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Forcer les géométries à passer en "mode bidimensionnel".
- ST_Force3D - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Force les géométries en mode XYZ. Il s'agit d'un alias de ST_Force3DZ.
- ST_Force3DZ - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Forcer les géométries en mode XYZ.
- ST_ForceCollection - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Convertir la géométrie en une GEOMETRYCOLLECTION.
- ST_ForceRHR - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Force l'orientation des sommets d'un polygone à suivre la règle de la main droite.
- ST_GMLToSQL - Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques. Retourne un objet de type ST_Geometry à partir de sa représentation GML. Alias pour ST_GeomFromGML
- ST_GMLToSQL - Amélioration : 2.0.0 paramètre optionnel de srid par défaut ajouté. Retourne un objet de type ST_Geometry à partir de sa représentation GML. Alias pour ST_GeomFromGML
- ST_GeomFromEWKB - Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques. Retourne un objet ST_Geometry à partir de sa représentation binaire étendue (Extended Well-Known Binary representation, EWKB).
- ST_GeomFromEWKT - Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques. Retourne un objet ST_Geometry à partir de sa représentation textuelle étendue (Extended Well-Known Text representation, EWKT).
- ST_GeomFromGML - Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques. Prend en paramètre une représentation GML d'une géométrie et renvoie un objet PostGIS de type geometry
- ST_GeomFromGML - Amélioration : 2.0.0 paramètre optionnel de srid par défaut ajouté. Prend en paramètre une représentation GML d'une géométrie et renvoie un objet PostGIS de type geometry
- ST_GeometryN - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie un élément d'une collection de géométries.

- ST_GeometryType - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Renvoie le type SQL-MM d'une géométrie sous forme de texte.
- ST_IsClosed - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Teste si les points de départ et d'arrivée d'une LineString coïncident. Pour une PolyhedralSurface, teste si elle est fermée (volumétrique).
- ST_MakeEnvelope - Amélioré : 2.0 : La possibilité de spécifier une enveloppe sans spécifier un SRID a été introduite. Crée un polygone rectangulaire à partir des coordonnées minimales et maximales.
- ST_MakeValid - Amélioré : 2.0.1, améliorations de la vitesse Tente de rendre valide une géométrie invalide sans perdre de sommets.
- ST_NPoints - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Retourne le nombre de points (vertex) d'un objet géométrique.
- ST_NumGeometries - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Renvoie le nombre d'éléments dans une collection de géométrie.
- ST_Relate - Amélioration : 2.0.0 - ajout de la prise en charge de la spécification de boundary node rule. Teste si deux géométries ont une relation topologique correspondant à un modèle de matrice d'intersection, ou calcule leur matrice d'intersection
- ST_Rotate - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Fait pivoter une géométrie autour d'un point d'origine.
- ST_Rotate - Amélioration : 2.0.0 des paramètres supplémentaires ont été ajoutés pour spécifier l'origine de la rotation. Fait pivoter une géométrie autour d'un point d'origine.
- ST_RotateX - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Fait pivoter une géométrie autour de l'axe X.
- ST_RotateY - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Fait pivoter une géométrie autour de l'axe Y.
- ST_RotateZ - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Fait pivoter une géométrie autour de l'axe Z.
- ST_Scale - Amélioration : 2.0.0 introduction du support TIN, Triangles et surfaces polyédriques. Met à l'échelle une géométrie en fonction de facteurs donnés.
- ST_ShiftLongitude - Amélioration : 2.0.0 introduction du support TIN et surfaces polyédriques. Décale les coordonnées de longitude d'une géométrie entre -180..180 et 0..360.
- ST_Summary - Amélioré : la version 2.0.0 a ajouté la prise en charge du type geography Renvoie un résumé textuel du contenu d'une géométrie.
- ST_Transform - Amélioration : 2.0.0 introduction du support des surfaces polyédriques. Renvoie une nouvelle géométrie dont les coordonnées ont été transformées dans un système de référence spatial différent.
- ST_Value - Amélioration : 2.0.0 ajout du paramètre optionnel exclude_nodata_value. Retourne la valeur d'une bande raster spécifiée au pixel donné par columnx, rowy, ou à un point géométrique spécifié. Le numéro de bande démarre à 1, et la bande 1 est utilisée si non spécifié. Si exclude_nodata_value vaut false, tous les pixels y compris ceux ayant la valeur nodata sont considérés comme intersectés et leur valeur sera retournée. Si exclude_nodata_value n'est pas spécifié, la valeur est lue depuis les méta-données du raster.
- ValidateTopology - Amélioration : 2.0.0 détection plus efficace des croisements d'arêtes et correction des faux positifs qui existaient dans les versions précédentes. Renvoie un ensemble d'objets validatetopology_returntype détaillant les problèmes liés à la topologie.

Fonctions modifiées dans PostGIS 2.0

- AddGeometryColumn - Changement : 2.0.0 Cette fonction ne met plus à jour geometry_columns maintenant que geometry_columns est une vue basée sur le catalogue système. Par défaut, elle ne créée plus de contraintes mais utilise le modificateur de type de PostgreSQL. Ainsi, par exemple, créer une colonne de type POINT WGS84 est désormais équivalent à : ALTER TABLE some_table ADD COLUMN geom geometry(Point,4326) ; Ajoute une colonne de géométrie à une table existante.
- AddGeometryColumn - Changement : 2.0.0 Si l'ancien mécanisme basé sur les contraintes est nécessaire, utiliser le paramètre use_typmod avec la valeur false. Ajoute une colonne de géométrie à une table existante.
- AddGeometryColumn - Changement : 2.0.0 Les vues ne peuvent plus être enregistrées dans geometry_columns. Cependant, les vues construites à partir de tables contenant des géométries définies avec le modificateur de type et n'utilisant pas de fonctions d'encapsulation seront enregistrées dans la vue geometry_columns car elles héritent du mécanisme des tables dont elles sont issues. Les vues utilisant des fonctions renvoyant d'autres géométries doivent être transtypées vers des géométries avec modificateur de type pour pouvoir être correctement référencées dans la vue geometry_columns. Cf. . Ajoute une colonne de géométrie à une table existante.
- Box3D - Changement : 2.0.0 Dans les versions antérieures à la version 2.0, utilisait le type box2d au lieu de box3d. Comme box2d est un type obsolète, il a été remplacé par box3d. Retourne la représentation 3d de la boîte englobante du raster.
- DropGeometryColumn - Changement : 2.0.0 Fonction assurant la rétro compatibilité. Maintenant que geometry_columns est une vue basée sur les catalogues du système, la colonne géométrique peut être supprimée d'une table comme tout autre colonne en utilisant ALTER TABLE Supprime une colonne géométrique d'une table spatiale.
- DropGeometryTable - Changement : 2.0.0 Function assurant la rétro compatibilité. Maintenant que geometry_columns est une vue basée sur les catalogues du système, une table spatiale peut etre supprimée comme tout autre table en utilisant ALTER TABLE Supprime une table et toutes ces références dans geometry_columns.
- Populate_Geometry_Columns - Changement : 2.0.0 Par défaut, utilise les modificateurs de type au lieu de contraintes de vérification pour contraindre les types géométriques. Le comportement basé sur les contraintes peut être activé en mettant le nouveau paramètre use_typmod à false. Assure que les colonnes géométriques sont définies avec des modificateurs de type ou qu'elles sont soumises à des contraintes spatiales appropriées.
- ST_3DExtent - Modifié : 2.0.0 Dans les versions précédentes, cette fonction était appelée ST_Extent3D Fonction d'agrégation qui renvoie la boîte de délimitation 3D des géométries.
- ST_3DLength - Modifié : 2.0.0 Dans les versions précédentes, cette fonction était appelée ST_Length3D Renvoie la longueur 3D d'une géométrie linéaire.
- ST_3DMakeBox - Modifié : 2.0.0 Dans les versions précédentes, cette fonction s'appelait ST_MakeBox3D Crée un BOX3D défini par deux géométries de points 3D.
- ST_3DPerimeter - Modifié : 2.0.0 Dans les versions antérieures, il s'appelait ST_Perimeter3D Renvoie le périmètre 3D d'une géométrie polygonale.
- ST_AsBinary - Modifié : 2.0.0 Les entrées de cette fonction ne peuvent pas être inconnues, elles doivent être des géométries. Des constructions telles que ST_AsBinary('POINT(1 2)') ne sont plus valides et vous obtiendrez une erreur de type n st_asbinary(unkno is not unique. Un code comme celui-là doit être changé en ST_AsBinary('POINT(1 2)'::geometry);. Si cela n'est pas possible, alors installez legacy.sql. Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsGML - Modifié : 2.0.0 utiliser les args nommés par défaut Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsGeoJSON - Modifié : 2.0.0 supporte les args par défaut et les args nommés. Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_AsSVG - Modifié : 2.0.0 pour utiliser les args par défaut et supporter les args nommés Renvoie les données de chemin SVG pour une géométrie.
- ST_EndPoint - Modifié : 2.0.0 ne fonctionne plus avec les MultiLineStrings à géométrie unique. Dans les anciennes versions de PostGIS, une MultiLineString à géométrie unique fonctionnait avec cette fonction et renvoyait le point final. Dans la version 2.0.0, elle renvoie NULL comme toute autre MultiLineString. L'ancien comportement était une fonctionnalité non documentée, mais les personnes qui supposaient que leurs données étaient stockées en tant que LINESTRING peuvent voir ces dernières retourner NULL dans la version 2.0.0. Renvoie le dernier point d'une LineString ou CircularLineString.

- ST_GDALDrivers - Changement : 2.0.6, 2.1.3 - aucun driver n'est activé par défaut, sauf si la variable GUC ou d'environnement gdal_enabled_drivers est définie. Retourne la liste des formats raster supportés par PostGIS via GDAL. Seuls les formats avec can_write=True peuvent être utilisés par ST_AsGDALRaster
- ST_GeomFromText - Changement : 2.0.0 dans les version précédentes de PostGIS ST_GeomFromText('GEOMETRYCOLL etait autorisé. C'est désormais interdit dans PostGIS 2.0.0 pour respecter la norme SQL/MM. La forme privilégiée désormais est : ST_GeomFromText('GEOMETRYCOLLECTION EMPTY') Retourne un objet ST_Geometry à partir de sa représentation textuelle Well-Known Text (WKT).
- ST_GeometryN - Changement : 2.0.0. Les versions antérieures renvoient NULL pour les géometries simples (un seul objet). Renvoie désormais la géométrie pour le cas ST_GeometryN(..,1). Renvoie un élément d'une collection de géométries.
- ST_IsEmpty - Modifié : 2.0.0 Dans les versions précédentes de PostGIS, ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY) était autorisé. Ceci est maintenant illégal dans PostGIS 2.0.0 pour mieux se conformer aux normes SQL/MM Teste si une géométrie est vide.
- ST_Length - Modifié : 2.0.0 Rupture -- dans les versions précédentes, appliquer ceci à un MULTI/POLYGONE de type geography donnait le périmètre du POLYGONE/MULTIPOLYGONE. Dans la version 2.0.0, cette fonction a été modifiée pour retourner 0 afin d'être en ligne avec le comportement de la géométrie. Veuillez utiliser ST_Perimeter si vous souhaitez obtenir le périmètre d'un polygone Renvoie la longueur 2D d'une géométrie linéaire.
- ST_LocateAlong - Modifié : 2.0.0 dans les versions précédentes, cette fonction était appelée ST_Locate_Along_Measure. Renvoie le(s) point(s) d'une géométrie qui correspond(ent) à une valeur de mesure.
- ST_LocateBetween - Modifié : 2.0.0 - dans les versions précédentes, cette fonction s'appelait ST_Locate_Between_Measures. Renvoie les parties d'une géométrie qui correspondent à un intervalle de mesure.
- ST_ModEdgeSplit - Modifié : 2.0 - Dans les versions antérieures, cette fonction était mal nommée ST_ModEdgesSplit Fractionner une arête en créant un nouveau nœud le long d'une arête existante, en modifiant l'arête d'origine et en ajoutant une nouvelle arête.
- ST_NumGeometries - Modifié : 2.0.0 Dans les versions précédentes, cette fonction renvoyait NULL si la géométrie n'était pas de type collection/MULTI. 2.0.0+ renvoie maintenant 1 pour les géométries simples, par exemple POLYGONE, LINESTRING, POINT. Renvoie le nombre d'éléments dans une collection de géométrie.
- ST_NumInteriorRings - Modifié : 2.0.0 - dans les versions antérieures, il permettait de passer un MULTIPOLYGONE, renvoyant le nombre d'anneaux intérieurs du premier POLYGONE. Renvoie le nombre d'anneaux intérieurs (trous) d'un polygone.
- ST_PointN - Modifié : la version 2.0.0 ne fonctionne plus avec les multi-lignes à géométrie unique. Dans les anciennes versions de PostGIS, une multi-ligne d'une seule ligne fonctionnait parfaitement avec cette fonction et renvoyait le point de départ. Dans la version 2.0.0, elle renvoie simplement NULL comme n'importe quelle autre multi-ligne. Renvoie le Nième point de la première LineString ou LineString circulaire d'une géométrie.
- ST_ScaleX - Changement : 2.0.0. Dans les versions WKTRaster, cette fonction était appelée ST_PixelSizeX. Renvoie la composante X de la largeur du pixel dans l'unité du système de référence spatial.
- ST_ScaleY - Changement : 2.0.0. Dans les versions WKTRaster, cette fonction était appelée ST_PixelSizeY. Renvoie la composante Y de la hauteur du pixel dans l'unité du système de référence spatial.
- ST_SetScale - Changement : 2.0.0 Dans les versions WKTRaster, cette fonction était appelée ST_SetPixelSize. Définit la résolution des pixels en X et Y en unité du système de référence spatial : nombre d'unités/pixel en largeur/hauteur.
- ST_StartPoint - Modifié : 2.0.0 ne fonctionne plus avec les MultiLineStrings à géométrie unique. Dans les anciennes versions de PostGIS, une MultiLineString à géométrie unique fonctionnait sans problème avec cette fonction et renvoyait le point de départ. Dans la version 2.0.0, elle renvoie simplement NULL comme toute autre MultiLineString. L'ancien comportement était une fonctionnalité non documentée, mais les personnes qui supposaient que leurs données étaient stockées en tant que LINESTRING peuvent voir ces données retourner NULL dans la version 2.0.0. Renvoie le premier point d'une LineString.

13.12.13 Fonctions PostGIS nouvelles ou améliorées en 1.5

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 1.5

- && - Disponibilité : 1.5.0 le support de la géographie a été introduit. Renvoi VRAI si la boite englobante 2D de A intersecte la boite englobante 2D de B.
- PostGIS_LibXML_Version - Disponibilité : 1.5 Renvoie le numéro de version de la librairie libxml2.
- ST_AddMeasure - Disponibilité : 1.5.0 Interpole les mesures le long d'une géométrie linéaire.
- ST_AsBinary - Disponibilité : 1.5.0 Le support de la géographie a été introduit. Renvoie la représentation OGC/ISO Well-Known Binary (WKB) de la géométrie/geography sans les métadonnées SRID.
- ST_AsGML - Disponibilité : 1.5.0 Le support de la géographie a été introduit. Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsGeoJSON - Disponibilité : 1.5.0 Le support de la géographie a été introduit. Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_AsText - Disponibilité : 1.5 - le support de la geography a été introduit. Renvoie la représentation Well-Known Text (WKT) de la géométrie/geography sans métadonnées SRID.
- ST_Buffer - Disponibilité : 1.5 - ST_Buffer a été amélioré pour prendre en charge différents types de terminaisons et de jointures. Ceux-ci sont utiles, par exemple, pour convertir les lignes de route en routes polygonales avec des bords plats ou carrés au lieu de bords arrondis. Un petit wrapper pour la geography a été ajouté. Calcule une géométrie couvrant tous les points situés à une distance donnée d'une géométrie.
- ST_ClosestPoint - Disponibilité : 1.5.0 Renvoie le point 2D sur g1 qui est le plus proche de g2. Il s'agit du premier point de la ligne la plus courte d'une géométrie à l'autre.
- ST_CollectionExtract - Disponibilité : 1.5.0 Pour une collection de géométries spécifiée, renvoie une multi-géométrie contenant uniquement des éléments d'un type spécifié.
- ST_Covers - Disponibilité : 1.5 - le support de la geography a été introduit. Tests si chaque point de B est situé dans A
- ST_DFullyWithin - Disponibilité : 1.5.0 Teste si une géométrie se trouve entièrement à une distance d'une autre géométrie
- ST_DWithin - Disponibilité : la prise en charge du type geography a été introduite dans la version 1.5.0 Teste si deux géométries se trouvent à une distance donnée
- ST_Distance - Disponibilité : 1.5.0 La prise en charge du type geography a été introduite dans la version 1.5. Amélioration de la vitesse pour les géométries planaires afin de mieux gérer les géométries de grande taille ou à nombreux sommets Renvoie la distance entre deux valeurs de geometry ou geography.
- ST_DistanceSphere - Disponibilité : 1.5 - la prise en charge d'autres types de géométrie que les points a été introduite. Les versions précédentes ne fonctionnaient qu'avec des points. Renvoie la distance minimale en mètres entre deux géométries lon/lat en utilisant un modèle de terre sphérique.
- ST_DistanceSpheroid - Disponibilité : 1.5 - la prise en charge d'autres types de géométrie que les points a été introduite. Les versions précédentes ne fonctionnaient qu'avec des points. Renvoie la distance minimale entre deux géométries lon/lat en utilisant un modèle de terre sphéroïdale.
- ST_DumpPoints - Disponibilité : 1.5.0 Renvoie un ensemble de lignes geometry_dump pour les coordonnées dans une géométrie.
- ST_Envelope - Disponibilité : 1.5.0 changement pour renvoyer un type double précision à la place de float4 Renvoie une géométrie représentant la boîte de délimitation d'une géométrie.
- ST_Expand - Disponibilité : 1.5.0 comportement modifié pour afficher les coordonnées en double précision au lieu des coordonnées float4. Renvoie une boîte de délimitation développée à partir d'une autre boîte de délimitation ou d'une géométrie.

- ST_GMLToSQL - Disponibilité : 1.5, nécessite libxml2 1.6+ Retourne un objet de type ST_Geometry à partir de sa représentation GML. Alias pour ST_GeomFromGML
- ST_GeomFromGML - Disponibilité : 1.5, nécessite libxml2 1.6+ Prend en paramètre une représentation GML d'une géométrie et renvoie un objet PostGIS de type geometry
- ST_GeomFromKML - Disponibilité : 1.5, nécessite libxml2 2.6+ Prend en entrée une géométrie au format KML et renvoie un objet Postgis de type geometry
- ST_HausdorffDistance - Disponibilité : 1.5.0 Renvoie la distance de Hausdorff entre deux géométries.
- ST_Intersection - Disponibilité : La version 1.5 a introduit la prise en charge du type de données geography. Calcule une géométrie représentant la partie partagée des géométries A et B.
- ST_Intersects - Disponibilité : la version 1.5 a introduit la prise en charge du type geography. Teste si deux géométries se croisent (elles ont au moins un point en commun)
- ST_Length - Disponibilité : 1.5.0 La prise en charge du type geography a été introduite dans la version 1.5. Renvoie la longueur 2D d'une géométrie linéaire.
- ST_LongestLine - Disponibilité : 1.5.0 Renvoie la ligne 2D la plus longue entre deux géométries.
- ST_MakeEnvelope - Disponibilité : 1.5 Crée un polygone rectangulaire à partir des coordonnées minimales et maximales.
- ST_MaxDistance - Disponibilité : 1.5.0 Renvoie la plus grande distance 2D entre deux géométries en unités projetées.
- ST_ShortestLine - Disponibilité : 1.5.0 Renvoie la ligne 2D la plus courte entre deux géométries
- ~= - Disponibilité : 1.5.0 comportement changé Renvoie TRUE si la boîte de délimitation de A est la même que celle de B.

13.12.14 Fonctions PostGIS nouvelles ou améliorées en 1.4

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 1.4

- Populate_Geometry_Columns - Disponibilité : 1.4.0 Assure que les colonnes géométriques sont définies avec des modificateurs de type ou qu'elles sont soumises à des contraintes spatiales appropriées.
- ST_Collect - Disponibilité : 1.4.0 - ST_Collect(geomarray) a été introduit. ST_Collect a été amélioré pour gérer plus de géométries plus rapidement. Crée une géométrie GeometryCollection ou Multi à partir d'un ensemble de géométries.
- ST_ContainsProperly - Disponibilité : 1.4.0 Tests si chaque point de B se trouve à l'intérieur de A
- ST_GeoHash - Disponibilité : 1.4.0 Retourne une représentation GeoHash de la géométrie.
- ST_IsValidReason - Disponibilité : 1.4 Renvoie un texte indiquant si une géométrie est valide, ou la raison de son invalidité.
- ST_LineCrossingDirection - Disponibilité : 1.4 Renvoie un nombre indiquant le comportement de croisement de deux LineStrings
- ST_LocateBetweenElevations - Disponibilité : 1.4.0 Renvoie les parties d'une géométrie qui se trouvent dans un intervalle d'élévation (Z).
- ST_MakeLine - Disponibilité : 1.4.0 - création de ST_MakeLine(geomarray). L'agrégat spatial ST_MakeLine amélioré pour supporter plus de points plus rapidement. Crée une LineString à partir de géométries Point, MultiPoint ou LineString.
- ST_MinimumBoundingCircle - Disponibilité : 1.4.0 Retourne le plus petit cercle polygonal qui contient une géométrie.
- ST_Union - Disponibilité : 1.4.0 - ST_Union a été amélioré. ST_Union(geomarray) a été introduit ainsi qu'une collecte d'agrégats plus rapide dans PostgreSQL. Calcule une géométrie représentant l'union des ensembles de points des géométries d'entrée.

13.12.15 Fonctions PostGIS nouvelles ou améliorées en 1.3

Les fonctions indiquées ci-dessous sont des fonctions PostGIS qui ont été ajoutées ou améliorées.

Fonctions nouvelles dans PostGIS 1.3

- ST_AsGML - Disponibilité : 1.3.2 Renvoyer la géométrie en tant qu'élément GML version 2 ou 3.
- ST_AsGeoJSON - Disponibilité : 1.3.4 Renvoyer une géométrie ou un élément au format GeoJSON.
- ST_CurveToLine - Disponibilité : 1.3.0 Convertit une géométrie contenant des courbes en une géométrie linéaire.
- ST_LineToCurve - Disponibilité : 1.3.0 Convertit une géométrie linéaire en une géométrie courbe.
- ST_SimplifyPreserveTopology - Disponibilité : 1.3.3 Renvoie une représentation simplifiée et valide d'une géométrie, en utilisant l'algorithme de Douglas-Peucker.

Chapter 14

Rapporter un problème

14.1 Rapporter un problème logiciel

Rapporter un problème est effectivement fondamental afin d'aider le développement de PostGIS. Un rapport de bug efficace permet à l'équipe des développeurs de PostGIS de reproduire le problème. Le rapport est donc pertinent s'il contient le script qui le déclenche et toutes les informations à propos de l'environnement dans lequel il a été détecté. De bonnes informations peuvent être obtenus par l'exécution de `SELECT postgis_full_version()` [pour postgis] et `SELECT version()` [pour postgresql].

Si vous n'utilisez pas la dernière version, il vaut la peine de jeter un coup d'œil à son journal des changements de la version d'abord, pour savoir si votre bug a déjà été corrigé.

L'utilisation du PostGIS bug tracker permettra de s'assurer que vos rapports ne sont pas rejetés et vous tiendra informé du processus de traitement. Avant de signaler un nouveau bug, veuillez interroger la base de données pour voir s'il s'agit d'un bug connu, et si c'est le cas, veuillez ajouter toute nouvelle information que vous possédez à son sujet.

Vous pouvez lire l'article de Simon Tatham sur Comment signaler efficacement les bugs avant de déposer un nouveau rapport.

14.2 Signaler les problèmes de documentation

La documentation doit refléter fidèlement les caractéristiques et le comportement du logiciel. Si ce n'est pas le cas, cela peut être dû à un bug du logiciel ou à une erreur ou une lacune dans la documentation.

Les problèmes de documentation peuvent également être signalés au PostGIS bug tracker.

Si votre révision est triviale, décrivez-la simplement dans une nouvelle question du bug tracker, en précisant son emplacement dans la documentation.

Si vos modifications sont plus importantes, il est préférable d'utiliser un correctif. Il s'agit d'un processus en quatre étapes sous Unix (en supposant que vous ayez déjà installé git) :

1. Clonez le dépôt git de PostGIS. Sous Unix, tapez :

   git clone https://git.osgeo.org/gitea/postgis/postgis.git

   Cela sera stocké dans le répertoire postgis

2. Faites vos changements sur la documentation avec votre éditeur de texte favori. Sur Unix, tapez (pour exemple) :

   vim doc/postgis.xml

   Remarquez que la documentation est écrite avec DocBook XML au lieu de HTML, donc si vous ne vous sentez pas familier avec cela, vous devriez suivre les exemples dans la suite de la documentation.

3. Faites un correctif contenant les différences avec une copie de la documentation principale. Sur Unix, tapez :

   git diff doc/postgis.xml > doc.patch

4. Ajoutez le correctif à un nouveau ticket dans le système de suivi de bogues.

Appendix A

Annexes

A.1 PostGIS 3.5.3

2025/05/17

This version requires PostgreSQL 12-18beta1, GEOS 3.8 or higher, and Proj 6.1+. To take advantage of all features, GEOS 3.12+ is needed. To take advantage of all SFCGAL features, SFCGAL 1.5.0+ is needed.

A.1.1 Bug Fixes

Do not complain about illegal option when calling shp2pgsql -? (Sandro Santilli, Giovanni Zezza)

5862, [topology] Prevent another topology corruption with TopoGeo_addPoint near almost collinear edges (Sandro Santilli)

5841, Change approach to interrupt handling to conform to PgSQL recommended practice (Paul Ramsey)

5855, Fix index binding in ST_DFullyWithin (Paul Ramsey)

5819, Support longer names in estimated extent (Paul Ramsey)

Fix misassignment of result in _lwt_HealEdges (Maxim Korotkov)

5876, ST_AddPoint with empty argument adds garbage (Paul Ramsey)

5874, Line substring returns wrong answer (Paul Ramsey)

5829, geometry_columns with non-standard constraints (Paul Ramsey)

5818, GT-244 Fix CG_IsSolid function (Loïc Bartoletti)

5885, Fix documentation about grid-based overlay operations (Sandro Santilli)

For SFCGAL 2.1.0+ prevent using deprecated functions (Regina Obe)

A.2 PostGIS 3.5.2

2025/01/18

This version requires PostgreSQL 12-17, GEOS 3.8 or higher, and Proj 6.1+. To take advantage of all features, GEOS 3.12+ is needed. To take advantage of all SFCGAL features, SFCGAL 1.5.0+ is needed.

A.2.1 Bug Fixes

#5677, Retain SRID during unary union (Paul Ramsey)

#5833, pg_upgrade fix for postgis_sfcgal (Regina Obe)

#5564, BRIN crash fix and support for parallel in PG17+ (Paul Ramsey, Regina Obe)

A.3 PostGIS 3.5.1

2024/12/22

This version requires PostgreSQL 12-17, GEOS 3.8 or higher, and Proj 6.1+. To take advantage of all features, GEOS 3.12+ is needed. To take advantage of all SFCGAL features, SFCGAL 1.5.0+ is needed.

A.3.1 Changements avec rupture

#5677, Retain SRID during unary union (Paul Ramsey)

#5792, [topology] Prevent topology corruption with TopoGeo_addPoint near almost collinear edges (Sandro Santilli)

#5795, [topology] Fix ST_NewEdgesSplit can cause invalid topology (Björn Harrtell)

#5794, [topology] Fix crash in TopoGeo_addPoint (Sandro Santilli)

#5785, [raster] ST_MapAlgebra segfaults when expression references a supernumerary rast argument (Dian M Fay)

#5787, Check that ST_ChangeEdgeGeom doesn't change winding of rings (Sandro Santilli)

#5791, Add legacy stubs for old transaction functions to allow pg_upgrade (Regina Obe)

#5800, PROJ compiled version reading the wrong minor and micro (Regina Obe)

#5790, Non-schema qualified calls causing issue with materialized views (Regina Obe)

#5812, Performance regression in ST_Within (Paul Ramsey)

#5815, Remove hash/merge promise from <> operator (Paul Ramsey)

#5823, Build support for Pg18 (Paul Ramsey)

A.3.2 Améliorations

#5782, Improve robustness of min distance calculation (Sandro Santilli)

[topology] Speedup topology building when closing large rings with many holes (Björn Harrtell)

#5810, Update tiger geocoder to handle TIGER 2024 data (Regina Obe)

A.3.3 Changements avec rupture

#5799, make ST_TileEnvelope clip envelopes to tile plane extent (Paul Ramsey)

A.4 PostGIS 3.5.0

2024/09/25

This version requires PostgreSQL 12-17, GEOS 3.8 or higher, and Proj 6.1+. To take advantage of all features, GEOS 3.12+ is needed. To take advantage of all SFCGAL features, SFCGAL 1.5.0+ is needed.

Un grand merci à nos équipes de traduction, en particulier :

Dapeng Wang, Zuo Chenwei from HighGo (Chinese Team)

Teramoto Ikuhiro (Japanese Team)

Vincent Bre (French Team)

A.4.1 Changements avec rupture

#5546, TopoGeometry <> TopoGeometry is now ambiguous, to get the old behaviour, assuming your TopoGeometry objects are named tg1 and tg2, use: (id(tg1) <> id(tg2) OR topology_id(tg1) <> topology_id(tg2) OR layer_id(tg1) <> layer_id(tg2) OR type(tg1) <> type(tg2)) (Sandro Santilli)

#5536, comments are not anymore included in PostGIS extensions (Sandro Santilli)

xmllint is now required to build comments (Sandro Santilli)

DocBook5 XSL is now required to build html (Sandro Santilli)

#5602, Drop support for GEOS 3.6 and 3.7 (Regina Obe)

#5571, Improve ST_GeneratePoints performance, but old seeded pseudo random points will need to be regenerated.

#5596, GH-749, Allow promoting column as an id in ST_AsGeoJson(record,..). Views and materialized views that use the ST_AsGeoJSON(record ..) will need rebuilding to upgrade to new signature (Jan Tojnar)

#5496, ST_Clip all variants replaced, will require rebuilding of materialized views that use them (funding from The National Institute for Agricultural and Food Research and Technology (INIA-CSIC)), Regina Obe

#5659, ST_DFullyWithin behaviour has changed to be ST_Contains(ST_Buffer(A, R), B) (Paul Ramsey)

Remove the WFS_locks extra package. (Paul Ramsey)

5747, GH-776, ST_Length: Return 0 for CurvePolygon (Dan Baston)

5770, support for GEOS 3.13 and RelateNG. Most functionality remains the same, but new GEOS predicate implementation has a few small changes.

Boundary Node Rule relate matrices might be different when using the "multi-valent end point" rule.

Relate matrices for situations with invalid MultiPolygons with shared boundaries might be different. Run ST_MakeValid to get valid inputs to feed to the calculation.

Zero length LineStrings are treated as if they are the equivalent Point object.

A.4.2 Deprecated signatures

GH-761, ST_StraightSkeleton = > CG_StraightSkeleton (Loïc Bartoletti)

GH-189, All SFCGAL functions now use the prefix CG_, with the old ones using ST_ being deprecated. (Loïc Bartoletti)

A.4.3 Nouvelles fonctionnalités

Improvements in the 'postgis' script:

- new command list-enabled
- new command list-all
- command upgrade upgrades all databases that need to be
- command status reports status of all databases

(Sandro Santilli)

#5742, expose version of PROJ at compile time (Sandro Santilli)

#5721, postgis_topology: Allow sharing sequences between different topologies (Lars Opsahl)

#5667, postgis_topology: TopoGeo_LoadGeometry (Sandro Santilli)

#5055, add explicit <> geometry operator to prevent non-unique error with <> and != (Paul Ramsey)

Add ST_HasZ/ST_HasM (Loïc Bartoletti)

GT-123, postgis_sfcgal: CG_YMonotonePartition, CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition and CG_OptimalC (Loïc Bartoletti)

GT-156, postgis_sfcgal: CG_Visibility (Loïc Bartoletti)

GT-157, postgis_sfcgal: Add ST_ExtrudeStraightSkeleton (Loïc Bartoletti)

#5496, postgis_raster: ST_Clip support for touched (Regina Obe)

GH-760, postgis_sfcgal: CG_Intersection, CG_3DIntersects, CG_Intersects, CG_Difference, CG_Union (and aggregate), CG_Triangula CG_Area, CG_3DDistance, CG_Distance (Loïc Bartoletti)

#5687, Don't rely on search_path to determine postgis schema Fix for PG17 security change (Regina Obe)

#5705, GH-767, ST_RemoveIrrelevantPointsForView (Sam Peters)

#5706, GH-768, ST_RemoveSmallParts (Sam Peters)

A.4.4 Améliorations

5550, Fix upgrades from 2.x in sandboxed systems (Sandro Santilli)

#3587, postgis_topology: faster load of big lines in topologies (Sandro Santilli)

#5670, postgis_topology: faster ST_CreateTopoGeo (Sandro Santilli)

#5531, documentation format upgraded to DocBook 5 (Sandro Santilli)

#5543, allow building without documentation (Sandro Santilli)

#5596, GH-749, Allow promoting column as an id in ST_AsGeoJson(record,..). (Jan Tojnar)

GH-744, Don't create docbook.css for the HTML manual, use style.css instead (Chris Mayo)

Faster implementation of point-in-poly cached index (Paul Ramsey)

Improve performance of ST_GeneratePoints (Paul Ramsey)

#5361, ST_CurveN, ST_NumCurves and consistency in accessors on curved geometry (Paul Ramsey)

GH-761, postgis_sfcgal: Add an optional parameter to CG_StraightSkeleton (was ST_StraightSkeleton) to use m as a distance in result (Hannes Janetzek, Loïc Bartoletti)