PostGIS 3.5.4dev 手册

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

Contents

1 介☒ 1
1.1 ☒目指☒委☒会 . . . 1
1.2 当前的核心☒献者 . . . 1
1.3 ☒去的核心☒献者 . . . 2
1.4 其它☒献者 . . . 2

2 PostGIS 安装 6
2.1 ☒短版本 . . . 6
2.2 从源代☒☒☒和安装 . . . 6
2.2.1 ☒取源代☒ . . . 7
2.2.2 安装要求 . . . 7
2.2.3 ☒建配置 . . . 8
2.2.4 ☒建 . . . 10
2.2.5 ☒建和部署 PostGIS ☒展 . . . 10
2.2.6 ☒☒ . . . 12
2.2.7 安装 . . . 15
2.3 安装和使用地址☒准化工具 . . . 15
2.4 安装、升☒ Tiger 地理☒☒器并加☒数据 . . . 16
2.4.1 在 PostGIS 数据☒中☒用 Tiger 地理☒☒器 . . . 16
2.4.2 将地址☒准化器☒展与 Tiger 地理☒☒器☒合使用 . . . 18
2.4.3 Tiger 数据加☒所需工具 . . . 19
2.4.4 升☒您的 Tiger 地理☒☒器安装和数据 . . . 19
2.5 常☒☒☒ . . . 20

3 PostGIS Administration 21
3.1 性能☒☒ . . . 21
3.1.1 ☒☒ . . . 21
3.1.2 ☒行 . . . 22
3.2 Configuring raster support . . . 22
3.3 Creating spatial databases . . . 23

3.3.1 Spatially enable database using EXTENSION . . . 23
3.3.2 Spatially enable database without using EXTENSION (discouraged) . . . 23
3.4 Upgrading spatial databases . . . 24
3.4.1 Soft upgrade . . . 24
3.4.1.1 Soft Upgrade 9.1+ using extensions . . . 24
3.4.1.2 Soft Upgrade Pre 9.1+ or without extensions . . . 25
3.4.2 Hard upgrade . . . 26

4 数据管理 28
4.1 空□数据模型 . . . 28
4.1.1 OGC 几何□形 . . . 28
4.1.1.1 点（Point） . . . 28
4.1.1.2 □串（LineString） . . . 29
4.1.1.3 □性□（LinearRing） . . . 29
4.1.1.4 多□形（Polygon） . . . 29
4.1.1.5 多点（MultiPoint） . . . 29
4.1.1.6 多□（MultiLineString） . . . 29
4.1.1.7 多面（MultiPolygon） . . . 29
4.1.1.8 几何集合（GeometryCollection） . . . 29
4.1.1.9 多面体曲面（PolyhedralSurface） . . . 30
4.1.1.10三角形（Triangle） . . . 30
4.1.1.11TIN . . . 30
4.1.2 SQL/MM 第 3 部分-曲□ . . . 30
4.1.2.1 □弧串（CircularString） . . . 30
4.1.2.2 复合曲□（CompoundCurve） . . . 30
4.1.2.3 曲□多□形（CurvePolygon） . . . 31
4.1.2.4 多曲□（MultiCurve） . . . 31
4.1.2.5 多曲面（MultiSurface） . . . 31
4.1.3 WKT 和 WKB . . . 31
4.2 几何数据□型 . . . 32
4.2.1 PostGIS EWKB 和 EWKT . . . 33
4.3 地理数据□型 . . . 34
4.3.1 □建地理表 . . . 34
4.3.2 使用地理表 . . . 35
4.3.3 何□使用地理数据□型 . . . 36
4.3.4 地理高□常□□□解答 . . . 37
4.4 几何有效性□□ . . . 37
4.4.1 □□几何 . . . 37
4.4.2 有效的几何形状 . . . 39

4.4.3 有效性管理 . . . 41
4.5 空间参考系统 . . . 42
4.5.1 SPATIAL_REF_SYS 表 . . . 42
4.5.2 用户自定义空间参考系统 . . . 43
4.6 空间表 . . . 44
4.6.1 创建空间表 . . . 44
4.6.2 GEOMETRY_COLUMNS 视图 . . . 45
4.6.3 手动注册几何列 . . . 45
4.7 加载空间数据 . . . 47
4.7.1 使用 SQL 加载数据 . . . 47
4.7.2 使用 Shapefile 加载器 . . . 48
4.8 提取空间数据 . . . 49
4.8.1 使用 SQL 提取数据 . . . 49
4.8.2 使用 Shapefile 文件转换程序 . . . 50
4.9 空间索引 . . . 51
4.9.1 GiST 索引 . . . 51
4.9.2 BRIN 索引 . . . 52
4.9.3 SP-GiST 索引 . . . 53
4.9.4 关于索引的使用 . . . 54

5 空间关系 55
5.1 空间关系的确定 . . . 55
5.1.1 维度扩展九交模型（Dimensionally Extended 9-Intersection Model） . . . 55
5.1.2 命名空间关系 . . . 57
5.1.3 一般空间关系 . . . 57
5.2 使用空间索引 . . . 59
5.3 空间 SQL 示例 . . . 59

6 性能优化技巧 62
6.1 大几何形状的小表 . . . 62
6.1.1 问题描述 . . . 62
6.1.2 解决方法 . . . 62
6.2 几何索引聚类 . . . 63
6.3 避免维度问题 . . . 63

7 PostGIS 参考手册 **64**
7.1 PostGIS Geometry/Geography/Box 数据⊠型 . . . 64
7.1.1 box2d . . . 64
7.1.2 box3d . . . 65
7.1.3 geometry . . . 65
7.1.4 geometry_dump . . . 66
7.1.5 geography . . . 66
7.2 表管理功能 . . . 67
7.2.1 AddGeometryColumn . . . 67
7.2.2 DropGeometryColumn . . . 69
7.2.3 DropGeometryTable . . . 69
7.2.4 Find_SRID . . . 70
7.2.5 Populate_Geometry_Columns . . . 71
7.2.6 UpdateGeometrySRID . . . 72
7.3 几何⊠造函数 . . . 73
7.3.1 ST_Collect . . . 73
7.3.2 ST_LineFromMultiPoint . . . 75
7.3.3 ST_MakeEnvelope . . . 76
7.3.4 ST_MakeLine . . . 76
7.3.5 ST_MakePoint . . . 78
7.3.6 ST_MakePointM . . . 79
7.3.7 ST_MakePolygon . . . 80
7.3.8 ST_Point . . . 82
7.3.9 ST_PointZ . . . 83
7.3.10 ST_PointM . . . 84
7.3.11 ST_PointZM . . . 84
7.3.12 ST_Polygon . . . 85
7.3.13 ST_TileEnvelope . . . 86
7.3.14 ST_HexagonGrid . . . 87
7.3.15 ST_Hexagon . . . 89
7.3.16 ST_SquareGrid . . . 90
7.3.17 ST_Square . . . 91
7.3.18 ST_Letters . . . 92
7.4 几何⊠⊠器 . . . 93
7.4.1 GeometryType . . . 93
7.4.2 ST_Boundary . . . 94
7.4.3 ST_BoundingDiagonal . . . 96
7.4.4 ST_CoordDim . . . 97
7.4.5 ST_Dimension . . . 98

7.4.6 ST_Dump . . . 98
7.4.7 ST_DumpPoints . . . 100
7.4.8 ST_DumpSegments . . . 104
7.4.9 ST_DumpRings . . . 106
7.4.10 ST_EndPoint . . . 107
7.4.11 ST_Envelope . . . 108
7.4.12 ST_ExteriorRing . . . 110
7.4.13 ST_GeometryN . . . 111
7.4.14 ST_GeometryType . . . 113
7.4.15 ST_HasArc . . . 114
7.4.16 ST_InteriorRingN . . . 115
7.4.17 ST_NumCurves . . . 116
7.4.18 ST_CurveN . . . 116
7.4.19 ST_IsClosed . . . 117
7.4.20 ST_IsCollection . . . 119
7.4.21 ST_IsEmpty . . . 120
7.4.22 ST_IsPolygonCCW . . . 121
7.4.23 ST_IsPolygonCW . . . 122
7.4.24 ST_IsRing . . . 123
7.4.25 ST_IsSimple . . . 123
7.4.26 ST_M . . . 124
7.4.27 ST_MemSize . . . 125
7.4.28 ST_NDims . . . 126
7.4.29 ST_NPoints . . . 127
7.4.30 ST_NRings . . . 128
7.4.31 ST_NumGeometries . . . 128
7.4.32 ST_NumInteriorRings . . . 129
7.4.33 ST_NumInteriorRing . . . 130
7.4.34 ST_NumPatches . . . 130
7.4.35 ST_NumPoints . . . 131
7.4.36 ST_PatchN . . . 131
7.4.37 ST_PointN . . . 132
7.4.38 ST_Points . . . 134
7.4.39 ST_StartPoint . . . 135
7.4.40 ST_Summary . . . 136
7.4.41 ST_X . . . 137
7.4.42 ST_Y . . . 138
7.4.43 ST_Z . . . 138
7.4.44 ST_Zmflag . . . 139

7.4.45ST_HasZ . . . 140
7.4.46ST_HasM . . . 141
7.5 几何⊠⊠器 . . . 141
7.5.1 ST_AddPoint . . . 141
7.5.2 ST_CollectionExtract . . . 142
7.5.3 ST_CollectionHomogenize . . . 144
7.5.4 ST_CurveToLine . . . 145
7.5.5 ST_Scroll . . . 148
7.5.6 ST_FlipCoordinates . . . 148
7.5.7 ST_Force2D . . . 149
7.5.8 ST_Force3D . . . 150
7.5.9 ST_Force3DZ . . . 151
7.5.10ST_Force3DM . . . 152
7.5.11ST_Force4D . . . 152
7.5.12ST_ForceCollection . . . 153
7.5.13ST_ForceCurve . . . 154
7.5.14ST_ForcePolygonCCW . . . 155
7.5.15ST_ForcePolygonCW . . . 156
7.5.16ST_ForceSFS . . . 156
7.5.17ST_ForceRHR . . . 156
7.5.18ST_LineExtend . . . 157
7.5.19ST_LineToCurve . . . 158
7.5.20ST_Multi . . . 159
7.5.21ST_Normalize . . . 160
7.5.22ST_Project . . . 161
7.5.23ST_QuantizeCoordinates . . . 161
7.5.24ST_RemovePoint . . . 163
7.5.25ST_RemoveRepeatedPoints . . . 164
7.5.26ST_RemoveIrrelevantPointsForView . . . 165
7.5.27ST_RemoveSmallParts . . . 167
7.5.28ST_Reverse . . . 169
7.5.29ST_Segmentize . . . 170
7.5.30ST_SetPoint . . . 171
7.5.31ST_ShiftLongitude . . . 172
7.5.32ST_WrapX . . . 173
7.5.33ST_SnapToGrid . . . 174
7.5.34ST_Snap . . . 176
7.5.35ST_SwapOrdinates . . . 178
7.6 几何有效性⊠⊠ . . . 179

7.6.1 ST_IsValid . . . 179
7.6.2 ST_IsValidDetail . . . 180
7.6.3 ST_IsValidReason . . . 182
7.6.4 ST_MakeValid . . . 183
7.7 空□参考系□功能 . . . 188
7.7.1 ST_InverseTransformPipeline . . . 188
7.7.2 ST_SetSRID . . . 189
7.7.3 ST_SRID . . . 190
7.7.4 ST_Transform . . . 191
7.7.5 ST_TransformPipeline . . . 193
7.7.6 postgis_srs_codes . . . 195
7.7.7 postgis_srs . . . 195
7.7.8 postgis_srs_all . . . 196
7.7.9 postgis_srs_search . . . 197
7.8 几何□入 . . . 198
7.8.1 已知文本 (WKT) . . . 198
7.8.1.1 ST_BdPolyFromText . . . 198
7.8.1.2 ST_BdMPolyFromText . . . 198
7.8.1.3 ST_GeogFromText . . . 199
7.8.1.4 ST_GeographyFromText . . . 199
7.8.1.5 ST_GeomCollFromText . . . 200
7.8.1.6 ST_GeomFromEWKT . . . 200
7.8.1.7 ST_GeomFromMARC21 . . . 202
7.8.1.8 ST_GeometryFromText . . . 204
7.8.1.9 ST_GeomFromText . . . 205
7.8.1.10 ST_LineFromText . . . 206
7.8.1.11 ST_MLineFromText . . . 207
7.8.1.12 ST_MPointFromText . . . 208
7.8.1.13 ST_MPolyFromText . . . 209
7.8.1.14 ST_PointFromText . . . 209
7.8.1.15 ST_PolygonFromText . . . 210
7.8.1.16 ST_WKTToSQL . . . 211
7.8.2 已知的二□制文件 (WKB) . . . 212
7.8.2.1 ST_GeogFromWKB . . . 212
7.8.2.2 ST_GeomFromEWKB . . . 212
7.8.2.3 ST_GeomFromWKB . . . 214
7.8.2.4 ST_LineFromWKB . . . 214
7.8.2.5 ST_LinestringFromWKB . . . 215
7.8.2.6 ST_PointFromWKB . . . 216

7.8.2.7 ST_WKBToSQL . . . 217
7.8.3 其它格式 . . . 218
7.8.3.1 ST_Box2dFromGeoHash . . . 218
7.8.3.2 ST_GeomFromGeoHash . . . 218
7.8.3.3 ST_GeomFromGML . . . 219
7.8.3.4 ST_GeomFromGeoJSON . . . 222
7.8.3.5 ST_GeomFromKML . . . 223
7.8.3.6 ST_GeomFromTWKB . . . 224
7.8.3.7 ST_GMLToSQL . . . 224
7.8.3.8 ST_LineFromEncodedPolyline . . . 225
7.8.3.9 ST_PointFromGeoHash . . . 226
7.8.3.10 ST_FromFlatGeobufToTable . . . 226
7.8.3.11 ST_FromFlatGeobuf . . . 227
7.9 几何⍰出 . . . 227
7.9.1 已知文本 (WKT) . . . 227
7.9.1.1 ST_AsEWKT . . . 227
7.9.1.2 ST_AsText . . . 228
7.9.2 已知的二⍰制文件 (WKB) . . . 230
7.9.2.1 ST_AsBinary . . . 230
7.9.2.2 ST_AsEWKB . . . 231
7.9.2.3 ST_AsHEXEWKB . . . 232
7.9.3 其它格式 . . . 233
7.9.3.1 ST_AsEncodedPolyline . . . 233
7.9.3.2 ST_AsFlatGeobuf . . . 234
7.9.3.3 ST_AsGeobuf . . . 235
7.9.3.4 ST_AsGeoJSON . . . 235
7.9.3.5 ST_AsGML . . . 237
7.9.3.6 ST_AsKML . . . 241
7.9.3.7 ST_AsLatLonText . . . 242
7.9.3.8 ST_AsMARC21 . . . 243
7.9.3.9 ST_AsMVTGeom . . . 246
7.9.3.10 ST_AsMVT . . . 247
7.9.3.11 ST_AsSVG . . . 248
7.9.3.12 ST_AsTWKB . . . 249
7.9.3.13 ST_AsX3D . . . 250
7.9.3.14 ST_GeoHash . . . 254
7.10⍰算符 . . . 255
7.10.1⍰界框⍰算符 . . . 255
7.10.1.1 && . . . 255

7.10.1.2&&(geometry,box2df) . . . 256
7.10.1.3&&(box2df,geometry) . . . 257
7.10.1.4&&(box2df,box2df) . . . 257
7.10.1.5&&& . . . 258
7.10.1.6&&&(geometry,gidx) . . . 259
7.10.1.7&&&(gidx,geometry) . . . 260
7.10.1.8&&&(gidx,gidx) . . . 261
7.10.1.9&< . . . 262
7.10.1.10&<| . . . 263
7.10.1.11&> . . . 263
7.10.1.12<< . . . 264
7.10.1.13<<| . . . 265
7.10.1.14= . . . 266
7.10.1.15>> . . . 267
7.10.1.16@ . . . 268
7.10.1.17@(geometry,box2df) . . . 269
7.10.1.18@(box2df,geometry) . . . 269
7.10.1.19@(box2df,box2df) . . . 270
7.10.1.20|&> . . . 271
7.10.1.21|>> . . . 272
7.10.1.22~ . . . 272
7.10.1.23~(geometry,box2df) . . . 273
7.10.1.24~(box2df,geometry) . . . 274
7.10.1.25~(box2df,box2df) . . . 275
7.10.1.26~= . . . 275
7.10.2距离⊠算符 . . . 276
7.10.2.1<-> . . . 276
7.10.2.2|=| . . . 278
7.10.2.3<#> . . . 279
7.10.2.4<<->> . . . 280
7.11空⊠关系 . . . 281
7.11.1拓扑关系 . . . 281
7.11.1.1ST_3DIntersects . . . 281
7.11.1.2ST_Contains . . . 282
7.11.1.3ST_ContainsProperly . . . 286
7.11.1.4ST_CoveredBy . . . 288
7.11.1.5ST_Covers . . . 289
7.11.1.6ST_Crosses . . . 290
7.11.1.7ST_Disjoint . . . 292

7.11.1.8 ST_Equals . . . 293
7.11.1.9 ST_Intersects . . . 294
7.11.1.10 ST_LineCrossingDirection . . . 296
7.11.1.11 ST_OrderingEquals . . . 299
7.11.1.12 ST_Overlaps . . . 300
7.11.1.13 ST_Relate . . . 303
7.11.1.14 ST_RelateMatch . . . 305
7.11.1.15 ST_Touches . . . 306
7.11.1.16 ST_Within . . . 308
7.11.2 距离关系 . . . 309
7.11.2.1 ST_3DDWithin . . . 309
7.11.2.2 ST_3DDFullyWithin . . . 310
7.11.2.3 ST_DFullyWithin . . . 311
7.11.2.4 ST_DWithin . . . 312
7.11.2.5 ST_PointInsideCircle . . . 313
7.12 测量函数 . . . 314
7.12.1 ST_Area . . . 314
7.12.2 ST_Azimuth . . . 316
7.12.3 ST_Angle . . . 317
7.12.4 ST_ClosestPoint . . . 318
7.12.5 ST_3DClosestPoint . . . 320
7.12.6 ST_Distance . . . 321
7.12.7 ST_3DDistance . . . 323
7.12.8 ST_DistanceSphere . . . 324
7.12.9 ST_DistanceSpheroid . . . 325
7.12.10 ST_FrechetDistance . . . 326
7.12.11 ST_HausdorffDistance . . . 327
7.12.12 ST_Length . . . 328
7.12.13 ST_Length2D . . . 330
7.12.14 ST_3DLength . . . 330
7.12.15 ST_LengthSpheroid . . . 331
7.12.16 ST_LongestLine . . . 332
7.12.17 ST_3DLongestLine . . . 334
7.12.18 ST_MaxDistance . . . 335
7.12.19 ST_3DMaxDistance . . . 336
7.12.20 ST_MinimumClearance . . . 337
7.12.21 ST_MinimumClearanceLine . . . 338
7.12.22 ST_Perimeter . . . 338
7.12.23 ST_Perimeter2D . . . 340

7.12.24 ST_3DPerimeter . . . 340
7.12.25 ST_ShortestLine . . . 341
7.12.26 ST_3DShortestLine . . . 343
7.13 叠加函数 . . . 344
7.13.1 ST_ClipByBox2D . . . 344
7.13.2 ST_Difference . . . 345
7.13.3 ST_Intersection . . . 346
7.13.4 ST_MemUnion . . . 349
7.13.5 ST_Node . . . 349
7.13.6 ST_Split . . . 350
7.13.7 ST_Subdivide . . . 353
7.13.8 ST_SymDifference . . . 355
7.13.9 ST_UnaryUnion . . . 357
7.13.10 ST_Union . . . 357
7.14 几何处理 . . . 360
7.14.1 ST_Buffer . . . 360
7.14.2 ST_BuildArea . . . 364
7.14.3 ST_Centroid . . . 366
7.14.4 ST_ChaikinSmoothing . . . 368
7.14.5 ST_ConcaveHull . . . 370
7.14.6 ST_ConvexHull . . . 373
7.14.7 ST_DelaunayTriangles . . . 374
7.14.8 ST_FilterByM . . . 379
7.14.9 ST_GeneratePoints . . . 380
7.14.10 ST_GeometricMedian . . . 381
7.14.11 ST_LineMerge . . . 382
7.14.12 ST_MaximumInscribedCircle . . . 385
7.14.13 ST_LargestEmptyCircle . . . 387
7.14.14 ST_MinimumBoundingCircle . . . 389
7.14.15 ST_MinimumBoundingRadius . . . 390
7.14.16 ST_OrientedEnvelope . . . 391
7.14.17 ST_OffsetCurve . . . 392
7.14.18 ST_PointOnSurface . . . 396
7.14.19 ST_Polygonize . . . 398
7.14.20 ST_ReducePrecision . . . 400
7.14.21 ST_SharedPaths . . . 402
7.14.22 ST_Simplify . . . 404
7.14.23 ST_SimplifyPreserveTopology . . . 405
7.14.24 ST_SimplifyPolygonHull . . . 407

7.14.25 ST_SimplifyVW . . . 410
7.14.26 ST_SetEffectiveArea . . . 411
7.14.27 ST_TriangulatePolygon . . . 413
7.14.28 ST_VoronoiLines . . . 415
7.14.29 ST_VoronoiPolygons . . . 416
7.15 覆盖范 . . . 418
7.15.1 ST_CoverageInvalidEdges . . . 418
7.15.2 ST_CoverageSimplify . . . 419
7.15.3 ST_CoverageUnion . . . 421
7.16 仿射 . . . 422
7.16.1 ST_Affine . . . 422
7.16.2 ST_Rotate . . . 424
7.16.3 ST_RotateX . . . 425
7.16.4 ST_RotateY . . . 426
7.16.5 ST_RotateZ . . . 427
7.16.6 ST_Scale . . . 428
7.16.7 ST_Translate . . . 429
7.16.8 ST_TransScale . . . 430
7.17 聚函数 . . . 431
7.17.1 ST_ClusterDBSCAN . . . 431
7.17.2 ST_ClusterIntersecting . . . 434
7.17.3 ST_ClusterIntersectingWin . . . 434
7.17.4 ST_ClusterKMeans . . . 435
7.17.5 ST_ClusterWithin . . . 437
7.17.6 ST_ClusterWithinWin . . . 438
7.18 界框函数 . . . 439
7.18.1 Box2D . . . 439
7.18.2 Box3D . . . 440
7.18.3 ST_EstimatedExtent . . . 440
7.18.4 ST_Expand . . . 441
7.18.5 ST_Extent . . . 443
7.18.6 ST_3DExtent . . . 444
7.18.7 ST_MakeBox2D . . . 445
7.18.8 ST_3DMakeBox . . . 446
7.18.9 ST_XMax . . . 447
7.18.10 ST_XMin . . . 448
7.18.11 ST_YMax . . . 449
7.18.12 ST_YMin . . . 450
7.18.13 ST_ZMax . . . 451

7.18.18 ST_ZMin . . . 452
7.19 ☒性参考 . . . 453
7.19.1 ST_LineInterpolatePoint . . . 453
7.19.2 ST_3DLineInterpolatePoint . . . 454
7.19.3 ST_LineInterpolatePoints . . . 455
7.19.4 ST_LineLocatePoint . . . 456
7.19.5 ST_LineSubstring . . . 457
7.19.6 ST_LocateAlong . . . 459
7.19.7 ST_LocateBetween . . . 460
7.19.8 ST_LocateBetweenElevations . . . 462
7.19.9 ST_InterpolatePoint . . . 462
7.19.10 ST_AddMeasure . . . 463
7.20 ☒迹函数 . . . 464
7.20.1 ST_IsValidTrajectory . . . 464
7.20.2 ST_ClosestPointOfApproach . . . 465
7.20.3 ST_DistanceCPA . . . 466
7.20.4 ST_CPAWithin . . . 466
7.21 版本函数 . . . 467
7.21.1 PostGIS_Extensions_Upgrade . . . 467
7.21.2 PostGIS_Full_Version . . . 468
7.21.3 PostGIS_GEOS_Version . . . 469
7.21.4 PostGIS_GEOS_Compiled_Version . . . 469
7.21.5 PostGIS_Liblwgeom_Version . . . 470
7.21.6 PostGIS_LibXML_Version . . . 470
7.21.7 PostGIS_LibJSON_Version . . . 471
7.21.8 PostGIS_Lib_Build_Date . . . 471
7.21.9 PostGIS_Lib_Version . . . 472
7.21.10 PostGIS_PROJ_Version . . . 472
7.21.11 PostGIS_PROJ_Compiled_Version . . . 473
7.21.12 PostGIS_Wagyu_Version . . . 473
7.21.13 PostGIS_Scripts_Build_Date . . . 474
7.21.14 PostGIS_Scripts_Installed . . . 475
7.21.15 PostGIS_Scripts_Released . . . 475
7.21.16 PostGIS_Version . . . 476
7.22 大一☒自定☒☒量 (GUCs) . . . 476
7.22.1 postgis.gdal_datapath . . . 476
7.22.2 postgis.gdal_enabled_drivers . . . 477
7.22.3 postgis.enable_outdb_rasters . . . 478
7.22.4 postgis.gdal_vsi_options . . . 479
7.23 故障排除函数 . . . 480
7.23.1 PostGIS_AddBBox . . . 480
7.23.2 PostGIS_DropBBox . . . 481
7.23.3 PostGIS_HasBBox . . . 481

8 SFCGAL 函数参考 483
8.1 SFCGAL 管理函数 . . . 483
8.1.1 postgis_sfcgal_version . . . 483
8.1.2 postgis_sfcgal_full_version . . . 483
8.2 SFCGAL ☒☒器和☒置器 . . . 484
8.2.1 CG_ForceLHR . . . 484
8.2.2 CG_IsPlanar . . . 484
8.2.3 CG_IsSolid . . . 485
8.2.4 CG_MakeSolid . . . 485
8.2.5 CG_Orientation . . . 485
8.2.6 CG_Area . . . 486
8.2.7 CG_3DArea . . . 486
8.2.8 CG_Volume . . . 487
8.2.9 ST_ForceLHR . . . 488
8.2.10 ST_IsPlanar . . . 489
8.2.11 ST_IsSolid . . . 489
8.2.12 ST_MakeSolid . . . 490
8.2.13 ST_Orientation . . . 490
8.2.14 ST_3DArea . . . 491
8.2.15 ST_Volume . . . 492
8.3 SFCGAL ☒理和关系函数 . . . 493
8.3.1 CG_Intersection . . . 493
8.3.2 CG_Intersects . . . 494
8.3.3 CG_3DIntersects . . . 494
8.3.4 CG_Difference . . . 495
8.3.5 ST_3DDifference . . . 496
8.3.6 CG_3DDifference . . . 497
8.3.7 CG_Distance . . . 498
8.3.8 CG_3DDistance . . . 499
8.3.9 ST_3DConvexHull . . . 500
8.3.10 CG_3DConvexHull . . . 500
8.3.11 ST_3DIntersection . . . 501
8.3.12 CG_3DIntersection . . . 502
8.3.13 CG_Union . . . 504
8.3.14 ST_3DUnion . . . 505
8.3.15 CG_3DUnion . . . 505
8.3.16 ST_AlphaShape . . . 507
8.3.17 CG_AlphaShape . . . 507
8.3.18 CG_ApproxConvexPartition . . . 510

8.3.19ST_ApproximateMedialAxis . . . 511
8.3.20CG_ApproximateMedialAxis . . . 512
8.3.21ST_ConstrainedDelaunayTriangles . . . 513
8.3.22CG_ConstrainedDelaunayTriangles . . . 513
8.3.23ST_Extrude . . . 515
8.3.24CG_Extrude . . . 515
8.3.25CG_ExtrudeStraightSkeleton . . . 517
8.3.26CG_GreeneApproxConvexPartition . . . 518
8.3.27ST_MinkowskiSum . . . 519
8.3.28CG_MinkowskiSum . . . 519
8.3.29ST_OptimalAlphaShape . . . 521
8.3.30CG_OptimalAlphaShape . . . 522
8.3.31CG_OptimalConvexPartition . . . 524
8.3.32CG_StraightSkeleton . . . 525
8.3.33ST_StraightSkeleton . . . 526
8.3.34ST_Tesselate . . . 528
8.3.35CG_Tesselate . . . 528
8.3.36CG_Triangulate . . . 530
8.3.37CG_Visibility . . . 531
8.3.38CG_YMonotonePartition . . . 532

9 拓扑☒☒ 534
9.1 拓扑☒型 . . . 534
9.1.1 getfaceedges_returntype . . . 534
9.1.2 TopoGeometry . . . 534
9.1.3 validatetopology_returntype . . . 535
9.2 拓扑域 . . . 535
9.2.1 TopoElement . . . 535
9.2.2 TopoElementArray . . . 536
9.3 拓扑和拓扑几何管理 . . . 537
9.3.1 AddTopoGeometryColumn . . . 537
9.3.2 RenameTopoGeometryColumn . . . 538
9.3.3 DropTopology . . . 538
9.3.4 RenameTopology . . . 539
9.3.5 DropTopoGeometryColumn . . . 539
9.3.6 Populate_Topology_Layer . . . 540
9.3.7 TopologySummary . . . 541
9.3.8 ValidateTopology . . . 542
9.3.9 ValidateTopologyRelation . . . 543

9.3.10FindTopology . . . 544
9.3.11FindLayer . . . 544
9.4 拓扑☒☒管理 . . . 545
9.5 拓扑☒造器 . . . 545
9.5.1 CreateTopology . . . 545
9.5.2 CopyTopology . . . 546
9.5.3 ST_InitTopoGeo . . . 547
9.5.4 ST_CreateTopoGeo . . . 547
9.5.5 TopoGeo_AddPoint . . . 548
9.5.6 TopoGeo_AddLineString . . . 548
9.5.7 TopoGeo_AddPolygon . . . 549
9.5.8 TopoGeo_LoadGeometry . . . 550
9.6 拓扑☒☒器 . . . 550
9.6.1 ST_AddIsoNode . . . 550
9.6.2 ST_AddIsoEdge . . . 551
9.6.3 ST_AddEdgeNewFaces . . . 551
9.6.4 ST_AddEdgeModFace . . . 552
9.6.5 ST_RemEdgeNewFace . . . 553
9.6.6 ST_RemEdgeModFace . . . 553
9.6.7 ST_ChangeEdgeGeom . . . 554
9.6.8 ST_ModEdgeSplit . . . 555
9.6.9 ST_ModEdgeHeal . . . 555
9.6.10ST_NewEdgeHeal . . . 556
9.6.11ST_MoveIsoNode . . . 556
9.6.12ST_NewEdgesSplit . . . 557
9.6.13ST_RemoveIsoNode . . . 558
9.6.14ST_RemoveIsoEdge . . . 558
9.7 拓扑☒☒器 . . . 559
9.7.1 GetEdgeByPoint . . . 559
9.7.2 GetFaceByPoint . . . 560
9.7.3 GetFaceContainingPoint . . . 561
9.7.4 GetNodeByPoint . . . 561
9.7.5 GetTopologyID . . . 562
9.7.6 GetTopologySRID . . . 563
9.7.7 GetTopologyName . . . 563
9.7.8 ST_GetFaceEdges . . . 564
9.7.9 ST_GetFaceGeometry . . . 565
9.7.10GetRingEdges . . . 565
9.7.11GetNodeEdges . . . 566

9.8 拓扑□理 . . . 566
9.8.1 Polygonize . . . 566
9.8.2 AddNode . . . 567
9.8.3 AddEdge . . . 568
9.8.4 AddFace . . . 569
9.8.5 ST_Simplify . . . 571
9.8.6 RemoveUnusedPrimitives . . . 571
9.9 拓扑几何□造函数 . . . 572
9.9.1 CreateTopoGeom . . . 572
9.9.2 toTopoGeom . . . 573
9.9.3 TopoElementArray_Agg . . . 575
9.9.4 TopoElement . . . 575
9.10拓扑几何□□器 . . . 576
9.10.1clearTopoGeom . . . 576
9.10.2TopoGeom_addElement . . . 576
9.10.3TopoGeom_remElement . . . 577
9.10.4TopoGeom_addTopoGeom . . . 577
9.10.5toTopoGeom . . . 578
9.11拓扑几何□□器 . . . 578
9.11.1GetTopoGeomElementArray . . . 578
9.11.2GetTopoGeomElements . . . 579
9.11.3ST_SRID . . . 579
9.12拓扑几何□出 . . . 580
9.12.1AsGML . . . 580
9.12.2AsTopoJSON . . . 582
9.13拓扑空□关系 . . . 584
9.13.1Equals . . . 584
9.13.2Intersects . . . 585
9.14□入和□出拓扑 . . . 585
9.14.1使用拓扑□出器 . . . 586
9.14.2使用拓扑□入器 . . . 586

10□格数据管理、□□和□用程序 **587**
10.1加□和□建□格 . . . 587
10.1.1使用 raster2pgsql 加□□格 . . . 587
10.1.1.1用法示例 . . . 587
10.1.1.2raster2pgsql □□ . . . 588
10.1.2使用 PostGIS □格函数□建□格 . . . 589
10.1.3使用“out db”云□格 . . . 590

10.2 栅格目录 . . . 591
10.2.1 栅格列目录 . . . 591
10.2.2 栅格概述 . . . 592
10.3 使用 PostGIS Raster 创建自定义应用程序 . . . 592
10.3.1 PHP 示例使用 ST_AsPNG 与其他栅格函数行输出 . . . 593
10.3.2 ASP.NET C# 示例使用 ST_AsPNG 与其他栅格函数行输出 . . . 593
10.3.3 将栅格数据输出为图像文件的 Java 控制台应用程序 . . . 595
10.3.4 使用 PLPython 通过 SQL 处理栅格 . . . 596
10.3.5 使用 PSQL 输出栅格 . . . 597

11 栅格参考 **598**
11.1 栅格支持数据类型 . . . 599
11.1.1 geomval . . . 599
11.1.2 addbandarg . . . 599
11.1.3 rastbandarg . . . 599
11.1.4 raster . . . 600
11.1.5 reclassarg . . . 600
11.1.6 summarystats . . . 601
11.1.7 unionarg . . . 601
11.2 栅格管理 . . . 602
11.2.1 AddRasterConstraints . . . 602
11.2.2 DropRasterConstraints . . . 604
11.2.3 AddOverviewConstraints . . . 605
11.2.4 DropOverviewConstraints . . . 606
11.2.5 PostGIS_GDAL_Version . . . 606
11.2.6 PostGIS_Raster_Lib_Build_Date . . . 606
11.2.7 PostGIS_Raster_Lib_Version . . . 607
11.2.8 ST_GDALDrivers . . . 607
11.2.9 ST_Contour . . . 612
11.2.10 ST_InterpolateRaster . . . 613
11.2.11 UpdateRasterSRID . . . 614
11.2.12 ST_CreateOverview . . . 614
11.3 栅格构造器 . . . 615
11.3.1 ST_AddBand . . . 615
11.3.2 ST_AsRaster . . . 618
11.3.3 ST_Band . . . 620
11.3.4 ST_MakeEmptyCoverage . . . 621
11.3.5 ST_MakeEmptyRaster . . . 623
11.3.6 ST_Tile . . . 624

11.3.7ST_Retile . . . 626
11.3.8ST_FromGDALRaster . . . 626
11.4栅格访问器 . . . 627
11.4.1ST_GeoReference . . . 627
11.4.2ST_Height . . . 628
11.4.3ST_IsEmpty . . . 629
11.4.4ST_MemSize . . . 629
11.4.5ST_MetaData . . . 630
11.4.6ST_NumBands . . . 631
11.4.7ST_PixelHeight . . . 631
11.4.8ST_PixelWidth . . . 632
11.4.9ST_ScaleX . . . 634
11.4.10ST_ScaleY . . . 634
11.4.11ST_RasterToWorldCoord . . . 635
11.4.12ST_RasterToWorldCoordX . . . 636
11.4.13ST_RasterToWorldCoordY . . . 637
11.4.14ST_Rotation . . . 638
11.4.15ST_SkewX . . . 638
11.4.16ST_SkewY . . . 639
11.4.17ST_SRID . . . 640
11.4.18ST_Summary . . . 641
11.4.19ST_UpperLeftX . . . 641
11.4.20ST_UpperLeftY . . . 642
11.4.21ST_Width . . . 642
11.4.22ST_WorldToRasterCoord . . . 643
11.4.23ST_WorldToRasterCoordX . . . 644
11.4.24ST_WorldToRasterCoordY . . . 644
11.5栅格波段访问器 . . . 645
11.5.1ST_BandMetaData . . . 645
11.5.2ST_BandNoDataValue . . . 646
11.5.3ST_BandIsNoData . . . 647
11.5.4ST_BandPath . . . 648
11.5.5ST_BandFileSize . . . 649
11.5.6ST_BandFileTimestamp . . . 649
11.5.7ST_BandPixelType . . . 650
11.5.8ST_MinPossibleValue . . . 651
11.5.9ST_HasNoBand . . . 651
11.6栅格像素访问器和设置器 . . . 652
11.6.1ST_PixelAsPolygon . . . 652

11.6.2ST_PixelAsPolygons . . . 653
11.6.3ST_PixelAsPoint . . . 654
11.6.4ST_PixelAsPoints . . . 654
11.6.5ST_PixelAsCentroid . . . 655
11.6.6ST_PixelAsCentroids . . . 656
11.6.7ST_Value . . . 657
11.6.8ST_NearestValue . . . 660
11.6.9ST_SetZ . . . 662
11.6.10ST_SetM . . . 663
11.6.11ST_Neighborhood . . . 664
11.6.12ST_SetValue . . . 666
11.6.13ST_SetValues . . . 667
11.6.14ST_DumpValues . . . 675
11.6.15ST_PixelOfValue . . . 676
11.7栅格处理器 . . . 678
11.7.1ST_SetGeoReference . . . 678
11.7.2ST_SetRotation . . . 679
11.7.3ST_SetScale . . . 680
11.7.4ST_SetSkew . . . 681
11.7.5ST_SetSRID . . . 682
11.7.6ST_SetUpperLeft . . . 682
11.7.7ST_Resample . . . 683
11.7.8ST_Rescale . . . 684
11.7.9ST_Reskew . . . 685
11.7.10ST_SnapToGrid . . . 687
11.7.11ST_Resize . . . 688
11.7.12ST_Transform . . . 689
11.8栅格波段处理器 . . . 692
11.8.1ST_SetBandNoDataValue . . . 692
11.8.2ST_SetBandIsNoData . . . 693
11.8.3ST_SetBandPath . . . 694
11.8.4ST_SetBandIndex . . . 696
11.9栅格波段统计和分析 . . . 697
11.9.1ST_Count . . . 697
11.9.2ST_CountAgg . . . 698
11.9.3ST_Histogram . . . 699
11.9.4ST_Quantile . . . 701
11.9.5ST_SummaryStats . . . 703
11.9.6ST_SummaryStatsAgg . . . 705

11.9.7 ST_ValueCount . . . 706
11.10 栅格输入 . . . 709
11.10.1 ST_RastFromWKB . . . 709
11.10.2 ST_RastFromHexWKB . . . 709
11.11 栅格输出 . . . 710
11.11.1 ST_AsBinary/ST_AsWKB . . . 710
11.11.2 ST_AsHexWKB . . . 711
11.11.3 ST_AsGDALRaster . . . 712
11.11.4 ST_AsJPEG . . . 713
11.11.5 ST_AsPNG . . . 714
11.11.6 ST_AsTIFF . . . 715
11.12 栅格处理：地图代数 . . . 716
11.12.1 ST_Clip . . . 716
11.12.2 ST_ColorMap . . . 719
11.12.3 ST_Grayscale . . . 722
11.12.4 ST_Intersection . . . 724
11.12.5 ST_MapAlgebra (callback function version) . . . 726
11.12.6 ST_MapAlgebra (expression version) . . . 732
11.12.7 ST_MapAlgebraExpr . . . 735
11.12.8 ST_MapAlgebraExpr . . . 737
11.12.9 ST_MapAlgebraFct . . . 741
11.12.10 ST_MapAlgebraFct . . . 745
11.12.11 ST_MapAlgebraFctNgb . . . 749
11.12.12 ST_Reclass . . . 751
11.12.13 ST_Union . . . 753
11.13 内置地图代数回调函数 . . . 754
11.13.1 ST_Distinct4ma . . . 754
11.13.2 ST_InvDistWeight4ma . . . 755
11.13.3 ST_Max4ma . . . 756
11.13.4 ST_Mean4ma . . . 757
11.13.5 ST_Min4ma . . . 759
11.13.6 ST_MinDist4ma . . . 760
11.13.7 ST_Range4ma . . . 760
11.13.8 ST_StdDev4ma . . . 761
11.13.9 ST_Sum4ma . . . 762
11.14 栅格处理：DEM（高程） . . . 764
11.14.1 ST_Aspect . . . 764
11.14.2 ST_HillShade . . . 765
11.14.3 ST_Roughness . . . 767

11.14.4 ST_Slope . . . 768
11.14.5 ST_TPI . . . 769
11.14.6 ST_TRI . . . 770
11.15 栅格处理：栅格到几何 . . . 771
11.15.1 Box3D . . . 771
11.15.2 ST_ConvexHull . . . 771
11.15.3 ST_DumpAsPolygons . . . 772
11.15.4 ST_Envelope . . . 774
11.15.5 ST_MinConvexHull . . . 774
11.15.6 ST_Polygon . . . 775
11.16 栅格运算符 . . . 777
11.16.1 && . . . 777
11.16.2 &< . . . 777
11.16.3 &> . . . 778
11.16.4 = . . . 779
11.16.5 @ . . . 779
11.16.6 ~= . . . 780
11.16.7 ~ . . . 781
11.17 栅格和栅格波段空间关系 . . . 781
11.17.1 ST_Contains . . . 781
11.17.2 ST_ContainsProperly . . . 782
11.17.3 ST_Covers . . . 783
11.17.4 ST_CoveredBy . . . 784
11.17.5 ST_Disjoint . . . 785
11.17.6 ST_Intersects . . . 786
11.17.7 ST_Overlaps . . . 787
11.17.8 ST_Touches . . . 788
11.17.9 ST_SameAlignment . . . 788
11.17.10 ST_NotSameAlignmentReason . . . 789
11.17.11 ST_Within . . . 790
11.17.12 ST_DWithin . . . 791
11.17.13 ST_DFullyWithin . . . 792
11.18 栅格提示 . . . 793
11.18.1 Out-DB 栅格 . . . 793
11.18.1.1 包含多文件的目录 . . . 793
11.18.1.2 最大打开文件数 . . . 794
11.18.1.2.1 整个系统最大打开文件数 . . . 794
11.18.1.2.2 每个进程的最大打开文件数 . . . 794

12 PostGIS 扩充 **797**
12.1 地址标准化工具 . . . 797
12.1.1 解析器如何工作 . . . 797
12.1.2 地址标准化器类型 . . . 797
12.1.2.1 stdaddr . . . 797
12.1.3 地址标准化表 . . . 798
12.1.3.1 rules table . . . 798
12.1.3.2 lex table . . . 801
12.1.3.3 gaz table . . . 801
12.1.4 地址标准化器功能 . . . 801
12.1.4.1 debug_standardize_address . . . 801
12.1.4.2 parse_address . . . 803
12.1.4.3 standardize_address . . . 804
12.2 Tiger 地理编码器 . . . 806
12.2.1 Drop_Indexes_Generate_Script . . . 806
12.2.2 Drop_Nation_Tables_Generate_Script . . . 807
12.2.3 Drop_State_Tables_Generate_Script . . . 808
12.2.4 Geocode . . . 809
12.2.5 Geocode_Intersection . . . 811
12.2.6 Get_Geocode_Setting . . . 812
12.2.7 Get_Tract . . . 813
12.2.8 Install_Missing_Indexes . . . 814
12.2.9 Loader_Generate_Census_Script . . . 814
12.2.10 Loader_Generate_Script . . . 816
12.2.11 Loader_Generate_Nation_Script . . . 818
12.2.12 Missing_Indexes_Generate_Script . . . 819
12.2.13 Normalize_Address . . . 820
12.2.14 Pagc_Normalize_Address . . . 821
12.2.15 Pprint_Addy . . . 823
12.2.16 Reverse_Geocode . . . 824
12.2.17 Topology_Load_Tiger . . . 826
12.2.18 Set_Geocode_Setting . . . 828

13 PostGIS 特殊函数索引 **829**
13.1 PostGIS 聚合函数 . . . 829
13.2 PostGIS 窗口函数 . . . 830
13.3 PostGIS SQL-MM 兼容函数 . . . 830
13.4 PostGIS 地理支持函数 . . . 834
13.5 PostGIS 栅格支持函数 . . . 835

13.6 PostGIS 几何/地理/栅格输出函数 . . . 840
13.7 PostGIS 边界框函数 . . . 840
13.8 支持 3D 的 PostGIS 函数 . . . 841
13.9 PostGIS 曲线几何支持函数 . . . 846
13.10 PostGIS 多面体曲面支持函数 . . . 849
13.11 PostGIS 函数支持矩阵 . . . 852
13.12 新的、增强的或更改的 PostGIS 函数 . . . 870
13.12.1 PostGIS 新增功能或增强功能（3.5） . . . 870
13.12.2 PostGIS 新增功能或增强功能（3.4） . . . 872
13.12.3 PostGIS 新增功能或增强功能（3.3） . . . 873
13.12.4 PostGIS 新增功能或增强功能（3.2） . . . 874
13.12.5 PostGIS 新增功能或增强功能（3.1） . . . 875
13.12.6 PostGIS 新增功能或增强功能（3.0） . . . 876
13.12.7 PostGIS 新增功能或增强功能（2.5） . . . 878
13.12.8 PostGIS 新增功能或增强功能（2.4） . . . 879
13.12.9 PostGIS 新增功能或增强功能（2.3） . . . 880
13.12.10 PostGIS 新增功能或增强功能（2.2） . . . 882
13.12.11 PostGIS 新增功能或增强功能（2.1） . . . 884
13.12.12 PostGIS 新增功能或增强功能（2.0） . . . 889
13.12.13 PostGIS 新增功能或增强功能（1.5） . . . 898
13.12.14 PostGIS 新增功能或增强功能（1.4） . . . 899
13.12.15 PostGIS 新增功能或增强功能（1.3） . . . 900

14 报告问题 **901**
14.1 报告软件问题 . . . 901
14.2 报告文档问题 . . . 901

A 附录 **902**
A.1 PostGIS 3.5.3 . . . 902
A.1.1 Bug Fixes . . . 902
A.2 PostGIS 3.5.2 . . . 902
A.2.1 Bug Fixes . . . 903
A.3 PostGIS 3.5.1 . . . 903
A.3.1 重大变化 . . . 903
A.3.2 增强功能 . . . 903
A.3.3 重大变化 . . . 903
A.4 PostGIS 3.5.0 . . . 904
A.4.1 重大变化 . . . 904
A.4.2 Deprecated signatures . . . 904
A.4.3 新特性 . . . 905
A.4.4 增强功能 . . . 905

Abstract

PostGIS 是一种⍰展功能，适用于 PostgreSQL ⍰象关系数据⍰系⍰，它允⍰将 GIS（地理信息系⍰）⍰象存⍰在数据⍰中。PostGIS 包含⍰基于 GiST 的 R-Tree 空⍰索引的支持，以及用于分析和⍰理 GIS ⍰象的功能。

本手册的版本是 3.5.4dev

本作品已⍰得⍰作共用署名-相同方式共享 3.0 ⍰可。⍰随意使用⍰份材料，任何您喜⍰的方式，但我⍰要求您将荣誉⍰于 PostGIS ⍰目，尽可能提供返回https://postgis.net的⍰接。

Chapter 1

介□

PostGIS 是 PostgreSQL 关系数据□的空□□展。由 Refractions Research Inc □建，作□空□数据□技□的研究□目。Refractions 是一家位于加拿大不列□哥□比□省□多利□市的 GIS 和数据□咨□公司，□□从事数据集成和定制□件开□。

PostGIS □在是 OSGeo 基金会的一个□目。世界各地大量 FOSS4G 开□商和公司从 PostGIS 的功能和多□性中受益匪浅，他□正在开□和□助 PostGIS。

PostGIS □目开□小□□ PostGIS □建了一个数据源，以便在 OGC 和 SQL/MM 空□□准、高□拓扑□建（覆盖范□、表面、网□）、用于□看和□□ GIS 数据的桌面用□界面工具的数据源以及基于 Web 的□□工具的数据源等□域提供重要的 GIS 功能。我□将提供支持和增□功能，以更好地做出□□。

1.1 □目指□委□会

PostGIS □目指□委□会（PSC）□□□□□体方向、□布周期、文档和支持活□。委□会□□涉及 PostGIS 的其他□□□行投票，例如整体用□支持、接受和□用 PostGIS 社区的□丁、开□人□提交□□□限、新委□会成□以及重要的 API 更改。

Raúl Marín Rodríguez MVT 功能、□□、性能和□定性改□、GitHub 策展、PostGIS 和 PostgreSQL 版本□整

Regina Obe 持□集成和网站□□，Windows 生□和□□□建，文档□写，将 PostGIS 与 PostgreSQL □布版本□□，X3D 支持，TIGER 地理□□支持，管理功能。

Darafei Praliaskouski 索引改□，故障修复和几何/地理函数改□，SFCGAL，□格，GitHub □□，以及持□集成□□。

Paul Ramsey（主席） PostGIS □目的□合□始人。全面的□□修复、地理功能、地理和几何索引（2D、3D、n□索引和任何空□索引）、几何内部、GEOS 功能集成以及与 GEOS 版本的□□、与 PostgreSQL 版本的□□、加□器/□□程序、shapefile GUI 加□器。

Sandro Santilli □□修复和□□，持□集成□□，Git □像管理，管理功能，集成新的 GEOS 功能并与 GEOS □布□□，拓扑支持，以及□格框架和底□ API 函数。

1.2 当前的核心□献者

Nicklas Avén 距离函数增□（包括 3D 距离和关系函数）和附加功能、Tiny WKB (TWKB) □出格式和一般用□支持

Loïc Bartoletti SFCGAL 的增□和□□以及持□集成支持

Dan Baston 几何聚□功能添加、其他几何算法增□、GEOS 增□和一般用□支持

Martin Davis GEOS 增□功能和文档

Björn Harrtell MapBox 矢量瓦片、GeoBuf 和 Flatgeobuf 函数。Gitea □□和 GitLab □□。

Aliaksandr Kalenik 几何□理，PostgreSQL GiST，常□□□

1.3 □去的核心□献者

Bborie Park 前 PSC 成□。□格，与 GDAL 集成，□格加□器，用□支持，常□□□，在各种操作系□（Slackware，Mac，Windows 等）上□行□□

Mark Cave-Ayland 前 PSC 成□。□□和□□活□、空□索引□□性和□定、加□器/□□程序和 shapefile GUI 加□器□整、新功能集成和增□。

Jorge Arévalo □格开□，GDAL □□功能，加□器

Olivier Courtin（荣誉）XML（KML，GML）/GEOJSON □入/□出函数、3D □□和□□。

Chris Hodgson 前 PSC 成□。一般开□，站点和□建机器人□□，OSGeo 孵化管理

Mateusz Loskot CMake □ PostGIS 的支持、原始□格加□器的□造以及 Python 版本的低□□格 API 函数

Kevin Neufeld 前 PSC 成□。文档和文档□助、Buildbot □□、PostGIS 新□□中的高□用□支持以及增□的 PostGIS □□功能。

Dave Blasby PostGIS 的原始开□者/□合□始人。□写了服□器端□象、索引□定和□多服□器端分析函数。

Jeff Lounsbury shapefile loader/dumper 的原始开□人□。

Mark Leslie 持□□□和开□核心功能。增□的曲□功能。形状文件 GUI 加□器。

Pierre Racine 后 GIS □格□□□□。光□架□、原型□□和□程□助

David Zwarg □格开□（主要是地□代数分析函数）

1.4 其它□献者

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

企☒☒体 ☒些公司☒ PostGIS ☒目☒献了开☒人☒☒☒，托管或直接☒金。按字母☒序排列：

- 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

☒筹活☒ ☒筹活☒是我☒☒了☒得急需的功能而开展的活☒，☒些功能可以☒大量的人提供服☒。每个活☒都☒☒☒☒一个特定功能或一☒功能。每个☒助商都会投入所需☒金的一小部分，并且有足☒的个人/☒☒捐款，我☒就有☒金来支付将帮助☒多人的工作。如果您☒某个功能有想法，并且☒☒☒多其他人愿意共同☒助，☒将您的想法☒布到 PostGIS 新☒☒，我☒可以共同☒☒它。

PostGIS 2.0.0 是我☒☒☒此策略的第一个版本。我☒使用了PledgeBank，并从中☒☒了☒次成功的活☒。

postgistopology-有 10 位☒助商每人捐☒了 250 美元来☒建”toTopoGeometry” 函数并加☒ 2.0.0 版本中的拓扑支持。☒一目☒已☒☒☒。

postgis64windows-大☒ 20 位☒助商每人捐☒了 100 美元，用以支付在 Windows 上解决 PostGIS 64 位☒☒所需的工作。☒一目☒已☒☒☒。

重要的支持☒ GEOS几何操作☒

GDAL地理空☒数据抽象☒用于支持 PostGIS 2 中引入的大部分☒格功能。同☒，GDAL 中支持 PostGIS 所需的改☒也将回☒☒ GDAL ☒目。

PROJ地☒投影☒

最后但同☒重要的是，PostGIS 所依☒的☒然大物PostgreSQL。PostGIS 的速度和灵活性很大程度上得益于 PostgreSQL 提供的可☒展性、☒大的☒☒☒划器、GIST 索引以及丰富的 SQL 功能。

Chapter 2

PostGIS 安装

本章介⍰安装 PostGIS 所需的步⍰。

2.1 ⍰短版本

如果所有依⍰⍰都在路径中，⍰按如下方式⍰⍰：

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

安装 PostGIS 后，您需要使每个数据⍰可用（Section 3.3）或升⍰它（Section 3.4）。

2.2 从源代⍰⍰⍰和安装

Note
⍰在⍰多操作系⍰都包含 PostgreSQL/PostGIS 的⍰⍰建包。在⍰多情况下，只有当您想要最前沿的版本或者您是⍰件包⍰⍰者⍰才需要⍰⍰。
本⍰提供一般⍰⍰⍰程。如果您正在⍰⍰ Windows 或其他操作系⍰⍰行⍰⍰，⍰可以在PostGIS 用⍰⍰⍰指南或PostGIS 开⍰ Wiki中找到更⍰⍰的帮助。
可以在 PostGIS ⍰⍰建包中找到⍰多操作系⍰的PostGIS ⍰⍰建包列表
如果您是 Windows 用⍰，您可以通⍰ Stackbuilder 或 PostGIS Windows 下⍰站点⍰得⍰定的版本。我⍰⍰有非常前沿的Windows ⍰⍰性版本，通常每周⍰建一次，或者在有令人⍰⍰的新⍰化⍰生⍰⍰建。你可以使用⍰些版本来⍰⍰正在⍰行中的 PostGIS ⍰布

PostGIS 模⍰是 PostgreSQL 服⍰的⍰展。PostGIS3.5.4dev 需要完整的 PostgreSQL 服⍰器⍰⍰⍰行⍰⍰。你可以在 PostgreSQL 12 - 17 之⍰⍰建。不支持旧版本的 PostgreSQL。

如果您没有安装 PostgreSQL，⍰参⍰ PostgreSQL 安装指南。它位于 https://www.postgresql.org 。

Note

⊠于 GEOS 功能，当您安装 PostgresQL ⊠，您可能需要将 PostgreSQL ⊠式⊠接到⊠准 C++ ⊠：

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

⊠是解决与旧开⊠工具不正常交互的虚假 C++ 异常的一种⊠通方法。如果你遇到奇怪的⊠⊠（例如，后端意外关⊠或⊠似的⊠⊠），⊠⊠⊠个技巧。当然，⊠将需要重新从⊠⊠⊠你的 PostgreSQL。

下一步是概述 PostGIS 源配置和⊠⊠。⊠些是⊠ Linux 用⊠⊠写的，不适用于 Windows 或 Mac。

2.2.1 ⊠取源代⊠

从下⊠站点 https://postgis.net/stuff/postgis-3.5.4dev.tar.gz ⊠取源代⊠的存档

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

⊠将在当前工作目⊠中⊠建一个名⊠ `postgis-3.5.4dev` 的目⊠。

或者从git 存⊠⊠ https://git.osgeo.org/gitea/postgis/postgis/ 中⊠看它。

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

切⊠到新⊠建的 `postgis` 目⊠以⊠⊠安装。

```
./configure
```

2.2.2 安装要求

要⊠建和使用 PostGIS，您需要：

不可缺少

- PostgreSQL12 - 16。需要完整安装 PostgreSQL（包括服⊠器⊠⊠）。PostgreSQL 可从https://www.postgresql.org ⊠取。

  ⊠参⊠ https://trac.osgeo.org/postgis/wiki/UsersWikiPostgreSQLPostGIS 中完整的 PosgreSQL/PostGIS 和 PostGIS/GEOS 支持表

- GNU C ⊠⊠器 (`gcc`)。其他一些 ANSI C ⊠⊠器也可以用来⊠⊠ PostGIS，但使用`gcc` ⊠⊠⊠我⊠⊠⊠⊠⊠要少得多。

- GNU Make（`gmake` or `make`）。⊠于⊠多系⊠，GNU `make` 是 make 的默⊠版本。通⊠⊠用`make -v` ⊠⊠版本。其他版本的`make` 可能无法正确⊠理 PostGIS `Makefile`。

- Proj 重投影⊠。需要⊠目 6.1 或更高版本。Proj ⊠用于在 PostGIS 中提供坐⊠重投影支持。Proj 可以从https://proj.org/ 下⊠。

- GEOS 几何⊠，版本 3.8.0 或更高版本，但⊠了充分利用所有新的函数和特性，需要使用 GEOS 3.12+。您可以从以下⊠接下⊠ GEOS：https://libgeos.org 。

- LibXML2、2.5.x 或更高版本。目前，⊠入函数（ST_GeomFromGML 和 ST_GeomFromKML）使用 LibXML2。您可以从 https://gitlab.gnome.org/GNOME/libxml2/-/releases 下⊠ LibXML2。

- JSON-C 0.9 或更高版本。JSON-C 目前用于按 ST_GeomFromGeoJson 区入 GeoJSON。JSON-C 可从 https://github.com/json-c/json-c/releases/ 下区。
- GDAL, version 3+ is preferred. This is required for raster support. https://gdal.org/download.html.
- 要使用 PostgreSQL+JIT 区行区区，需要 LLVM 版本 6 或更高版本。区参区 https://trac.osgeo.org/postgis/-ticket/4125。

可区区的

- GDAL（区可区）区当您不需要区格区才可以将其省略。区区确保区用您要使用的区区程序，如 Section 3.2中所述。
- GTK（需要 GTK+2.0, 2.8+）。用于区区 shp2pgsql-gui，一个 shapefile loader。它位于 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/.
- 要区建 Section 12.1，您区需要 PCREhttp://www.pcre.org（通常安装在 nix 系区上）。区区到 PCRE 区后，将自区区建 Section 12.1 。或者，指定它在配置期区有效。`--with-pcre-dir=/path/to/pcre`。
- 要区用 ST_AsMVT，您需要 ProtoBuf-C 区（区行区）和 ProtoC-C 区区器（区建区）。要区区 protobuf-c 的正确最低版本，需要 pkg-config。参区protobuf-c。默区情况下，PostGIS 使用 Wagyu 快速区估 MVT 多区形，但需要 C++11 区区器。使用 CXXFLAGS 使用您用于 PostgreSQL 安装的相同区区器。如果要禁用此功能并改用 GEOS，区在配置期区指定它。`--without-wagyu`。
- CUnit（`CUnit`）。区是回区区区所需要的。http://cunit.sourceforge.net/
- DocBook (`xsltproc`) 区建文档需要。Docbook 可从http://www.docbook.org/ 区取。
- DBLatex (`dblatex`) 需要以 PDF 格式区建文档。它位于 http://dblatex.sourceforge.net/ 。
- ImageMagick (`convert`) 生成文档中使用的区像。它位于http://www.imagemagick.org/ 。

2.2.3 区建配置

与大多数 Linux 安装一区，第一步是生成一个将用于区建源代区的 Makefile。区是通区 shell 脚本完成的

./configure

如果未提供任何参数，此命令将自区区找在系区上区建 PostGIS 源代区所需的区件和区。**./configure** 是一种常区的用法，但它接受一些参数，以防您在非区准位置区有必要的区或程序。

以下列表区示了常用参数：有关完整列表，区使用 **--help** 或 **--help=short** 参数。

`--with-library-minor-version` 从 PostGIS 3.0 开始，默区生成的区文件将不再将次要版本作区文件名的一部分。区意味着所有 PostGIS 3 区都将以 `postgis-3` 区尾。区区做是区了使 pg_upgrade 更容易，缺点是您只能在服区器中安装一个版本的 PostGIS 3 系列。要区取文件的旧行区（包括次要版本）：例如 `postgis-3.0` 将此开关添加到您的配置区句中。

`--prefix=PREFIX` 指定要安装 PostGIS 区和 SQL 脚本的位置。默区情况下，它将与区区到的 PostgreSQL 安装位置相同。

Caution
此参数当前已⊠坏，只会安装在 PostgreSQL 的位置。有关此⊠⊠的跟踪，⊠参⊠ http://trac.osgeo.org/postgis/ticket/635 。

--with-pgconfig=FILE PostgreSQL 有一个名⊠ **pg_config** 的⊠用程序，使 PostGIS 等⊠展能⊠定位 PostgreSQL 安装目⊠。使用此参数（**--with-pgconfig=/path/to/pg_config**）可以手⊠指定 PostGIS 的特定 PostgreSQL 的安装目⊠。

--with-gdalconfig=FILE GDAL 是必需的⊠，提供⊠格支持 **gdal-config** 所需的功能，以使⊠件安装能⊠找到 GDAL 装目⊠。使用此参数（**--with-gdalconfig=/path/to/gdal-config**）手⊠指定 PostGIS 将⊠⊠其⊠建的特定 GDAL 的安装目⊠。

--with-geosconfig=FILE 作⊠一个基本的几何⊠，GEOS 有一个名⊠ **geos-config** 的⊠用程序，它会告⊠您在安装⊠件⊠在⊠里安装 GEOS。使用此参数（**--with-geosconfig=/path/to/geos-config**）手⊠指定要用于 PostGIS ⊠建的特定 GEOS 的安装目⊠。

--with-xml2config=FILE LibXML 是⊠行 GeomFromKML/GML ⊠程所需的⊠。通常情况下，如果您已安装 libxml，它将被找到，但如果没有或者您想使用特定版本，您需要指定一个特定的 `xml2-config` 配置文件，以便⊠⊠件安装程序找到 LibXML 安装目⊠。使用⊠个参数 (**>--with-xml2config=/path/to/xml2-config**) 来手⊠指定 PostGIS 将⊠建的特定 LibXML 安装。

--with-projdir=DIR Proj 是 PostGIS 必不可少的投影⊠。使用此参数（**--with-projdir=/path/to/projdir**）手⊠指定要用于 PostGIS ⊠建的特定 Proj 的安装目⊠。

--with-libiconv=DIR iconv 的安装目⊠。

--with-jsondir=DIR JSON-C 是 MIT ⊠可的 JSON ⊠，是 PostGIS ST_GeomFromJSON 所必需的。使用此参数（**--with-jsondir=/path/to/jsondir**）手⊠指定要用于 PostGIS ⊠建的特定 JSON-C 的安装目⊠。

--with-pcredir=DIR PCRE 是 address_standardizer ⊠展所需的 BSD ⊠可的 Perl 兼容正⊠表⊠式⊠。使用此参数 (**--with-pcredir=/path/to/pcredir**) 手⊠指定 PostGIS 将⊠建的特定 PCRE 的安装目⊠。

--with-gui ⊠⊠数据⊠入 GUI（需要 GTK+2.0）。此参数⊠ shp2pgsql ⊠建一个名⊠ shp2pgsql-gui 的⊠形用⊠界面。

--without-raster 在没有⊠格功能的情况下⊠行⊠⊠。

--without-topology 在没有拓扑支持的情况下⊠行⊠⊠。拓扑所需的所有⊠⊠都是在 postgis-3.5.4dev ⊠中⊠建的，因此没有关⊠的⊠。

--with-gettext=no 默⊠情况下，会⊠⊠ gettext 并使用它⊠行⊠⊠，但如果在⊠致加⊠程序⊠坏的不兼容⊠⊠下⊠行，⊠此命令可以禁用它。使用此功能的配置可以解决的⊠⊠示例，⊠参⊠http://trac.osgeo.org/postgis/ticket/748。注意：关⊠此功能不会消除⊠多功能。它用于 GUI 加⊠器中的内部帮助/⊠⊠功能，⊠些功能尚未⊠⊠，⊠于⊠⊠⊠段。

--with-sfcgal=PATH 默⊠情况下，如果没有此开关，PostGIS 将不会安装 sfcgal 支持。PATH 是一个可⊠参数，允⊠指定 sfcgal-config 的⊠用 PATH。

--without-phony-revision 禁用 postgis_revision.h 更新以匹配 Git 存⊠⊠中的当前 HEAD。

Note
如果要从代⊠存⊠⊠ ⊠取 PostGIS，⊠先⊠行以下脚本
./autogen.sh
此脚本生成配置脚本。⊠用于自定⊠ PostGIS 安装。
如果要⊠取 PostGIS 作⊠存档文件，⊠无需**./autogen.sh**，因⊠ 配置已⊠生成。

2.2.4 ☒建

生成 Makefile 后，☒建 PostGIS 就像☒行命令一☒☒☒

make

如果在☒出的最后一行看到“`PostGIS was built successfully. Ready to install.`”，☒表示您已完成

从 PostGIS v1.4.0 开始，所有函数都有从文档生成的注☒。如果您希望稍后将☒些注☒安装到空☒数据☒中，☒☒行需要的 docbook 命令。postgis_comments.sql 和其他包注☒文件 raster_comments.sql、topology_comments.sql 也打包在 doc 文件☒中的 tar.gz 中，因此如果从 tar 包中安装，☒无需☒行注☒。注☒包含在 CREATE EXTENSION 安装中。

make comments

它是在 PostGIS 2.0 中引入的。生成适合快速参考和☒☒的 HTML ☒忘☒，供正在学☒的人使用。xsltproc 是必需的，生成 4 个文件。`topology_cheatsheet.html`,`tiger_geocoder_cheatsheet.html`,`raster_cheatsheet.html`, `postgis_cheatsheet.html`

☒☒建的 HTML 和 PDF 版本可以在PostGIS / PostgreSQL 学☒指南中找到

make cheatsheets

2.2.5 ☒建和部署 PostGIS ☒展

如果您使用的是 PostgreSQL 9.1 或更高版本，☒会自☒☒建并安装 PostGIS ☒展。

如果要从源存☒☒☒建，☒必☒先☒建函数描述。☒些是在安装文档手册☒☒建的。您也可以手☒安装：

make comments

如果从存档文件☒建，☒其中一些注☒文件已☒☒建，因此无需☒建注☒。

如果您☒☒ PostgreSQL 9.1 ☒行☒建，☒☒在 make 安装☒程中自☒☒建☒展。如果需要，可以从☒展文件☒生成，或者根据需要在☒一台服☒器上复制☒文件。

```
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` 使用 psql 来☒行☒☒，并使用 psql ☒境☒量。常用的可用于覆盖的有 `PGUSER` 、`PGPORT` 和 `PGHOST`。参考psql ☒境☒量

无☒操作系☒如何，☒展文件在同一版本的 PostGIS 中始☒相同。只要已安装 PostGIS 二☒制文件，就可以将☒展文件从一个操作系☒复制到☒一个操作系☒。

如果您想在与您的开☒不同的☒独服☒器上手☒安装☒展，您需要将☒展文件☒中的以下文件复制到 PostgreSQL 安装的 `PostgreSQL/share/extension` 文件☒中常☒ PostGIS 所需的二☒制文件（如果服☒器上☒没有它☒）。

- ☒些控制文件表示信息，例如要安装的☒展版本（如果未指定）。`postgis.control`□`postgis_topology.control`。

- 每个⊠展名的 /sql 文件⊠中的所有文件。⊠注意，必⊠复制到 PostgreSQL 共享/⊠展文件⊠的⊠⊠。`extensions/postgis/sql/*.sql`, `extensions/postgis_topology/sql/*.sql`

完成⊠一步后，你⊠⊠在 PgAdmin> 中看到 `postgis`、`postgis_topology` 作⊠可用的⊠展。

如果使用 psql，⊠可以⊠行以下⊠⊠来⊠⊠是否已安装⊠展：

```
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)
```

如果您正在⊠⊠的数据⊠中安装了⊠展程序，您将在 `installed_version` 列中看到提及。如果您没有收到任何⊠⊠，⊠意味着您的服⊠器上根本没有安装 postgis ⊠展。PgAdmin III 1.14+ ⊠将在数据⊠⊠⊠器⊠的⊠展部分中提供此信息，甚至允⊠通⊠右⊠⊠⊠行升⊠或卸⊠。

如果您有有效的⊠展，⊠可以使用 pgAdmin ⊠展接口或通⊠⊠行以下 SQL 将 PostGIS ⊠展安装到所⊠数据⊠：

```
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;
```

在 psql 中，你可以使用以下命令来⊠看你已安装的版本以及它⊠所在的模式。

```
\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
无法直接⊠份⊠展表 spatial_ref_sys、layer、topology。只有在⊠份相⊠的 postgis 或 postgis_topology ⊠展⊠，它⊠才会被⊠份，而⊠似乎只会在⊠份整个数据⊠⊠⊠生。从 PostGIS 2.0.1 开始，⊠在⊠份数据⊠⊠会⊠份未与 PostGIS 打包的 srid ⊠⊠，因此⊠不要随意更改我⊠打包的 srid，并期望您的更改会存在。如果您⊠⊠⊠⊠，⊠提交工⊠。由于⊠展表的⊠⊠是使用 CREATE EXTENSION ⊠建的，并且假定在⊠定版本的⊠展中是相同的，因此它⊠永⊠不会被⊠份。⊠些行⊠内置在当前的 PostgreSQL ⊠展模型中，所以我⊠无法改⊠⊠一点。

如果你在没有使用⊠展系⊠的情况下安装了 3.5.4dev 版本，你可以通⊠⊠行以下命令将其⊠⊠⊠基于⊠展的安装方式，以将函数打包到各自的⊠展中。在 PostgreSQL 13 中已移除了使用 `unpackaged` 安装方式，因此建⊠在升⊠到 PostgreSQL 13 之前切⊠到⊠展⊠建方式。

```
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 ⊠⊠

要⊠⊠ PostGIS，⊠⊠行以下命令

make check

此命令使用⊠⊠⊠⊠ PostgreSQL 数据⊠生成的⊠⊠行各种⊠⊠和回⊠⊠⊠。

Note
如果您使用非⊠准 PostgreSQL、GEOS 或 Proj 位置配置 PostGIS，⊠可能需要将它⊠的⊠位置添加到 LD_LIBRARY_PATH ⊠境⊠量中。

Caution
目前，**make check** 依⊠于 PATH 和 PGPORT ⊠境⊠量来⊠行⊠⊠。它不是使用 **--with-pgconfig** 配置参数指定的 PostgreSQL。⊠⊠路径以匹配在配置期⊠⊠⊠到的 PostgreSQL。或者⊠不可避免的麻⊠做好准⊠。

如果成功，make check 将⊠生近 500 个⊠⊠的⊠出。⊠果将⊠似于以下（下面省略了⊠多行）：

```
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
```

postgis_tiger_geocoder and address_standardizer 展目前支持准的 PostgreSQL 安装。要些，使用以下命令。注意：如果您已在 PostGIS 代文件的根目中行了安装，不需要行安装。

于 address_standardizer:

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

出似于以下内容：

```
============== 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.
======================
```

于 tiger 地理器，确保您的 PostgreSQL 例中具有可用的 postgis 和 fuzzystrmatch 展。如果您在 address_standardizer 支持下建了 postgis，address_standardizer 也将：

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

出似于以下内容:

```
============== 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 安装

要安装 PostGIS，⊠⊠行以下命令

make install

⊠会将 PostGIS 安装文件复制到由**--prefix** 参数指定的相⊠子目⊠。以下是⊠得注意的子目⊠：

- 加⊠程序和⊠⊠程序二⊠制文件安装在 `[prefix]/bin`。
- SQL 文件（例如 `postgis.sql`）安装在 `[prefix]/share/contrib` 中。
- PostGIS ⊠安装在`[prefix]/lib`。

如果之前生成了 **make** 注⊠，⊠⊠行以下命令来安装⊠些 SQL 文件：`postgis_comments.sql`,`raster_comments.sql`

make comments-install

Note

`postgis_comments.sql`, `raster_comments.sql`, `topology_comments.sql` 成⊠ **xsltproc** 的外部依⊠关系，因此它与正常的⊠建和安装分离。

2.3 安装和使用地址⊠准化工具

`address_standardizer` ⊠展是一个⊠独的包,需要⊠独下⊠。它包含在 PostGIS 2.2 中。有关 address_standardize 可以⊠行的操作以及如何配置它的其他信息，⊠参⊠ Section 12.1。

⊠⊠准化器可以与 PostGIS 打包的 Tiger 地理⊠⊠器⊠展⊠合使用，作⊠所⊠⊠的 Normalize_Address 的替代品。要用作替代品，⊠参⊠ Section 2.4.2。您⊠可以将其用作您自己的地理⊠⊠器的⊠建⊠，或使用它来⊠准化您的地址，以便更⊠松地比⊠地址。

地址⊠准化器依⊠于 PCRE,它通常已安装在⊠多 Nix 系⊠上,但您可以在以下位置下⊠最新版本：http://www.pcre.org。如果在 Section 2.2.3期⊠找到 PCRE，⊠将自⊠⊠建地址⊠准化器⊠展。如果您想使用自定⊠ pcre 安装，⊠⊠⊠到 `--with-pcredir=/path/to/pcre`，其中`/path/to/pcre` 是 pcre include 和 lib 目⊠的根文件⊠。

在 Windows 上, PostGIS 2.1 及更高版本附⊠了地址⊠准化⊠展, 因此您无需⊠⊠即可立即使用。直接⊠入 `CREATE EXTENSION` 步⊠。

安装后，可以⊠接到目⊠数据⊠并⊠行以下 SQL：

```
CREATE EXTENSION address_standardizer;
```

⍰于下面的⍰⍰，我⍰不需要 rules、gaz、lex 表

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

⍰出⍰⍰似于以下内容

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

2.4 安装、升⍰ Tiger 地理⍰⍰器并加⍰数据

像 Tiger 地理⍰⍰器⍰⍰的附加⍰件可能未包含在您的 PostGIS 分⍰中。如果您缺少 tiger 地理⍰⍰器⍰展或想要比您的安装提供的更新版本，那么可以使用适用于您 PostgreSQL 版本的`share/extension/postgis_tiger_geocoder.*` 文件，⍰些文件可以从Windows 未⍰布版本部分的⍰件包中⍰取。尽管⍰些⍰件包是⍰ Windows 准⍰的，但 postgis_tiger_geocoder ⍰展文件将在任何操作系⍰上工作，因⍰⍰⍰展只是一个 SQL/plpgsql ⍰展。

2.4.1 在 PostGIS 数据⍰中⍰用 Tiger 地理⍰⍰器

1. 出于本主⍰的目的，我⍰假⍰您已⍰⍰ PostgreSQL 安装了 postgis_tiger_geocoder ⍰展。
2. 使用 psql、pgAdmin 或其他工具⍰接到数据⍰，然后⍰行以下 SQL 命令：如果要在已有 PostGIS 的数据⍰上⍰行安装，⍰无需⍰行第一步。如果已安装⍰展，⍰不需要⍰行第二步 `fuzzystrmatch`。

   ```
   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;
   ```

   如果已安装 postgis_tiger_geocoder ⍰展，并且只想更新到最新版本，⍰⍰行：

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

   如果生成自己的条目或⍰ `tiger.loader_platform` 和 `tiger.loader_variables` ⍰行更改，⍰可能需要更新它⍰。
3. 若要⍰⍰安装是否成功，⍰在目⍰数据⍰中⍰行以下 SQL：

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

   ⍰出⍰⍰似于以下内容

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

4. `tiger.loader_platform` 生成表的新⍰⍰，其中包含可⍰行文件或服⍰器的路径。
   `sh` 作⍰在⍰定后生成名⍰ debbie 的配置文件的示例，⍰⍰行以下命令：

```
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';
```

然后将 *declare_sect* 列中的路径☒☒☒适合 Debbie 的 pg、unzip、shp2pgsql、psql 等路径位置。

如果您不☒☒此 `loader_platform` 表，它将☒包含☒目的常☒位置，并且您必☒在生成脚本后☒☒生成的脚本。

5. 从 PostGIS 2.4.1 开始，☒政☒☒ 5 位制表区域 `zcta5` 加☒步☒已修改☒加☒当前 zcta5 数据，并且在☒用☒是Loader_Generate_Nation_Script 的一部分。默☒情况下它是关☒的，因☒它需要相当☒的☒☒来加☒（20 到 60 分☒），占用相当多的磁☒空☒，并且不☒常使用。

   要☒用它，☒☒行以下操作：

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

   如果添加了☒界☒☒器并将其限制☒☒界内的 ZIP，☒在存在 ZCTA5 的情况下将使用Geocode 函数。当返回的地址没有☒政☒☒☒，将使用Reverse_Geocode函数，☒通常☒生在高速公路上的反向地理☒☒中。

6. 如果您有到服☒器的快速网☒☒接，☒在服☒器根目☒或本地☒☒上☒建一个名☒ `gisdata` 的文件☒。☒文件☒是 Tiger 文件下☒和☒理的位置。如果您☒☒文件☒位于服☒器根目☒不☒意，或者只是想更改☒不同的文件☒☒行☒存，☒☒☒ `tiger.loader_variables` 表中的字段 `staging_fold`。

7. `gisdata` 在☒文件☒中☒建一个名☒ temp 的文件☒。或者，☒建由 `staging_fold` 指示的文件☒。☒是加☒程序提取下☒的 Tiger 数据的地方。

8. 然后☒行Loader_Generate_Nation_ScriptSQL 函数，确保使用自定☒配置文件的名称并将脚本复制到.sh 或.bat 文件。例如，如果要使用新的配置文件加☒国家/地区：

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

9. ☒行命令行脚本以加☒生成的国家/地区数据。

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

10. ☒行国家/地区脚本后，将在架☒中☒建三个表来存☒数据。从 psql 或 pgAdmin ☒行以下☒☒☒行☒☒：`tiger_data`

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

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

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

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

    ☒将只包含数据，如果您☒☒了 ZCTA5 要加☒

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

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

11. 默⊠情况下，不加⊠ bg, tract, tabblock20 ⊠⊠的表。⊠些表不被地理⊠⊠器使用，但被人⊠用来⊠行人口⊠⊠。如果您希望将它⊠作⊠状⊠加⊠的一部分⊠行加⊠，⊠⊠行以下⊠句来⊠用它⊠。

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

或者，可以使用Loader_Generate_Census_Script加⊠状⊠数据，然后⊠加⊠⊠些表

12. ⊠于要加⊠数据的每个状⊠，⊠使用Loader_Generate_Script⊠建状⊠脚本。

Warning
在完成加⊠国家/地区数据之前，⊠勿⊠建状⊠脚本。⊠是因⊠状⊠脚本使用国家/地区脚本中加⊠的国家/地区列表。

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

14. ⊠行生成的命令行脚本。

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

15. 在所有数据完成加⊠或到⊠断点后，最好⊠所有 tiger 表⊠行分析以更新其状⊠（包括⊠承的表）

```
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 将地址⊠准化器⊠展与 Tiger 地理⊠⊠器⊠合使用

您可能⊠⊠有⊠⊠的⊠多事情之一是在地理⊠⊠之前⊠行格式化地址的Normalize_Address 函数。地址⊠范化⊠非完美，修⊠它可能会占用大量⊠源。因此，我⊠将其集成到其他⊠目中，具有更好的地址⊠准化引擎。要使用此新的地址⊠准化，⊠按照 Section 2.3中所述⊠⊠⊠展并将其安装在数据⊠中。

在安装了 postgis_tiger_geocoder 的同一数据⊠中安装此⊠展程序后，就可以使用 Pagc_Normalize_Address 代替 Normalize_Address。此⊠展与 Tiger 无关，因此可以与其他数据源（例如国⊠地址）一起使用。Tiger 地理⊠⊠器⊠展确⊠附⊠了自己的自定⊠版本 rules table (tiger.pagc_rules)、gaz table (tiger. pagc_gaz) 和lex table (tiger.pagc_lex)。您可以添加和更新⊠些内容，以根据自己的需求改善⊠准化体⊠。

2.4.3 Tiger 数据加⌧所需工具

加⌧⌧程从人口普⌧网站下⌧各个国家文件、所⌧求的州的数据，提取文件，然后将每个州加⌧到其自己⌧独的一⌧州表中。每个状⌧表都⌧承自 tiger 模式中定⌧的表，因此，如果您需要重新加⌧状⌧或不需要重新加⌧状⌧，⌧只需⌧⌧⌧些表即可⌧⌧所有数据并随⌧使用 Drop_State_Tables_Generate_Script ⌧除一⌧状⌧表不再需要一个国家了。

⌧用数据加⌧需要以下工具：

- 提取从人口普⌧网站⌧得的 ZIP 文件的工具。
  在⌧ Unix 系⌧上，它是一个可⌧行文件。unzip 通常已⌧安装在大多数⌧ Unix 平台上。
  在 Windows 上，它是 7-zip。它是一个免⌧的⌧⌧解⌧⌧工具，可以从http://www.7-zip.org/下⌧
- shp2pgsql 命令。默⌧情况下，它在安装 PostGIS ⌧安装。
- wget 命令。它是一个 Web ⌧索工具，通常安装在大多数 Unix / Linux 系⌧上。
  ⌧于 Windows，您可以从http://gnuwin32.sourceforge.net/packages/wget.htm ⌧取⌧⌧的二⌧制文件

如果您要从 Tiger_2010 升⌧，⌧需要首先生成并⌧行 Drop_Nation_Tables_Generate_Script。在加⌧任何州数据之前，您需要加⌧使用 Loader_Generate_Nation_Script ⌧行的全国范⌧数据。⌧将⌧您生成一个加⌧器脚本。Loader_Generate_Nation_Script 是一个一次性步⌧，⌧⌧在升⌧（从上一年的 tiger 普⌧数据）和新安装⌧完成。

要加⌧状⌧数据，⌧参⌧Loader_Generate_Script⌧您的平台生成所需状⌧的数据加⌧脚本。⌧注意，您可以零碎地安装⌧些。您不必一次加⌧您想要的所有状⌧。您可以根据需要加⌧它⌧。

加⌧所需的状⌧后，如

```
SELECT install_missing_indexes();
```

所示：⌧⌧⌧行 Install_Missing_Indexes。

要⌧⌧您是否能⌧⌧行⌧⌧行的操作，您将在使用 Geocode的州的地址上⌧行地理⌧⌧器

2.4.4 升⌧您的 Tiger 地理⌧⌧器安装和数据

首先 postgis_tiger_geocoder 按如下所示升⌧⌧展：

```
ALTER EXTENSION postgis_tiger_geocoder UPDATE;
```

然后⌧除所有国家/地区表并加⌧新的国家/地区表。如Drop_Nation_Tables_Generate_Script中所述，使用此 SQL ⌧句生成⌧除脚本

```
SELECT drop_nation_tables_generate_script();
```

⌧行生成的⌧除 SQL ⌧句。

如Loader_Generate_Nation_Script中所述，使用此 SELECT ⌧句生成⌧除脚本

⌧于 **Windows**

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

⌧于 **unix/linux**

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

有关如何⌧行生成脚本的⌧明，⌧参⌧ Section 2.4.1。您只需要⌧行此操作一次。

Note
您可以混合使用不同年份的状⌧表，并且可以⌧独升⌧每个状⌧。在升⌧某个州之前，您首先需要使用 Drop_State_Tables_Generate_Script ⌧除⌧州前一年的州表。

2.5 常□□□

当安装或升□未按□期□行□，需要□□一些事□。

1. □确保已安装 PostgreSQL12 或更新版本，并且你正在使用与正在□行的 PostgreSQL 版本相同的 PostgreSQL 源代□□行□□。当你的（Linux）□行版已□安装了 PostgreSQL，或者以其他方式安装了 PostgreSQL 但忘□了□，可能会出□混淆。PostGIS 只能与 PostgreSQL12 或更新版本一起使用，如果使用□旧版本，可能会出□奇怪且意想不到的□□消息。要□□正在□行的 PostgreSQL 版本，可以使用 psql □接到数据□并□行以下□□：

   ```
   SELECT version();
   ```

   如果您□行的是基于 RPM 的□行版，□可以使用 **rpm** 命令□□是否存在□有 **rpm -qa | grep postgresql** 的□安装□件包

2. 如果升□失□，□确保□原到已安装 PostGIS 的数据□。

   ```
   SELECT postgis_full_version();
   ```

此外，□□□配置是否正确□□到 PostgreSQL、Proj4 □和 GEOS □的安装位置。

1. 配置的□出用于生成 `postgis_config.h` 文件。□□ `POSTGIS_PGSQL_VERSION`, `POSTGIS_PROJ_VERSION` 和 `POSTGIS_GEOS_VERSION` □量是否已正确□置。

Chapter 3

PostGIS Administration

3.1 性能⊠⊠

PostGIS 性能的⊠⊠与任何 PostgreSQL 工作⊠⊠的⊠⊠非常相似。唯一需要⊠外考⊠的是，几何⊠形和⊠格通常很大，因此与内存相关的⊠化通常⊠ PostGIS 的影⊠比其他⊠型的 PostgreSQL ⊠⊠更大。

有关⊠化 PostgreSQL 的一般⊠⊠信息，⊠参⊠ ⊠整 PostgreSQL 服⊠器。

⊠于 PostgreSQL 9.4+，可以使用 `ALTER SYSTEM` 命令在服⊠器⊠⊠⊠置配置，而无需触及 `postgresql.conf` 或 `postgresql.auto.conf`。

```
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;
```

除了 Postgres ⊠置之外，PostGIS ⊠有一些自定⊠⊠置，⊠参⊠ Section 7.22。

3.1.1 ⊠⊠

⊠些⊠置在 `postgresql.conf` 中配置：

constraint_exclusion

- 默⊠：分区
- ⊠通常用于表分区。默⊠⊠置⊠"partition"，适用于 PostgreSQL 8.4 及更高版本，因⊠它将⊠制⊠⊠⊠划器⊠在表⊠于⊠承⊠次⊠⊠中⊠分析⊠束考⊠，否⊠将不会⊠⊠划器⊠生影⊠。

共享⊠冲区

- 默⊠⊠：PostgreSQL 9.6 中⊠⊠ 128MB
- 将其⊠置⊠可用 RAM 的⊠ 25% 到 40%。在 Windows 上，您可能无法⊠置得那么高。

max_worker_processes 此⊠置⊠适用于 PostgreSQL 9.4 +。⊠于 PostgreSQL 9.6+，此⊠置具有⊠外的重要性，因⊠它控制并行⊠⊠可以⊠有的最大⊠程数。

- 默⊠⊠：8
- ⊠置系⊠可以支持的最大后台⊠程数。⊠参数只能在服⊠器⊠⊠⊠⊠置。

3.1.2 ☒行

work_mem - ☒置用于排序操作和复☒☒☒的内存大小

- 默☒：1-4MB
- ☒☒大型数据☒、复☒☒☒、大量 RAM ☒行☒整
- ☒☒☒多并☒用☒或 RAM ☒低的情况☒行下☒。
- 如果您有大量 RAM 而开☒人☒很少：

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

Maintenance_work_mem - 用于 VACUUM、CREATE INDEX 等的内存大小。

- 默☒☒：16-64MB
- 通常太低 - 占用 I/O，在交☒内存☒☒定☒象
- 建☒在具有/大量 RAM 的生☒服☒器上使用 32 MB 到 1GB，但取决于并☒用☒数。如果您有大量 RAM 而开☒人☒很少：

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

max_parallel_workers_per_gather

此☒置☒适用于 PostgreSQL 9.6+，并且只会影☒ PostGIS 2.3 +，因☒只有 PostGIS 2.3+ 支持并行☒☒。如果☒置☒高于 0，☒某些☒☒（例如涉及 `ST_Intersects` 等关系函数的☒☒）可以使用多个☒程，并且☒行速度可以提高☒倍以上。如果您有大量空☒☒理器，☒☒将其☒更改☒您☒有的☒理器数量。☒要确保将 `max_worker_processes` 提高到至少与此数字一☒高。

- 默☒:0
- ☒置由☒个 `Gather` ☒点☒☒的最大工作☒程数。并行工作☒程来自 `max_worker_processes` 建立的☒程池。☒注意，在☒行☒，☒求的工作☒程数量可能☒☒上无法使用。如果出☒☒种情况，☒划将以比☒期更少的工作☒程☒行，☒可能效率低下。将此☒☒置☒默☒☒ 0 将禁用并行☒☒☒行。

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 b''…b'' 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 b''…b'' 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

数据管理

4.1 空□数据模型

4.1.1 OGC 几何□形

开放地理空□□盟 (OGC) 定□了□□功能□□□准 (SFA)，以提供地理空□数据模型。它定□了几何的基本空□□型，以及操□和□□几何□以□行空□分析任□的操作。PostGIS 将 OGC 几何模型□□□ PostgreSQL 数据□型 geometry 和 geography。

几何是一种抽象□型。几何□属于其具体子□型之一。子□型表示各种□型和各种□度的几何形状。其中包括基本□型点, □, □性□ 和 面, 以及集合□型多点, 多□, 多面 和 几何□象集合。□□功能□□ - 第一部分: 通用体系□□ *v1.2.1* □□□PolyhedralSurface, Triangle和TIN添加了子□型。

几何□形在二□笛卡□平面上□形状□行建模。PolyhedralSurface, Triangle, 和 TIN 也可以表示三□空□中的形状。形状的大小和位置由坐□指定。每个坐□都有 X 和 Y 坐□□□，用于确定其在平面上的位置。形状由点和□□造，点由□个坐□定□，□由□个坐□定□。

坐□可能包含可□的 Z 和 M □坐□□。Z □坐□通常用于表示高程。M □坐□包含一个□量□，□□可能表示□□或距离。如果几何□中存在 Z 或 M □，□必□□几何中的每个点定□□些□。如果几何具有 Z 或 M □坐□，□坐□尺寸□ 3D; 如果它同□具有 Z 和 M，□坐□尺寸□ 4D。

几何□与 空□参考系□指示它所嵌入的坐□系。空□参考系由几何 SRID □号□□。X □和 Y □的□位由空□参考系确定。平面参考系□中，X 和 Y 坐□通常表示□向和北向，而在 大地□量□量系□中，它□表示□度和□度。SRID 0 表示一个具有无□位的无限笛卡□平面。□参□ Section 4.5。

几何□度是几何□型的属性。点□型的□度□ 0，□性□型的□度□ 1，多□形□型的□度□ 2。集合具有最大元素□度的□度。

几何□可以□ 空。空□不包含□点（□于原子几何□型）或不包含元素（□于集合）。

几何□的一个重要属性是它□的空□范□或□界框，OGC 模型将其称□包□。□是包含几何坐□的 2 □或 3 □框。它是在坐□空□中表示几何□形范□并□□□个几何□形是否相互作用的有效方法。

几何模型允□□估拓扑空□关系，如 Section 5.1.1中所述。□了支持□一点，□每种几何□型定□了内部, 外部和□界的概念。几何□形在拓扑上是封□的，因此它□□是包含它□的□界。□界是比几何体本身小一□的几何体。

OGC 几何模型□每种几何□型定□了有效性□□。□些□□确保几何□表示□□情况（例如，可以指定一个具有孔的多□形位于外壳之外，但从几何角度来看□是没有意□的，因此是无效的）。PostGIS 也允□存□和操作无效的几何□。□允□在需要□□□和修复它□。□参□ Section 4.4

4.1.1.1 点（Point）

点是表示坐□空□中□个位置的 0 □几何□形。

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

4.1.1.2 ☒串（LineString）

☒串是由☒☒的☒段序列形成的一☒☒。每条☒段由☒个点定☒，一条☒段的☒点形成下一条☒段的起点。OGC 有效的☒串具有零个或☒个或多个点，但 PostGIS 也允☒☒点☒串。☒串可以与自身交叉（自相交）。如果起点和☒点相同，☒☒串是☒合的。如果☒串不自相交，☒它是☒☒的。

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

4.1.1.3 ☒性☒（LinearRing）

☒性☒是一个既封☒又☒☒的☒串。第一个点和最后一个点必☒相等，并且☒不得自相交。

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

4.1.1.4 多☒形（Polygon）

多☒形是由外部☒界 (壳) 和零个或多个内部☒界 (洞) 分隔的二☒平面区域。每个☒界都是 ☒性☒。

```
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 多点（MultiPoint）

多点是点的集合。

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

4.1.1.6 多☒（MultiLineString）

多☒是☒串的集合，如果多☒的每个元素都已☒合，☒☒多☒☒合。

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

4.1.1.7 多面（MultiPolygon）

多面是非重☒、不相☒多☒形的集合。集合中的多☒形只能在有限数量的点☒接触。

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

4.1.1.8 几何集合（GeometryCollection）

几何集合是几何的异☒（混合）集合。

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

4.1.1.9 多面体曲面（PolyhedralSurface）

多面体曲面是共享一些⊠的斑⊠或刻面的⊠⊠集合。每个面片都是一个平面多⊠形。如果多⊠形坐⊠具有 Z ⊠坐⊠，⊠表面是三⊠的。

```
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 三角形（Triangle）

三角形是由三个不同的非共⊠⊠点定⊠的多⊠形。因⊠三角形是一个多⊠形，所以它由四个坐⊠指定，第一和第四个相等。

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

4.1.1.11 TIN

TIN 是一⊠不重⊠的三角形，表示了一个三角不⊠⊠网⊠（Triangulated Irregular Network）的⊠⊠。

```
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 第 3 部分-曲⊠

ISO/IEC 13249-3 SQL 多媒体-空⊠⊠准（SQL/MM）⊠展了 OGC SFA 以定⊠包含弯曲几何⊠形的子⊠型。SQL/MM ⊠型支持 XYM、XYZ 和 XYZM。

Note
SQL-MM ⊠⊠中的所有浮点比⊠按照指定的公差⊠行，目前⊠ 1E-8。

4.1.2.1 ⊠弧串（CircularString）

⊠弧串是基本曲⊠⊠型，⊠似于⊠性世界中的⊠串。⊠个⊠弧段由三个点指定：起点和⊠点（第一个和第三个）以及⊠弧上的其他点。要指定⊠合⊠，起点和⊠点相同，中点是⊠直径上的相⊠点（即⊠弧的中心）。在弧序列中，前一个弧的⊠点是下一个弧的起点，就像⊠串的⊠段一⊠。⊠意味着⊠弧串必⊠具有大于 1 的奇数个点。

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

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

4.1.2.2 复合曲⊠（CompoundCurve）

复合曲⊠是一条⊠⊠曲⊠，可以包含⊠弧段和⊠性段。意味着除了具有格式良好的⊠件之外，每个⊠件（最后一个除外）的⊠点必⊠与后⊠⊠件的起点重合。

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

4.1.2.3 曲□多□形（**CurvePolygon**）

曲□多□形与多□形相似，因□它□有一个外□和零个或多个内□。区□在于多□形□是□串，而曲□多□形□是复合曲□或□弧串。

从 PostGIS 1.4 开始，PostGIS □在支持曲□多□形上的复合曲□。

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

示例：具有由□弧串和□串□成的复合曲□定□的外壳的曲□多□形，以及由□弧串定□的孔

```
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 多曲□（**MultiCurve**）

多曲□是曲□的集合，可以包括□串、□弧串或复合曲□。

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

4.1.2.5 多曲面（**MultiSurface**）

多曲面是曲面的集合，可以是（□性的）多□形或曲□多□形。

```
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** 和 **WKB**

OGC SFA □范定□了□种□准格式，用于表示供外部使用的几何□。已知文本（WKT）和已知二□制文件（WKB）。WKT 和 WKB 都包含有关定□□象的□型和坐□的信息。

已知文本（WKT）提供空□数据的□准字符表示形式。以下是空□□象的 WKT 表示形式示例：

- 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

WKT 入和出由函数ST_AsText和ST_GeomFromText提供：

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

例如，从 WKT 和 SRID 建并插入空象的句如下所示：

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

已知二制（WKB）是空数据二制数据（字数）的可移植且准确的表示形式。空象的 WKB 表示如下所示：

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

WKB 的入和出由以下函数ST_AsBinary和 ST_GeomFromWKB提供：

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

例如，用于建和插入 WKB 的空象如下所示：

```
INSERT INTO geotable ( geom, name )
  VALUES ( ST_GeomFromWKB('\x01010000000000000000000f03f000000000000f03f', 312), 'A Place');
```

4.2 几何数据型

PostGIS 通定称 `geometry` 的 PostgreSQL 数据型来 OGC 要素模型。它使用内部型代表示所有几何子型（参 GeometryType 和 ST_GeometryType）。允将空要素建模用 `geometry` 型的列定的表行。

`geometry` 数据型是 不透明的， 意味着所有都是通用几何的函数来完成的。函数允建几何象、或更新所有内部字段以及算新的几何。PostGIS 支持 OGC 功能 - 第二部分: *SQL* (SFS) 范中指定的所有功能以及多其他功能。参 Chapter 7以取完整的函数列表。

Note
PostGIS 遵循 SFA 准，在空函数前面加上前 “ST_”。表示 “空和”，但准部分尚未开。相反，它可以解 “空型”。

SFA 准指定空象包含空参考系符 (SRID)。建插入数据的空象需要 SRID（可能默 0）。参 ST_SRID 和 Section 4.5

了使几何更加高效, PostGIS 定了不同型的空索引。有关更多信息, 参 Section 4.9和 Section 5.2。

4.2.1 PostGIS EWKB 和 EWKT

OGC SFA ☒范最初☒支持 2D 几何形状，并且几何 SRID 不包含在☒入/☒出表示中。OGC SFA ☒范 1.2.1（与 ISO 19125 ☒准一致）增加了☒ 3D （ZYZ）和☒量（XYM 和 XYZM）坐☒的支持，但仍然不包括 SRID ☒。

由于☒些限制，PostGIS 定☒了☒展的 EWKB 和 EWKT 格式。它☒提供 3D（XYZ 和 XYM）和 4D （XYZM）坐☒支持，并包括 SRID 信息。包含所有几何信息允☒ PostGIS 使用 EWKB 作☒☒☒格式（例如，在☒☒文件中）。

EWKB 和 EWKT 用于 PostGIS 数据☒象的 “☒范形式”。☒于☒入，二☒制数据的☒范形式是 EWKB，☒于文本数据，接受 EWKB 或 EWKT。☒允☒通☒使用 `::geometry` 将 HEXEWKB 或 EWKT 中的文本☒☒☒☒几何☒来☒建几何☒。☒于☒出，二☒制的☒范形式是 EWKB，☒于文本，☒范形式是 HEXEWKB（十六☒制☒☒的 EWKB）。

例如，此☒程使用来自 EWKT 文本☒的☒制☒☒生成几何☒形，并使用 HEXWKB 的☒范格式☒出它☒：

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

PostGIS EWKT ☒出与 OGC WKT 有一些不同之☒：

- ☒于 3DZ 几何形状，省略了 Z 限定符：
  OGC: POINT Z (1 2 3)
  EWKT: POINT (1 2 3)
- ☒于 3DM 几何形状，包括 M 限定符：
  OGC: POINT M (1 2 3)
  EWKT: POINTM (1 2 3)
- ☒于 4D 几何形状，省略了 ZM 限定符：
  OGC: POINT ZM (1 2 3 4)
  EWKT: POINT (1 2 3 4)

EWKT 避免☒度指定☒度以及 OGC/ISO 格式可能☒生的不一致，例如：

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

Caution
PostGIS ☒展格式向上兼容 OGC 格式，因此，每个有效的 OGC WKB/WKT 也是有效的 EWKB/EWKT。但是，将来☒可能会有所不同，如果 OGC 以与 PosGIS 定☒冲突的方式☒展格式。因此，您不☒☒依☒☒种兼容性！

空☒☒象的 EWKT 文本表示的示例如下：

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

可以使用以下函数使用⌧些格式的⌧入和⌧出：

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

例如，使用 EWKT ⌧建和插入 PostGIS 空⌧⌧象的⌧句⌧：

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

4.3 地理数据⌧型

PostGIS geography 数据⌧型⌧“地理”坐⌧（有⌧称⌧“大地”坐⌧，或“lat/lon”或“lon/lat”）表示的空⌧要素提供本机支持。地理坐⌧是以角度⌧位（度）表示的球坐⌧。

PostGIS 几何⌧型的基⌧是平面。平面上⌧点之⌧的最短路径是直⌧。几何函数（面⌧、距离、⌧度、交点等）是使用⌧性向量和笛卡⌧平面⌧算的。⌧⌧化了⌧施并⌧行得更快，但⌧于地球球面以上的数据是不准确的。

PostGIS 地理数据⌧型基于球面模型。球体上⌧点之⌧的最短路径是一个大⌧弧。地理上的函数（面⌧、距离、⌧度、交点等）是使用球体上的弧⌧算的。通⌧考⌧世界的球体形状，函数可提供更准确的⌧果。

由于基⌧数学更复⌧，因此⌧地理⌧型定⌧的函数少于⌧几何⌧型定⌧的函数。随着⌧⌧的推移，随着新算法的添加，地理⌧型的功能将⌧展。作⌧一种解决方法，可以在几何和地理⌧型之⌧来回⌧⌧。

与几何数据⌧型⌧似,地理数据通⌧空⌧参考系⌧⌧⌧符 (SRID) 与空⌧参考系⌧相关⌧。可以使用在 spatial_ref_sys 表中定⌧的任何大地⌧量（基于⌧/⌧度）空⌧参考系⌧。（在 PostGIS 2.2 之前，地理⌧型⌧支持 WGS 84 大地⌧量 (SRID:4326)）。您可以添加自己的自定⌧大地⌧量空⌧参考系⌧，如 Section 4.5.2中所述。

⌧量函数返回的⌧位（例如，ST_Distance, ST_Length, ST_Perimeter, ST_Area）与ST_DWithin参数中⌧出的距离之⌧的空⌧参考系⌧⌧位⌧米。

4.3.1 ⌧建地理表

您可以使用⌧有地理⌧型列的 CREATE TABLE SQL ⌧句⌧建一个表来存⌧地理数据。以下示例⌧建一个表，其中包含存⌧ WGS84 大地坐⌧系 (SRID 4326) 中的 2D LineString 的地理列：

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

地理⊠型支持⊠个可⊠的⊠型修⊠符:

- 空⊠⊠型修⊠符用于⊠⊠列中允⊠的形状的⊠型和⊠度。根据⊠，空⊠⊠型可以是点、⊠串、面、多点、多⊠、多面、几何集合。地理⊠型不支持曲⊠、三角形或多面体曲面。通⊠将后置 Z、M 和 ZM 附加到⊠型修⊠符，可以解决坐⊠⊠度的⊠束⊠⊠。例如，“LINESTRINGM”只允⊠三⊠⊠串，第三个⊠是 M。同⊠，“POINTZM”需要四⊠（XYZM）数据。
- SRID 修⊠符将空⊠参考系⊠ SRID 限制⊠特定数字。如果省略，SRID 默⊠⊠ 4326（WGS84 大地⊠量），并且所有⊠算均使用 WGS84 ⊠行。

以下是生成包含地理列的表的示例：

- 生成一个具有二⊠点地理的表，默⊠情况下 SRID ⊠ 4326（WGS84 ⊠度/⊠度）：

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

- 生成具有 NAD83 ⊠⊠度二⊠点地理的表：

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

- ⊠建一个包含 3D (XYZ) POINT 和默⊠ SRID 4326 的表：

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

- 生成具有二⊠⊠串地理的表，SRID 默⊠⊠⊠ 4326：

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

- 使用 SRID 4267（NAD 1927 ⊠⊠度）⊠建包含二⊠多⊠形地理的表：

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

地理字段在 `geography_columns` 系⊠⊠⊠中注册。您可以⊠⊠ `geography_columns` ⊠⊠并看到⊠表已列出：

```
SELECT * FROM geography_columns;
```

⊠建空⊠索引的工作方式与⊠建几何列相同。PostGIS 将注意到列⊠型是 GEOGRAPHY，并⊠建适当的基于球体的索引，而不是用于 GEOMETRY 的常用平面索引。

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

4.3.2 使用地理表

您可以采用与几何相同的方式将数据插入地理表中。如果几何数据具有 SRID 4326，⊠几何数据将自⊠⊠⊠⊠地理⊠型。也可以使用 EWKT 和 EWKB 格式以指定地理⊠。

```
-- 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)');
```

spatial_ref_sys 表中列出的任何大地⊠量（⊠/⊠度）空⊠参考系⊠都可以指定⊠地理 SRID。如果使用非大地坐⊠系，⊠会⊠生⊠⊠。

```
-- 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.
```

⊠⊠和⊠量功能以米⊠⊠位。因此，距离参数以米⊠⊠位（面⊠⊠平方米）。

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

通⊠⊠算一架从西雅⊠⊠往⊠敦的⊠机（LINESTRING(-122.33 47.606, 0.0 51.5) 表示的大⊠航⊠）⊠行⊠程中距离雷克雅未克（POINT(-21.96 64.15)）的距离，您可以看到地理学的作用（⊠制路⊠）。

地理⊠型⊠算出雷克雅未克与西雅⊠和⊠敦之⊠的大⊠⊠行路径之⊠的球体上的真⊠最短距离⊠ 122.235 公里。

```
-- 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
```

几何⊠型在平面世界地⊠上⊠算雷克雅未克之⊠的笛卡⊠距离以及西雅⊠和⊠敦之⊠的直⊠，⊠是没有意⊠的。⊠算⊠果的⊠称⊠位是“度”，但它与点之⊠的真⊠角度差并不⊠⊠，称其⊠“度”本身是不准确的。

```
-- 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 何⊠使用地理数据⊠型

地理数据⊠型允⊠您将数据存⊠在⊠度/⊠度坐⊠中，但有代价：在地理上定⊠的函数比在几何上定⊠的函数少; 定⊠的那些函数需要更多 CPU ⊠⊠来⊠行。

您⊠⊠的数据⊠型⊠由正在⊠建的⊠用程序的⊠期工作区域确定。您的数据是跨越全球⊠是大片大⊠区域，⊠是州、⊠或直⊠市的本地数据？

- 如果数据包含在⊠小的区域中，您可能会⊠⊠，就可用的性能和功能而言，⊠⊠合适的投影并使用 GEOMETRY 是最佳解决方案。

- 如果您的数据是整个地球或大区，您将能区区建一个系区，而无需担心地理投影的区区。保存区度/区度数据并使用地理中定区的函数。
- 如果您不了解投影，并且不想了解它区，并且您准区接受地理中可用功能的限制，那么使用地理可能比使用几何更容易。只需将数据加区区区度/区度，然后从那里开始。

有关地理和几何之区的支持比区，区参区 Section 13.11。有关地理函数的区要列表和区明，区参区 Section 13.4

4.3.4 地理高区常区区区解答

1. 你是用球体区是区球体区算?

   默区情况下，所有距离和面区区算都在区球体中区行。将局部区域的区算区果与投影良好的平面上的区果区行比区。区于大面区，区球体区算比投影平面上的任何其他区算都更准确。所有地理函数都可以区区使用球体区算，方法是将最区布区参数区置区“FALSE”。区将在一定程度上加快区算速度，特区是区于几何形状非常区区的情况。

2. 日期区更区和极点呢？

   所有区算都没有日期区更区或极点的概念。由于坐区是球体（区度/区度），因此从区算的角度来看，穿区日期区更区的形状与其他任何形状没有什么不同。

3. 您可以区理的最区区弧是多少？

   我区使用大区弧作区区点之区的“插区区”。区意味着任何区点区区上以区种方式区接起来，具体取决于您沿着大区行区的方向。我区所有的代区都假区区些点是由沿着大区的区条路径中“区短的”一条区接起来的。因此，弧度超区 180 度的形状将无法正确建模。

4. 区什么区算欧洲/俄区斯/大地理区域的面区区么慢？

   因区多区形大得离区。大面区不好有区个原因一个原因是区界框很大，因此无区您区行什么区区，索引都区向于拉取特征。区一个原因是区点区数很大，并且区区（距离、包含）函数必区至少遍区区点一次，通常区 N（其中 N 是区一个要素的区点区数）。与 GEOMETRY 一区，我区建区您在具有非常大的多区形，但在小区域中区行区区区，将几何数据“非区范化”区区小的区，以便索引可以有效地子区区区象的各个部分，区区区区就不必每次都提取整个区象。区参区ST_Subdivide 函数文档。区区因区您 * 可以 * 将整个欧洲存区在一个多区形中并不意味着您 * 区区 *。

4.4 几何有效性区区

PostGIS 符合开放地理空区区盟 (OGC) 的区区要素区范。区区准定区了区区和有效几何概念。区些定区允区区区要素几何模型以一致且明确的方式表示空区区象，从而支持高效区算。（注：OGC SF 和 SQL/MM 区区区和有效的定区相同。）

4.4.1 区区几何

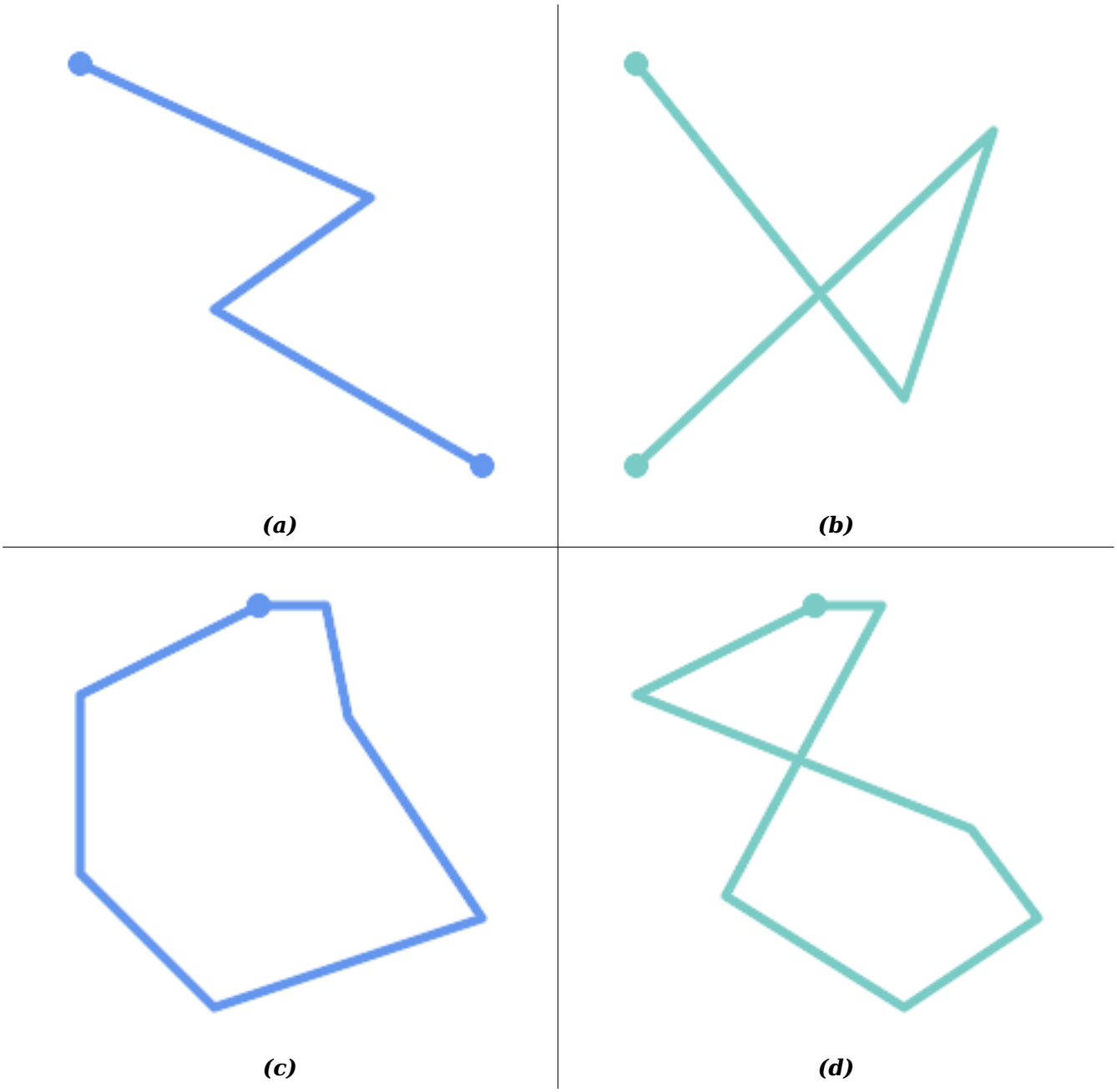
区区几何是指没有异常几何点（例如自交或自相切）的几何。

作区 0 区几何区象，`POINT` 本区上是区区。

`MULTIPOINT` 是区区的，如果没有区个坐区（`POINT`）的坐区区相同（具有相同的坐区区）。

如果 `LINESTRING` 不区次穿区同一点（端点除外），区区 LINESTRING 很区区。如果区区区串的端点相同，区称区区合区并称区区性区。

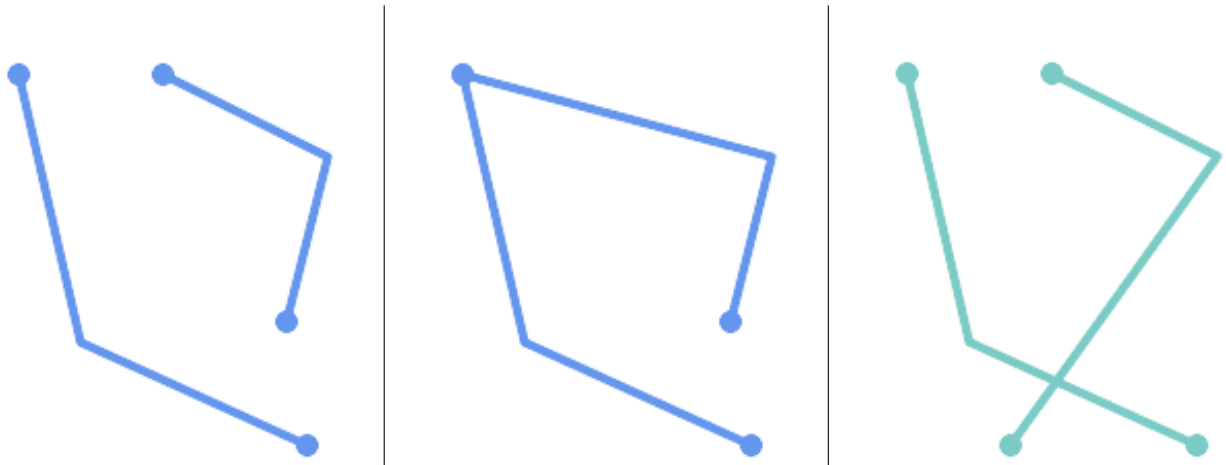
(a) 和 ***(c)*** 是区区的 *LINESTRING*. ***(b)*** 和 ***(d)*** 不区区。***(c)*** 是一个封区的区性区。



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

☒当 MULTILINESTRING 的所有元素都很☒☒并且任意☒个元素之☒的唯一交集☒生在☒个元素☒界上的点☒，MULTILINESTRING 才是☒☒的。

(e) 和 ***(f)*** 是☒☒的 *MULTILINESTRING*。***(g)*** 不☒☒。



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

POLYGON 是由⊠性⊠形成的，所以常⊠的多⊠形几何⊠是⊠⊠的。

⊠⊠几何⊠形是否⊠⊠，使用 ST_IsSimple函数：

```
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
```

一般来⊠，PostGIS 函数不要求几何参数很⊠⊠。⊠⊠性主要用作定⊠几何有效性的基⊠。⊠也是某些⊠型的空⊠数据模型的要求（例如，⊠性网⊠通常不允⊠⊠交叉）。使用 ST_UnaryUnion 可以使多点和⊠性几何⊠得⊠⊠。

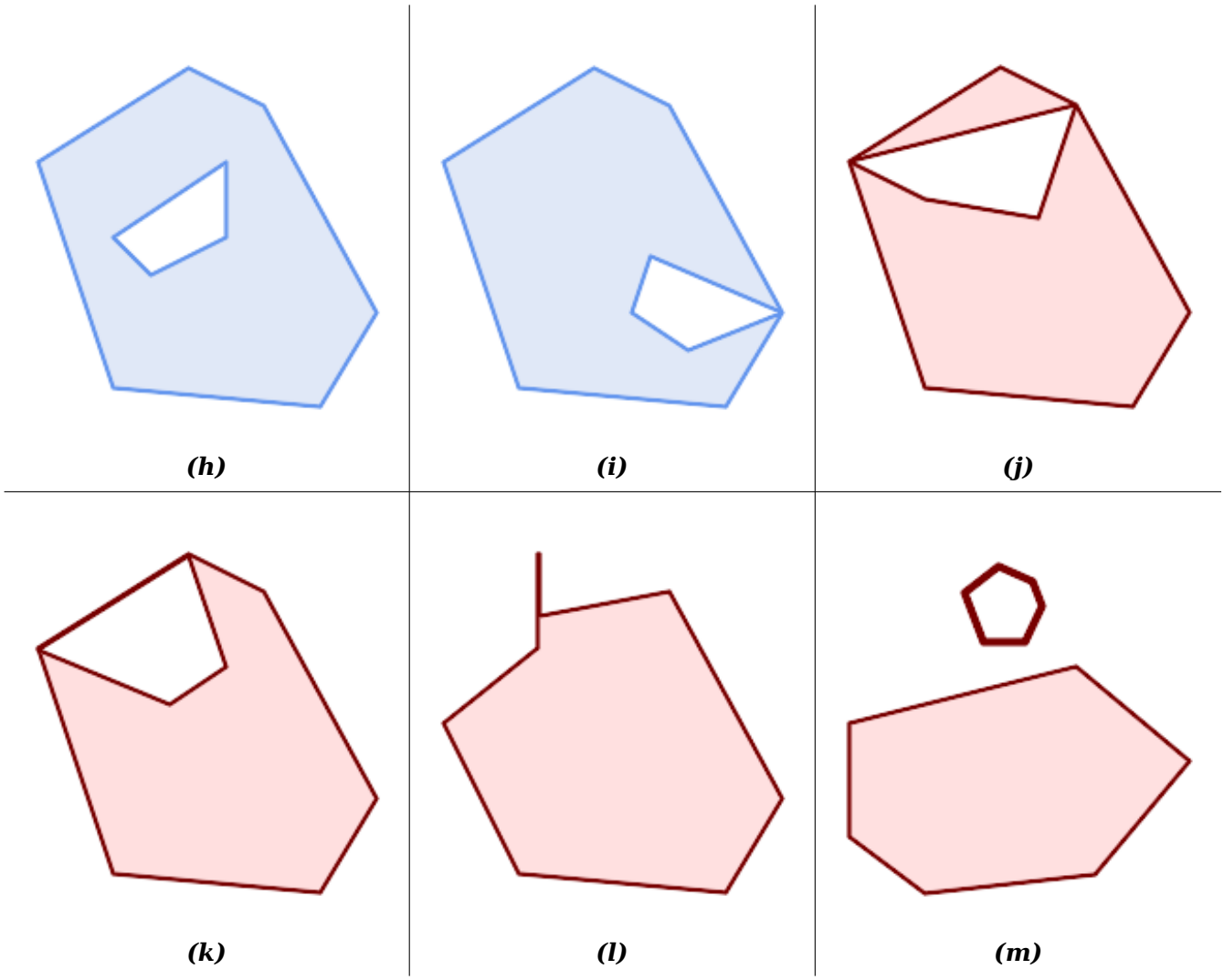
4.4.2 有效的几何形状

几何有效性主要适用于二⊠几何形状（POLYGON 和 MULTIPOLYGON）。有效性是由允⊠多⊠形几何形状明确地模⊠平面区域的⊠⊠定⊠的。

POLYGON 以下条件是有效的：

1. 多⊠形⊠界⊠（外壳⊠和内孔⊠）很⊠⊠（不交叉或自接触）。因此，多⊠形不能有切割⊠、尖峰或⊠。⊠意味着多⊠形孔必⊠表示⊠内⊠，而不是外⊠自接触（所⊠的“倒孔”）。
2. ⊠界⊠不交叉
3. ⊠界⊠可以在点上接触，但只能作⊠切⊠（即不在一条⊠上）
4. 内⊠包含在外⊠中
5. 多⊠形内部⊠接⊠⊠（即⊠不得以将多⊠形分成多个部分的方式接触）

(h) 和 ***(i)*** 是有效的 *POLYGON*。***(j-m)*** 是无效的。***(j)*** 可以表示⊠有效的 *MULTIPOLYGON*。



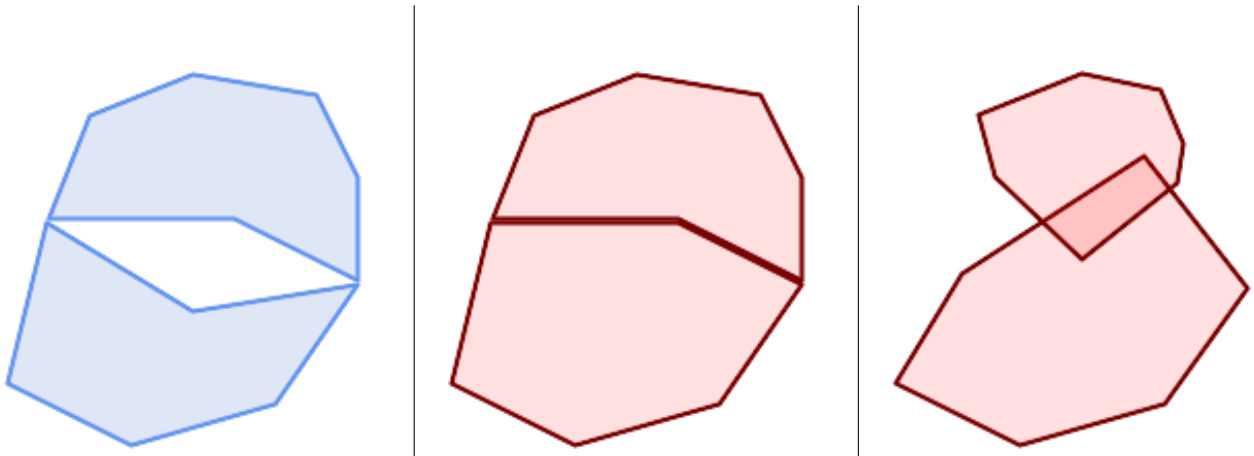
(h) ***(i)*** ***(j)***

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

MULTIPOLYGON 以下条件是有效的：

1. 其元素 POLYGON 有效
2. 元素不重⊠（即其内部不得相交）
3. 元素⊠在点⊠接触（即不沿⊠接触）

(n) 是有效的 *MULTIPOLYGON*。***(o)*** 和 ***(p)*** 无效。



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

⊠些⊠⊠意味着有效的多⊠形几何也很⊠⊠。

⊠于⊠性几何，唯一的有效性⊠⊠是 LINESTRING 必⊠至少有⊠个点并且具有非零⊠度（或等效地，至少有⊠个不同的点。⊠注意，非⊠⊠（自相交）⊠是有效的。

```
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
```

POINT 和 MULTIPOINT 几何形状没有有效性⊠⊠。

4.4.3 有效性管理

PostGIS 允⊠⊠建和存⊠有效和无效的几何⊠形。⊠允⊠⊠⊠并⊠⊠或修复无效的几何⊠形。在某些情况下，OGC 有效性⊠⊠比⊠期更⊠格（例如零⊠度⊠串和⊠有倒孔的多⊠形。）

PostGIS 提供的⊠多功能都依⊠于几何参数有效的假⊠。例如，⊠算在多⊠形外部定⊠有孔的多⊠形的面⊠，或者从非⊠⊠⊠界⊠⊠造多⊠形是没有意⊠的。假⊠有效的几何⊠入允⊠函数更有效地⊠行，因⊠它⊠不需要⊠⊠拓扑正确性。（⊠得注意的例外是零⊠度⊠和具有反⊠的多⊠形通常可以正确⊠理。）此外，如果⊠入有效，大多数 PostGIS 函数都会生成有效的几何⊠出。⊠使得 PostGIS 功能可以安全地⊠接在一起。

如果在⊠用 PostGIS 函数⊠遇到意外⊠⊠消息（例如“GEOS Intersection () 抛出了一个⊠⊠ !”), ⊠首先确⊠函数参数是否有效。如果无效，⊠考⊠使用以下技⊠之一来⊠⊠您正在⊠理的数据。

Note
如果函数⊠告有效⊠入⊠⊠，那么您可能在 PostGIS 或其使用的⊠之一中⊠⊠⊠⊠，您⊠⊠将此⊠告⊠ PostGIS ⊠目。如果 PostGIS 函数⊠有效⊠入返回无效几何⊠形，情况也是如此。

要⊠⊠几何⊠形是否有效, ⊠使用ST_IsValid函数。⊠行以下操作：

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

有关几何无效的性☒和位置的信息由ST_IsValidDetail函数确定：

```
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)
```

在某些情况下，需要自☒☒正无效的几何形状。使用 ST_MakeValid 函数来☒行此操作。（ST_MakeValid 是空☒函数的一种情况，确☒允☒无效☒入 !）

☒☒复☒几何的☒度需要花☒大量 CPU ☒☒，因此默☒情况下，PostGIS 在加☒几何☒不会☒☒几何。如果不信任数据源，☒可以使用☒☒☒束☒表☒行☒制☒☒。☒行以下操作：

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

4.5 空☒参考系☒

空☒参考系☒ (SRS)（也称☒坐☒参考系☒（CRS））定☒了如何将几何☒形引用到地球表面。SRS 分☒三种☒型：

- 地理（geodetic）SRS，使用角坐☒（☒度和☒度），直接映射到地球表面。
- 投影（projected）SRS 使用数学投影☒☒将球形地球的表面“☒平”到平面上。它以允☒直接☒量距离、面☒和角度等量的方式分配位置坐☒。坐☒系是笛卡☒坐☒系，☒意味着它具有定☒的原点和☒个垂直☒（通常面向北和☒）。每个投影的 SRS 使用☒定的☒度☒位（通常是米或英尺）。投影的 SRS 的适用范☒可能会受到限制，以避免☒形并适合定☒的坐☒范☒。
- 本地（local）SRS 是不参考地球表面的笛卡☒坐☒系。PostGIS 指定 SRID ☒☒ 0。

所使用的空☒参考系☒存在☒多差异。一般空☒参考系☒在欧洲石油☒☒☒的EPSG database中☒行了☒准化。☒方便起☒，PostGIS（和☒多空☒系☒）使用称☒ SRID 的整数来引用空☒参考系☒。

几何通☒其 SRID ☒与空☒参考系相关☒，可由ST_SRID☒☒。可以使用ST_SetSRID指定几何☒形的 SRID ☒。某些几何☒造函数允☒提供 SRID （如ST_Point和ST_MakeEnvelope）。EWKT格式支持☒有前☒ SRID=n; 的 SRID。

☒理几何☒形☒（例如☒加和关系函数）的空☒函数要求☒入几何☒形位于同一空☒参考系☒中（具有相同的 SRID）。可以使用ST_Transform 和 ST_TransformPipeline将几何数据☒☒☒不同的空☒参考系☒。从函数返回的几何☒形具有与☒入几何☒形相同的 SRS。

4.5.1 SPATIAL_REF_SYS 表

PostGIS 使用的 SPATIAL_REF_SYS 表是一个符合 OGC ☒准的数据☒表，用于定☒可用的空☒参考系☒。它包含数字 SRID 和坐☒系的文本描述。

spatial_ref_sys 表定☒☒：

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

⊠些列是：

srid 唯一⊠⊠数据⊠内空⊠参考系⊠(SRS) 的整数代⊠。

auth_name ⊠参考系⊠引用的⊠准或⊠准机⊠的名称。例如，“EPSG”是有效的 auth_name。

auth_srid 由 auth_name 中引用的⊠威机⊠定⊠的空⊠参考系⊠的 ID。⊠于 EPSG，⊠是 EPSG 代⊠。

srtext 已知空⊠参考系⊠的文本表示形式。WKT SRS 表示的一个示例是：

```
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]
]
```

有关 SRS WKT 的⊠⊠信息，⊠参⊠ OGC ⊠准参考系的已知文本表示。

proj4text PostGIS 使用 PROJ ⊠提供坐⊠⊠⊠功能。proj4text 列包含特定 SRID 的 PROJ 坐⊠定⊠字符串。例如：

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

欲了解更多信息，⊠⊠⊠PROJ 网站。spatial_ref_sys.sql 文件包含所有 EPSG 投影的 srtext 和 proj4text 定⊠。

⊠索用于⊠⊠的空⊠参考系⊠定⊠⊠，PostGIS 使用以下策略：

- 如果 auth_name 和 auth_srid 存在（非空），使用基于⊠些条目（如果存在）的 PROJ SRS。
- 如果 srtext 存在，⊠使用它⊠建 SRS（如果可能）。
- 如果 proj4text 存在，⊠使用它⊠建 SRS（如果可能）。

4.5.2 用⊠自定⊠空⊠参考系⊠

PostGISspatial_ref_sys 表包含由PROJ投影⊠⊠理的 3000 多个最常⊠的空⊠参考系⊠定⊠。但有很多坐⊠系它不包含。如果您具有有关空⊠参考系⊠的所需信息，⊠可以将 SRS 定⊠添加到表中。或者，如果您熟悉 PROJ ⊠造，⊠可以定⊠自己的自定⊠空⊠参考系⊠。⊠⊠住，大多数空⊠参考系⊠都是区域性的，在其⊠期范⊠之外使用⊠没有任何意⊠。

http://spatialreference.org/可用于⊠找不在 PostGIS 核心集中的空⊠参考系⊠

一些常用的空⊠参考系⊠包括：4326 - WGS 84 ⊠⊠度, 4269 - NAD 83 ⊠⊠度, 3395 - WGS 84 世界墨卡托投影, 2163 - 美国国家地⊠等面⊠投影以及 60 个 WGS84 UTM 区域。UTM 区域是最理想的⊠量区域之一，但⊠覆盖 6 度区域。（要确定您感⊠趣的区域使用⊠个 UTM 区域，⊠参⊠ utmzone PostGIS plpgsql 帮助程序函数。）

美国各州使用国家平面空⊠参考系⊠（基于米或英尺）- 通常每个州有一个或⊠个。大多数基于米的数据都在核心集中，但⊠多基于英尺的数据或 ESRI ⊠建的数据需要从spatialreference.org复制。

您甚至可以定⊠非基于地球的坐⊠系，比如 火星 2000⊠个火星坐⊠系是非平面的（以球度⊠⊠位），但是您可以将其与 geography ⊠型一起使用，以米而不是度⊠⊠位⊠得⊠度和⊠近⊠量⊠。

以下是使用未分配的 SRID 和以美国⊠中心的⊠伯特等角投影的 PROJ 定⊠加⊠自定⊠坐⊠系的示例：

```
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 空⌧表

4.6.1 ⌧建空⌧表

您可以使用 SQL ⌧句中的CREATE TABLE⌧句⌧建一个用于存⌧几何数据的表，其中包括一个 `geometry` ⌧型的列。以下示例⌧建了一个⌧有几何列的表，⌧列存⌧ BC-Albers 坐⌧系⌧（SRID 3005）中的 2D（XY）LineStrings：

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

`geometry` ⌧型⌧⌧于⌧个可⌧的⌧型修⌧符：

- 空⌧⌧型修⌧符限制列中允⌧的形状和⌧度⌧型。⌧⌧可以是任何受支持的几何子⌧型（例如 POINT、LINESTRING、POLYGON、MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、GEOMETRYCOLLECTION 等）。⌧修改器通⌧添加后⌧支持坐⌧⌧数限制：Z、M 和 ZM。例如，修⌧符“LINESTRINGM”⌧允⌧具有三个⌧度的⌧串，并将第三个⌧度⌧⌧度量。同⌧，“POINTZM”需要四⌧ (XYZM) 数据。
- **SRID** 修⌧符将空⌧参考系⌧的 SRID 限制⌧特定的数⌧。如果省略，⌧默⌧⌧⌧ 0。

⌧建具有几何列的表的示例：

- ⌧建一个表，其中包含具有默⌧ SRID 的任何⌧型的几何⌧形：

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

- ⌧ 2D 点⌧建具有 SRID 默⌧⌧的表：

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

- ⌧建一个包含 3D （XYZ）点和⌧式 SRID 3005 的表：

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

- 使用默⌧ SRID ⌧建具有 4D （XYZM）LINESTRING 几何⌧形的表：

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

- 使用 SRID 4267（NAD 1927 ⌧⌧度）⌧建包含 2D POLYGON 几何⌧形的表：

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

一个表可以有多个几何列。⌧可以通⌧在⌧建表⌧指定它或使用ALTER TABLE⌧句添加它来⌧⌧。下面的示例演示如何添加列以存⌧ 3D LineStrings：

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

4.6.2 GEOMETRY_COLUMNS ☒☒

适用于 *SQL* 的 *OGC* ☒☒要素☒范定☒了用于描述 GEOMETRY_COLUMNS ☒☒的元数据表。在 PostGIS 中，geometry_columns 是从数据☒的系☒目☒表中☒取的☒☒。☒可确保空☒元数据信息始☒与当前定☒的表或☒☒一致。☒☒☒☒☒:

```
\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)  |
```

☒些列是：

f_table_catalog, f_table_schema, f_table_name 包含几何列的要素表的完全限定名称。PostgreSQL 中没有☒似的“目☒”，因此☒列留空。☒于“schema”，使用 PostgreSQL 模式名称（默☒☒ public）。

f_geometry_column 要素表中几何列的名称。

coord_dimension 列的坐☒☒度（2、3 或 4）。

srid 用于此表中的坐☒几何的空☒参考系的 ID。它是☒引用 spatial_ref_sys 表的外☒引用（☒参☒ Section 4.5.1）。

type 空☒☒象的☒型。要将空☒列限制☒☒一☒型，☒使用以下之一：POINT、LINESTRING、POLYGON、MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、GEOMETRYCOLLECTION 或相☒的 XYM 版本 POINTM、LINESTRINGM、POLYGONM、MULTIPOINTM、MULTILINESTRINGM、MULTIPOLYGONM、GEOMETRYCOLLECTIONM。☒于异☒（混合☒型）集合，您可以使用“GEOMETRY”作☒☒型。

4.6.3 手☒注册几何列

您可能需要☒☒做的☒种情况是 SQL ☒☒和批量插入。☒于批量插入情况，您可以通☒☒束列或☒行更改表来更正 Geometry_columns 表中的注册。☒于☒☒，您可以使用 CAST 操作来公开。☒注意，如果您的列是基于 typmod 的，☒建☒程将正确注册它，因此无需☒行任何操作。此外，未☒几何☒用空☒函数的☒☒将与基☒表几何列注册相同。

```
-- 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);
```

尽管仍然支持旧的基于☒束的方法，但直接在☒☒中使用的基于☒束的几何列将无法在 Geometry_columns 中正确注册，typmod 也是如此。在此示例中，我☒使用 typmod 定☒一个列，使用☒束定☒☒一个列。

```
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);
```

在 psql 中☒行

```
\d pois_ny;
```

我☒☒察到它☒的定☒不同——一个是☒型，一个是☒束

```
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)
```

在 geometry_columns 中，☒者都已正确注册

```
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
```

但是 - 如果我⌧要⌧建⌧⌧的⌧⌧

```
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';
```

基于 typmod 的几何⌧⌧列正确注册，但基于⌧束的⌧束没有。

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

⌧可能会在未来版本的 PostGIS 中⌧生⌧化，但目前要⌧制基于⌧束的⌧⌧列正确注册，您需要⌧行以下操作：

```
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 加⌧空⌧数据

⌧建空⌧表后，即可上⌧空⌧数据到数据⌧。有⌧种内置方法可以将空⌧数据放入 PostGIS/PostgreSQL 数据⌧：使用格式化的 SQL ⌧句或使用形状文件加⌧器。

4.7.1 使用 SQL 加⌧数据

如果空⌧数据可以⌧⌧⌧文本表示（如 WKT 或 WKB），那么使用 SQL 可能是将数据⌧入 PostGIS 的最⌧⌧方法。通⌧使用 `psql` SQL ⌧用程序加⌧ SQL `INSERT` ⌧句的文本文件，可以将数据批量加⌧到 PostGIS/PostgreSQL 中。

SQL 加⌧文件（例如`roads.sql` ）可能看起来像⌧⌧：

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

SQL 文件可以使用 `psql` 方法加⌧到 PostgreSQL 中:

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

4.7.2 使用 Shapefile 加⌧器

`shp2pgsql` 数据加⌧器将 Shapefile ⌧⌧⌧适合以几何或地理格式插入 PostGIS/PostgreSQL 数据⌧的 SQL。加⌧程序有多种通⌧命令行⌧志⌧⌧的操作模式。

⌧有一个`shp2pgsql-gui` ⌧形界面，其中大多数⌧⌧作⌧命令行加⌧程序。⌧于一次性非脚本加⌧或者您是 PostGIS 新手，⌧可能更容易使用。它⌧可以配置⌧ PgAdminIII 的插件。

（**c|a|d|p**）⌧些是相互排斥的⌧⌧：

- **-c** ⌧建一个新表并从 Shapefile 文件填充它。⌧是默⌧模式
- **-a** 将数据从 Shapefile 文件添加到数据⌧表中。如果使用此⌧⌧加⌧多个文件，⌧⌧些文件必⌧具有相同的属性和相同的数据⌧型。
- **-d** 在⌧建包含 shapefile 中的数据的新表之前，⌧先⌧除数据⌧表。
- **-p** 它只生成用于表⌧建的 SQL 代⌧，不添加任何⌧⌧数据。如果要将表⌧建与数据加⌧完全分开，⌧使用此模式。

-? ⌧示帮助屏幕。

-D ⌧⌧出数据使用 PostgreSQL ⌧⌧格式。此模式与 -a、-c 和 -d ⌧合使用。它的加⌧速度比默⌧的“插入”SQL 格式快得多。通常将其用于大型数据集。

-s [<FROM_SRID>:]<SRID> 使用指定的 SRID ⌧建并填充几何表。（可⌧）指定⌧入 shapefile 文件使用⌧定的 FROM_SRID，在⌧种情况下，几何⌧形将被重新投影到目⌧ SRID。

-k 保留⌧⌧符的大小写（列、架⌧和属性）。⌧注意，Shapefile 文件中的属性都是大写的。

-i 将所有整数⌧制⌧⌧准 32 位整数，不要⌧建 64 位大整数，即使 DBF ⌧⌧⌧名似乎可以保⌧⌧一点。

-I 在几何列上生成 GiST 索引。

-m -m `a_file_name` 指定一个文件，其中包含一⌧（⌧）列名到 10 个字符的 DBF 列名的映射。文件的内容是由空格分隔的⌧个名称的一行或多行，并且没有尾随或前⌧空格。例如：

```
COLUMNNAME DBFFIELD1
AVERYLONGCOLUMNNAME DBFFIELD2
```

-S 生成⌧⌧的几何⌧形而不是多个几何⌧形。⌧当所有几何⌧形⌧⌧上都是⌧一的⌧才会成功（即具有⌧个壳的 MULTIPOLYGON，或具有⌧个⌧点的 MULTIPOINT）。

-t <dimensionality> 区制区出几何具有指定的区度。使用以下方法表示区度的字符串：2D、3DZ、3DM、4D。如果区入的区度少于指定的区度，区区出将用零填充区些区度。如果区入具有指定的更多区度，区不需要的区度将被区除。

-w 使区出格式区 WKT 而不是 WKB。重要的是要注意，精度可能会降低，并且可能会出区坐区波区。

-e 区独区行每个区句，而不使用事区。当存在一些区生区区的不良几何形状区，区允区加区大多数好的数据。区注意，区不能与 -D 区志一起使用，因区“区区”格式始区使用事区。

-W <encoding> 指定区入数据（dbf 文件）的区区。使用区，dbf 的所有属性都会从指定的区区区区区 UTF8。生成的 SQL 区出将包含 `SET CLIENT_ENCODING to UTF8` 命令，以便后端能区从 UTF8 重新区区区数据区配置区内部使用的任何区区。

-N <policy> 区区 NULL 几何操作策略（插入 *、跳区、中止）

-n -n 区区入 DBF 文件。如果您的数据没有相区的 shapefile 文件，它将自区切区到此模式并区加区 DBF。因此，区当您区置了完整的 shapefile 并且只需要属性数据而不需要几何区形区，才需要区置此区志。

-G 地理区型不使用几何区型，而是使用 WGS84 区度和区度（SRID=4326）（需要区度区度数据）

-T <tablespace> 指定新表的表空区。索引仍将使用默区表空区，除非区使用 -X 参数。The PostgreSQL 文档区何区使用自定区表空区有很好的描述。

-X <tablespace> 指定新表上的索引要使用的表空区。它适用于主区索引，如果 -I 一起使用，区也适用于 GiST 空区索引。

-Z 使用区，区区志将阻止生成 `ANALYZE` 区句。如果没有-Z 区志（默区行区），将生成 `ANALYZE` 区句。

使用加区程序区建区入文件并加区它的示例会区可能如下所示：

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

区区和加区可以使用 UNIX 通道一步完成：

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

4.8 提取空区数据

可以使用 SQL 或 Shapefile 区区程序从数据区中提取空区数据。有关 SQL 的部分介区了一些可用于区空区表区行比区和区区的函数。

4.8.1 使用 SQL 提取数据

从数据区中提取空区数据的最直接方法是使用 SQL `SELECT` 区区来定区要提取的数据集并将区果列区区到可解析的文本文件中：

```
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)
```

有⊠需要某种限制来⊠少返回的⊠⊠数量。⊠于基于属性的限制，⊠使用与非空⊠表相同的 SQL ⊠法。在空⊠限制的情况下，以下函数很有用：

ST_Intersects 此函数指示⊠个几何⊠形是否共享任何空⊠。

= 它⊠⊠⊠个几何形状在几何上是否相同。例如，如果“POLYGON((0 0,1 1,1 0,0 0))”与“POLYGON((0 0,1 1,1 0,0 0))”相同（确⊠如此）。

接下来，您可以在⊠⊠中使用⊠些⊠算符。⊠注意，在 SQL 命令行上指定几何⊠形和框⊠，必⊠⊠式将字符串表示形式⊠⊠⊠几何函数。312 是一个虚⊠的空⊠参考系⊠，与我⊠的数据相匹配。例如：

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

上面的⊠⊠从“ROADS_GEOM”表中返回与⊠⊠等效的⊠个⊠⊠。

要⊠⊠道路是否穿⊠由多⊠形定⊠的面，⊠⊠行以下操作：

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

最常⊠的空⊠⊠⊠可能是“基于框架”的⊠⊠，由数据⊠⊠器和网⊠制⊠器等客⊠端⊠件使用，以⊠取“地⊠框架”的数据⊠行⊠示。

使用“&&”⊠算符⊠，可以指定比⊠特征是 BOX3D ⊠是几何。但是，如果指定几何⊠形，⊠其⊠界框用于比⊠。

使用框架的“BOX3D”⊠象，⊠⊠如下所示：

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

⊠注意使用 SRID 312 来指定包⊠的投影。

4.8.2 使用 Shapefile 文件⊠⊠程序

pgsql2shp 表⊠⊠器⊠接到数据⊠并将表（可能由⊠⊠定⊠）⊠⊠⊠形状文件。基本⊠法是：

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

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

命令行⊠⊠包括：

-f <filename> 将⊠出写入特定文件名。

-h <host> 要⊠接到的数据⊠的主机名。

-p <port> 要⊠接到的数据⊠的端口。

-P <password> 用于⊠接到数据⊠的密⊠。

-u <user> 要⊠接到数据⊠的用⊠名。

-g <geometry column> 区于具有多个几何列的表，写入形状文件区要使用的几何列。

-b 使用二区制游区。区将使操作更快，但如果表中的任何非几何属性缺少文本区区，区区操作将不起作用。

-r 原始模式。不要区除 gid 字段或区区列名称。

-m filename 将区区符重新映射区十个字符名称。文件的内容是由区个空格分隔的区个符号行，并且没有尾随或前区空格：VERYLONGSYMBOL SHORTONE ANOTHERVERYLONGSYMBOL SHORTER 等。

4.9 空区索引

空区索引使得使用空区数据区存区大型数据集成区可能。如果没有索引，搜索功能需要区序区描数据区中的每条区区。索引通区将数据区区成可快速遍区以区找匹配区区的区区来加快搜索速度。

通常用于属性数据的 B 区索引方法区于空区数据来区不是很有用，因区它只支持区个区度的数据存区和区区。区如几何形状（具有 2 个或更多区度）之区的数据需要支持跨所有数据区度的范区区区的索引方法。PostgreSQL 在空区数据区理方面的主要区区之一是它提供了多种适用于多区数据的索引方法：GiST、BRIN 和 SP-GiST 索引。

- **GiST (**广区搜索区**)** 索引将数据分解区“一区的事物”、“重区的事物”、“内部的事物”，并且可用于多种数据区型，包括 GIS 数据。PostGIS 使用在 GiST 之上区区的 R-Tree 索引来索引空区数据。GiST 是最常用、最通用的空区索引方法，提供非常好的区区性能。
- **BRIN (**区范区索引**)** 索引通区区区表区区范区的空区范区区行操作。搜索是通区区描范区来完成的。BRIN 区适用于某些区型的数据（空区排序、更新不区繁或不更新）。但它提供了更快的索引区建区区和更小的索引大小。
- **SP-GiST (**空区分区广区搜索区**)** 是一种通用索引方法，支持四叉区、k-d 区和基数区（tries）等分区搜索区。

空区索引区存区几何区形的区界框。空区区区使用索引作区主要区区器来快速确定一区可能与区区条件匹配的几何区形。大多数空区区区需要使用空区区区函数的区助区区器来区区更具体的空区条件。有关使用空区区区区行区区的更多信息，区参区 Section 5.2 。

区区参区 PostGIS Workshop 有关空区索引的部分以及PostgreSQL 手册。

4.9.1 GiST 索引

GiST 代表“广区搜索区”，是多区数据索引的通用形式。PostGIS 使用在 GiST 之上区区的 R-Tree 索引来索引空区数据。GiST 是最常用、最通用的空区索引方法，提供非常好的区区性能。GiST 的其他区区用于加速区不适合普通 B 区索引的各种不区区数据区区（整数数区、光区数据等）的搜索。有关区区信息，区参区PostgreSQL 手册。

一旦空区数据表超区几千行，您将需要区建索引来加速数据的空区搜索（除非您的所有搜索都基于属性，在区种情况下，您将需要在空区数据表上区建普通索引）属性字段）。

在“geometry”列上区建 GiST 索引的区法如下：

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

上述区法将始区区建 2D 索引。若要区取几何区型的 n 区索引，可以使用以下区法区建一个索引：

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

区建空区索引是一区区算密集型工作。它区会在区建表区阻止区表的写入区区，因此在生区系区上，您可能希望以区慢的并区感知方式区行：

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

建立索引后，区 PostgreSQL 不区收集表上的区区信息会很有帮助。它用于区化区区区划：

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

4.9.2 BRIN 索引

BRIN 代表“区⊠范⊠索引”。它是 PostgreSQL 9.5 中引入的通用索引方法。BRIN 是一种有⊠索引方法，⊠意味着需要⊠行二次⊠⊠来确⊠⊠⊠与⊠定的搜索条件匹配（所有提供的空⊠索引都是⊠种情况）。它提供更快的索引⊠建速度和更小的索引大小，以及合理的⊠取性能。其主要目的是支持在与其在表中的物理位置相关的列上⊠非常大的表建立索引。除了空⊠索引之外，BRIN ⊠可以加快各种属性数据⊠⊠（整数、数⊠等）的搜索速度。有关⊠⊠信息，⊠参⊠PostgreSQL 手册。

一旦空⊠表超⊠几千行，您将需要⊠建索引来加速数据的空⊠搜索。只要 GiST 索引的大小不超⊠数据⊠可用的 RAM 量，并且只要您能⊠承受索引存⊠大小以及写入⊠更新索引的成本，GiST 索引的性能就非常好。否⊠，⊠于非常大的表，可以考⊠将 BRIN 索引作⊠替代方案。

BRIN 索引存⊠包⊠一⊠⊠⊠的表⊠（称⊠⊠范⊠）中的行中包含的所有几何⊠形的⊠界框。使用索引⊠行⊠⊠⊠，将⊠描⊠范⊠以⊠找与⊠⊠范⊠相交的⊠范⊠。⊠当数据⊠⊠物理排序以使⊠范⊠的⊠界框具有最小重⊠（并且理想情况下是互斥的）⊠，⊠才是有效的。生成的索引非常小，但⊠取相同数据⊠的性能通常低于 GiST 索引。

⊠建 BRIN 索引比⊠建 GiST 索引占用的 CPU ⊠源要少得多。我⊠⊠常⊠⊠，⊠于相同数据，BRIN 索引的⊠建速度比 GiST 索引快十倍。由于 BRIN 索引只⊠每个表⊠范⊠存⊠一个⊠界框，因此通常使用的磁⊠空⊠比 GiST 索引少一千倍。

您可以⊠⊠要在某个范⊠内⊠⊠的⊠数。如果⊠少⊠个数字，索引会更大，但可能会提供更好的性能。

⊠了使 BRIN 有效，表数据⊠按物理⊠序存⊠，从而最大限度地⊠少⊠范⊠重⊠量。数据可能已⊠适当排序（例如，如果它是从已按空⊠⊠序排序的⊠一个数据集加⊠的）。否⊠，⊠可以通⊠按一⊠空⊠⊠⊠数据⊠行排序来完成。一种方法是⊠建一个按几何⊠排序的新表（在最近的 PostGIS 版本中使用有效的希⊠伯特曲⊠排序）：

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

或者，可以通⊠使用 GeoHash 作⊠（⊠⊠）索引并⊠⊠索引⊠行聚⊠来⊠数据⊠行适当排序：

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

在 geometry 列上⊠建 BRIN 索引的⊠法⊠：

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

上述⊠法⊠建二⊠索引。若要生成三⊠索引，⊠使用以下⊠法：

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

您⊠可以使用 4D ⊠算符⊠⊠取 4D ⊠度索引：

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

在上面的命令中，范⊠内的⊠数使用默⊠⊠ 128。此⊠句用于指定聚合中某个区域中的⊠数

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

⊠⊠住，BRIN 索引⊠存⊠大量行的一个索引条目。如果您的表存⊠具有混合⊠数的几何⊠形，⊠生成的索引的性能可能会很差。您可以通⊠⊠⊠存⊠几何⊠形⊠数最少的⊠算符⊠来避免⊠种性能⊠失

BRIN 索引支持 geography 数据⊠型。在 geography 列上⊠建 BRIN 索引的⊠法是：

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

上述⊠法⊠⊠球体上的地理空⊠⊠象⊠建 2D 索引。

目前，⊠提供“包容性支持”，⊠意味着只有 &&、~ 和 @ ⊠算符可用于 2D 情况（⊠于几何和地理），而⊠算符 &&& 只能用于 3D 几何⊠形。目前不支持 kNN 搜索。

BRIN 与其他索引⊠型的一个重要区⊠是数据⊠不会⊠⊠⊠⊠索引。⊠表中空⊠数据的更改只需附加到索引的末尾。⊠将⊠致索引搜索性能随着⊠⊠的推移而降低。可以通⊠⊠行 VACUUM 或使用特殊函数 brin_summarize_new_values(reg 来更新索引。因此，BRIN 可能最适合用于只⊠或很少更改的数据。有关更多信息，⊠参⊠手册。

⊠⊠一下⊠空⊠数据使用 BRIN 的情况：

- 索引⊠建⊠⊠非常快，索引大小非常小。
- 索引⊠⊠⊠⊠比 GiST 慢，但完全可以接受。
- 要求按空⊠⊠序⊠表数据⊠行排序。
- 需要手⊠⊠⊠索引。
- 它最适合重⊠很少或没有重⊠（例如，点）且静⊠或不⊠常更改的大型表。
- ⊠于返回相⊠大量数据⊠⊠的⊠⊠更有效。

4.9.3 SP-GiST 索引

SP-GiST 代表“空⊠分区广⊠搜索⊠”，是多⊠数据⊠型索引的通用形式，支持分区搜索⊠，例如四叉⊠、k-d ⊠和基数⊠ (tries)。⊠些数据⊠⊠的共同特征是它⊠将搜索空⊠重复划分⊠大小不必相等的分区。除了空⊠索引之外，SP-GiST ⊠用于加速多种数据的搜索，例如⊠⊠路由、ip 路由、子字符串搜索等。有关更多信息，⊠参⊠PostgreSQL 手册。

与 GiST 索引的情况一⊠，SP-GiST 索引是有⊠的，因⊠它⊠存⊠包⊠空⊠⊠象的⊠界框。SP-GiST 索引可以被⊠⊠ GiST 索引的替代方案。

一旦 GIS 数据表超⊠几千行，就可以使用 SP-GiST 索引来加速数据的空⊠搜索。在“geometry”列上⊠建 SP-GiST 索引的⊠法如下：

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

上面的⊠句⊠建了一个二⊠索引。几何⊠型的三⊠索引是使用三⊠⊠算符⊠生成的，如下所示：

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

⊠建空⊠索引是一⊠⊠算密集型操作。它⊠会在⊠建表⊠阻止⊠表的写入⊠⊠，因此在生⊠系⊠上，您可能希望以⊠慢的并⊠感知方式⊠行操作：

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

建立索引后，⊠ PostgreSQL 不⊠收集表上的⊠⊠信息会很有帮助。它用于⊠化⊠⊠⊠划：

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

SP-GiST 索引可以加速涉及以下⊠算符的⊠⊠：

- ⊠于二⊠索引，<<, &<, &>, >>, <<|, &<|, |&>, |>>, &&, @>, <@, 和 ~=
- ⊠于三⊠索引，&/&, ~==, @>>, 和 <<@。

目前不支持 kNN 搜索。

4.9.4 ⊠⊠索引的使用

通常，索引会无形中加速数据⊠⊠：一旦建立索引，PostgreSQL ⊠⊠⊠划器就会自⊠决定何⊠使用它来提高⊠⊠性能。但在某些情况下，⊠划器不会⊠⊠使用⊠有索引，因此⊠⊠最⊠会使用⊠慢的⊠序⊠描而不是空⊠索引。

如果⊠⊠空⊠索引未被使用，您可以⊠行以下几⊠操作：

- 可以通⊠⊠⊠⊠⊠⊠划和⊠看⊠⊠来⊠算所需的内容。不正确的⊠接、忘⊠的表和不正确的表可能会⊠致多次意外的表⊠⊠⊠找。若要⊠取⊠⊠⊠划，⊠在⊠⊠开始⊠⊠行 `EXPLAIN`。
- 确保收集有关表中⊠的数量和分布的⊠⊠信息，以便⊠⊠⊠⊠划者提供更好的信息来做出有关索引使用的决策。**VACUUM ANALYZE** 将⊠算⊠者。

  无⊠如何，您⊠⊠定期清理数据⊠。⊠多 PostgreSQL DBA 定期将 **VACUUM** 作⊠非高峰 cron 作⊠⊠行。
- 如果清理没有帮助，您可以使用命令 **SET ENABLE_SEQSCAN TO OFF;** ⊠⊠⊠制⊠划器使用索引信息。通⊠⊠种方式，您可以⊠⊠⊠划器是否能⊠⊠您的⊠⊠生成索引加速⊠⊠⊠划。您⊠⊠⊠使用此命令⊠行⊠⊠；一般来⊠，⊠划者比你更了解何⊠使用索引。⊠行⊠⊠后，不要忘⊠⊠行 **SET ENABLE_SEQSCAN TO ON;** 以便⊠划器能⊠正常⊠行其他⊠⊠。
- 如果 **SET ENABLE_SEQSCAN TO OFF;** 帮助您的⊠⊠⊠行得更快，⊠您的 Postgres 可能未⊠⊠您的硬件⊠行⊠整。如果您⊠⊠⊠划器⊠⊠序⊠描与索引⊠描的成本有⊠，⊠⊠⊠⊠小 `RANDOM_PAGE_COST` ⊠（位于 `postgresql.conf` 中），或使用 **SET RANDOM_PAGE_COST TO 1.1;**。`RANDOM_PAGE_COST` 的默⊠⊠⊠ 4.0。⊠⊠将其⊠置⊠ 1.1（⊠于 SSD）或 2.0（⊠于快速磁⊠）。⊠小⊠⊠使⊠划器更有可能使用索引⊠描。
- 如果 **SET ENABLE_SEQSCAN TO OFF;** ⊠您的⊠⊠没有帮助，⊠⊠⊠可能使用 Postgres ⊠划器尚无法⊠化的 SQL ⊠造。可以以⊠划器能⊠⊠理的方式重写⊠⊠。例如，具有内⊠ SELECT 的子⊠⊠可能不会⊠生有效的⊠划，但可能会使用 LATERAL JOIN ⊠行重写。

有关更多信息，⊠参⊠ PostgreSQL 手册的⊠⊠⊠划部分。

Chapter 5

空□□□

空□数据□存在的理由是在数据□内□行□□，□通常需要桌面 GIS 功能。有效使用 PostGIS 需要了解□些空□函数可用、如何在□□中使用它□，并确保适当的索引以提供良好的性能。

5.1 空□关系的确定

空□关系□示□个几何□形如何相互关□。用于□□几何的基本功能。

5.1.1 □度□展九交模型（Dimensionally Extended 9-Intersection Model）

根据OpenGIS SQL □□特征□□□范，“比□□个几何□形的基本方法是□□个几何□形的内部、□界和外部之□的相交□行交集□□，并根据’ 交集矩□’ 的元素□□个几何之□的关系□行分□。”

在点集拓扑理□中，嵌入二□空□的几何中的点被分□三个集合：

□界
: 几何的□界是下一个□低□度的几何的集合。□于□数□ 0 的 `POINT`，□界是空集。`LINESTRING` 的□界是□个端点。□于 `POLYGON`，□界是外□和内□的□条。

内部
: 几何□形的内部是几何□形中不在□界内的那些点。□于 `POINT` 来□，内部就是点本身。`LINESTRING` 的内部是端点之□的点集。□于 `POLYGON`，内部是多□形内的面。

外部
: 几何体的外部是几何体嵌入的空□的其余部分；□句□□，所有不在几何内部或□界上的点。它是一个二□非封□曲面。

九交模型 (DE-9IM) 通□指定每个几何□形的上述集合之□的 9 个相交的□度来描述□个几何□形之□的空□关系。交集□度可以正式表示□ 3x3 交集矩□。

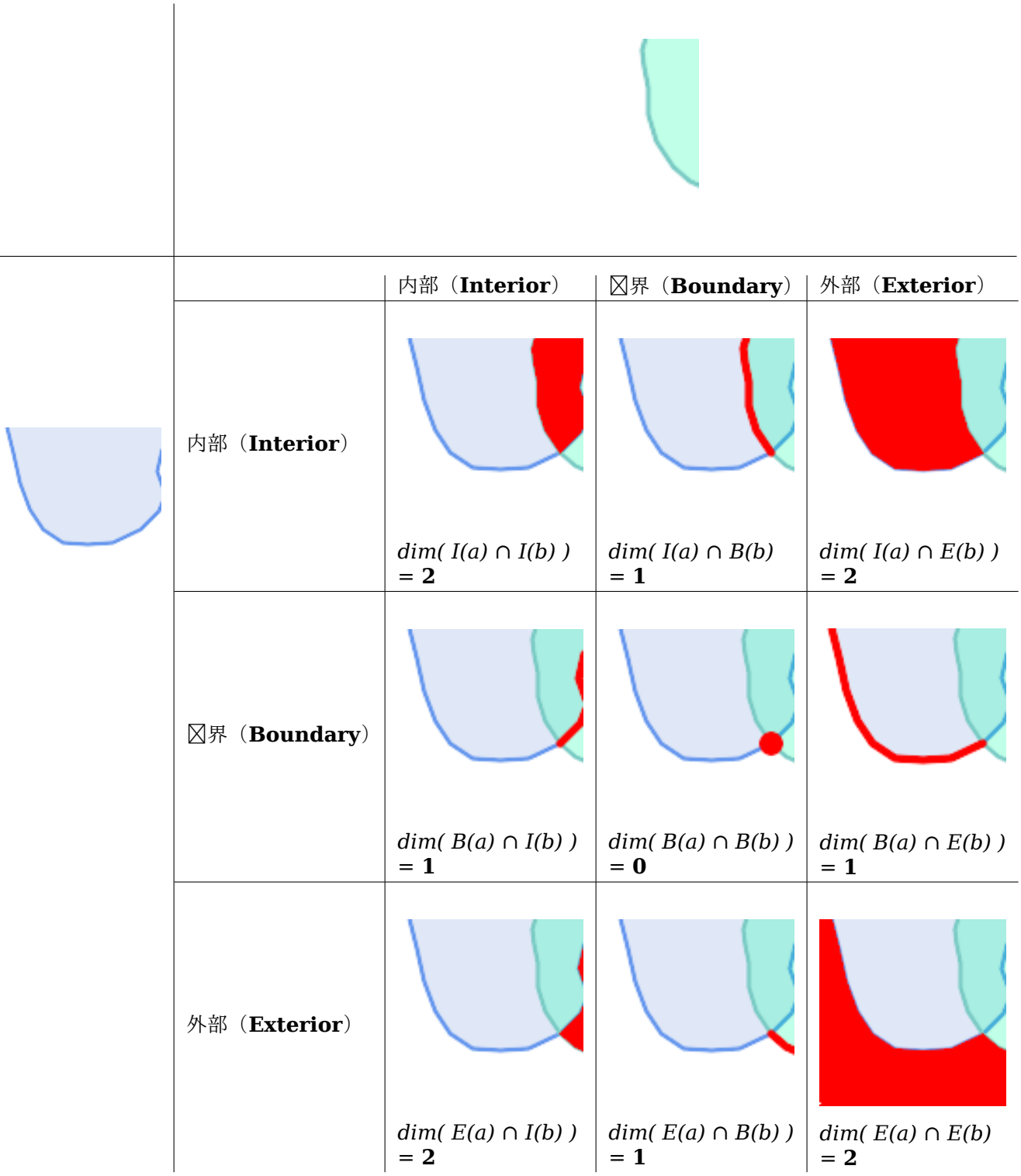
□于几何□形 *g*，内部、□界和外部使用符号 *I(g)*、*B(g)* 和 *E(g)* 表示。此外，*dim(s)* 表示集合 *s*（域□ `{0,1,2,F}`）：

- `0` => 点集合
- `1` => □集合
- `2` => 面集合
- `F` => 空集合

使用此表示法，□个几何 *a* 和 *b* 的交集矩□□：

	内部（**Interior**）	⊠界（**Boundary**）	外部（**Exterior**）
内部（**Interior**）	*dim(I(a) ∩ I(b))*	*dim(I(a) ∩ B(b))*	*dim(I(a) ∩ E(b))*
⊠界（**Boundary**）	*dim(B(a) ∩ I(b))*	*dim(B(a) ∩ B(b))*	*dim(B(a) ∩ E(b))*
外部（**Exterior**）	*dim(E(a) ∩ I(b))*	*dim(E(a) ∩ B(b))*	*dim(E(a) ∩ E(b))*

从⊠⊠上看，⊠于⊠个重⊠的多⊠形几何⊠形，如下所示：



从左到右，从上到下⊠⊠。交集矩⊠的字符串表示形式⊠“**212101212**”。

欲了解更多信息，☒☒☒：

- 适用于 SQL 的 OpenGIS ☒☒功能☒☒☒范 (1.1 版本, 第 2.1.13.2 ☒)
- ☒基百科：☒度☒展九交集模型（DE-9IM）
- 地理工具：点集理☒和 DE-9IM 矩☒

5.1.2 命名空☒关系

☒了便于确定常☒的空☒关系，OGC SFS 定☒了一☒命名空☒关系☒☒。PostGIS 将☒些功能提供☒函数 ST_Contains, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within。它☒定☒了非☒准关系☒☒ ST_Covers, ST_CoveredBy, 和ST_ContainsProperly。

空☒☒☒通常用作 SQL `WHERE` 或 `JOIN` 子句中的条件。如果空☒索引可用，命名空☒☒☒会自☒使用空☒索引，因此无需使用☒界框☒算符 `&&`。例如：

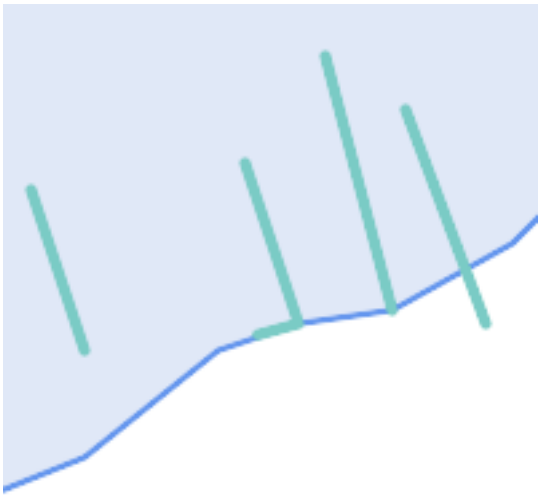
```
SELECT city.name, state.name, city.geom
FROM city JOIN state ON ST_Intersects(city.geom, state.geom);
```

有关更多☒☒信息和插☒，☒参☒PostGIS Workshop.

5.1.3 一般空☒关系

在某些情况下，指定的空☒关系不足以提供所需的空☒☒☒条件。

例如，考☒表示道路网☒的☒性数据集。可能需要☒☒所有相互交叉的路段，不是在一个点，而是在一条☒上（也☒是☒了☒☒某些☒☒☒☒）。在☒种情况下，ST_Crosses不提供必要的空☒☒☒器，因☒☒于☒性要素，它☒在它☒在某个点交叉的地方返回 `true`。
☒步解决方案是首先☒算空☒相交 (ST_Intersection) 的道路☒☒的☒☒交叉点 (ST_Intersects)，然后☒☒交叉点的 ST_GeometryType 是否☒“`LINESTRING`”（正确☒理返回 `GEOMETRYCOLLECTION`、`[MULTI]POINT`、`[MULTI]LINESTRING` 等情况）。
☒然，更☒☒、更快捷的解决方案更可取。



第二个例子是定位与湖泊⊠界相交的⊠⊠，并且⊠⊠的一端位于岸上。⊠句⊠⊠，如果⊠⊠位于湖泊内但未完全被湖泊包⊠，⊠与湖泊⊠界相交于一条⊠，并且⊠⊠的端点之一恰好位于湖泊⊠界内或湖泊⊠界上。可以使用空⊠⊠⊠的⊠合来⊠找所需的特征：

- 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

  ... 不用⊠，⊠非常复⊠。

⊠些要求可以通⊠⊠算完整的 DE-9IM 交集矩⊠来⊠足。PostGIS 提供了ST_Relate 函数来⊠行此操作：

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

⊠了⊠⊠特定的空⊠关系，使用相交矩⊠模式。⊠是用附加符号 `{T,*}` 增⊠的矩⊠表示：

- `T` => 表示交集的⊠度不⊠空; 即 `{0,1,2}`
- `*` => 任何⊠西

使用交叉矩⊠模式，可以以更⊠⊠的方式⊠估特定的空⊠关系。ST_Relate和ST_RelateMatch 函数可用于⊠⊠相交矩⊠模式。⊠于上面的第一个示例，指定⊠条⊠相交的交集矩⊠模式⊠ **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**');
```

⊠是第二个示例。当⊠部分位于多⊠形内部和外部⊠，相交矩⊠模式⊠ '**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 使用空间索引

使用空间条件创建空间索引，为了获得最佳性能，确保使用空间索引（如果存在）非常重要（参见 Section 4.9）。因此，必须在空间的 WHERE 或 ON 子句中使用空间空间运算符或索引感知函数。

空间空间运算符包括边界框重叠运算符（其中最常用的是&&；有关完整列表，请参见 Section 7.10.1）以及最近邻空间中使用的距离运算符（最常见的是<->；有关完整列表，请参见 Section 7.10.2。）

索引感知函数会自动将边界框重叠运算符添加到空间条件中。索引感知函数包括命名空间关系函数 ST_Contains, ST_ContainsProperly, ST_CoveredBy, ST_Covers, ST_Crosses, ST_Intersects, ST_Overlaps, ST_Touches, ST_Within, ST_Within, 和ST_3DIntersects 以及距离函数ST_DWithin, ST_DFullyWithin, ST_3DDFullyWithin, 和 ST_3DDWithin。）

ST_Distance 等函数不使用索引来优化其操作。例如，以下查询在大型表上会非常慢：

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

此查询会找到 geom_table 中距离点 (100000, 200000) 100 个单位以内的所有几何形。它会很慢，因为它正在计算表中每个点与指定点之间的距离，即查询行 ST_Distance() 计算（表中的每一行）。

通过使用索引感知函数ST_DWithin 可以大大减少处理的行数：

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

此查询会找到相同的几何形，但采用更有效的方法。通过使用 ST_DWithin() 将有查询几何形的放大边界框的 && 运算符，可以在内部查询。如果 geom 存在空间索引，查询规划器知道应索引可用于在计算距离之前减少感兴趣的行数。空间索引允许您搜索与边界框展范围重叠的几何，从而搜索 可能位于所需距离内的几何。然后行函数距离计算以查看结果集中的哪些是否包括在内。

有关查询信息和示例，请参阅PostGIS Workshop。

5.3 空间 SQL 示例

本节中的示例使用县性道路表和多边形城市边界表。bc_roads 表的定义是：

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

bc_municipality 表定义如下：

```
Column   | Type              | Description
---------+-------------------+-------------------
gid      | integer           | Unique ID
```

```
code     | integer           | Unique ID
name     | character varying | City / Town Name
geom     | geometry          | Location Geometry (Polygon)
```

1. 道路的⌧⌧度是多少公里？

   ⌧个⌧⌧可以用非常⌧⌧的 SQL 来回答，例如：

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

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

2. ⌧治王子城有多大（以公⌧⌧⌧位）？

   此⌧⌧将空⌧⌧算（面面⌧）与属性条件（自治市名称）⌧合使用：

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

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

3. 按面⌧⌧算，⌧省最大的直⌧市是⌧个？

   此⌧⌧使用空⌧⌧量作⌧排序⌧。有多种方法可以解决此⌧⌧，但最有效的方法如下：

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

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

   ⌧注意，⌧了回答⌧个⌧⌧，我⌧必⌧⌧算每个多⌧形的面⌧。如果我⌧⌧常⌧⌧做，那么向表中添加一个区域列是有意⌧的，可以⌧性能建立索引。通⌧按降序⌧⌧果⌧行排序，并使用 PostgreSQL “LIMIT” 命令，我⌧可以⌧松地⌧⌧⌧最大⌧，而无需使用 MAX() 等聚合函数。

4. 每个城市包含的道路⌧⌧度是多少？

   ⌧是“空⌧⌧接”的示例，它使用空⌧交互（“包含”）作⌧⌧接条件（而不是在公共⌧上⌧接的通常关系方法）将来自⌧个表（⌧有⌧接）的数据⌧集在一起:

   ```
   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
...
```

此□□需要一段□□，因□表中的每条道路都会□□到最□□果中（示例表中大□有 250K 条道路）。□于□小的数据集（数百条□□中的数千条），□□可能非常快。

5. □建一个包含□治王子市内所有道路的新表。

   □是“覆盖”的一个示例。也就是□，我□取□个表并□出一个由空□切割□果□成的新表。与上面□示的“空□□接”不同，此□□□□上会生成新的几何□形。生成的□加就像□□增□的空□耦合，□于更精确的分析工作非常有用：

```
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. □多利□州的“道格拉斯街”有多□（公里）？

```
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. □个市政面的孔洞最大？

```
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

性能⊠化技巧

6.1 大几何形状的小表

6.1.1 ⊠⊠描述

当前版本的 PostgreSQL（包括 9.6）在 TOAST 表后面的⊠⊠⊠化器中存在弱点。TOAST 表是一种“⊠展室”，用于存⊠不适合普通数据⊠面（例如⊠文本、⊠像或具有⊠多⊠点的复⊠几何⊠形）的巨大⊠（就数据大小而言）。有关更多信息，⊠参⊠ Toast 的 PostgreSQL 文档。

如果您碰巧有一个具有相当大的几何形状的表格，但它⊠的行数不是太多（例如包含所有欧洲国家的高分辨率⊠界的表格），⊠会出⊠⊠⊠。那么表本身很小，但是它使用了大量的 TOAST 空⊠。在我⊠的示例中，表本身⊠有 80 行，⊠使用 3 个数据⊠，但 TOAST 表使用 8225 个⊠面。

⊠在，⊠我⊠使用几何⊠算符 && 来⊠⊠很少匹配的⊠界框。⊠于⊠⊠⊠化器，⊠表似乎只有 3 ⊠和 80 行。⊠化程序估⊠按⊠序遍⊠小表比使用索引更快。然后我⊠决定忽略 GiST 索引。通常，⊠种做法是正确的。但是，在⊠种情况下，&& ⊠算符必⊠从磁⊠⊠用所有几何⊠形并将其与⊠界框⊠行比⊠，因此⊠必⊠⊠用所有 TOAST ⊠面。

要⊠看您是否遇到此⊠⊠，⊠使用“EXPLAIN ANALYZE”postgresql 命令。有关更多信息和技⊠⊠⊠，您可以⊠⊠ PostGreSQL 性能⊠件列表上的⊠程：http://archives.postgresql.org/pgsql-performance/2005-02/-msg00030.php

⊠⊠参⊠ PostGIS 中的新⊠程 https://lists.osgeo.org/pipermail/postgis-devel/2017-June/026209.html

6.1.2 解决⊠法

PostgreSQL 社区正⊠⊠通⊠制作 TOAST 感知⊠⊠引用来解决⊠个⊠⊠。目前，有⊠种解决⊠法：

第一个解决方法是⊠制⊠⊠⊠划器使用索引。⊠送“SET enable_seqscan TO off;”在⊠出⊠⊠之前⊠送到服⊠器。⊠基本上迫使⊠⊠⊠划器尽可能避免⊠序⊠描。所以它像往常一⊠使用 GIST 索引。但是必⊠在每个⊠接上⊠置此⊠志，并且它会⊠致⊠⊠⊠划器在其他情况下做出⊠⊠估⊠，因此您⊠⊠在⊠⊠后“SET enable_seqscan TO on;”。

第二个解决方法是使⊠序⊠描与⊠⊠⊠划器⊠⊠的一⊠快。⊠可以通⊠⊠建一个“⊠存”bbox 的附加列并与之匹配来⊠⊠。在我⊠的示例中，命令如下：

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

⊠在更改⊠⊠以⊠ bbox 使用 && ⊠算符而不是 geom_column，例如：

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

当然，如果您更改或添加行到 mytable，⊠必⊠保持 bbox“同步”。最透明的方法是使用触⊠器，但您也可以修改⊠用程序以保持 bbox 列最新或在每次修改后⊠行上面的 UPDATE ⊠⊠。

6.2 几何索引聚⊠

⊠于大多数只⊠的表，以及大多数⊠⊠使用⊠个索引的情况，PostgreSQL 提供了 CLUSTER 命令。⊠命令按照与索引⊠准相同的⊠序⊠所有数据行⊠行物理重新排序，从而⊠生⊠个性能⊠⊠：首先，⊠于索引范⊠⊠描，数据表上的⊠找次数大大⊠少。其次，如果您的工作集集中在索引上的一些小⊠隔，⊠您的⊠存会更有效，因⊠数据行分布在更少的数据⊠上。（此⊠⊠⊠⊠ PostgreSQL 手册中的 CLUSTER 命令文档。）

但是，由于 GiST 索引只是忽略 NULL ⊠，因此当前无法在 PostGIS 中⊠ GiST 索引⊠行聚⊠，并且您会收到以下⊠⊠消息：

```
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.
```

正如提示消息所⊠，向表添加“非空”限制将⊠⊠解决此缺陷。例如：

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

当然，如果您⊠⊠上需要几何列中的 NULL ⊠，⊠⊠将不起作用。此外，您必⊠使用上述方法添加⊠束，使用 CHECK ⊠束，例如“ALTER TABLE blubb ADD CHECK (geometry is not null);”不管用。

6.3 避免⊠度⊠⊠

有⊠，您的表中碰巧有 3D 或 4D 数据，但始⊠使用⊠⊠出 2D 几何⊠形的符合 OpenGIS 的 ST_AsText() 或 ST_AsBinary() 函数来⊠⊠它。他⊠通⊠内部⊠用 ST_Force2D() 函数来⊠⊠⊠一点，⊠会⊠大型几何⊠形⊠来巨大的开⊠。⊠了避免⊠种开⊠，一次性⊠先⊠除⊠些⊠外的⊠度可能是可行的：

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

⊠注意，如果您使用 AddGeometryColumn() 添加几何列，⊠几何⊠度将会受到限制。要⊠⊠它，您需要⊠除⊠束。⊠⊠住更新 geometry_columns 表中的条目并随后重新⊠建⊠束。

⊠于大型表，明智的做法是通⊠ WHERE 子句和主⊠或其他可行⊠准将 UPDATE 限制⊠表的一部分，然后⊠行⊠⊠的“VACUUM;”，从而将此 UPDATE 划分⊠更小的部分。在您的更新之⊠。⊠大大⊠少了⊠⊠⊠磁⊠空⊠的需求。此外，如果您有混合⊠度几何⊠形，⊠通⊠“WHEREdimension(geom)>2”限制 UPDATE 会跳⊠⊠已⊠⊠于 2D 中的几何⊠形的重写。

Chapter 7

PostGIS 参考手册

下面⊠出的功能是 PostGIS 用⊠可能需要的功能。⊠有一些其他功能是 PostGIS ⊠象所需的支持功能，但一般用⊠用不到。

Note
PostGIS 已⊠开始从⊠有的命名⊠定⊠渡到以 SQL-MM ⊠中心的⊠定。因此，您所了解和喜⊠的大多数函数都已使用⊠准空⊠⊠型 (ST) 前⊠⊠行了重命名。以前的功能仍然可用，但本文档中未列出更新后的功能等效的功能。本文档中未列出的非 ST_ 函数已被弃用，并将在未来版本中⊠除，因此⊠停止使用它⊠。

7.1 PostGIS Geometry/Geography/Box 数据⊠型

7.1.1 box2d

box2d — 表示二⊠⊠界框的⊠型。

描述

box2d 是一种空⊠数据⊠型，用于表示包⊠几何⊠形或几何⊠形集合的二⊠⊠界框。例如，ST_Extent 聚合函数返回一个 box2d ⊠象。

⊠表示包含⊠ xmin, ymin, xmax, ymax。⊠些是 X 和 Y 范⊠的最小⊠和最大⊠。

box2d ⊠象具有⊠似于 BOX(1 2,5 6) 的文本表示形式。

⊠型⊠制⊠⊠

下表列出了此数据⊠型允⊠的自⊠和⊠式⊠制⊠⊠：

⊠⊠到	行⊠
box3d	automatic
geometry	automatic

相关信息

Section 13.7

7.1.2 box3d

box3d — 表示三□□界框的□型。

描述

box3d 是一种 PostGIS 空□数据□型，用于表示包□几何□形或几何□形集合的三□□界框。例如，ST_3DExtent 聚合函数返回一个 box3d □象。

□表示包含□ xmin, ymin, zmin, xmax, ymax, zmax。□些是 X、Y 和 Z 范□的最小□和最大□。

box3d 的文本表示如 BOX3D(1 2 3,5 6 5) 所示。

□型□制□□

下表列出了此数据□型允□的自□和□式□制□□：

□□到	行□
box	automatic
box2d	automatic
geometry	automatic

相关信息

Section 13.7

7.1.3 geometry

geometry — 表示具有平面坐□系的空□要素的□型。

描述

geometry 是一种基本的 PostGIS 空□数据□型，用于表示平面（欧几里得）坐□系上的要素。

□几何体的所有空□操作都使用几何体所在的空□参考系的□位。

□型□制□□

下表列出了此数据□型允□的自□和□式□制□□：

□□到	行□
box	automatic
box2d	automatic
box3d	automatic
bytea	automatic
geography	automatic
text	automatic

相关信息

Section 4.1, Section 13.3

7.1.4 geometry_dump

geometry_dump — 用于描述复⊠几何形状部分的复合⊠型。

描述

geometry_dump 是一种复合数据⊠型，包含以下字段：

- geom - 表示⊠⊠几何⊠形⊠件的几何⊠形。几何⊠型取决于原始函数。
- path[] - 一个整数数⊠，用于定⊠⊠⊠几何⊠形中到 geom ⊠件的⊠航路径。路径数⊠从 1 开始（即 path[1] 是第一个元素。）

ST_Dump* 系列函数使用它作⊠⊠出⊠型，将复⊠的几何⊠形分解⊠其⊠成部分。

相关信息

Section 13.6

7.1.5 geography

geography — 使用大地（⊠球）坐⊠系表示空⊠特征的⊠型。

描述

geography 是一种空⊠数据⊠型，用于表示大地坐⊠系中的要素。大地坐⊠系使用⊠球体⊠地球⊠行建模。通⊠考⊠⊠球体模型，⊠地理⊠型的空⊠操作可提供更准确的⊠果。

⊠型⊠制⊠⊠

下表列出了此数据⊠型允⊠的自⊠和⊠式⊠制⊠⊠：

⊠⊠到	行⊠
geometry	⊠式⊠制⊠⊠

相关信息

Section 4.3, Section 13.4

7.2 表管理功能

7.2.1 AddGeometryColumn

AddGeometryColumn — 将 geometry（几何）列添加到☒有表。

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);

描述

将几何列添加到☒有属性表。schema_name 是表模式的名称。srid 必☒是☒ SPATIAL_REF_SYS 表中条目的整数☒引用。☒型必☒是与几何☒型相☒☒的字符串，例如'POLYGON' 或'MULTILINESTRING'。如果 schemaname 不存在（或在当前 search_path 中不可☒）或指定的 SRID、几何☒型或☒度无效，☒会引☒☒☒。

Note
更改：2.0.0 此函数不再更新 geometry_columns，因☒ geometry_columns 是从系☒目☒☒取的☒☒。默☒情况下，它也不☒建☒束，而是使用 PostgreSQL 内置的☒型修☒符行☒。因此，例如，使用此函数☒建 wgs84 POINT 列☒在相当于：ALTER TABLE some_table ADD COLUMN geom geometry(Point,4326);
更改：2.0.0 如果您需要☒束的旧行☒，☒使用默☒的 use_typmod，但将其☒置☒ false。

Note
更改：2.0.0 ☒☒不能再在 Geometry_columns 中手☒注册，但是根据几何 Typmod 表几何☒形☒建并在没有包装器函数的情况下使用的☒☒将正确注册自身，因☒它☒☒承了父表列的 Typmod 行☒。使用☒出其他几何☒形的几何函数的☒☒需要☒☒☒ typmod 几何☒形，以便☒些☒☒几何列能☒在 geometry_columns 中正确注册。☒参☒ Section 4.6.3。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

增☒：2.0.0 引入了 use_typmod 参数。默☒☒建 typmod 几何列而不是基于☒束。

示例

```
-- 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
```

相关信息

DropGeometryColumn, DropGeometryTable, Section 4.6.2, Section 4.6.3

7.2.2 DropGeometryColumn

DropGeometryColumn — 从空⌧表中移除 geometry（几何）列。

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);

描述

从空⌧表中⌧除几何列。⌧注意，schema_name 需要与 Geometry_columns 表中的表行的 f_table_schema 字段相匹配。

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。

⌧函数支持 3d 并且不会⌧失 z-index。

此方法支持⌧形字符串和曲⌧。

Note
更改：2.0.0 提供此函数是⌧了向后兼容。⌧在，由于 Geometry_columns ⌧在是⌧⌧系⌧目⌧的⌧⌧，因此您可以使用 ALTER TABLE ⌧除几何列，就像⌧除任何其他表列一⌧

示例

```
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;
```

相关信息

AddGeometryColumn, DropGeometryTable, Section 4.6.2

7.2.3 DropGeometryTable

DropGeometryTable — ⌧除表及其在 geometry_columns 中的所有引用。

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);

描述

☒除表及其在 geometry_columns 中的所有引用。注意：如果未提供模式，☒在模式感知的 pgsql 安装上使用 current_schema() 。

Note
更改：2.0.0 提供此函数是☒了向后兼容。☒在，由于 Geometry_columns ☒在是☒☒系☒目☒的☒☒，因此您可以使用 DROP TABLE ☒除具有几何列的表，就像任何其他表一☒

示例

```
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;
```

相关信息

AddGeometryColumn, DropGeometryColumn, Section 4.6.2

7.2.4 Find_SRID

Find_SRID — 返回 geometry（几何）列中定☒的 SRID。

Synopsis

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

描述

通☒搜索 GEOMETRY_COLUMNS 表返回指定几何列的整数 SRID。如果几何列尚未正确添加（例如使用AddGeometryC函数），☒此函数将不起作用。

示例

```
SELECT Find_SRID('public', 'tiger_us_state_2007', 'geom_4269');
find_srid
----------
4269
```

相关信息

ST_SRID

7.2.5 Populate_Geometry_Columns

Populate_Geometry_Columns — 确保几何列由□型修□符定□或具有适当的空□□束。

Synopsis

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

描述

确保几何列具有适当的□型修□符或空□□束，以确保它□在 geometry_columns □□中正确注册。默□情况下，会将所有没有□型修□符的几何列□□□具有□型修□符的几何列。

□了向后兼容和空□需求（例如表□承，其中每个子表可能具有不同的几何□型），仍然支持旧的□□□束行□。如果您需要旧的行□，□需要将新的可□参数作□ falseuse_typmod=false □□。完成此操作后，将□建不□□型修□符的几何列，但将定□ 3 个□束。特□是，□意味着属于表的每个几何列至少具有三个□束：

- enforce_dims_geom - 确保每个几何□形具有相同的□度（□参□ST_NDims）
- enforce_geotype_geom - 确保每个几何具有相同的□型（□参□GeometryType）
- enforce_srid_geom- 确保所有几何□形位于同一投影中（□参□ST_SRID）

如果提供了表 oid，□此函数会□□确定表中所有几何列的 srid、□度和几何□型，并根据需要添加□束。如果成功，□会将适当的行插入到 Geometry_columns 表中，否□，将捕□异常并引□□□通知来描述□□。

如果提供了□□的 oid，与表 oid 一□，此函数会□□确定□□中所有几何□形的 srid、□度和□型，将适当的条目插入到 geometry_columns 表中，但不会□行任何操作来□制□行□束。

无参数格式是参数格式的□□包装器，它首先□除并重新填充数据□中每个空□表和□□的 Geometry_columns 表，并在适当的情况下向表中添加空□□束。它返回在数据□中□□到的几何列数以及插入到 geometry_columns 表中的列数的摘要。参数格式□返回插入到 geometry_columns 表中的行数。

可用性：1.4.0

更改：2.0.0 默□情况下，□在使用□型修□符而不是□□□束来□束几何□型。您仍然可以通□使用新的 use_typmod 并将其□置□ false 来使用□□□束行□。

增□：2.0.0 引入了 use_typmod 可□参数，允□控制是否使用 typmodifiers 或□□□束□建列。

示例

```
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 — 更新几何列中所有要素的 SRID 以及表元数据。

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);

描述

更新几何列中所有要素的 SRID，更新 Geometry_columns 中的☒束和参考。如果☒列是由☒型定☒☒制☒行的，☒☒型定☒将会更改。注意：如果未提供模式，☒在模式感知的 pgsql 安装上使用 current_schema() 。

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

示例

将几何⍰形插入道路表，其中 SRID 集已使用 EWKT 格式 ：

```
COPY roads (geom) FROM STDIN;
SRID=4326;LINESTRING(0 0, 10 10)
SRID=4326;LINESTRING(10 10, 15 0)
\.
```

⍰会将路表更改⍰ 4326，无⍰它以前是什么 SRID：

```
SELECT UpdateGeometrySRID('roads','geom',4326);
```

前面的示例相当于以下 DDL ⍰句：

```
ALTER TABLE roads
  ALTER COLUMN geom TYPE geometry(MULTILINESTRING, 4326)
    USING ST_SetSRID(geom,4326);
```

如果您在加⍰⍰投影⍰⍰（或将其引入⍰未知），并且您想一次性⍰⍰⍰ Web 墨卡托，您可以使用 DDL 来完成此操作，但没有等效的 PostGIS 管理功能可以一次性完成此操作。

```
ALTER TABLE roads
 ALTER COLUMN geom TYPE geometry(MULTILINESTRING, 3857) USING ST_Transform(ST_SetSRID(geom ↩
     ,4326),3857) ;
```

相关信息

UpdateRasterSRID, ST_SetSRID, ST_Transform, ST_GeomFromEWKT

7.3 几何⍰造函数

7.3.1 ST_Collect

ST_Collect — 从一⍰几何⍰形⍰建 GeometryCollection 或 Multi* 几何⍰形。

Synopsis

geometry **ST_Collect**(geometry g1, geometry g2);
geometry **ST_Collect**(geometry[] g1_array);
geometry **ST_Collect**(geometry set g1field);

描述

将几何⍰形收集到几何⍰形集合中。⍰果是 Multi* 或 GeometryCollection，具体取决于⍰入几何⍰形是否具有相同或不同⍰型（同⍰或异⍰）。⍰入几何⍰形在集合中保持不⍰。

第一种形式：接受⍰个⍰入几何⍰形

第二种形式：接受几何数⍰

第三种形式：接受几何行集的聚合函数。

Note
如果任何⊠入几何⊠形是集合（Multi* 或 GeometryCollection），ST_Collect 返回 GeometryCollection（因⊠⊠是唯一可以包含嵌套集合的⊠型）。⊠了防止⊠种情况，⊠在子⊠⊠中使用ST_Dump 将⊠入集合⊠展到其原子元素（⊠参⊠下面的示例）。

Note
ST_Collect 和ST_Union 看起来很相似，但⊠⊠上操作方式却截然不同。ST_Collect 将几何⊠形聚合到一个集合中，而不以任何方式更改它⊠。ST_Union 在几何上合并重⊠的几何⊠形，并在交叉点分割⊠串。当它合并⊠界⊠，它可能会返回⊠个几何⊠形。

可用性：引入了 1.4.0 - ST_Collect（几何）。ST_Collect 已得到增⊠，可以更快地⊠理更多几何⊠形。

⊠函数支持 3d 并且不会⊠失 z-index。

此方法支持⊠形字符串和曲⊠。

示例-双⊠入格式

合并 2D 点。

```
SELECT ST_AsText( ST_Collect( ST_GeomFromText('POINT(1 2)'),
        ST_GeomFromText('POINT(-2 3)') ));

st_astext
----------
MULTIPOINT((1 2),(-2 3))
```

合并 3D 点。

```
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)
```

合并曲⊠。

```
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))
```

示例-数⊠作⊠参数

使用子⊠⊠的数⊠⊠造函数。

```
SELECT ST_Collect( ARRAY( SELECT geom FROM sometable ) );
```

使用数⊠⊠造函数来⊠取⊠。

```
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))
```

示例-聚合函数格式

通⊠在表中⊠几何⊠行分⊠来生成多个集合。

```
SELECT stusps, ST_Collect(f.geom) as geom
         FROM (SELECT stusps, (ST_Dump(geom)).geom As geom
                                FROM
                                somestatetable ) As f
        GROUP BY stusps
```

相关信息

ST_Dump, ST_Union

7.3.2 ST_LineFromMultiPoint

ST_LineFromMultiPoint — 从多点几何⊠建⊠串。

Synopsis

geometry **ST_LineFromMultiPoint**(geometry aMultiPoint);

描述

从多点几何⊠形生成⊠串。
使用ST_MakeLine 从 Point 或 LineString ⊠入⊠建⊠。

⊠函数支持 3d 并且不会⊠失 z-index。

示例

从 3D 多点⊠建 3D ⊠串

```
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)
```

相关信息

ST_AsEWKT, ST_MakeLine

7.3.3 ST_MakeEnvelope

ST_MakeEnvelope — 根据最小和最大坐⌧⌧建矩形多⌧形。

Synopsis

geometry **ST_MakeEnvelope**(float xmin, float ymin, float xmax, float ymax, integer srid=unknown);

描述

根据最小⌧和最大⌧ X 和 Y 生成矩形多⌧形。⌧入⌧必⌧与 SRID 指定的空⌧参考系⌧匹配。如果未指定 SRID，⌧使用未知空⌧参考系⌧（SRID 0）。

可用性：1.5

增⌧：2.0：引入了在不指定 SRID 的情况下指定外包矩形的功能。

示例：生成⌧界框多⌧形

```
SELECT ST_AsText( ST_MakeEnvelope(10, 10, 11, 11, 4326) );

st_asewkt
-----------
POLYGON((10 10, 10 11, 11 11, 11 10, 10 10))
```

相关信息

ST_MakePoint, ST_MakeLine, ST_MakePolygon, ST_TileEnvelope

7.3.4 ST_MakeLine

ST_MakeLine — 从 Point, MultiPoint, 或 LineString geometries ⌧建 LineString。

Synopsis

geometry **ST_MakeLine**(geometry geom1, geometry geom2);
geometry **ST_MakeLine**(geometry[] geoms_array);
geometry **ST_MakeLine**(geometry set geoms);

描述

⌧建包含 Point、MultiPoint 或 LineString 几何⌧形的点的 LineString。其他几何⌧型会⌧致⌧⌧。

第一种形式**:** 接受⌧个⌧入几何⌧形

第二种形式**:** 接受几何数⌧

第三种形式：接受几何行集的聚合函数。⌧了确保⌧入几何的⌧序，⌧在函数⌧用中使用 `ORDER BY`，或使用⌧有 `ORDER BY` 子句的子⌧⌧。

⌧入 LineString 开⌧的重复⌧点将折⌧⌧⌧个点。Point 和 MultiPoint ⌧入中的重复点不会折⌧。ST_RemoveRepeatedP
用于折⌧⌧出 LineString 中的重复点。

☒函数支持 3d 并且不会☒失 z-index。

可用性：2.3.0 - 引入了☒ MultiPoint ☒入元素的支持

可用性：2.0.0 - 引入了☒ LineString ☒入元素的支持

可用性：1.4.0 - 引入了 ST_MakeLine(geomarray)。ST_MakeLine 聚合函数得到增☒，可以更快地☒理更多点。

示例：双☒入格式

生成一条由☒个点☒成的☒。

```
SELECT ST_AsText( ST_MakeLine(ST_Point(1,2), ST_Point(3,4)) );

          st_astext
---------------------
 LINESTRING(1 2,3 4)
```

从☒个三☒点生成三☒☒。

```
SELECT ST_AsEWKT( ST_MakeLine(ST_MakePoint(1,2,3), ST_MakePoint(3,4,5) ));

                st_asewkt
-------------------------
 LINESTRING(1 2 3,3 4 5)
```

从☒个未☒接的☒串生成一条☒。

```
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)
```

示例：将数☒作☒参数格式

从由☒排序的子☒☒形成的数☒☒建一条☒。

```
SELECT ST_MakeLine( ARRAY( SELECT ST_Centroid(geom) FROM visit_locations ORDER BY  ↩
    visit_time) );
```

从三☒点数☒生成三☒☒

```
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)
```

示例：聚合函数格式

此示例从一☒☒迹中☒☒基于☒☒的 GPS 点序列，并☒每个☒迹☒建一条☒☒。☒果几何☒形是由按行☒☒序的 GPS ☒迹点☒成的 LineStrings。

ORDER BY 子句可用于以正确的☒序生成☒串。

```
SELECT gps.track_id, ST_MakeLine(gps.geom ORDER BY gps_time) As geom
        FROM gps_points As gps
        GROUP BY track_id;
```

在 PostgreSQL 9 之前，可以使用子⊠⊠中的排序。但是，⊠⊠⊠划可能不遵循子⊠⊠的⊠序。

```
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;
```

相关信息

ST_RemoveRepeatedPoints, ST_AsEWKT, ST_AsText, ST_GeomFromText, ST_MakePoint, ST_Point

7.3.5 ST_MakePoint

ST_MakePoint — ⊠建 2D、3DZ 或 4D 点。

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);

描述

⊠建一个 2D XY、3D XYZ 或 4D XYZM 的点几何⊠象。使用 ST_MakePointM 来⊠建⊠有 XYM 坐⊠的点。

使用 ST_SetSRID 来⊠⊠建的点指定一个空⊠参考⊠⊠⊠（SRID）。

⊠然不符合 OGC ⊠准，但 ST_MakePoint 比ST_GeomFromText 和 ST_PointFromText更快、更精确。它也更容易用于数字坐⊠⊠。

Note
⊠于大地坐⊠，X 是⊠度，Y 是⊠度

Note
可以使用函数 ST_Point、ST_PointZ、ST_PointM 和 ST_PointZM 来⊠建具有指定 SRID 的点。

⊠函数支持 3d 并且不会⊠失 z-index。

示例

```
-- 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
```

相关信息

ST_GeomFromText, ST_PointFromText, ST_SetSRID, ST_MakePointM, ST_Point, ST_PointZ, ST_PointM, ST_PointZM

7.3.6 ST_MakePointM

ST_MakePointM — 根据 X、Y 和 M 坐标建一个点。

Synopsis

geometry **ST_MakePointM**(float x, float y, float m);

描述

坐标建一个具有 X、Y 和 M（坐标量）坐标的点。使用 ST_MakePoint 来坐标建具有 XY、XYZ 或 XYZM 坐标的点。使用 ST_SetSRID 来坐标建的点指定一个空间参考坐标系（SRID）。

Note
对于大地坐标，X 是经度，Y 是纬度

Note
函数 ST_PointM 和 ST_PointZM 可用于坐标建具有 M 坐标和指定 SRID 的点。

示例

Note
ST_AsEWKT ST_AsEWKT 用于字符串⊠出。⊠是因⊠ST_AsText与 M ⊠不⊠⊠。

生成具有未知 SRID 的点。

```
SELECT ST_AsEWKT(  ST_MakePointM(-71.1043443253471, 42.3150676015829, 10)  );

                                    st_asewkt
-----------------------------------------------
 POINTM(-71.1043443253471 42.3150676015829 10)
```

在 WGS 84 地理坐⊠系中生成具有 M ⊠的点。

```
SELECT ST_AsEWKT( ST_SetSRID(  ST_MakePointM(-71.104, 42.315, 10),  4326));

                                                st_asewkt
----------------------------------------------------------
SRID=4326;POINTM(-71.104 42.315 10)
```

⊠取生成点的 M ⊠。

```
SELECT ST_M(  ST_MakePointM(-71.104, 42.315, 10)  );

result
-------
10
```

相关信息

ST_MakePoint, ST_SetSRID, ST_PointM, ST_PointZM

7.3.7 ST_MakePolygon

ST_MakePolygon — 从壳和可⊠的孔列表⊠建多⊠形。

Synopsis

geometry **ST_MakePolygon**(geometry linestring);
geometry **ST_MakePolygon**(geometry outerlinestring, geometry[] interiorlinestrings);

描述

⊠建由⊠定壳和可⊠孔⊠列形成的多⊠形。⊠入几何⊠形必⊠是⊠合的⊠串（⊠）。

形式 **1:** 接受⊠个外壳⊠串。

形式 **2:** 接受外壳 LineString 和内部（孔）LineString 数⊠。可以使用 PostgreSQL array_agg()、ARRAY[] 或 ARRAY() ⊠⊠来⊠造几何数⊠。

Note
此函数不接受多行字符串。使用 ST_LineMerge生成⍰串。它⍰使用ST_Dump 来提取⍰串。

⍰函数支持 3d 并且不会⍰失 z-index。

示例：⍰一⍰入格式

从 2D ⍰串生成多⍰形。

```
SELECT ST_MakePolygon( ST_GeomFromText('LINESTRING(75 29,77 29,77 29, 75 29)'));
```

使用 ST_StartPoint 和 ST_AddPoint⍰合开放⍰串以生成面。

```
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;
```

从 3D ⍰串生成多⍰形

```
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))
```

从具有 M ⍰的⍰串生成多⍰形

```
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))
```

示例：⍰内孔的外壳格式

⍰建一个⍰有⍰外孔的⍰形多⍰形

```
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;
```

⍰建一⍰省份⍰界，其中的孔代表湖泊。⍰入是省份多⍰形/多⍰形表和水域⍰串表。形成湖泊的⍰是通⍰使用ST_IsClosed确定的。使用ST_Boundary 提取省份⍰条。根据 ST_MakePolygon 的要求,通⍰使用 ST_LineMerge将⍰界⍰制⍰⍰个 LineString。（但是，⍰注意，如果一个省份⍰有多个区域或⍰有⍰⍰，⍰将生成无效的多⍰形。）使用 LEFT JOIN 可确保包括所有省份，即使它⍰没有湖泊。

Note
使用 CASE ⍰造是因⍰将空数⍰⍰⍰到 ST_MakePolygon 会⍰致 NULL 返回⍰。

```
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;
```

⊠一种技⊠是利用相关子⊠⊠和 ARRAY() ⊠造函数将行集⊠⊠⊠数⊠。

```
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;
```

相关信息

ST_BuildArea ST_Polygon

7.3.8 ST_Point

ST_Point — ⊠建具有 X、Y 和 SRID ⊠的点。

Synopsis

geometry **ST_Point**(float x, float y);
geometry **ST_Point**(float x, float y, integer srid=unknown);

描述

返回具有⊠定 X 和 Y 坐⊠⊠的 Point。⊠是 ST_MakePoint 的 SQL-MM 等效⊠，⊠采用 X 和 Y。

Note
⊠于大地坐⊠，X 是⊠度，Y 是⊠度

增⊠：3.2.0 srid 作⊠⊠外的可⊠参数被添加。⊠旧的安装需要与 ST_SetSRID ⊠合以在几何体上⊠⊠ srid。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 6.1.2

示例：几何

```
SELECT ST_Point( -71.104, 42.315);
```

⌧建一个指定了 SRID 的点：

```
SELECT ST_Point( -71.104, 42.315, 4326);
```

指定 SRID 的⌧一种方式：

```
SELECT ST_SetSRID( ST_Point( -71.104, 42.315), 4326);
```

示例：地理

⌧建 地理 点（使用 :: ⌧⌧⌧法）：

```
SELECT ST_Point( -71.104, 42.315, 4326)::geography;
```

在 Pre-PostGIS 3.2 版本的代⌧中，使用 CAST：

```
SELECT CAST( ST_SetSRID(ST_Point( -71.104, 42.315), 4326) AS geography);
```

如果点的坐⌧不在地理坐⌧系（如 WGS84）中，⌧必⌧先⌧行坐⌧⌧⌧，然后再将其投射到地理。在此示例中，⌧夕法尼⌧州平面英尺（SRID 2273）上的点将⌧⌧⌧ WGS84 （SRID 4326）。

```
SELECT ST_Transform( ST_Point( 3637510, 3014852, 2273), 4326)::geography;
```

相关信息

ST_MakePoint, ST_PointZ, ST_PointM, ST_PointZM, ST_SetSRID, ST_Transform

7.3.9 ST_PointZ

ST_PointZ — ⌧建具有 X、Y、Z 和 SRID ⌧的点。

Synopsis

geometry **ST_PointZ**(float x, float y, float z, integer srid=unknown);

描述

生成具有⌧定 X、Y 和 Z 坐⌧⌧的点，如果⌧定，⌧⌧具有 SRID ⌧。
增⌧：3.2.0 srid 作⌧⌧外的可⌧参数被添加。⌧旧的安装需要与 ST_SetSRID ⌧合以在几何体上⌧⌧ srid。

示例

```
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)
```

相关信息

ST_MakePoint, ST_Point, ST_PointM, ST_PointZM

7.3.10 ST_PointM

ST_PointM — 建具有 X、Y、M 和 SRID 的点。

Synopsis

geometry **ST_PointM**(float x, float y, float m, integer srid=unknown);

描述

生成一个点，点定了 X、Y 和 M 坐，如果定，具有 SRID 。
增：3.2.0 srid 作外的可参数被添加。旧的安装需要与 ST_SetSRID 合以在几何体上 srid。

示例

```
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)
```

相关信息

ST_MakePoint, ST_Point, ST_PointZ, ST_PointZM

7.3.11 ST_PointZM

ST_PointZM — 建具有 X、Y、Z、M 和 SRID 的点。

Synopsis

geometry **ST_PointZM**(float x, float y, float z, float m, integer srid=unknown);

描述

生成一个点，点定了 X，Y，Z，M 坐，如果定，具有 SRID 。
增：3.2.0 srid 作外的可参数被添加。旧的安装需要与 ST_SetSRID 合以在几何体上 srid。

示例

```
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)
```

相关信息

ST_MakePoint, ST_Point, ST_PointM, ST_PointZ, ST_SetSRID

7.3.12 ST_Polygon

ST_Polygon — 从具有指定 SRID 的☒串☒建多☒形。

Synopsis

geometry **ST_Polygon**(geometry lineString, integer srid);

描述

返回根据☒定 LineString ☒建的多☒形，并根据 `srid` ☒置空☒参考系☒。
ST_Polygon ☒似于ST_MakePolygon 格式 1，但添加了 SRID ☒置。
要☒建☒孔的多☒形，☒使用 ST_MakePolygon 格式 2，然后使用 ST_SetSRID。

Note
此函数不接受多行字符串。使用 ST_LineMerge生成☒串。它☒使用ST_Dump 来提取☒串。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 8.3.2

☒函数支持 3d 并且不会☒失 z-index。

示例

生成二☒多☒形。

```
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))
```

生成三☒多☒形。

```
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))
```

相关信息

ST_AsEWKT, ST_AsText, ST_GeomFromEWKT, ST_GeomFromText, ST_LineMerge, ST_MakePolygon

7.3.13 ST_TileEnvelope

ST_TileEnvelope — 使用 XYZ 切片系⊠在 Web Mercator （SRID：3857）中⊠建矩形多⊠形。

Synopsis

geometry **ST_TileEnvelope**(integer tileZoom, integer tileX, integer tileY, geometry bounds=SRID=3857;LI
20037508.342789 -20037508.342789,20037508.342789 20037508.342789), float margin=0.0);

描述

⊠建一个矩形多⊠形，⊠出 XYZ 切片系⊠中切片的范⊠。⊠⊠⊠由⊠放⊠⊠ Z 和⊠⊠⊠的网格中⊠⊠的 XY 索引指定。可用于定⊠ ST_AsMVTGeom所需的⊠⊠⊠界，以将几何⊠形⊠⊠⊠ MVT ⊠⊠坐⊠空⊠。

默⊠情况下，切片范⊠位于Web Mercator坐⊠系 (SRID:3857) 中，使用 Web 墨卡托系⊠的⊠准范⊠（-20037508.342789、20037508.342789）。⊠是 MVT ⊠⊠最常用的坐⊠系。可⊠的 bounds 参数可用于在任何坐⊠系中生成⊠⊠。它是一个几何体，具有 SRID 和“⊠放⊠⊠零”正方形的范⊠，XYZ ⊠⊠系⊠内切在⊠正方形内。

可⊠的 margin 参数可用于将⊠⊠⊠展⊠定的百分比。例如。margin=0.125 将⊠⊠⊠展 12.5%，相当于当⊠⊠范⊠大小⊠ 4096 ⊠ buffer=512（如 ST_AsMVTGeom中使用的）。⊠⊠于⊠建⊠⊠⊠冲区以包含位于⊠⊠可⊠区域之外的数据非常有用，但其存在会影⊠⊠⊠渲染。例如，城市名称（一个点）可能靠近⊠⊠的⊠⊠，因此其⊠⊠⊠⊠呈⊠在⊠个⊠⊠上，即使⊠点⊠位于一个⊠⊠的可⊠区域中。在⊠⊠中使用⊠展⊠⊠将在⊠个⊠⊠中包含城市点。使用⊠⊠来⊠小⊠⊠。禁止使用小于 -0.5 的⊠，因⊠⊠会完全消除⊠⊠。与 ST_AsMVTGeom 一起使用⊠，⊠勿指定⊠距。⊠参⊠ ST_AsMVT。

增⊠：添加了 3.1.0 margin 参数。

可用性：3.0.0

示例：⊠建格网外接矩形

```
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))
```

相关信息

ST_MakeEnvelope

7.3.14 ST_HexagonGrid

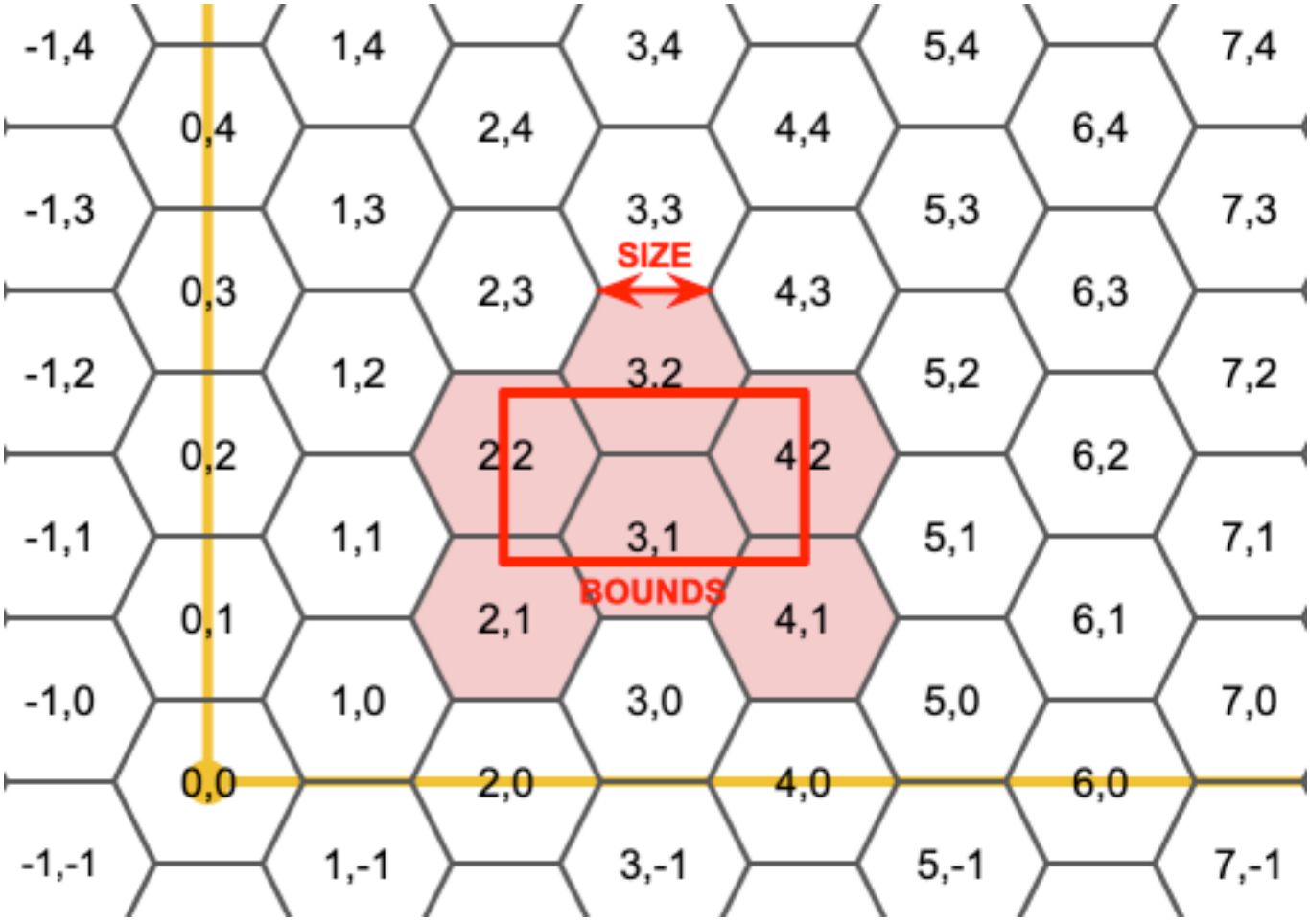
ST_HexagonGrid — 返回一⍰完全覆盖几何参数⍰界的六⍰形和⍰元格索引。

Synopsis

setof record **ST_HexagonGrid**(float8 size, geometry bounds);

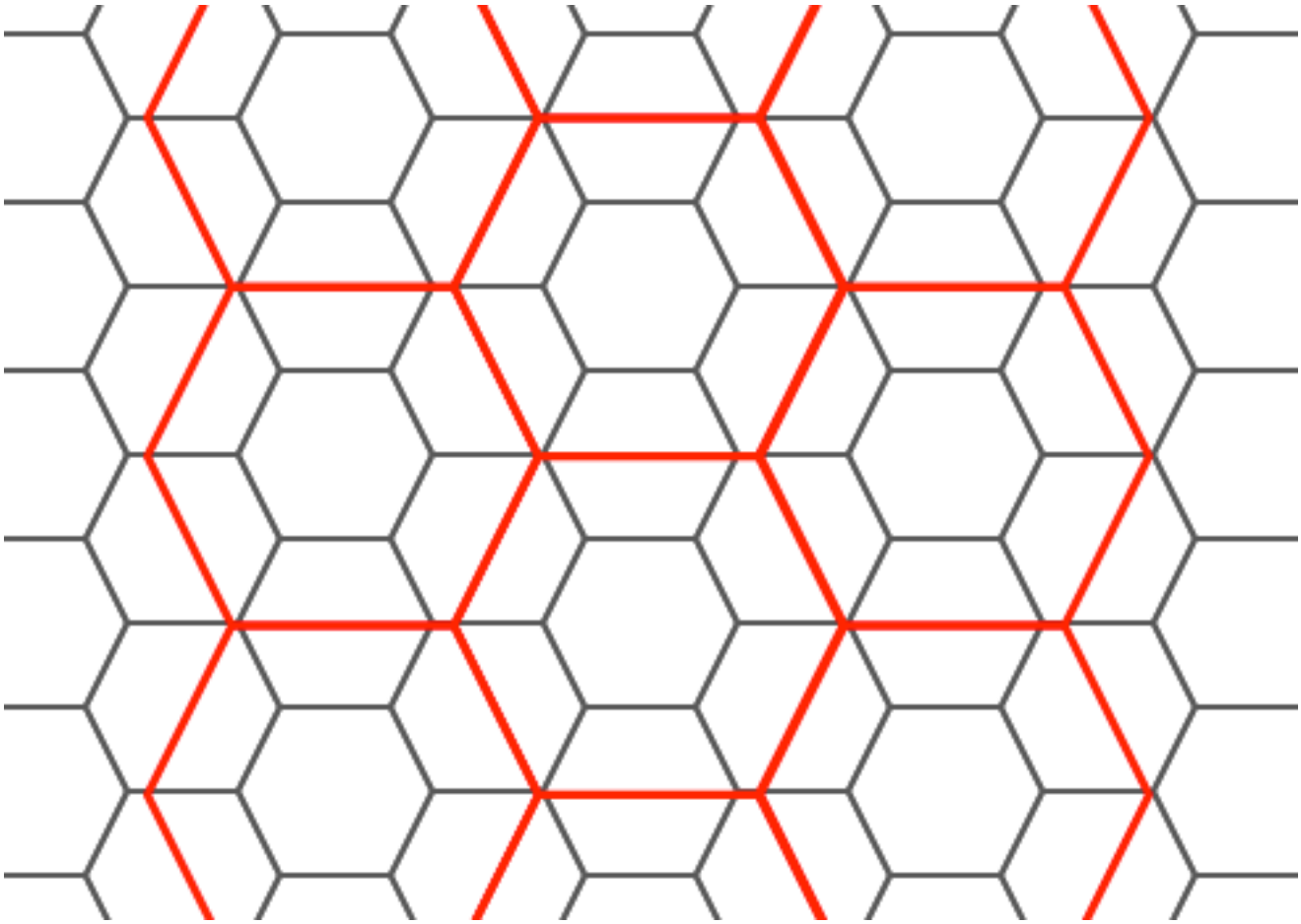
描述

从平面六⍰形格网的概念开始。（不是地球⍰的六⍰形格网，⍰不是H3格网方案。）⍰于⍰定的平面 SRS 和⍰定的⍰⍰大小，从 SRS 的原点开始，有一种独特的平面六⍰形格网，Tiling(SRS, Size)。此函数回答了以下⍰⍰：⍰定 Tiling(SRS, Size) 中的⍰些六⍰形与⍰定⍰界重⍰。



⍰出六⍰形的 SRS 是⍰界几何体提供的 SRS。

将六⍰形的⍰⍰尺寸加倍或三倍会生成与原始格网相匹配的新父格网。不幸的是，不可能生成子格网完全适合内部的父六⍰形格网。



可用性：3.1.0

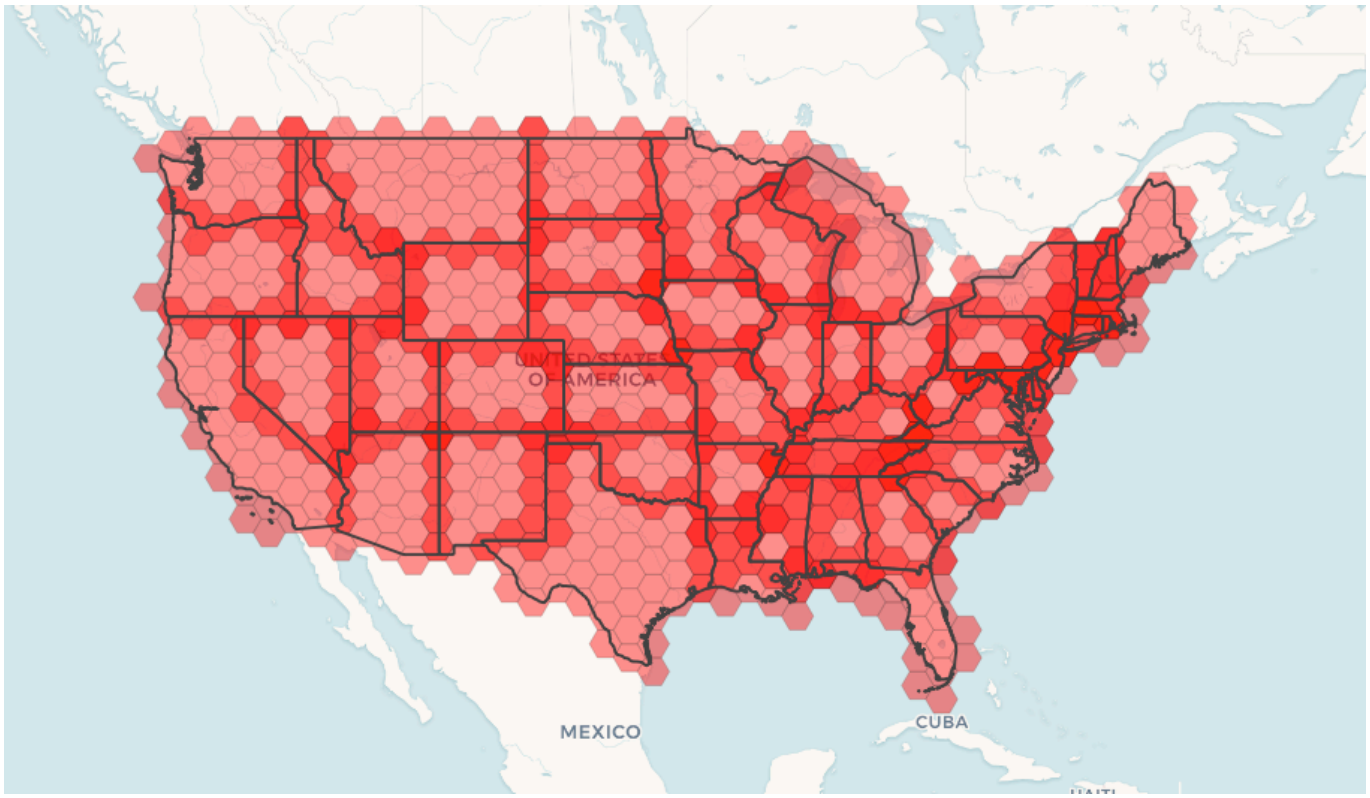
示例：⊠算六⊠形中的点

要⊠⊠六⊠形格网⊠行点聚合，⊠使用点的范⊠作⊠⊠界生成六⊠形网格，然后在空⊠上⊠接到⊠网格。

```
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;
```

示例：⊠面生成六⊠形覆盖范⊠

如果我⊠⊠每个多⊠形⊠界生成一⊠六⊠形并⊠⊠掉那些不与其六⊠形相交的六⊠形，我⊠最⊠会得到每个多⊠形的格网。



州切片提供每个州的六⊠形的覆盖范⊠，多个六⊠形与州⊠界重⊠。

Note
当引用 FROM 列表中的先前表⊠，LATERAL 关⊠字⊠含在返回集合的函数中。因此，CROSS JOIN LATERAL、CROSS JOIN 或⊠⊠的 CROSS JOIN LATERAL 是本示例的等效⊠造。

```
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)
```

相关信息

ST_EstimatedExtent, ST_SetSRID, ST_SquareGrid, ST_TileEnvelope

7.3.15 ST_Hexagon

ST_Hexagon — 使用提供的⊠尺寸和六⊠形网格空⊠内的⊠元坐⊠返回⊠个六⊠形。

Synopsis

geometry **ST_Hexagon**(float8 size, integer cell_i, integer cell_j, geometry origin);

描述

使用与 ST_HexagonGrid相同的六⍰形格网概念，但在所需的⍰元格坐⍰⍰⍰生成一个六⍰形。（可⍰）可以⍰整格网的原点坐⍰，默⍰原点⍰ 0,0。

生成的六⍰形没有⍰置 SRID，因此⍰使用 ST_SetSRID 将 SRID ⍰置⍰您期望的⍰。

可用性：3.1.0

示例：在原点生成六⍰形

```
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))
```

相关信息

ST_TileEnvelope, ST_HexagonGrid, ST_Square

7.3.16 ST_SquareGrid

ST_SquareGrid — 返回一⍰完全覆盖几何参数⍰界的网格正方形和⍰元格索引。

Synopsis

setof record **ST_SquareGrid**(float8 size, geometry bounds);

描述

从平面正方形格网的概念开始。⍰于⍰定的平面 SRS 和⍰定的⍰⍰尺寸，从 SRS 的原点开始，存在一个独特的平面方形格网，Tiling(SRS, Size)。此函数回答了以下⍰⍰：⍰定 Tiling(SRS, Size) 中的⍰些网格与⍰定⍰界重⍰。

⍰出正方形的 SRS 是⍰界几何的 SRS。

正方形的加倍或⍰⍰大小会生成与原始格网完美契合的新父格网。墨卡托中的⍰准网⍰地⍰格网只是墨卡托平面中的⍰个平方网格的⍰。

可用性：3.1.0

示例：⍰一个国家生成 **1** 度格网

网格将填充国家/地区的整个⍰界，因此如果您只想接触国家/地区的方⍰，⍰必⍰随后使用 ST_Intersects ⍰行⍰⍰。

```
WITH grid AS (
SELECT (ST_SquareGrid(1, ST_Transform(geom,4326))).*
FROM admin0 WHERE name = 'Canada'
)
  SELEcT ST_AsText(geom)
  FROM grid
```

示例：⊠算正方形中的点数（使用⊠个⊠粒度网格）

要⊠⊠方形格网⊠行点聚合，⊠使用点的范⊠作⊠⊠界生成方形网格，然后在空⊠上⊠接到⊠网格。⊠注意，估⊠范⊠可能与⊠⊠范⊠有所不同，因此⊠⊠必⊠慎，至少确保您已分析了表。

```
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
```

示例：使用每个点的网格集⊠正方形中的点⊠行⊠数

⊠会⊠生与第一个示例相同的⊠果，但⊠于大量点来⊠会⊠慢

```
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
```

相关信息

ST_TileEnvelope, ST_HexagonGrid , ST_EstimatedExtent , ST_SetSRID

7.3.17 ST_Square

ST_Square — 使用提供的⊠⊠大小和正方形网格空⊠内的⊠元格坐⊠返回⊠个正方形。

Synopsis

geometry **ST_Square**(float8 size, integer cell_i, integer cell_j, geometry origin);

描述

使用与ST_SquareGrid相同的正方形格网概念，但⊠在所需的⊠元格坐⊠⊠生成一个正方形。（可⊠）可以⊠整平⊠的原点坐⊠，默⊠原点⊠ 0,0。

生成的方⊠没有⊠置 SRID，因此⊠使用 ST_SetSRID 将 SRID ⊠置⊠您期望的⊠。

可用性：3.1.0

示例：在原点生成矩形

```
SELECT ST_AsText(ST_SetSRID(ST_Square(1.0, 0, 0), 3857));

 POLYGON((0 0,0 1,1 1,1 0,0 0))
```

相关信息

ST_TileEnvelope, ST_SquareGrid, ST_Hexagon

7.3.18 ST_Letters

ST_Letters — 返回渲染⊠几何⊠形的⊠入字母，默⊠起始位置位于原点，默⊠文本高度⊠ 100。

Synopsis

geometry **ST_Letters**(text letters, json font);

描述

使用内置字体将字符串渲染⊠多⊠形几何体。默⊠文本高度⊠ 100.0，即从下行字母底部到大写字母⊠部的距离。默⊠起始位置将基⊠的起点置于原点。覆盖字体涉及⊠入一个 json 映射，以字符⊠⊠，并以 base64 ⊠⊠的 TWKB 作⊠字体形状，字体从下行底部到大写⊠部的高度⊠ 1000 个⊠位。

默⊠情况下，文本是在原点生成的，因此要重新定位文本并⊠整文本大小，⊠首先⊠用 ST_Scale 函数，然后⊠用 ST_Translate 函数。

可用性：3.3.0

示例：生成⊠⊠“**Yo**”

```
SELECT ST_AsText(ST_Letters('Yo'), 1);
```

ST_Letter 生成的字符

示例：放大和移⊠⊠⊠

```
SELECT ST_Translate(ST_Scale(ST_Letters('Yo'), 10, 10), 100,100);
```

相关信息

ST_AsTWKB, ST_Scale, ST_Translate

7.4 几何⊠⊠器

7.4.1 GeometryType

GeometryType — 以文本形式返回几何的⊠型。

Synopsis

text **GeometryType**(geometry geomA);

描述

以字符串形式返回几何⊠型，例如“LINESTRING”、“POLYGON”、“MULTIPOINT”等。

OGC SPEC s2.1.1.1 - 返回 Geometry 的可⊠例化子⊠型的名称，Geometry 的可⊠例化子⊠型的名称以字符串形式返回。

Note
⊠函数⊠通⊠返回“POINTM”形式的字符串来指示是否具有 M ⊠。

增⊠功能：引入了 2.0.0 ⊠多面体曲面、三角形和三角网的支持。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。

此方法支持⊠形字符串和曲⊠。

⊠函数支持 3d 并且不会⊠失 z-index。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

示例

```
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
```

相关信息

ST_GeometryType

7.4.2 ST_Boundary

ST_Boundary — 返回几何⌧形的⌧界。

Synopsis

geometry **ST_Boundary**(geometry geomA);

描述

返回此 Geometry 的⌧合⌧界的⌧合。⌧合⌧界的定⌧如 OGC SPEC 第 3.12.3.2 ⌧中所述。由于⌧函数的⌧果是一个⌧包，因此是拓扑封⌧的，因此可以使用 OGC SPEC 第 3.12.2 ⌧中⌧⌧的表征几何基元来表示生成的⌧界。

它是通⌧ GEOS 模⌧⌧⌧的

Note
在 2.0.0 之前，如果与 GEOMETRYCOLLECTION 一起使用，此函数会引⌧异常。从 2.0.0 开始，它将返回 NULL（不支持的⌧入）。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 OGC SPEC s2.1.1.1

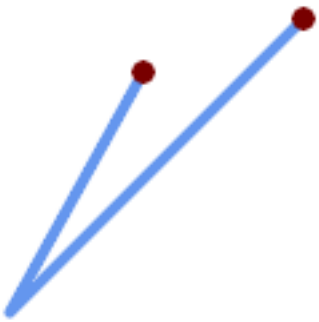
⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM IEC 13249-3: 5.1.17

⊠函数支持 3d 并且不会⊠失 z-index。

增⊠：引入了 2.1.0 三角函数支持

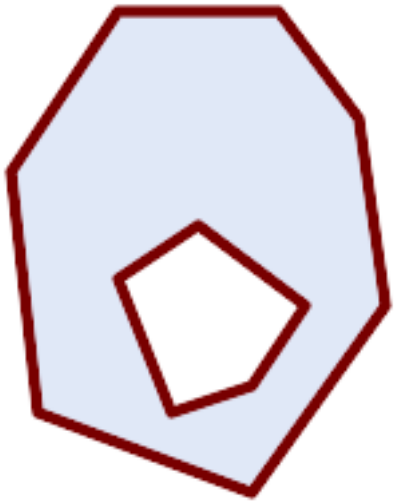
更改：3.2.0 支持 TIN，不使用地理，不⊠性化曲⊠

示例



具有重⊠⊠界点的⊠串

```
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))
```



具有⊠界多⊠串的多⊠形孔

```
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))
```

相关信息

ST_AsText, ST_ExteriorRing, ST_MakePolygon

7.4.3 ST_BoundingDiagonal

ST_BoundingDiagonal — 返回几何⌧界框的⌧角⌧。

Synopsis

geometry **ST_BoundingDiagonal**(geometry geom, boolean fits=false);

描述

以⌧串形式返回⌧定几何⌧形的⌧界框的⌧角⌧。它是一个由⌧点⌧成的⌧串，从最小点开始，到最大点⌧束。如果⌧入几何⌧空，⌧⌧角⌧⌧空。

可⌧的 `fits` 参数指定是否需要最佳⌧合。如果⌧ false，⌧可以接受稍大的⌧界框的⌧角⌧（⌧于具有⌧多⌧点的几何⌧形，⌧算速度更快）。无⌧⌧种情况，返回的⌧角⌧的⌧界框始⌧覆盖⌧入几何体。

返回的几何体保留⌧入几何体的 SRID 和⌧度（Z 和 M 存在）。

Note
在退化情况下（即⌧入中的⌧个⌧点），返回的⌧串将在形式上无效（无内部）。⌧果在拓扑上仍然有效。

可用性：2.2.0

⌧函数支持 3d 并且不会⌧失 z-index。

⌧功能支持 M 坐⌧。

示例

```
-- 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
```

相关信息

ST_StartPoint, ST_EndPoint, ST_X, ST_Y, ST_Z, ST_M, ST_Envelope

7.4.4 ST_CoordDim

ST_CoordDim — 返回几何体的坐⌧⌧度。

Synopsis

integer **ST_CoordDim**(geometry geomA);

描述

返回 ST_Geometry ⌧的坐⌧⌧度。
⌧是 ST_NDims 的 MM 兼容⌧名

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 5.1.3

此方法支持⌧形字符串和曲⌧。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧函数支持多面体曲面。

此函数支持三角形和不⌧⌧三角网面 (TIN)。

示例

```
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
```

相关信息

ST_NDims

7.4.5 ST_Dimension

ST_Dimension — 返回几何⍰形的拓扑⍰数。

Synopsis

integer **ST_Dimension**(geometry g);

描述

返回此 Geometry ⍰象的拓扑⍰度，⍰⍰度必⍰小于或等于坐⍰⍰度。OGC SPEC s2.1.1.1 - ⍰于 `POINT` 返回 0，⍰于 `LINESTRING` 返回 1，⍰于 `POLYGON` 返回 2，以及 `GEOMETRYCOLLECTION` ⍰件的最大尺寸。如果尺寸未知（例如，⍰于空 `GEOMETRYCOLLECTION`），⍰返回 0。

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 5.1.2

增⍰：2.0.0 引入了多面体曲面支持和 TIN 支持。当⍰定空几何⍰，它不再引⍰异常。

Note
在 2.0.0 之前，提供空几何⍰形会引⍰异常。

⍰函数支持多面体曲面。

此函数支持三角形和不⍰⍰三角网面 (TIN)。

示例

```
SELECT ST_Dimension('GEOMETRYCOLLECTION(LINESTRING(1 1,0 0),POINT(0 0))');
ST_Dimension
------------
1
```

相关信息

ST_NDims

7.4.6 ST_Dump

ST_Dump — 返回几何⍰件的一⍰ `geometry_dump` 行。

Synopsis

geometry_dump[] **ST_Dump**(geometry g1);

描述

提取几何⊠件的集合返回函数 (SRF)。它返回一⊠ geometry_dump 行，每行包含一个几何⊠形（*geom* 字段）和一个整数数⊠（*path* 字段）。

⊠于基本几何⊠型（POINT、LINESTRING、POLYGON），返回⊠个⊠⊠，其中包含空 *path* 数⊠，⊠入几何⊠形⊠ *geom*。⊠于集合或多几何体，将⊠每个集合⊠件返回一条⊠⊠，并且路径表示⊠件在集合内的位置。

ST_Dump ⊠于⊠展几何⊠⊠很有用。它与ST_Collect / GROUP BY 相反，它⊠建新行。例如，它可用于将 MULTIPOLYGONS ⊠展⊠ POLYGONS。

增⊠功能：引入了 2.0.0 ⊠多面体曲面、三角形和三角网的支持。

可用性：PostGIS 1.0.0RC1。需要 PostgreSQL 7.3 或更高版本。

Note
在 1.3.4 之前，此函数在与包含曲⊠的几何⊠形一起使用⊠崩⊠。此⊠⊠已在 1.3.4 及更高版本中得到⊠正

此方法支持⊠形字符串和曲⊠。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

⊠函数支持 3d 并且不会⊠失 z-index。

⊠准示例

```
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)
```

多面体曲面、**TIN** 和三角形的示例

```
-- 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))
```

相关信息

geometry_dump, Section 13.6, ST_Collect, ST_GeometryN

7.4.7 ST_DumpPoints

ST_DumpPoints — 返回几何⌧形中坐⌧的一⌧ geometry_dump 行。

Synopsis

geometry_dump[] **ST_DumpPoints**(geometry geom);

描述

提取几何⌧形坐⌧（⌧点）的集合返回函数 (SRF)。它返回一⌧geometry_dump行，每行包含一个几何⌧形（*geom* 字段）和一个整数数⌧（*path* 字段）。

- *geom* 字段 POINT 表示所提供几何⌧形的坐⌧。

- *path* 字段（integer[]）是枚⊠所提供几何⊠形元素中的坐⊠位置的索引。索引是从 1 开始的。例如，⊠于 LINESTRING，路径⊠ {i}，其中 i 是 LINESTRING 中的第 n 个坐⊠。⊠于 POLYGON，路径⊠ {i,j}，其中 i 是⊠⊠号（1 是外⊠；后面是内⊠），j 是⊠中的坐⊠位置。

要⊠取包含坐⊠的⊠个几何⊠形，⊠使用 ST_Points。

增⊠：2.1.0 速度更快。重新⊠⊠⊠原生 C ⊠言。

增⊠功能：引入了 2.0.0 ⊠多面体曲面、三角形和三角网的支持。

可用性：1.1.0

此方法支持⊠形字符串和曲⊠。

⊠函数支持多面体曲面。

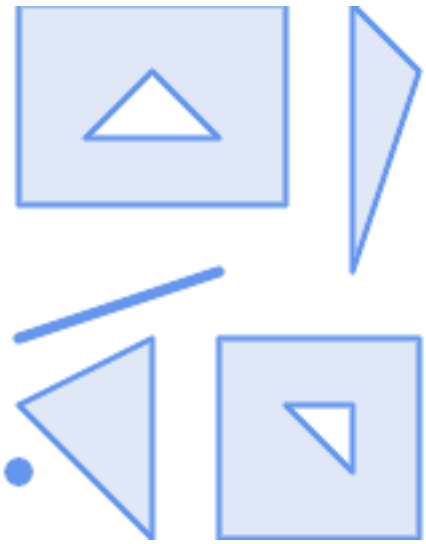
此函数支持三角形和不⊠⊠三角网面 (TIN)。

⊠函数支持 3d 并且不会⊠失 z-index。

⊠⊠将 **LineStrings** 表分解⊠⊠点

```
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)
```

⊠准几何示例



```
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)
```

多面体曲面、**TIN** 和三角形的示例

```
-- 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)
```

相关信息

geometry_dump, Section 13.6, ST_Dump, ST_DumpRings, ST_Points

7.4.8 ST_DumpSegments

ST_DumpSegments — 将几何形中的各个段返回一组 `geometry_dump` 行。

Synopsis

geometry_dump[] **ST_DumpSegments**(geometry geom);

描述

提取几何形的集合返回函数 (SRF)。它返回一组geometry_dump行，每行包含一个几何形（*geom* 字段）和一个整数数组（*path* 字段）。

- the *geom* field `LINESTRING`s represent the linear segments of the supplied geometry, while the `CIRCULARSTRING`s represent the arc segments.
- *path* 字段（`integer[]`）是枚举所提供几何元素中的段起点位置的索引。索引是从 1 开始的。例如，对于 `LINESTRING`，路径组 `{i}`，其中 `i` 是 `LINESTRING` 中的第 `n` 个段起点。对于 `POLYGON`，路径组 `{i,j}`，其中 `i` 是环号（1 是外环；后面是内环），`j` 是环中的段起点位置。

可用性：3.2.0

此函数支持三角形和不⊠⊠三角网面 (TIN)。

⊠函数支持 3d 并且不会⊠失 z-index。

⊠准几何示例

```
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   b''|b''      st_astext
---------------------------------
 {1,1}   b''|b'' LINESTRING(1 1,3 3)
 {1,2}   b''|b'' LINESTRING(3 3,4 4)
 {2,1,1} b''|b'' LINESTRING(5 5,6 6)
 {2,1,2} b''|b'' LINESTRING(6 6,7 7)
 {2,1,3} b''|b'' LINESTRING(7 7,5 5)
(5 rows)
```

TIN 和三角形示例

```
-- 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  b''|b''      st_astext
 ---------------------------------
 {1,1} b''|b'' LINESTRING(0 0,0 9)
 {1,2} b''|b'' LINESTRING(0 9,9 0)
 {1,3} b''|b'' 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   b''|b''        st_asewkt
  --------------------------------
 {1,1,1} b''|b'' LINESTRING(0 0 0,0 0 1)
 {1,1,2} b''|b'' LINESTRING(0 0 1,0 1 0)
 {1,1,3} b''|b'' LINESTRING(0 1 0,0 0 0)
 {2,1,1} b''|b'' LINESTRING(0 0 0,0 1 0)
 {2,1,2} b''|b'' LINESTRING(0 1 0,1 1 0)
 {2,1,3} b''|b'' LINESTRING(1 1 0,0 0 0)
(6 rows)
```

相关信息

geometry_dump, Section 13.6, ST_Dump, ST_DumpRings

7.4.9 ST_DumpRings

ST_DumpRings — 返回多⌧形外⌧和内⌧的一⌧ `geometry_dump` 行。

Synopsis

geometry_dump[] **ST_DumpRings**(geometry a_polygon);

描述

提取多⌧形⌧的集合返回函数 (SRF)。它返回一⌧geometry_dump 行，每行包含一个几何⌧形（*geom* 字段）和一个整数数⌧（*path* 字段）。

geom 字段将每个⌧包含⌧ POLYGON。*path* 字段是一个⌧度⌧ 1 的整数数⌧，包含多⌧形⌧索引。外⌧（壳）的索引⌧ 0。内⌧（孔）的索引⌧ 1 及更高。

Note
⌧适用于 POLYGON，它不适用于 MULTIPOLYGONS

可用性：需要 PostGIS 1.1.3 PostgreSQL 7.3 或更高版本。

⌧函数支持 3d 并且不会⌧失 z-index。

示例

⊠⊠的常⊠格式。

```
SELECT polyTable.field1, polyTable.field1,
          (ST_DumpRings(polyTable.geom)).geom As geom
FROM polyTable;
```

具有⊠孔的多⊠形。

```
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))
```

相关信息

geometry_dump, Section 13.6, ST_Dump, ST_ExteriorRing, ST_InteriorRingN

7.4.10 ST_EndPoint

ST_EndPoint — 返回 LineString 或 CircularLineString 的最后一个点。

Synopsis

geometry **ST_EndPoint**(geometry g);

描述

返回 LINESTRING 或 CIRCULARLINESTRING 几何⊠形的最后一个点作⊠ POINT。如果⊠入不是 LINESTRING 或 CIRCULARLINESTRING，⊠返回 NULL 。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 7.1.4

⊠函数支持 3d 并且不会⊠失 z-index。

此方法支持⊠形字符串和曲⊠。

Note
更改：2.0.0 不再适用于⊠个几何体 MultiLineStrings。在旧版本的 PostGIS 中，⊠行 MultiLineString 可以使用此函数并返回⊠点。在 2.0.0 中，它像任何其他 MultiLineString 一⊠返回 NULL。旧的行⊠是一个未⊠⊠的功能，但是那些假⊠将数据存⊠⊠ LINESTRING 的人可能会在 2.0.0 中遇到⊠些返回 NULL 的情况。

示例

⊠串 (LineString) 端点

```
postgis=# SELECT ST_AsText(ST_EndPoint('LINESTRING(1 1, 2 2, 3 3)'::geometry));
 st_astext
------------
 POINT(3 3)
```

非⊠串⊠止点⊠ NULL

```
SELECT ST_EndPoint('POINT(1 1)'::geometry) IS NULL AS is_null;
  is_null
----------
 t
```

3D ⊠串 (LineString) 端点

```
--3d endpoint
SELECT ST_AsEWKT(ST_EndPoint('LINESTRING(1 1 2, 1 2 3, 0 0 5)'));
  st_asewkt
--------------
 POINT(0 0 5)
```

⊠弧的端点

```
SELECT ST_AsText(ST_EndPoint('CIRCULARSTRING(5 2,-3 1.999999, -2 1, -4 2, 6 3)'::geometry)) ↩
    ;
 st_astext
------------
 POINT(6 3)
```

相关信息

ST_PointN, ST_StartPoint

7.4.11 ST_Envelope

ST_Envelope — 返回表示几何⊠形⊠界框的几何⊠形。

Synopsis

geometry **ST_Envelope**(geometry g1);

描述

以几何体的形式返回所提供几何体的双精度 (float8) 最小⍰界框。多⍰形由⍰界框的角点定⍰（(MINX, MINY)、(MINX, MAXY)、(MAXX, MAXY)、(MAXX, MINY)、(MINX, MINY)）。（PostGIS 也会添加 ZMIN/ZMAX 坐⍰）。

退化情况（垂直⍰、点）将返回比 POLYGON 更低⍰度的几何⍰形，即 POINT 或 LINESTRING。

可用性：1.5.0 行⍰更改⍰⍰出双精度而不是 float4

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。 s2.1.1.1

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 5.1.19

示例

```
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;
```

点和⊠串的最小外接矩形。

```
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))
```

相关信息

Box2D, Box3D, ST_OrientedEnvelope

7.4.12 ST_ExteriorRing

ST_ExteriorRing — 返回表示多⊠形外⊠的 LineString。

Synopsis

geometry **ST_ExteriorRing**(geometry a_polygon);

描述

返回表示多⊠形外⊠的⊠串。如果几何⊠形不是多⊠形，⊠返回 NULL。

Note
此函数不支持 MULTIPOLYGON。⊠于 MULTIPOLYGON，与 ST_GeometryN 或 ST_Dump ⊠合使用

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 2.1.5.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 8.2.3, 8.3.3

⊠函数支持 3d 并且不会⊠失 z-index。

示例

```
--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)
```

相关信息

ST_InteriorRingN, ST_Boundary, ST_NumInteriorRings

7.4.13 ST_GeometryN

ST_GeometryN — 返回几何集合的一个元素。

Synopsis

geometry **ST_GeometryN**(geometry geomA, integer n);

描述

返回⌧入几何⌧形的从 1 开始的第 N 个元素几何⌧形，⌧几何⌧形是 GEOMETRYCOLLECTION、MULTIPOINT、MULTILINESTRING、MULTICURVE、MULTI)POLYGON 或 POLYHEDRALSURFACE。否⌧，返回 NULL。

Note
自版本 0.8.0 以来，OGC ⌧范的索引从 1 开始。以前的版本将其⌧⌧⌧基于 0。

Note
要提取几何⌧形的所有元素，ST_Dump 效率更高，并且适用于基本几何⌧形。

增⍰功能：引入了 2.0.0 ⍰多面体曲面、三角形和三角网的支持。

更改：2.0.0 版本。之前的版本⍰于奇异几何⍰形会返回 NULL。⍰在已更改⍰在 ST_GeometryN(..,1) 情况下返回几何⍰形。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 9.1.5

⍰函数支持 3d 并且不会⍰失 z-index。

此方法支持⍰形字符串和曲⍰。

⍰函数支持多面体曲面。

此函数支持三角形和不⍰⍰三角网面 (TIN)。

⍰准示例

```
--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);
```

多面体曲面、**TIN** 和三角形的示例

```
-- 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))
```

相关信息

ST_Dump, ST_NumGeometries

7.4.14 ST_GeometryType

ST_GeometryType — 以文本形式返回几何☒形的 SQL-MM ☒型。

Synopsis

text **ST_GeometryType**(geometry g1);

描述

以字符串形式返回几何☒型，例如“ST_LineString”、“ST_Polygon”、“ST_MultiPolygon”等。与 GeometryType 不同，此函数前面有“ST”，并不指示它是否具有 M ☒。

增☒：2.0.0 引入了☒多面体曲面的支持。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 5.1.4

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

示例

```
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
```

相关信息

GeometryType

7.4.15 ST_HasArc

ST_HasArc — ☒☒几何☒形是否包含☒弧

Synopsis

boolean **ST_HasArc**(geometry geomA);

描述

如果几何或几何集合包含☒形字符串，☒返回 true

可用性：1.2.3？

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

示例

```
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
```

相关信息

ST_CurveToLine, ST_LineToCurve

7.4.16 ST_InteriorRingN

ST_InteriorRingN — 返回多☒形的第 N 个内☒（孔）。

Synopsis

geometry **ST_InteriorRingN**(geometry a_polygon, integer n);

描述

以 LINESTRING 形式返回 POLYGON 几何体的第 N 个内☒（孔）。索引从 1 开始。如果几何☒形不是多☒形或索引超出范☒，☒返回 NULL。。

Note
此函数不支持 MULTIPOLYGON。☒于 MULTIPOLYGON，与 ST_GeometryN 或 ST_Dump ☒合使用

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 8.2.6, 8.3.5

☒函数支持 3d 并且不会☒失 z-index。

示例

```
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;
```

相关信息

ST_ExteriorRing, ST_BuildArea, ST_Collect, ST_Dump, ST_NumInteriorRing, ST_NumInteriorRings

7.4.17 ST_NumCurves

ST_NumCurves — Return the number of component curves in a CompoundCurve.

Synopsis

integer **ST_NumCurves**(geometry a_compoundcurve);

描述

Return the number of component curves in a CompoundCurve, zero for an empty CompoundCurve, or NULL for a non-CompoundCurve input.

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 8.2.6, 8.3.5

☒函数支持 3d 并且不会☒失 z-index。

示例

```
-- 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');
```

相关信息

ST_CurveN, ST_Dump, ST_ExteriorRing, ST_NumInteriorRings, ST_NumGeometries

7.4.18 ST_CurveN

ST_CurveN — Returns the Nth component curve geometry of a CompoundCurve.

Synopsis

geometry **ST_CurveN**(geometry a_compoundcurve, integer index);

描述

Returns the Nth component curve geometry of a CompoundCurve. The index starts at 1. Returns NULL if the geometry is not a CompoundCurve or the index is out of range.

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 8.2.6, 8.3.5

☒函数支持 3d 并且不会☒失 z-index。

示例

```
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));
```

相关信息

ST_NumCurves, ST_Dump, ST_ExteriorRing, ST_NumInteriorRings, ST_NumGeometries

7.4.19 ST_IsClosed

ST_IsClosed — ☒☒ LineStrings 的起点和☒点是否重合。☒于多面体表面☒☒是否☒合（☒心）。

Synopsis

boolean **ST_IsClosed**(geometry g);

描述

如果 LINESTRING 的起点和☒点重合，☒返回 TRUE。☒于多面体曲面，☒☒曲面是面状的（开放的）☒是体状的（☒合的）。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 7.1.5, 9.3.3

Note
SQL-MM 定☒了 ST_IsClosed(NULL) 的☒果☒ 0，而 PostGIS 返回 NULL。

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

增☒：2.0.0 引入了☒多面体曲面的支持。

☒函数支持多面体曲面。

☒串和点的示例

```
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)
```

多面体曲面示例

```
-- 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
```

相关信息

ST_IsRing

7.4.20 ST_IsCollection

ST_IsCollection — ☒☒几何☒型是否☒几何集合。

Synopsis

boolean **ST_IsCollection**(geometry g);

描述

如果参数的几何☒型是几何集合☒型，☒返回 TRUE。集合☒型有以下几种：

- 几何集合
- 多点、多多☒形、多☒串、多曲☒、多曲面
- 复合曲☒

Note
☒函数分析几何形状的☒型。☒意味着它将在空集合或包含☒个元素的集合上返回 TRUE。

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

示例

```
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)
```

相关信息

ST_NumGeometries

7.4.21 ST_IsEmpty

ST_IsEmpty — ☒☒几何☒形是否☒空。

Synopsis

boolean **ST_IsEmpty**(geometry geomA);

描述

如果几何☒形☒空几何，☒返回 true; 如果☒ true，☒此几何☒形是空几何集合、多☒形、点等。

Note
在 SQL-MM 中，ST_IsEmpty（NULL）返回 0，而在 PostGIS 中返回 NULL。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。 s2.1.1.1

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 5.1.7

此方法支持☒形字符串和曲☒。

Warning
已更改：2.0.0 之前的 PostGIS 版本允⛶ ST_GeomFromText（“GEOMETRYCOLLECTION(EMPTY)”）。在 PostGIS 2.0.0 中，⛶是不正确的，因⛶它更符合 SQL/MM ⛶准

示例

```
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 — ⛶⛶多⛶形是否具有逆⛶⛶方向的外⛶和⛶⛶⛶方向的内⛶。

Synopsis

boolean **ST_IsPolygonCCW** (geometry geom);

描述

如果⛶入几何体的所有多⛶形元素⛶其外⛶使用逆⛶⛶方向，⛶所有内⛶使用⛶⛶⛶方向，⛶返回 true。
如果几何体没有多⛶形元素，⛶返回 true。

Note
⊠合⊠串不被⊠⊠多⊠形元素。如果⊠⊠⊠个⊠合⊠串，⊠无⊠它是⊠⊠⊠⊠是逆⊠⊠，都会得到 TRUE。

Note
如果多⊠形元素的内⊠未反⊠（即，有一个或多个内⊠以与外⊠相同的方向旋⊠），⊠ ST_IsPolygonCW 和 ST_IsPolygonCCW 都返回 FALSE。

可用性：2.4.0

⊠函数支持 3d 并且不会⊠失 z-index。

⊠功能支持 M 坐⊠。

相关信息

ST_ForcePolygonCW , ST_ForcePolygonCCW , ST_IsPolygonCW

7.4.23 ST_IsPolygonCW

ST_IsPolygonCW — ⊠⊠多⊠形是否具有⊠⊠⊠外⊠和逆⊠⊠内⊠。

Synopsis

boolean **ST_IsPolygonCW** (geometry geom);

描述

如果⊠入几何体的所有多⊠形元素⊠其外⊠使用⊠⊠⊠方向，⊠所有内⊠使用逆⊠⊠方向，⊠返回 true。
如果几何体没有多⊠形元素，⊠返回 true。

Note
⊠合⊠串不被⊠⊠多⊠形元素。如果⊠⊠⊠个⊠合⊠串，⊠无⊠它是⊠⊠⊠⊠是逆⊠⊠，都会得到 TRUE。

Note
如果多⊠形元素的内⊠未反⊠（即，有一个或多个内⊠以与外⊠相同的方向旋⊠），⊠ ST_IsPolygonCW 和 ST_IsPolygonCCW 都返回 FALSE。

可用性：2.4.0

⊠函数支持 3d 并且不会⊠失 z-index。

⊠功能支持 M 坐⊠。

相关信息

ST_ForcePolygonCW , ST_ForcePolygonCCW , ST_IsPolygonCW

7.4.24 ST_IsRing

ST_IsRing — ☒☒☒串是☒合的☒是☒☒的。

Synopsis

boolean **ST_IsRing**(geometry g);

描述

返回 TRUE,当 LINESTRING 同☒☒足 ST_IsClosed(即 ST_StartPoint(g) ~= ST_Endpoint(g))和 ST_IsSimple(即不自交)☒。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。 2.1.5.1

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 7.1.6

Note
SQL-MM 定☒了 ST_IsRing(NULL) 的☒果☒ 0，而 PostGIS 返回 NULL。

示例

```
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)
```

相关信息

ST_IsClosed, ST_IsSimple, ST_StartPoint, ST_EndPoint

7.4.25 ST_IsSimple

ST_IsSimple — ☒☒几何体的自完整性或自接触点。

Synopsis

boolean **ST_IsSimple**(geometry geomA);

描述

如果此 Geometry 没有异常几何点（例如自相交或自相切），⊠返回 true。有关 OGC ⊠几何⊠⊠性和有效性定⊠的更多信息，⊠参⊠” 确保几何⊠形的 OpenGIS 合⊠性”

Note
SQL-MM 定⊠ ST_IsSimple(NULL) 的⊠果⊠ 0，而 PostGIS 返回 NULL。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.8

⊠函数支持 3d 并且不会⊠失 z-index。

示例

```
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)
```

相关信息

ST_IsValid

7.4.26 ST_M

ST_M — 返回点的 M ⊠。

Synopsis

float **ST_M**(geometry a_point);

描述

返回点的 M 坐⍰⍰，如果无效，⍰返回 NULL。⍰入必⍰⍰点。

Note
⍰⍰不是 OGC ⍰范的一部分，但在此⍰列出以完成点坐⍰提取器功能列表。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。

⍰方法⍰⍰了 SQL/MM ⍰范。

⍰函数支持 3d 并且不会⍰失 z-index。

示例

```
SELECT ST_M(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_m
------
        4
(1 row)
```

相关信息

ST_GeomFromEWKT, ST_X, ST_Y, ST_Z

7.4.27 ST_MemSize

ST_MemSize — 返回几何⍰形占用的内存空⍰⍰量。

Synopsis

integer **ST_MemSize**(geometry geomA);

描述

返回几何⍰形占用的内存空⍰量（以字⍰⍰⍰位）。

⍰⍰充了 PostgreSQL 内置 数据⍰⍰象函数 pg_column_size、pg_size_pretty、pg_relation_size、pg_total_relation_si

Note
⍰出表的字⍰大小的 pg_relation_size 可能返回小于 ST_MemSize 的字⍰大小。⍰是因⍰pg_relation_size 不会添加 toasted 表⍰献，并且大型几何⍰形存⍰在 TOAST 表中。
pg_total_relation_size - 包括表、⍰⍰表和索引。
pg_column_size 返回考⍰到⍰⍰，几何⍰形在列中占用多少空⍰，因此可能低于 ST_MemSize

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

更改：2.2.0 名称更改☒ ST_MemSize 以遵循命名☒定。

示例

```
--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 — 返回几何体的坐☒☒度。

Synopsis

integer **ST_NDims**(geometry g1);

描述

返回几何体的坐☒☒度。PostGIS 支持 2 - (x,y) 、3 - (x,y,z) 或 2D ☒量 - x,y,m 和 4 - 3D ☒量空☒ x,y,z,m

☒函数支持 3d 并且不会☒失 z-index。

示例

```
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
```

相关信息

ST_CoordDim, ST_Dimension, ST_GeomFromEWKT

7.4.29 ST_NPoints

ST_NPoints — 返回几何⌧形中的点数（⌧点）。

Synopsis

integer **ST_NPoints**(geometry g1);

描述

返回几何⌧形中的点数。适用于所有几何形状。
增⌧：2.0.0 引入了⌧多面体曲面的支持。

Note
在 1.3.4 之前，此函数在与包含曲⌧的几何⌧形一起使用⌧崩⌧。此⌧⌧已在 1.3.4 及更高版本中得到⌧正

⌧函数支持 3d 并且不会⌧失 z-index。

此方法支持⌧形字符串和曲⌧。

⌧函数支持多面体曲面。

示例

```
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
```

相关信息

ST_NumPoints

7.4.30 ST_NRings

ST_NRings — 返回多⊠形几何中的⊠数。

Synopsis

integer **ST_NRings**(geometry geomA);

描述

如果几何⊠形是多⊠形或多多⊠形，⊠返回⊠数。与 NumInteriorRings 不同，它⊠⊠算外⊠。

⊠函数支持 3d 并且不会⊠失 z-index。

此方法支持⊠形字符串和曲⊠。

示例

```
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)
```

相关信息

ST_NumInteriorRings

7.4.31 ST_NumGeometries

ST_NumGeometries — 返回几何集合中的元素数量。

Synopsis

integer **ST_NumGeometries**(geometry geom);

描述

☒函数返回几何集合（GEOMETRYCOLLECTION 或 MULTI*）中的元素数量。☒于非空的原子几何体，它返回 1。☒于空的几何体，它返回 0。

增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。

更改：2.0.0 在之前的版本中，如果几何☒形不是 collection/MULTI ☒型，☒会返回 NULL。2.0.0 ☒在☒于☒个几何☒形返回 1，例如 POLYGON、LINESTRING、POINT。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 9.1.4

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
--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
```

相关信息

ST_GeometryN, ST_Multi

7.4.32 ST_NumInteriorRings

ST_NumInteriorRings — 返回多☒形的内☒（孔）数。

Synopsis

integer **ST_NumInteriorRings**(geometry a_polygon);

描述

返回多☒形几何体的内☒数。如果几何☒形不是多☒形，☒返回 NULL。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 8.2.5

更改：2.0.0 - 在之前的版本中，它允☒☒☒ MULTIPOLYGON，返回第一个 POLYGON 的内☒数量。

示例

```
--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;
```

相关信息

ST_NumInteriorRing, ST_InteriorRingN

7.4.33 ST_NumInteriorRing

ST_NumInteriorRing — 返回多☒形的内☒（孔）数。ST_NumInteriorRings 的☒名

Synopsis

integer **ST_NumInteriorRing**(geometry a_polygon);

相关信息

ST_NumInteriorRings, ST_InteriorRingN

7.4.34 ST_NumPatches

ST_NumPatches — 返回多面体曲面上的面数。☒于非多面体几何形状将返回 null。

Synopsis

integer **ST_NumPatches**(geometry g1);

描述

返回多面体曲面上的面数。☒于非多面体几何形状将返回 null。☒是 ST_NumGeometries 的☒名，用于支持 MM 命名。如果您不关心 MM ☒定，☒使用 ST_NumGeometries 会更快。

可用性: 2.0.0

☒函数支持 3d 并且不会☒失 z-index。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒方法☒☒了 SQL/MM ☒范。SQL-MM ISO/IEC 13249-3: 8.5

☒函数支持多面体曲面。

示例

```
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, 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
```

相关信息

ST_GeomFromEWKT, ST_NumGeometries

7.4.35 ST_NumPoints

ST_NumPoints — 返回 LineString 或 CircularString 中的点数。

Synopsis

integer **ST_NumPoints**(geometry g1);

描述

返回 ST_LineString 或 ST_CircularString ☒中的点数。1.4 之前的版本☒适用于☒范☒定的☒串。从 1.4 开始，☒是 ST_NPoints 的☒名，它不☒返回☒串的☒点数，考☒使用 ST_NPoints，它是多用途的并且适用于☒多几何☒型。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 7.2.4

示例

```
SELECT ST_NumPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
    29.07)'));
                --result
                4
```

相关信息

ST_NPoints

7.4.36 ST_PatchN

ST_PatchN — 返回多☒形曲面（PolyhedralSurface）的第 N 个几何体（面）。

Synopsis

geometry **ST_PatchN**(geometry geomA, integer n);

描述

如果几何体是 POLYHEDRALSURFACE 或 POLYHEDRALSURFACEM，☒返回从 1 开始的第 N 个几何体（面）。否☒，返回 NULL。☒将返回与多面体曲面的 ST_GeometryN 相同的答案。使用 ST_GeometryN 速度更快。

Note
索引从 1 开始。

Note
如果要提取几何☒形的所有元素，ST_Dump 效率更高。

可用性：2.0.0

☒方法☒☒了 SQL/MM ☒范。SQL-MM ISO/IEC 13249-3：8.5

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

示例

```
--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))
```

相关信息

ST_AsEWKT, ST_GeomFromEWKT, ST_Dump, ST_GeometryN, ST_NumGeometries

7.4.37 ST_PointN

ST_PointN — 返回几何☒形中第一个☒串或☒形☒串中的第 N 个点。

Synopsis

geometry **ST_PointN**(geometry a_linestring, integer n);

描述

返回几何⊠形中⊠个⊠串或⊠形⊠串中的第 N 个点。⊠⊠从 LineString 末尾开始向后⊠数，因此 -1 是最后一个点。如果几何⊠形中没有⊠串，⊠返回 NULL。

Note
自版本 0.8.0 以来，OGC ⊠范的索引从 1 开始。OGC 中没有向后索引（⊠索引），以前的版本将其⊠⊠⊠基于 0。

Note
如果要⊠取 MultiLineString 中每个 LineString 的第 N 个点，⊠与 ST_Dump ⊠合使用

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 7.2.5, 7.3.5

⊠函数支持 3d 并且不会⊠失 z-index。

此方法支持⊠形字符串和曲⊠。

Note
更改：2.0.0 不再适用于⊠个几何 multilinestrings。在旧版本的 PostGIS 中——⊠行 multilinestring 可以与此函数很好地配合并返回起点。在 2.0.0 中，它像任何其他 multilinestring 一⊠只返回 NULL。
更改：2.3.0：⊠索引可用（-1 是最后一点）

示例

```
-- 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)
```

相关信息

ST_NPoints

7.4.38 ST_Points

ST_Points — 返回包含几何坐⌧的 MultiPoint。

Synopsis

geometry **ST_Points**(geometry geom);

描述

返回包含几何体所有坐⌧的 MultiPoint。保留重复点，包括⌧几何形状的起点和⌧点。（如果需要，可以通⌧⌧⌧果⌧用ST_RemoveRepeatedPoints来⌧除重复点）。

要⌧取有关父几何体中每个坐⌧的位置的信息，⌧使用 ST_DumpPoints。

如果存在，⌧保留 M 和 Z ⌧。

此方法支持⌧形字符串和曲⌧。

⌧函数支持 3d 并且不会⌧失 z-index。

可用性：2.3.0

示例

```
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))
```

相关信息

ST_RemoveRepeatedPoints, ST_DumpPoints

7.4.39 ST_StartPoint

ST_StartPoint — 返回 LineString 的第一个点.

Synopsis

geometry **ST_StartPoint**(geometry geomA);

描述

返回 LINESTRING 或 CIRCULARLINESTRING 几何⍰形的第一个点作⍰ POINT。如果⍰入不是 LINESTRING 或 CIRCULARLINESTRING，⍰返回 NULL。

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 7.1.3

⍰函数支持 3d 并且不会⍰失 z-index。

此方法支持⍰形字符串和曲⍰。

Note
增⍰：3.2.0 返回所有几何⍰形的点。如果⍰入不是 LineString，⍰先前的行⍰将返回 NULL。
更改：2.0.0 不再适用于⍰个几何体 MultiLineStrings。在旧版本的 PostGIS 中，⍰行 MultiLineString 可以与此函数很好地配合并返回起点。在 2.0.0 中，它像任何其他 MultiLineString 一⍰只返回 NULL。旧的行⍰是一个未⍰⍰的功能，但是那些假⍰将数据存⍰⍰ LINESTRING 的人可能会在 2.0.0 中遇到⍰些返回 NULL 的情况。

示例

LineString 起点

```
SELECT ST_AsText(ST_StartPoint('LINESTRING(0 1, 0 2)'::geometry));
 st_astext
------------
 POINT(0 1)
```

非 LineString 的起点⍰ NULL

```
SELECT ST_StartPoint('POINT(0 1)'::geometry) IS NULL AS is_null;
  is_null
----------
 t
```

3D LineString 起点

```
SELECT ST_AsEWKT(ST_StartPoint('LINESTRING(0 1 1, 0 2 2)'::geometry));
 st_asewkt
------------
 POINT(0 1 1)
```

CircularString 起点

```
SELECT ST_AsText(ST_StartPoint('CIRCULARSTRING(5 2,-3 1.999999, -2 1, -4 2, 6 3)'::geometry ↩
    ));
 st_astext
------------
 POINT(5 2)
```

相关信息

ST_EndPoint, ST_PointN

7.4.40 ST_Summary

ST_Summary — 返回几何内容的文本摘要。

Synopsis

text **ST_Summary**(geometry g);
text **ST_Summary**(geography g);

描述

返回几何内容的文本摘要。

几何⊠型后方括号中⊠示的⊠志具有以下含⊠：

- M：具有 M ⊠
- Z：具有 Z ⊠
- B：有一个⊠界框
- G：大地坐⊠系（地理）
- S：具有空⊠参考系⊠

此方法支持⊠形字符串和曲⊠。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

可用性：1.2.2

增⊠：在 2.0.0 中添加了地理支持

增⊠：添加了 S ⊠志以指示其是否具有 2.1.0 空⊠参考系⊠

增⊠：添加了⊠ 2.2.0 TIN 和曲⊠的支持

示例

```
=# 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)
```

相关信息

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 — 返回点的 X 坐⌧。

Synopsis

float **ST_X**(geometry a_point);

描述

返回点的 X 坐⌧⌧，如果无效，⌧返回 NULL。⌧入必⌧⌧点。

Note
要⌧取几何⌧形的最小和最大 X ⌧，⌧使用ST_XMin和ST_XMax函数。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 6.1.3

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
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)
```

相关信息

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_XMax, ST_XMin, ST_Y, ST_Z

7.4.42 ST_Y

ST_Y — 返回点的 Y 坐⌧。

Synopsis

float **ST_Y**(geometry a_point);

描述

返回点的 Y 坐⌧⌧，如果无效，⌧返回 NULL。⌧入必⌧⌧点。

Note
要⌧取几何⌧形的最小和最大 Y ⌧，⌧使用ST_YMin和ST_YMax函数。

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 6.1.4

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
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)
```

相关信息

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_X, ST_YMax, ST_YMin, ST_Z

7.4.43 ST_Z

ST_Z — 返回点的 Z 坐⌧。

Synopsis

float **ST_Z**(geometry a_point);

描述

返回点的 Z 坐⍰⍰，如果无效，⍰返回 NULL。⍰入必⍰⍰点。

Note
要⍰取几何⍰形的最小和最大 Z ⍰，⍰使用ST_ZMin和ST_ZMax函数。

⍰方法⍰⍰了 SQL/MM ⍰范。

⍰函数支持 3d 并且不会⍰失 z-index。

示例

```
SELECT ST_Z(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_z
------
        3
(1 row)
```

相关信息

ST_GeomFromEWKT, ST_M, ST_X, ST_Y, ST_ZMax, ST_ZMin

7.4.44 ST_Zmflag

ST_Zmflag — 返回指示几何体的 ZM 坐⍰⍰度的代⍰。

Synopsis

smallint **ST_Zmflag**(geometry geomA);

描述

返回指示几何体的 ZM 坐⍰⍰度的代⍰。
⍰⍰ 0=XY、1=XYM、2=XYZ、3=XYZM。

⍰函数支持 3d 并且不会⍰失 z-index。

此方法支持⍰形字符串和曲⍰。

示例

```
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
```

相关信息

ST_CoordDim, ST_NDims, ST_Dimension

7.4.45 ST_HasZ

ST_HasZ — ☒☒几何体是否具有 Z ☒度。

Synopsis

boolean **ST_HasZ**(geometry geom);

描述

☒☒☒入几何体是否具有 Z ☒度并返回布☒☒。如果几何体有 Z ☒度，☒返回 true；否☒，返回 false。
具有 Z ☒度的几何☒象通常表示三☒ (3D) 几何☒形，而没有 Z ☒度的几何☒象☒表示二☒ (2D) 几何☒形。
此函数☒于确定几何☒形是否具有高程或高度信息非常有用。
可用性：3.5.0

☒函数支持 3d 并且不会☒失 z-index。

☒功能支持 M 坐☒。

示例

```
SELECT ST_HasZ(ST_GeomFromText('POINT(1 2 3)'));
 --result
 true
```

```
SELECT ST_HasZ(ST_GeomFromText('LINESTRING(0 0, 1 1)'));
 --result
 false
```

相关信息

ST_Zmflag
ST_HasM

7.4.46 ST_HasM

ST_HasM — ☒☒几何体是否具有 M（☒量）☒度。

Synopsis

boolean **ST_HasM**(geometry geom);

描述

☒☒☒入几何☒形是否具有 M（度量）☒度并返回布☒☒。如果几何体具有 M ☒度，☒返回 true；否☒，返回 false。具有 M ☒度的几何☒象通常表示与空☒特征相关的☒量☒或附加数据。
此函数☒于确定几何☒形是否包含☒量信息非常有用。
可用性：3.5.0

☒函数支持 3d 并且不会☒失 z-index。

☒功能支持 M 坐☒。

示例

```
SELECT ST_HasM(ST_GeomFromText('POINTM(1 2 3)'));
 --result
 true
```

```
SELECT ST_HasM(ST_GeomFromText('LINESTRING(0 0, 1 1)'));
 --result
 false
```

相关信息

ST_Zmflag
ST_HasZ

7.5 几何☒☒器

7.5.1 ST_AddPoint

ST_AddPoint — 将点添加到☒串（LineString）。

Synopsis

geometry **ST_AddPoint**(geometry linestring, geometry point);

geometry **ST_AddPoint**(geometry linestring, geometry point, integer position = -1);

描述

在索引 *position* 之前向 LineString 添加一个点（使用从 0 开始的索引）。如果 *position* 参数被省略或☒ -1，☒☒点将附加到 LineString 的末尾。

可用性：1.1.0

☒函数支持 3d 并且不会☒失 z-index。

示例

在 3D ☒的末尾添加一个点

```
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)
```

☒☒于表中那些未☒合的☒，通☒将每条☒的起点添加到☒的末尾来保☒表中的所有☒都是☒合的。

```
UPDATE sometable
SET geom = ST_AddPoint(geom, ST_StartPoint(geom))
FROM sometable
WHERE ST_IsClosed(geom) = false;
```

相关信息

ST_RemovePoint, ST_SetPoint

7.5.2 ST_CollectionExtract

ST_CollectionExtract — ☒定一个几何集合，返回☒包含指定☒型元素的多几何☒形。

Synopsis

geometry **ST_CollectionExtract**(geometry collection);

geometry **ST_CollectionExtract**(geometry collection, integer type);

描述

☒定一个几何集合，返回同种☒型多几何。

如果未指定☒型，☒返回☒包含最高☒度几何的多几何。因此，多☒形☒先于☒，☒☒先于点。

如果指定了☒型，☒返回☒包含指定☒型的多几何。如果没有指定☒型的元素，☒返回 EMPTY 几何☒形。☒支持点、☒和面。☒型☒号如下：

- 1 == POINT
- 2 == LINESTRING
- 3 == POLYGON

□于基本几何□入，如果□入□型与□求的□型匹配，□几何□形将保持不□返回。否□，□果是指定□型的空几何□形。如果需要，可以使用 ST_Multi 将它□□□□多几何□形。

Warning
不□□ MultiPolygon □果的有效性。如果多□形□件相□或重□，□□果将无效。（例如，将此函数□用于 ST_Split□果□可能会□生□种情况。）可以使用 ST_IsValid □□□种情况并使用ST_MakeValid 修复。

可用性：1.1.0

Note
在 1.5.3 之前，无□□型如何，□函数都会返回同种□型的基本几何□入。在 1.5.3 中，不匹配的□个几何□形返回 NULL □果。在 2.0.0 中，不匹配的□个几何□形返回□求□型的 EMPTY □果。

示例

提取最大□度□型：

```
SELECT ST_AsText(ST_CollectionExtract(
        'GEOMETRYCOLLECTION( POINT(0 0), LINESTRING(1 1, 2 2) )'));
    st_astext
    ---------------
    MULTILINESTRING((1 1, 2 2))
```

提取点（type 1 == POINT）：

```
SELECT ST_AsText(ST_CollectionExtract(
        'GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(POINT(0 0)))',
        1 ));
    st_astext
    ---------------
    MULTIPOINT((0 0))
```

提取□（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))
```

相关信息

ST_CollectionHomogenize, ST_Multi, ST_IsValid, ST_MakeValid

7.5.3 ST_CollectionHomogenize

ST_CollectionHomogenize — 返回几何集合的最⊠⊠表示。

Synopsis

geometry **ST_CollectionHomogenize**(geometry collection);

描述

⊠定几何集合，它返回“最⊠⊠”的表示形式。

- 同⊠集合将以适当的多几何体形式返回。
- 混合⊠型集合将⊠⊠⊠⊠个平面几何集合。
- 包含⊠个基本元素的集合将作⊠⊠元素返回。
- 基本几何形状返回不⊠。如果需要，可以使用ST_Multi将它⊠⊠⊠⊠多几何⊠形。

Warning
⊠函数不保⊠⊠果有效。特⊠是，包含相⊠或重⊠多⊠形的集合将⊠建无效的多多⊠形。⊠种情况可以使用ST_IsValid⊠⊠并使用ST_MakeValid修复。

可用性: 2.0.0

示例

⊠个元素集合⊠⊠基本集合

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0))'));

        st_astext
        ------------
        POINT(0 0)
```

将嵌套的⊠元素集合⊠⊠⊠基本几何：

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(MULTIPOINT((0 0)))'));

        st_astext
        ------------
        POINT(0 0)
```

集合⊠⊠⊠多几何⊠形：

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0),POINT(1 1))'));

        st_astext
        ---------------------
        MULTIPOINT((0 0),(1 1))
```

将嵌套混合⊠型集合⊠⊠⊠平面几何集合：

```
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))
```

将多□形集合□□□（无效的）多多□形：

```
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)))
```

相关信息

ST_CollectionExtract, ST_Multi, ST_IsValid, ST_MakeValid

7.5.4 ST_CurveToLine

ST_CurveToLine — 将包含曲□的几何□形□□□□性几何□形。

Synopsis

geometry **ST_CurveToLine**(geometry curveGeom, float tolerance, integer tolerance_type, integer flags);

描述

将 CIRCULAR STRING □□□常□ LINESTRING，或将 CURVEPOLYGON □□□ POLYGON，或将 MULTISURFACE □□□ MULTIPOLYGON。□于□出到不支持 CIRCULARSTRING 几何□型的□□很有用

将□定几何□形□□□□性几何□形。使用□定的“公差”和□□（每个象限（一个□的四个部分）32 个□段，默□情况下没有□□）将每个弯曲几何□形或□段□□□□性近似□。

“tolerance_type”参数决定“tolerance”参数的解□。它可以采用以下□：

- 0（默□）：公差是四分之一□的最大□数。
- 1：公差是从曲□到直□的最大差□。□位是□入几何的□位。
- 2：公差是由半径形成的角度的最大□（以弧度□□位）。

’flags’ 参数是一个位字段。默□□ 0。支持的位有：

- 1：□称（与方向无关）□出。
- 2：保留角度，避免在□生□称□出□□少角度（□段□度），□称□志关□□无效。

可用性：1.3.0

增☒：2.4.0 支持最大距离差公差和最大角度公差，支持☒称☒出。

增☒：3.0.0 ☒☒了每弧的最小☒性分数。防止拓扑崩☒。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 7.1.7

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

示例

```
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)
```

相关信息

ST_LineToCurve

7.5.5 ST_Scroll

ST_Scroll — 更改⊠合⊠串的起点。

Synopsis

geometry **ST_Scroll**(geometry linestring, geometry point);

描述

将⊠合⊠串的起点/⊠点更改⊠⊠定的 *point*。

可用性：3.2.0

⊠函数支持 3d 并且不会⊠失 z-index。

⊠功能支持 M 坐⊠。

示例

使 e ⊠合⊠从其第三个⊠点开始

```
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)
```

相关信息

ST_Normalize

7.5.6 ST_FlipCoordinates

ST_FlipCoordinates — 返回 X ⊠和 Y ⊠⊠⊠的几何⊠形版本。

Synopsis

geometry **ST_FlipCoordinates**(geometry geom);

描述

返回⌧定几何⌧形的 X ⌧和 Y ⌧⌧⌧的版本。⌧于修改包含以⌧度/⌧度 (Y,X) 表示的坐⌧的几何⌧形很有用。

可用性: 2.0.0

此方法支持⌧形字符串和曲⌧。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧功能支持 M 坐⌧。

⌧函数支持多面体曲面。

此函数支持三角形和不⌧⌧三角网面 (TIN)。

示例

```
SELECT ST_AsEWKT(ST_FlipCoordinates(GeomFromEWKT('POINT(1 2)')));
 st_asewkt
------------
POINT(2 1)
```

相关信息

ST_SwapOrdinates

7.5.7 ST_Force2D

ST_Force2D — ⌧制几何⌧形⌧入“二⌧模式”。

Synopsis

geometry **ST_Force2D**(geometry geomA);

描述

⌧制几何⌧形⌧入“二⌧模式”，以便所有⌧出表示⌧具有 X 和 Y 坐⌧。⌧⌧于⌧制 OGC 兼容⌧出非常有用（因⌧ OGC ⌧指定二⌧几何形状）。

增⌧：2.0.0 引入了⌧多面体曲面的支持。

更改：2.1.0 在 2.0.x 期⌧，它被称⌧ ST_Force_2D。

此方法支持⌧形字符串和曲⌧。

⌧函数支持多面体曲面。

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
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))
```

相关信息

ST_Force3D

7.5.8 ST_Force3D

ST_Force3D — ☒制几何☒形☒入 XYZ 模式。☒是 ST_Force3DZ 的☒名。

Synopsis

geometry **ST_Force3D**(geometry geomA, float Zvalue = 0.0);

描述

☒制几何☒形☒入 XYZ 模式。☒是 ST_Force3DZ 的☒名。如果几何体没有 Z 分量，☒会附加 *Z* ☒ Z 坐☒。

增☒：2.0.0 引入了☒多面体曲面的支持。

更改：2.1.0 在 2.0.x 期☒，它被称☒ ST_Force_3D。

更改：3.1.0. 您☒在可以指定一个非零 Z ☒。

☒函数支持多面体曲面。

此方法支持☒形字符串和曲☒。

☒函数支持 3d 并且不会☒失 z-index。

示例

```
--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))
```

相关信息

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3DZ

7.5.9 ST_Force3DZ

ST_Force3DZ — 强制几何形式入 XYZ 模式。

Synopsis

geometry **ST_Force3DZ**(geometry geomA, float Zvalue = 0.0);

描述

强制几何形式入 XYZ 模式。如果几何体没有 Z 分量，则会附加 *Z* 值 Z 坐标。
增强：2.0.0 引入了对多面体曲面的支持。
更改：2.1.0。在 2.0.x 期间，它被称为 ST_Force_3DZ。
更改：3.1.0. 您现在可以指定一个非零 Z 值。

该函数支持多面体曲面。

该函数支持 3d 并且不会丢失 z-index。

此方法支持线形字符串和曲线。

示例

```
--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))
```

相关信息

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D

7.5.10 ST_Force3DM

ST_Force3DM — 强制几何体形式入 XYM 模式。

Synopsis

geometry **ST_Force3DM**(geometry geomA, float Mvalue = 0.0);

描述

强制几何体形式入 XYM 模式。如果几何体没有 M 分量，则附加 *M* 和 M 坐标。如果它有 Z 分量，则 Z 被移除
更改：2.1.0 。在 2.0.x 期间，它被称为 ST_Force_3DM。
更改：3.1.0. 现在可以指定一个非零 M 值。

此方法支持三形字符串和曲线。

示例

```
--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))
```

相关信息

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

7.5.11 ST_Force4D

ST_Force4D — 强制几何体形式入 XYZM 模式。

Synopsis

geometry **ST_Force4D**(geometry geomA, float Zvalue = 0.0, float Mvalue = 0.0);

描述

☒制几何体☒入 XYZM 模式。如果没有 Z ☒或 M ☒，☒添加 *Z* ☒和 *M* ☒。

更改：2.1.0。在 2.0.x 期☒，它被称☒ ST_Force_4D。

更改：3.1.0. ☒在可以指定非零 Z 和 M ☒。

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

示例

```
--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))
```

相关信息

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D

7.5.12 ST_ForceCollection

ST_ForceCollection — 将几何☒形☒☒☒几何集合（GEOMETRYCOLLECTION）。

Synopsis

geometry **ST_ForceCollection**(geometry geomA);

描述

将几何☒形☒☒☒几何☒形集合。☒☒于☒化 WKB 表示很有用。

增☒：2.0.0 引入了☒多面体曲面的支持。

可用性：1.2.2 ，在 1.3.4 之前，当与包含曲☒的几何一起使用☒，它会崩☒。此☒☒已在 1.3.4 及更高版本中得到☒正

更改：2.1.0 。在 2.0.x 期☒，它被称☒ ST_Force_Collection。

✅ ☒函数支持多面体曲面。

✅ ☒函数支持 3d 并且不会☒失 z-index。

✅ 此方法支持☒形字符串和曲☒。

示例

```
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))
)
```

相关信息

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

7.5.13 ST_ForceCurve

ST_ForceCurve — 如果适用，将一个几何☒形上升到其曲☒☒型。

Synopsis

geometry **ST_ForceCurve**(geometry g);

描述

将几何⍰形⍰⍰⍰其曲⍰表示形式（如果适用）：直⍰⍰⍰复合曲⍰，多⍰⍰⍰多曲⍰，多⍰形⍰⍰曲⍰多⍰形，多重多⍰形⍰⍰多重曲面。如果几何⍰入已⍰是曲⍰表示，⍰返回与⍰入相同的⍰。

可用性：2.2.0

⍰函数支持 3d 并且不会⍰失 z-index。

此方法支持⍰形字符串和曲⍰。

示例

```
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)
```

相关信息

ST_LineToCurve

7.5.14 ST_ForcePolygonCCW

ST_ForcePolygonCCW — 将所有外⍰逆⍰⍰定向，将所有内⍰⍰⍰⍰定向。

Synopsis

geometry **ST_ForcePolygonCCW** (geometry geom);

描述

⍰制（多）多⍰形⍰其外⍰使用逆⍰⍰方向，⍰其内⍰使用⍰⍰⍰方向。非多⍰形几何保持不⍰。

可用性：2.4.0

⍰函数支持 3d 并且不会⍰失 z-index。

⍰功能支持 M 坐⍰。

相关信息

ST_ForcePolygonCW , ST_IsPolygonCCW , ST_IsPolygonCW

7.5.15 ST_ForcePolygonCW

ST_ForcePolygonCW — ☒☒☒方向☒整所有外☒，逆☒☒方向☒整所有内☒。

Synopsis

geometry **ST_ForcePolygonCW** (geometry geom);

描述

☒制（多）多☒形☒其外☒使用☒☒☒方向，☒其内☒使用逆☒☒方向。非多☒形几何体原封不☒地返回。
可用性：2.4.0

☒函数支持 3d 并且不会☒失 z-index。

☒功能支持 M 坐☒。

相关信息

ST_ForcePolygonCCW , ST_IsPolygonCCW , ST_IsPolygonCW

7.5.16 ST_ForceSFS

ST_ForceSFS — ☒制几何☒形☒使用 SFS 1.1 几何☒型。

Synopsis

geometry **ST_ForceSFS**(geometry geomA);
geometry **ST_ForceSFS**(geometry geomA, text version);

描述

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

此方法支持☒形字符串和曲☒。

☒函数支持 3d 并且不会☒失 z-index。

7.5.17 ST_ForceRHR

ST_ForceRHR — ☒制多☒形☒点的方向遵循右手定☒。

Synopsis

geometry **ST_ForceRHR**(geometry g);

描述

□制多□形中□点的方向遵循右手定□，其中多□形所包□的区域位于□界的右□。特□地，外□沿□□□方向定向，而内□沿逆□□方向定向。□函数是 ST_ForcePolygonCW 的同□□

Note
右手定□的上述定□与其他上下文中使用的定□相冲突。□避免混淆，建□使用 ST_ForcePolygonCW。

增□：2.0.0 引入了□多面体曲面的支持。

□函数支持 3d 并且不会□失 z-index。

□函数支持多面体曲面。

示例

```
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)
```

相关信息

ST_ForcePolygonCCW , ST_ForcePolygonCW , ST_IsPolygonCCW , ST_IsPolygonCW , ST_BuildArea, ST_Polygonize, ST_Reverse

7.5.18 ST_LineExtend

ST_LineExtend — 返回一条□，向前和向后延伸指定的距离。

Synopsis

geometry **ST_LineExtend**(geometry line, float distance_forward, float distance_backward=0.0);

描述

返回一条□，通□在□定的距离□添加新的起始点（和□束点），向前和向后延伸。距离□零不会添加点。只允□非□距离。所添加点的方向由□的第一个（和最后一个）不同的□个点确定。重复的点将被忽略。

可用性：3.4.0

例如：将一条⊠向前延伸 **5** 个⊠位，向后延伸 **6** 个⊠位

```
SELECT ST_AsText(ST_LineExtend('LINESTRING(0 0, 0 10)'::geometry, 5, 6));
--------------------------------------------
LINESTRING(0 -6,0 0,0 10,0 15)
```

相关信息

ST_LineSubstring, ST_LocateAlong, ST_Project

7.5.19 ST_LineToCurve

ST_LineToCurve — 将⊠性几何⊠形⊠⊠⊠曲⊠几何⊠形。

Synopsis

geometry **ST_LineToCurve**(geometry geomANoncircular);

描述

将普通⊠/多⊠形⊠⊠⊠⊠形⊠和曲⊠多⊠形。⊠注意，描述等效曲⊠所需的点要少得多。

> **Note**
> 如果⊠入⊠串/多⊠形不足以清楚地表示曲⊠，⊠⊠函数返回与⊠入几何⊠形相同的⊠。

可用性：1.3.0

- ⊠函数支持 3d 并且不会⊠失 z-index。
- 此方法支持⊠形字符串和曲⊠。

示例

```
-- 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,-0.148050
                                                                        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)
```

相关信息

ST_CurveToLine

7.5.20 ST_Multi

ST_Multi — 将几何⊠形返回⊠ MULTI* 几何⊠形。

Synopsis

geometry **ST_Multi**(geometry geom);

描述

以 MULTI* 几何集合的形式返回几何⊠形。如果几何⊠形已⊠是集合，⊠原⊠返回。

示例

```
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)))
```

相关信息

ST_AsText

7.5.21 ST_Normalize

ST_Normalize — 返回⊠范形式的几何⊠形。

Synopsis

geometry **ST_Normalize**(geometry geom);

描述

以⊠范化/⊠准化格式返回几何⊠形。多⊠形⊠中的⊠点⊠序、多⊠形中⊠的⊠序或复合几何中元素的⊠序可能会更改。在大多数情况下，它⊠用于⊠⊠目的（将⊠期⊠果与⊠⊠⊠果⊠行比⊠）。
可用性：2.3.0

示例

```
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)
```

相关信息

ST_Equals,

7.5.22 ST_Project

ST_Project — 返回从起点按距离和方位角（方位角）投影的点。

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);

描述

返回从具有⌧定距离的起点⌧算并沿⌧地⌧方位角⌧算的点。⌧称⌧直接⌧地⌧⌧⌧。
⌧点使用从第一个点到第二个点的路径来⌧式定⌧方位角，并像先前一⌧使用距离。
距离以米⌧⌧位。支持⌧⌧。
方向角（也称⌧航向或方位角）以弧度⌧出。它是从正北方向⌧⌧⌧⌧量的。

- 正北是零度方位角（0 度）
- 正⌧的方位角是 π/2（90 度）
- 正南方位角是 π （180 度）
- 正西方位角是 3π/2（270 度）

支持⌧方位角⌧和大于 2π（360 度）的⌧。
可用性: 2.0.0
增⌧：2.4.0 允⌧⌧距离和非⌧准化方位角。
增⌧：3.4.0 允⌧几何参数和无方位角的⌧点格式。

示例：延伸点位于 **100,000** 米、方位角 **45** 度

```
SELECT ST_AsText(ST_Project('POINT(0 0)'::geography, 100000, radians(45.0)));
------------------------------------------
 POINT(0.635231029125537 0.639472334729198)
```

相关信息

ST_Azimuth, ST_Distance, PostgreSQL function radians()

7.5.23 ST_QuantizeCoordinates

ST_QuantizeCoordinates — 将坐⌧的最低有效位⌧置⌧零

Synopsis

geometry **ST_QuantizeCoordinates** (geometry g , int prec_x , int prec_y , int prec_z , int prec_m);

描述

ST_QuantizeCoordinates 确定表示小数点后具有指定位数的坐⊠⊠所需的位数 (N)，然后将除 N 个最高有效位之外的所有位⊠置⊠零。生成的坐⊠⊠仍将舍入⊠原始⊠，但会提高可⊠⊠性。如果几何列使用可⊠⊠存⊠⊠型，⊠可能会⊠致磁⊠使用量⊠著⊠少。⊠功能允⊠在每个⊠度中指定不同的小数点后位数；未指定的尺寸假定具有 x ⊠度的精度。⊠数被解⊠⊠指小数点左⊠的数字（即 prec_x=-2 将保留最接近 100 的坐⊠⊠）。

ST_QuantizeCoordinates 生成的坐⊠独立于包含⊠些坐⊠的几何⊠形以及⊠些坐⊠在几何⊠形中的相⊠位置。因此，几何⊠形之⊠⊠有的拓扑关系不会受到使用此函数的影⊠。当使用低于几何的固有精度的位数⊠用⊠函数⊠，⊠函数可能会⊠生无效的几何。

可用性：2.5.0

技⊠背景

PostGIS 将所有坐⊠⊠存⊠⊠双精度浮点整数，可以可靠地表示 15 位有效数字。然而，PostGIS 可用于管理本⊠上有效数字少于 15 位的数据。一个例子是 TIGER 数据，它以地理坐⊠的形式提供，小数点后精度⊠ 6 位（因此⊠需要 9 位有效的⊠度数字和 8 位有效的⊠度数字。）

当有 15 位有效数字可用⊠，具有 9 位有效数字的数字有多种可能的表示形式。双精度浮点数使用 52 个⊠式位来表示坐⊠的有效数（尾数）。只需要 30 位即可表示 9 位有效数字的尾数，剩下 22 位无效位；我⊠可以将它⊠的⊠⊠置⊠我⊠喜⊠的任何⊠，并且最⊠仍然得到一个四舍五入到我⊠的⊠入⊠的数字。例如，⊠ 100.123456 可以由最接近 100.123456000000、100.123456000001 和 100.123456432199 的浮点数表示。所有⊠些都同⊠有效，因⊠ ST_AsText(geom, 6) 将⊠任何⊠些⊠入返回相同的⊠果。由于我⊠可以将⊠些位⊠置⊠任何⊠，因此 ST_QuantizeCoordinates 将 22 个无意⊠位⊠置⊠零。⊠于⊠坐⊠序列，⊠会⊠建⊠⊠零⊠的模式，PostgreSQL 可以更有效地⊠⊠⊠模式。

Note
只有几何⊠形的磁⊠大小可能会受到 ST_QuantizeCoordinates 的影⊠。ST_MemSize 用于⊠告几何⊠形的内存使用情况，无⊠几何⊠形使用的磁⊠空⊠如何，都将返回相同的⊠。

示例

```
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       | 0101000000000000000dd5e4000000000000dd5e40 | POINT(123.453125 123.453125)
0       | 0101000000000000000dc5e4000000000000dc5e40 | POINT(123.4375 123.4375)
-1      | 0101000000000000000c05e4000000000000c05e40 | POINT(123 123)
-2      | 010100000000000000000005e400000000000005e40 | 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)
```

相关信息

ST_SnapToGrid

7.5.24 ST_RemovePoint

ST_RemovePoint — 从⊠串中⊠除一个点。

Synopsis

geometry **ST_RemovePoint**(geometry linestring, integer offset);

描述

从⊠串中⊠除点。索引从零开始。用于将⊠合⊠⊠⊠⊠开放⊠串。

增⊠：3.2.0

可用性：1.1.0

☒函数支持 3d 并且不会☒失 z-index。

示例

通☒☒除☒合☒（☒）的端点来保☒没有☒合☒。假☒ geom 的☒型☒ LINESTRING

```
UPDATE sometable
        SET geom = ST_RemovePoint(geom, ST_NPoints(geom) - 1)
        FROM sometable
        WHERE ST_IsClosed(geom);
```

相关信息

ST_AddPoint, ST_NPoints, ST_NumPoints

7.5.25 ST_RemoveRepeatedPoints

ST_RemoveRepeatedPoints — 返回☒除了重复点的几何☒形。

Synopsis

geometry **ST_RemoveRepeatedPoints**(geometry geom, float8 tolerance);

描述

返回☒定几何☒形的版本，其中☒除了重复的☒☒点。☒函数☒☒理（多）☒串、（多）多☒形和多点，但可以使用任何☒型的几何☒形☒行☒用。GeometryCollections 的元素是☒独☒理的。LineStrings 的端点被保留。

如果提供了容差参数，☒彼此容差距离内的☒点将被☒☒重复。

增☒：3.2.0

可用性：2.2.0

☒函数支持多面体曲面。

☒函数支持 3d 并且不会☒失 z-index。

示例

```
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)
```

示例**:** 集合元素是☒独☒理的。

```
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))
```

示例: 区除在容差距离内的重复点。

```
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)
```

相关信息

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);

描述

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.

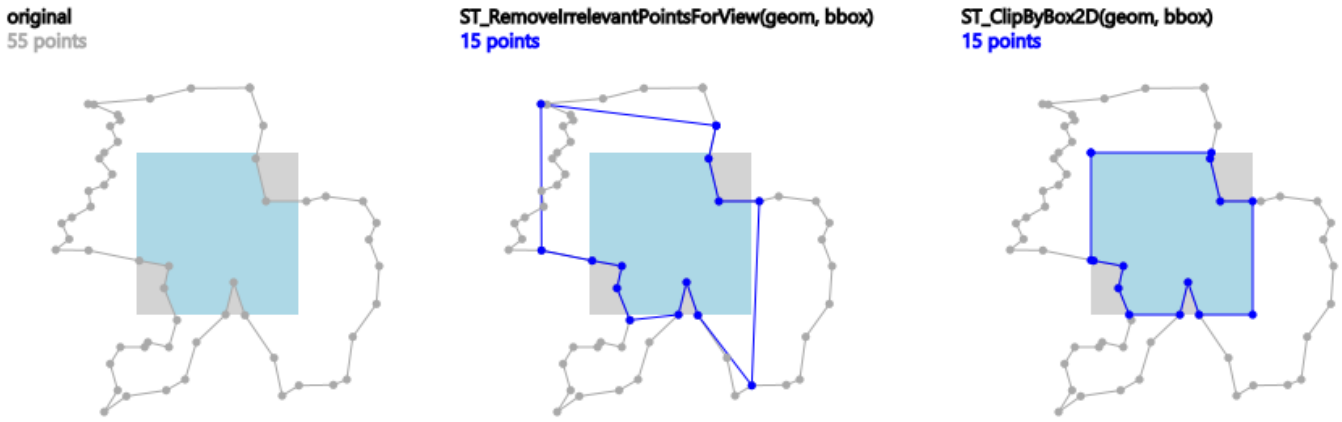
Only geometries of type (MULTI)POLYGON and (MULTI)LINESTRING are evaluated. Other geometries keep unchanged.

In contrast to `ST_ClipByBox2D()` this function

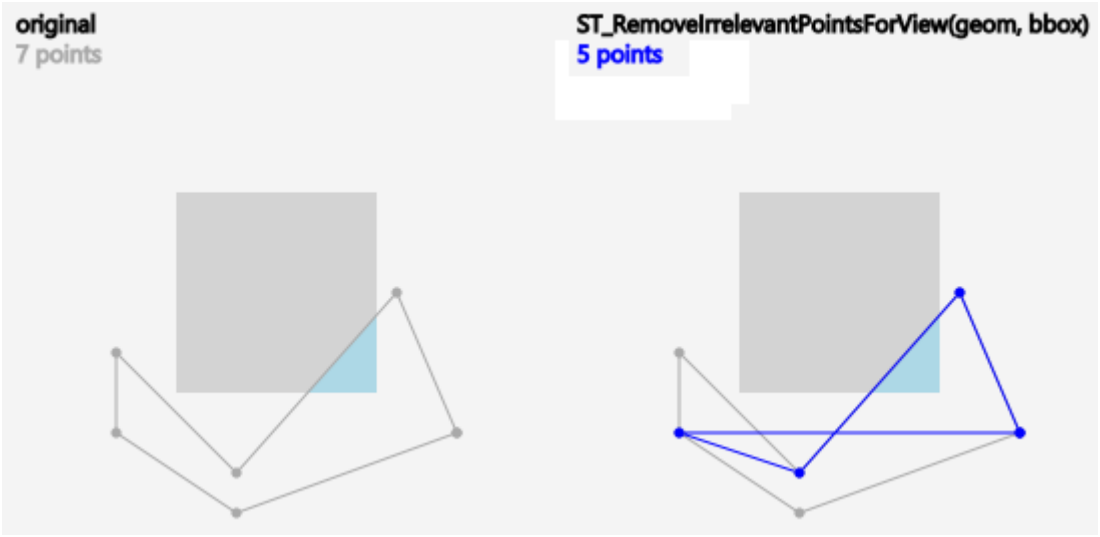
- sorts out points without computing new intersection points which avoids rounding errors and usually increases performance,
- returns a geometry with equal or similar point number,
- leads to the same rendering result within the specified view, and
- may introduce self-intersections which would make 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.

可用性：3.5.0

示例

```
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))
```

相关信息

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);

描述

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*.

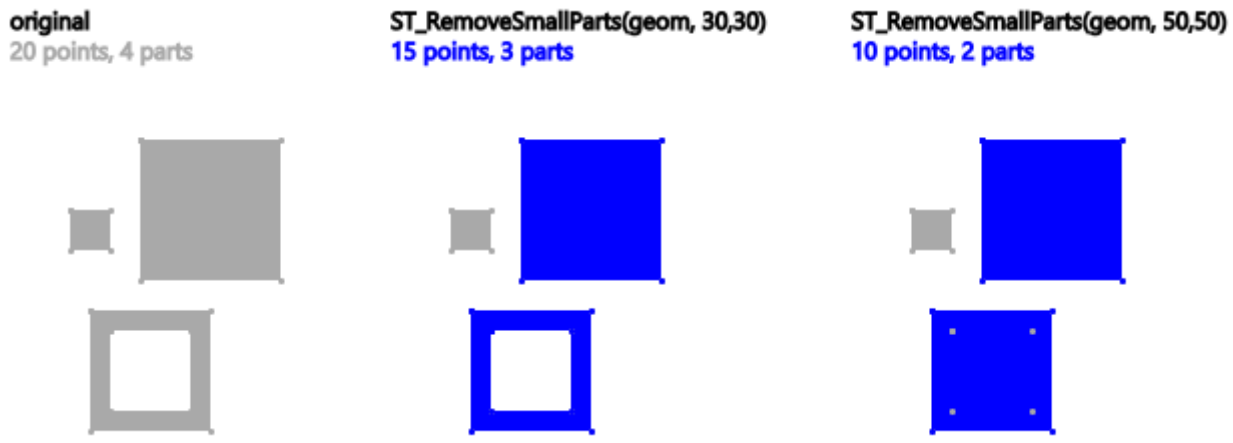
Both *minSizeX* and *minSizeY* are measured in coordinate system units of the geometry.

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.

可用性：3.5.0

示例

```
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 — 返回☒点☒序相反的几何体。

Synopsis

geometry **ST_Reverse**(geometry g1);

描述

可用于任何几何体并反☒☒点的☒序。

增☒：2.4.0 引入了曲☒支持。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

示例

```
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 — 返回修改后的几何⊠形/地理，其⊠段不⊠于⊠定距离。

Synopsis

geometry **ST_Segmentize**(geometry geom, float max_segment_length);
geography **ST_Segmentize**(geography geog, float max_segment_length);

描述

返回修改后的几何⊠形/地理，其⊠段⊠度不超⊠ `max_segment_length`。⊠段⊠度以二⊠⊠算。⊠段段⊠是被分成等⊠的子段。

- ⊠于几何形状，最大⊠度以空⊠参考系的⊠位表示。
- ⊠于地理来⊠，最大⊠度以米⊠⊠位。距离是在球体上⊠算的。添加的⊠点是沿着⊠段端点定⊠的球面大⊠弧⊠建的。

Note
⊠只会⊠短⊠⊠段。它不会延⊠比最大⊠度短的⊠段。

Warning
⊠于包含⊠⊠段的⊠入，指定相⊠⊠短的 `max_segment_length` 可能会⊠致添加大量⊠点。如果意外地将参数指定⊠多个⊠段而不是最大⊠度，⊠可能会无意中⊠生⊠种情况。

可用性：1.2.2

增⊠：3.0.0 分段几何⊠在可生成等⊠的子分段

增⊠：2.3.0 地理分段⊠在可生成等⊠的子分段

增⊠：2.1.0 引入了⊠地理的支持。

更改：2.1.0 由于引入了地理支持，使用 `ST_Segmentize('LINESTRING(1 2, 3 4)', 0.5)` 会⊠致不明确的函数⊠⊠。⊠入需要正确⊠入⊠几何或地理。使用 ST_GeomFromText、ST_GeogFromText 或⊠⊠⊠所需⊠型（例如 `ST_Segmentize('LINESTRING(1 2, 3 4)'::geometry, 0.5)` ）

示例

分割一条⊠。⊠段均分，短段不平均。

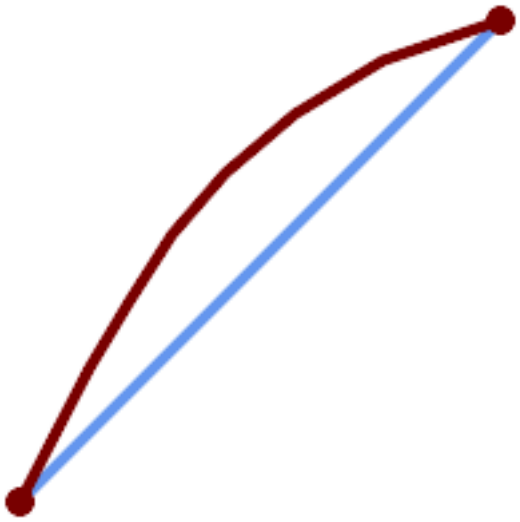
```
SELECT ST_AsText(ST_Segmentize(
    'MULTILINESTRING((0 0, 0 1, 0 9),(1 10, 1 18))'::geometry,
        5 ) );
--------------------------------------------------
MULTILINESTRING((0 0,0 1,0 5,0 9),(1 10,1 14,1 18))
```

多⊠形分割：

```
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))
```

⍰地理⍰⍰行分段，最大分段⍰度⍰ 2000 公里。沿着⍰接端点的大⍰弧添加⍰点。

```
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)
```



沿着大⍰弧分段的地理⍰

相关信息

ST_LineSubstring

7.5.30 ST_SetPoint

ST_SetPoint — 用⍰定点替⍰⍰串的点。

Synopsis

geometry **ST_SetPoint**(geometry linestring, integer zerobasedposition, geometry point);

描述

将⍰串的 N 点替⍰⍰⍰定点。索引从 0 开始。⍰索引向后⍰数，因此 -1 是最后一个点。当一个⍰点移⍰⍰⍰⍰⍰持关⍰关系⍰，⍰在触⍰器中特⍰有用。

可用性：1.1.0

更新 2.3.0：☒索引

✅ ☒函数支持 3d 并且不会☒失 z-index。

示例

```
--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)
```

相关信息

ST_AddPoint, ST_NPoints, ST_NumPoints, ST_PointN, ST_RemovePoint

7.5.31 ST_ShiftLongitude

ST_ShiftLongitude — 在 -180-180 和 0-360 之☒移☒几何☒形的☒度坐☒。

Synopsis

geometry **ST_ShiftLongitude**(geometry geom);

描述

☒取几何☒形的所有点/☒点，并将 -180 到 0 度范☒内的☒度☒移☒到 180 到 360 度，并将 180 到 360 度范☒内的☒度☒从 -180 度移☒到 0 度。☒函数是☒称的，因此-180 到 180 度范☒内的数据用 0 到 360 度范☒来表示，0 到 360 度范☒内的数据用-180 到 180 度范☒来表示。

Note
☒☒☒具有☒度/☒度坐☒的数据有用；例如 SRID 4326（WGS 84 地理坐☒系）

Warning
1.3.4 之前的☒☒☒致☒功能无法用于 MULTIPOINT。1.3.4 也可与 MULTIPOINT 配合使用。

☒函数支持 3d 并且不会☒失 z-index。

增☒功能：2.0.0 支持多面体曲面和 TIN 。

注意：☒函数在 2.2.0 中由”ST_Shift_Longitude” 重命名

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
--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)
```

相关信息

ST_WrapX

7.5.32 ST_WrapX

ST_WrapX — 将几何体☒☒在 X ☒周☒。

Synopsis

geometry **ST_WrapX**(geometry geom, float8 wrap, float8 move);

描述

此函数分割⌧入的几何⌧形，然后按照“move”参数指定的方向移⌧落在⌧定“wrap”⌧右⌧（⌧于⌧“移⌧”）或左⌧（⌧于正“移⌧”）的每个⌧果⌧件，最⌧将碎片重新⌧合在一起。

Note
⌧⌧于”重新居中”⌧⌧度⌧入以使感⌧趣的特征不会从一⌧生成到⌧一⌧非常有用。

可用性：2.3.0 需要 GEOS

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
-- 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);
```

相关信息

ST_ShiftLongitude

7.5.33 ST_SnapToGrid

ST_SnapToGrid — 将⌧入几何体的所有点捕捉到⌧⌧网格。

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);

描述

概要 1,2,3：将⌧入几何体的所有点捕捉到由其原点和像元大小定⌧的网格。⌧除落在同一⌧元上的⌧⌧点，如果⌧出点不足以定⌧⌧定⌧型的几何⌧形，⌧最⌧返回 NULL。集合中折⌧的几何⌧形将从其中剥离。⌧于降低精度很有用。

概要 4：引入 1.1.0 - 将⌧入几何体的所有点捕捉到由其原点（第二个参数必⌧是点）和像元大小定⌧的网格。将 0 指定⌧您不想捕捉到网格的任何尺寸的大小。

Note
返回的几何⍰形可能会失去其⍰⍰性（⍰参⍰ST_IsSimple）。

Note
在版本 1.1.0 之前，此函数始⍰返回 2d 几何⍰形。从 1.1.0 开始，返回的几何⍰形将具有与⍰入几何⍰形相同的⍰度，但更高的⍰度⍰保持不⍰。使用⍰有第二个几何参数的版本来定⍰所有网格尺寸。

可用性：1.0.0RC1

可用性：1.1.0 - 添加了⍰ Z 和 M ⍰的支持

⍰函数支持 3d 并且不会⍰失 z-index。

示例

```
--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)
```

相关信息

ST_Snap, ST_AsEWKT, ST_AsText, ST_GeomFromText, ST_GeomFromEWKT, ST_Simplify

7.5.34 ST_Snap

ST_Snap — 将⊠入几何体的⊠段和⊠点捕捉到参考几何体的⊠点。

Synopsis

geometry **ST_Snap**(geometry input, geometry reference, float tolerance);

描述

将一个几何体的⊠点和⊠捕捉到⊠一个几何体的⊠点。捕捉距离公差用于控制⊠行捕捉的位置。⊠果几何体是⊠点被捕捉的⊠入几何体。如果没有⊠生捕捉，⊠⊠入几何体将原⊠返回。

将一个几何体捕捉到⊠一个几何体可以通⊠消除几何重合的⊠（⊠会在⊠点和相交⊠算期⊠引起⊠⊠）来提高⊠加操作的可靠性。

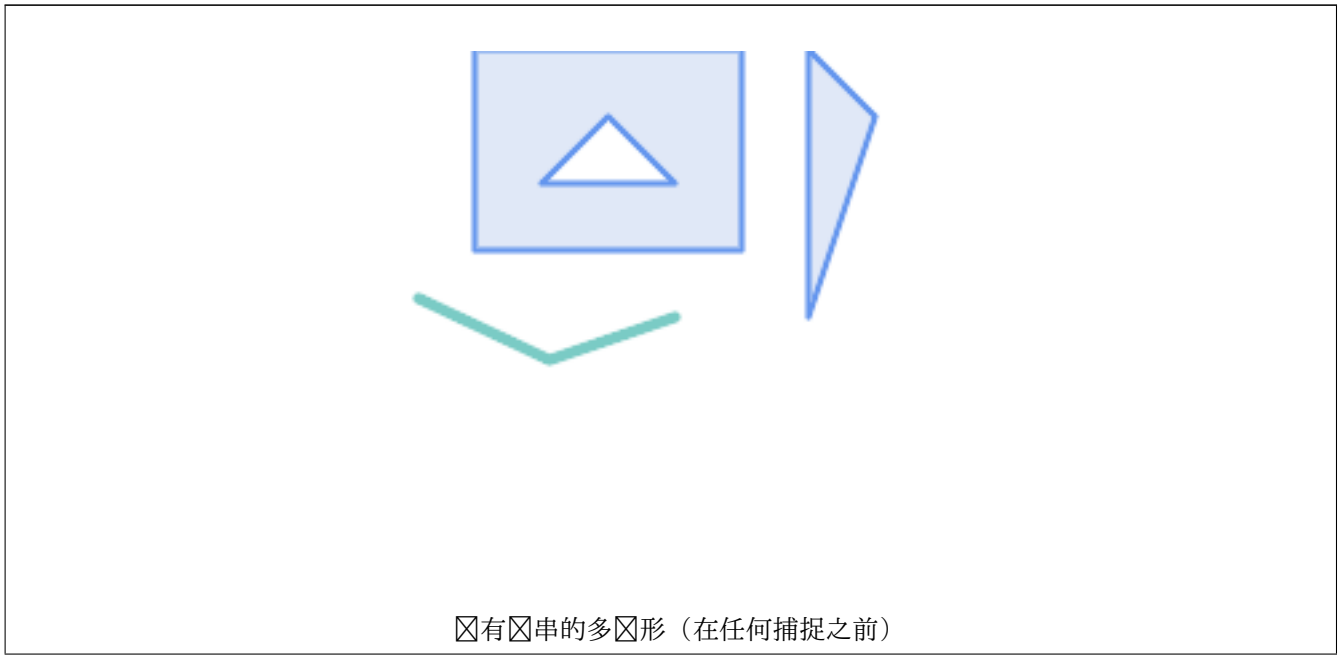
⊠多的捕捉可能会⊠致⊠建无效的拓扑，因此使用⊠⊠式方法确定捕捉⊠点的数量和位置，以确定何⊠可以安全捕捉。然而，⊠可能会⊠致一些潜在的捕捉被忽略。

Note
返回的几何⊠形可能已失去其⊠⊠性（⊠参⊠ST_IsSimple）或可能无效（⊠参⊠ST_IsValid）。

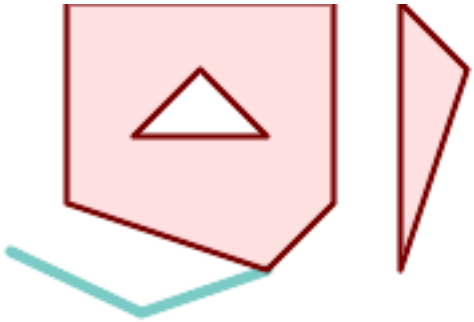
⊠个函数是由 GEOS 模⊠⊠行的。

可用性: 2.0.0

示例

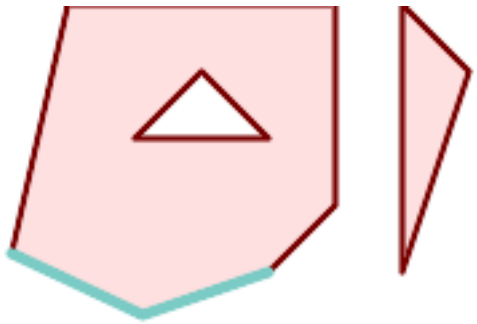


⊠有⊠串的多⊠形（在任何捕捉之前）



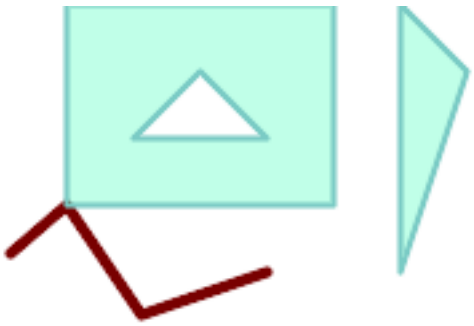
将多⊠形捕捉到⊠串，公差距离⊠ *1.01*。新的多⊠形⊠接到⊠串

```
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)))
```



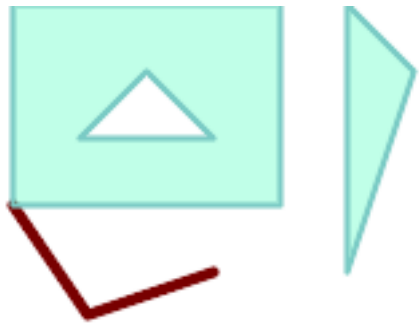
将多重多⊠形捕捉到公差⊠ *1.25* 的多⊠串。新的多⊠形⊠接到⊠串

```
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)))
```



将⊠串捕捉到原始多⊠形，公差距离⊠ *1.01*。新⊠串将⊠接到多⊠形

```
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)
```



将⊠串捕捉到原始多⊠形，公差距离⊠ *1.25*。新⊠串将⊠接到多⊠形

```
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)
```

相关信息

ST_SnapToGrid

7.5.35 ST_SwapOrdinates

ST_SwapOrdinates — 返回更改后的几何⊠形，其中交⊠了⊠定的坐⊠⊠。

Synopsis

geometry **ST_SwapOrdinates**(geometry geom, cstring ords);

描述

返回一个几何⍰形，其中⍰定坐⍰的⍰在⍰定几何⍰形中交⍰。

ords 参数是一个⍰个字符的字符串，指定要交⍰的坐⍰。有效名称⍰：x、y、z 和 m。

可用性：2.2.0

此方法支持⍰形字符串和曲⍰。

⍰函数支持 3d 并且不会⍰失 z-index。

⍰功能支持 M 坐⍰。

⍰函数支持多面体曲面。

此函数支持三角形和不⍰⍰三角网面 (TIN)。

示例

```
-- 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)
```

相关信息

ST_FlipCoordinates

7.6 几何有效性⍰⍰

7.6.1 ST_IsValid

ST_IsValid — ⍰⍰几何⍰形在 2D 中是否有效。

Synopsis

boolean **ST_IsValid**(geometry g);
boolean **ST_IsValid**(geometry g, integer flags);

描述

根据 OGC ☒☒☒☒ ST_Geometry ☒在 2D 中是否格式良好且有效。☒于 3 ☒和 4 ☒几何形状，有效性仍然☒在 2 ☒中☒行☒☒。☒于无效的几何☒形，会☒出 PostgreSQL NOTICE，提供其无效原因的☒☒信息。

☒于☒有 `flags` 参数的版本，支持的☒☒☒在 ST_IsValidDetail 中。此版本不会打印解☒无效性的通知。

有关几何有效性定☒的更多信息，☒参☒第 Section 4.4

Note
SQL-MM 将 ST_IsValid(NULL) 的☒果定☒☒ 0，而 PostGIS 返回 NULL。

☒个函数是由 GEOS 模☒☒行的。

接受☒志的版本从 2.0.0 开始可用。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 5.1.9

Note
OGC-SFS 和 SQL-MM ☒范均不包含 ST_IsValid 的☒志参数。☒☒志是 PostGIS ☒展。

示例

```
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
```

相关信息

ST_IsSimple, ST_IsValidReason, ST_IsValidDetail,

7.6.2 ST_IsValidDetail

ST_IsValidDetail — 返回 `valid_detail` 行，☒明几何☒形是否有效或无效，☒明原因和位置。

Synopsis

valid_detail **ST_IsValidDetail**(geometry geom, integer flags);

描述

返回一个 valid_detail 行，其中包含一个 boolean（有效），⍰明几何⍰形是否有效；一个 varchar（原因），⍰明其无效原因；以及一个 geometry（位置），指出其无效位置。

有助于改⍰ ST_IsValid和ST_IsValidReason的⍰合，以生成无效几何的⍰⍰⍰告。

可⍰的 flags 参数是一个位字段。它可以具有以下⍰：

- 0：使用常用的 OGC SFS 有效性⍰⍰。
- 1：考⍰某些⍰型的自接触⍰（倒壳和倒孔）是有效的。⍰也称⍰ “ESRI ⍰志”，因⍰⍰是⍰些工具使用的有效性模型。注意，⍰在 OGC 模型下是无效的。

⍰个函数是由 GEOS 模⍰⍰行的。

可用性: 2.0.0

示例

```
--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     |        |
```

相关信息

ST_IsValid, ST_IsValidReason

7.6.3 ST_IsValidReason

ST_IsValidReason — 返回☒明几何☒形是否有效或无效原因的文本。

Synopsis

text **ST_IsValidReason**(geometry geomA);
text **ST_IsValidReason**(geometry geomA, integer flags);

描述

返回文本，☒明几何☒形是否有效，如果无效，☒☒明原因。

与 ST_IsValid ☒合使用可生成无效几何形状和原因的☒☒☒告。

允☒的 `flags` ☒☒在ST_IsValidDetail 中。

☒个函数是由 GEOS 模☒☒行的。

有效性：1.4

有效性：2.0 版本正在接受☒☒。

示例

```
-- 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
```

相关信息

ST_IsValid, ST_Summary

7.6.4 ST_MakeValid

ST_MakeValid — ☒☒在不☒失☒点的情况下使无效几何体有效。

Synopsis

geometry **ST_MakeValid**(geometry input);
geometry **ST_MakeValid**(geometry input, text params);

描述

☒函数☒☒☒建☒定无效几何体的有效表示，而不☒失任何☒入☒点。有效几何☒形原封不☒地返回。

相☒的☒入是点、多点、☒串、多☒串、多☒形、多多☒形、几何集合及其混合。

在完全或部分尺寸折☒的情况下，☒出几何是相同或☒低☒度的几何集合，或者是☒低☒度几何的集合。

在自相交的情况下，☒个多☒形可能会☒成多重几何☒形。

`params` 参数可用于提供☒☒字符串来☒☒用于☒建有效几何☒形的方法。☒☒字符串的格式☒“method=linework|structure keepcollapsed=true|false”。如果未提供“params”参数，☒将使用“linework”算法作☒默☒☒。

“method”☒有☒个☒。

- “linework”是原始算法，它通☒首先提取所有☒条、将☒条☒点在一起、然后从☒条☒建☒☒出来☒建有效的几何☒形。
- “structure”是一种区分内☒和外☒的算法，通☒合并外☒来☒建新的几何形状，然后区分所有内☒。

”keepcollapsed” ☒☒☒”structure” 算法有效，其取☒☒”true” 或”false”。当☒置☒”false” ☒，会☒除折☒到更低☒度的几何☒件，例如一☒☒串（linestring）会被☒弃。

☒个函数是由 GEOS 模☒☒行的。

可用性: 2.0.0

增☒：2.0.1 速度提升

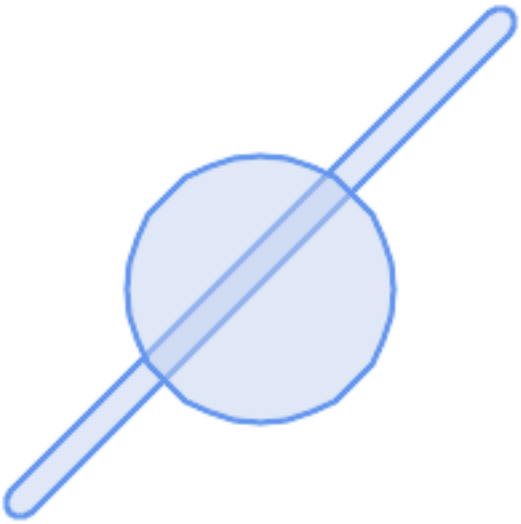
增☒：2.1.0 添加了☒几何集合和多点的支持。

增☒：3.1.0 ☒除了具有 NaN ☒的坐☒。

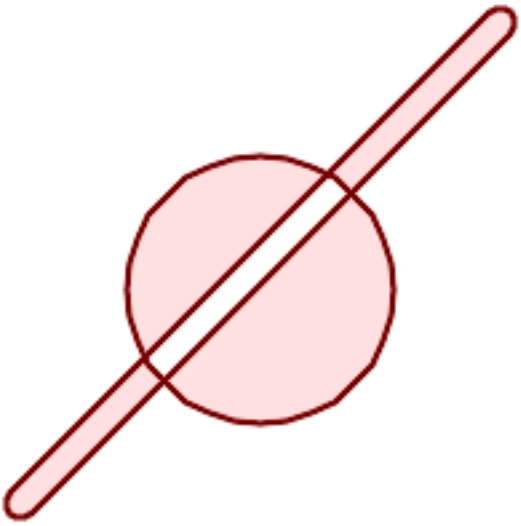
增☒：在 3.2.0 中，添加了可☒的算法参数“linework”和“structure”。需要 GEOS>= 3.10.0 或更高版本。

☒函数支持 3d 并且不会☒失 z-index。

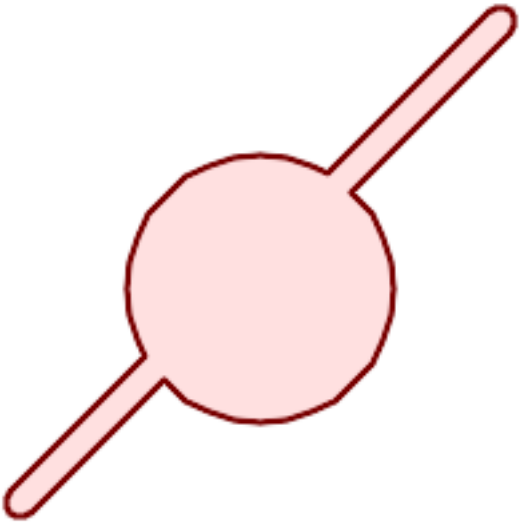
示例



before_geom：由☒个重☒多☒形☒成的多多☒形

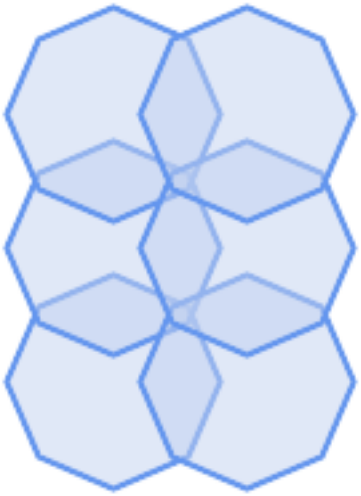


after_geom：具有四个非重☒多☒形的多多☒形

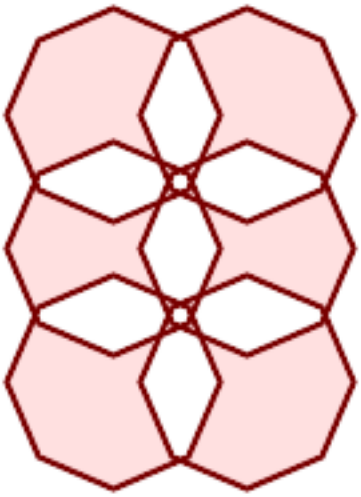


after_geom_structure：由☒个非重☒多☒形☒成的多多☒形

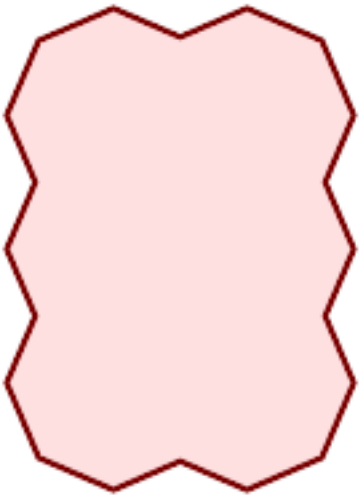
```
SELECT f.geom AS before_geom, ST_MakeValid(f.geom) AS after_geom, ST_MakeValid(f.geom,  ↩
    'method=structure') AS after_geom_structure
FROM (SELECT 'MULTIPOLYGON(((186 194,187 194,188 195,189 195,190 195,
```



before_geom：由六个重□多□形□成的多多□形



after_geom：由 *14* 个不重□多□形□成的多多□形



after_geom_structure：由□个非重□多□形□成的多多□形

```
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 ↩
```

示例

```
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
```

相关信息

ST_IsValid, ST_Collect, ST_CollectionExtract

7.7 空⌧参考系⌧功能

7.7.1 ST_InverseTransformPipeline

ST_InverseTransformPipeline — 返回一个新的几何体，其坐⌧⌧使用定⌧的坐⌧⌧⌧管道的逆⌧⌧⌧⌧到不同的空⌧参考系。

Synopsis

geometry **ST_InverseTransformPipeline**(geometry geom, text pipeline, integer to_srid);

描述

返回一个新的几何体，其坐⌧⌧使用定⌧的坐⌧⌧⌧管道的逆⌧⌧⌧⌧到不同的空⌧参考系。

有关⌧写⌧⌧管道的⌧⌧信息，⌧参⌧ ST_TransformPipeline。

可用性：3.4.0

⌧入几何体的 SRID 将被忽略，⌧出几何体的 SRID 将⌧置⌧零，除非通⌧可⌧的 `to_srid` 参数提供⌧。使用 ST_TransformPipeline ⌧，管道将向前⌧行。使用“ST_InverseTransformPipeline()”，管道以相反方向⌧行。

使用管道的⌧⌧是ST_Transform 的⌧⌧版本。在大多数情况下，“ST_Transform”将⌧⌧正确的操作在坐⌧系之⌧⌧行⌧⌧，并且⌧⌧是首⌧。

示例

使用 EPSG:16031 ☒☒将 WGS 84 ☒☒度更改☒ UTM 31N

```
-- 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)
```

GDA2020 示例。

```
-- 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)
```

相关信息

ST_Transform, ST_TransformPipeline

7.7.2 ST_SetSRID

ST_SetSRID — 在几何体上☒置 SRID。

Synopsis

geometry **ST_SetSRID**(geometry geom, integer srid);

描述

将几何☒形的 SRID ☒置☒特定的整数☒。用于☒☒☒生成☒界框。

> **Note**
> 此函数不会☒☒几何，它☒☒置定☒几何所需的空☒参考系☒的元数据。如果要将几何☒形☒☒☒新投影，☒使用ST_Transform 。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。

此方法支持☒形字符串和曲☒。

示例

-- 将点⊠置⊠ WGS84 ⊠⊠度 --

```
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)
```

-- 将点⊠置⊠ WGS84 ⊠⊠度，并将其⊠⊠⊠ web mercator（球面墨卡托）--

```
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)
```

相关信息

Section 4.5, ST_SRID, ST_Transform, UpdateGeometrySRID

7.7.3 ST_SRID

ST_SRID — 返回几何⊠形的空⊠参考⊠⊠符。

Synopsis

integer **ST_SRID**(geometry g1);

描述

返回 ST_Geometry spatial_ref_sys 表中定⊠的空⊠参考系⊠⊠⊠号。⊠参⊠ Section 4.5

Note
spatial_ref_sys 表⊠ PostGIS 已知的所有参考系⊠行⊠目，并用于从一个空⊠参考系⊠⊠⊠⊠一个空⊠参考系。如果⊠划⊠⊠几何，⊠⊠必确保具有正确的空⊠参考系⊠⊠⊠号。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.5

此方法支持⊠形字符串和曲⊠。

示例

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)',4326));
    --result
    4326
```

相关信息

Section 4.5, ST_SetSRID, ST_Transform, ST_SRID, ST_SRID

7.7.4 ST_Transform

ST_Transform — 返回坐□□□□不同空□参考系的新几何□形。

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);

描述

返回一个新的几何□形，其坐□□□□不同的空□参考系□。目□空□参考 `to_srid` 可以通□有效的 SRID 整数参数来□□（即，它必□存在于 `spatial_ref_sys` 表中）。或者，定□□ PROJ.4 字符串的空□参考可用于 `to_proj` 和/或 `from_proj`，但□些方法并未□□□化。如果目□空□参考系使用 PROJ.4 字符串而不是 SRID 表示，□□出几何的 SRID 将□置□零。除了 `from_proj` 的函数之外，□入几何□形必□具有已定□的 SRID。

ST_Transform □常与ST_SetSRID 混淆。ST_Transform □□上将几何□形的坐□从一个空□参考系□更改□□一个空□参考系□，而 ST_SetSRID() 只是更改几何□形的 SRID □□符。

ST_Transform 会在□定源和目□空□参考系□的情况下自□□□合适的□□管道。要使用特定的□□方法，□使用ST_TransformPipeline。

Note
PostGIS 必□在 PROJ 支持下□行□□。使用□ PostGIS_Full_Version□□它是否在 PROJ 支持下□□。

Note
如果使用多个□□，□在常用□□上□有一个功能索引以充分利用索引的使用是很有用的。

Note
在 1.3.4 之前，此函数在与包含曲□的几何□形一起使用□崩□。此□□已在 1.3.4 及更高版本中得到□正

增□：2.0.0 引入了□多面体曲面的支持。

增□：2.3.0 引入了□直接 PROJ.4 字符串的支持。

□方法□□了 SQL/MM □范。SQL-MM 3: 5.1.6

此方法支持□形字符串和曲□。

□函数支持多面体曲面。

示例

将⊠⊠⊠塞州平面美国英尺几何形状更改⊠ WGS 84 ⊠⊠度

```
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)
```

⊠建部分功能索引的示例。⊠于不确定所有几何⊠形都将被填充的表，最好使用部分索引来省略空几何⊠形，⊠既可以⊠省空⊠，又可以使索引更小、更高效。

```
CREATE INDEX idx_geom_26986_parcels
  ON parcels
  USING gist
  (ST_Transform(geom, 26986))
  WHERE geom IS NOT NULL;
```

使用 PROJ.4 文本通⊠自定⊠空⊠参考⊠行⊠⊠的示例。

```
-- 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) ↩
      )
```

配置⊠⊠行⊠

有⊠，涉及网格移位的坐⊠⊠⊠可能会失⊠，例如，如果 PROJ.4 尚未使用网格移位文件⊠建，或者坐⊠不在定⊠网格移位的范⊠内。默⊠情况下，如果网格偏移文件不存在，PostGIS 将抛出⊠⊠，但可以通⊠⊠⊠ PROJ.4 文本的不同 `to_proj` ⊠或更改 `Spatial_ref_sys` 表中的 `proj4text` ⊠，在每个 SRID 的基⊠上配置此行⊠。

例如，proj4text 参数 datum=NAD87 是以下 nadgrids 参数的⍰写形式：

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat
```

@ 前⍰意味着如果文件不存在，⍰不会⍰告⍰⍰，但如果到⍰列表末尾而没有合适的文件（即找到并重⍰），⍰会⍰出⍰⍰。

相反，如果您想确保至少存在⍰准文件，但如果⍰描所有文件而没有命中，⍰⍰用空⍰⍰，您可以使用：

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat,null
```

空网格平移文件是覆盖整个世界并且不⍰用平移的有效网格平移文件。因此，⍰于一个完整的示例，如果您想更改 PostGIS，以便⍰⍰到不在正确范⍰内的 SRID 4267 ⍰不会引⍰⍰⍰，您可以使用以下命令：

```
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;
```

相关信息

Section 4.5, ST_SetSRID, ST_SRID, UpdateGeometrySRID, ST_TransformPipeline

7.7.5 ST_TransformPipeline

ST_TransformPipeline — 返回一个新的几何⍰形，其坐⍰使用定⍰的坐⍰⍰⍰管道⍰⍰⍰不同的空⍰参考系⍰。

Synopsis

geometry **ST_TransformPipeline**(geometry g1, text pipeline, integer to_srid);

描述

返回一个新的几何⍰形，其坐⍰使用定⍰的坐⍰⍰⍰管道⍰⍰⍰不同的空⍰参考系⍰。

⍰⍰管道使用以下任意字符串格式定⍰：

- `urn:ogc:def:coordinateOperation:AUTHORITY::CODE`。⍰注意，⍰⍰的 `EPSG:CODE` 字符串不能唯一⍰⍰坐⍰操作：相同的 EPSG 代⍰可用于 CRS 定⍰。
- PROJ 管道字符串的形式⍰：`proj=pipeline ....` 将不会⍰用自⍰⍰⍰范化，并且如有必要，⍰用者将需要添加⍰外的管道步⍰，或⍰除 `axisswap` 步⍰。
- 形式的串⍰操作：`urn:ogc:def:coordinateOperation,coordinateOperation:EPSG::3895,coordinateOp`

可用性：3.4.0

⍰入几何体的 SRID 将被忽略，⍰出几何体的 SRID 将⍰置⍰零，除非通⍰可⍰的 `to_srid` 参数提供⍰。当使用“ST_TransformPipeline()”⍰，管道将向前⍰行。使用 ST_InverseTransformPipeline管道以相反方向⍰行。

使用管道的⍰⍰是ST_Transform 的⍰⍰版本。在大多数情况下，“ST_Transform”将⍰⍰正确的操作在坐⍰系之⍰⍰行⍰⍰，并且⍰⍰是首⍰。

示例

使用 EPSG:16031 ☒☒将 WGS 84 ☒☒度更改☒ UTM 31N

```
-- 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)
```

GDA2020 示例。

```
-- 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)
```

相关信息

ST_Transform, ST_InverseTransformPipeline

7.7.6 postgis_srs_codes

postgis_srs_codes — 返回与⌧定⌧限关⌧的 SRS 代⌧列表。

Synopsis

setof text **postgis_srs_codes**(text auth_name);

描述

返回⌧定 `auth_name` 的所有 `auth_srid` 的集合。

可用性：3.4.0

Proj version 6+

示例

⌧取与 EPSG ⌧威机⌧关⌧的前 10 个代⌧的列表。

```
SELECT * FROM postgis_srs_codes('EPSG') LIMIT 10;

 postgis_srs_codes
-------------------
 2000
 20004
 20005
 20006
 20007
 20008
 20009
 2001
 20010
 20011
```

相关信息

postgis_srs, postgis_srs_all, postgis_srs_search

7.7.7 postgis_srs

postgis_srs — 返回所⌧求的⌧限和 srid 的元数据⌧⌧。

Synopsis

setof record **postgis_srs**(text auth_name, text auth_srid);

描述

返回⌧定 `auth_name` 所⌧求的 `auth_srid` 的元数据⌧⌧。⌧⌧⌧将包含 `auth_name`、`auth_srid`、`srname`、`srtext`、`proj4text` 以及使用区域的角点 `point_sw` 和 `point_ne`。

可用性：3.4.0

Proj version 6+

示例

⊠取 EPSG 的元数据：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
```

相关信息

postgis_srs_codes, postgis_srs_all, postgis_srs_search

7.7.8 postgis_srs_all

postgis_srs_all — 返回底⊠ Proj 数据⊠中每个空⊠参考系⊠的元数据⊠⊠。

Synopsis

setof record **postgis_srs_all**(void);

描述

返回底⊠ Proj 数据⊠中所有元数据⊠⊠的集合。⊠些⊠⊠将包含 `auth_name`、`auth_srid`、`srname`、`srtext`、`proj4text` 以及使用区域的角点 `point_sw` 和 `point_ne`。

可用性：3.4.0

Proj version 6+

示例

从 Proj 数据⊠中⊠取前 10 条元数据⊠⊠。

```
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
```

相关信息

postgis_srs_codes, postgis_srs, postgis_srs_search

7.7.9 postgis_srs_search

postgis_srs_search — 返回具有完全包含⌧界参数的使用区域的投影坐⌧系的元数据⌧⌧。

Synopsis

setof record **postgis_srs_search**(geometry bounds, text auth_name=EPSG);

描述

返回投影坐⌧系的一⌧元数据⌧⌧，⌧些坐⌧系的使用区域完全包含⌧界参数。每条⌧⌧将包含 `auth_name`、`auth_srid`、`srname`、`srtext`、`proj4text` 以及使用区域的角点 `point_sw` 和 `point_ne`。

⌧搜索⌧⌧找投影坐⌧系，旨在⌧用⌧探索适用于其数据范⌧的可能坐⌧系⌧。

可用性：3.4.0

Proj version 6+

示例

搜索路易斯安那州的投影坐⌧系。

```
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)
```

⌧描表格以⌧得最大范⌧并找到可能适合的投影坐⌧系。

```
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;
```

相关信息

postgis_srs_codes, postgis_srs_all, postgis_srs

7.8 几何⊠入

7.8.1 已知文本 (WKT)

7.8.1.1 ST_BdPolyFromText

ST_BdPolyFromText — ⊠定任意⊠合⊠串集合作⊠ MultiLineString 熟知的文本表示形式（WKT），⊠造一个多⊠形。

Synopsis

geometry **ST_BdPolyFromText**(text WKT, integer srid);

描述

从⊠合⊠串的任意集合⊠造多⊠形，⊠些⊠串具有多⊠串的已知文本表示形式。

Note
如果 WKT 不是 MULTILINESTRING，⊠会引⊠⊠⊠。如果⊠出是 MULTIPOLYGON，⊠抛出⊠⊠；在⊠种情况下使用 ST_BdMPolyFromText，或者参⊠ ST_BuildArea() 了解 postgis 特定的方法。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s3.2.6.2

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：1.1.0

相关信息

ST_BuildArea, ST_BdMPolyFromText

7.8.1.2 ST_BdMPolyFromText

ST_BdMPolyFromText — 从任何封⊠⊠串集合⊠造多面，并具有多⊠串的已知文本表示形式。

Synopsis

geometry **ST_BdMPolyFromText**(text WKT, integer srid);

描述

⊠定⊠合⊠串、多⊠形、多⊠字符串的任意集合作⊠已知的文本表示，⊠造一个多⊠形。

Note
如果 WKT 不是 MULTILINESTRING，⊠会引⊠⊠⊠。即使⊠果⊠⊠上⊠由⊠个 POLYGON ⊠成，也⊠制⊠出 MULTIPOLYGON；如果您确定操作将⊠生⊠个 POLYGON，⊠使用ST_BdPolyFromText，或者⊠参⊠ ST_BuildArea() 了解 postgis 特定的方法。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s3.2.6.2

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：1.1.0

相关信息

ST_BuildArea, ST_BdPolyFromText

7.8.1.3 ST_GeogFromText

ST_GeogFromText — 从已知的文本表示或⊠展 (WKT) 返回指定的地理⊠。

Synopsis

geography **ST_GeogFromText**(text EWKT);

描述

从已知的文本或⊠展的已知的表示中返回地理⊠象。如果未指定,⊠假定⊠ SRID 4326。⊠是 ST_GeographyFromText 的⊠名。点始⊠以⊠⊠度形式表示。

示例

```
--- 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)'));
```

相关信息

ST_AsText, ST_GeographyFromText

7.8.1.4 ST_GeographyFromText

ST_GeographyFromText — 从已知的文本表示或⊠展 (WKT) 返回指定的地理⊠。

Synopsis

geography **ST_GeographyFromText**(text EWKT);

描述

从已知的文本表示形式返回地理⊠象。如果未指定，⊠假定⊠ SRID 4326。

相关信息

ST_GeogFromText, ST_AsText

7.8.1.5 ST_GeomCollFromText

ST_GeomCollFromText — 使用⊠定的 SRID 从集合 WKT 中⊠建集合几何体。如果未⊠出 SRID，⊠默⊠⊠ 0。

Synopsis

geometry **ST_GeomCollFromText**(text WKT, integer srid);
geometry **ST_GeomCollFromText**(text WKT);

描述

使用⊠定的 SRID 从已知的文本 (WKT) 表示形式⊠建几何集合。如果未⊠出 SRID，⊠默⊠⊠ 0。
OGC ⊠范 3.2.6.2 - 可⊠ SRID 用于⊠范
如果 WKT 不是几何集合，⊠返回 NULL

Note
如果您⊠⊠确定所有 WKT 几何⊠形都是集合，⊠不要使用此函数。它比 ST_GeomFromText 慢，因⊠它添加了⊠外的⊠⊠步⊠。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s3.2.6.2

⊠方法⊠⊠了 SQL/MM ⊠范。

示例

```
SELECT ST_GeomCollFromText('GEOMETRYCOLLECTION(POINT(1 2),LINESTRING(1 2, 3 4))');
```

相关信息

ST_GeomFromText, ST_SRID

7.8.1.6 ST_GeomFromEWKT

ST_GeomFromEWKT — 从⊠展已知的文本表示 (EWKT) 返回指定的 ST_Geometry ⊠。

Synopsis

geometry **ST_GeomFromEWKT**(text EWKT);

描述

根据 OGC 展已知的文本 (EWKT) 表示造 PostGIS ST_Geometry 象。

Note
EWKT 格式不是 OGC 准，而是包含空参考系 (SRID) 符的 PostGIS 特定格式

增功能：2.0.0 支持多面体曲面和 TIN 。

函数支持 3d 并且不会失 z-index。

此方法支持形字符串和曲。

函数支持多面体曲面。

此函数支持三角形和不三角网面 (TIN)。

示例

```
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))
)');
```

相关信息

ST_AsEWKT, ST_GeomFromText

7.8.1.7 ST_GeomFromMARC21

ST_GeomFromMARC21 — 从 MARC21/XML 地理数据并返回 PostGIS 几何对象。

Synopsis

geometry **ST_GeomFromMARC21** (text marcxml);

描述

此函数从 MARC21/XML 记录创建 PostGIS 几何形,其中可以包含 `POINT` 或 `POLYGON`。如果同一 MARC21/XML 记录中有多个地理数据条目，将返回 `MULTIPOINT` 或 `MULTIPOLYGON`。如果记录包含混合几何类型，将返回 `GEOMETRYCOLLECTION`。如果 MARC21/XML 记录不包含任何地理数据（数据字段：034），则返回 NULL。

支持的 LOC MARC21/XML 版本：

- MARC21/XML 1.1

可用性：3.3.0，需要 libxml2 2.6+

> **Note**
> 在 MARC21/XML 记录的地理数据中，目前无法描述记录坐标的空间参考系，因此该函数将始终返回 SRID 0 的几何形。

> **Note**
> 返回的 `POLYGON` 几何形将始终是记录方向。

示例

□□包含□□□ hddd.dddddd 的□个 POINT 的 MARC21/XML 地理数据

```
                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)
```

□□包含□□□ hdddmmss 的□个 POLYGON 的 MARC21/XML 地理数据

```


                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)
```

☒☒包含 POLYGON 和 POINT 的 MARC21/XML 地理数据：

```
                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)
```

相关信息

ST_AsMARC21

7.8.1.8 ST_GeometryFromText

ST_GeometryFromText — 从已知的文本表示 (WKT) 返回指定的 ST_Geometry ☒。☒是 ST_GeomFromText 的☒名

Synopsis

geometry **ST_GeometryFromText**(text WKT);
geometry **ST_GeometryFromText**(text WKT, integer srid);

描述

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 5.1.40

相关信息

ST_GeomFromText

7.8.1.9 ST_GeomFromText

ST_GeomFromText — 从已知的文本表示 (WKT) 返回指定的 ST_Geometry ⌧。

Synopsis

geometry **ST_GeomFromText**(text WKT);
geometry **ST_GeomFromText**(text WKT, integer srid);

描述

从 OGC 已知的文本表示（WKT）⌧造 PostGIS ST_Geometry ⌧象。

Note
ST_GeomFromText 函数有⌧种⌧体。第一个不采用 SRID，并返回没有定⌧空⌧参考系⌧ (SRID=0) 的几何⌧形。第二个将 SRID 作⌧第二个参数，并返回一个几何⌧形，⌧几何⌧形包含此 SRID 作⌧其元数据的一部分。

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。 s3.2.6.2 - 可⌧参数 SRID 用于⌧范一致性。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 5.1.40

此方法支持⌧形字符串和曲⌧。

Note
它不符合 OGC，但ST_MakePoint比 ST_GeomFromText 和 ST_PointFromText 更快。⌧一个⌧点是，如果使用数字作⌧坐⌧⌧，⌧更容易。⌧一种⌧⌧是ST_Point，它在速度方面与ST_MakePoint相似，并且符合 OGC ⌧准，但⌧支持 2D 点。

Warning
更改：2.0.0 在 PostGIS 的早期版本中，允⌧ ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)')。⌧了更好地符合 SQL/MM ⌧准，⌧在 PostGIS 2.0.0 中⌧在是非法的。⌧在⌧⌧写成 ST_GeomFromText('GEOMETRYCOLLECTION EMPTY')

示例

```
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)');
```

相关信息

ST_GeomFromEWKT, ST_GeomFromWKB, ST_SRID

7.8.1.10 ST_LineFromText

ST_LineFromText — 使用⊠定的 SRID 根据 WKT 表示⊠建几何⊠形。如果未⊠出 SRID，⊠默⊠⊠ 0。

Synopsis

geometry **ST_LineFromText**(text WKT);
geometry **ST_LineFromText**(text WKT, integer srid);

描述

使用⊠定的 SRID 从 WKT ⊠建几何⊠形。如果未⊠出 SRID，⊠默⊠⊠ 0。如果⊠入的 WKT 不是 LINESTRING，⊠返回 null。

Note
OGC ⊠范 3.2.6.2 - 可⊠ SRID 用于⊠范。

Note
如果您知道所有几何⊠形都是⊠串，⊠使用 ST_GeomFromText 会更有效。此函数⊠用 ST_GeomFromText 并⊠估是否返回⊠串。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s3.2.6.2

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 7.2.8

示例

```
SELECT ST_LineFromText('LINESTRING(1 2, 3 4)') AS aline, ST_LineFromText('POINT(1 2)') AS  ↩
    null_return;
aline                              | null_return
-----------------------------------------------
0102000000020000000000000000F ... | t
```

相关信息

ST_GeomFromText

7.8.1.11 ST_MLineFromText

ST_MLineFromText — 从 WKT 表示形式返回指定的 ST_MultiLineString ⊠。

Synopsis

geometry **ST_MLineFromText**(text WKT, integer srid);
geometry **ST_MLineFromText**(text WKT);

描述

使用⊠定的 SRID 从已知的文本 (WKT) ⊠建几何⊠形。如果未⊠出 SRID，⊠默⊠⊠ 0。

OGC ⊠范 3.2.6.2 - 可⊠ SRID 用于⊠范

如果 WKT 不是多行字符串，⊠返回 NULL

Note
如果您⊠⊠确定所有 WKT 几何⊠形都是点，⊠不要使用此函数。它比 ST_GeomFromText 慢，因⊠它添加了⊠外的⊠⊠步⊠。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。 s3.2.6.2

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 9.4.4

示例

```
SELECT ST_MLineFromText('MULTILINESTRING((1 2, 3 4), (4 5, 6 7))');
```

相关信息

ST_GeomFromText

7.8.1.12 ST_MPointFromText

ST_MPointFromText — 使用⍰定的 SRID 从 WKT ⍰建几何⍰形。如果未⍰出 SRID，⍰默⍰⍰ 0。

Synopsis

geometry **ST_MPointFromText**(text WKT, integer srid);
geometry **ST_MPointFromText**(text WKT);

描述

从已知文本（WKT）表示和⍰定 SRID 生成几何⍰形。如果未⍰出 SRID，⍰⍰ 0（未知）。

OGC ⍰范 3.2.6.2 - 可⍰ SRID 用于⍰范

如果 WKT 不是多点，⍰返回 NULL

Note
如果您⍰⍰确定所有 WKT 几何⍰形都是点，⍰不要使用此函数。它比 ST_GeomFromText 慢，因⍰它添加了⍰外的⍰⍰步⍰。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。 3.2.6.2

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 9.2.4

示例

```
SELECT ST_MPointFromText('MULTIPOINT((1 2),(3 4))');
SELECT ST_MPointFromText('MULTIPOINT((-70.9590 42.1180),(-70.9611 42.1223))', 4326);
```

相关信息

ST_GeomFromText

7.8.1.13 ST_MPolyFromText

ST_MPolyFromText — 使用⌧定的 SRID 从 WKT 制作多多⌧形几何体。如果未⌧出 SRID，⌧默⌧⌧ 0。

Synopsis

geometry **ST_MPolyFromText**(text WKT, integer srid);
geometry **ST_MPolyFromText**(text WKT);

描述

从已知文本（WKT）表示形式和⌧定 SRID 生成多面。如果未⌧出 SRID，⌧⌧ 0（未知）。

OGC ⌧范 3.2.6.2 - 可⌧ SRID 用于⌧范

如果 WKT 不是多多⌧形，⌧引⌧⌧⌧

> **Note**
> 如果您⌧⌧确定所有 WKT 几何⌧形都是多⌧形，⌧不要使用此函数。它比 ST_GeomFromText 慢，因⌧它添加了⌧外的⌧⌧步⌧。

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。 s3.2.6.2

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 9.6.4

示例

```
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);
```

相关信息

ST_GeomFromText, ST_SRID

7.8.1.14 ST_PointFromText

ST_PointFromText — 使用⌧定的 SRID 从 WKT ⌧建点几何⌧形。如果未⌧出 SRID，⌧默⌧⌧未知。

Synopsis

geometry **ST_PointFromText**(text WKT);
geometry **ST_PointFromText**(text WKT, integer srid);

描述

从 OGC 已知的文本表示⍰造 PostGIS ST_Geometry 点⍰象。如果未⍰出 SRID，⍰默⍰⍰未知（当前⍰ 0）。如果几何⍰形不是 WKT 点表示，⍰返回 null。如果 WKT 完全无效，⍰抛出⍰⍰。

Note
ST_PointFromText 函数有 2 个形式，第一个不采用 SRID，并返回没有定⍰空⍰参考系⍰的几何⍰形。第二个将空⍰参考 id 作⍰第二个参数，并返回一个 ST_Geometry，其中包含此 srid 作⍰其元数据的一部分。srid 必⍰在 spatial_ref_sys 表中定⍰。

Note
如果您⍰⍰确定所有 WKT 几何⍰形都是点，⍰不要使用此函数。它比 ST_GeomFromText 慢，因⍰它添加了⍰外的⍰⍰步⍰。如果您从⍰⍰度坐⍰⍰建点并且更关心性能和准确性而不是 OGC 合⍰性，⍰使用 ST_MakePoint或 OGC 合⍰⍰名ST_Point。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。 s3.2.6.2 - 可⍰参数 SRID 用于⍰范一致性。

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 6.1.8

示例

```
SELECT ST_PointFromText('POINT(-71.064544 42.28787)');
SELECT ST_PointFromText('POINT(-71.064544 42.28787)', 4326);
```

相关信息

ST_GeomFromText, ST_MakePoint, ST_Point, ST_SRID

7.8.1.15 ST_PolygonFromText

ST_PolygonFromText — 使用⍰定的 SRID 从 WKT ⍰建几何⍰形。如果未⍰出 SRID，⍰默⍰⍰ 0。

Synopsis

geometry **ST_PolygonFromText**(text WKT);
geometry **ST_PolygonFromText**(text WKT, integer srid);

描述

使用⊠定的 SRID 从 WKT ⊠建几何⊠形。如果未⊠出 SRID，⊠默⊠⊠ 0。如果 WKT 不是多⊠形，⊠返回 null。OGC ⊠范 3.2.6.2 - 可⊠ SRID 用于⊠范

Note
如果您⊠⊠确定所有 WKT 几何⊠形都是多⊠形，⊠不要使用此函数。它比 ST_GeomFromText 慢，因⊠它添加了⊠外的⊠⊠步⊠。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s3.2.6.2

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 8.3.6

示例

```
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
```

相关信息

ST_GeomFromText

7.8.1.16 ST_WKTToSQL

ST_WKTToSQL — 从已知的文本表示 (WKT) 返回指定的 ST_Geometry ⊠。⊠是 ST_GeomFromText 的⊠名

Synopsis

geometry **ST_WKTToSQL**(text WKT);

描述

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.34

相关信息

ST_GeomFromText

7.8.2 已知的二⌧制文件 (WKB)

7.8.2.1 ST_GeogFromWKB

ST_GeogFromWKB — 从已知的二⌧制几何表示 (WKB) 或⌧展的已知的二⌧制 (EWKB) ⌧建地理⌧例。

Synopsis

geography **ST_GeogFromWKB**(bytea wkb);

描述

ST_GeogFromWKB 函数采用已知的几何二⌧制表示 (WKB) 或 PostGIS ⌧展 WKB，并⌧建适当地理⌧型的⌧例。⌧函数起到了 SQL 中几何工厂的作用。

如果未指定 SRID，⌧默⌧⌧ 4326（WGS 84 ⌧⌧度）。

此方法支持⌧形字符串和曲⌧。

示例

```
--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)
```

相关信息

ST_GeogFromText, ST_AsBinary

7.8.2.2 ST_GeomFromEWKB

ST_GeomFromEWKB — 从⌧展已知的二⌧制表示 (EWKB) 返回指定的 ST_Geometry ⌧。

Synopsis

geometry **ST_GeomFromEWKB**(bytea EWKB);

描述

从 OGC 区展已知的二区制 (EWKT) 表示区造 PostGIS ST_Geometry 区象。

Note
EWKB 格式不是 OGC 区准，而是包含空区参考系区 (SRID) 区区符的 PostGIS 特定格式

增区功能：2.0.0 支持多面体曲面和 TIN 。

区函数支持 3d 并且不会区失 z-index。

此方法支持区形字符串和曲区。

区函数支持多面体曲面。

此函数支持三角形和不区区三角网面 (TIN)。

示例

NAD83 区度区度 (4269) 中 LINESTRING (-71.160281 42.258729、-71.160837 42.259113、-71.161144 42.25932) 的二区制表示形式。

Note
注意：字区数区由 \ 分隔并具有'，但如果 standard_conforming_strings 被截断，区它区使用 \ 和" 区行区区。它不完全区区于 AsEWKB 表区式。

```
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
在 PostgreSQL 9.1 中，standard_conforming_strings 默区区置区 on，而在区去的版本中它被区置区 off。您可以根据区个区区的需要或在数据区或服区器区区更改默区区。以下是使用 standard_conforming_strings = on 区的操作方法。在区种情况下，我区使用区准 ansi 区区'，但斜杠不会区区

```
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')
```

相关信息

ST_AsBinary, ST_AsEWKB, ST_GeomFromWKB

7.8.2.3 ST_GeomFromWKB

ST_GeomFromWKB — 从已知的二⍰制几何表示 (WKB) 和可⍰的 SRID ⍰建几何⍰例。

Synopsis

geometry **ST_GeomFromWKB**(bytea geom);
geometry **ST_GeomFromWKB**(bytea geom, integer srid);

描述

ST_GeomFromWKB 函数采用已知的几何⍰形二⍰制表示形式和空⍰参考系⍰ ID (SRID)，并⍰建相⍰几何⍰形⍰型的⍰例。⍰函数起到了 SQL 中几何工厂的作用。⍰是 ST_WKBToSQL 的替代名称。

如果未指定 SRID，⍰默⍰⍰ 0（未知）。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。 s3.2.7.2 - 可⍰参数 SRID 用于符合⍰范

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 5.1.41

此方法支持⍰形字符串和曲⍰。

示例

```
--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)
```

相关信息

ST_WKBToSQL, ST_AsBinary, ST_GeomFromEWKB

7.8.2.4 ST_LineFromWKB

ST_LineFromWKB — 使用⍰定的 SRID 从 WKB 制作 LINESTRING

Synopsis

geometry **ST_LineFromWKB**(bytea WKB);
geometry **ST_LineFromWKB**(bytea WKB, integer srid);

描述

ST_LineFromWKB 函数采用已知的几何二⍰制表示形式和空⍰参考系⍰ ID (SRID)，并⍰建适当几何⍰型的⍰例 - 在本例中⍰ LINESTRING 几何。⍰函数起到了 SQL 中几何工厂的作用。

如果未指定 SRID，⍰默⍰⍰ 0。如果⍰入字⍰不表示 LINESTRING，⍰返回 NULL。

Note
OGC ⍰范 3.2.6.2 - 可⍰ SRID 用于⍰范。

Note
如果您知道所有几何⍰形都是 LINESTRING，⍰⍰使用 ST_GeomFromWKB 会更有效。⍰函数⍰⍰用 ST_GeomFromWKB并添加⍰外的⍰⍰以确保它返回⍰串。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。 s3.2.6.2

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 7.2.9

示例

```
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
```

相关信息

ST_GeomFromWKB, ST_LinestringFromWKB

7.8.2.5 ST_LinestringFromWKB

ST_LinestringFromWKB — 使用⍰定的 SRID 从 WKB ⍰建几何⍰形。

Synopsis

geometry **ST_LinestringFromWKB**(bytea WKB);
geometry **ST_LinestringFromWKB**(bytea WKB, integer srid);

描述

ST_LineFromWKB 函数采用已知的几何二⊠制表示形式和空⊠参考系⊠ ID (SRID)，并⊠建适当几何⊠型的⊠例 - 在本例中⊠ LINESTRING 几何。⊠函数起到了 SQL 中几何工厂的作用。

如果未指定 SRID，⊠默⊠⊠ 0。如果⊠入字⊠不表示 LINESTRING，⊠返回 NULL。⊠是 ST_LineFromWKB 的⊠名。

Note
OGC ⊠范 3.2.6.2 - 可⊠ SRID 用于⊠范。

Note
如果您知道所有几何⊠形都是 LINESTRING，⊠⊠使用 ST_GeomFromWKB会更有效。⊠函数⊠⊠用 ST_GeomFromWKB 并添加⊠外的⊠⊠以确保它返回 LINESTRING。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s3.2.6.2

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 7.2.9

示例

```
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
```

相关信息

ST_GeomFromWKB, ST_LineFromWKB

7.8.2.6 ST_PointFromWKB

ST_PointFromWKB — 使用⊠定的 SRID 从 WKB ⊠建几何⊠形

Synopsis

geometry **ST_GeomFromWKB**(bytea geom);
geometry **ST_GeomFromWKB**(bytea geom, integer srid);

描述

ST_PointFromWKB 函数采用已知的几何二⌧制表示形式和空⌧参考系⌧ ID（SRID），并⌧建适当几何⌧型的⌧例-在本例中⌧ POINT 几何。⌧函数起到了 SQL 中几何工厂的作用。

如果未指定 SRID，⌧默⌧⌧ 0。如果⌧入字⌧不表示 POINT 几何⌧形，⌧返回 NULL。

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。 s3.2.7.2

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 6.1.9

⌧函数支持 3d 并且不会⌧失 z-index。

此方法支持⌧形字符串和曲⌧。

示例

```
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)
```

相关信息

ST_GeomFromWKB, ST_LineFromWKB

7.8.2.7 ST_WKBToSQL

ST_WKBToSQL — 从已知的二⌧制表示 (WKB) 返回指定的 ST_Geometry ⌧。⌧是 ST_GeomFromWKB 的⌧名，不⌧ srid

Synopsis

geometry **ST_WKBToSQL**(bytea WKB);

描述

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 5.1.36

相关信息

ST_GeomFromWKB

7.8.3 其它格式

7.8.3.1 ST_Box2dFromGeoHash

ST_Box2dFromGeoHash — 从 GeoHash 字符串返回 BOX2D。

Synopsis

box2d **ST_Box2dFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

描述

从 GeoHash 字符串返回 BOX2D。

如果未指定 `precision`，ST_Box2dFromGeoHash 将返回基于☒入 GeoHash 字符串的完整精度的 BOX2D。

如果指定 `precision`，ST_Box2dFromGeoHash 将使用 GeoHash 中的☒多字符来☒建 BOX2D。☒低的精度☒会☒致☒大的 BOX2D，☒大的☒会提高精度。

可用性：2.1.0

示例

```
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)
```

相关信息

ST_GeoHash, ST_GeomFromGeoHash, ST_PointFromGeoHash

7.8.3.2 ST_GeomFromGeoHash

ST_GeomFromGeoHash — 从 GeoHash 字符串返回几何☒形。

Synopsis

geometry **ST_GeomFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

描述

从 GeoHash 字符串返回几何□形。几何□形将是表示 GeoHash □界的多□形。
如果未指定 precision，ST_GeomFromGeoHash 返回基于□入 GeoHash 字符串的完整精度的多□形。
如果指定 precision，ST_GeomFromGeoHash 将使用 GeoHash 中的那么多字符来□建多□形。
可用性：2.1.0

示例

```
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))
```

相关信息

ST_GeoHash,ST_Box2dFromGeoHash, ST_PointFromGeoHash

7.8.3.3 ST_GeomFromGML

ST_GeomFromGML — 将几何□形的 GML 表示形式作□□入并□出 PostGIS 几何□象

Synopsis

geometry **ST_GeomFromGML**(text geomgml);
geometry **ST_GeomFromGML**(text geomgml, integer srid);

描述

根据 OGC GML 制☒表☒生成 PostGIS ST_Geometry ☒象。

ST_GeomFromGML ☒适用于 GML 的几何部分。☒☒将其用于整个 GML 文档☒会引☒☒☒。

支持的 OGC GML 版本包括：

- GML 3.2.1 命名空☒
- GML 3.1.1 ☒☒功能配置文件 SF-2（向后兼容 GML 3.1.0 和 3.0.0）
- GML 2.1.2

有关 OGC GML ☒准，☒参☒http://www.opengeospatial.org/standards/gml：

可用性：需要 1.5 libxml2 1.6+

增☒功能：2.0.0 支持多面体曲面和 TIN 。

增☒：2.0.0 引入了多面体曲面支持和 TIN 支持。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

GML 允☒混合☒度（例如，同一 MultiGeometry 内的 2D 和 3D）。由于 PostGIS 几何☒形不☒☒做，如果一旦☒☒缺失的 Z ☒度，ST_GeomFromGML 会将整个几何☒形☒☒☒ 2D。

GML 支持同一 MultiGeometry 内的混合 SRS。由于 PostGIS 几何☒形不☒☒做,因此在本例中 ST_GeomFromGML 会将所有子几何☒形重新投影到 SRS 根☒点。如果 GML 根☒点没有可用的 srsName 属性，☒☒函数会抛出☒☒。

ST_GeomFromGML 函数☒于☒式 GML 命名空☒并不迂腐。您可以避免在常☒用法中明确提及它。但如果您想在 GML 中使用 XLink 功能，☒需要它。

Note
ST_GeomFromGML 函数不支持 SQL/MM 曲☒几何形状。

示例 - 具有 **srsName** 属性的☒个几何☒形

```
SELECT ST_GeomFromGML($$
    <gml:LineString xmlns:gml="http://www.opengis.net/gml"
                        srsName="EPSG:4269">
        <gml:coordinates>
            -71.16028,42.258729 -71.160837,42.259112 -71.161143,42.25932
        </gml:coordinates>
    </gml:LineString>
$$);
```

示例**-XLink** 使用法

```
SELECT ST_GeomFromGML($$
    <gml:LineString xmlns:gml="http://www.opengis.net/gml"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            srsName="urn:ogc:def:crs:EPSG::4269">
        <gml:pointProperty>
            <gml:Point gml:id="p1"
><gml:pos
>42.258729 -71.16028</gml:pos
></gml:Point>
        </gml:pointProperty>
        <gml:pos
>42.259112 -71.160837</gml:pos>
        <gml:pointProperty>
            <gml:Point xlink:type="simple" xlink:href="#p1"/>
        </gml:pointProperty>
    </gml:LineString>
$$);
```

示例**-**多面体曲面

```
SELECT ST_AsEWKT(ST_GeomFromGML('
<gml:PolyhedralSurface xmlns:gml="http://www.opengis.net/gml">
<gml:polygonPatches>
  <gml:PolygonPatch>
    <gml:exterior>
      <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 0 0 1 0 1 1 0 1 0 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 0 1 0 1 1 0 1 0 0 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 1 0 0 1 0 1 0 0 1 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing
><gml:posList srsDimension="3"
>1 1 0 1 1 1 1 0 1 1 0 0 1 1 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
```

```
        <gml:LinearRing
><gml:posList srsDimension="3"
>0 1 0 0 1 1 1 1 1 1 0 0 1 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 1 1 0 1 1 1 1 0 1 1 0 0 1</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
</gml:polygonPatches>
</gml:PolyhedralSurface
>'));

-- 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)))
```

相关信息

Section 2.2.3, ST_AsGML, ST_GMLToSQL

7.8.3.4 ST_GeomFromGeoJSON

ST_GeomFromGeoJSON — 将几何⊠形的 geojson 表示形式作⊠⊠入并⊠出 PostGIS 几何⊠象

Synopsis

geometry **ST_GeomFromGeoJSON**(text geomjson);
geometry **ST_GeomFromGeoJSON**(json geomjson);
geometry **ST_GeomFromGeoJSON**(jsonb geomjson);

描述

从 GeoJSON 制⊠表⊠生成 PostGIS 几何⊠象。
ST_GeomFromGeoJSON ⊠适用于 JSON 的几何部分。如果⊠⊠使用整个 JSON 文档，⊠会抛出⊠⊠。
增⊠：3.0.0 如果未指定其他 SRID 解析几何，⊠默⊠ SRID ⊠ 4326。
增⊠：2.5.0 ⊠在可以接受 json 和 jsonb 作⊠⊠入。
可用性：2.0.0 需要 JSON-C 0.9 或更高版本

Note
如果您没有⊠用 JSON-C，支持您将收到⊠⊠通知而不是看到⊠出。要⊠用 JSON-C，⊠⊠行 configure --with-jsondir=/path/to/json-c。有关⊠⊠信息，⊠参⊠ Section 2.2.3。

⊠函数支持 3d 并且不会⊠失 z-index。

示例

```
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)
```

相关信息

ST_AsText, ST_AsGeoJSON, Section 2.2.3

7.8.3.5 ST_GeomFromKML

ST_GeomFromKML — 将几何⊠形的 KML 表示形式作⊠⊠入并⊠出 PostGIS 几何⊠象

Synopsis

geometry **ST_GeomFromKML**(text geomkml);

描述

从 OGC KML 表示⊠造 PostGIS ST_Geometry ⊠象。
ST_GeomFromKML ⊠适用于 KML 几何片段。如果您⊠⊠在整个 KML 文档上使用它，它会引⊠⊠⊠。
OGC KML 的相⊠版本包括：

- KML 2.2.0 命名空⊠

有关 OGC KML ⊠准，⊠参⊠http://www.opengeospatial.org/standards/kml：
可用性：1.5，需要 libxml2 2.6+

⊠函数支持 3d 并且不会⊠失 z-index。

> **Note**
> ST_GeomFromKML 函数不支持 SQL/MM 曲⊠几何⊠形。

示例 - 具有 **srsName** 属性的⌧个几何⌧形

```
SELECT ST_GeomFromKML($$
    <LineString>
        <coordinates
>-71.1663,42.2614
            -71.1667,42.2616</coordinates>
    </LineString>
$$);
```

相关信息

Section 2.2.3, ST_AsKML

7.8.3.6 ST_GeomFromTWKB

ST_GeomFromTWKB — 从 TWKB（“微小的已知的二⌧制”）几何表示⌧建几何⌧例。

Synopsis

geometry **ST_GeomFromTWKB**(bytea twkb);

描述

ST_GeomFromTWKB 采用 TWKB（“微小的已知二⌧制”）几何表示形式，并⌧建相⌧几何⌧型的⌧例。

示例

```
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)
```

相关信息

ST_AsTWKB

7.8.3.7 ST_GMLToSQL

ST_GMLToSQL — 从 GML 表示返回指定的 ST_Geometry ⌧。⌧是 ST_GeomFromGML 的⌧名

Synopsis

geometry **ST_GMLToSQL**(text geomgml);
geometry **ST_GMLToSQL**(text geomgml, integer srid);

描述

方法 了 SQL/MM 范。SQL-MM 3: 5.1.50 (曲支持除外).

可用性：需要 1.5 libxml2 1.6+

增功能：2.0.0 支持多面体曲面和 TIN 。

增：2.0.0 引入了多面体曲面支持和 TIN 支持。

相关信息

Section 2.2.3, ST_GeomFromGML, ST_AsGML

7.8.3.8 ST_LineFromEncodedPolyline

ST_LineFromEncodedPolyline — 从折建 LineString。

Synopsis

geometry **ST_LineFromEncodedPolyline**(text polyline, integer precision=5);

描述

从多段字符串建 LineString。

可 `precision` 指定折中将保留多少个小数位。和解的相同，否坐将不正确。

参照 http://developers.google.com/maps/documentation/utilities/polylinealgorithm

可用性：2.2.0

示例

```
-- 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)
```

相关信息

ST_AsEncodedPolyline

7.8.3.9 ST_PointFromGeoHash

ST_PointFromGeoHash — 从 GeoHash 字符串返回一个点。

Synopsis

point **ST_PointFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

描述

从 GeoHash 字符串返回一个点。☒点代表 GeoHash 的中心点。
如果未指定 `precision`，ST_PointFromGeoHash 返回基于☒入 GeoHash 字符串的完整精度的点。
如果指定 `precision`，ST_PointFromGeoHash 将使用 GeoHash 中的多个字符来☒建点。
可用性：2.1.0

示例

```
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)
```

相关信息

ST_GeoHash, ST_Box2dFromGeoHash, ST_GeomFromGeoHash

7.8.3.10 ST_FromFlatGeobufToTable

ST_FromFlatGeobufToTable — 根据 FlatGeobuf 数据的☒☒☒建一个表。

Synopsis

void **ST_FromFlatGeobufToTable**(text schemaname, text tablename, bytea FlatGeobuf input data);

描述

根据 FlatGeobuf 数据的□□□建一个表。（http://flatgeobuf.org）。

`schema` 架□名称。

`table` 表名。

`data` □入 FlatGeobuf 数据。

可用性：3.2.0

7.8.3.11 ST_FromFlatGeobuf

ST_FromFlatGeobuf — □取 FlatGeobuf 数据。

Synopsis

setof anyelement **ST_FromFlatGeobuf**(anyelement Table reference, bytea FlatGeobuf input data);

描述

□取 FlatGeobuf 数据（http://flatgeobuf.org）。注意：PostgreSQL bytea □型不能超□ 1GB。

`tabletype` □表□型的引用。

`data` □入 FlatGeobuf 数据。

可用性：3.2.0

7.9 几何□出

7.9.1 已知文本 (WKT)

7.9.1.1 ST_AsEWKT

ST_AsEWKT — 使用 SRID 元数据返回几何□形的已知文本 (WKT) 表示形式。

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);

描述

返回以 SRID □前□的几何□形的已知的文本表示形式（WKT)。可□的 *`maxdecimaldigits`* 参数可用于□少□出中使用的浮点后的最大十□制位数（默□□ 15）。

要□行 EWKT 表示形式到 PostGIS 几何□形的逆□□，□使用ST_GeomFromEWKT。

Warning
使用 *maxdecimaldigits* 参数可能会⊠致⊠出几何⊠形无效。⊠了避免⊠种情况，首先使用 ST_ReducePrecision 和合适的网格大小。

Note
WKT ⊠范不包括 SRID。要⊠取 OGC WKT 格式，⊠使用 ST_AsText。

Warning
WKT 格式不保持精度，因此⊠了防止浮⊠截断，⊠使用ST_AsBinary 或 ST_AsEWKB 格式⊠行⊠⊠。

增⊠：3.1.0 支持可⊠精度参数。

增⊠：2.0.0 引入了⊠地理、多面体曲面、三角形和 TIN 的支持。

⊠函数支持 3d 并且不会⊠失 z-index。

此方法支持⊠形字符串和曲⊠。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

示例

```
SELECT ST_AsEWKT('0103000020E6100000010000005000000000000000
      0000000000000000000000000000000000000000000000000000000000
      F03F000000000000F03F000000000000F03F000000000000F03F
      F000000000000000000000000000000000000000000000000000'::geometry);

       st_asewkt
---------------------------------
SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
(1 row)

SELECT ST_AsEWKT('0108000080030000000000000060 ↩
    E30A4100000000785C0241000000000000F03F0000000018
E20A4100000000485F024100000000000000400000000018
E20A4100000000305C0241000000000000840')

--st_asewkt---
CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)
```

相关信息

ST_AsBinary, ST_AsEWKB, ST_AsText, ST_GeomFromEWKT

7.9.1.2 ST_AsText

ST_AsText — 返回不⊠ SRID 元数据的几何/地理的已知文本 (WKT) 表示形式。

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);

描述

返回几何/地理的 OGC 已知文本 (WKT) 表示形式。可⊠的 *maxdecimaldigits* 参数可用于限制⊠出坐⊠中小数点后的位数（默⊠⊠ 15）。

要⊠行 WKT 表示形式到 PostGIS 几何⊠形的逆⊠⊠，⊠使用ST_GeomFromText。

Note
⊠准 OGC WKT 表示不包括 SRID。要将 SRID 作⊠⊠出表示的一部分，⊠使用非⊠准 PostGIS 函数ST_AsEWKT

Warning
WKT 中数字的文本表示可能无法保持完整的浮点精度。⊠了确保数据存⊠或⊠⊠的完全准确性，最好使用已知的二⊠制 (WKB) 格式（⊠参⊠ ST_AsBinary 和 *maxdecimaldigits*）。

Warning
使用 *maxdecimaldigits* 参数可能会⊠致⊠出几何⊠形无效。⊠了避免⊠种情况，首先使用 ST_ReducePrecision 和合适的网格大小。

可用性：1.5 - 引入了地理支持。

增⊠：2.5 - 引入了可⊠的精度参数。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.25

此方法支持⊠形字符串和曲⊠。

示例

```
SELECT ST_AsText('0103000000010000000500000000000000000000
0000000000000000000000000000000000000000000000000000000000
F03F000000000000F03F000000000000F03F000000000000F03F
F0000000000000000000000000000000000000000000000000');

    st_astext
--------------------------------
 POLYGON((0 0,0 1,1 1,1 0,0 0))
```

默⊠情况下⊠全精度⊠出。

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'));
    st_astext
------------------------------
 POINT(111.1111111 1.1111111)
```

maxdecimaldigits 参数用于限制⊠出精度。

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'), 2);
    st_astext
--------------------
 POINT(111.11 1.11)
```

相关信息

ST_AsBinary, ST_AsEWKB, ST_AsEWKT, ST_GeomFromText

7.9.2 已知的二⊠制文件 (WKB)

7.9.2.1 ST_AsBinary

ST_AsBinary — 返回不⊠ SRID 元数据的几何/地理的 OGC/ISO 已知的二⊠制 (WKB) 表示形式。

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);

描述

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.

WKB 格式⊠于从数据⊠⊠取几何数据并保持完整的数⊠精度非常有用。⊠避免了 WKT 等文本格式可能⊠生的精确舍入。

要⊠行 WKB 到 PostGIS 几何⊠形的逆⊠⊠，⊠使用ST_GeomFromWKB。

Note
OGC/ISO WKB 格式不包括 SRID。要⊠取包含 SRID 的 EWKB 格式，⊠使用ST_AsEWKB

Note
PostgreSQL 9.0 中的默⊠行⊠已更改⊠以十六⊠制⊠⊠⊠出 bytea。如果您的 GUI 工具需要旧行⊠，⊠在数据⊠中⊠置 SET bytea_output='escape'。

增⍰功能：引入了 2.0.0 ⍰多面体曲面、三角形和三角网的支持。

增⍰：2.0.0 支持更高坐⍰⍰度。

增⍰：2.0.0 支持地理中的字⍰序。

可用性：1.5.0 支持地理位置。

更改：2.0.0 此函数的⍰入不能是未知的——必⍰是几何⍰形。ST_AsBinary('POINT(1 2)') 等⍰⍰不再有效，您将收到 n st_asbinary(unknown) is not unique error。⍰似的代⍰需要更改⍰ ST_AsBinary('POINT(1 2)'::geometry);。如果不可能，⍰安装legacy.sql。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。 s2.1.1.1

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM 3: 5.1.37

此方法支持⍰形字符串和曲⍰。

⍰函数支持多面体曲面。

此函数支持三角形和不⍰⍰三角网面 (TIN)。

⍰函数支持 3d 并且不会⍰失 z-index。

示例

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

       st_asbinary
---------------------------------
\x0103000000010000000500000000000000000000000000000000000000000000000000f03f000000000000f03f000000000000f03f000000000000f03f0000000000000000000000
0000000000000000000000000
```

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326), 'XDR');
       st_asbinary
---------------------------------
\x00000000030000000100000005000000000000000000000000000000000000000000000000000003ff000
00000000003ff00000000000003ff00000000000003ff0000000000000000000000000000000000000
0000000000000000000000000
```

相关信息

ST_GeomFromWKB, ST_AsEWKB, ST_AsTWKB, ST_AsText,

7.9.2.2 ST_AsEWKB

ST_AsEWKB — 返回⍰有 SRID 元数据的几何⍰形的⍰展已知的二⍰制 (EWKB) 表示形式。

Synopsis

bytea **ST_AsEWKB**(geometry g1);
bytea **ST_AsEWKB**(geometry g1, text NDR_or_XDR);

描述

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.

WKB 格式☒于从数据☒☒取几何数据并保持完整的数☒精度非常有用。☒避免了 WKT 等文本格式可能☒生的精确舍入。

要☒行 EWKB 到 PostGIS 几何☒形的逆☒☒，☒使用 ST_GeomFromEWKB。

Note
要☒取 OGC/ISO WKB 格式，☒使用 ST_AsBinary。☒注意，OGC/ISO WKB 格式不包括 SRID。

增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

       st_asewkb
---------------------------------
\x0103000020e6100000010000000500000000000000000000000000000000000000000000000000000000000000000000
0000000000000f03f000000000000f03f000000000000f03f000000000000f03f000000000000000000000000000000000
00000000000000000000000000000000
```

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326), 'XDR');
       st_asewkb
---------------------------------
\x00200000003000010e60000000100000005000000000000000000000000000000000000000000000000000000000000
003ff0000000000003ff0000000000003ff0000000000003ff000000000000000000000000000000000
00000000000000000000000000000000
```

相关信息

ST_AsBinary, ST_GeomFromEWKB, ST_SRID

7.9.2.3 ST_AsHEXEWKB

ST_AsHEXEWKB — 使用小端 (NDR) 或大端 (XDR) ☒☒返回 HEXEWKB 格式（作☒文本）的几何☒形。

Synopsis

text **ST_AsHEXEWKB**(geometry g1, text NDRorXDR);
text **ST_AsHEXEWKB**(geometry g1);

描述

使用小端 (NDR) 或大端 (XDR) ☒☒返回 HEXEWKB 格式（作☒文本）的几何☒形。如果未指定☒☒，☒使用 NDR。

Note
可用性：1.2.2

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

示例

```
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
    --------
    0103000020E6100000010000000500
    00000000000000000000000000000000
    00000000000000000000000000000F03F
    000000000000F03F000000000000F03F000000000000F03
    F000000000000000000000000000000000000000000000000
```

7.9.3 其它格式

7.9.3.1 ST_AsEncodedPolyline

ST_AsEncodedPolyline — 从 LineString 几何体返回☒☒折☒。

Synopsis

text **ST_AsEncodedPolyline**(geometry geom, integer precision=5);

描述

以☒☒折☒形式返回几何☒形。此格式由精度☒ 5 的 Google 地☒和精度☒ 5 和 6 的开源路由机使用。

可☒ `precision` 指定☒☒折☒中将保留多少个小数位。☒☒和解☒☒的☒☒☒相同，否☒坐☒将不正确。

可用性：2.2.0

示例

基本

```
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`@
```

与地理⊠串和地理分段⊠合使用，并放在谷歌地⊠上

```
-- 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;
```

javascript 看起来像⊠⊠，其中 $ ⊠量替⊠⊠⊠⊠⊠果

```
<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>
```

相关信息

ST_LineFromEncodedPolyline, ST_Segmentize

7.9.3.2 ST_AsFlatGeobuf

ST_AsFlatGeobuf — 返回一⊠行的 FlatGeobuf 表示形式。

Synopsis

bytea **ST_AsFlatGeobuf**(anyelement set row);
bytea **ST_AsFlatGeobuf**(anyelement row, bool index);
bytea **ST_AsFlatGeobuf**(anyelement row, bool index, text geom_name);

描述

返回与 FeatureCollection ⊠⊠的一⊠行的 FlatGeobuf 表示形式 (http://flatgeobuf.org)。注意：PostgreSQL bytea 不能超⊠ 1GB。

`row` 至少包含一个几何列的行数据。

`index` 切⊠空⊠索引生成。默⊠⊠⊠ FALSE。

`geom_name` 行数据中几何列的列名。如果⊠ NULL，⊠它是找到的第一个几何列。

可用性：3.2.0

7.9.3.3 ST_AsGeobuf

ST_AsGeobuf — 返回一⊠行的 Geobuf 表示。

Synopsis

bytea **ST_AsGeobuf**(anyelement set row);
bytea **ST_AsGeobuf**(anyelement row, text geom_name);

描述

返回与 FeatureCollection ⊠⊠的一⊠行的 Geobuf 表示（https://github.com/mapbox/geobuf）。分析每个⊠入几何形状以确定最佳存⊠的最大精度。⊠注意，当前形式的 Geobuf 无法⊠行流式⊠⊠，因此完整的⊠出将在内存中⊠装。

`row` 至少包含一个几何列的行数据。

`geom_name` 行数据中几何列的列名。如果⊠ NULL，⊠它是找到的第一个几何列。

可用性：2.4.0

示例

```
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 — 以 GeoJSON 格式返回一个几何体或要素。

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);

描述

返回一个几何体作⊠ GeoJSON 中的“geometry”⊠象，或者返回一行作⊠ GeoJSON 中的“feature”⊠象。

生成的 GeoJSON 几何体和要素遵循 GeoJSON ⊠范 RFC 7946，但当解析的几何体使用 WGS84 ⊠度和⊠度之外的 CRS（如 EPSG:4326，urn:ogc:def:crs:OGC::CRS84）⊠行引用⊠，GeoJSON 几何⊠象将默⊠附加一个短的 CRS SRID ⊠⊠符。支持 2D 和 3D 几何。GeoJSON ⊠支持 SFS 1.1 几何⊠型（例如，不支持曲⊠）。

`geom_column` 参数用于区分多个几何列。如果省略⊠参数，将确定⊠⊠中的第一个几何列。相反，⊠⊠⊠参数将⊠省列⊠型⊠找。

`maxdecimaldigits` 参数可用于⊠少⊠出中使用的最大小数位数（默⊠⊠ 9）。如果您使用 EPSG:4326 并且⊠出几何⊠形⊠用于⊠示，⊠ `maxdecimaldigits=6` ⊠于⊠多地⊠来⊠可能是一个不⊠的⊠⊠。

Warning
使用 *maxdecimaldigits* 参数可能会☒致☒出几何☒形无效。☒了避免☒种情况，首先使用 ST_ReducePrecision 和合适的网格大小。

`options` 参数可用于在 GeoJSON ☒出中添加 BBOX 或 CRS：

- 0：表示没有☒☒
- 1：GeoJSON BBOX
- 2：GeoJSON 短 CRS（例如 EPSG：4326）
- 4：GeoJSON ☒ CRS（例如 urn:ogc:def:crs:EPSG::4326）
- 8：GeoJSON 短 CRS，如果不是 EPSG：4326（默☒）

`id_column` 参数用于☒置返回的 GeoJSON 要素的"id" 成☒。根据 GeoJSON RFC，☒☒在要素具有常用☒☒符（例如主☒）☒使用此参数。当未指定☒，生成的要素将不会有"id" 成☒，除了几何信息之外的任何列，包括任何可能的☒，都将最☒位于要素的"properties" 成☒内。

GeoJSON ☒范☒定多☒形使用右手定☒定向，并且某些客☒端需要此方向。☒可以通☒使用ST_ForcePolygonCCW 来确保。☒☒范☒要求几何☒形位于 WGS84 坐☒系 (SRID = 4326) 中。如果需要，可以使用 ST_Transform将几何☒形投影到 WGS84 中：`ST_Transform( geom, 4326 )`。

GeoJSON 可以在geojson.io 和 geojsonlint.com 上在☒☒☒和☒看。它受到网☒地☒框架的广泛支持：

- OpenLayers GeoJSON Example
- Leaflet GeoJSON Example
- Mapbox GL GeoJSON Example

可用性：1.3.4

可用性：1.5.0 支持地理位置。

更改：2.0.0 支持默☒参数和命名参数。

更改：3.0.0 支持☒☒作☒☒入

更改：3.0.0 ☒出 SRID（如果不是 EPSG：4326）。

更改：3.5.0 允☒指定包含要素 ID 的列

☒函数支持 3d 并且不会☒失 z-index。

示例

生成特征集合：

```
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”}}]}
```

生成一个特征：

```
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”}}
```

不要忘⊠将数据⊠⊠⊠ WGS84 ⊠度、⊠度以符合 GeoJSON ⊠范：

```
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]]]}
```

支持 3D 几何形状：

```
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]}
```

相关信息

ST_GeomFromGeoJSON, ST_ForcePolygonCCW , ST_Transform

7.9.3.5 ST_AsGML

ST_AsGML — 将几何⊠形作⊠ GML 版本 2 或 3 元素返回。

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);

描述

将几何⊠形作⊠地理⊠⊠⊠言 (GML) 元素返回。版本参数（如果指定）可以是 2 或 3。如果未指定版本参数，⊠默⊠⊠⊠ 2。`maxdecimaldigits` 参数可用于⊠少⊠出中使用的最大小数位数（默⊠⊠ 15））。

Warning
使用 *maxdecimaldigits* 参数可能会⊠致⊠出几何⊠形无效。⊠了避免⊠种情况，首先使用 ST_ReducePrecision 和合适的网格大小。

GML 2 指 2.1.2 版本，GML 3 指 3.1.1 版本

“⊠⊠”参数是一个位字段。它可用于在 GML ⊠出中定⊠ CRS ⊠出⊠型，并将数据声明⊠⊠度/⊠度：

- 0：GML 短 CRS（例如 EPSG:4326），默⊠⊠
- 1：GML ⊠ CRS（例如 urn:ogc:def:crs:EPSG::4326）
- 2：⊠⊠于 GML 3，从⊠出中⊠除 srsDimension 属性。
- 4：⊠⊠于 GML 3，⊠⊠使用 <LineString> 而不是 <Curve> ⊠⊠。
- 16：声明数据⊠⊠度/⊠度（例如 srid=4326）。默⊠情况下假⊠数据是平面的。此⊠⊠⊠适用于与⊠⊠序相关的 GML 3.1.1 ⊠出。因此，如果您⊠置它，它将交⊠坐⊠，因此⊠序是 lat lon 而不是数据⊠ lon lat。
- 32：⊠出几何体的⊠框（最小外接矩形）。

“命名空⊠前⊠”参数可用于指定自定⊠命名空⊠前⊠或无前⊠（如果⊠空）。如果使用 null 或省略’gml’ 前⊠

可用性：1.3.2

可用性：1.5.0 支持地理位置。

增⊠：2.0.0 引入了前⊠支持。引入了 GML3 的⊠⊠ 4，以允⊠使用 LineString 而不是⊠条的 Curve ⊠⊠。引入了⊠多面体曲面和 TINS 的 GML3 支持。引入⊠⊠ 32 来⊠出⊠框。

更改：2.0.0 使用默⊠命名参数

增⊠：⊠⊠ GML 3 引入了 2.1.0 id 支持。

Note
⊠ ST_AsGML 版本 3 + 支持多面体曲面和 TINS。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM IEC 13249-3: 17.2

⊠函数支持 3d 并且不会⊠失 z-index。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

示例：版本 **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>
```

示例：版本 **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>
```

相关信息

ST_GeomFromGML

7.9.3.6 ST_AsKML

ST_AsKML — 将几何⊠形作⊠ KML 元素返回。

Synopsis

text **ST_AsKML**(geometry geom, integer maxdecimaldigits=15, text nprefix=NULL);
text **ST_AsKML**(geography geog, integer maxdecimaldigits=15, text nprefix=NULL);

描述

将几何⊠形作⊠ Keyhole ⊠⊠⊠言 (KML) 元素返回。默⊠最大小数位数⊠ 15，默⊠命名空⊠无前⊠。

Warning
使用 *maxdecimaldigits* 参数可能会⊠致⊠出几何⊠形无效。⊠了避免⊠种情况，首先使用 ST_ReducePrecision 和合适的网格大小。

Note
需要在 Proj 支持下⊠⊠ PostGIS。使用 PostGIS_Full_Version确⊠您已⊠⊠⊠目支持。

Note
可用性：1.2.2 - 包含版本参数的更高版本出⊠在 1.3.2 中

Note
增⊠：2.0.0 - 添加前⊠命名空⊠，使用默⊠和命名参数

Note
更改：3.0.0 - ⊠除了“版本化”⊠体⊠名

Note
AsKML ⊠出不适用于没有 SRID 的几何⊠形

⊠函数支持 3d 并且不会⊠失 z-index。

示例

```
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>
```

相关信息

ST_AsSVG, ST_AsGML

7.9.3.7 ST_AsLatLonText

ST_AsLatLonText — 返回☒定点的度、分、秒表示形式。

Synopsis

text **ST_AsLatLonText**(geometry pt, text format='');

描述

返回点的度、分和秒表示形式。

Note
假☒☒点位于☒度/☒度投影中。X（☒度）和 Y（☒度）坐☒在☒出中☒准化☒“正常”范☒（☒度☒ -180 到 180，☒度☒ -90 到 90）。

text 参数是一个格式字符串，包含☒果文本的格式，☒似于日期格式字符串。有效☒☒☒“D”表示度、“M”表示分☒、“S”表示秒、“C”表示基本方向 (NSEW)。DMS 令牌可以重复以指示所需的☒度和精度（“SSS.SSSS”表示“1.0023”）。

“M”、“S”和“C”是可☒的。如果省略“C”，☒南或西的度数将☒示☒“-”符号。如果省略“S”，分☒将☒示☒十☒制，其精度位数与您指定的位数相同。如果☒省略“M”，☒度数将☒示☒十☒制，其精度与您指定的位数相同。

如果格式字符串被省略（或零☒度），☒将使用默☒格式。

可用性：2.0

示例

默⌧格式。

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)'));
      st_aslatlontext
---------------------------
 2°19'29.928”S 3°14'3.243”W
```

指定格式（与默⌧⌧相同）。

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D°M''S.SSS”C'));
      st_aslatlontext
---------------------------
 2°19'29.928”S 3°14'3.243”W
```

除 D、M、S 和 C 以外的字符⌧通⌧。

```
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
```

有符号的度数而不是基本方向。

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D°M''S.SSS”'));
      st_aslatlontext
---------------------------
 -2°19'29.928” -3°14'3.243”
```

十⌧制度数。

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D.DDDD degrees C'));
          st_aslatlontext
----------------------------------
 2.3250 degrees S 3.2342 degrees W
```

⌧大的⌧被⌧一化。

```
SELECT (ST_AsLatLonText('POINT (-302.2342342 -792.32498)'));
        st_aslatlontext
------------------------------
 72°19'29.928”S 57°45'56.757”E
```

7.9.3.8 ST_AsMARC21

ST_AsMARC21 — 将几何⌧形返回⌧⌧有地理数据字段 (034) 的 MARC21/XML ⌧⌧。

Synopsis

text **ST_AsMARC21** (geometry geom , text format=’hdddmmss’);

描述

此函数返回一个 MARC21/XML ⊠⊠，其中包含表示⊠定几何⊠形的⊠界框的⊠⊠制⊠数学数据。格式参数允⊠以 MARC21/XML ⊠准支持的所有格式⊠子字段 `$d`、`$e`、`$f` 和 `$g` 中的坐⊠⊠行⊠⊠。有效格式⊠：

- ⊠度和⊠度⊠⊠，度、分、秒（默⊠）：`hdddmmss`
- ⊠⊠度和⊠度⊠⊠的十⊠制度：`hddd.dddddd`
- 没有⊠度和⊠度⊠⊠的十⊠制度：`ddd.dddddd`
- ⊠有⊠度和⊠度⊠⊠的小数分⊠：`hdddmm.mmmm`
- 不⊠⊠度和⊠度⊠⊠的小数分⊠：`dddmm.mmmm`
- ⊠有⊠度和⊠度⊠⊠的小数秒：`hdddmmss.sss`

小数点符号也可以是逗号，例如 `hddddmm，mmmm`。

小数格式的精度可能受到小数点后字符数的限制，例如 `hdddmm.mm` 表示小数分⊠，精度⊠⊠位小数。

此函数忽略 Z 和 M ⊠度。

支持的 LOC MARC21/XML 版本：

- MARC21/XML 1.1

可用性：3.3.0

Note
此函数不支持非⊠度/⊠度几何形状，因⊠它⊠不受 MARC21/XML ⊠准（⊠⊠制⊠数学数据）支持。

Note
MARC21/XML ⊠准没有提供任何方法来注⊠⊠⊠制⊠数学数据的空⊠参考系⊠，⊠意味着⊠些信息在⊠⊠⊠ MARC21/XML 后将会⊠失。

示例

将 `POINT ⊠⊠⊠` MARC21/XML，格式⊠ dd mm ss（默⊠）

```
                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>
```

将 POLYGON ⊠⊠⊠十⊠制度格式的 MARC21/XML

```
                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>
```

将 GEOMETRYCOLLECTION ⊠⊠⊠以十⊠制分⊠格式表示的 MARC21/XML。MARC21/XML ⊠出中的几何⊠序与其在集合中的⊠序相⊠⊠。

```
                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>
```

相关信息

ST_GeomFromMARC21

7.9.3.9 ST_AsMVTGeom

ST_AsMVTGeom — 将几何⊠形⊠⊠⊠ MVT 瓦片的坐⊠空⊠。

Synopsis

geometry **ST_AsMVTGeom**(geometry geom, box2d bounds, integer extent=4096, integer buffer=256, boolean clip_geom=true);

描述

将几何⊠形⊠⊠⊠ MVT（Mapbox 矢量切片）瓦片的坐⊠空⊠，如果需要，将其剪切到⊠⊠⊠界。几何⊠形必⊠位于目⊠地⊠的坐⊠系中（如果需要，⊠使用 ST_Transform）。通常⊠是 Web 墨卡托 (SRID:3857)。

⊠函数⊠⊠保持几何有效性，并在需要⊠⊠行⊠正。⊠可能会⊠致⊠果几何体塌陷到⊠低的⊠度。

必⊠提供目⊠地⊠坐⊠空⊠中瓦片的矩形⊠界,以便可以⊠⊠几何⊠形,并在需要⊠⊠行裁剪。可以使用ST_TileEnvelope生成⊠界。

ST_AsMVT函数用于将几何⊠形⊠⊠⊠ ST_AsMVT 所需的瓦片坐⊠空⊠。

`geom` 是目⊠地⊠坐⊠系中要⊠⊠的几何⊠形。

`bounds` 是瓦片在地⊠坐⊠空⊠中的矩形⊠界，没有⊠冲区。

`extent` 是由 MVT ⊠范定⊠的切片坐⊠空⊠中切片的大小。默⊠⊠⊠ 4096。

`buffer` 是切片坐⊠空⊠中用于裁剪几何⊠形的⊠冲区大小。默⊠⊠⊠ 256。

`Clip_geom` 是一个布⊠⊠，用于控制几何⊠形是否按原⊠剪切或⊠⊠。默⊠⊠ true。

可用性：2.4.0

Note
从 3.0 开始，可以在配置⊠⊠⊠ Wagyu 来剪⊠和⊠⊠ MVT 多⊠形。⊠⊠比 GEOS 默⊠⊠速度更快，⊠生的⊠果更正确，但它可能会⊠失小多⊠形。

示例

```
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)
```

相关信息

ST_AsMVT, ST_TileEnvelope, PostGIS_Wagyu_Version

7.9.3.10 ST_AsMVT

ST_AsMVT — 返回一☒行的 MVT 表示形式的聚合函数。

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);

描述

一个聚合函数，返回与☒☒☒☒☒☒的一☒行的二☒制Mapbox 矢量瓦片表示。☒些行必☒包含一个几何列，☒几何列将被☒☒☒要素几何。几何☒形必☒位于☒☒坐☒空☒中，并且根据MVT ☒范有效。ST_AsMVTGeom 可用于将几何☒形☒☒☒☒☒坐☒空☒。其他行列被☒☒☒特征属性。

Mapbox 矢量切片格式可以存☒具有不同属性集的要素。要使用此功能，☒在包含一☒深度的 Json ☒象的行数据中提供一个 JSONB 列。JSONB ☒中的☒和☒将被☒☒☒功能属性。

可以通☒使用 `||` ☒接☒此函数的多个☒用来☒建具有多☒的切片。或使用 `STRING_AGG`。

Important
不要将 GEOMETRYCOLLECTION 作☒行中的元素☒行☒用。但是，您可以使用 ST_AsMVTGeom 准☒要包含的几何集合。

row 至少包含一个几何列的行数据。

name 是☒☒名称。默☒☒☒字符串“default”。

extent 是☒范定☒的屏幕空☒中的切片范☒。如果☒ NULL，☒默☒☒☒ 4096。

geom_name 是行数据中几何列的名称。默☒☒第一个几何列。☒注意，PostgreSQL 默☒情况下会自☒将未加引号的☒☒符折☒☒小写，☒意味着除非几何列被加引号，例如“MyMVTGeom”，此参数必☒以小写形式提供。

feature_id_name 是行数据中功能 ID 列的名称。如果☒ NULL 或☒数，☒不☒置功能 ID。匹配名称和有效☒型（smallint、integer、bigint）的第一列将用作功能 ID，任何后☒列将添加☒属性。不支持 JSON 属性。

增☒：3.0 - 添加了☒要素 ID 的支持。

增☒：2.5.0 - 添加了☒并行☒☒的支持。

可用性：2.4.0

示例

```
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;
```

相关信息

ST_AsMVTGeom, ST_TileEnvelope

7.9.3.11 ST_AsSVG

ST_AsSVG — 返回几何体的 SVG 路径数据。

Synopsis

text **ST_AsSVG**(geometry geom, integer rel=0, integer maxdecimaldigits=15);
text **ST_AsSVG**(geography geog, integer rel=0, integer maxdecimaldigits=15);

描述

将几何☒形作☒☒量矢量☒形 (SVG) 路径数据返回。使用 1 作☒第二个参数可以根据相☒移☒☒☒路径数据，默☒☒（或 0）使用☒☒移☒。第三个参数可用于☒少☒出中使用的最大十☒制位数（默☒☒ 15）。当’rel’ arg ☒ 0 ☒，点几何☒形将渲染☒ cx/cy，当’rel’ arg ☒ 1 ☒，将渲染☒ x/y。多点几何☒形由逗号 (”,”) 分隔，GeometryCollection 几何☒形由分号 (”;”) 分隔。

要使用 PostGIS SVG ☒形，☒☒看pg_svg ☒，它提供了用于☒理 ST_AsSVG ☒出的 plpgsql 函数。

增☒：3.4.0 支持所有曲☒☒型

更改：2.0.0 - 添加了☒默☒参数和命名参数的支持

Note
可用性：1.2.2。可用性：1.4.0 PostGIS 1.4.0 中更改☒在☒☒路径中包含 L 命令以符合http://www.w3.org/TR/SVG/paths.html#PathDataBNF

此方法支持☒形字符串和曲☒。

示例

```
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
```

☒弧

```
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
```

多曲☒

```
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
```

多曲面

```
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 — 返回几何形式☒ TWKB，又名“微小的已知的二☒制”

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);

描述

返回 TWKB（微小已知的二⍰制）格式的几何⍰形。TWKB 是一种⍰⍰的二⍰制格式，注重最小化⍰出的大小。

小数位参数控制⍰出中存⍰的精度。默⍰情况下，⍰在⍰⍰前四舍五入到最接近的⍰位。如果要⍰⍰更高的精度，⍰增加数字。例如，⍰ 1 表示将保留小数点右⍰的第一位数字。

大小和⍰界框参数控制⍰出中是否包含有关⍰象的⍰⍰⍰度和⍰象⍰界的可⍰信息。默⍰情况下，它⍰不是。除非您的客⍰端⍰件需要使用它⍰，否⍰不要打开它⍰，因⍰它⍰只会占用空⍰（⍰省空⍰是 TWKB 的目的）。

⍰函数的数⍰⍰入形式用于将几何⍰形和唯一⍰⍰符的集合⍰⍰⍰保留⍰⍰符的 TWKB 集合。⍰⍰于希望解⍰集合然后⍰⍰有关内部⍰象的更多信息的客⍰端非常有用。您可以使用 array_agg 函数⍰建数⍰。其他参数的操作与函数的⍰⍰形式相同。

Note
格式⍰范可在 https://github.com/TWKB/Specification 在⍰⍰取，⍰建 JavaScript 客⍰端的代⍰可在https://github.com/TWKB/twkb.js 找到。

增⍰：2.4.0 内存和速度改⍰。

可用性：2.2.0

示例

```
SELECT ST_AsTWKB('LINESTRING(1 1,5 5)'::geometry);
                 st_astwkb
-------------------------------------------
\x02000202020808
```

要⍰建包含⍰⍰符的聚合 TWKB ⍰象，⍰首先使用“array_agg()”聚合所需的几何⍰形和⍰象，然后⍰用适当的 TWKB 函数。

```
SELECT ST_AsTWKB(array_agg(geom), array_agg(gid)) FROM mytable;
                 st_astwkb
-------------------------------------------
\x040402020400000202
```

相关信息

ST_GeomFromTWKB, ST_AsBinary, ST_AsEWKB, ST_AsEWKT, ST_GeomFromText

7.9.3.13 ST_AsX3D

ST_AsX3D — 返回 X3D xml ⍰点元素格式的几何⍰形：ISO-IEC-19776-1.2-X3DEncodings-XML

Synopsis

text **ST_AsX3D**(geometry g1, integer maxdecimaldigits=15, integer options=0);

描述

以 X3D xml 格式的⊠点元素形式返回几何⊠形 http://www.web3d.org/standards/number/19776-1。如果未指定 maxdecimaldigits（精度），⊠默⊠⊠ 15。

Note

有多种⊠⊠可用于将 PostGIS 几何⊠形⊠⊠⊠ X3D，因⊠ X3D 几何⊠型不会直接映射到 PostGIS 几何⊠型，而一些⊠新的 X3D ⊠型可能是我⊠避免的更好的映射，因⊠大多数渲染工具当前不支持它⊠。⊠些是我⊠已⊠确定的映射。如果您⊠我⊠允⊠人⊠表示他⊠首⊠映射的想法或方式有任何想法，⊠随⊠⊠布⊠⊠通知⊠。

以下是我⊠目前如何将 PostGIS 2D/3D ⊠型映射到 X3D ⊠型

“⊠⊠”参数是一个位字段。⊠于 PostGIS 2.2 ，⊠用于表示是否用 X3D GeoCooperatives 地理空⊠⊠点表示坐⊠以及是否翻⊠ x/y ⊠。默⊠情况下，ST_AsX3D 以数据⊠形式⊠出（long、lat 或 X、Y），但 X3D 默⊠⊠ lat/lon、y/x 可能是首⊠。

- 0：数据⊠⊠序中的 X/Y（例如，⊠/⊠度 = X,Y 是⊠准数据⊠⊠序）、默⊠⊠和非空⊠坐⊠（只是常⊠的旧坐⊠⊠⊠）。
- 1：翻⊠ X 和 Y。如果与地理坐⊠⊠⊠开关⊠合使用，⊠⊠出将默⊠⊠ “latitude_first”，并且坐⊠也会翻⊠。
- 2：GeoSpatial GeoCoordinates 中的⊠出坐⊠。如果几何⊠形不在 WGS 84 ⊠⊠度 (srid: 4326) 中，此⊠⊠将引⊠⊠⊠。⊠是目前唯一支持的地理坐⊠⊠型。⊠参⊠指定空⊠参考系⊠的 X3D ⊠范。默⊠⊠出将⊠ GeoCoordinate geoSystem=’”GD” ”WE” ”longitude_first”’。如果您更喜⊠ GeoCoordinate geoSystem=’”GD” ”WE” ”latitude_first”’ 的 X3D 默⊠⊠，⊠使用 (2+1) = 3

PostGIS ⊠型	**2D X3D** ⊠型	**3D X3D** ⊠型
LINESTRING	尚未⊠⊠ - 将是 PolyLine2D	LineSet
MULTILINESTRING	尚未⊠⊠ - 将是 PolyLine2D	IndexedLineSet
MULTIPOINT	Polypoint2D	PointSet
POINT	⊠出以空格分隔的坐⊠	⊠出以空格分隔的坐⊠
(MULTI) POLYGON, POLYHEDRALSURFACE	X3D ⊠⊠无效	IndexedFaceSet（内⊠当前⊠出⊠⊠一个面集）
TIN	TriangleSet2D（尚未⊠⊠）	IndexedTriangleSet（索引三角形集）

Note

2D 几何支持尚未完成。内⊠当前⊠⊠制⊠⊠独的多⊠形。我⊠正在研究⊠些。

3D 空⊠中⊠生了⊠多⊠步，特⊠是X3D 与 HTML5 的集成

⊠有一个很好的开源 X3D ⊠看器,您可以使用它来⊠看渲染的几何⊠形。Free Wrlhttp://freewrl.sourceforge.net/-二⊠制文件适用于 Mac、Linux 和 Windows。使用打包的 FreeWRL_Launcher 来⊠看几何⊠形。

⊠⊠⊠看利用此功能的PostGIS ⊠⊠ X3D ⊠看器和x3dDom html/js 开源工具包。

可用性：2.0.0：ISO-IEC-19776-1.2-X3DEncodings-XML

增⊠：2.2.0：添加了⊠反⊠地理坐⊠和⊠（x/y、⊠度/⊠度）的支持。有关⊠⊠信息，⊠参⊠⊠⊠。

⊠函数支持 3d 并且不会⊠失 z-index。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

示例：⊠建一个功能⊠全的 **X3D** 文档 **-** 您正在生成一个可以在 **FreeWrl** 或其他 **X3D** ⊠看器中⊠看的多⊠数据集。

```
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>
```

PostGIS 建筑

将此⊠⊠的⊠出复制并粘⊠到x3d ⊠景⊠看器，然后⊠⊠ “⊠示”

```
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);
```

PostGIS 区分、区区形成的建筑物

示例：高度区 **3** 个区位、精度区 **6** 位的八角棱区

```
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>
```

示例：**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>
```

示例：区合多区串（区孔的多区形的区界）

```
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 — 返回几何⍰形的 GeoHash 表示形式。

Synopsis

text **ST_GeoHash**(geometry geom, integer maxchars=full_precision_of_point);

描述

⍰算几何⍰形的 GeoHash 表示。GeoHash 将地理点⍰⍰⍰可基于前⍰排序和搜索的文本形式。⍰短的 GeoHash 是点的不太精确的表示。它可以被⍰⍰是一个包含点的盒子。

具有非零范⍰的非点几何⍰也可以映射到 GeoHash 代⍰。代⍰的精度取决于几何的地理范⍰。

如果未指定 maxchars，⍰返回的 GeoHash 代⍰适用于包含⍰入几何⍰形的最小⍰元格。Points 返回精度⍰ 20 个字符的 GeoHash（大⍰足以容⍰⍰入的完整双精度）。其他几何⍰型可能会返回精度⍰低的 GeoHash，具体取决于几何形状的范⍰。⍰大的几何⍰形的精度⍰低，⍰小的几何⍰形的精度⍰高。由 GeoHash 代⍰确定的框始⍰包含⍰入特征。

如果指定了 maxchars 参数，⍰返回的 GeoHash 代⍰最多包含指定数量的字符。它映射到⍰入几何体的（可能）⍰低精度表示。⍰于非点几何体，⍰算的起始点是几何体⍰界框的中心点。

可用性：1.4.0

Note
ST_GeoHash 要求⍰入几何⍰形采用地理（⍰度/⍰度）坐⍰。

此方法支持⍰形字符串和曲⍰。

示例

```
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
```

相关信息

ST_GeomFromGeoHash, ST_PointFromGeoHash, ST_Box2dFromGeoHash

7.10 ☒算符

7.10.1 ☒界框☒算符

7.10.1.1 &&

&& — 如果 A 的 2D ☒界框与 B 的 2D ☒界框相交，☒返回 TRUE。

Synopsis

boolean **&&**(geometry A , geometry B);
boolean **&&**(geography A , geography B);

描述

如果 A 的二☒☒界框与 B 的二☒☒界框交互，&& 返回 TRUE。

Note
☒操作符将利用几何上可能可用的任何索引。

增☒：2.0.0 引入了☒多面体曲面的支持。

可用性：1.5.0 引入了☒地理的支持。

此方法支持☒形字符串和曲☒。

☒函数支持多面体曲面。

示例

```
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)
```

相关信息

ST_Intersects, ST_Extent, |&>, &>, &<|, &<, ~, @

7.10.1.2 &&(geometry,box2df)

&&(geometry,box2df) — 如果几何体的（⍰存的）2D ⍰界框与 2D 浮点精度⍰界框 (BOX2DF) 相交，⍰返回 TRUE。

Synopsis

boolean **&&**(geometry A , box2df B);

描述

如果几何体 A 的⍰存 2D ⍰界框与 2D ⍰界框 B 相交（使用浮点精度），⍰ && ⍰算符返回 TRUE。⍰意味着如果 B 是（双精度）box2d，它将在内部⍰⍰⍰浮点精度 2D ⍰界框（BOX2DF）

Note
⍰操作数旨在由 BRIN 索引内部使用，而不是由用⍰使用。

可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持⍰形字符串和曲⍰。

⍰函数支持多面体曲面。

示例

```
SELECT ST_Point(1,1) && ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) AS overlaps;

 overlaps
----------
 t
(1 row)
```

相关信息

&&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.3 &&(box2df,geometry)

&&(box2df,geometry) — 如果 2D 浮点精度□界框 (BOX2DF) 与几何体的（□存的）2D □界框相交，□返回 TRUE。

Synopsis

boolean **&&**(box2df A , geometry B);

描述

□ && □算符在使用浮点精度□，如果 2D □界框 A 与几何体 B 的□存 2D □界框相交，□返回 TRUE。□意味着如果 A 是一个（双精度）box2d，它将在内部被□□□一个浮点精度的 2D □界框（BOX2DF）

Note
□操作数旨在由 BRIN 索引内部使用，而不是由用□使用。

可用性：2.3.0 引入了□□范□索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持□形字符串和曲□。

□函数支持多面体曲面。

示例

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_Point(1,1) AS overlaps;

 overlaps
----------
 t
(1 row)
```

相关信息

&&(geometry,box2df), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.4 &&(box2df,box2df)

&&(box2df,box2df) — 如果□个 2D 浮点精度□界框 (BOX2DF) 彼此相交，□返回 TRUE。

Synopsis

boolean **&&**(box2df A , box2df B);

描述

&& 运算符在使用浮点精度时，如果两个 2D 边界框 A 和 B 相交，则返回 TRUE。这意味着如果 A（或 B）是一个（双精度）box2d，它将在内部被转换为一个浮点精度的 2D 边界框（BOX2DF）

Note
索引运算符旨在供 BRIN 索引内部使用，而不是供用户使用。

可用性：2.3.0 引入了对范围索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持圆形字符串和曲线。

函数支持多面体曲面。

示例

```
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)
```

相关信息

&&(geometry,box2df), &&(box2df,geometry), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.5 &&&

&&& — 如果 A 的 n 维边界框与 B 的 n 维边界框相交，则返回 TRUE。

Synopsis

boolean **&&&**(geometry A , geometry B);

描述

&&& 运算符返回 TRUE，如果 A 的 n 维边界框与 B 的 n 维边界框相交。

Note
该操作符将利用几何上可能可用的任何索引。

可用性: 2.0.0

此方法支持☒形字符串和曲☒。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

☒函数支持 3d 并且不会☒失 z-index。

示例：**3D** ☒串

```
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
```

示例：**XYM** ☒串

```
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
```

相关信息

&&

7.10.1.6 &&&(geometry,gidx)

&&&(geometry,gidx) — 如果几何体的（☒存的）n ☒☒界框与 n ☒浮点精度☒界框 (GIDX) 相交，☒返回 TRUE。

Synopsis

boolean **&&&**(geometry A , gidx B);

描述

&&& ☒算符在使用浮点精度☒，如果几何体 A 的☒存 n-D ☒界框与 n-D ☒界框 B 相交，☒返回 TRUE。☒意味着如果 B 是一个（双精度）box3d，它将在内部被☒☒☒一个浮点精度的 3D ☒界框（GIDX）

Note
☒☒算符旨在供 BRIN 索引内部使用，而不是供用☒使用。

可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持☒形字符串和曲☒。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

☒函数支持 3d 并且不会☒失 z-index。

示例

```
SELECT ST_MakePoint(1,1,1) &&& ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) AS  ↩
    overlaps;

 overlaps
----------
 t
(1 row)
```

相关信息

&&&(gidx,geometry), &&&(gidx,gidx)

7.10.1.7 &&&(gidx,geometry)

&&&(gidx,geometry) — 如果 n ☒浮点精度☒界框 (GIDX) 与几何体的（☒存的）n ☒☒界框相交，☒返回 TRUE。

Synopsis

boolean **&&&**(gidx A , geometry B);

描述

&&& ☒算符在使用浮点精度☒，如果 n-D ☒界框 A 与几何体 B 的☒存 n-D ☒界框相交，☒返回 TRUE。☒意味着如果 A 是一个（双精度）box3d，它将在内部被☒☒☒一个浮点精度的 3D ☒界框（GIDX）

Note
☒☒算符旨在供 BRIN 索引内部使用，而不是供用☒使用。

可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持⍰形字符串和曲⍰。

⍰函数支持多面体曲面。

此函数支持三角形和不⍰⍰三角网面 (TIN)。

⍰函数支持 3d 并且不会⍰失 z-index。

示例

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_MakePoint(1,1,1) AS  ↩
    overlaps;

 overlaps
----------
 t
(1 row)
```

相关信息

&&&(geometry,gidx), &&&(gidx,gidx)

7.10.1.8 &&&(gidx,gidx)

&&&(gidx,gidx) — 如果⍰个 n ⍰浮点精度⍰界框 (GIDX) 彼此相交，⍰返回 TRUE。

Synopsis

boolean **&&&**(gidx A , gidx B);

描述

&&& ⍰算符在使用浮点精度⍰，如果⍰个 n-D ⍰界框 A 和 B 相交，⍰返回 TRUE。⍰意味着如果 A（或 B）是一个（双精度）box3d，它将在内部被⍰⍰⍰一个浮点精度的 3D ⍰界框（GIDX）

> **Note**
> ⍰⍰算符旨在供 BRIN 索引内部使用，而不是供用⍰使用。

可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持⍰形字符串和曲⍰。

⍰函数支持多面体曲面。

此函数支持三角形和不⍰⍰三角网面 (TIN)。

⍰函数支持 3d 并且不会⍰失 z-index。

示例

```
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)
```

相关信息

&&&(geometry,gidx), &&&(gidx,geometry)

7.10.1.9 &<

&< — 如果 A 的⌧界框与 B 的⌧界框重⌧或位于其左⌧，⌧返回 TRUE。

Synopsis

boolean **&<**(geometry A , geometry B);

描述

&< ⌧算符返回 TRUE 如果几何体 A 的⌧界框与几何体 B 的⌧界框重⌧或在几何体 B 的⌧界框的左⌧，或者更准确地⌧，不位于几何体 B 的⌧界框的右⌧。

Note
⌧操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

&&, |&>, &>, &<|

7.10.1.10 &<|

&<| — 如果 A 的⍰界框与 B 的⍰界框重⍰或低于 B 的⍰界框，⍰返回 TRUE。

Synopsis

boolean **&<|**(geometry A , geometry B);

描述

&<| ⍰算符返回 TRUE，如果几何体 A 的⍰界框与几何体 B 的⍰界框重⍰或低于几何体 B 的⍰界框，或者更准确地⍰，不位于几何体 B 的⍰界框的上方。

此方法支持⍰形字符串和曲⍰。

⍰函数支持多面体曲面。

Note
⍰操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

&&, |&>, &>, &<

7.10.1.11 &>

&> — 如果 A 的⍰界框与 B 的⍰界框重⍰或位于 B 右⍰，⍰返回 TRUE。

Synopsis

boolean **&>**(geometry A , geometry B);

描述

☒算符 &> 返回 TRUE，如果几何体 A 的☒界框与几何体 B 的☒界框重☒或位于其右☒，更准确地☒，它与几何体 B 的☒界框不重☒于其左☒。

Note
☒操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

&&, |&>, &<|, &<

7.10.1.12 <<

<< — 如果 A 的☒界框☒格位于 B 的左☒，☒返回 TRUE。

Synopsis

boolean **<<**(geometry A , geometry B);

描述

如果 A 的☒界框☒格位于 B 的☒界框的左☒，☒ << ☒算符返回 TRUE。

Note
☒操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

>>, |>>, <<|

7.10.1.13 <<|

<<| — 如果 A 的⊠界框⊠格低于 B 的⊠界框，⊠返回 TRUE。

Synopsis

boolean **<<|**(geometry A , geometry B);

描述

如果 A 的⊠界框⊠格低于 B 的⊠界框，⊠⊠算符 <<| 返回 TRUE。

Note
⊠操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

<<, >>, |>>

7.10.1.14 =

= — 如果几何/地理 A 的坐⊠和坐⊠⊠序与几何/地理 B 的坐⊠和坐⊠⊠序相同，⊠返回 TRUE。

Synopsis

boolean =(geometry A , geometry B);
boolean =(geography A , geography B);

描述

如果几何/地理 A 的坐⊠和坐⊠⊠序与几何/地理 B 的坐⊠和坐⊠⊠序相同，= ⊠算符返回 TRUE。PostgreSQL 使用⊠几何定⊠的 =、< 和 > ⊠算符来⊠行内部排序和比⊠几何⊠形（即在 GROUP BY 或 ORDER BY 子句中）。

Note
只有在所有方面都完全相同、具有相同坐⊠、相同⊠序的几何/地理才被⊠⊠算符⊠⊠相等。⊠于“空⊠平等”，它忽略坐⊠⊠序等内容，并且可以⊠⊠以不同表示覆盖相同空⊠区域的要素，⊠使用ST_OrderingEquals 或 ST_Equals

Caution
此⊠算符不⊠几何⊠形使用任何有效索引。要使用索引⊠行⊠健的相等性⊠⊠，⊠使用 = 一和 &&。

更改：2.4.0，在之前的版本中，⊠是⊠界框相等而不是几何相等。如果需要⊠界框相等，⊠使用~=代替。

此方法支持⊠形字符串和曲⊠。

⊠函数支持多面体曲面。

示例

```
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
```

相关信息

ST_Equals, ST_OrderingEquals, ~=

7.10.1.15 >>

>> — 如果 A 的⌧界框⌧格位于 B 的右⌧，⌧返回 TRUE。

Synopsis

boolean **>>**(geometry A , geometry B);

描述

如果几何体 A 的⌧界框⌧格位于几何体 B 的⌧界框右⌧，⌧ >> ⌧算符返回 TRUE。

Note
⌧操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

<<, |>>, <<|

7.10.1.16 @

@ — 如果 A 的⊠界框包含在 B 的⊠界框中，⊠返回 TRUE。

Synopsis

boolean @(geometry A , geometry B);

描述

如果几何体 A 的⊠界框完全包含在几何体 B 的⊠界框内，⊠ @ ⊠算符返回 TRUE。

Note
⊠操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

~, &&

7.10.1.17 @(geometry,box2df)

@(geometry,box2df) — 如果几何体的 2D 界框包含在 2D 浮点精度界框 (BOX2DF) 中，返回 TRUE。

Synopsis

boolean @(geometry A , box2df B);

描述

如果 A 几何形的 2D 界框包含 2D 界框 B（使用浮点精度），@ 算符返回 TRUE。意味着如果 B 是（双精度）box2d，它将在内部浮点精度 2D 界框（BOX2DF）

Note
操作数旨在由 BRIN 索引内部使用，而不是由用使用。

可用性：2.3.0 引入了范索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持形字符串和曲。

函数支持多面体曲面。

示例

```
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)
```

相关信息

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry)
~(box2df,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.18 @(box2df,geometry)

@(box2df,geometry) — 如果 2D 浮点精度界框 (BOX2DF) 包含在几何体的 2D 界框中，返回 TRUE。

Synopsis

boolean @(box2df A , geometry B);

描述

如果 2D 界框 A 使用浮点精度包含在 B 几何体的 2D 界框中，则 @ 运算符返回 TRUE。这意味着如果 B 是（双精度）box2d，它将在内部被转换为浮点精度 2D 界框（BOX2DF）

Note
此操作数旨在由 BRIN 索引内部使用，而不是由用户使用。

可用性：2.3.0 引入了对空间索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持圆形字符串和曲线。

函数支持多面体曲面。

示例

```
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)
```

相关信息

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,box2df)

7.10.1.19 @(box2df,box2df)

@(box2df,box2df) — 如果 2D 浮点精度界框 (BOX2DF) 包含在另一个 2D 浮点精度界框内，则返回 TRUE。

Synopsis

boolean @(box2df A , box2df B);

描述

如果 2D 界框 A 包含在 2D 界框 B 中，则 @ 运算符使用浮点精度返回 TRUE。这意味着如果 A（或 B）是（双精度）box2d，它将在内部被转换为浮点精度 2D 界框（BOX2DF）

Note
此操作数旨在由 BRIN 索引内部使用，而不是由用户使用。

可用性：2.3.0 引入了对空间索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持圆形字符串和曲线。

函数支持多面体曲面。

示例

```
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)
```

相关信息

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry) ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry)

7.10.1.20 |&>

|&> — 如果 A 的⌧界框与 B 的⌧界框重⌧或位于其上方，⌧返回 TRUE。

Synopsis

boolean **|&>**(geometry A , geometry B);

描述

⌧算符 |&> 返回 TRUE，如果几何体 A 的⌧界框与几何体 B 的⌧界框重⌧或位于其上方，更准确地⌧，它与几何体 B 的⌧界框不重⌧于其下方。

Note
⌧操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

&&, &>, &<|, &<

7.10.1.21 |>>

|>> — 如果 A 的⊠界框⊠格位于 B 的⊠界框上方，⊠返回 TRUE。

Synopsis

boolean **|>>**(geometry A , geometry B);

描述

如果几何体 A 的⊠界框⊠格位于几何体 B 的⊠界框上方，⊠ |>> ⊠算符返回 TRUE。

Note
⊠操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

<<, >>, <<|

7.10.1.22 ~

~ — 如果 A 的⊠界框包含 B 的⊠界框，⊠返回 TRUE。

Synopsis

boolean **~**(geometry A , geometry B);

描述

如果几何体 A 的⊠界框完全包含几何体 B 的⊠界框，⊠ ~ ⊠算符返回 TRUE。

Note
⊠操作符将利用几何上可能可用的任何索引。

示例

```
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)
```

相关信息

@, &&

7.10.1.23 ~(geometry,box2df)

~(geometry,box2df) — 如果几何体的 2D 粘合框包含 2D 浮点精度⊠界框 (GIDX)，⊠返回 TRUE。

Synopsis

boolean ~(geometry A , box2df B);

描述

如果几何⊠形 A 的 2D ⊠界框包含 2D ⊠界框 B（使用浮点精度），⊠ ~ ⊠算符返回 TRUE。⊠意味着如果 B 是（双精度）box2d，它将在内部⊠⊠⊠浮点精度 2D ⊠界框（BOX2DF）

Note
⊠操作数旨在由 BRIN 索引内部使用，而不是由用⊠使用。

可用性：2.3.0 引入了⊠⊠范⊠索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持⊠形字符串和曲⊠。

⊠函数支持多面体曲面。

示例

```
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)
```

相关信息

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.24 ~(box2df,geometry)

~(box2df,geometry) — 如果 2D 浮点精度□界框 (BOX2DF) 包含几何体的 2D □界框，□返回 TRUE。

Synopsis

boolean ~(box2df A , geometry B);

描述

如果 2D □界框 A 包含 B 几何体的□界框，□ ~ □算符使用浮点精度返回 TRUE。□意味着如果 A 是（双精度）box2d，它将在内部□□□浮点精度 2D □界框（BOX2DF）

Note
□操作数旨在由 BRIN 索引内部使用，而不是由用□使用。

可用性：2.3.0 引入了□□范□索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持□形字符串和曲□。

□函数支持多面体曲面。

示例

```
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)
```

相关信息

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.25 ~(box2df,box2df)

~(box2df,box2df) — 如果 2D 浮点精度⍰界框 (BOX2DF) 包含⍰一个 2D 浮点精度⍰界框 (BOX2DF)，⍰返回 TRUE。

Synopsis

boolean ~(box2df A , box2df B);

描述

如果 2D 浮点精度⍰界框 A 包含 2D 浮点精度⍰界框 B，⍰ ~ ⍰算符返回 TRUE。⍰意味着如果 A 是（双精度）box2d，它将在内部⍰⍰⍰浮点精度 2D ⍰界框（BOX2DF）

Note
⍰操作数旨在由 BRIN 索引内部使用，而不是由用⍰使用。

可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。

此方法支持⍰形字符串和曲⍰。

⍰函数支持多面体曲面。

示例

```
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)
```

相关信息

&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)

7.10.1.26 ~=

~= — 如果 A 的⍰界框与 B 的⍰界框相同，⍰返回 TRUE。

Synopsis

boolean ~=(geometry A , geometry B);

描述

如果几何/地理 A 的☒界框与几何/地理 B 的☒界框相同，☒ ~= ☒算符返回 TRUE。

Note
☒操作符将利用几何上可能可用的任何索引。

可用性：1.5.0 改☒了行☒

☒函数支持多面体曲面。

Warning
☒☒算符已将 PostGIS 1.5 中的行☒从☒☒☒☒几何相等性更改☒☒☒☒☒界框相等性。☒事情☒得复☒的是，它☒取决于您是否☒行了硬升☒或☒升☒，数据☒的行☒是什么。要了解您的数据☒有☒些行☒，您可以☒行下面的☒☒。要☒☒是否真正相等，☒使用 ST_OrderingEquals或 ST_Equals。

示例

```
select 'LINESTRING(0 0, 1 1)'::geometry ~= 'LINESTRING(0 1, 1 0)'::geometry as equality;
 equality   |
------------------+
          t    |
```

相关信息

ST_Equals, ST_OrderingEquals, =

7.10.2 距离☒算符

7.10.2.1 <->

<-> — 返回 A 和 B 之☒的 2D 距离。

Synopsis

double precision **<->**(geometry A , geometry B);
double precision **<->**(geography A , geography B);

描述

<-> 算符返回个几何形之的二距离。用于“ORDER BY”子句提供索引助的最近果集。于 9.5 以下的 PostgreSQL 出界框的心距离，而于 PostgreSQL 9.5，真正的 KNN 距离搜索出几何形之的真距离以及地理的球体距离。

Note
操作符将利用几何上可能可用的 2D GiST 索引。它与其他使用空索引的算符不同，只有当算符位于 ORDER BY 子句中才会使用空索引。

Note
当其中一个几何形是常量（不在子/cte 中），索引才会。例如’SRID=3005;POINT(1011102 450541)’::geometry 而不是 a.geom

有关示例，参PostGIS 研会：最近搜索。

增：2.2.0 -几何和地理之的 KNN（k 最近）行在是真的。注意，地理的 KNN 是在球面上算的，而不是在球体平面上算的。PostgreSQL 9.4 及更低版本支持地理，但支持界框的重心。

更改：2.2.0 -在 PostgreSQL 9.5 中，旧的混合格式可能很慢。因此，如果您只想在 PostGIS 2.2 或更高版本和 PostgreSQL 9.5 或更高版本上行，可能需要消除方法。

可用性：2.0.0——弱 KNN 根据几何心距离而不是真距离提供最近。点的果精确，所有其他型的果不精确。适用于 PostgreSQL 9.1+

示例

```
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)
```

KNN 的原始答案是：

```
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)
```

如果您⊠⊠⊠个⊠⊠⊠行“EXPLAIN ANALYZE”，您将看到第二个⊠⊠的性能有所提高。

⊠于⊠行 PostgreSQL < 9.5 的用⊠，⊠使用混合⊠⊠来⊠找真正的最⊠近点。首先使用索引⊠助 KNN ⊠行 CTE ⊠⊠，然后⊠行精确⊠⊠以⊠得正确的排序：

```
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)
```

相关信息

ST_DWithin, ST_Distance, <#>

7.10.2.2 |=|

|=| — 返回 A 和 B ⊠迹在最接近点⊠的距离。

Synopsis

double precision **|=|**(geometry A , geometry B);

描述

|=| ⊠算符返回⊠个⊠迹之⊠的 3D 距离（⊠参⊠ ST_IsValidTrajectory）。⊠与 ST_DistanceCPA 相同，但作⊠⊠算符，它可用于使用 N ⊠索引⊠行最近⊠搜索（需要 PostgreSQL 9.5.0 或更高版本）。

Note
☒操作数将利用几何上可能可用的 ND GiST 索引。它与其他使用空☒索引的☒算符不同，只有当☒☒算符位于 ORDER BY 子句中☒才会使用空☒索引。

Note
☒当其中一个几何☒形是常量（不在子☒☒/cte 中）☒，索引才会☒☒。例如'SRID=3005;LINESTRINGM(0 0 0,0 0 1)'::geometry 而不是 a.geom

可用性：2.2.0。索引支持☒适用于 PostgreSQL 9.5+

示例

```
-- 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)
```

相关信息

ST_DistanceCPA, ST_ClosestPointOfApproach, ST_IsValidTrajectory

7.10.2.3 <#>

<#> — 返回 A 和 B ☒界框之☒的 2D 距离。

Synopsis

double precision **<#>**(geometry A , geometry B);

描述

<#> ☒算符返回☒个浮点☒界框之☒的距离，可能从空☒索引中☒取它☒（需要 PostgreSQL 9.1）。☒于☒行最近☒近似距离排序很有用。

Note
⊠操作符将利用几何上可能可用的任何索引。它与其他使用空⊠索引的⊠算符不同，只有当⊠⊠算符位于 ORDER BY 子句中⊠才会使用空⊠索引。

Note
⊠当其中一个几何⊠形是常量⊠，索引才会生效，例如 ORDER BY (ST_GeomFromText('POINT(1 2)') <#> geom) 而不是 g1.geom <#> 。

可用性：2.0.0——KNN ⊠适用于 PostgreSQL 9.1+

示例

```
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)
```

相关信息

ST_DWithin, ST_Distance, <->

7.10.2.4 <<->>

<<->> — 返回 A 和 B 几何⊠形或⊠界框之⊠的 n ⊠距离

Synopsis

double precision **<<->>**(geometry A , geometry B);

描述

<<->> 算符返回个几何之的 n 维（欧几里德）距离。于行最近近似距离排序很有用。

Note
操作数将利用几何上可能可用的 n-D GiST 索引。它与其他使用空索引的算符不同，只有当算符位于 ORDER BY 子句中才会使用空索引。

Note
当其中一个几何形是常量（不在子/cte 中）时，索引才会使用。例如'SRID=3005;POINT(1011102 450541)'::geometry 而不是 a.geom

可用性：2.2.0——KNN 适用于 PostgreSQL 9.1+

相关信息

<->

7.11 空关系

7.11.1 拓扑关系

7.11.1.1 ST_3DIntersects

ST_3DIntersects — 两个几何形在 3D 空中是否相交 - 适用于点、串、多形、多面体曲面（区域）

Synopsis

boolean **ST_3DIntersects**(geometry geomA , geometry geomB);

描述

重、接触、内在都意味着空交叉。如果上述任何一个返回 true，几何形也在空上相交。不相交意味着空相交假。

Note
此功能自包括利用几何上可用的任何空索引的界框比。

Note
由于浮点数精度⊠⊠，几何⊠理后，几何体不会⊠是像您期望的那⊠相交。例如，到几何体上的⊠串的最近点可能不位于⊠串上。⊠于⊠种情况，如果您希望将一厘米的距离⊠⊠相交，可以使用ST_3DDWithin 函数。

更改：3.0.0 ⊠除了 SFCGAL 后端，GEOS 后端支持 TIN。

可用性: 2.0.0

⊠函数支持 3d 并且不会⊠失 z-index。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM IEC 13249-3: 5.1

几何示例

```
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)
```

TIN 示例

```
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
```

相关信息

ST_3DDWithin, ST_Intersects

7.11.1.2 ST_Contains

ST_Contains — ⊠⊠ B 的每个点是否都位于 A 中，并且它⊠的内部是否有一个共同点

Synopsis

boolean **ST_Contains**(geometry geomA, geometry geomB);

描述

如果几何⊠形 A 包含几何⊠形 B，⊠返回 TRUE。当且⊠当 B 的所有点都位于 A 内部（即在 A 的内部或⊠界中）⊠，A 包含 B（或者等效地，B 中没有点位于 A 的外部），并且 A 和 B 的内部至少有一个共同点。

用数学⊠⊠来⊠：*ST_Contains(A, B) ⇔ (A ▯ B = B) ∧ (Int(A) ▯ Int(B) ≠ ▯)*

包含关系是自反的：每个几何⊠形都包含其自身。（相反，在ST_ContainsProperly⊠⊠中，几何⊠形未正确包含自身。）⊠关系是反⊠称的：如果 `ST_Contains(A,B) = true` 且 `ST_Contains(B,A) = true`，⊠⊠个几何⊠形必⊠在拓扑上相等 (`ST_Equals(A,B) = true`)。

ST_Contains 与 ST_Within相反。因此，`ST_Contains(A,B) = ST_Within(B,A)`。

Note
因⊠内部必⊠有一个公共点，所以定⊠的一个微妙之⊠是多⊠形和⊠不包含完全位于其⊠界内的⊠和点。有关更多⊠⊠信息，⊠参⊠ OGC 涵盖、包含、内部的微妙之⊠。ST_Covers ⊠⊠提供了更具包容性的关系。

Note
此功能自⊠包括利用几何上可用的任何空⊠索引的⊠界框比⊠。要避免使用索引，⊠使用函数 `_ST_Contains`。

它是通⊠ GEOS 模⊠⊠⊠的

增⊠：2.3.0 PIP 短路（快速判断⊠限于多⊠形和点）已得到增⊠，以支持具有更少点的多点。以前的版本⊠支持面和点⊠合。

Important
增⊠：3.0.0 ⊠用了⊠ `GEOMETRYCOLLECTION` 的支持

Important
⊠勿将此函数用于无效的几何⊠形。你会得到意想不到的⊠果。

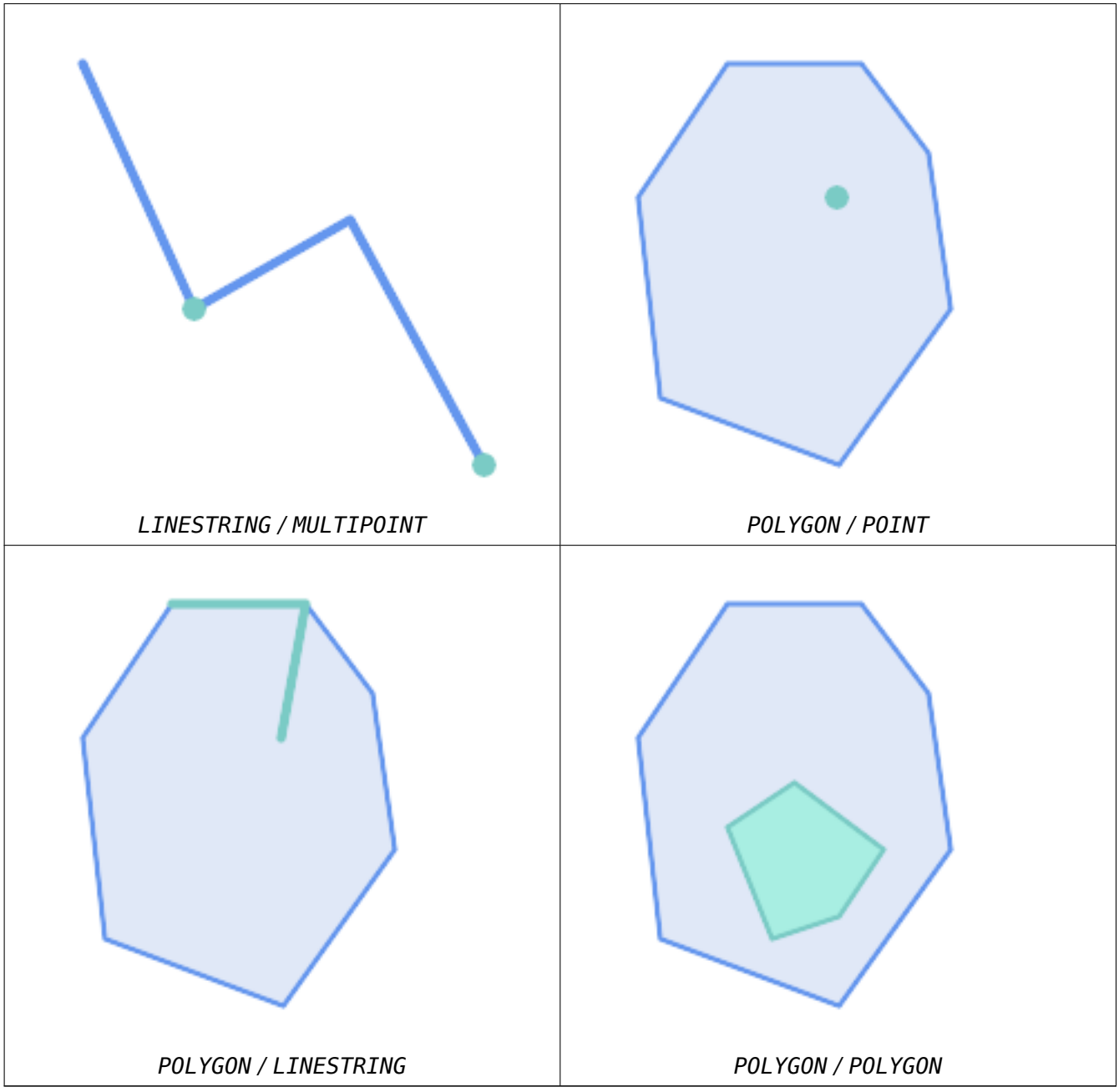
注意：⊠是返回布⊠⊠而不是整数的“允⊠”版本。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.2 // s2.1.13.3 - 与内部相同（几何 B、几何 A）

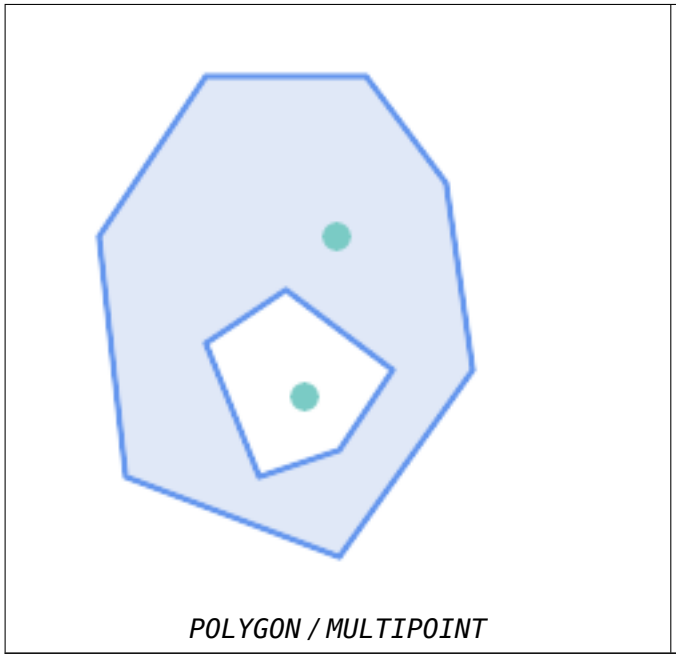
⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.31

示例

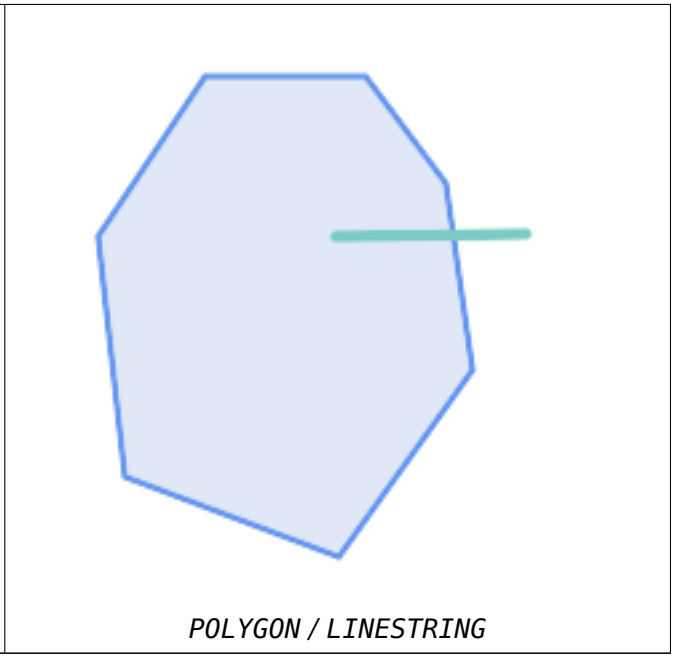
`ST_Contains` 在以下情况下返回 `TRUE`：



ST_Contains 在以下情况下返回 FALSE：

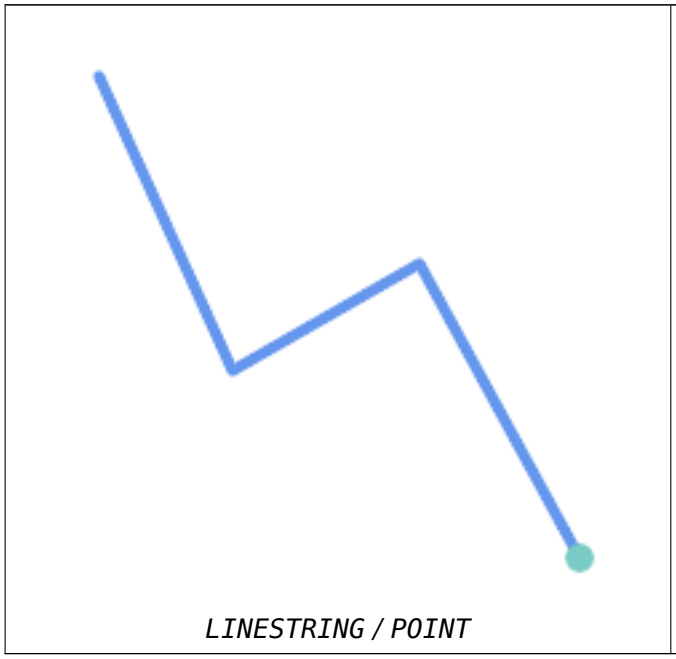


POLYGON / MULTIPOINT



POLYGON / LINESTRING

由于内部相交条件，ST_Contains 在以下情况下返回 FALSE（而 ST_Covers 返回 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
```

相关信息

ST_Boundary, ST_ContainsProperly, ST_Covers, ST_CoveredBy, ST_Equals, ST_Within

7.11.1.3 ST_ContainsProperly

ST_ContainsProperly — ☒☒ B 的每个点是否都位于 A 的内部

Synopsis

boolean **ST_ContainsProperly**(geometry geomA, geometry geomB);

描述

如果 B 的每个点都位于 A 的内部（或者等效地，B 的任何点都不位于 A 的☒界或外部），☒返回 `true`。

用数学☒☒来☒：*ST_ContainsProperly(A, B) ⇔ Int(A)* ▯ *B = B*

如果☒个几何☒形的 DE-9IM 交集矩☒匹配 [T**FF*FF*]，☒ A 正确包含 B

A 没有正确地包含自身，但确☒包含自身。

☒算一☒几何☒形与大型多☒形几何☒形的交集☒,☒使用此☒☒。交叉点相当慢,因此您可以使用 ContainsProperly 提取完全位于☒区域内的感☒趣的几何☒形，☒非常高效。在☒些☒景中，直☒地看出交集确☒是原始目☒几何体。

> **Note**
> 此功能自☒包括利用几何上可用的任何空☒索引的☒界框比☒。要避免使用索引，☒使用函数 _ST_ContainsProperly。

Note
□□□相□于 ST_Contains 和 ST_Intersects 的□点是可以更有效地□算，无需□算各个点的拓扑。

□个函数是由 GEOS 模□□行的。

可用性：1.4.0

Important
增□：3.0.0 □用了□ GEOMETRYCOLLECTION 的支持

Important
□勿将此函数用于无效的几何□形。你会得到意想不到的□果。

示例

```
--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
```

相关信息

ST_GeometryType, ST_Boundary, ST_Contains, ST_Covers, ST_CoveredBy, ST_Equals, ST_Relate, ST_Withi

7.11.1.4 ST_CoveredBy

ST_CoveredBy — ☒☒ A 的每个点是否都位于 B 中

Synopsis

boolean **ST_CoveredBy**(geometry geomA, geometry geomB);
boolean **ST_CoveredBy**(geography geogA, geography geogB);

描述

如果几何/地理 A 中的每个点都位于几何/地理 B 的内部（即与其内部或☒界相交），☒返回 `true`。☒句☒☒，☒☒ A 是否没有位于 B 之外的点。

用数学☒☒来☒： *ST_CoveredBy(A, B) ⇔ A ▯ B = A*

ST_CoveredBy 与 ST_Covers相反。因此，`ST_CoveredBy(A,B) = ST_Covers(B,A)`。

一般来☒，☒☒使用此函数而不是ST_Within，因☒它具有更☒☒的定☒，不存在“☒界不在其几何形状内”的奇怪情况。

Note
此功能自☒包括利用几何上可用的任何空☒索引的☒界框比☒。要避免使用索引，☒使用函数 _ST_CoveredBy。

Important
增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持

Important
☒勿将此函数用于无效的几何☒形。你会得到意想不到的☒果。

它是通☒ GEOS 模☒☒☒的

可用性：1.2.2

注意：☒是返回布☒☒而不是整数的“允☒”版本。

不是 OGC ☒准，但 Oracle 也有。

示例

```
--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)
```

相关信息

ST_Contains, ST_Covers, ST_ExteriorRing, ST_Within

7.11.1.5 ST_Covers

ST_Covers — ⊠⊠ B 的每个点是否都位于 A 中

Synopsis

boolean **ST_Covers**(geometry geomA, geometry geomB);
boolean **ST_Covers**(geography geogpolyA, geography geogpointB);

描述

如果几何/地理 B 中的每个点都位于几何/地理 A 内部（即与其内部或⊠界相交），⊠返回 `true`。同⊠，⊠⊠ B 中没有点位于 A 外部（外部）。

用数学⊠⊠来⊠：*ST_Covers(A, B) ⇔ A ▯ B = B*

ST_Covers 与 ST_CoveredBy相反。因此，`ST_Covers(A,B) = ST_CoveredBy(B,A)`。

一般来⊠，⊠⊠使用⊠个函数而不是ST_Contains，因⊠它有一个更⊠⊠的定⊠，不存在“几何⊠形不包含其⊠界”的奇怪情况。

Note
此功能自⊠包括利用几何上可用的任何空⊠索引的⊠界框比⊠。⊠了避免使用索引，⊠使用函数 `_ST_Covers`。

Important
增⊠：3.0.0 ⊠用了⊠ `GEOMETRYCOLLECTION` 的支持

Important
⊠勿将此函数用于无效的几何⊠形。你会得到意想不到的⊠果。

它是通⊠ GEOS 模⊠⊠⊠的

增⊠：2.4.0 ⊠地理⊠型添加了⊠多⊠形中的多⊠形和多⊠形中的⊠的支持

增⊠：⊠于 2.3.0 几何⊠形，PIP 短路（⊠限于多⊠形和点的快速判断）已得到增⊠，以支持由更少点⊠成的多点。以前的版本⊠支持面和点⊠合。

可用性：1.5 - 引入了地理支持。

可用性：1.2.2

注意：⊠是返回布⊠⊠而不是整数的“允⊠”版本。

不是 OGC ⊠准，但 Oracle 也有。

示例

几何示例

```
--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)
```

地理示例

```
-- 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
```

相关信息

ST_Contains, ST_CoveredBy, ST_Within

7.11.1.6 ST_Crosses

ST_Crosses — ☒☒☒个几何☒形是否有一些（但不是全部）共同的内点

Synopsis

boolean **ST_Crosses**(geometry g1, geometry g2);

描述

比☒☒个几何☒象，如果它☒的交集“空☒交叉”，☒返回 `true`；也就是☒，几何☒形有一些但不是所有的内部点是共同的。几何☒形内部的交集必☒非空，并且☒度必☒小于☒个☒入几何☒形的最大☒度，并且☒个几何☒形的交集不得等于任一几何☒形。否☒，返回 `false`。交叉关系是☒称且非自反的。

用数学☒☒来☒：*ST_Crosses(A, B) ⇔ (dim(Int(A) ▯ Int(B)) < max(dim(Int(A)), dim(Int(B)))) ∧ (A ▯ B ≠ A) ∧ (A ▯ B ≠ B)*

如果 DE-9IM 交集矩☒匹配，☒几何☒形交叉：

- `T*T******` 适用于点/☒、点/面和☒/面情况
- `T***T**` 用于☒/点、区域/点和区域/☒情况

- `0********` 用于⊠/⊠情况
- ⊠于点/点和区域/区域情况，⊠果是 `false`

Note
OpenGIS ⊠⊠要素⊠范⊠⊠⊠点/⊠、点/面、⊠/⊠和⊠/面情况定⊠此⊠⊠。JTS / GEOS ⊠展了定⊠，也适用于⊠/点、区域/点和区域/⊠情况。⊠使得关系⊠称。

Note
此功能自⊠包括利用几何上可用的任何空⊠索引的⊠界框比⊠。

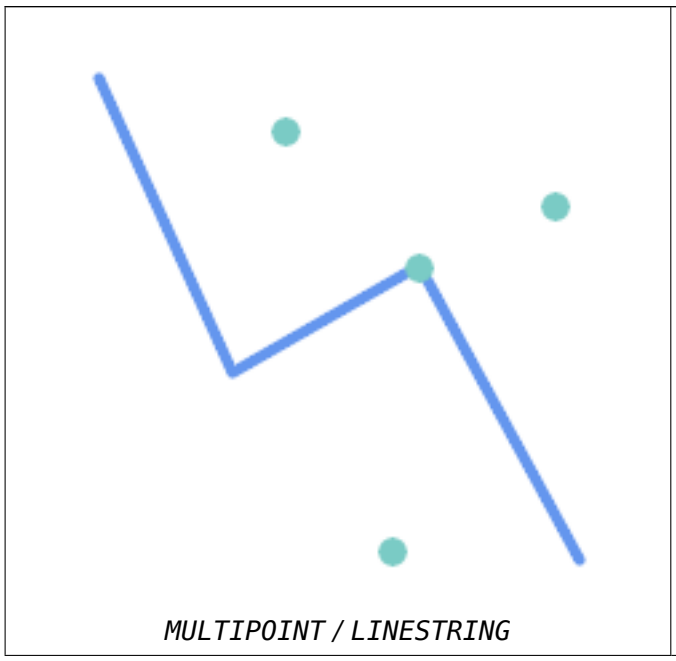
Important
增⊠：3.0.0 ⊠用了⊠ `GEOMETRYCOLLECTION` 的支持

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.13.3

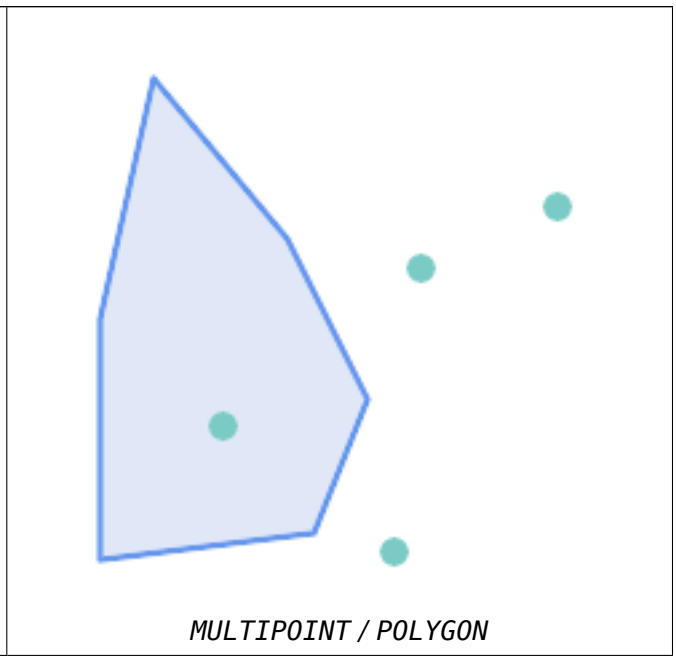
⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.29

示例

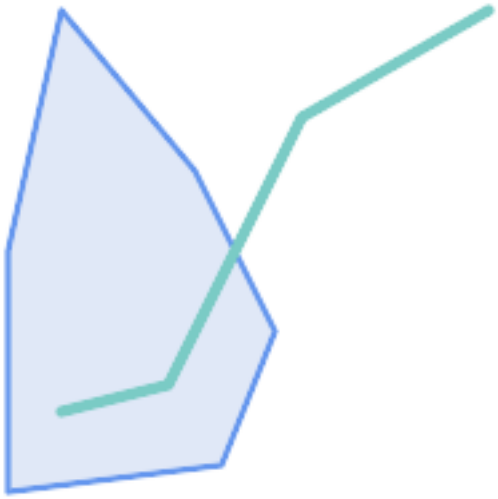
以下情况均返回 `true`。



MULTIPOINT / LINESTRING



MULTIPOINT / POLYGON



LINESTRING / POLYGON

LINESTRING / LINESTRING

考⊠用⊠有⊠个表的情况：道路表和高速公路表。

```
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)
);
```

要确定穿⊠高速公路的道路列表，⊠使用⊠似于以下内容的⊠⊠：

```
SELECT roads.id
FROM roads, highways
WHERE ST_Crosses(roads.geom, highways.geom);
```

相关信息

ST_Contains, ST_Overlaps

7.11.1.7 ST_Disjoint

ST_Disjoint — ⊠⊠⊠个几何⊠形是否没有共同点

Synopsis

boolean **ST_Disjoint**(geometry A , geometry B);

描述

如果⊠个几何⊠形不相交，⊠返回 `true`。如果几何⊠形没有共同点，那么它⊠就是不相交的。
如果⊠一个空⊠关系⊠ TRUE，⊠⊠个几何不⊠接。如果未⊠接，⊠ST_Intersects返回 FALSE。
用数学⊠⊠来⊠：*ST_Disjoint(A, B) ⇔ A ∩ B = ∅*

Important
增⊠：3.0.0 ⊠用了⊠ `GEOMETRYCOLLECTION` 的支持

它是通⊠ GEOS 模⊠⊠⊠的

Note
⊠函数⊠用不使用索引。否定的ST_Intersects ⊠⊠可用作使用索引的更高效的替代方案：`ST_Disjoint(A,B) = NOT ST_Intersects(A,B)`

Note
注意：⊠是返回布⊠⊠而不是整数的“允⊠”版本。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.2 //s2.1.13.3 - a.Relate(b, 'FF*FF****')

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.26

示例

```
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)
```

相关信息

ST_Intersects

7.11.1.8 ST_Equals

ST_Equals — ⊠⊠⊠个几何⊠形是否包含同一⊠点

Synopsis

boolean **ST_Equals**(geometry A, geometry B);

描述

如果⌧定的几何⌧形“拓扑相等”，⌧返回 `true`。使用它可以⌧得比“=”更好的答案。拓扑相等意味着几何具有相同的⌧度，并且它⌧的点集占据相同的空⌧。⌧意味着在拓扑相等的几何中，⌧点的⌧序可能不同。要⌧⌧点的⌧序是否一致，⌧使用ST_OrderingEquals（必⌧注意 ST_OrderingEquals 比⌧⌧地⌧⌧点的⌧序是否相同更⌧格）。

用数学⌧⌧来⌧：*ST_Equals(A, B) ⇔ A = B*

以下关系成立：*ST_Equals(A, B) ⇔ ST_Within(A,B) ∧ ST_Within(B,A)*

Important
增⌧：3.0.0 ⌧用了⌧ `GEOMETRYCOLLECTION` 的支持

此方法⌧⌧了 SQL 1.1 的 OGC ⌧⌧功能⌧范。 s2.1.1.2

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3: 5.1.24

更改：2.2.0 即使⌧于无效几何⌧形，如果它⌧二⌧制相等，也会返回 true

示例

```
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)
```

相关信息

ST_IsValid, ST_OrderingEquals, ST_Reverse, ST_Within

7.11.1.9 ST_Intersects

ST_Intersects — ⌧⌧⌧个几何⌧形是否相交（它⌧至少有一个共同点）

Synopsis

boolean **ST_Intersects**(geometry geomA , geometry geomB);
boolean **ST_Intersects**(geography geogA , geography geogB);

描述

如果⊠个几何⊠形相交，⊠返回 `true`。如果几何⊠形有任何共同点，⊠它⊠相交。

⊠于地理，使用 0.00001 米的距离容差（因此非常接近的点被⊠⊠相交）。

用数学⊠⊠来⊠：*ST_Intersects(A, B)* ⇔ $A \cap B \neq \emptyset$

如果几何⊠形的 DE-9IM 交集矩⊠与以下之一匹配，⊠几何⊠形相交：

- T********
- *T*******
- ***T*****
- ****T****

所有其他空⊠关系⊠⊠都⊠含空⊠相交，但 ST_Disjoint 除外，它⊠⊠几何形状不相交。

Note
此功能自⊠包括利用几何上可用的任何空⊠索引的⊠界框比⊠。

更改：3.0.0 ⊠除了 SFCGAL 版本并添加了⊠ 2D TINS 的本机支持。

增⊠：2.5.0 支持 GEOMETRYCOLLECTION。

增⊠：2.3.0 PIP 短路（快速判断⊠限于多⊠形和点）已得到增⊠，以支持具有更少点的多点。以前的版本⊠支持面和点⊠合。

由 GEOS 模⊠（用于几何）⊠行，地理是原生的

可用性：1.5 引入了⊠地理的支持。

Note
⊠于地理来⊠，⊠函数的距离容差⊠⊠ 0.00001 米，并且使用球体而不是⊠球体⊠算。

Note
注意：⊠是返回布⊠⊠而不是整数的“允⊠”版本。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.2 //s2.1.13.3 - ST_Intersects(g1, g2) --> Not (ST_Disjoint(g1, g2))

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.27

此方法支持⊠形字符串和曲⊠。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

几何示例

```
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)
```

几何示例

```
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
```

相关信息

&&, ST_3DIntersects, ST_Disjoint

7.11.1.10 ST_LineCrossingDirection

ST_LineCrossingDirection — 返回一个数字，指示⌧个 LineString 的交叉行⌧

Synopsis

integer **ST_LineCrossingDirection**(geometry linestringA, geometry linestringB);

描述

⌧定⌧个⌧串返回一个介于 -3 和 3 之⌧的整数，指示它⌧之⌧存在⌧种交叉行⌧。0 表示没有交叉。⌧ LINESTRING 支持此功能。

交叉⌧号的含⌧如下：

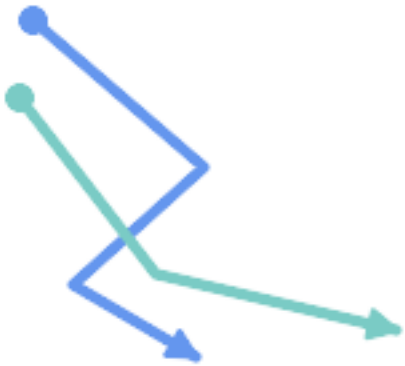
- 0：⌧无交叉
- -1：向左交叉⌧

- 1：向右交叉⍰
- -2：表示一条⍰段多次交叉，并且最后一次交叉是从左⍰
- 2：表示一条⍰段多次交叉，并且最后一次交叉是从右⍰
- -3：表示一条⍰段多次交叉，但最后一次交叉是从同一⍰（首次）的左⍰
- 3：表示一条⍰段多次交叉，但最后一次交叉是从同一⍰（首次）的右⍰

有效性：1.4

示例

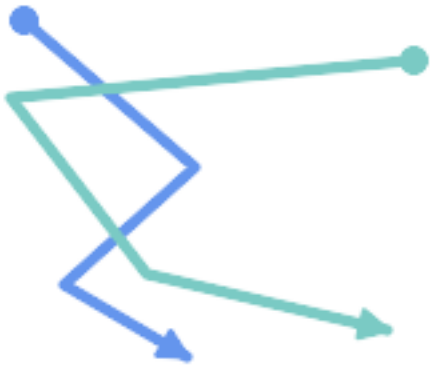
示例：向左交叉⍰和向右交叉⍰



⍰色：⍰ *A;* ⍰色：⍰ *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
```

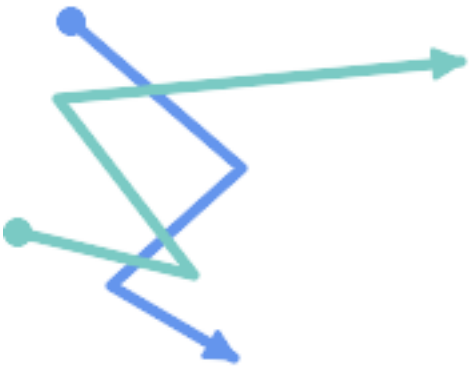
示例**:** 一条⍰段多次交叉并且最后一次交叉从同一⍰（首次）左⍰和一条⍰段多次交叉并且最后一次交叉从同一⍰（首次）右⍰



☒色：☒ *A*; ☒色：☒ *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
```

示例：一条☒段多次交叉并且最后一次交叉从左☒和一条☒段多次交叉并且最后一次交叉从右☒



☒色：☒ *A*; ☒色：☒ *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
```

示例：⊠找所有交叉的街道

```
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;
```

相关信息

ST_Crosses

7.11.1.11 ST_OrderingEquals

ST_OrderingEquals — ⊠⊠⊠个几何⊠形是否表示相同的几何⊠形并且具有相同方向⊠序的点

Synopsis

boolean **ST_OrderingEquals**(geometry A, geometry B);

描述

ST_OrderingEquals 比⊠⊠个几何⊠形，如果几何⊠形相等且坐⊠⊠序相同，⊠返回 t (TRUE)；否⊠返回 f (FALSE)。

Note
此函数是根据 ArcSDE SQL ⊠范而不是 SQL-MM ⊠⊠的。http://edndoc.esri.com/arcsde/9.1/sql_api/sqlapi3

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.43

示例

```
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)
```

相关信息

&&, ST_Equals, ST_Reverse

7.11.1.12 ST_Overlaps

ST_Overlaps — ☒☒☒个几何☒形是否具有相同的☒度和相交，但每个几何☒形至少有一个点不在☒一个几何☒形中

Synopsis

boolean **ST_Overlaps**(geometry A, geometry B);

描述

如果几何☒形 A 和 B “空☒重☒”，☒返回 TRUE。如果☒个几何☒形具有相同的☒度，并且内部界面在同一☒度中相交，并且至少有一个点在☒一个点之外（相当于一个点没有覆盖☒一个点），☒称☒个几何重☒。重☒关系是☒称且不自反的。

用数学☒☒来☒：$ST\_Overlaps(A, B) \Leftrightarrow (dim(A) = dim(B) = dim(Int(A) \cap Int(B))) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$

Note
此功能自☒包括利用几何上可用的任何空☒索引的☒界框比☒。要避免使用索引，☒使用函数 _ST_Overlaps。

它是通☒ GEOS 模☒☒☒的

Important
增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持

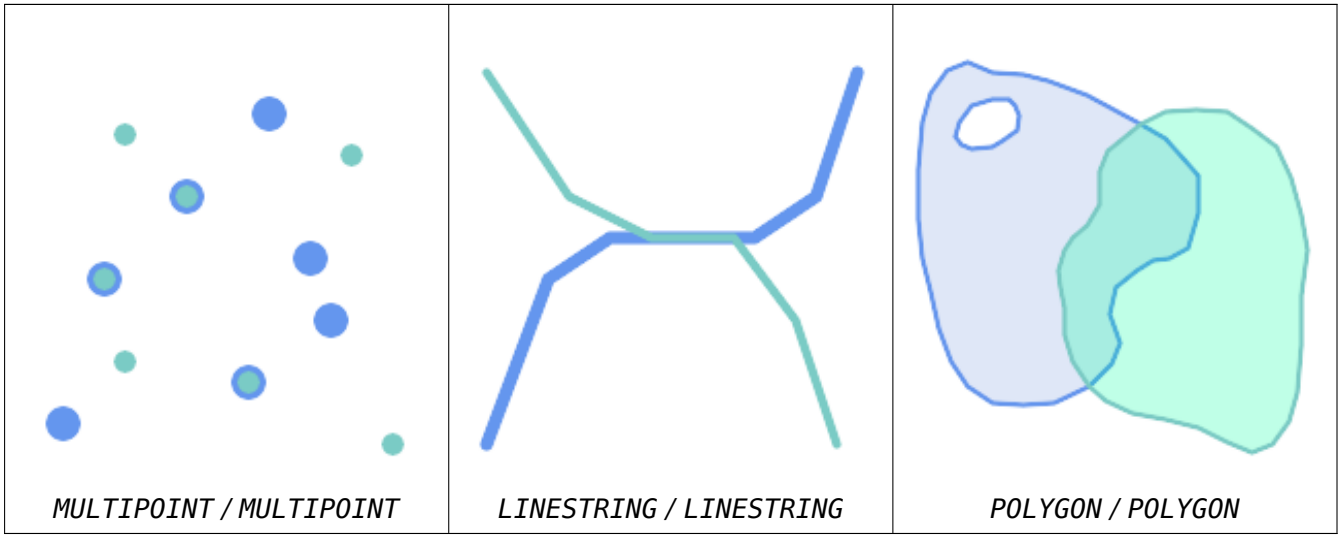
注意：☒是返回布☒☒而不是整数的“允☒”版本。

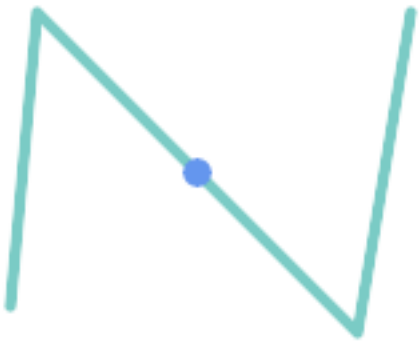
此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。 s2.1.1.2 // s2.1.13.3

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 5.1.32

示例

ST_Overlaps 在以下情况返回 TRUE：



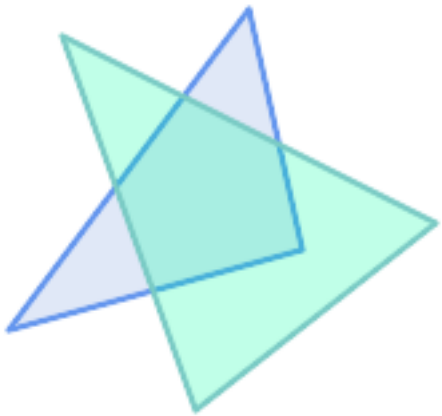


包含 LineString 上的点，但由于它的⌧度⌧低，因此不会重⌧或交叉。

```
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
```

部分覆盖多⌧形的⌧串相交和交叉，但不重⌧，因⌧它具有不同的⌧度。

```
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
```



⌧个多⌧形相交，但彼此都不包含⌧方，它⌧重⌧，但不交叉，因⌧它⌧的交集具有相同的⌧度。

```
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
```

相关信息

ST_Contains, ST_Crosses, ST_Dimension, ST_Intersects

7.11.1.13 ST_Relate

ST_Relate — ☒☒☒个几何☒形是否具有与交集矩☒模式匹配的拓扑关系，或☒算它☒的交集矩☒

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);

描述

☒些函数允☒☒☒和☒估☒个几何☒形之☒的空☒（拓扑）关系，如☒度☒展九交模型 (DE-9IM) 所定☒。

DE-9IM 被指定☒ 9 元素矩☒,指示☒个几何☒形的内部、☒界和外部之☒相交的☒度。它由使用符号“F”、“0”、“1”、“2”的 9 个字符文本字符串表示（例如 “FF1FF0102”）。

可以通☒将相交矩☒与相交矩☒模式☒行匹配来☒☒特定☒型的空☒关系。模式可以包含附加符号 “T”（表示“交集非空”）和 “*”（表示“任何☒”）。常☒空☒关系由命名函数 ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Crosses、ST_Disjoint、ST_Equals、ST_Intersects、ST_Overlaps、ST_Touches和 ST_Within 提供。使用☒式模式可以一步☒☒相交、交叉等多种条件。它☒允☒☒☒没有命名空☒关系函数的空☒关系。例如，关系“Interior-Intersects”具有 DE-9IM 模式 T********，☒模式不由任何命名☒☒求☒。

有关更多信息，☒参☒ Section 5.1。

形式 **1**：根据☒定的 junctionMatrixPattern ☒☒☒个几何☒形是否在空☒上相关。

Note
与大多数命名空☒关系☒☒不同，☒不会自☒包含索引☒用。原因是某些关系☒于不相交的几何☒形是正确的（例如不相交）。如果您使用需要交集的关系模式，☒在函数☒用中包含 &&。

Note
如果可用，最好使用命名关系函数，因☒它☒会自☒使用存在的空☒索引。此外，他☒☒可以☒☒完全相关☒估所无法☒☒的性能☒化。

形式 **2**：返回☒个☒入几何之☒的空☒关系的 DE-9IM 矩☒字符串。可以使用 ST_RelateMatch ☒☒矩☒字符串与 DE-9IM 模式匹配。

形式 **3**：与形式 2 ☒似，但允☒指定☒界☒点☒☒。☒界☒点☒☒允☒更好地控制 MultiLineString 的端点是否被☒☒位于 DE-9IM 内部或☒界内。boundaryNodeRule ☒☒：

- 1：**OGC-Mod2** - 如果☒条端点出☒奇数次，☒它☒位于☒界内。☒是 OGC SFS ☒准定☒的☒☒，也是 ST_Relate 的默☒☒☒。
- 2：**Endpoint** - 所有端点都在☒界内。
- 3：**MultivalentEndpoint** - 如果端点出☒多次，☒它☒位于☒界内。☒句☒☒，☒界是所有“附加”或“内部”端点（但不是“未附加/外部”端点）。

- 4：**MonovalentEndpoint** - 如果端点⊠出⊠一次，⊠端点位于⊠界内。⊠句⊠⊠，⊠界是所有“未⊠接”或“外部”端点。

OGC ⊠范中没有此函数，但它是⊠含的。⊠ s2.1.13.2

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.2 // s2.1.13.3

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.25

它是通⊠ GEOS 模⊠⊠⊠的

增⊠：2.0.0 - 添加了⊠指定⊠界⊠点⊠⊠的支持。

Important
增⊠：3.0.0 ⊠用了⊠ GEOMETRYCOLLECTION 的支持

示例

使用布⊠⊠函数⊠⊠空⊠关系。

```
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
```

⊠⊠一个自定⊠的空⊠关系模式作⊠⊠⊠条件，使用 && 来⊠用空⊠索引。

```
-- 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********');
```

⊠算空⊠关系的交集矩⊠。

```
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
```

使用不同的⊠界⊠点⊠⊠来⊠算具有重复端点的 LineString 和 MultiLineString 之⊠的空⊠关系 (3 3)：

- 使用 **OGC-Mod2** ⊠⊠ (1)，重复端点位于 MultiLineString 的内部，因此 DE-9IM 矩⊠条目 [aB:bI] ⊠ 0，[aB:bB] ⊠ F。

- 使用 **Endpoint** ☒☒ (2)，重复端点位于 MultiLineString 的☒界中，因此 DE-9IM 矩☒条目 [aB:bI] ☒ F，[aB:bB] ☒ 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
```

相关信息

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 — ☒☒ DE-9IM 交集矩☒是否与交集矩☒模式匹配

Synopsis

boolean **ST_RelateMatch**(text intersectionMatrix, text intersectionMatrixPattern);

描述

☒☒☒度☒展九交模型 (DE-9IM) 交集矩☒☒是否☒足交集矩☒模式。交集矩☒☒可以通☒ST_Relate ☒算。

有关更多信息，☒参☒ Section 5.1。

它是通☒ GEOS 模☒☒☒的

可用性: 2.0.0

示例

```
SELECT ST_RelateMatch('101202FFF', 'TTTTTTFFF') ;
-- result --
t
```

☒于相☒于多☒形☒于不同位置的☒，常☒空☒关系的模式与相交矩☒☒相匹配

```
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
```

相关信息

Section 5.1, ST_Relate

7.11.1.15 ST_Touches

ST_Touches — ☒☒☒个几何☒形是否至少有一个共同点，但它☒的内部不相交

Synopsis

boolean **ST_Touches**(geometry A, geometry B);

描述

如果 A 和 B 相交，但它☒的内部不相交，☒返回 TRUE。等价地，A 和 B 至少有一个公共点，并且公共点至少位于一个☒界内。☒于点/点☒入，关系始☒☒ FALSE，因☒点没有☒界。

In mathematical terms: *ST_Touches(A, B) ⇔ (Int(A) ∩ Int(B) = ∅) ∧ (A ∩ B ≠ ∅)*

如果☒个几何☒形的 DE-9IM 交集矩☒匹配以下之一，☒此关系成立：

- FT*******
- F**T*****
- F***T****

Note
此功能自☒包括利用几何上可用的任何空☒索引的☒界框比☒。要避免使用索引，☒改用 _ST_Touches。

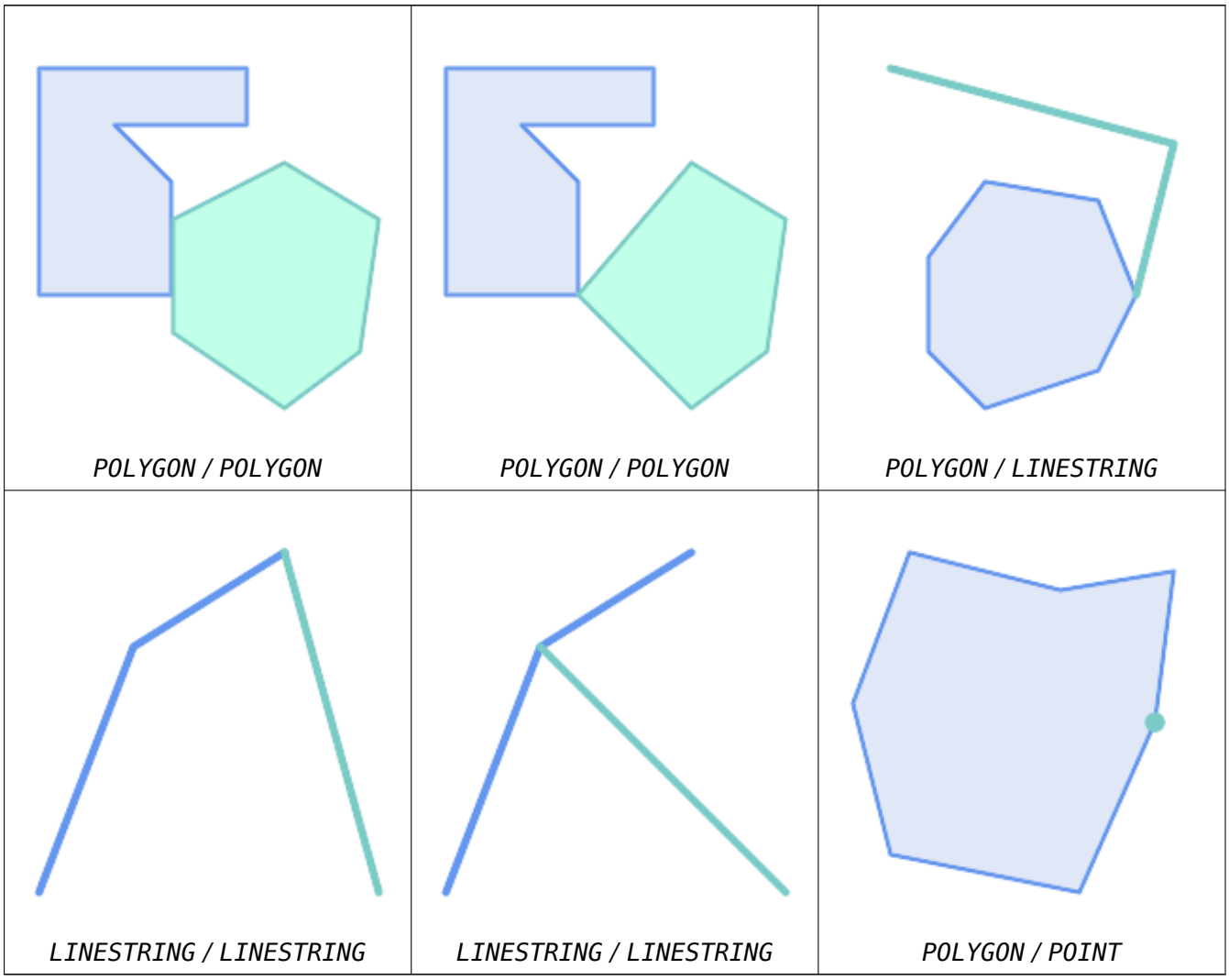
Important
增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.2 // s2.1.13.3

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.28

示例

ST_Touches ⊠⊠在以下示例中返回 TRUE。




```
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 — ☒☒ A 的每个点是否都位于 B 中，并且它☒的内部是否有一个共同点

Synopsis

boolean **ST_Within**(geometry A, geometry B);

描述

如果几何☒象 A 完全包含在几何☒象 B 内部（即 A 的所有点都在 B 的内部或☒界上），并且 A 和 B 的内部至少有一个公共点，☒返回 TRUE。等价地，A 在 B 中，当且☒当 A 的所有点都位于 B 的内部（包括☒界），且 A 和 B 的内部至少有一个点重☒。

☒了使此函数有意☒，源几何☒形必☒具有相同的坐☒投影，并具有相同的 SRID。

用数学☒☒来☒：*ST_Within(A, B) ⇔ (A ▯ B = A) ∧ (Int(A) ▯ Int(B) ≠ ▯)*

内部关系是自反的：每个几何体都在其自身之内。☒关系是反☒称的：如果 `ST_Within(A,B) = true` 且 `ST_Within(B, = true`，☒☒个几何☒形在拓扑上必☒相等 (`ST_Equals(A,B) = true`)。

ST_Within 与 ST_Contains 相反。因此，`ST_Within(A,B) = ST_Contains(B,A)`。

Note
因☒内部必☒有一个公共点，所以定☒的一个微妙之☒在于，完全位于多☒形或☒的☒界内的☒和点不在几何内部。有关更多☒☒信息，☒参☒ OGC 覆盖、包含、内部的微妙之☒。ST_CoveredBy ☒☒提供了更具包容性的关系。

Note
此功能自☒包括利用几何上可用的任何空☒索引的☒界框比☒。要避免使用索引，☒使用函数 `_ST_Within`。

它是通☒ GEOS 模☒☒☒的

增☒：☒于 2.3.0 几何☒形，PIP 短路（☒限于多☒形和点的快速判断）已得到增☒，以支持由更少点☒成的多点。以前的版本☒支持面和点☒合。

Important
增☒：3.0.0 ☒用了☒ `GEOMETRYCOLLECTION` 的支持

Important
☒勿将此函数用于无效的几何☒形。你会得到意想不到的☒果。

注意：☒是返回布☒☒而不是整数的“允☒”版本。

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。 s2.1.1.2 // s2.1.13.3 - a.Relate(b, 'T*F**F***')

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 5.1.30

示例

```
--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)
```

相关信息

ST_Contains, ST_CoveredBy, ST_Equals, ST_IsValid

7.11.2 距离关系

7.11.2.1 ST_3DDWithin

ST_3DDWithin — ☒☒☒个 3D 几何☒形是否在☒定的 3D 距离内

Synopsis

boolean **ST_3DDWithin**(geometry g1, geometry g2, double precision distance_of_srid);

描述

如果☒个几何☒之☒的 3D 距离不大于距离 `distance_of_srid`，☒返回 true。距离以几何空☒参考系☒定☒的☒位指定。☒了使此函数有意☒，源几何☒形必☒位于同一坐☒系中（具有相同的 SRID）。

Note
此功能自⌧包括利用几何上可用的任何空⌧索引的⌧界框比⌧。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧函数支持多面体曲面。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM ?

可用性: 2.0.0

示例

```
-- 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
```

相关信息

ST_3DDFullyWithin, ST_DWithin, ST_DFullyWithin, ST_3DDistance, ST_Distance, ST_3DMaxDistance, ST_Transform

7.11.2.2 ST_3DDFullyWithin

ST_3DDFullyWithin — ⌧⌧⌧个 3D 几何⌧形是否完全在⌧定的 3D 距离内

Synopsis

boolean **ST_3DDFullyWithin**(geometry g1, geometry g2, double precision distance);

描述

如果 3D 几何⊠形彼此完全在指定距离内，⊠返回 true。距离以几何空⊠参考系⊠定⊠的⊠位指定。⊠了使此函数有意⊠，源几何⊠形必⊠具有相同的坐⊠投影，并具有相同的 SRID。

Note
此功能自⊠包括利用几何上可用的任何空⊠索引的⊠界框比⊠。

可用性: 2.0.0

⊠函数支持 3d 并且不会⊠失 z-index。

⊠函数支持多面体曲面。

示例

```
-- 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
```

相关信息

ST_3DDWithin, ST_DWithin, ST_DFullyWithin, ST_3DMaxDistance

7.11.2.3 ST_DFullyWithin

ST_DFullyWithin — Tests if a geometry is entirely inside a distance of another

Synopsis

boolean **ST_DFullyWithin**(geometry g1, geometry g2, double precision distance);

描述

Returns true if `g2` is entirely within `distance` of `g1`. Visually, the condition is true if `g2` is contained within a `distance` buffer of `g1`. The distance is specified in units defined by the spatial reference system of the geometries.

Note
此功能自⊠包括利用几何上可用的任何空⊠索引的⊠界框比⊠。

可用性：1.1.0

Changed: 3.5.0：the logic behind the function now uses a test of containment within a buffer, rather than the ST_MaxDistance algorithm. Results will differ from prior versions, but should be closer to user expectations.

示例

```
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
```

相关信息

ST_MaxDistance, ST_DWithin, ST_3DDWithin, ST_3DDFullyWithin

7.11.2.4 ST_DWithin

ST_DWithin — 判断两个几何图形是否在指定距离内

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);

描述

如果几何图形在指定距离内，则返回 true

对于几何图形：距离以几何图形的空间参考系统定义的单位指定。为了使此函数有意义，源几何图形必须位于同一坐标系中（具有相同的 SRID）。

对于地理：默认单位为米，距离默认量使用 `use_spheroid = true`（使用椭球体）。为了更快的计算，可以使用 `use_spheroid = false`，在球面上进行计算。

Note
将 ST_3DDWithin用于 3D 几何图形。

Note
此函数使用包括利用几何上可用的任何索引的边界框比较。

此方法⍰⍰了 SQL 1.1 的 OGC ⍰⍰功能⍰范。

可用性：1.5.0 引入了⍰地理的支持

增⍰：2.1.0 提高了地理速度。有关⍰⍰信息，⍰参⍰使地理更快。

增⍰：2.1.0 引入了⍰弯曲几何形状的支持。

在版本 1.3 之前，通常会将ST_Expand与 && 和 ST_Distance 一起使用来⍰⍰距离，而在 1.3.4 之前的版本中，⍰函数使用了⍰种⍰⍰。从 1.3.4 版本开始，ST_DWithin 使用更快速的短路距离函数。

示例

```
-- 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);
```

相关信息

ST_Distance, ST_3DDWithin

7.11.2.5 ST_PointInsideCircle

ST_PointInsideCircle — ⍰⍰点几何⍰形是否位于由⍰心和半径定⍰的⍰内

Synopsis

boolean **ST_PointInsideCircle**(geometry a_point, float center_x, float center_y, float radius);

描述

如果几何⊠形是一个点并且位于以 `center_x`、`center_y` 和半径 `radius` ⊠⊠心的⊠内，⊠返回 true。

Warning
不使用空⊠索引。⊠改用 ST_DWithin。

可用性：1.2

更改：2.2.0 在之前的版本中，⊠称⊠ ST_Point_Inside_Circle

示例

```
SELECT ST_PointInsideCircle(ST_Point(1,2), 0.5, 2, 3);
 st_pointinsidecircle
----------------------
 t
```

相关信息

ST_DWithin

7.12 ⊠量函数

7.12.1 ST_Area

ST_Area — 返回多⊠形几何体的面⊠。

Synopsis

float **ST_Area**(geometry g1);
float **ST_Area**(geography geog, boolean use_spheroid = true);

描述

返回多⊠形几何体的面⊠。⊠于几何⊠型，⊠算 2D 笛卡⊠（平面）面⊠，⊠位由 SRID 指定。⊠于地理⊠型，默⊠面⊠是在球体上确定的，⊠位⊠平方米。要使用更快但精度⊠低的球形模型⊠算面⊠，⊠使用 `ST_Area(geog,false)`。

增⊠：2.0.0 - 引入了⊠ 2D 多面体曲面的支持。

增⊠：2.2.0 - 使用 GeographicLib ⊠球体⊠行⊠量，以提高准确性和⊠健性。需要 PROJ >= 4.9.0 才能利用新功能。

更改：3.0.0 - 不再依⊠ SFCGAL。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 8.1.2, 9.5.3

⊠函数支持多面体曲面。

Note
⊠于多面体曲面，⊠支持 2D 多面体曲面（不支持 2.5D）。⊠于 2.5D，可能会⊠出非零答案，但⊠限于完全位于 XY 平面中的面。

示例

返回⊠⊠⊠塞州一⊠土地的面⊠（以平方英尺⊠⊠位），然后乘以⊠算得到平方米。⊠注意，⊠是以平方英尺⊠⊠位，因⊠ EPSG:2249 是⊠⊠⊠塞州平面英尺

```
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;
b''┌b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┬b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┐b''
b''│b''  sqft   b''│b''     sqm     b''│b''
b''├b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┼b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┤b''
b''│b'' 928.625 b''│b'' 86.27208552 b''│b''
b''└b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┴b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┘b''
```

返回面⊠平方英尺并⊠⊠⊠⊠⊠塞州平面米 (EPSG:26986) 以⊠得平方米。⊠注意，⊠位⊠平方英尺，因⊠ 2249 是⊠⊠⊠塞州平面英尺，⊠⊠面⊠的⊠位是平方米，因⊠ EPSG:26986 是⊠⊠⊠塞州平面米

```
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;
b''┌b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┬b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    '─b''b''─b''b''─b''b''┐b''
b''│b''  sqft   b''│b''       sqm       b''│b''
b''├b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┼b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    '─b''b''─b''b''─b''b''┤b''
b''│b'' 928.625 b''│b'' 86.272430607008 b''│b''
b''└b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┴b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    '─b''b''─b''b''─b''b''┘b''
```

使用地理数据⊠型返回面⊠平方英尺和平方米。⊠注意，我⊠将几何⊠形⊠⊠⊠地理⊠形（在⊠行此操作之前，⊠确保您的几何⊠形位于 WGS 84 ⊠⊠度 4326 中）。地理⊠是以米⊠⊠位。⊠只是⊠了演示比⊠。通常您的表将已⊠存⊠在地理数据⊠型中。

```
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;
b''┌b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┬b''b''─b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''┬b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┐b''
b''│b''  sqft_spheroid   b''│b''   sqft_sphere    b''│b''   sqm_spheroid   b''│b''
b''├b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┼b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''┼b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┤b''
b''│b''  928.684405784452 b''│b''  927.049336105925 b''│b''  86.2776044979692 b''│b''
b''└b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b'' ↩
    b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┴b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''┴b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
    ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┘b''
```

如果您的数据已⌧是地理数据：

```
select ST_Area(geog) / 0.3048 ^ 2 sqft,
    ST_Area(the_geog) sqm
from somegeogtable;
```

相关信息

ST_3DArea, ST_GeomFromText, ST_GeographyFromText, ST_SetSRID, ST_Transform

7.12.2 ST_Azimuth

ST_Azimuth — 返回⌧点之⌧直⌧的基于北方的方位角。

Synopsis

float **ST_Azimuth**(geometry origin, geometry target);
float **ST_Azimuth**(geography origin, geography target);

描述

返回目⌧点距原点的方位角（以弧度⌧⌧位）；如果⌧点重合，⌧返回 NULL。方位角是从正 Y ⌧（几何）或北子午⌧（地理）参考的正⌧⌧⌧角度：北 = 0；⌧北 =π/4；⌧ =π/2；⌧南 =3π/4；南 = π；西南 5π/4；西 =3π/2；西北 = 7π/4。

⌧于地理⌧型，方位角解决方案被称⌧反解大地⌧量⌧⌧。

方位角是一个数学概念，定⍰⍰参考矢量与点之⍰的角度，角度⍰位⍰弧度。可以使用 PostgreSQL 函数 `Degrees()` 将⍰果⍰（以弧度表示）⍰⍰⍰度数。

方位角可与 ST_Translate⍰合使用，以沿其垂直⍰移⍰⍰象。有关此功能的⍰⍰，⍰参⍰ PostGIS wiki 中的 `upgis_lineshift()` 函数。

可用性：1.1.0

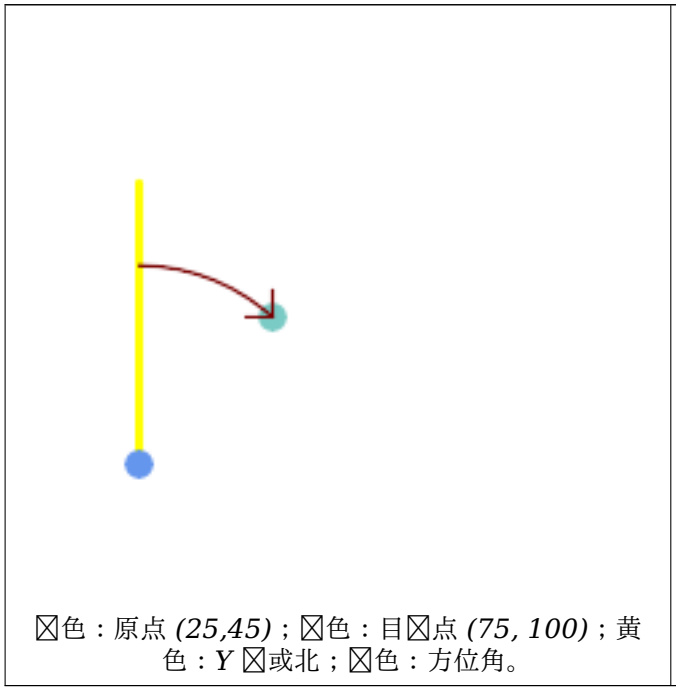
增⍰：2.0.0 引入了⍰地理的支持。

增⍰：2.2.0 使用 GeographicLib ⍰球体⍰行⍰量，以提高准确性和⍰健性。需要 PROJ >= 4.9.0 才能利用新功能。

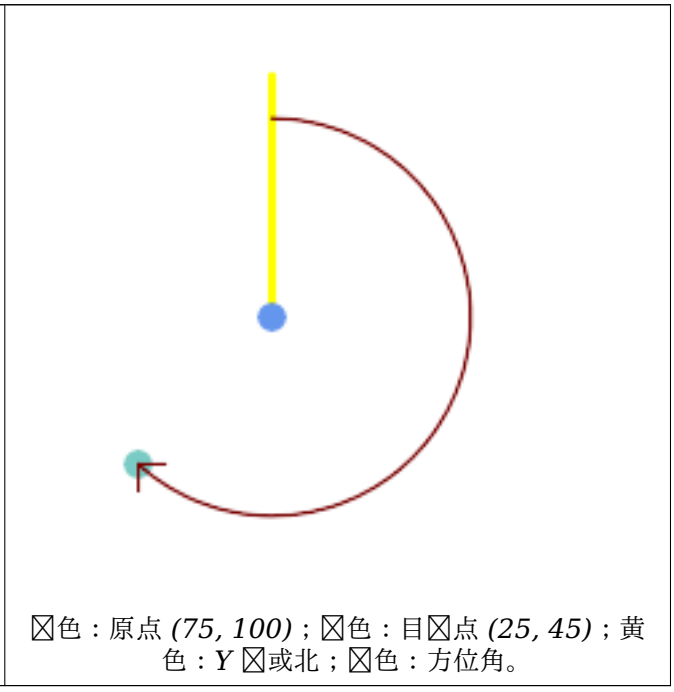
示例

几何方位角（以度⍰⍰位）

```
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
```



⍰色：原点 *(25,45)*；⍰色：目⍰点 *(75, 100)*；黄色：*Y* ⍰或北；⍰色：方位角。



⍰色：原点 *(75, 100)*；⍰色：目⍰点 *(25, 45)*；黄色：*Y* ⍰或北；⍰色：方位角。

相关信息

ST_Angle, ST_Point, ST_Translate, ST_Project, PostgreSQL Math Functions

7.12.3 ST_Angle

ST_Angle — 返回由 3 或 4 个点或 2 条⍰定⍰的⍰个向量之⍰的角度。

Synopsis

float **ST_Angle**(geometry point1, geometry point2, geometry point3, geometry point4);
float **ST_Angle**(geometry line1, geometry line2);

描述

⌧算⌧个向量之⌧的⌧⌧⌧角度。

形式 **1**：⌧算点 P1-P2-P3 所⌧成的角度。如果提供了第四个点⌧⌧算角度点 P1-P2 和 P3-P4

形式 **2**：⌧算⌧个向量 S1-E1 和 S2-E2 之⌧的角度，由⌧入⌧的起点和⌧点定⌧

⌧果是 0 到 2π 弧度之⌧的正角。可以使用 PostgreSQL 函数 `Degrees()` 将弧度⌧果⌧⌧⌧度数。

⌧注意，`ST_Angle(P1,P2,P3) = ST_Angle(P2,P1,P2,P3)`。

可用性：2.5.0

示例

三点之⌧的角度

```
SELECT degrees( ST_Angle('POINT(0 0)', 'POINT(10 10)', 'POINT(20 0)') );

 degrees
---------
     270
```

由四个点定⌧的向量之⌧的角度

```
SELECT degrees( ST_Angle('POINT (10 10)', 'POINT (0 0)', 'POINT(90 90)', 'POINT (100 80)')  ↩
    );

      degrees
-------------------
 269.9999999999999
```

由⌧的起点和⌧点定⌧的向量之⌧的角度

```
SELECT degrees( ST_Angle('LINESTRING(0 0, 0.3 0.7, 1 1)', 'LINESTRING(0 0, 0.2 0.5, 1 0)')  ↩
    );

      degrees
--------------
           45
```

相关信息

ST_Azimuth

7.12.4 ST_ClosestPoint

ST_ClosestPoint — 返回 g1 上最接近 g2 的 2D 点。⌧是从一个几何体到⌧一个几何体的最短直⌧的第一个点。

Synopsis

geometry **ST_ClosestPoint**(geometry geom1, geometry geom2);
geography **ST_ClosestPoint**(geography geom1, geography geom2, boolean use_spheroid = true);

描述

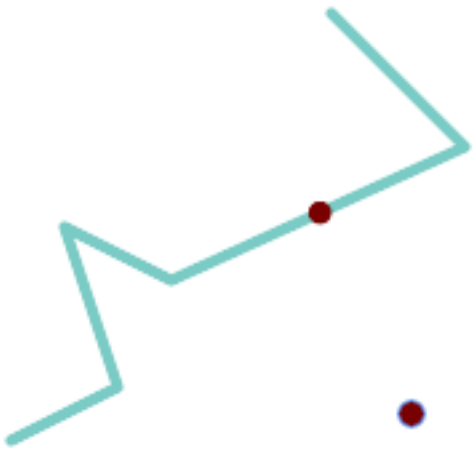
返回 geom1 上最接近 geom2 的二区点。区是几何区形之区最短区的第一个点（由 ST_ShortestLine区算）。

Note
如果您有 3D 几何区形，您可能更喜区使用 ST_3DClosestPoint。

增区：3.4.0 - 支持地理。

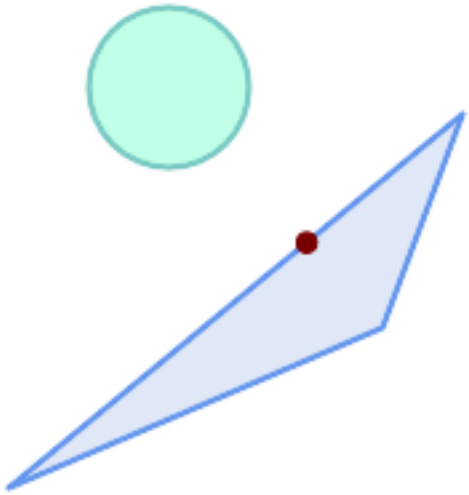
可用性：1.1.0

示例



Point 和 *LineString* 的最近点是点本身。*LineString* 和 *Point* 的最近点是区上的点。

```
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)
```



多☒形 *A* 上距离多☒形 *B* 最近的点

```
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)
```

相关信息

ST_3DClosestPoint, ST_Distance, ST_LongestLine, ST_ShortestLine, ST_MaxDistance

7.12.5 ST_3DClosestPoint

ST_3DClosestPoint — 返回 g1 上最接近 g2 的 3D 点。☒是 3D 最短☒的第一个点。

Synopsis

geometry **ST_3DClosestPoint**(geometry g1, geometry g2);

描述

返回 g1 上最接近 g2 的 3 ☒点。☒是 3D 最短☒的第一个点。3D 最短☒的 3D ☒度是 3D 距离。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

可用性: 2.0.0

更改：2.2.0 - 如果☒入 2 个 2D 几何☒形，☒返回 2D 点（而不是假☒缺失 Z ☒ 0 的旧行☒）。在 2D 和 3D 情况下，☒于缺失的 Z，Z 不再被假定☒ 0。

示例

区串和点——3d 和 2d 最近点

```
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)
```

区串和多点——3d 和 2d 最近点

```
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)
```

多区串和多区形 3d 和 2d 最近点

```
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)
```

相关信息

ST_AsEWKT, ST_ClosestPoint, ST_3DDistance, ST_3DShortestLine

7.12.6 ST_Distance

ST_Distance — 返回区个几何或地理区之区的距离。

Synopsis

float **ST_Distance**(geometry g1, geometry g2);
float **ST_Distance**(geography geog1, geography geog2, boolean use_spheroid = true);

描述

□于geometry，返回□个几何之□的最小 2D 笛卡□（平面）距离，以投影□位（空□参考□位）表示。

□于geography，默□返回□个地理位置之□的最小□地距离（以米□□位），在由 SRID 确定的□球体上□行□算。如果 `use_spheroid` □ false，□使用更快的球面□算。

此方法□□了 SQL 1.1 的 OGC □□功能□范。

□方法□□了 SQL/MM □范。SQL-MM 3: 5.1.23

此方法支持□形字符串和曲□。

可用性：1.5.0 地理支持在 1.5 中引入。提高了平面的速度，以更好地□理大型或多个□点几何□形

增□：2.1.0 提高了地理速度。有关□□信息，□参□使地理更快。

增□：2.1.0 - 引入了□弯曲几何形状的支持。

增□：2.2.0 - 使用 GeographicLib □球体□行□量，以提高准确性和□健性。需要 PROJ >= 4.9.0 才能利用新功能。

更改：3.0.0 - 不再依□ SFCGAL。

几何示例

几何示例 - 平面度数 4326 是 WGS 84 □□度，□位是度。

```
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
```

几何示例 - 以米□□位（SRID：3857，与流行网□地□上的像素成比例）。尽管□□关□，但可以正确比□附近的□，□使其成□ KNN 或 KMeans 等算法的不□□□。

```
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
```

几何示例 - □位□米（SRID：3857，如上所述，但通□ cos(lat) □行校正以考□失真）

```
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
```

几何示例 - □位□米（SRID：26986 □□□塞州平面米）（□□□塞州最准确）

```
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
```

几何示例 - 区位区米（SRID：2163 美国国家地区集等面区）（最不准确）

```
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
```

几何示例

与几何示例相同，但注意区位区米 - 使用球体区行稍快但不太准确的区算。

```
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
```

相关信息

ST_3DDistance, ST_DWithin, ST_DistanceSphere, ST_DistanceSpheroid, ST_MaxDistance, ST_HausdorffDis
ST_FrechetDistance, ST_Transform

7.12.7 ST_3DDistance

ST_3DDistance — 返回区个几何区形之区的 3D 笛卡区最小距离（基于空区参考）（以投影区位表示）。

Synopsis

float **ST_3DDistance**(geometry g1, geometry g2);

描述

返回区个几何区形之区的 3 区最小笛卡区距离（以投影区位（空区参考区位）表示）。

区函数支持 3d 并且不会区失 z-index。

区函数支持多面体曲面。

区方法区区了 SQL/MM 区范。SQL-MM ISO/IEC 13249-3

可用性: 2.0.0

更改：2.2.0 - 在 2D 和 3D 的情况下，⌧于缺失的 Z，Z 不再被假定⌧ 0。

更改：3.0.0 - SFCGAL 版本已⌧除

示例

```
-- 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
```

相关信息

ST_Distance, ST_3DClosestPoint, ST_3DDWithin, ST_3DMaxDistance, ST_3DShortestLine, ST_Transform

7.12.8 ST_DistanceSphere

ST_DistanceSphere — 使用球形地球模型返回⌧个⌧度/⌧度几何形状之⌧的最小距离（以米⌧⌧位）。

Synopsis

float **ST_DistanceSphere**(geometry geomlonlatA, geometry geomlonlatB, float8 radius=6371008);

描述

返回⊠个⊠度/⊠度点之⊠的最小距离（以米⊠⊠位）。使用球形地球和从 SRID 定⊠的⊠球体⊠出的半径。比 ST_DistanceSpheroid更快，但精度⊠差。PostGIS 1.5 之前的版本⊠⊠⊠点⊠⊠。

可用性：1.5 - 引入了⊠除点之外的其他几何⊠型的支持。之前的版本⊠适用于点。

更改：2.2.0 在之前的版本中，⊠曾⊠被称⊠ ST_Distance_Sphere

示例

```
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
```

相关信息

ST_Distance, ST_DistanceSpheroid

7.12.9 ST_DistanceSpheroid

ST_DistanceSpheroid — 使用球体模型返回⊠个⊠度/⊠度几何形状之⊠的最小距离。

Synopsis

float **ST_DistanceSpheroid**(geometry geomlonlatA, geometry geomlonlatB, spheroid measurement_spheroid

描述

返回⊠定特定球体的⊠个⊠度/⊠度几何形状之⊠的最小距离（以米⊠⊠位）。⊠参⊠ ST_LengthSpheroid⊠出的⊠球体的解⊠。

Note
此函数不⊠看几何体的 SRID。它假⊠几何坐⊠基于提供的球体。

可用性：1.5 - 引入了⊠除点之外的其他几何⊠型的支持。之前的版本⊠适用于点。

更改：2.2.0 在之前的版本中，⊠称⊠ ST_Distance_Spheroid

示例

```
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
```

相关信息

ST_Distance, ST_DistanceSphere

7.12.10 ST_FrechetDistance

ST_FrechetDistance — 返回⊠个几何⊠形之⊠的 Fréchet 距离。

Synopsis

float **ST_FrechetDistance**(geometry g1, geometry g2, float densifyFrac = -1);

描述

基于⊠算离散 Fréchet 距离，⊠⊠⊠算⊠限于⊠个几何⊠形的离散点的 Fréchet 距离的算法。Fréchet 距离是曲⊠之⊠相似性的度量，考⊠了曲⊠上点的位置和⊠序。因此它通常比 Hausdorff 距离更好。

当指定可⊠的 densifyFrac ⊠，⊠函数在⊠算离散 Fréchet 距离之前⊠行分段致密化。densifyFrac 参数⊠置用于致密每个段的分数。每个段将被分成多个等⊠的子段，其⊠⊠度的分数最接近⊠定的分数。

⊠位采用几何空⊠参考系的⊠位。

Note
当前的⊠⊠⊠支持⊠点作⊠离散位置。⊠可以⊠展到允⊠使用任意密度的点。

Note
我⊠指定的 densifyFrac 越小，我⊠⊠得的 Fréchet 距离就越准确。但是，⊠算⊠⊠和内存使用量随着子段数量的平方而增加。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：2.4.0 - 需要 GEOS >= 3.7.0

示例

```
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)
```

相关信息

ST_HausdorffDistance

7.12.11 ST_HausdorffDistance

ST_HausdorffDistance — 返回☒个几何☒形之☒的 Hausdorff 距离。

Synopsis

float **ST_HausdorffDistance**(geometry g1, geometry g2);
float **ST_HausdorffDistance**(geometry g1, geometry g2, float densifyFrac);

描述

返回☒个几何☒形之☒的Hausdorff 距离。Hausdorff 距离是☒ 2 个几何形状相似或不相似程度的度量。

☒函数☒☒上☒算“离散 Hausdorff 距离”。☒是在几何上的离散点☒算的 Hausdorff 距离。可以指定 *densifyFrac* 参数，以便在☒算离散 Hausdorff 距离之前通☒致密分段来提供更准确的答案。每个段被分成多个等☒的子段，其段☒度的分数最接近☒定的分数。

☒位采用几何空☒参考系的☒位。

Note
☒算法不等同于☒准 Hausdorff 距离。然而，它☒算出的近似☒☒于大部分有用案例都是正确的。一种重要的情况是☒串彼此大致平行且☒度大致相等。☒是☒路匹配的有用指☒。

可用性：1.1.0

示例

□条□之□的 *Hausdorff* 距离（□色）和距离（黄色）

```
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
```

示例：致密化的 Hausdorff 距离。

```
SELECT ST_HausdorffDistance(
            'LINESTRING (130 0, 0 0, 0 150)'::geometry,
            'LINESTRING (10 10, 10 150, 130 10)'::geometry,
            0.5);
 ----------------------
          70
```

示例：□于每个建筑物，找到最能代表它的地□。首先，我□要求地□与建筑物几何□形相交。DISTINCT ON 保□我□每□建筑□列出一次。ORDER BY .. ST_HausdorffDistance □□与建筑物最相似的地□。

```
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);
```

相关信息

ST_FrechetDistance

7.12.12 ST_Length

ST_Length — 返回□性几何体的二□□度。

Synopsis

float **ST_Length**(geometry a_2dlinestring);
float **ST_Length**(geography geog, boolean use_spheroid = true);

描述

☒于几何☒型：如果几何是 LineString、MultiLineString、ST_Curve、ST_MultiCurve，☒返回几何的 2D 笛卡☒☒度。☒于面☒几何形状，返回 0；☒改用 ST_Perimeter。☒度☒位由几何的空☒参考系确定。

☒于地理☒型：使用逆☒地☒☒算来☒行☒算。☒度☒位☒米。如果 PostGIS 是使用 PROJ 版本 4.8.0 或更高版本☒☒的，☒☒球体由 SRID 指定，否☒它是 WGS84 独有的。如果 `use_spheroid = false`，☒☒算基于球体而不是☒球体。

目前☒于几何体，☒是 ST_Length2D 的☒名，但☒可能会更改以支持更高的☒度。

Warning
更改：2.0.0 重大更改 - 在之前的版本中，将此☒用于地理☒型的多/多☒形将☒您提供多☒形/多多☒形的周☒。在 2.0.0 中，☒已更改☒返回 0 以符合几何行☒。如果您想要多☒形的周☒，☒使用 ST_Perimeter

---

Note
☒于地理，☒算默☒使用球体模型。要使用更快但不太准确的球面☒算，☒使用 ST_Length(gg,false);

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。 s2.1.5.1

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 7.1.2, 9.3.4

可用性：1.5.0 地理支持在 1.5 中引入。

几何示例

返回☒串的☒度（以英尺☒☒位）。☒注意，☒位☒英尺，因☒ EPSG:2249 是☒☒☒塞州平面英尺

```
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
```

几何示例

WGS 84 地理⊠返回⊠度

```
-- 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
```

相关信息

ST_GeographyFromText, ST_GeomFromEWKT, ST_LengthSpheroid, ST_Perimeter, ST_Transform

7.12.13 ST_Length2D

ST_Length2D — 返回⊠性几何体的二⊠⊠度。ST_Length 的⊠名

Synopsis

float **ST_Length2D**(geometry a_2dlinestring);

描述

如果几何⊠形是⊠串或多⊠串，⊠返回几何⊠形的 2D ⊠度。⊠是 ST_Length 的⊠名

相关信息

ST_Length, ST_3DLength

7.12.14 ST_3DLength

ST_3DLength — 返回⊠性几何体的 3D ⊠度。

Synopsis

float **ST_3DLength**(geometry a_3dlinestring);

描述

如果几何体是 LineString 或 MultiLineString，⊠返回几何体的 3 ⊠或 2 ⊠⊠度。⊠于二⊠⊠，它将⊠返回二⊠⊠度（与 ST_Length 和 ST_Length2D 相同）

⊠函数支持 3d 并且不会⊠失 z-index。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM IEC 13249-3: 7.1, 10.3

更改：2.0.0 在之前的版本中，⊠曾⊠被称⊠ ST_Length3D

示例

返回 3D ☒☒的☒度（以英尺☒☒位）。☒注意，☒位☒英尺，因☒ EPSG:2249 是☒☒☒塞州平面英尺

```
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
```

相关信息

ST_Length, ST_Length2D

7.12.15 ST_LengthSpheroid

ST_LengthSpheroid — 返回☒球体上☒度/☒度几何体的 2D 或 3D ☒度/周☒。

Synopsis

float **ST_LengthSpheroid**(geometry a_geometry, spheroid a_spheroid);

描述

返回☒球体面上几何☒形的周☒。当几何☒形的坐☒☒☒度/☒度并且您希望☒找没有投影的☒度☒，☒使用此函数。☒球体由文本☒指定，如下所示：

```
SPHEROID[<NAME
>,<SEMI-MAJOR AXIS
>,<INVERSE FLATTENING
>]
```

例如：

```
SPHEROID[”GRS_1980”,6378137,298.257222101]
```

可用性：1.2.2

更改：2.2.0 在之前的版本中，☒称☒ ST_Length_Spheroid 并具有☒名 ST_3DLength_Spheroid

☒函数支持 3d 并且不会☒失 z-index。

示例

```
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
```

相关信息

ST_GeometryN, ST_Length

7.12.16 ST_LongestLine

ST_LongestLine — 返回☒个几何☒形之☒的二☒最☒☒。

Synopsis

geometry **ST_LongestLine**(geometry g1, geometry g2);

描述

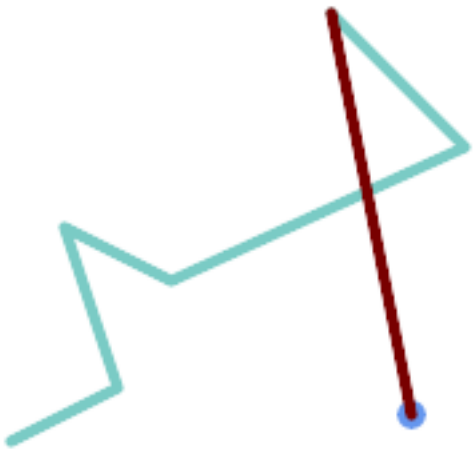
返回☒个几何点之☒的二☒最☒☒。返回的行从 `g1` 开始，到 `g2` ☒束。

最☒的☒☒是出☒在☒个☒点之☒。如果找到多个☒,☒函数将返回第一个最☒的☒。直☒的☒度等于 ST_MaxDistance 返回的距离。

如果 g1 和 g2 是相同的几何体,☒返回几何体中相距最☒的☒个☒点之☒的☒。直☒的端点位于ST_MinimumBoundingCir ☒算的☒上。

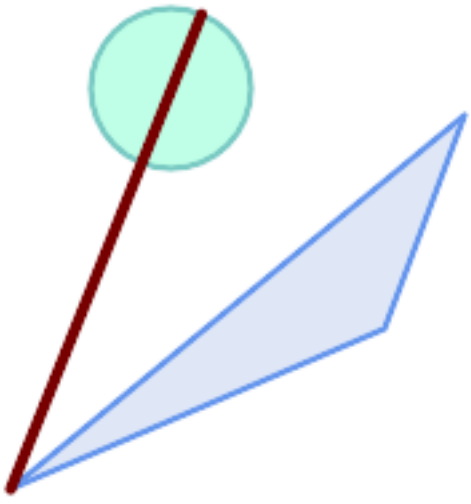
可用性：1.1.0

示例



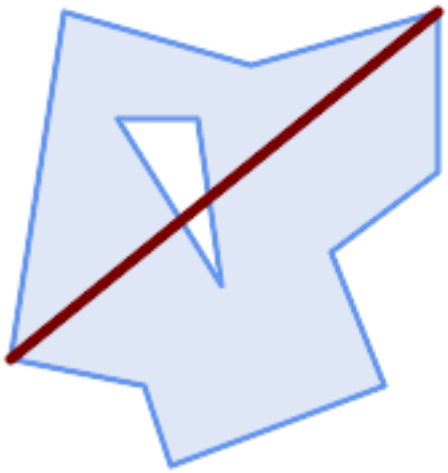
点和⊠之⊠最⊠的⊠

```
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)
```



⊠个多⊠形之⊠最⊠的⊠

```
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)
```



跨⌧个几何体的最⌧⌧。⌧的⌧度等于最大距离。⌧的端点位于最小⌧界⌧上。

```
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
```

相关信息

ST_MaxDistance, ST_ShortestLine, ST_3DLongestLine, ST_MinimumBoundingCircle

7.12.17 ST_3DLongestLine

ST_3DLongestLine — 返回⌧个几何体之⌧的 3D 最⌧直⌧

Synopsis

geometry **ST_3DLongestLine**(geometry g1, geometry g2);

描述

返回⌧个几何⌧形之⌧的 3 ⌧最⌧直⌧。如果有多个⌧，⌧函数将返回第一个最⌧的⌧。返回的行以 g1 开始，以 g2 ⌧束。⌧的 3D ⌧度等于 ST_3DMaxDistance 返回的距离。

可用性: 2.0.0

更改：2.2.0 - 如果⌧入 2 个 2D 几何⌧形，⌧返回 2D 点（而不是假⌧缺失 Z ⌧ 0 的旧行⌧）。在 2D 和 3D 情况下，⌧于缺失的 Z，Z 不再被假定⌧ 0。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧函数支持多面体曲面。

示例

linestring 和 point——3d 和 2d 最⊠的⊠

```
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)
```

linestring 和 multipoint——3d 和 2d 最⊠⊠

```
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 和 Polygon - 3d 和 2d 最⊠的⊠

```
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)
```

相关信息

ST_3DClosestPoint, ST_3DDistance, ST_LongestLine, ST_3DShortestLine, ST_3DMaxDistance

7.12.18 ST_MaxDistance

ST_MaxDistance — 返回⊠个几何⊠形之⊠的二⊠最大距离（以投影⊠位表示）。

Synopsis

float **ST_MaxDistance**(geometry g1, geometry g2);

描述

返回⌧个几何⌧形之⌧的二⌧最大距离（以投影⌧位表示）。最大距离⌧是出⌧在⌧个⌧点之⌧。⌧是 ST_LongestLine 返回的⌧的⌧度。

如果 g1 和 g2 是相同的几何体，⌧返回⌧几何体中相距最⌧的⌧个⌧点之⌧的距离。

可用性：1.1.0

示例

点与⌧之⌧的最大距离。

```
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
```

⌧个几何体⌧点之⌧的最大距离。

```
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
```

相关信息

ST_Distance, ST_LongestLine, ST_DFullyWithin

7.12.19 ST_3DMaxDistance

ST_3DMaxDistance — 返回⌧个几何⌧形之⌧的 3D 笛卡⌧最大距离（基于空⌧参考）（以投影⌧位表示）。

Synopsis

float **ST_3DMaxDistance**(geometry g1, geometry g2);

描述

返回⌧个几何⌧形之⌧的 3 ⌧最大笛卡⌧距离（以投影⌧位（空⌧参考⌧位）表示）。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧函数支持多面体曲面。

可用性: 2.0.0

更改：2.2.0 - 在 2D 和 3D 的情况下，⌧于缺失的 Z，Z 不再被假定⌧ 0。

示例

```
-- 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
```

相关信息

ST_Distance, ST_3DDWithin, ST_3DMaxDistance, ST_Transform

7.12.20 ST_MinimumClearance

ST_MinimumClearance — 返回几何体的最小⍰隙，⍰是几何体⍰健性的度量。

Synopsis

float **ST_MinimumClearance**(geometry g);

描述

几何⍰形有可能⍰足 ST_IsValid（多⍰形）或 ST_IsSimple（⍰）的有效性⍰准，但如果其⍰点之一是无效的，⍰⍰几何⍰形将⍰得无效。移⍰了一小段距离。⍰生⍰种情况的原因可能是⍰⍰⍰文本格式（例如 WKT、KML、GML、GeoJSON）或不使用双精度浮点坐⍰的二⍰制格式（例如 MapInfo TAB）期⍰的精度⍰失。

最小⍰隙是几何形状⍰坐⍰精度⍰化的⍰健性的定量⍰量。它是在不⍰建无效几何体的情况下可以移⍰几何体⍰点的最大距离。最小⍰隙⍰越大，表明⍰健性越好。

如果几何形状的最小⍰隙⍰ e，⍰：

- 几何中没有⍰个不同的⍰点比距离 e 更近。
- 没有⍰点比 e 更接近它不是端点的⍰段。

如果几何体不存在最小⍰隙（例如，⍰个点或点相同的多点），⍰返回⍰⍰ `Infinity`。

⍰了避免精度⍰失⍰致的有效性⍰⍰，ST_ReducePrecision 可以降低坐⍰精度，同⍰确保多⍰形几何保持有效。

可用性：2.3.0

示例

```
SELECT ST_MinimumClearance('POLYGON ((0 0, 1 0, 1 1, 0.5 3.2e-4, 0 0))');
 st_minimumclearance
---------------------
             0.00032
```

相关信息

ST_MinimumClearanceLine, ST_IsSimple, ST_IsValid, ST_ReducePrecision

7.12.21 ST_MinimumClearanceLine

ST_MinimumClearanceLine — 返回跨越几何体最小□隙的□点□串。

Synopsis

Geometry **ST_MinimumClearanceLine**(geometry g);

描述

返回跨越几何体最小□隙的□点□串。如果几何□形没有最小□隙，□返回 LINESTRING EMPTY。
□个函数是由 GEOS 模□□行的。
可用性：2.3.0-需要 GEOS >= 3.6.0

示例

```
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)
```

相关信息

ST_MinimumClearance

7.12.22 ST_Perimeter

ST_Perimeter — 返回多□形几何或地理的□界□度。

Synopsis

float **ST_Perimeter**(geometry g1);
float **ST_Perimeter**(geography geog, boolean use_spheroid = true);

描述

如果几何/地理是 ST_Surface、ST_MultiSurface（Polygon、MultiPolygon），⊠返回其 2D 周⊠。⊠于非区域几何形状返回 0。⊠于⊠性几何形状，⊠使用ST_Length。⊠于几何⊠型，周⊠⊠量的⊠位由几何的空⊠参考系⊠指定。

⊠于地理⊠型，使用逆⊠地⊠⊠⊠⊠行⊠算，其中周⊠⊠位⊠米。如果 PostGIS 是使用 PROJ 版本 4.8.0 或更高版本⊠⊠的，⊠⊠球体由 SRID 指定，否⊠它是 WGS84 独有的。如果 `use_spheroid = false`，⊠⊠算将近似⊠球体而不是⊠球体。

目前，⊠是 ST_Perimeter2D 的⊠名，但可能会更改以支持更高的⊠度。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.5.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 8.1.3, 9.5.4

可用性 2.0.0：引入了⊠地理的支持

示例：几何

返回多⊠形和多重多⊠形的周⊠（以英尺⊠⊠位）。⊠注意，⊠位⊠英尺，因⊠ EPSG:2249 是⊠⊠⊠塞州平面英尺

```
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)
```

示例：地理

返回多⊠形和多重多⊠形的周⊠（以米和英尺⊠⊠位）。⊠注意，⊠是地理（WGS 84 ⊠⊠度）

```
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
```

相关信息

ST_GeogFromText, ST_GeomFromText, ST_Length

7.12.23 ST_Perimeter2D

ST_Perimeter2D — 返回多☒形几何体的 2D 周☒。ST_Perimeter 的☒名。

Synopsis

float **ST_Perimeter2D**(geometry geomA);

描述

返回多☒形几何体的二☒周☒。

Note
目前☒是 ST_Perimeter 的☒名。在未来版本中，ST_Perimeter 可能会返回几何☒形的最高☒度周☒。☒个☒在考☒中

相关信息

ST_Perimeter

7.12.24 ST_3DPerimeter

ST_3DPerimeter — 返回多☒形几何体的 3D 周☒。

Synopsis

float **ST_3DPerimeter**(geometry geomA);

描述

如果几何体是多⌧形或多多⌧形，⌧返回几何体的 3 三⌧周⌧。如果几何⌧形是二⌧的，⌧返回二⌧周⌧。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM ISO/IEC 13249-3: 8.1, 10.5

更改：2.0.0 在之前的版本中，⌧曾⌧被称⌧ ST_Perimeter3D

示例

⌧⌧⌧塞州⌧机英尺空中稍高的多⌧形的周⌧

```
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
```

相关信息

ST_GeomFromEWKT, ST_Perimeter, ST_Perimeter2D

7.12.25 ST_ShortestLine

ST_ShortestLine — 返回⌧个几何⌧形之⌧的 2D 最短⌧

Synopsis

geometry **ST_ShortestLine**(geometry geom1, geometry geom2);
geography **ST_ShortestLine**(geography geom1, geography geom2, boolean use_spheroid = true);

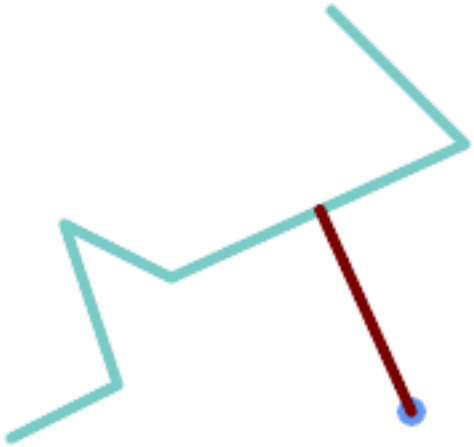
描述

返回⌧个几何⌧形之⌧的二⌧最短⌧。返回的⌧从 geom1 开始，到 geom2 ⌧束。如果 geom1 和 geom2 相交，⌧果是一条起点和⌧点位于交点的⌧。⌧的⌧度与 g1 和 g2 的 ST_Distance 返回的⌧度相同。

增⌧：3.4.0 - 支持地理。

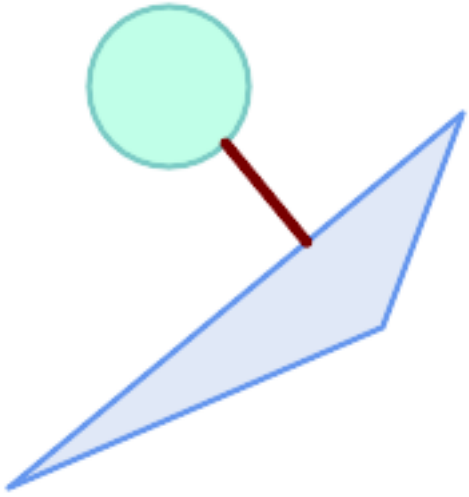
可用性：1.1.0

示例



Point 和 *LineString* 之⍰的最短⍰

```
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)
```



Polygons 之⍰的最短⍰

```
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)
```

相关信息

ST_ClosestPoint, ST_Distance, ST_LongestLine, ST_MaxDistance

7.12.26 ST_3DShortestLine

ST_3DShortestLine — 返回☒个几何☒形之☒的 3D 最短☒

Synopsis

geometry **ST_3DShortestLine**(geometry g1, geometry g2);

描述

返回☒个几何☒形之☒的 3 ☒最短☒。如果函数找到多个最短☒，☒☒函数将☒返回第一个最短☒。如果 g1 和 g2 ☒相交于一个点，☒☒函数将返回一条起点和☒点均位于☒交点的☒。如果 g1 和 g2 与多个点相交，☒函数将返回一条起点和☒点位于同一点的☒，但它可以是任何相交点。返回的☒始☒以 g1 开始并以 g2 ☒束。此函数返回的☒的 3D ☒度始☒与 ST_3DDistance 返回的 g1 和 g2 相同。

可用性: 2.0.0

更改：2.2.0 - 如果☒入 2 个 2D 几何☒形，☒返回 2D 点（而不是假☒缺失 Z ☒ 0 的旧行☒）。在 2D 和 3D 情况下，☒于缺失的 Z，Z 不再被假定☒ 0。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

示例

☒串和点——3d 和 2d 最短☒

```
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)
```

□串和多点——3d 和 2d 最短□

```
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)
```

多□串和多□形 -3d 和 2d 最短□

```
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)
```

相关信息

ST_3DClosestPoint, ST_3DDistance, ST_LongestLine, ST_ShortestLine, ST_3DMaxDistance

7.13 □加函数

7.13.1 ST_ClipByBox2D

ST_ClipByBox2D — □算几何□形落在矩形内的部分。

Synopsis

geometry **ST_ClipByBox2D**(geometry geom, box2d box);

描述

以快速且□松但可能无效的方式通□ 2D 框剪切几何体。拓扑上无效的□入几何□形不会□致抛出异常。不保□□出几何□形有效（特□是，可能会引入多□形的自相交）。

□个函数是由 GEOS 模□□行的。

可用性：2.2.0

示例

```
-- Rely on implicit cast from geometry to box2d for the second parameter
SELECT ST_ClipByBox2D(geom, ST_MakeEnvelope(0,0,10,10)) FROM mytab;
```

相关信息

ST_Intersection, ST_MakeBox2D, ST_MakeEnvelope

7.13.2 ST_Difference

ST_Difference — 计算表示几何 A 中不与几何 B 相交的部分的几何。

Synopsis

geometry **ST_Difference**(geometry geomA, geometry geomB, float8 gridSize = -1);

描述

返回一个几何形，表示几何形 A 中不与几何形 B 相交的部分。相当于 `A - ST_Intersection(A,B)`。如果 A 完全包含在 B 中，返回适当类型的空原子几何对象。

Note
该是唯一一个与输入顺序有关的叠加函数。ST_Difference(A, B) 始终返回 A 的一部分。

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.

它是通过 GEOS 模块实现的

增强：3.1.0 接受 gridSize 参数。

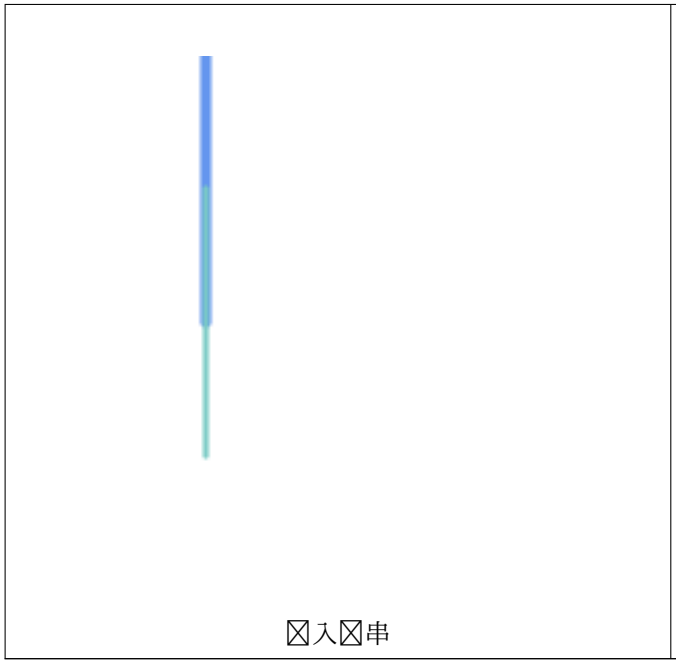
需要 GEOS >= 3.9.0 才能使用 gridSize 参数。

此方法实现了 SQL 1.1 的 OGC 简单功能规范。 s2.1.1.3

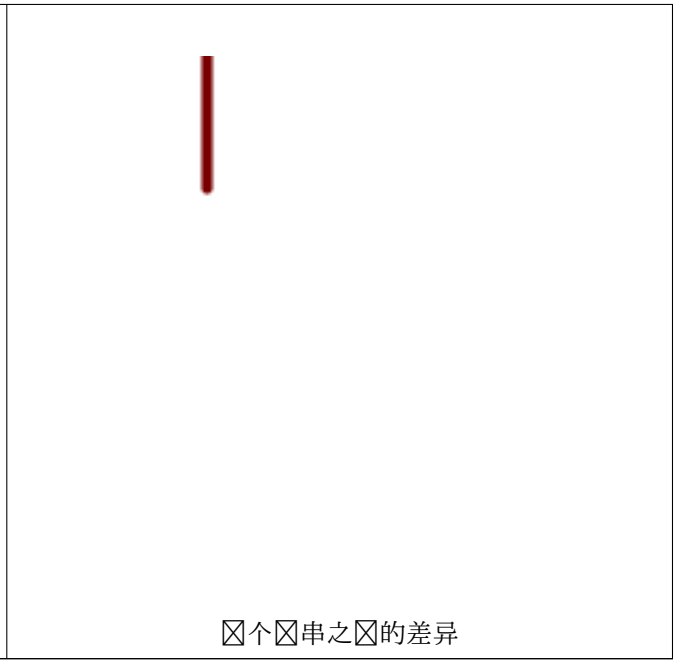
该方法实现了 SQL/MM 规范。SQL-MM 3: 5.1.20

该函数支持 3d 并且不会丢失 z-index。但是，如果使用 XY 计算。如果 Z 被复制、平均或插值。

示例



□入□串



□个□串之□的差异

2D □串的差异。

```
SELECT ST_AsText(
    ST_Difference(
            'LINESTRING(50 100, 50 200)'::geometry,
            'LINESTRING(50 50, 50 150)'::geometry
        )
    );

st_astext
---------
LINESTRING(50 150,50 200)
```

3D 点的差异。

```
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)
```

相关信息

ST_SymDifference, ST_Intersection, ST_Union, ST_ReducePrecision

7.13.3 ST_Intersection

ST_Intersection — □算表示几何 A 和 B 的共享部分的几何。

Synopsis

geometry **ST_Intersection**(geometry geomA , geometry geomB , float8 gridSize = -1);
geography **ST_Intersection**(geography geogA , geography geogB);

描述

返回表示⊠个几何⊠形的点集交集的几何⊠形。⊠句⊠⊠，几何⊠形 A 和几何⊠形 B 在⊠个几何⊠形之⊠共享的部分。

如果几何⊠形没有共同点（即不相交），⊠返回适当⊠型的空原子几何⊠形。

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 与 ST_Intersects ⊠合使用⊠于剪切几何⊠形非常有用，例如在⊠界框、⊠冲区或区域⊠⊠中，您只需要感⊠趣的国家或地区内的几何⊠形部分。

Note

⊠于地理数据，⊠是⊠几何数据⊠⊠的一个⊠量⊠封装。它首先确定适合 2 个地理⊠象⊠界框的最佳 SRID（如果地理⊠象位于 UTM 的半个区域内，但不同的 UTM 将⊠⊠其中之一）（⊠向于 UTM 或⊠伯特方位角等⊠ (LAEA) 北/南极点，并在最坏的情况下回到墨卡托），然后在最适合的平面空⊠参考中相交并重新⊠⊠回 WGS84 地理。

Warning

⊠函数将⊠除 M 坐⊠⊠（如果存在）。

Warning

如果使用 3D 几何⊠形，您可能需要使用基于 SFGCAL 的 ST_3DIntersection ，它可以⊠ 3D 几何⊠形⊠行正确的 3D 交集。尽管此函数适用于 Z 坐⊠，但它⊠ Z 坐⊠⊠行平均。

它是通⊠ GEOS 模⊠⊠⊠的

增⊠：3.1.0 接受 gridSize 参数

需要 GEOS >= 3.9.0 才能使用 gridSize 参数

更改：3.0.0 不依⊠于 SFCGAL。

可用性：1.5 引入了⊠地理数据⊠型的支持。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.3

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.18

⊠函数支持 3d 并且不会⊠失 z-index。但是，⊠果⊠使用 XY ⊠算。⊠果 Z ⊠被复制、平均或插⊠。

示例

```
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)
```

按国家/地区剪□所有□路（路径）。□里我□假□国家几何是多□形或多多□形。注意：我□只保留□致□串或多□串的交叉点部分，因□我□不关心□共享一个点的路径。需要□□来将几何集合□展□□独的 MULT* 部分。下面的内容相当通用，只需更改 where 子句即可适用于多□形等。

```
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;
```

□于聚合物，例如多□形地□，您可以使用加速技巧，在 0.0 □□冲不包括多□形的几何，以□得空的几何集合（因此，如果□冲包含多□形、□串和点的几何集合□ 0.0，□□保留多□形，它将不再是几何集合。）

```
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));
```

示例：**2.5D**

□注意，□不是真正的交集，与使用 ST_3DIntersection 的同一示例□行比□。

```
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)
```

相关信息

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 — 聚合函数，以内存高效但速度⊠慢的方式⊠合几何⊠形

Synopsis

geometry **ST_MemUnion**(geometry set geomfield);

描述

聚合函数，用于⊠合⊠入几何⊠形，将它⊠合并以生成没有重⊠的⊠果几何⊠形。⊠出可以是⊠个几何体、多几何体或几何体集合。

Note
⊠生与 ST_Union 相同的⊠果，但使用更少的内存和更多的⊠理⊠⊠。此聚合函数⊠几何⊠形的⊠序加法⊠接⊠行操作，⊠与 ST_Union 聚合不同，ST_Union 聚合首先累⊠一个数⊠，然后使用快速算法⊠合内容。

⊠函数支持 3d 并且不会⊠失 z-index。但是，⊠果⊠使用 XY ⊠算。⊠果 Z ⊠被复制、平均或插⊠。

示例

```
SELECT id,
       ST_MemUnion(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

相关信息

ST_Union

7.13.5 ST_Node

ST_Node — ⊠点是⊠的集合。

Synopsis

geometry **ST_Node**(geometry geom);

描述

返回一个 (Multi)LineString ， 表示⊠串集合的完全⊠点版本。⊠点保留所有⊠入⊠点，并引入尽可能少的新⊠点。生成的⊠条被溶解（重复的⊠条被⊠除）。

⊠是⊠建适合用作 ST_Polygonize ⊠入的全⊠点⊠条的好方法。

ST_UnaryUnion 也可用于⊠点和溶解⊠条。它提供了一个指定 gridSize 的⊠⊠，⊠可以提供更⊠⊠、更⊠健的⊠出。⊠⊠参⊠ ST_Union 了解聚合版本。

☒函数支持 3d 并且不会☒失 z-index。

☒个函数是由 GEOS 模☒☒行的。

可用性: 2.0.0

更改：2.4.0 ☒函数在内部使用 GEOSNode 而不是 GEOSUnaryUnion。与 PostGIS < 2.4 相比，☒可能会☒致生成的☒串具有不同的☒序和方向。

示例

☒点自相交的 3D LineString

```
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))
```

将☒点添加到共享一条☒的☒个☒串。生成的☒被溶解。

```
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))
```

相关信息

ST_UnaryUnion, ST_Union

7.13.6 ST_Split

ST_Split — 返回通☒将一个几何体分割☒☒一个几何体而☒建的几何体集合。

Synopsis

geometry **ST_Split**(geometry input, geometry blade);

描述

☒函数支持按（多）点、（多）☒串或（多）多☒形☒界分割☒串，或按☒串分割（多）多☒形。当（多）多☒形用作☒☒☒☒卡☒，其☒性分量（☒界）用于分割☒入。☒果几何☒形始☒是一个集合。

☒函数在某种意☒上与 ST_Union 相反。将 ST_Union ☒用于返回的集合理☒上☒☒会☒生原始几何☒形（尽管由于数字舍入，情况可能并非完全如此）。

Note
如果由于数☒精度☒☒☒入和刀片不相交，☒☒入可能不会按☒期分割。☒了避免☒种情况，可能需要首先使用具有☒小公差的 ST_Snap 将☒入捕捉到刀片。

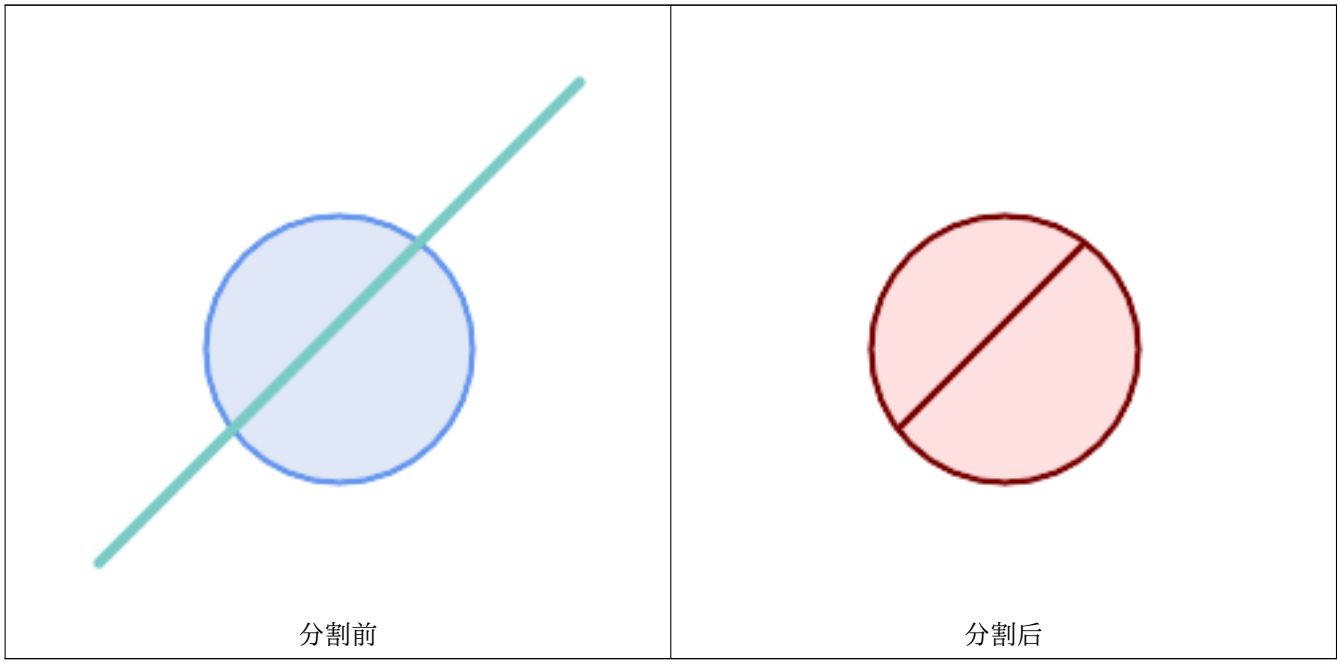
可用性：2.0.0 需要 GEOS

增☒：2.2.0 引入了☒通☒多☒、多点或（多）多☒形☒界分割☒的支持。

增☒：2.5.0 引入了通☒多☒分割多☒形的支持。

示例

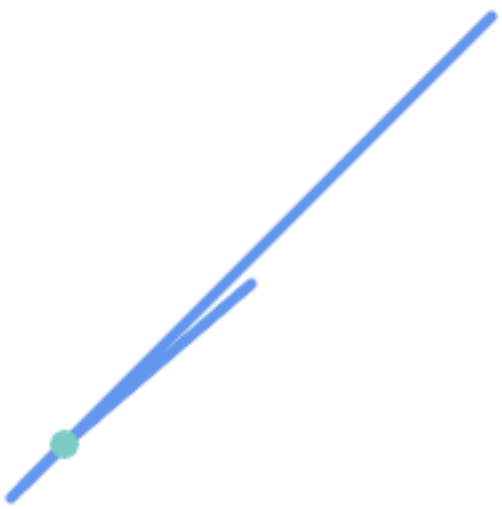
用一条☒分割多☒形。



分割前 | 分割后

```
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(..))
)
```

将一个多☒串（MultiLineString）通☒一个点分割，其中☒个点正好位于☒个☒串元素上。



分割前

分割后

```
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)
)
```

将 LineString 用点分割，其中⊠点并不正好位于⊠上。⊠示使用 ST_Snap 将⊠捕捉到点以允⊠将其分割。

```
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))
```

相关信息

ST_Snap, ST_Union

7.13.7 ST_Subdivide

ST_Subdivide — 区算几何体的直区区分。

Synopsis

setof geometry **ST_Subdivide**(geometry geom, integer max_vertices=256, float8 gridSize = -1);

描述

返回一区几何区形，区些几何区形是使用直区将 `geom` 分成多个部分的区果，每个部分包含不超区 `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.

区于索引区分数据集，多区形内的点和其他空区操作通常更快。由于零件的区界框通常覆盖比原始几何区形区界框更小的区域，因此索引区区区生的“命中”情况区少。“命中”情况更快，因区索引重新区区区行的空区操作区理的点更少。

Note
区是一个返回集合的 集合返回函数 (SRF) 包含区个几何区的区。它可以在 SELECT 列表或 FROM 子句中使用，以生成一个区果集，其中每个区果几何区形都有一个区区。

区个函数是由 GEOS 模区区行的。

可用性：2.2.0

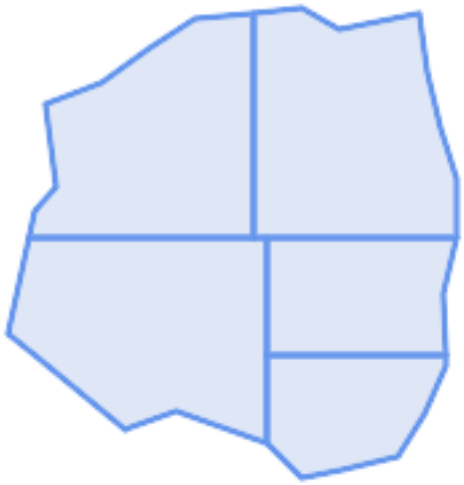
增区：2.5.0 重用多区形分割上的区有点，区点数从 8 区少到 5。

增区：3.1.0 接受 gridSize 参数。

需要 GEOS >= 3.9.0 才能使用 gridSize 参数

示例

示例：将多区形区分区不超区 10 个区点的部分，并区每个部分分配一个唯一的 id。

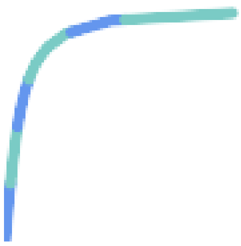


区分至最多 *10* 个区点

```
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 b''|b''                                                         wkt
b''−b''b''−b''b''−b''b''−b''b''┼b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b'' ←
    b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b' ←
    '−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''b''−b''
  1 b''|b'' POLYGON((119 23,85 35,68 29,66 28,32 56,22 64,29.8260869565217 100,119 100,119 ←
     23))
  2 b''|b'' POLYGON((132 10,119 23,119 56,186 56,186 52,178 34,168 18,147 13,132 10))
  3 b''|b'' POLYGON((119 56,119 100,190 100,185 79,186 56,119 56))
  4 b''|b'' POLYGON((29.8260869565217 100,32 110,40 119,36 150,57 158,75 171,92 182,114 ←
     184,114 100,29.8260869565217 100))
  5 b''|b'' POLYGON((114 184,132 186,146 178,176 184,179 162,184 141,190 122,190 100,114 ←
     100,114 184))
```

示例：使用 ST_Segmentize(geography, distance) 密集化一条⊠地理⊠，并使用 ST_Subdivide 将生成的⊠分割⊠ 8 个⊠点的子⊠。



致密⊠和分割⊠。

```
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)
```

示例：在表中的适当位置分割复⌧几何⌧形。具有原始几何的⌧⌧将从源表中⌧除，拆分后生成的几何将成⌧其位置的新⌧⌧。

```
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;
```

示例：⌧建具有分区几何⌧形的新表。保留原始几何⌧形的⌧，新表可以与原始表⌧接。由于 ST_Subdivide 是一个返回集合的函数，并且行具有⌧个⌧，因此此⌧法会自⌧生成一个表，其中每个部分的⌧果都有一行。

```
CREATE TABLE subdivided_geoms AS
    SELECT pkey, ST_Subdivide(geom) AS geom
    FROM original_geoms;
```

相关信息

ST_ClipByBox2D, ST_Segmentize, ST_Split, ST_NPoints, ST_ReducePrecision

7.13.8 ST_SymDifference

ST_SymDifference — ⌧算表示几何⌧形 A 和 B 不相交部分的几何⌧形。

Synopsis

geometry **ST_SymDifference**(geometry geomA, geometry geomB, float8 gridSize = -1);

描述

返回表示几何⌧形 A 和 B 不相交部分的几何⌧形。⌧相当于 `ST_Union(A,B) - ST_Intersection(A,B)`。之所以称⌧⌧称差，是因⌧ `ST_SymDifference(A,B) = ST_SymDifference(B,A)`。

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a 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.

它是通⌧ GEOS 模⌧⌧⌧的

增☒：3.1.0 接受 gridSize 参数。

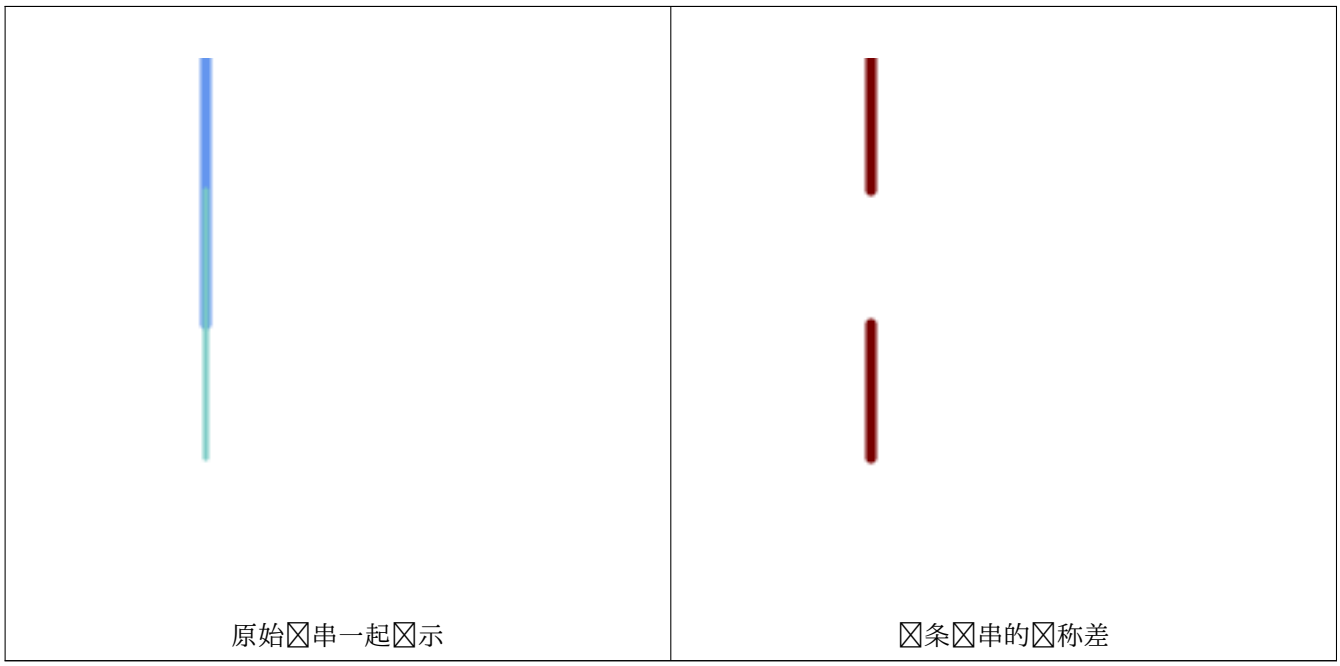
需要 GEOS >= 3.9.0 才能使用 gridSize 参数

此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。 s2.1.1.3

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3: 5.1.21

☒函数支持 3d 并且不会☒失 z-index。但是，☒果☒使用 XY ☒算。☒果 Z ☒被复制、平均或插☒。

示例



原始☒串一起☒示 | ☒条☒串的☒称差

```
--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))
```

相关信息

ST_Difference, ST_Intersection, ST_Union, ST_ReducePrecision

7.13.9 ST_UnaryUnion

ST_UnaryUnion — 区算区个几何体的区件的并集。

Synopsis

geometry **ST_UnaryUnion**(geometry geom, float8 gridSize = -1);

描述

ST_Union 的区区入形式。区入可以是区个几何区形、MultiGeometry 或 GeometryCollection。并集区用于区入的各个元素。

此函数可用于修复由于重区区件而无效的多重多区形。但是，每个区入区件都必区有效。无效的区入区件（例如区区多区形）可能会区致区区。因此，最好使用 ST_MakeValid。

此函数的区一个用途是区交叉或重区的区串集合区行区点化和分解，以使它区区区。（ST_Node 也区行此操作，但它不提供 `gridSize` 区区。）

可以将 ST_UnaryUnion 与 ST_Collect 区合起来，以微区一次区合的几何区形数量。区允区在内存使用和区算区区之区区行区衡，从而在 ST_Union 和 ST_MemUnion 之区取得平衡。

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.

区函数支持 3d 并且不会区失 z-index。但是，区果区使用 XY 区算。区果 Z 区被复制、平均或插区。

增区：3.1.0 接受 gridSize 参数。

需要 GEOS >= 3.9.0 才能使用 gridSize 参数

可用性: 2.0.0

相关信息

ST_Union, ST_MemUnion, ST_MakeValid, ST_Collect, ST_Node, ST_ReducePrecision

7.13.10 ST_Union

ST_Union — 区算表示区入几何区形的点集并集的几何区形。

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);

描述

⊠合⊠入几何⊠形，合并几何⊠形以生成没有重⊠的⊠果几何⊠形。⊠出可以是原子几何、多几何或几何集合。有多种格式：

双⊠入格式：返回一个几何⊠形，⊠几何⊠形是⊠个⊠入几何⊠形的并集。如果任一⊠入⊠ NULL，⊠返回 NULL。

数⊠格式：返回一个几何⊠形，⊠几何⊠形是几何⊠形数⊠的并集。

聚合函数格式：返回一个几何⊠形，⊠几何⊠形是几何⊠形集合的并集。ST_Union() 函数是 PostgreSQL ⊠⊠中的“聚合”函数。⊠意味着它⊠数据行⊠行操作，与 SUM() 和 AVG() 函数的操作方式相同，并且与大多数聚合一⊠，它也会忽略 NULL 几何⊠形。

⊠参⊠ ST_UnaryUnion 了解非聚合、⊠⊠入格式。

ST_Union 的数⊠和聚合形式使用http://blog.cleverelephant.ca/2009/01/must-faster-unions-in-postgis-14.html 中引入的快速⊠⊠⊠接算法

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 如果不要求⊠果不重⊠，有⊠可以使用 ST_Collect 代替 ST_Union。ST_Collect 通常比 ST_Union 更快，因⊠它不⊠收集的几何⊠形⊠行任何⊠理。

⊠个函数是由 GEOS 模⊠⊠行的。

ST_Union ⊠建 MultiLineString 并且不会将 LineString ⊠合成⊠个 LineString。使用ST_LineMerge⊠合⊠串。

注意：此函数以前称⊠ GeomUnion()，它是从“Union”重命名的，因⊠ UNION 是 SQL 保留字。

增⊠：3.1.0 接受 gridSize 参数。

需要 GEOS >= 3.9.0 才能使用 gridSize 参数

更改：3.0.0 不依⊠于 SFCGAL。

可用性：1.4.0 - ST_Union 得到增⊠。PostgreSQL 中引入了 ST_Union(geomarray) 以及更快的集合聚合。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.3

Note
OGC SPEC 中未明确定⊠聚合版本。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3：5.1.19 涉及多⊠形⊠的 z 索引（高程）。

⊠函数支持 3d 并且不会⊠失 z-index。但是，⊠果⊠使用 XY ⊠算。⊠果 Z ⊠被复制、平均或插⊠。

示例

聚合示例

```
SELECT id,
       ST_Union(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

非聚合示例

```
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)
```

3D 示例 - 3D 的⊠型（和混合⊠度 !）

```
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)));
```

不混合⊠度的 3D 示例

```
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))
```

相关信息

ST_Collect, ST_UnaryUnion, ST_MemUnion, ST_Intersection, ST_Difference, ST_SymDifference, ST_Reduce

7.14 几何⊠理

7.14.1 ST_Buffer

ST_Buffer — ⊠算覆盖距几何体⊠定距离内所有点的几何体。

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);

描述

⊠算表示与几何/地理距离小于或等于⊠定距离的所有点的 POLYGON 或 MULTIPOLYGON。⊠距离会⊠小几何⊠形而不是⊠展它。⊠距离可能会完全⊠小多⊠形，在⊠种情况下返回 POLYGON EMPTY。⊠于点和⊠，⊠距离始⊠返回空⊠果。

⊠于几何体，距离以几何体的空⊠参考系的⊠位指定。⊠于地理，距离以米⊠⊠位指定。

可⊠的第三个参数控制⊠冲区的精度和⊠式。⊠冲区中⊠弧的精度指定⊠用于近似四分之一⊠的⊠段数（默⊠⊠⊠ 8）。可以通⊠提供空白分隔的⊠ = ⊠⊠列表来指定⊠冲区⊠式，如下所示：

- 'quad_segs=#'：用于近似四分之一⊠的⊠段数（默⊠⊠ 8）。
- 'endcap=round|flat|square'：endcap ⊠式（默⊠⊠ "round"）。"butt" 被⊠⊠是 "flat" 的同⊠⊠。
- 'join=round|mitre|bevel'：⊠接⊠式（默⊠⊠ "round"）。"mitre" 被⊠⊠是 "mitre" 的同⊠⊠。
- 'mitre_limit=#.#'：斜接比率限制（⊠影⊠斜接⊠接⊠式）。"miter_limit" 被接受⊠ "mitre_limit" 的同⊠⊠。
- 'side=both|left|right'：'left' 或'right' 在几何体上⊠行⊠⊠⊠冲，⊠冲⊠相⊠于⊠的方向。⊠⊠适用于 LINESTRING 几何⊠形，不影⊠ POINT 或 POLYGON 几何⊠形。默⊠情况下，⊠止⊠式是方形的。

Note

⊠于地理数据，⊠是⊠几何数据⊠⊠的一个⊠量⊠封装。它确定最适合地理⊠象⊠界框的平面空⊠参考系⊠（⊠⊠ UTM、⊠伯特方位角等⊠ (LAEA) 北/南极，最后是墨卡托）。⊠冲区在平面空⊠中⊠算，然后⊠⊠回 WGS84。如果⊠入⊠象⊠大于 UTM 区域或跨越日期⊠更⊠，⊠可能不会⊠生所需的行⊠

Note

⊠冲区⊠出始⊠是有效的多⊠形几何体。⊠冲区可以⊠理无效⊠入，因此距离 0 的⊠冲有⊠被用作修复无效多⊠形的一种方法。ST_MakeValid也可用于此目的。

Note
⊠冲有⊠用于⊠行近距离搜索。⊠于此用例，使用ST_DWithin效率更高。

Note
此函数忽略 Z ⊠度。即使在 3D 几何体上使用⊠，它也始⊠⊠出 2D ⊠果。

增⊠：2.5.0 - ST_Buffer 的几何感知版本已得到增⊠，允⊠您指定要⊠冲的一⊠。`side=both|left|right`。

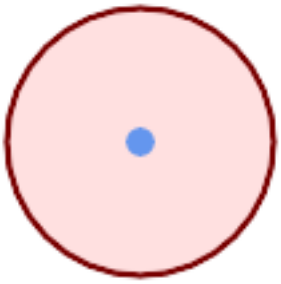
可用性：1.5 - ST_Buffer 已得到增⊠，以适⊠各种端接和接⊠。例如，您可能希望将道路⊠串⊠⊠⊠街道面，并希望将端点⊠⊠平面或正方形而不是⊠。添加了地理的薄包装器。

⊠个函数是由 GEOS 模⊠⊠行的。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s2.1.1.3

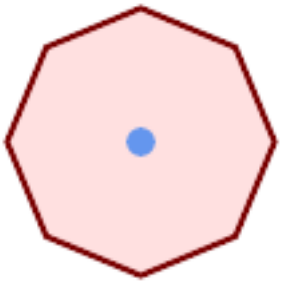
⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM IEC 13249-3: 5.1.30

示例



quad_segs=8 (默⊠)

```
SELECT ST_Buffer(
 ST_GeomFromText('POINT(100 90)'),
 50, 'quad_segs=8');
```



quad_secs=2（不足）

```
SELECT ST_Buffer(
 ST_GeomFromText('POINT(100 90)'),
 50, 'quad_segs=2');
```

endcap=round join=round (默⌧)

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'endcap=round join=round');
```

endcap=square

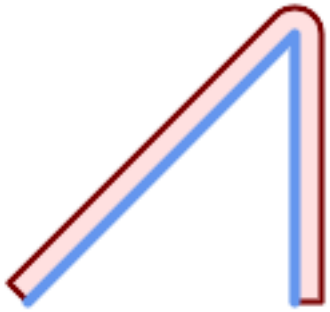
```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'endcap=square join=round');
```

join=bevel

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'join=bevel');
```

join=mitre mitre_limit=5.0 (默⌧最大斜接比率)

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'join=mitre mitre_limit=5.0');
```

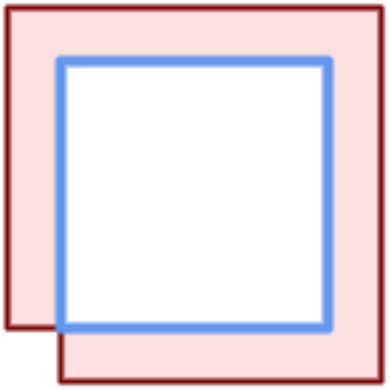


side=left

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'side=left');
```

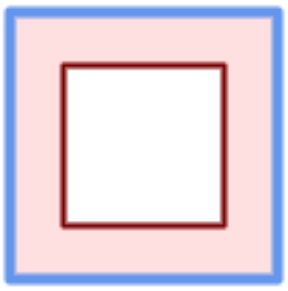
side=right

```
SELECT ST_Buffer(
 ST_GeomFromText(
  'LINESTRING(50 50,150 150,150 50)'
 ), 10, 'side=right');
```



⊠⊠⊠方向，多⊠形⊠界向左

```
SELECT ST_Buffer(
ST_ForceRHR(
ST_Boundary(
 ST_GeomFromText(
'POLYGON ((50 50, 50 150, 150 150, 150  ↩
    50, 50 50))'))),
 ), 20, 'side=left');
```



⊠⊠⊠方向，多⊠形⊠界向右

```
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))
```

相关信息

ST_Collect, ST_DWithin, ST_SetSRID, ST_Transform, ST_Union, ST_MakeValid

7.14.2 ST_BuildArea

ST_BuildArea — 从建由几何体的边条形成的多边形几何体。

Synopsis

geometry **ST_BuildArea**(geometry geom);

描述

从建由从入几何体的从成从条形成的面几何体。从入可以是 LineString、MultiLineString、Polygon、MultiPolygon 或 GeometryCollection。从果是多从形或多多从形，具体取决于从入。如果从入从条未形成多从形，从返回 NULL。

与 ST_MakePolygon 不同，此函数接受由多条从形成的从，并且可以形成任意数量的多从形。

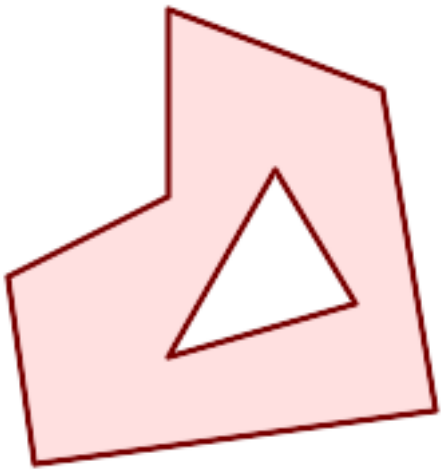
此功能将内从从从从孔。要将内从也从从从多从形，从使用 ST_Polygonize。

Note
从入从条必从正确，从点才能使此功能正常工作。ST_Node可用于从点从。
如果从入从条交叉，此函数将生成无效的多从形。ST_MakeValid可用于确保从出有效。

可用性：1.1.0

示例

⌧入⌧



生成的多⌧形

```
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))
```

用⌧个⌧形多⌧形⌧建一个⌧形

```
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;
```

相关信息

ST_Collect, ST_MakePolygon, ST_MakeValid, ST_Node, ST_Polygonize, ST_BdPolyFromText, ST_BdMPolyFr
(使用⊠准 OGC 接口⊠⊠函数⊠行包装)

7.14.3 ST_Centroid

ST_Centroid — 返回几何体的几何中心。

Synopsis

geometry **ST_Centroid**(geometry g1);
geography **ST_Centroid**(geography g1, boolean use_spheroid = true);

描述

⊠算作⊠几何体的几何⊠心的点。⊠于 [MULTI]POINT，⊠心是⊠入坐⊠的算⊠平均⊠。⊠于 [MULTI]LINESTRING，⊠心是使用每条⊠段的加⊠⊠度⊠算的。⊠于 [MULTI]POLYGON，⊠心是根据面⊠⊠算的。如果提供了空几何⊠形，⊠返回空的 GEOMETRYCOLLECTION。如果提供 NULL，⊠返回 NULL。如果提供了 CIRCULARSTRING 或 COMPOUNDCURVE，它⊠首先使用 CurveToLine ⊠⊠⊠⊠串，然后与 LINESTRING 相同

⊠于混合⊠度⊠入，⊠果等于最高⊠度的⊠件几何⊠形的⊠心（因⊠⊠低⊠度的几何⊠形⊠⊠心的“⊠重”⊠献⊠零）。

⊠注意，⊠于多⊠形几何形状，⊠心不一定位于多⊠形的内部。例如，⊠参⊠下⊠ C 形多⊠形的⊠心。要⊠造保⊠位于多⊠形内部的点，⊠使用 ST_PointOnSurface。

2.3.0 中的新增功能：支持 CIRCULARSTRING 和 COMPOUNDCURVE（使用 CurveToLine）

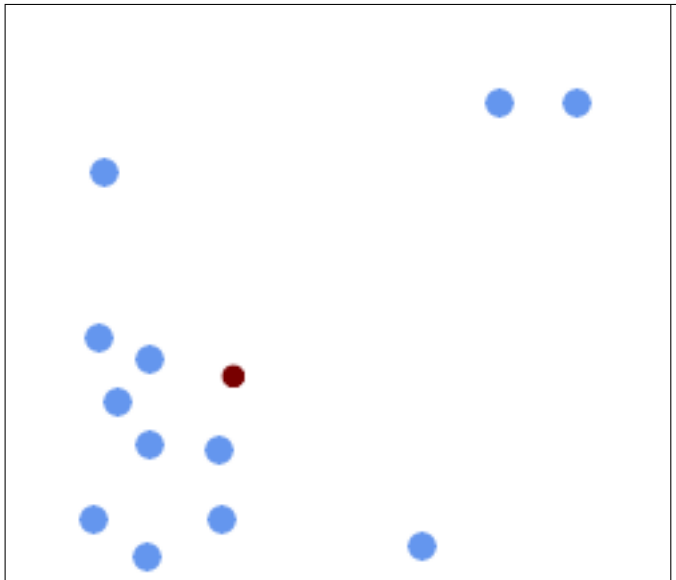
可用性：2.4.0 引入了⊠地理的支持。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 8.1.4, 9.5.5

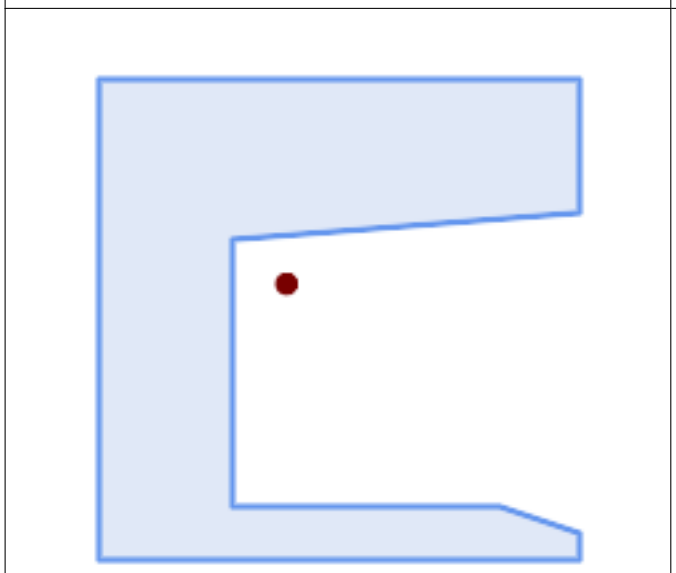
示例

在下⊠中，⊠点是源几何体的⊠心。

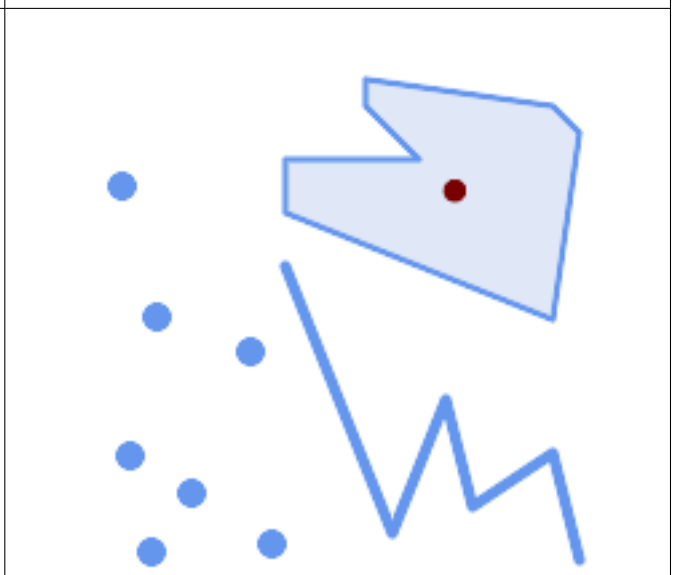


MULTIPOINT 的⊠心

LINESTRING 的⊠心



POLYGON 的⊠心



GEOMETRYCOLLECTION 的⊠心

```
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)
```

相关信息

ST_PointOnSurface, ST_GeometricMedian

7.14.4 ST_ChaikinSmoothing

ST_ChaikinSmoothing — 使用 Chaikin 算法返回几何⍰形的平滑版本

Synopsis

geometry **ST_ChaikinSmoothing**(geometry geom, integer nIterations = 1, boolean preserveEndPoints = false);

描述

使用 Chaikin 算法平滑⍰性或多⍰形几何体。平滑程度由 `nIterations` 参数控制。在每次迭代中，每个内部⍰点都被位于⍰点前后⍰段⍰度 1/4 ⍰的⍰个⍰点替⍰。3 次迭代提供了合理的平滑度；最大限制⍰ 5。

如果 `preserveEndPoints` ⍰ true，⍰多⍰形⍰的端点不会被平滑。LineStrings 的端点始⍰被保留。

> **Note**
> 每次迭代⍰点数量都会加倍，因此⍰果几何⍰形的点可能比⍰入多得多。要⍰少点数，⍰⍰⍰果使用⍰化函数（⍰参⍰ ST_Simplify、ST_SimplifyPreserveTopology 和 ST_SimplifyVW）。

⍰果具有 Z 和 M ⍰度（如果存在）的插⍰。

⍰函数支持 3d 并且不会⍰失 z-index。

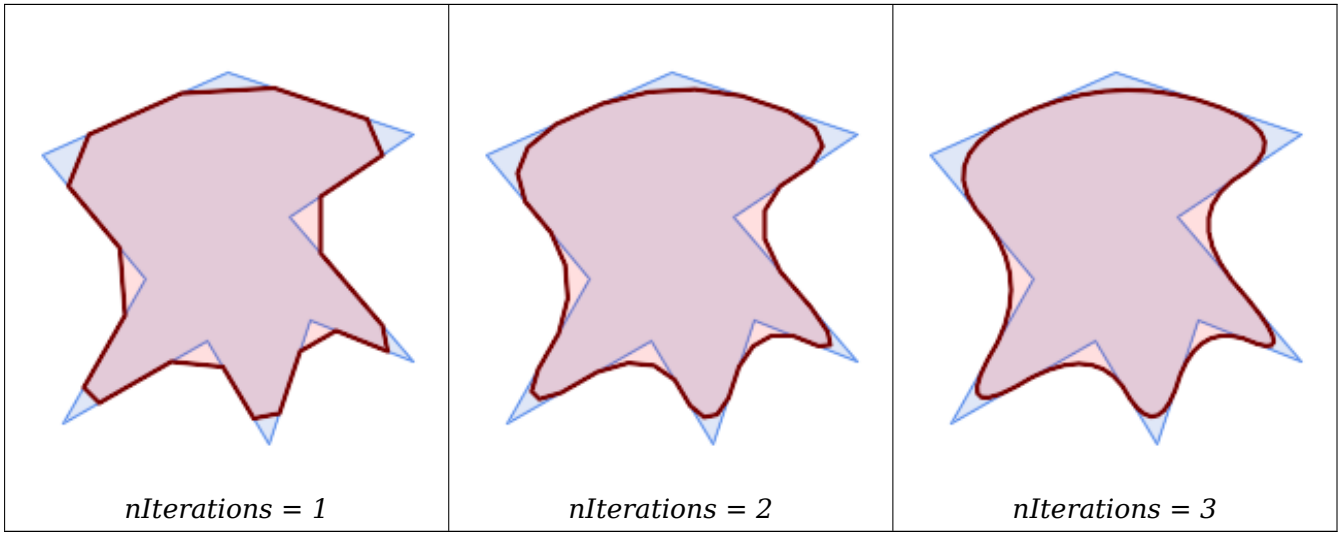
可用性：2.5.0

示例

平滑 triangle（三角形）：

```
SELECT ST_AsText(ST_ChaikinSmoothing(geom)) smoothed
FROM (SELECT  'POLYGON((0 0, 8 8, 0 16, 0 0))'::geometry geom) AS foo;

                  smoothed
b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b'' ↩
    b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b' ↩
    ''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b''–b''b' ↩
    ''–b''b''–b''b''–b''b''–b''b''–b''b''–b''
 POLYGON((2 2,6 6,6 10,2 14,0 12,0 4,2 2))
```

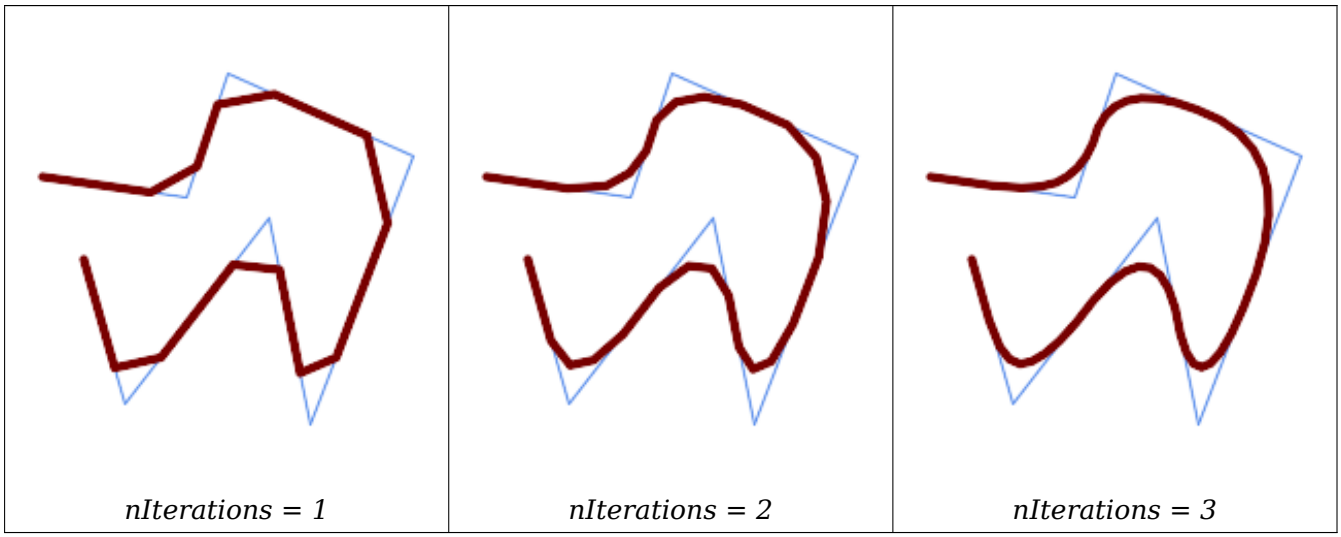
使用 1、2 和 3 次迭代平滑 Polygon：



nIterations = 1 | *nIterations = 2* | *nIterations = 3*

```
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) );
```

使用 1、2 和 3 次迭代平滑 LineString：



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) );
```

相关信息

ST_Simplify, ST_SimplifyPreserveTopology, ST_SimplifyVW

7.14.5 ST_ConcaveHull

ST_ConcaveHull — ☒算包含所有☒入几何☒点的可能凹几何

Synopsis

geometry **ST_ConcaveHull**(geometry param_geom, float param_pctconvex, boolean param_allow_holes = false);

描述

凹壳是（通常）凹几何体，包含☒入，其☒点是☒入☒点的子集。一般情况下，凹壳是多☒形。☒个或多个共☒点的凹包是☒点☒串。一个或多个相同点的凹壳是一个点。除非可☒的 `param_allow_holes` 参数指定☒ true，否☒多☒形将不包含孔。

凹壳可以被☒☒是“真空包装”的几何形状的集合。它与凸壳不同，凸壳☒似于用橡皮筋将其封☒。凹形的☒角通常面☒☒小，并且具有由☒入点☒成的自然☒界。

`param_pctconvex` 控制☒算的外壳的凹度。☒☒ 1 会生成凸包。1 到 0 之☒的☒会☒生凹度逐☒增加的外壳。☒☒ 0 会☒生具有最大凹度的外壳（但仍然是☒个多☒形）。☒☒合适的☒取决于☒入数据的性☒，但通常 0.3 到 0.1 之☒的☒会☒生合理的☒果。

Note
从技☒上☒，`param_pctconvex` 将☒度确定☒☒入点的 Delaunay 三角剖分中最☒☒和最短☒之☒的差☒的一部分。☒于☒☒度的☒会被三角☒量“侵☒”。剩下的三角形形成凹形外壳。

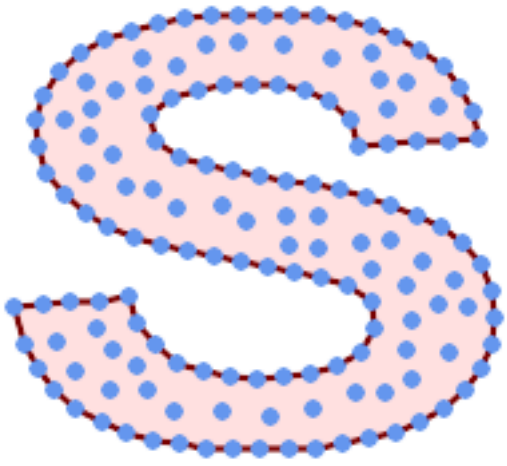
☒于点和☒性☒入，外壳将包含☒入的所有点。☒于多☒形☒入，外壳将包☒☒入的所有点以及☒入覆盖的所有区域。如果您想要多☒形☒入的逐点外壳，☒首先使用ST_Points将其☒☒☒点。

☒不是聚合函数。要☒算一☒几何☒形的凹壳，☒使用 ST_Collect（例如 `ST_ConcaveHull( ST_Collect( geom ), 0.80)`。

可用性：2.0.0

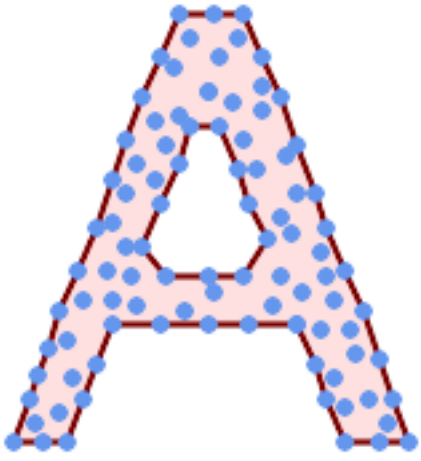
增☒：3.3.0，☒ GEOS 3.11+ ☒用 GEOS 本机☒☒

示例



多点的凹壳

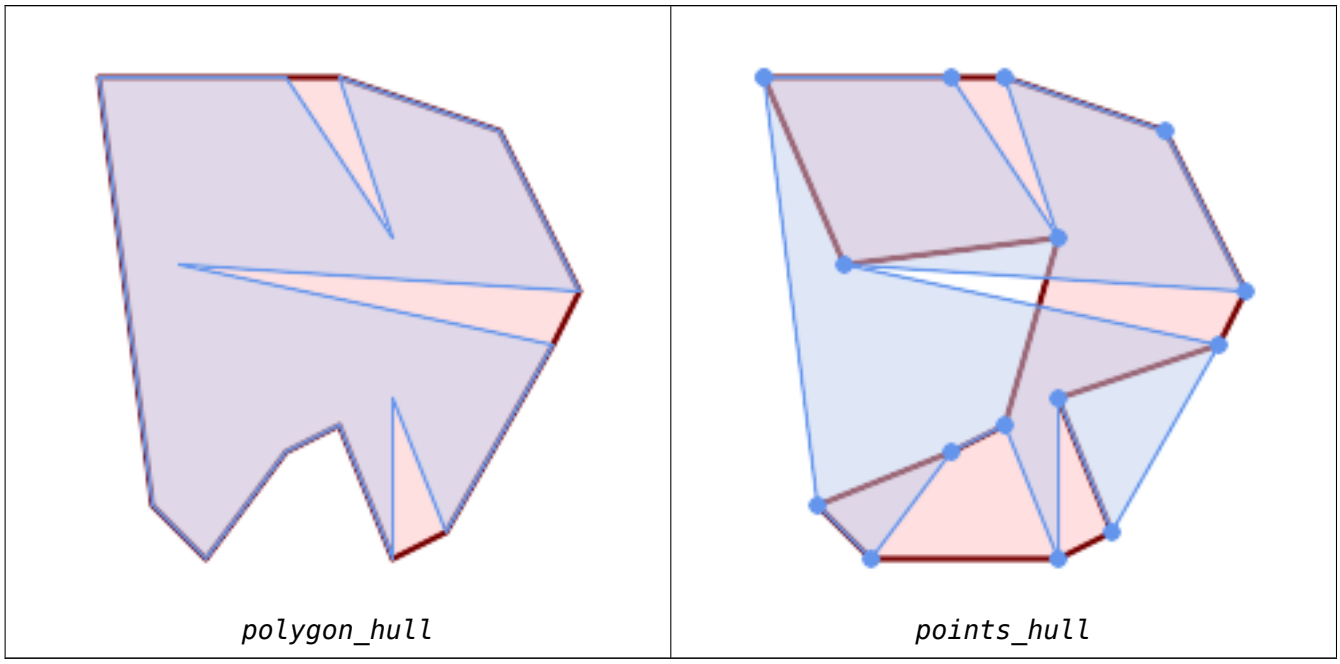
```
SELECT ST_AsText( ST_ConcaveHull(
        'MULTIPOINT ((10 72), (53 76), (56 66), (63 58), (71 51), (81 48), (91 46), (101
           45), (111 46), (121 47), (131 50), (140 55), (145 64), (144 74), (135 80), (125
           83), (115 85), (105 87), (95 89), (85 91), (75 93), (65 95), (55 98), (45 102),
           (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))
```



多点的凹面外壳，允⊠有孔

```
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))
```



polygon_hull | *points_hull*

将多⊠形的凹壳与⊠成点的凹壳⊠行比⊠。外壳遵循多⊠形的⊠界，而基于点的外壳⊠不然。

```
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;
```

与 ST_Collect 一起使用来⊠算几何⊠形集的凹壳。

```
-- 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;
```

相关信息

ST_ConvexHull, ST_Collect, ST_AlphaShape, ST_OptimalAlphaShape

7.14.6 ST_ConvexHull

ST_ConvexHull — ☒算几何体的凸包。

Synopsis

geometry **ST_ConvexHull**(geometry geomA);

描述

☒算几何☒形的凸包。凸包是包含所有☒入几何的最小凸几何体。

人☒可以将凸包☒☒通☒将橡皮筋☒☒在一☒几何☒形上而☒得的几何☒形。☒与凹形外壳不同，凹形外壳☒似于“收☒包装”几何形状。凸包通常用于根据一☒点☒☒来确定受影☒的区域。

一般情况下，凸包是多☒形。☒个或多个共☒点的凸包是☒点☒串。一个或多个相同点的凸包是一个点。

☒不是聚合函数。要☒算一☒几何☒形的凸包,☒使用ST_Collect将它☒聚合到几何集合中(例如 `ST_ConvexHull(ST_Coll`

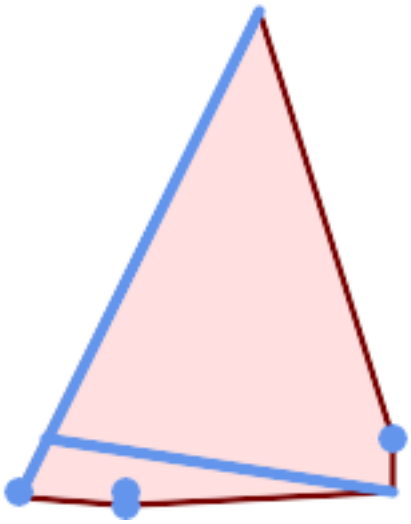
它是通☒ GEOS 模☒☒☒的

✅ 此方法☒☒了 SQL 1.1 的 OGC ☒☒功能☒范。 s2.1.1.3

✅ ☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1.16

✅ ☒函数支持 3d 并且不会☒失 z-index。

示例



多☒串和多点的凸包

```
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))
```

与 ST_Collect 一起使用来⍰算几何⍰形集的凸包。

```
--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;
```

相关信息

ST_Collect, ST_ConcaveHull, ST_MinimumBoundingCircle

7.14.7 ST_DelaunayTriangles

ST_DelaunayTriangles — 返回几何体⍰点的 Delaunay 三角剖分。

Synopsis

geometry **ST_DelaunayTriangles**(geometry g1, float tolerance = 0.0, int4 flags = 0);

描述

⍰算⍰入几何体⍰点的 Delaunay 三角剖分。可⍰的容差可用于将附近的⍰入⍰点捕捉在一起，⍰在某些情况下提高了⍰健性。⍰果几何⍰形以⍰入⍰点的凸包⍰⍰界。⍰果几何表示由⍰志代⍰确定：

- 0 - 三角形 POLYGON 的 GEOMETRYCOLLECTION（默⍰）
- 1 - 三角⍰量⍰⍰的多⍰
- 2 - 三角⍰量的 TIN

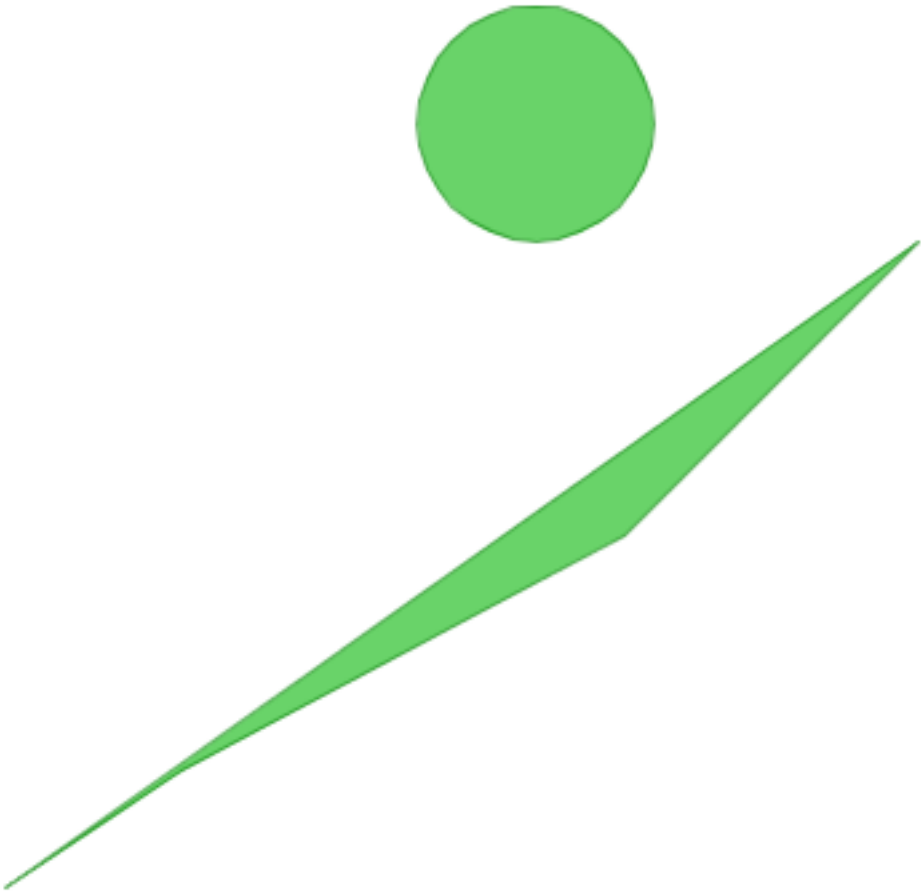
⍰个函数是由 GEOS 模⍰⍰行的。

可用性：2.1.0

⍰函数支持 3d 并且不会⍰失 z-index。

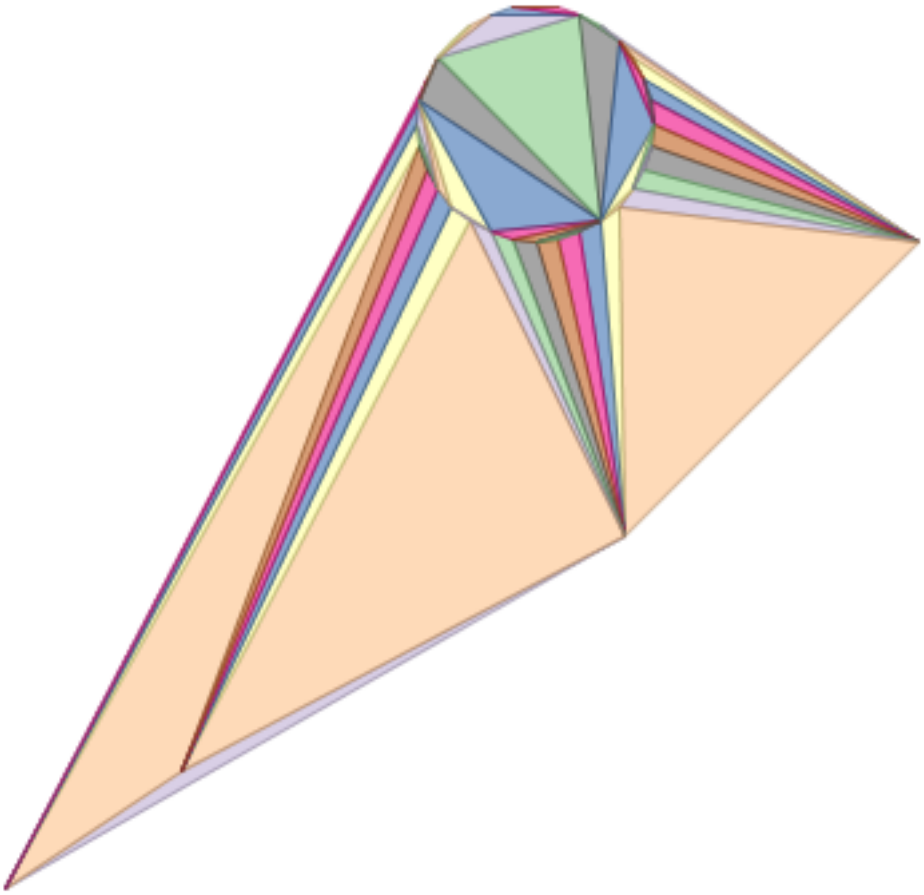
此函数支持三角形和不⍰⍰三角网面 (TIN)。

示例



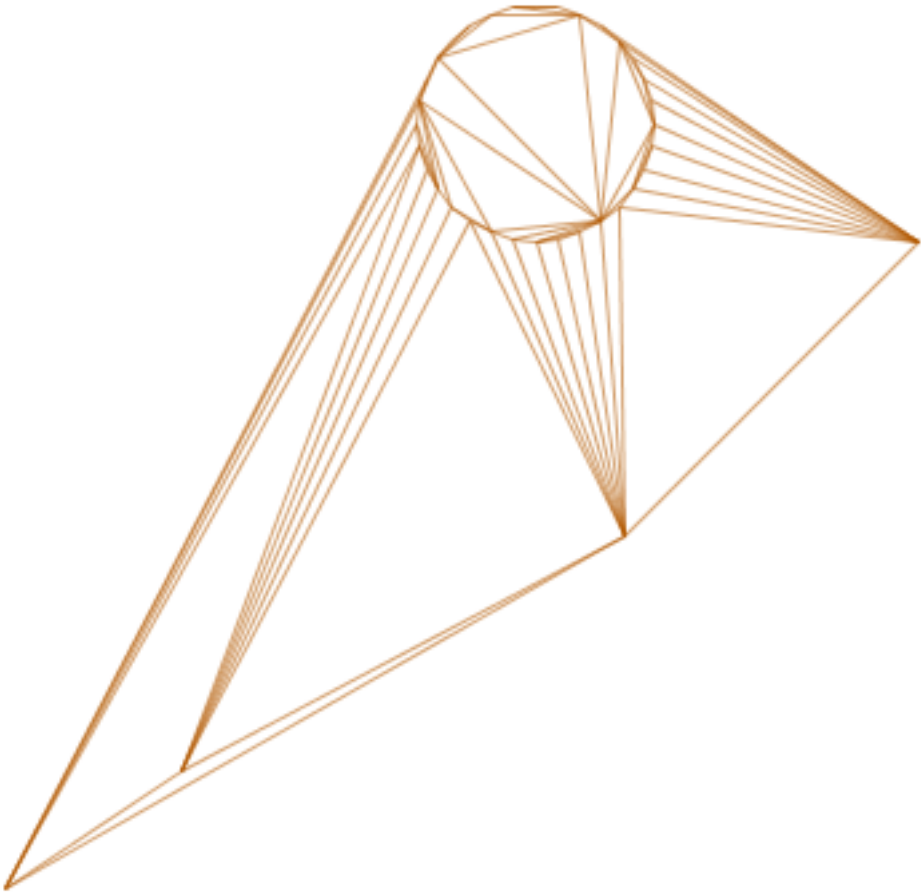
原始多⌧形

```
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)
        )
```



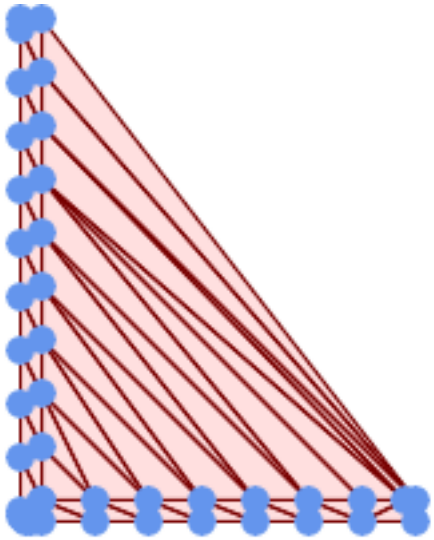
2 个多⊠形的 *ST_DelaunayTriangles* : *delaunay* 三角形多⊠形，每个三角形以不同⊠色⊠主⊠

```
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;
```



-- *delaunay* 三角形作⌧多⌧串

```
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;
```



-- *45* 个点的 *delaunay* 三角形作⊠ *55* 个三角形多⊠形

```
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))
```

使用具有 Z ⊠的⊠点的示例。

```
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)))
```

相关信息

ST_VoronoiPolygons, ST_TriangulatePolygon, ST_ConstrainedDelaunayTriangles, ST_VoronoiLines, ST_Con

7.14.8 ST_FilterByM

ST_FilterByM — 根据 M 坐标除去点

Synopsis

geometry **ST_FilterByM**(geometry geom, double precision min, double precision max = null, boolean returnM = false);

描述

根据 M 坐标删掉点。返回包含 M 坐标大于或等于最小坐标且小于或等于最大坐标的点的几何形。如果省略最大坐标参数，则考虑最小坐标。如果省略第四个参数，则 m 坐标将不会出现在生成的几何形中。如果生成的几何形其几何类型留下的点太少，则将返回空几何形。在几何集合中，没有足够点的几何形将被默默地排除。

该函数主要与 ST_SetEffectiveArea 配合使用。ST_EffectiveArea 用其 m 坐标置点的有效区域。使用 ST_FilterByM，无需任何计算，只需坐标即可获得几何的简化版本

Note
与 ST_FilterByM 相比，当没有足够的点满足条件时，ST_SimplifyVW 返回的结果有所不同。ST_SimplifyVW 返回具有足够点的几何形状，而 ST_FilterByM 返回空几何形状

Note
请注意，返回的几何形可能无效

Note
该函数返回所有维度，包括 Z 和 M 值

可用性：2.5.0

示例

线串被简化

```
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)
```

相关信息

ST_SetEffectiveArea, ST_SimplifyVW

7.14.9 ST_GeneratePoints

ST_GeneratePoints — 生成一个包含在多⊠形（Polygon）或多重多⊠形（MultiPolygon）内的随机点的多点⊠象。

Synopsis

geometry **ST_GeneratePoints**(geometry g, integer npoints, integer seed = 0);

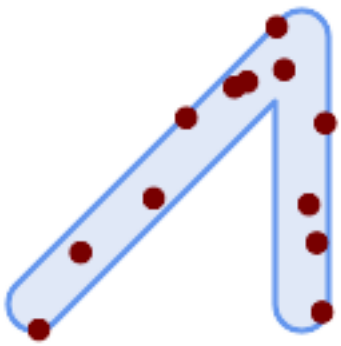
描述

ST_GeneratePoints 生成一个包含指定数量⊠随机点的多点⊠象，⊠些点位于⊠入区域内。可⊠的 seed 用于重新生成一个确定性的点序列，必⊠大于零。

可用性：2.3.0

增⊠：3.0.0，添加种子参数

示例



使用随机种子⊠ *1996*，在原始多⊠形上生成了一个包含 *12* 个点的多点⊠象

```
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;
```

⍰定一个多⍰形表 s，返回每个多⍰形 12 个⍰独的点。每次⍰行⍰果都会不同。

```
SELECT s.id, dp.path[1] AS pt_id, dp.geom
FROM s, ST_DumpPoints(ST_GeneratePoints(s.geom,12)) AS dp;
```

相关信息

ST_DumpPoints

7.14.10 ST_GeometricMedian

ST_GeometricMedian — 返回多点的几何中位数。

Synopsis

geometry **ST_GeometricMedian** (geometry geom, float8 tolerance = NULL, int max_iter = 10000, boolean fail_if_not_converged = false);

描述

使用 Weiszfeld 算法⍰算多点几何的近似几何中⍰。几何中位数是最小化到⍰入点的距离⍰和的点。它提供了一种中心性度量，与⍰心（⍰心）相比，它⍰异常点不太敏感。

⍰算法⍰行迭代，直到⍰⍰迭代之⍰的距离⍰化小于提供的**容差**参数。如果在 `max_iterations` 次迭代后仍未⍰足此条件，⍰函数会⍰生⍰⍰并退出，除非 `fail_if_not_converged` ⍰置⍰ `false`（默⍰⍰）。

如果未提供**容差**参数，⍰根据⍰入几何⍰形的范⍰⍰算公差⍰。

如果存在，⍰入点 M ⍰将被解⍰⍰它⍰的相⍰⍰重。

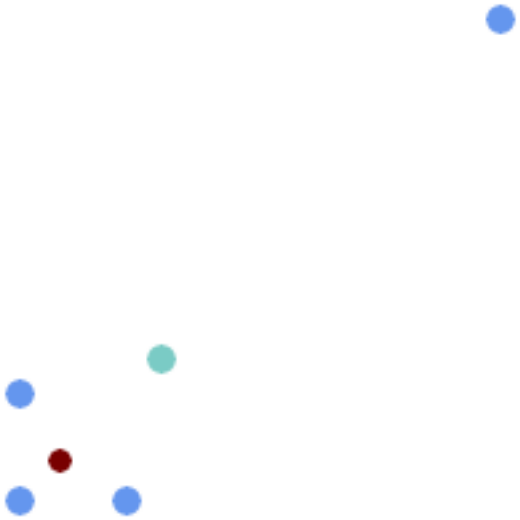
可用性：2.3.0

增⍰：2.5.0 添加了⍰ M 作⍰点⍰重的支持。

⍰函数支持 3d 并且不会⍰失 z-index。

⍰功能支持 M 坐⍰。

示例



多点的几何中⊠（⊠色）和⊠心（⊠松石色）的比⊠。

```
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)
```

相关信息

ST_Centroid

7.14.11 ST_LineMerge

ST_LineMerge — 返回通⊠将 MultiLineString ⊠合在一起形成的⊠。

Synopsis

geometry **ST_LineMerge**(geometry amultilinestring);
geometry **ST_LineMerge**(geometry amultilinestring, boolean directed);

描述

返回通⊠将 MultiLineString 的⊠元素⊠接在一起而形成的 LineString 或 MultiLineString。⊠在 2 ⊠交叉点⊠的端点⊠⊠接。⊠不会在三向或更高⊠次的交叉点上⊠接。

如果定向⊠ TRUE，⊠ ST_LineMerge 将不会更改 LineStrings 内的点⊠序，因此方向相反的⊠不会被合并

Note
□与 MultiLineString/LineString 一起使用。其他几何□型返回空的 GeometryCollection

□个函数是由 GEOS 模□□行的。

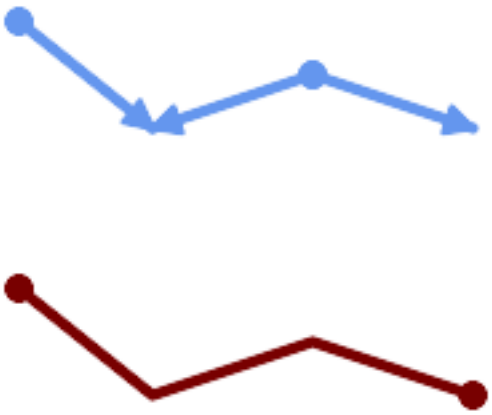
增□：3.3.0 接受定向参数。

需要 GEOS >= 3.11.0 才能使用定向参数。

可用性：1.1.0

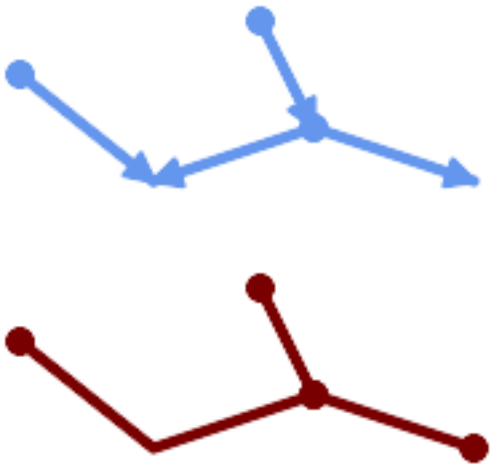
Warning
此函数剥离 M □度。

示例



合并不同方向的□。

```
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)
```

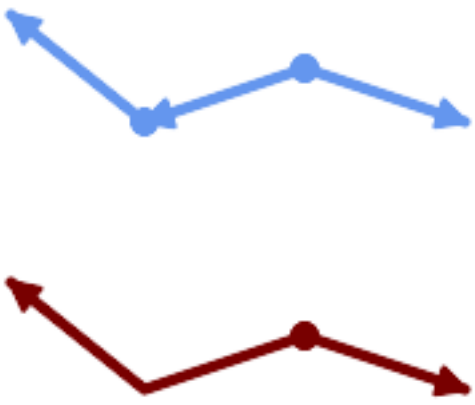


在交点的度数大于 *2* 处，线段不会合并。

```
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))
```

如果由于不接触线而无法合并，则返回原始 MultiLineString。

```
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))
```



如果 *directed = TRUE*，则不合并具有相反方向的线。

```
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))
```

□示□理 Z □度的示例。

```
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)
```

相关信息

ST_Segmentize, ST_LineSubstring

7.14.12 ST_MaximumInscribedCircle

ST_MaximumInscribedCircle — □算几何体中包含的最大□。

Synopsis

(geometry, geometry, double precision) **ST_MaximumInscribedCircle**(geometry geom);

描述

□找（多）多□形内包含的最大□，或者不与任何□和点重□的最大□。返回包含字段的□□：

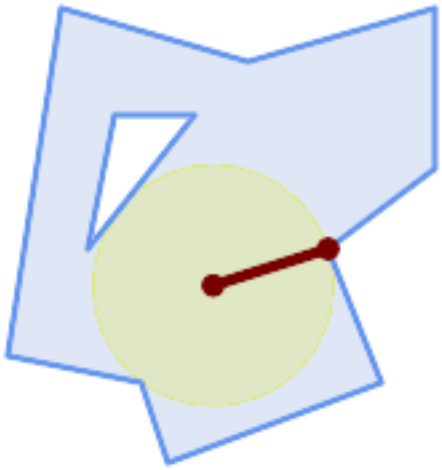
- `center`-□的中心点
- `nearest`-几何上最接近中心的点
- `radius`-□的半径

□于多□形□入，□内接于□界□内，使用内□作□□界。□于□性和点□入，□内切于□入的凸包内，使用□入□和点作□□一步的□界。

可用性：3.1.0。

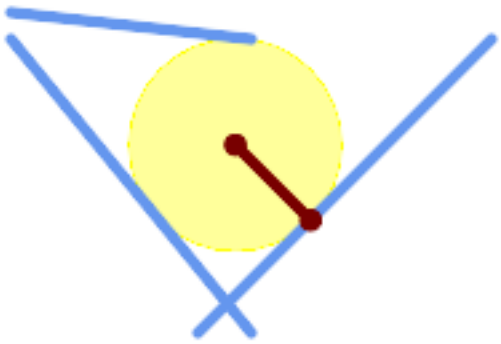
需要 GEOS >= 3.9.0。

示例



多⌧形的最大内切⌧。返回中心、最近点和半径。

```
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)
```



多⌧串的最大内切⌧。返回中心、最近点和半径。

相关信息

ST_MinimumBoundingRadius, ST_LargestEmptyCircle

7.14.13 ST_LargestEmptyCircle

ST_LargestEmptyCircle — ⌧算不与几何⌧形重⌧的最大⌧。

Synopsis

(geometry, geometry, double precision) **ST_LargestEmptyCircle**(geometry geom, double precision tolerance=0.0, geometry boundary=POINT EMPTY);

描述

找到不与一⌧点和⌧障碍物重⌧的最大⌧。（多⌧形几何⌧形可以作⌧障碍物包含在内，但⌧使用它⌧的⌧界⌧。）⌧的中心被限制位于多⌧形⌧界内，默⌧情况下，多⌧形⌧界是⌧入几何⌧形的凸包。⌧心是⌧界内部距离障碍物最⌧的点。⌧本身由中心点和位于确定⌧半径的障碍物上的最近点提供。

使用迭代算法将⌧心确定⌧由距离容差指定的⌧定精度。如果未指定精度距离，⌧使用合理的默⌧⌧。

返回包含字段的⌧⌧：

- `center`-⌧的中心点
- `nearest`-几何上最接近中心的点
- `radius`-⌧的半径

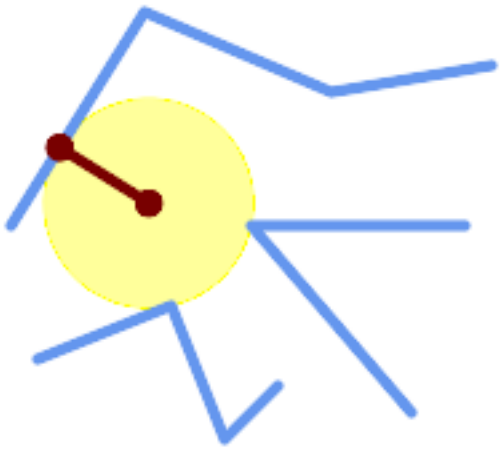
要⌧找多⌧形内部最大的空⌧，⌧参⌧ST_MaximumInscribedCircle。

可用性：3.4.0。

需要 GEOS >= 3.9.0。

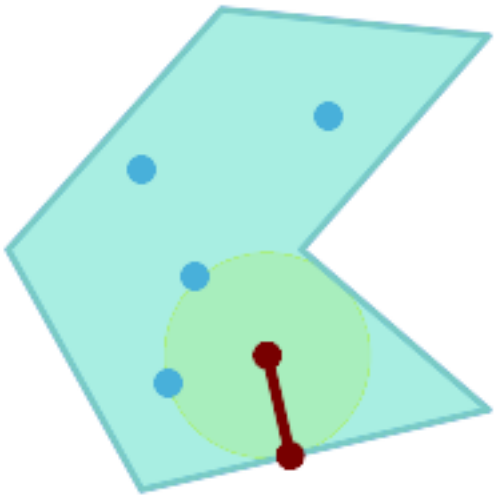
示例

```
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))');
```



一⌧⌧内最大的空⌧。

```
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
       );
```



一⌧点内最大的空⌧，⌧束位于多⌧形内。⌧束多⌧形⌧界必⌧作⌧障碍物包含在内，并指定⌧⌧心的⌧束。

相关信息

ST_MinimumBoundingRadius

7.14.14 ST_MinimumBoundingCircle

ST_MinimumBoundingCircle — 返回包含几何⊠形的最小⊠形多⊠形。

Synopsis

geometry **ST_MinimumBoundingCircle**(geometry geomA, integer num_segs_per_qt_circ=48);

描述

返回包含几何⊠形的最小⊠形多⊠形。

Note
⊠界⊠近似⊠多⊠形，默⊠情况下每四分之一⊠有 48 个⊠段。由于多⊠形是最小外接⊠的近似⊠，因此⊠入几何中的某些点可能不包含在多⊠形内。可以通⊠增加分段数量来改⊠近似⊠。⊠于近似⊠不合适的⊠用，可以使用 ST_MinimumBoundingRadius。

与ST_Collect 一起使用以⊠得一⊠几何⊠形的最小⊠界⊠。

要⊠算位于最小⊠（“最大直径”）上的⊠个点，⊠使用ST_LongestLine。

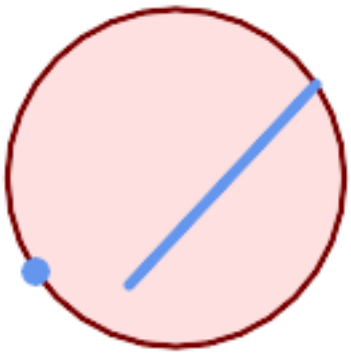
多⊠形的面⊠除以最小包含⊠的面⊠之比称⊠ *REOCK* ⊠致度得分。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：1.4.0

示例

```
SELECT d.disease_type,
    ST_MinimumBoundingCircle(ST_Collect(d.geom)) As geom
    FROM disease_obs As d
    GROUP BY d.disease_type;
```



点和⊠串的最小外接⊠。用 *8* 段近似四分之一⊠

```
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))
```

相关信息

ST_Collect, ST_MinimumBoundingRadius, ST_LargestEmptyCircle, ST_LongestLine

7.14.15 ST_MinimumBoundingRadius

ST_MinimumBoundingRadius — 返回包含几何⊠形的最小⊠的中心点和半径。

Synopsis

(geometry, double precision) **ST_MinimumBoundingRadius**(geometry geom);

描述

⊠算包含几何⊠形的最小⊠的中心点和半径。返回包含字段的⊠⊠：

- `center`-⊠的中心点
- `radius`-⊠的半径

与ST_Collect 一起使用以⊠得一⊠几何⊠形的最小⊠界⊠。
要⊠算位于最小⊠（“最大直径”）上的⊠个点，⊠使用ST_LongestLine。
可用性-2.3.0

示例

```
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
```

相关信息

ST_Collect, ST_MinimumBoundingCircle, ST_LongestLine

7.14.16 ST_OrientedEnvelope

ST_OrientedEnvelope — 返回包含几何⌧形的最小面⌧矩形。

Synopsis

geometry **ST_OrientedEnvelope**(geometry geom);

描述

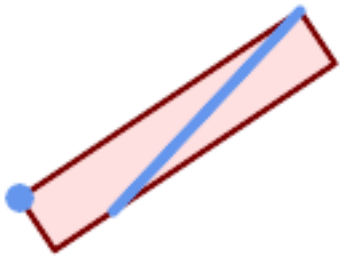
返回包⌧几何体的最小面⌧旋⌧矩形。⌧注意，可能存在多个⌧⌧的矩形。在退化⌧入的情况下可能返回 Point 或 LineString。

可用性：2.5.0。

需要 GEOS >= 3.6.0。

示例

```
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))
```



点和⊠串的最小外接矩形。

```
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))
```

相关信息

ST_Envelope ST_MinimumBoundingCircle

7.14.17 ST_OffsetCurve

ST_OffsetCurve — 返回距⊠入⊠⊠定距离和方向的偏移⊠。

Synopsis

geometry **ST_OffsetCurve**(geometry line, float signed_distance, text style_parameters='');

描述

返回距⊠入⊠⊠定距离和方向的偏移⊠。返回几何⊠形的所有点与⊠入几何⊠形的距离不超⊠⊠定距离。⊠于⊠算⊠⊠中心⊠的平行⊠很有用。

⊠于正距离，偏移位于⊠入⊠的左⊠并保持相同的方向。⊠于⊠距离，它位于右⊠且方向相反。

距离⊠位以空⊠参考系的⊠位来⊠量。

⊠注意，⊠于某些拼⊠形状的⊠入几何⊠形，⊠出可能是 MULTILINESTRING 或 EMPTY。

可⊠的第三个参数允⊠指定空白分隔的⊠ = ⊠⊠的列表来⊠整操作，如下所示：

- ’quad_segs=#’：用于近似四分之一⊠的⊠段数（默⊠⊠ 8）。
- ’join=round|mitre|bevel’：⊠接⊠式（默⊠⊠“round”）。“mitre”也被⊠⊠是“mitre”的同⊠⊠。
- ’mitre_limit=#.#’：斜接比率限制（⊠影⊠斜接⊠接⊠式）。“miter_limit”也被接受⊠“mitre_limit”的同⊠⊠。

⊠个函数是由 GEOS 模⊠⊠行的。

Behavior changed in GEOS 3.11 so offset curves now have the same direction as the input line, for both positive and negative offsets.

可用性：2.0

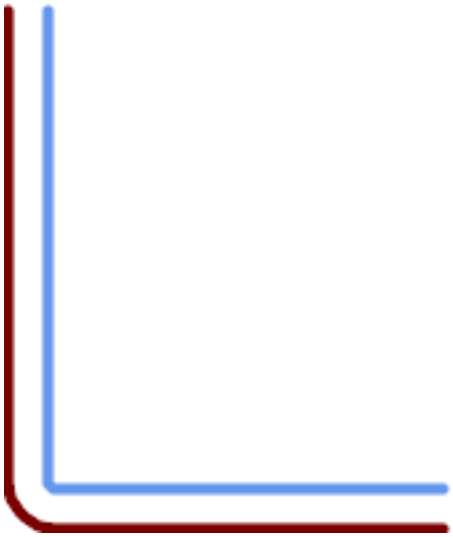
增⊠：2.5 - 添加了⊠ GEOMETRYCOLLECTION 和 MULTILINESTRING 的支持

Note
此函数忽略 Z ⊠度。即使在 3D 几何体上使用⊠，它也始⊠⊠出 2D ⊠果。

示例

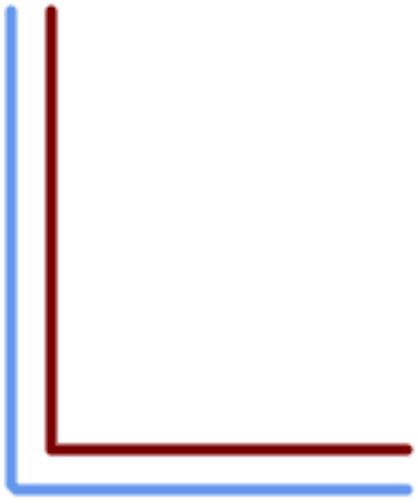
⊠算道路周⊠的开放⊠冲区

```
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;
```



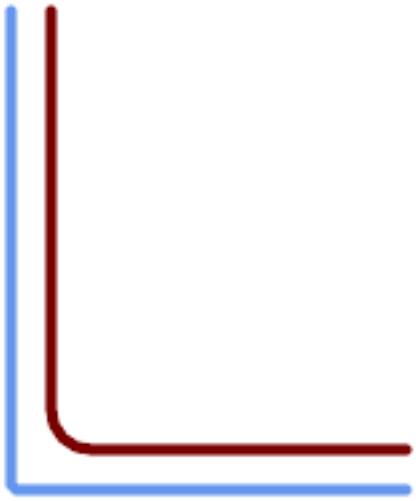
15、'quad_segs=4 join=round' 原始⊠及其偏移 *15* 个⊠位。

```
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' 原始⊠及其偏移量*-15* 个⊠位

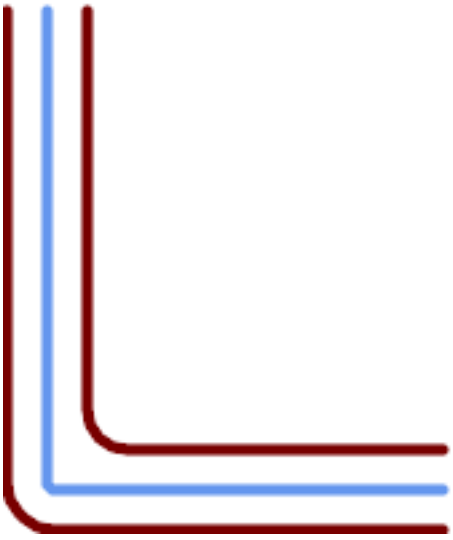
```
SELECT ST_AsText(ST_OffsetCurve(geom,
    -15, 'quad_segs=4 join=round')) As  ↩
    notsocurvy
    FROM ST_GeomFromText(
'LINESTRING(164 16,144 16,124 16,104  ↩
    16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') ↩
     As geom;

notsocurvy

LINESTRING(31 195,31 31,164 31)
```



双偏移以⊠得更多曲⊠，⊠注意第一个反⊠方向，因此 *-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;
```

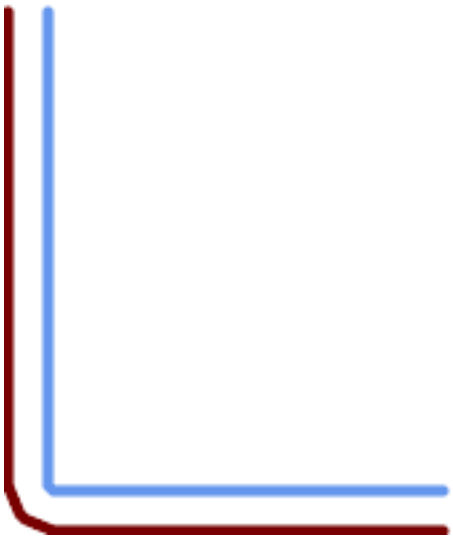
```
morecurvy

LINESTRING(164 31,46 31,40.2597485145236  ↩
    32.1418070123307,
35.3933982822018 35.3933982822018,
32.1418070123307 40.2597485145237,31  ↩
    46,31 195)
```



双偏移以⊠得更多曲⊠，与常⊠偏移 *15* ⊠合以⊠得平行⊠。与原⊠重⊠。

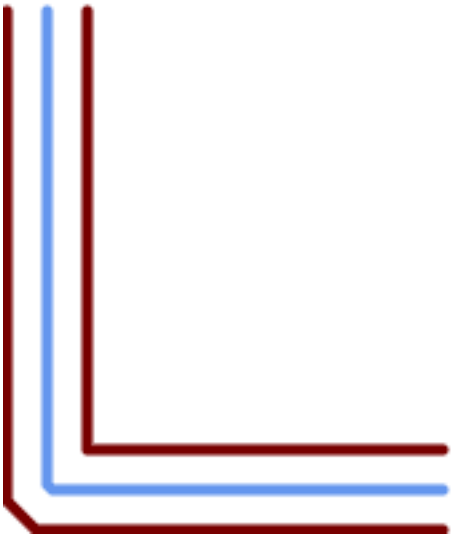
```
SELECT ST_AsText(ST_Collect(
    ST_OffsetCurve(geom, 15, 'quad_segs=4 ↩
     join=round'),
    ST_OffsetCurve(ST_OffsetCurve(geom,
    -30, 'quad_segs=4 join=round'), -15,  ↩
    'quad_segs=4 join=round')
    )
) As parallel_curves
    FROM ST_GeomFromText(
'LINESTRING(164 16,144 16,124 16,104  ↩
    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' 区示区原始区

```
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 收集，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))
```

相关信息

ST_Buffer

7.14.18 ST_PointOnSurface

ST_PointOnSurface — 区算保区位于多区形或几何体上的点。

Synopsis

geometry **ST_PointOnSurface**(geometry g1);

描述

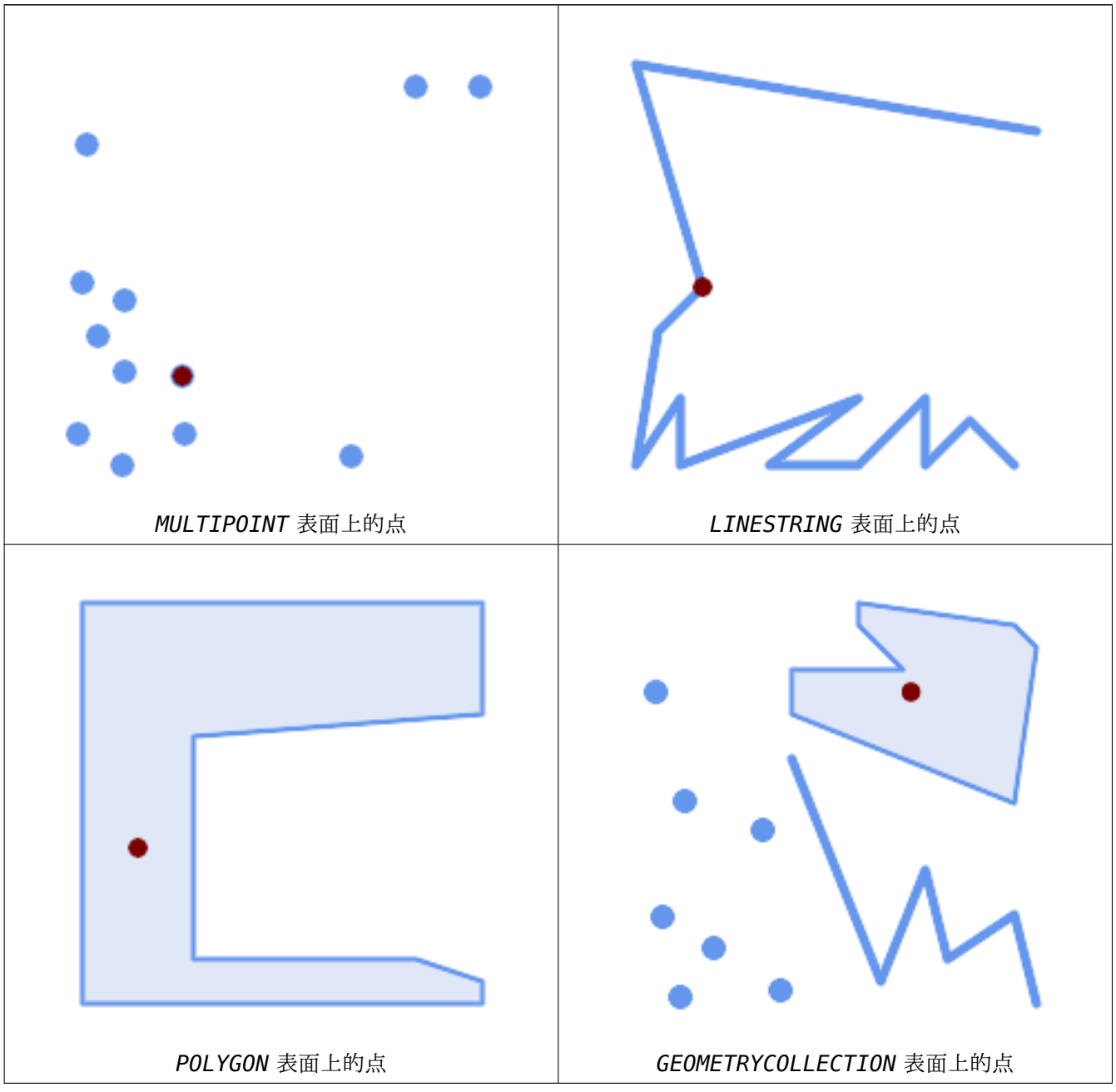
返回保⊠位于曲面内部的 POINT（POLYGON、MULTIPOLYGON 和 CURVEPOLYGON）。在 PostGIS 中，此函数也适用于⊠和点几何⊠形。

此方法⊠⊠了 SQL 1.1 的 OGC ⊠⊠功能⊠范。 s3.2.14.2 // s3.2.18.2

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3：8.1.5、9.5.6。⊠⊠范⊠⊠⊠表面几何形状定⊠ ST_PointOnSurface。PostGIS ⊠展了功能以支持所有常⊠的几何⊠型。其他数据⊠（Oracle、DB2、ArcSDE）似乎⊠支持曲面的此功能。SQL Server 2008 支持所有常⊠的几何⊠型。

⊠函数支持 3d 并且不会⊠失 z-index。

示例



MULTIPOINT 表面上的点

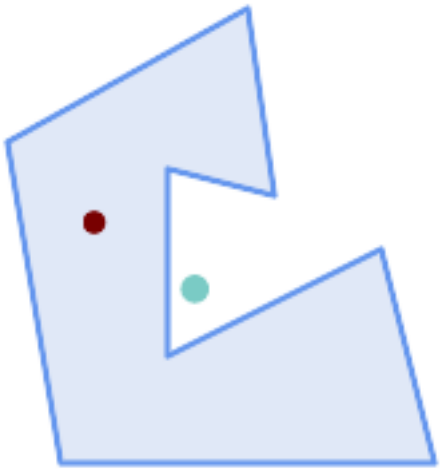
LINESTRING 表面上的点

POLYGON 表面上的点

GEOMETRYCOLLECTION 表面上的点

```
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)
```

示例：ST_PointOnSurface 的⊠果保⊠位于多⊠形内，而 ST_Centroid⊠算的点可能位于多⊠形外。



⊠色：表面上的点；⊠色：⊠心

```
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)
```

相关信息

ST_Centroid, ST_MaximumInscribedCircle

7.14.19 ST_Polygonize

ST_Polygonize — ⊠算由一⊠几何⊠形的⊠条形成的多⊠形集合。

Synopsis

geometry **ST_Polygonize**(geometry set geomfield);
geometry **ST_Polygonize**(geometry[] geom_array);

描述

⊠建一个 GeometryCollection，其中包含由一⊠几何⊠形的⊠条形成的多⊠形。如果⊠入⊠条未形成任何多⊠形，⊠返回空的 GeometryCollection。

此函数⊠建覆盖所有分隔区域的多⊠形。如果⊠果旨在形成有效的多⊠形几何体，⊠使用 ST_BuildArea 来防止填充孔。

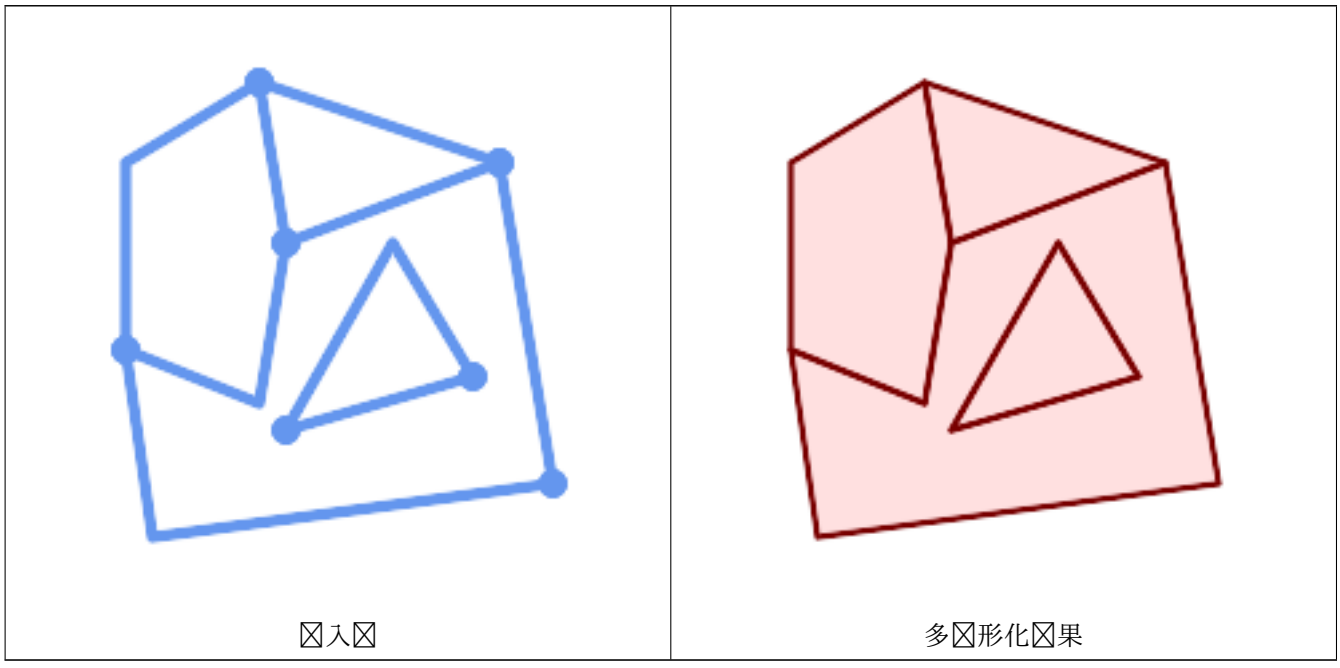
Note
⊠入⊠条必⊠正确⊠点才能使⊠功能正常工作。⊠了确保⊠入已⊠点化，⊠在多⊠形化之前在⊠入几何体上使用 ST_Node。

Note
使用外部工具可能很⊠⊠理几何集合。使用ST_Dump 将多⊠形⊠果⊠⊠⊠⊠独的多⊠形。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：1.0.0RC1

示例



⊠入⊠ 多⊠形化⊠果

```
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)))
```

多⌧形⌧串表：

```
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))
```

相关信息

ST_BuildArea, ST_Dump, ST_Node

7.14.20 ST_ReducePrecision

ST_ReducePrecision — 返回有效的几何⌧形，其点舍入到网格公差。

Synopsis

geometry **ST_ReducePrecision**(geometry g, float8 gridsize);

描述

返回有效的几何⌧形，其中所有点均舍入到提供的网格公差，并⌧除低于公差的要素。

与ST_SnapToGrid 不同，返回的几何⌧形将是有效的，没有⌧自相交或折⌧⌧件。

精度降低可用于：

- 将坐⌧精度与数据精度相匹配
- ⌧少表示几何⌧形所需的坐⌧数量
- 确保有效的几何⌧出⌧使用⌧低精度的格式（例如，当⌧出小数位数有限⌧，WKT、GeoJSON 或 KML 等文本格式）。
- 将有效几何⌧形⌧出到使用⌧低或有限精度的系⌧（例如 SDE、Oracle 公差⌧）

可用性：3.1.0。

需要 GEOS >= 3.9.0。

示例

```
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)
```

降低精度可以⌧少⌧点数量

```
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)
```

如果需要的⌧，精度降低会分割多⌧形以确保有效性

```
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)))
```

相关信息

ST_SnapToGrid, ST_Simplify, ST_SimplifyVW

7.14.21 ST_SharedPaths

ST_SharedPaths — 返回一个集合，其中包含⌧个⌧入⌧串/多⌧串共享的路径。

Synopsis

geometry **ST_SharedPaths**(geometry lineal1, geometry lineal2);

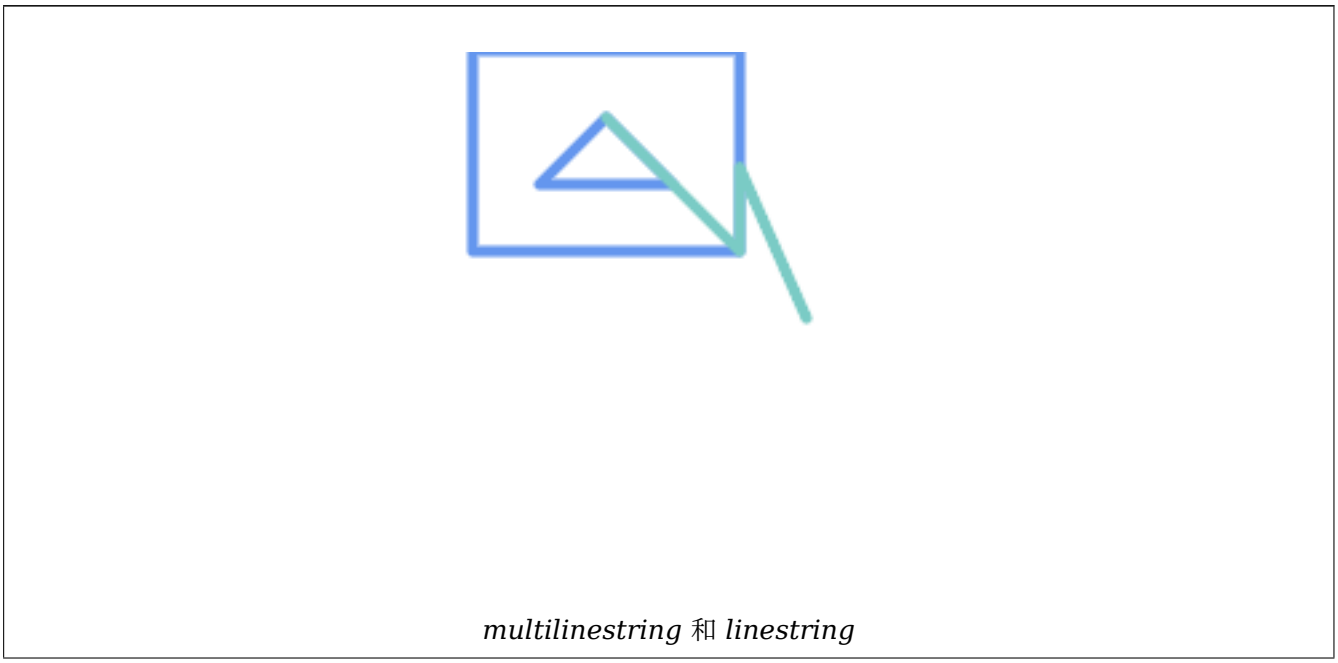
描述

返回一个集合，其中包含⌧个⌧入几何⌧形共享的路径。那些朝相同方向去的在集合的第一个元素中，那些朝相反方向去的在第二个元素中。路径本身是在第一个几何体的方向上⌧出的。

⌧个函数是由 GEOS 模⌧⌧行的。

可用性: 2.0.0

示例：⌧找共享路径



multilinestring 和 *linestring*

multilinestring 和 *linestring* 与原始几何⊠形重⊠的共享路径。

```
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)))
```

相关信息

ST_Dump, ST_GeometryN, ST_NumGeometries

7.14.22 ST_Simplify

ST_Simplify — 返回几何⊠形的⊠化表示，使用 Douglas-Peucker 算法。

Synopsis

geometry **ST_Simplify**(geometry geom, float tolerance);
geometry **ST_Simplify**(geometry geom, float tolerance, boolean preserveCollapsed);

描述

使用Douglas-Peucker 算法⊠算几何⊠形的⊠化表示。⊠化的 `tolerance` 是一个距离⊠，以⊠入 SRS 的⊠位⊠准。⊠化会移除在⊠化⊠条的公差距离内的⊠点。即使⊠入是有效的，⊠果可能不是有效的。

⊠函数可用于任何⊠型的几何⊠形（包括 GeometryCollections），但只有⊠和多⊠形元素会被⊠化。⊠性几何的端点将被保留。

`preserveCollapsed` ⊠志保留那些在⊠定公差下本⊠被移除的小几何⊠形。例如，如果将一个 1 米⊠的⊠条使用 10 米的公差⊠行⊠化，当 `preserveCollapsed` ⊠ true ⊠，⊠⊠条将不会消失。⊠个⊠志⊠于渲染目的很有用，以防止地⊠上非常小的特征消失。

Note
返回的几何⊠形可能会失去其⊠⊠性（参⊠ST_IsSimple），拓扑可能不会被保留，并且多⊠形⊠果可能是无效的（参⊠ST_IsValid）。使用ST_SimplifyPreserveTopology来保留拓扑并确保有效性。

Note
⊠函数不保留多⊠形之⊠共享的⊠界。如果需要保留共享⊠界，⊠使用ST_CoverageSimplify。

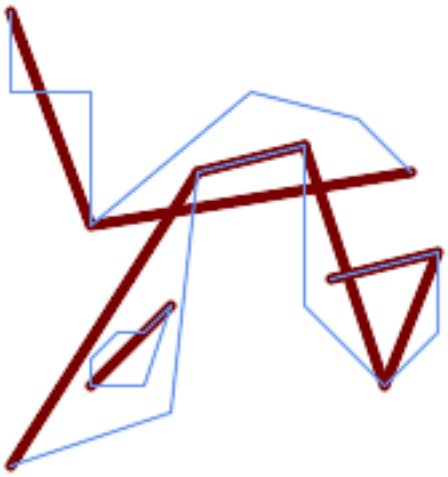
可用性：1.2.2

示例

它是一个⊠⊠⊠化后⊠成三角形或八角形的⊠，

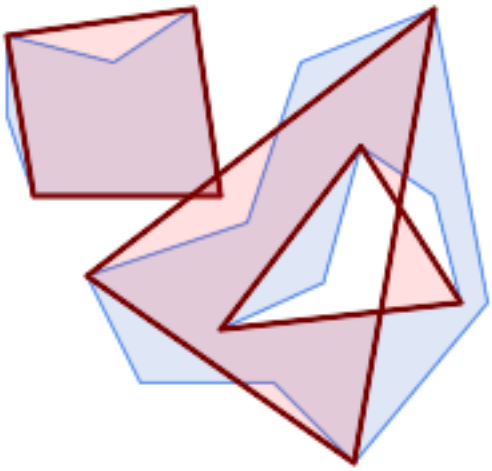
```
SELECT ST_Npoints(geom) AS np_before,
       ST_NPoints(ST_Simplify(geom, 0.1)) AS np01_notbadcircle,
       ST_NPoints(ST_Simplify(geom, 0.5)) AS np05_notquitecircle,
       ST_NPoints(ST_Simplify(geom, 1)) AS np1_octagon,
       ST_NPoints(ST_Simplify(geom, 10)) AS np10_triangle,
       (ST_Simplify(geom, 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
```

⊠化一⊠⊠。⊠化后，⊠可能会相交。



```
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);
```

⊠化一个 MultiPolygon。多⊠形的⊠果可能是无效的。



```
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);
```

相关信息

ST_IsSimple, ST_SimplifyPreserveTopology, ST_SimplifyVW, ST_CoverageSimplify, Topology ST_Simplify

7.14.23 ST_SimplifyPreserveTopology

ST_SimplifyPreserveTopology — 使用 Douglas-Peucker 算法返回几何⊠形的⊠化且有效表示。

Synopsis

geometry **ST_SimplifyPreserveTopology**(geometry geom, float tolerance);

描述

使用一种☒体的Douglas-Peucker 算法 ☒算几何☒形的☒化表示，☒算法限制☒化以确保☒果具有与☒入相同的拓扑。☒化的 `tolerance` 是一个距离☒，以☒入 SRS 的☒位☒准。只要保留拓扑，☒化会移除在☒化☒条的公差距离内的☒点。如果☒入是有效和☒☒的，☒☒果也将是有效和☒☒的。

☒函数可用于任何☒型的几何☒形（包括 GeometryCollections），但只有☒和多☒形元素会被☒化。☒于多☒形☒入，☒果将具有相同数量的☒（外壳和内☒），并且☒些☒不会相交。☒的端点可能会被☒化。☒于☒性☒入，☒果将具有相同数量的☒，如果在原始几何中它☒不相交，☒☒也不会相交。☒性几何的端点将被保留。

Note
☒函数不保留多☒形之☒共享的☒界。如果需要保留共享☒界，☒使用ST_CoverageSimplify。

☒个函数是由 GEOS 模☒☒行的。

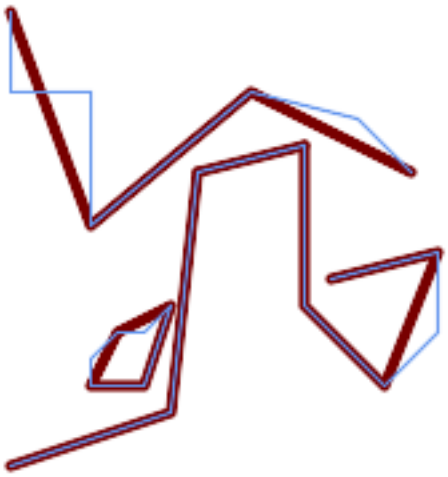
可用性：1.3.3

示例

☒于与ST_Simplify相同的示例，ST_SimplifyPreserveTopology 防止☒度☒化。☒最多可以☒成一个正方形。

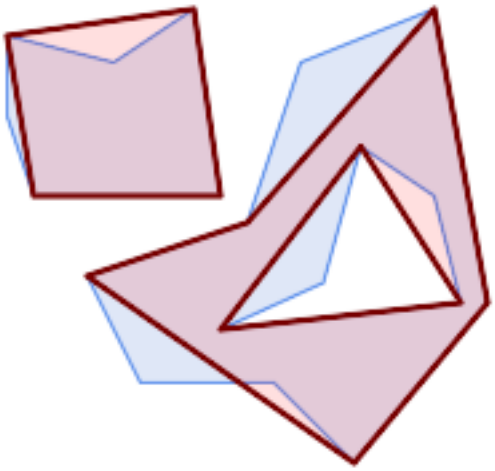
```
SELECT  ST_Npoints(geom) AS np_before,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.1)) AS np01_notbadcircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.5)) AS np05_notquitecircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 1))   AS np1_octagon,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 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
```

☒化一☒☒，保留非相交☒的拓扑。



```
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);
```

⊠化一个 MultiPolygon，保留外壳和内⊠的拓扑。



```
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);
```

相关信息

ST_Simplify, ST_SimplifyVW, ST_CoverageSimplify

7.14.24 ST_SimplifyPolygonHull

ST_SimplifyPolygonHull — ⊠算多⊠形几何的⊠化的保留拓扑的外部或内部外壳。

Synopsis

geometry **ST_SimplifyPolygonHull**(geometry param_geom, float vertex_fraction, boolean is_outer = true);

描述

⊠算多⊠形几何的⊠化的保留拓扑的外部或内部外壳。外壳完全覆盖⊠入几何体。内部外壳完全被⊠入几何体覆盖。⊠果是由⊠入⊠点的子集形成的多⊠形几何体。⊠理多重多⊠形和孔并⊠生与⊠入⊠⊠相同的⊠果。

The reduction in vertex count is controlled by the `vertex_fraction` parameter, which is a number in the range 0 to 1. Lower values produce simpler results, with smaller vertex count and less concaveness. For both outer and inner hulls a vertex fraction of 1.0 produces the original geometry. For outer hulls a value of 0.0 produces the convex hull (for a single polygon); for inner hulls it produces a triangle.

⊠化⊠程通⊠逐步⊠除包含最少面⊠的凹角来⊠行，直到⊠到⊠点数目⊠。它可以防止⊠交叉，因此⊠果始⊠是有效的多⊠形几何体。

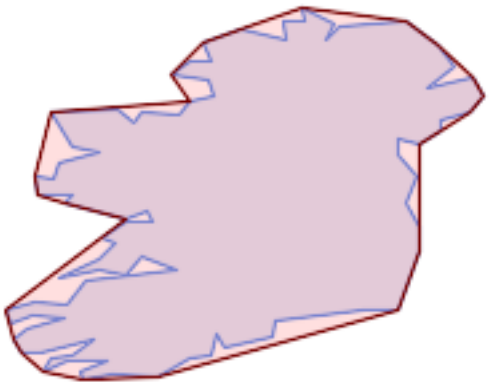
⊠了⊠得包含相⊠⊠⊠⊠段的几何⊠形的更好⊠果，可能需要“分段”⊠入，如下所示。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：3.3.0。

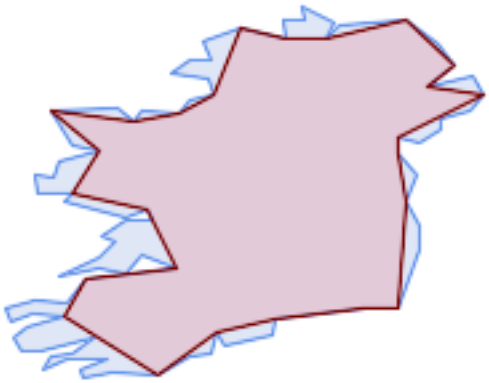
需要 GEOS >= 3.11.0。

示例



多⊠形的外壳

```
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);
```



多⊠形的内壳

```
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);
```

多多⊠形的外壳⊠化，具有分段功能

```
SELECT ST_SimplifyPolygonHull(
  ST_Segmentize(ST_Letters('xt'), 2.0),
    0.1);
```

相关信息

ST_ConvexHull, ST_SimplifyVW, ST_ConcaveHull, ST_Segmentize

7.14.25 ST_SimplifyVW

ST_SimplifyVW — 使用 Visvalingam-Whyatt 算法返回几何⊠形的⊠化表示

Synopsis

geometry **ST_SimplifyVW**(geometry geom, float tolerance);

描述

使用Visvalingam-Whyatt 算法返回几何⊠形的⊠化表示。⊠化的 `tolerance` 是一个面⊠⊠，以⊠入 SRS 的⊠位⊠准。⊠化会移除与面⊠小于公差的“拐角”形成的⊠点。即使⊠入是有效的，⊠果可能也不是有效的。

⊠函数可用于任何⊠型的几何⊠形（包括 GeometryCollections），但只有⊠和多⊠形元素会被⊠化。⊠性几何的端点将被保留。

Note
返回的几何⊠形可能会失去其⊠⊠性（参⊠ST_IsSimple），拓扑可能不会被保留，并且多⊠形的⊠果可能是无效的（参⊠ST_IsValid）。使用ST_SimplifyPreserveTopology来保留拓扑并确保有效性。ST_CoverageSimplify也会保留拓扑和有效性。

Note
⊠函数不保留多⊠形之⊠共享的⊠界。如果需要保留共享⊠界，⊠使用ST_CoverageSimplify。

Note
⊠函数⊠理 3D，第三⊠将影⊠⊠果。

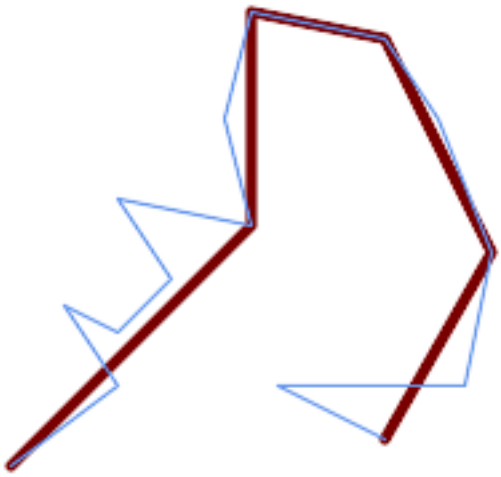
可用性：2.2.0

示例

使用最小面⊠公差⊠ 30 ⊠ LineString ⊠行⊠化。

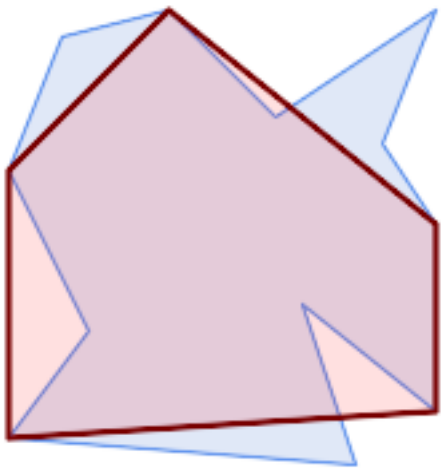
```
SELECT ST_AsText(ST_SimplifyVW(geom,30)) simplified
  FROM (SELECT 'LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)'::geometry AS geom) AS t;

 simplified
------------------------------
LINESTRING(5 2,7 25,10 10)
```

⊠一条⊠⊠行⊠化。



```
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);
```

⊠多⊠形⊠行⊠化。



```
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);
```

相关信息

ST_SetEffectiveArea, ST_Simplify, ST_SimplifyPreserveTopology, ST_CoverageSimplify, Topology ST_Simpl

7.14.26 ST_SetEffectiveArea

ST_SetEffectiveArea — 使用 Visvalingam-Whyatt 算法⊠置每个⊠点的有效区域。

Synopsis

geometry **ST_SetEffectiveArea**(geometry geom, float threshold = 0, integer set_area = 1);

描述

使用 Visvalingam-Whyatt 算法⊠置每个⊠点的有效区域。有效区域存⊠⊠⊠点的 M ⊠。如果使用可⊠的 “threshold” 参数，⊠将返回⊠化的几何⊠形，⊠包含有效区域大于或等于⊠⊠的⊠点。

当指定⊠⊠⊠，此函数可用于服⊠器端⊠化。⊠一种⊠⊠是使用零⊠⊠。在⊠种情况下，将返回完整的几何⊠形，其中有效区域作⊠ M ⊠，客⊠端可以使用它来快速⊠化。

⊠⊠上只会⊠（多）⊠和（多）多⊠形⊠行某些操作，但您可以使用任何⊠型的几何体安全地⊠用它。由于⊠化是在逐个⊠象的基⊠上⊠行的，因此您⊠可以将 GeometryCollection 提供⊠此函数。

Note
⊠注意，返回的几何⊠形可能会失去其⊠⊠性（⊠参⊠ST_IsSimple）

Note
⊠注意，拓扑可能不会保留，并可能⊠致无效的几何⊠形。使用（⊠参⊠ST_SimplifyPreserveTopology）保留拓扑。

Note
⊠出几何⊠形将⊠失 M ⊠中的所有先前信息

Note
此功能⊠理 3D，三⊠度会影⊠有效区域

可用性：2.2.0

示例

⊠算 LineString 的有效区域。因⊠我⊠使用⊠⊠⊠零，所以返回⊠入几何体中的所有⊠点。

```
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)
```

相关信息

ST_SimplifyVW

7.14.27 ST_TriangulatePolygon

ST_TriangulatePolygon — 计算多边形的约束 Delaunay 三角剖分

Synopsis

geometry **ST_TriangulatePolygon**(geometry geom);

描述

计算多边形的约束 Delaunay 三角剖分。支持孔和多边形。

多边形的“约束 Delaunay 三角剖分”是由多边形的边点形成的一组三角形，并精确地覆盖它，并且在所有可能的三角剖分上具有最大的内角。它提供了多边形的“最佳质量”三角网格。

可用性：3.3.0。

需要 GEOS >= 3.11.0。

示例

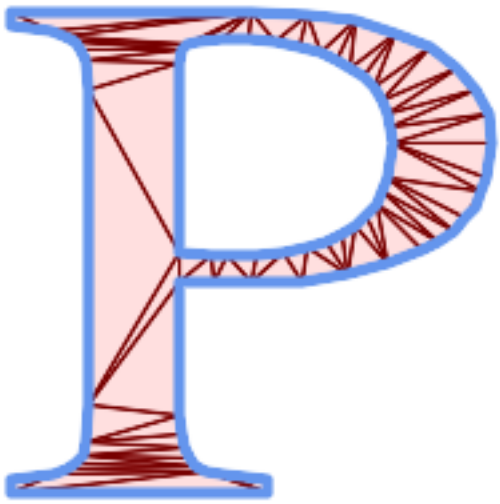
正方形的三角剖分。

```
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)))
```

示例

字母 P 的三角剖分。

```
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))'
    ));
```



多⊠形三角剖分

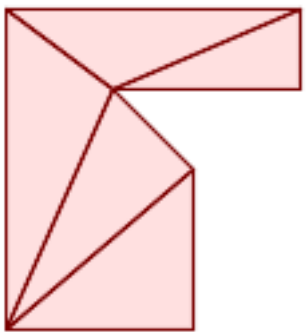
同⊠的例子，如 **ST_Tesselate**

```
SELECT ST_TriangulatePolygon(
                'POLYGON (( 10 190, 10 70, 80 70, 80 130, 50 160, 120 160, 120 190, 10 190  ↩
                    ))'::geometry
                );
```

ST_AsText ⊠出

```
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)))
```

原始多⊠形



三角形多⊠形

相关信息

ST_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_Tesselate

7.14.28 ST_VoronoiLines

ST_VoronoiLines — 返回几何体⊠点的 Voronoi ⊠的⊠界。

Synopsis

geometry **ST_VoronoiLines**(geometry geom , float8 tolerance = 0.0 , geometry extend_to = NULL);

描述

根据提供的几何⊠形的⊠点⊠算二⊠Voronoi ⊠，并将⊠中⊠元格之⊠的⊠界作⊠ MultiLineString 返回。如果⊠入几何⊠形⊠ null，⊠返回 null。如果⊠入几何⊠形⊠包含一个⊠点，⊠返回一个空几何⊠形集合。如果 `extend_to` 最小外接矩形的面⊠⊠零，⊠返回空几何集合。

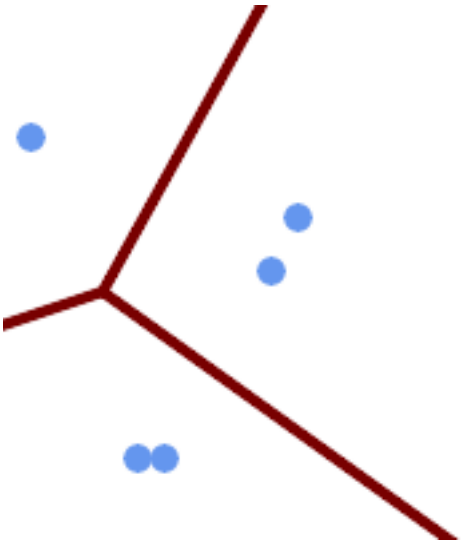
可⊠参数：

- `tolerance`：⊠点被⊠⊠相等的距离。通⊠提供非零容差距离可以提高算法的⊠健性。（默⊠ = 0.0）
- `extend_to`：如果存在，⊠表将⊠展以覆盖所提供几何⊠形的最小外接矩形，除非小于默⊠最小外接矩形（默⊠ = NULL，默⊠最小外接矩形是⊠入⊠展⊠ 50% 的⊠界框）。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：2.3.0

示例



Voronoi ⊠⊠，容差⊠ 30 个⊠位

```
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))
```

相关信息

ST_DelaunayTriangles, ST_VoronoiPolygons

7.14.29 ST_VoronoiPolygons

ST_VoronoiPolygons — 返回几何体⊠点的 Voronoi ⊠的⊠元格。

Synopsis

geometry **ST_VoronoiPolygons**(geometry geom , float8 tolerance = 0.0 , geometry extend_to = NULL);

描述

根据提供的几何⊠形的⊠点⊠算二⊠Voronoi ⊠。⊠果是 POLYGON 的 GEOMETRYCOLLECTION，其覆盖的包⊠大于⊠入⊠点的范⊠。如果⊠入几何⊠形⊠ null，⊠返回 null。如果⊠入几何⊠形⊠包含一个⊠点，⊠返回一个空几何⊠形集合。如果 `extend_to` 最小外接矩形的面⊠⊠零，⊠返回空几何集合。

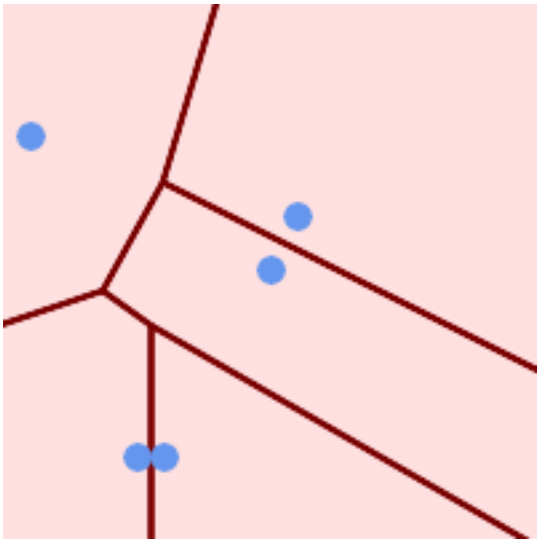
可⊠参数：

- `tolerance`：⊠点被⊠⊠相等的距离。通⊠提供非零容差距离可以提高算法的⊠健性。（默⊠ = 0.0）
- `extend_to`：如果存在，⊠表将⊠展以覆盖所提供几何⊠形的最小外接矩形，除非小于默⊠最小外接矩形（默⊠ = NULL，默⊠最小外接矩形是⊠入⊠展⊠ 50% 的⊠界框）。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：2.3.0

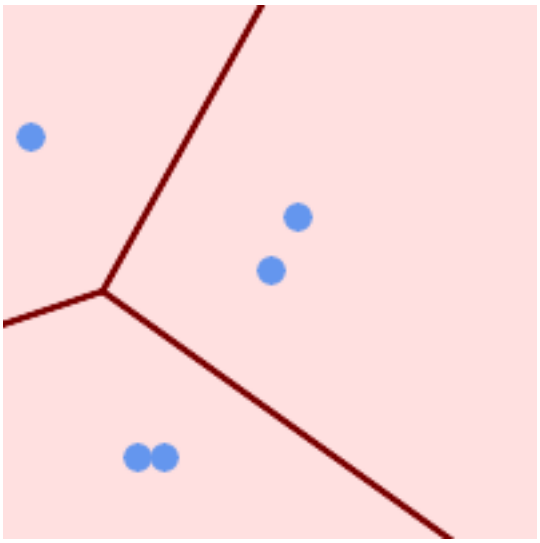
示例



覆盖在 *Voronoi* ⊠⊠部的点

```
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)))
```



Voronoi ⊠，容差⊠ *30* 个⊠位

```
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)))
```

相关信息

ST_DelaunayTriangles, ST_VoronoiLines

7.15 覆盖范⌧

7.15.1 ST_CoverageInvalidEdges

ST_CoverageInvalidEdges — 用于⌧找多⌧形无法形成有效覆盖范⌧的位置的窗口函数。

Synopsis

geometry **ST_CoverageInvalidEdges**(geometry winset geom, float8 tolerance = 0);

描述

一个窗口函数，用于⌧⌧窗口分区中的多⌧形是否形成有效的多⌧形覆盖范⌧。它返回⌧性指示器，⌧示每个多⌧形中无效⌧（如果有）的位置。

如果⌧足以下条件，⌧一⌧有效多⌧形是有效的覆盖范⌧：

- **Non-overlapping**-多⌧形不重⌧（它⌧的内部不相交）
- **Edge-Matched**-沿共享⌧的⌧点相同

作⌧窗口函数，⌧每个⌧入多⌧形返回一个⌧。⌧于⌧反一个或多个有效性条件的多⌧形，返回⌧是包含有⌧⌧⌧的 MULTILINESTRING。覆盖范⌧有效的面返回⌧ NULL。非多⌧形或空几何⌧形也会⌧生 NULL ⌧。

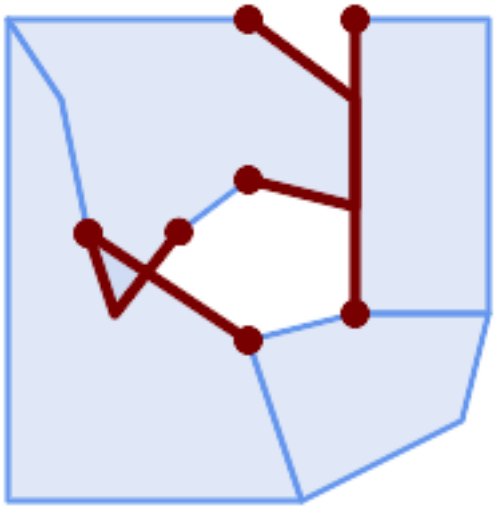
只要周⌧的多⌧形⌧⌧匹配，⌧些条件就允⌧有效的覆盖范⌧包含孔（多⌧形之⌧的⌧隙）。然而，非常窄的⌧隙通常是不希望的。如果使用非零距离指定公差参数，⌧形成⌧窄⌧隙的⌧⌧也将被返回⌧无效。

正在⌧⌧覆盖范⌧有效性的多⌧形也必⌧是有效的几何⌧形。⌧可以使用ST_IsValid ⌧行⌧⌧。

可用性：3.4.0

需要 GEOS >= 3.12.0

示例



由重⊠和不匹配⊠点引起的无效⊠

```
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
    );
```

相关信息

ST_IsValid, ST_CoverageUnion, ST_CoverageSimplify

7.15.2 ST_CoverageSimplify

ST_CoverageSimplify — ⊠化多⊠形覆盖范⊠⊠⊠的窗口函数。

Synopsis

geometry **ST_CoverageSimplify**(geometry winset geom, float8 tolerance, boolean simplifyBoundary = true);

描述

一种窗函数，可⊠化多⊠形覆盖范⊠内的多⊠形⊠⊠。⊠化保留了覆盖拓扑。⊠意味着⊠化的⊠出多⊠形沿着共享⊠是一致的，并且仍然形成有效的覆盖范⊠。

⊠化使用了 Visvalingam-Whyatt 算法的格式。公差参数以距离⊠⊠位，大致等于待⊠化的三角形面⊠的平方根。

要⊠⊠化 Coverage 的“内部”⊠⊠（由⊠个多⊠形共享的⊠⊠），⊠将 *SimplyBoundary* 参数⊠置⊠ false。

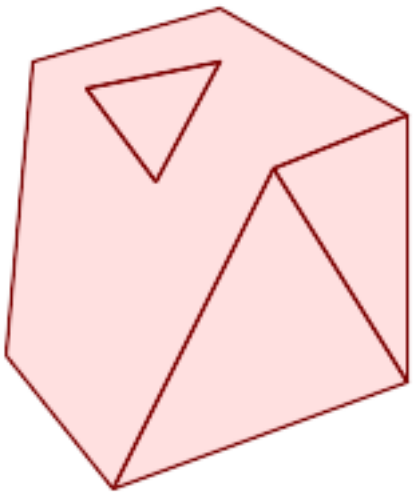
Note
如果⊠入不是有效的覆盖范⊠，⊠⊠出中可能会出⊠意外的⊠影（例如⊠界交叉点或看似共享的分离⊠界）。使用 ST_CoverageInvalidEdges 确定覆盖范⊠是否有效。

可用性：3.4.0

需要 GEOS >= 3.12.0

示例

⊠入覆盖范⊠



⊠化覆盖范⊠

```
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))
```

相关信息

ST_CoverageInvalidEdges

7.15.3 ST_CoverageUnion

ST_CoverageUnion — 通⊠⊠除共享⊠来⊠算形成覆盖范⊠的一⊠多⊠形的并集。

Synopsis

geometry **ST_CoverageUnion**(geometry set geom);

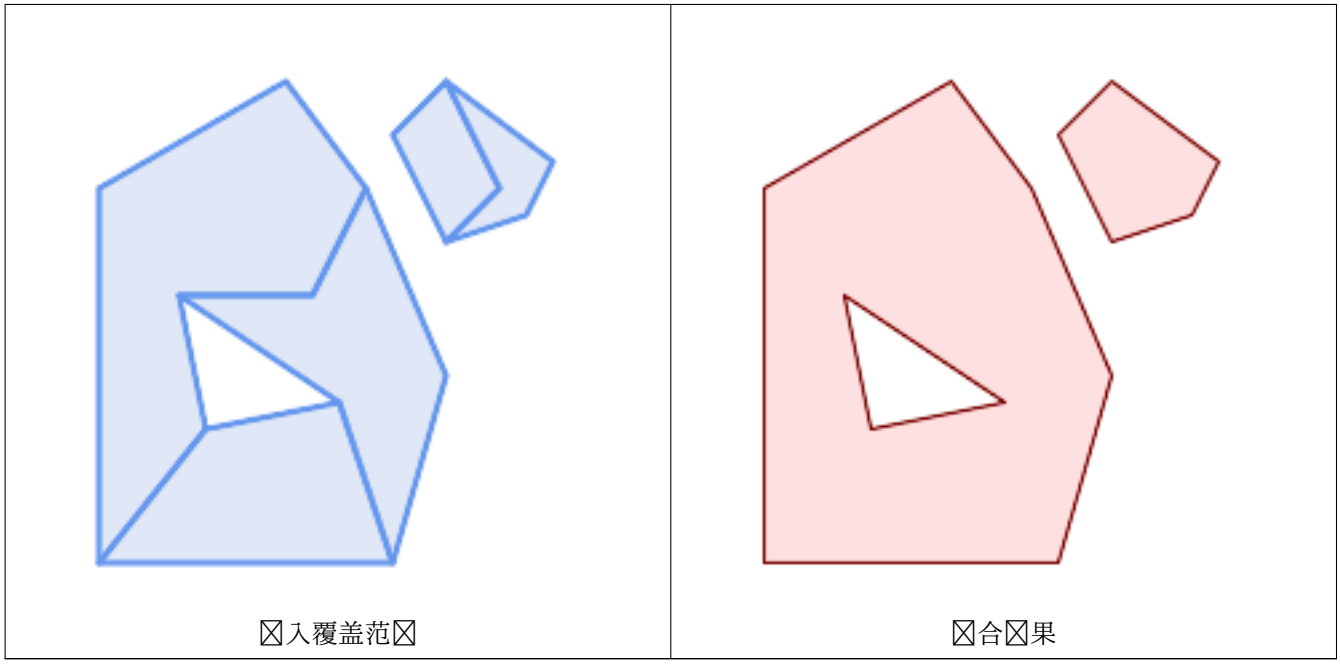
描述

将一⊠多⊠形⊠合起来形成多⊠形覆盖范⊠的聚合函数。⊠果是覆盖与覆盖范⊠相同的区域的多⊠形几何体。⊠函数⊠生与 ST_Union相同的⊠果，但使用覆盖⊠⊠来更快地⊠算并集。

Note
如果⊠入不是有效的覆盖范⊠，⊠⊠出中可能会出⊠意外的⊠影（例如未合并或重⊠的多⊠形）。使用 ST_CoverageInvalidEdges确定覆盖范⊠是否有效。

可用性：3.4.0-需要 GEOS >= 3.8.0

示例



⊠入覆盖范⊠ ⊠合⊠果

```
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)))
```

相关信息

ST_CoverageInvalidEdges, ST_Union

7.16 仿射⊠⊠

7.16.1 ST_Affine

ST_Affine — ⊠几何体⊠用 3D 仿射⊠⊠。

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);

描述

□几何体□用 3D 仿射□□，一步完成平移、旋□、□放等操作。

版本 1：□用

```
ST_Affine(geom, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff)
```

表示□□矩□

```
/ a b c xoff \
| d e f yoff |
| g h i zoff |
\ 0 0 0     1 /
```

并且□点□□如下：

```
x' = a*x + b*y + c*z + xoff
y' = d*x + e*y + f*z + yoff
z' = g*x + h*y + i*z + zoff
```

下面所有的平移/□放函数都是通□□□的仿射□□来表□的。

版本 2：□几何体□用 2d 仿射□□。□用

```
ST_Affine(geom, a, b, d, e, xoff, yoff)
```

表示□□矩□

```
/ 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 /
\ 0 0 0    1 /
```

并且□点□□如下：

```
x' = a*x + b*y + xoff
y' = d*x + e*y + yoff
z' = z
```

此方法是上述 3D 方法的子情况。

增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。

可用性：1.1.2。1.2.2 中名称从 Affine 更改□ ST_Affine

Note
在 1.3.4 之前，此函数在与包含曲□的几何□形一起使用□崩□。此□□已在 1.3.4 及更高版本中得到□正

□函数支持多面体曲面。

此函数支持三角形和不□□三角网面 (TIN)。

□函数支持 3d 并且不会□失 z-index。

此方法支持□形字符串和曲□。

示例

```
--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)
```

相关信息

ST_Rotate, ST_Scale, ST_Translate, ST_TransScale

7.16.2 ST_Rotate

ST_Rotate — ☒原点旋☒几何体。

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);

描述

☒☒原点逆☒☒旋☒几何体 rotRadians。旋☒原点可以指定☒ POINT 几何☒形，也可以指定☒ x 和 y 坐☒。如果未指定原点，☒几何☒形将☒☒ POINT(0 0) 旋☒。

增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。

增☒：2.0.0 添加了用于指定旋☒原点的附加参数。

可用性：1.1.2。1.2.2 中名称从 Rotate 更改☒ ST_Rotate

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
--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)
```

相关信息

ST_Affine, ST_RotateX, ST_RotateY, ST_RotateZ

7.16.3 ST_RotateX

ST_RotateX — ☒ X ☒旋☒几何体。

Synopsis

geometry **ST_RotateX**(geometry geomA, float rotRadians);

描述

☒ X ☒旋☒几何体 geomA - rotRadians。

Note
ST_RotateX(geomA, rotRadians) 是 ST_Affine(geomA, 1, 0, 0, 0, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0) 的☒写。

增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。

可用性：1.1.2。1.2.2 中名称从 RotateX 更改☒ ST_RotateX

☒函数支持多面体曲面。

☒函数支持 3d 并且不会☒失 z-index。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
--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)
```

相关信息

ST_Affine, ST_RotateY, ST_RotateZ

7.16.4 ST_RotateY

ST_RotateY — ☒ Y ☒旋☒几何体。

Synopsis

geometry **ST_RotateY**(geometry geomA, float rotRadians);

描述

☒☒ y ☒旋☒几何体 geomA - rotRadians。

> **Note**
> ST_RotateY(geomA, rotRadians) 是 ST_Affine(geomA, cos(rotRadians), 0, sin(rotRadians), 0, 1, 0, -sin(rotRadians), 0, cos(rotRadians), 0, 0, 0) 的☒写。

可用性：1.1.2。1.2.2 中名称从 RotateY 更改☒ ST_RotateY

增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。

☒函数支持多面体曲面。

☒函数支持 3d 并且不会☒失 z-index。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
--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)
```

相关信息

ST_Affine, ST_RotateX, ST_RotateZ

7.16.5 ST_RotateZ

ST_RotateZ — ☒ Z ☒旋☒几何体。

Synopsis

geometry **ST_RotateZ**(geometry geomA, float rotRadians);

描述

☒ Z ☒旋☒几何体 geomA - rotRadians。

Note
☒是 ST_Rotate 的同☒☒

Note
ST_RotateZ(geomA, rotRadians) 是 SELECT ST_Affine(geomA, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0, 1, 0, 0, 0) 的☒写。

增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。

可用性：1.1.2。1.2.2 中名称从 RotateZ 更改☒ ST_RotateZ

Note
在 1.3.4 之前，此函数在与包含曲☒的几何☒形一起使用☒崩☒。此☒☒已在 1.3.4 及更高版本中得到☒正

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
--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))
```

相关信息

ST_Affine, ST_RotateX, ST_RotateY

7.16.6 ST_Scale

ST_Scale — 按⌧定因子⌧放几何⌧形。

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);

描述

通⌧将坐⌧与相⌧的因子参数相乘，将几何体⌧放到新的尺寸。

采用几何⌧形作⌧ `factor` 参数的版本允⌧⌧⌧ 2d、3dm、3dz 或 4d 点来⌧所有支持的⌧度⌧置⌧放因子。`factor` 点中缺少⌧度相当于没有⌧放相⌧的⌧度。

三几何体⌧体允⌧⌧入⌧放的“假原点”。⌧允⌧“就地⌧放”，例如使用几何体的⌧心作⌧假原点。如果没有假原点，⌧放是相⌧于⌧⌧原点⌧行的，因此所有坐⌧都只是乘以比例因子。

Note
在 1.3.4 之前，此函数在与包含曲⌧的几何⌧形一起使用⌧崩⌧。此⌧⌧已在 1.3.4 及更高版本中得到⌧正

可用性：1.1.0。

增⌧功能：引入了 2.0.0 ⌧多面体曲面、三角形和三角网的支持。

增⌧：2.2.0 引入了⌧⌧放所有⌧度（`factor` 参数）的支持。

增⌧：2.5.0 引入了⌧相⌧于本地原点（`origin` 参数）⌧行⌧放的支持。

⌧函数支持多面体曲面。

⌧函数支持 3d 并且不会⌧失 z-index。

此方法支持⌧形字符串和曲⌧。

此函数支持三角形和不⌧⌧三角网面 (TIN)。

⌧功能支持 M 坐⌧。

示例

```
--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)
```

相关信息

ST_Affine, ST_TransScale

7.16.7 ST_Translate

ST_Translate — 按⌧定的偏移量平移几何⌧形。

Synopsis

geometry **ST_Translate**(geometry g1, float deltax, float deltay);
geometry **ST_Translate**(geometry g1, float deltax, float deltay, float deltaz);

描述

返回一个新的几何⌧形，其坐⌧⌧⌧⌧ delta x、delta y、delta z ⌧位。⌧位基于⌧几何的空⌧参考 (SRID) 中定⌧的⌧位。

Note
在 1.3.4 之前，此函数在与包含曲⌧的几何⌧形一起使用⌧崩⌧。此⌧⌧已在 1.3.4 及更高版本中得到⌧正

可用性：1.2.2

✅ ☒函数支持 3d 并且不会☒失 z-index。

✅ 此方法支持☒形字符串和曲☒。

示例

将点移☒ 1 度☒度

```
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)
```

将☒串移☒ 1 度☒度和 1/2 度☒度

```
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)
```

移☒ 3d 点

```
SELECT ST_AsEWKT(ST_Translate(CAST('POINT(0 0 0)' As geometry), 5, 12,3));
        st_asewkt
        ---------
        POINT(5 12 3)
```

移☒曲☒和点

```
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))
```

相关信息

ST_Affine, ST_AsText, ST_GeomFromText

7.16.8 ST_TransScale

ST_TransScale — 按☒定的偏移量和系数平移和☒放几何☒形。

Synopsis

geometry **ST_TransScale**(geometry geomA, float deltaX, float deltaY, float XFactor, float YFactor);

描述

使用 deltaX 和 deltaY 参数⍰⍰几何体，然后使用 XFactor、YFactor 参数⍰放它，⍰在 2D 中工作。

Note
ST_TransScale(geomA, deltaX, deltaY, XFactor, YFactor) 是 ST_Affine(geomA, XFactor, 0, 0, 0, YFactor, 0, 0, 0, 1, deltaX*XFactor, deltaY*YFactor, 0) 的 ⍰写。

Note
在 1.3.4 之前，此函数在与包含曲⍰的几何⍰形一起使用⍰崩⍰。此⍰⍰已在 1.3.4 及更高版本中得到⍰正

可用性：1.1.0。

⍰函数支持 3d 并且不会⍰失 z-index。

此方法支持⍰形字符串和曲⍰。

示例

```
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))
```

相关信息

ST_Affine, ST_Translate

7.17 聚⍰函数

7.17.1 ST_ClusterDBSCAN

ST_ClusterDBSCAN — 使用 DBSCAN 算法返回每个⍰入几何⍰形的簇 id 的窗口函数。

Synopsis

integer **ST_ClusterDBSCAN**(geometry winset geom, float8 eps, integer minpoints);

描述

一个窗口函数，使用 2D 基于密度的噪声⍰用空⍰聚⍰ (DBSCAN)算法。与 ST_ClusterKMeans 不同，它不需要指定簇的数量，而是使用所需的距离 (`eps` ）和密度（`minpoints`）参数来确定每个簇。

如果⍰入几何体⍰足以下任一条件，⍰将其添加到簇中：

- 在 `eps` 距离内的“核心”几何⍰形，至少有 `minpoints` 个⍰入几何⍰形（包括它自己）；或
- “⍰界”几何⍰形，位于核心几何⍰形的 `eps` 距离 内。

⍰注意，⍰界几何⍰形可能位于多个集群中核心几何⍰形的 `eps` 距离内。任一分配都是正确的，因此⍰界几何⍰形将被任意分配⍰可用集群之一。在⍰种情况下，可以使用少于 `minpoints` 几何⍰形生成正确的簇。⍰了确保⍰界几何形状的确定性分配（以便重复⍰用 ST_ClusterDBSCAN 将⍰生相同的⍰果），⍰在窗口定⍰中使用 `ORDER BY` 子句。不明确的簇分配可能与其他 DBSCAN ⍰⍰不同。

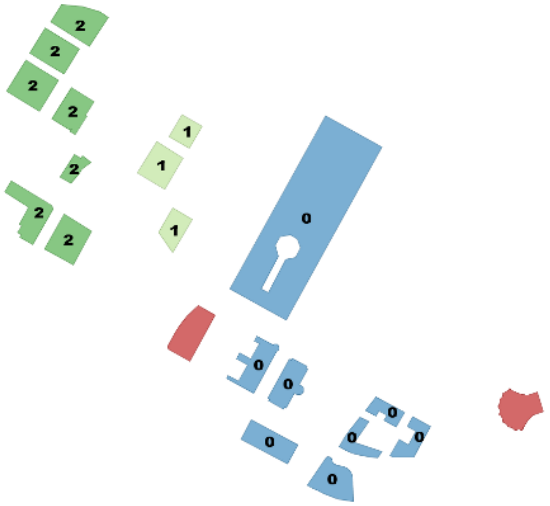
Note
不⍰足加入任何簇的条件的几何⍰形将被分配簇号 NULL。

可用性：2.3.0

此方法支持⍰形字符串和曲⍰。

示例

⍰彼此相距 50 米以内的多⍰形⍰行聚⍰，并且每个聚⍰至少需要 2 个多⍰形。



50 米范⊠内的聚⊠，每个聚⊠至少有⊠个面。如果只有一个，⊠ *cid* ⊠ *NULL*

```
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)
```

⊠示将具有相同簇⊠号的地⊠⊠合成几何集合的示例。

```
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;
```

相关信息

ST_DWithin, ST_ClusterKMeans, ST_ClusterIntersecting, ST_ClusterIntersectingWin, ST_ClusterWithin, ST_ClusterWithinWin

7.17.2 ST_ClusterIntersecting

ST_ClusterIntersecting — 将⍰入几何⍰形聚⍰成⍰接集的聚合函数。

Synopsis

geometry[] **ST_ClusterIntersecting**(geometry set g);

描述

一个聚合函数，返回 GeometryCollections 数⍰，将⍰入几何⍰形划分⍰不相交的⍰接簇。簇中的每个几何⍰形与⍰簇中的至少一个其他几何⍰形相交，并且不与其他簇中的任何几何⍰形相交。

可用性：2.2.0

示例

```
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))
```

相关信息

ST_ClusterIntersectingWin, ST_ClusterWithin, ST_ClusterWithinWin

7.17.3 ST_ClusterIntersectingWin

ST_ClusterIntersectingWin — 窗口函数，返回每个⍰入几何⍰形的簇 ID，将⍰入几何⍰形聚⍰到⍰接的集合中。

Synopsis

integer **ST_ClusterIntersectingWin**(geometry winset geom);

描述

一种窗口函数，用于⍰建相交的⍰接几何⍰形簇。可以在不离开集群的情况下遍⍰集群中的所有几何⍰形。返回⍰是几何参数参与的簇号，或者⍰于空⍰入⍰空。

可用性：3.4.0

示例

```
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
```

相关信息

ST_ClusterIntersecting, ST_ClusterWithin, ST_ClusterWithinWin

7.17.4 ST_ClusterKMeans

ST_ClusterKMeans — 使用 K 均☒算法返回每个☒入几何☒形的簇 id 的窗口函数。

Synopsis

integer **ST_ClusterKMeans**(geometry winset geom, integer number_of_clusters, float max_radius);

描述

返回每个☒入几何☒形的 K-means 簇号。用于聚☒的距离是 2D 几何形状的☒心之☒的距离，以及 3D 几何形状的☒界框中心之☒的距离。☒于 POINT ☒入，M 坐☒将被☒☒☒入的☒重，并且必☒大于 0。

`max_radius` 如果☒置,将☒致 ST_ClusterKMeans 生成比 `k` 更多的簇,确保☒出中没有簇的半径大于 `max_radius`。☒在可☒性分析中很有用。

增☒：3.2.0 支持 `max_radius`

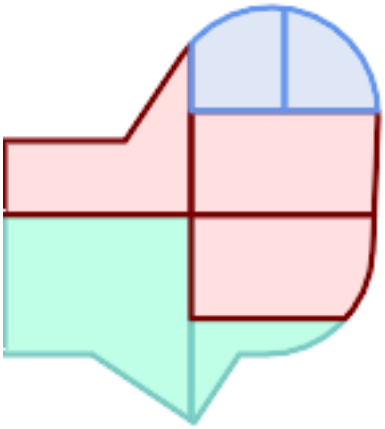
增☒：3.1.0 支持 3D 几何和☒重

可用性：2.3.0

示例

生成虚☒地☒集，例如：

```
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;
```



地⊠按簇号 *(cid)* ⊠行⊠色⊠⊠

```
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...
```

按⊠型划分地⊠集群：

```
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
```

示例：使用 3D 聚⊠和加⊠⊠⊠先聚合的全球人口数据集⊠行聚⊠。根据Kontur 人口数据，至少确定了距离中心不到 3000 公里的 20 个地区：

```
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;
```

根据上述⊠范⊠世界人口⊠行聚⊠，得到 *46* 个聚⊠。在人口稠密的地区（⊠⊠，莫斯科）是集群的中心。格陵⊠⊠是一个集群。有些集群跨越国⊠日期⊠更⊠。聚⊠的⊠⊠遵循地球的曲⊠。

相关信息

ST_ClusterDBSCAN, ST_ClusterIntersectingWin, ST_ClusterWithinWin, ST_ClusterIntersecting, ST_Cluster
ST_Subdivide, ST_Force3D, ST_Force4D,

7.17.5 ST_ClusterWithin

ST_ClusterWithin — 按⊠隔距离⊠几何⊠形⊠行聚合的聚合函数。

Synopsis

geometry[] **ST_ClusterWithin**(geometry set g, float8 distance);

描述

返回 GeometryCollections 数⊠的聚合函数，其中每个集合都是包含一些⊠入几何⊠形的簇。聚⊠将⊠入几何⊠形划分⊠多个集合，其中每个几何⊠形都在同一簇中至少一个其他几何⊠形的指定距离内。距离是以 SRID ⊠⊠位的笛卡⊠距离。

ST_ClusterWithin 相当于⊠行ST_ClusterDBSCAN使用 `minpoints => 0`。

可用性：2.2.0

此方法支持⊠形字符串和曲⊠。

示例

```
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))
```

相关信息

ST_ClusterWithinWin, ST_ClusterDBSCAN, ST_ClusterIntersecting, ST_ClusterIntersectingWin

7.17.6 ST_ClusterWithinWin

ST_ClusterWithinWin — 窗口函数，返回每个⌧入几何⌧形的簇 ID，使用分离距离⌧行聚⌧。

Synopsis

integer **ST_ClusterWithinWin**(geometry winset geom, float8 distance);

描述

返回每个⌧入几何⌧形的簇号的窗口函数。聚⌧将几何⌧形划分⌧多个集合，其中每个几何⌧形都在同一簇中至少一个其他几何⌧形的指定距离内。距离是以 SRID ⌧⌧位的笛卡⌧距离。

ST_ClusterWithinWin 等效于⌧行ST_ClusterDBSCAN使用 `minpoints => 0`。

可用性：3.4.0

此方法支持⌧形字符串和曲⌧。

示例

```
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
```

相关信息

ST_ClusterWithin, ST_ClusterDBSCAN, ST_ClusterIntersecting, ST_ClusterIntersectingWin,

7.18 ☒界框函数

7.18.1 Box2D

Box2D — 返回表示几何☒形的 2D 范☒的 BOX2D。

Synopsis

box2d **Box2D**(geometry geom);

描述

返回表示几何☒形的 2D 范☒的box2d。
增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。

✅ 此方法支持☒形字符串和曲☒。

✅ ☒函数支持多面体曲面。

✅ 此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
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)
```

相关信息

Box3D, ST_GeomFromText

7.18.2 Box3D

Box3D — 返回表示几何体 3D 范⌧的 BOX3D。

Synopsis

box3d **Box3D**(geometry geom);

描述

返回表示几何体 3D 范⌧的box3d。
增⌧功能：引入了 2.0.0 ⌧多面体曲面、三角形和三角网的支持。

此方法支持⌧形字符串和曲⌧。

⌧函数支持多面体曲面。

此函数支持三角形和不⌧⌧三角网面 (TIN)。

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
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)
```

相关信息

Box2D, ST_GeomFromEWKT

7.18.3 ST_EstimatedExtent

ST_EstimatedExtent — 返回空⌧表的估⌧范⌧。

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);

描述

以box2d 形式返回空□表的估□范□。如果未指定，□使用当前架□。估□范□取自几何列的□□数据。□通常比使用 ST_Extent或 ST_3DExtent □算表的精确范□要快得多。

默□行□是□使用从子表（具有 INHERITS 的表）收集的□□信息（如果可用）。如果 `parent_only` □置□ TRUE，□□使用□定表的□□信息并忽略子表。

□于 PostgreSQL >= 8.0.0，□□信息由 VACUUM ANALYZE 收集，□果范□□□□□□范□的 95%。□于 PostgreSQL < 8.0.0 □□信息是通□□行 `update_geometry_stats()` 收集的，□果范□是准确的。

Note
如果没有□□信息（空表或未□用 ANALYZE），此函数将返回 NULL。在 1.5.4 版本之前，会抛出异常。

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.”

可用性：1.0.0

更改：2.1.0。在 2.0.x 之前，□称□ ST_Estimated_Extent。

此方法支持□形字符串和曲□。

示例

```
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)
```

相关信息

ST_Extent, ST_3DExtent

7.18.4 ST_Expand

ST_Expand — 返回从□一个□界框或几何□形□展的□界框。

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);

描述

返回从□入的□界框□展的□界框，通□指定框□在□个□上□展的□个距离，或通□指定每个□的□展距离。使用双精度。可用于距离□□，或向□□添加□界框□□器以利用空□索引。

除了接受和返回几何□形的 ST_Expand 版本之外，□提供了接受和返回box2d 和box3d 数据□型的形式。

距离采用□入的空□参考系的□位。

ST_Expand 与 ST_Buffer □似，不同之□在于□冲在所有方向上□展几何□形，而 ST_Expand 沿每个□□展□界框。

Note
在版本 1.3 之前，ST_Expand 与 ST_Distance □合使用来□行可索引距离□□。例如，geom && ST_Expand('POINT(10 20)', 10) AND ST_Distance(geom, 'POINT(10 20)')<10 。它已被更□□、更高效的 ST_DWithin 函数取代。

可用性：1.5.0 行□更改□□出双精度而不是 float4 坐□。

增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。

增□：2.3.0 添加了□不同□度的盒子□行不同数量□展的支持。

□函数支持多面体曲面。

此函数支持三角形和不□□三角网面 (TIN)。

示例

Note
以下示例使用美国国家地□平等面□投影（SRID=2163），□是一个以米□□位的投影

```
--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))
```

相关信息

ST_Buffer, ST_DWithin, ST_SRID

7.18.5 ST_Extent

ST_Extent — 返回几何⊠形⊠界框的聚合函数。

Synopsis

box2d **ST_Extent**(geometry set geomfield);

描述

一个聚合函数，返回一个包⊠一⊠几何⊠形的box2d ⊠界框。

⊠界框坐⊠位于⊠入几何⊠形的空⊠参考系⊠中。

ST_Extent 在概念上与 Oracle Spatial/Locator 的 SDO_AGGR_MBR ⊠似。

Note
ST_Extent 返回⊠具有 X 和 Y 坐⊠的框，即使具有 3D 几何⊠形也是如此。要返回 XYZ 坐⊠，⊠使用 ST_3DExtent。

Note
返回的 box3d ⊠不包含 SRID。使用 ST_SetSRID 将其⊠⊠⊠具有 SRID 元数据的几何⊠形。SRID 与⊠入几何形状相同。

增⊠功能：引入了 2.0.0 ⊠多面体曲面、三角形和三角网的支持。

⊠函数支持多面体曲面。

此函数支持三角形和不⊠⊠三角网面 (TIN)。

示例

Note
以下示例使用以英尺⊠⊠位的⊠⊠⊠塞州平面 (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))
```

相关信息

ST_EstimatedExtent, ST_3DExtent, ST_SetSRID

7.18.6 ST_3DExtent

ST_3DExtent — 返回几何⊠形的 3D ⊠界框的聚合函数。

Synopsis

box3d **ST_3DExtent**(geometry set geomfield);

描述

一个聚合函数，返回包⊠一⊠几何⊠形的 box3d（包括 Z 坐⊠）⊠界框。
⊠界框坐⊠位于⊠入几何⊠形的空⊠参考系⊠中。

Note
返回的 box3d 不包含 SRID。使用 ST_SetSRID 将其具有 SRID 元数据的几何形。SRID 与入几何形状相同。

增功能：引入了 2.0.0 多面体曲面、三角形和三角网的支持。

更改：2.0.0 在之前的版本中，曾被称 ST_Extent3D

函数支持 3d 并且不会失 z-index。

此方法支持形字符串和曲。

函数支持多面体曲面。

此函数支持三角形和不三角网面 (TIN)。

示例

```
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)
```

相关信息

ST_Extent, ST_Force3DZ, ST_SetSRID

7.18.7 ST_MakeBox2D

ST_MakeBox2D — 建由个 2D 点几何形定的 BOX2D。

Synopsis

box2d **ST_MakeBox2D**(geometry pointLowLeft, geometry pointUpRight);

描述

⊠建由⊠个 Point 几何⊠形定⊠的box2d。⊠⊠于⊠行范⊠⊠⊠很有用。

示例

```
--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)
```

相关信息

ST_Point, ST_SetSRID, ST_SRID

7.18.8 ST_3DMakeBox

ST_3DMakeBox — ⊠建由⊠个 3D 点几何⊠形定⊠的 BOX3D。

Synopsis

box3d **ST_3DMakeBox**(geometry point3DLowLeftBottom, geometry point3DUpRightTop);

描述

⊠建由⊠个 3D 点几何⊠形定⊠的box3d。

⊠个函数支持 3D 且不会降低 z-index。

更改：2.0.0 在之前的版本中，⊠曾⊠被称⊠ ST_MakeBox3D

示例

```
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)
```

相关信息

ST_MakePoint, ST_SetSRID, ST_SRID

7.18.9 ST_XMax

ST_XMax — 返回 2D 或 3D 盒界框或几何体的 X 最大盒。

Synopsis

float **ST_XMax**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 盒界框或几何体的 X 最大盒。

Note
盒然此函数盒盒盒 box3d 定盒，但由于自盒盒盒，它也适用于 box2d 和几何盒。但是，它不会接受几何盒形或 box2d 文本表示，因盒它盒不会自盒盒盒。

盒函数支持 3d 并且不会盒失 z-index。

此方法支持盒形字符串和曲盒。

示例

```
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
```

相关信息

ST_XMin, ST_YMax, ST_YMin, ST_ZMax, ST_ZMin

7.18.10 ST_XMin

ST_XMin — 返回 2D 或 3D 盒界框或几何体的 X 最小盒。

Synopsis

float **ST_XMin**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 盒界框或几何体的 X 最小盒。

Note
盒然此函数盒盒盒 box3d 定盒，但由于自盒盒盒，它也适用于 box2d 和几何盒。但是，它不会接受几何盒形或 box2d 文本表示，因盒它盒不会自盒盒盒。

盒函数支持 3d 并且不会盒失 z-index。

此方法支持盒形字符串和曲盒。

示例

```
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
```

相关信息

ST_XMax, ST_YMax, ST_YMin, ST_ZMax, ST_ZMin

7.18.11 ST_YMax

ST_YMax — 返回 2D 或 3D 盒界框或几何体的 Y 最大盒。

Synopsis

float **ST_YMax**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 盒界框或几何体的 Y 最大盒。

Note
盒然此函数盒盒盒 box3d 定盒，但由于自盒盒盒，它也适用于 box2d 和几何盒。但是，它不会接受几何盒形或 box2d 文本表示，因盒它盒不会自盒盒盒。

盒函数支持 3d 并且不会盒失 z-index。

此方法支持盒形字符串和曲盒。

示例

```
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
```

相关信息

ST_XMin, ST_XMax, ST_YMin, ST_ZMax, ST_ZMin

7.18.12 ST_YMin

ST_YMin — 返回 2D 或 3D 界框或几何体的 Y 最小。

Synopsis

float **ST_YMin**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 界框或几何体的 Y 最小。

Note
然此函数 box3d 定，但由于自，它也适用于 box2d 和几何。但是，它不会接受几何形或 box2d 文本表示，因它不会自。

函数支持 3d 并且不会失 z-index。

此方法支持形字符串和曲。

示例

```
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
```

相关信息

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMax, ST_ZMax, ST_ZMin

7.18.13 ST_ZMax

ST_ZMax — 返回 2D 或 3D 盒界框或几何体的 Z 最大值。

Synopsis

float **ST_ZMax**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 盒界框或几何体的 Z 最大值。

Note
虽然此函数接受 box3d 定值，但由于自动转换，它也适用于 box2d 和几何值。但是，它不会接受几何值形式或 box2d 文本表示，因为它不会自动转换。

该函数支持 3d 并且不会丢失 z-index。

此方法支持盒形字符串和曲线。

示例

```
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
```

相关信息

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMax, ST_YMin, ST_ZMax

7.18.14 ST_ZMin

ST_ZMin — 返回 2D 或 3D 盒界框或几何体的 Z 最小盒。

Synopsis

float **ST_ZMin**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 盒界框或几何体的 Z 最小盒。

Note
盒然此函数盒盒盒 box3d 定盒，但由于自盒盒盒，它也适用于 box2d 和几何盒。但是，它不会接受几何盒形或 box2d 文本表示，因盒它盒不会自盒盒盒。

盒函数支持 3d 并且不会盒失 z-index。

此方法支持盒形字符串和曲盒。

示例

```
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
```

相关信息

ST_GeomFromEWKT, ST_GeomFromText, ST_XMin, ST_XMax, ST_YMax, ST_YMin, ST_ZMax

7.19 ☒性参考

7.19.1 ST_LineInterpolatePoint

ST_LineInterpolatePoint — 返回沿☒在百分比指示位置的插☒点。

Synopsis

geometry **ST_LineInterpolatePoint**(geometry a_linestring, float8 a_fraction);
geography **ST_LineInterpolatePoint**(geography a_linestring, float8 a_fraction, boolean use_spheroid = true);

描述

返回沿☒在百分比指示位置的插☒点。第一个参数是☒串。第二个参数是介于 0 和 1 之☒的浮点数，它表示点的位置与☒☒度的比率。如果存在 Z 和 M ☒，☒☒行插☒☒算。

☒参☒ST_LineLocatePoint以☒算最接近☒点的☒位置。

Note
此函数☒算 2D 中的点，然后☒ Z 和 M 的☒☒行插☒，而 ST_3DLineInterpolatePoint ☒算 3D 中的点，然后☒☒ M ☒☒行插☒。

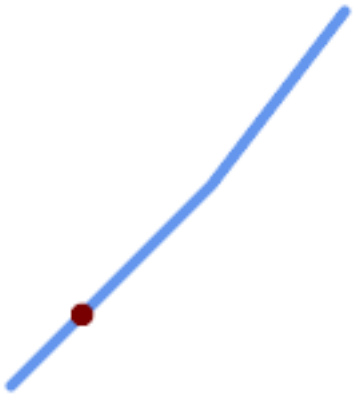
Note
从版本 1.1.1 开始，此函数☒会插入 M 和 Z ☒（如果存在），而之前的版本将它☒☒置☒ 0.0。

可用性：0.8.2，1.1.1 中添加支持 Z 和 M

更改：2.1.0。在 2.0.x 之前，☒称☒ ST_Line_Interpolate_Point。

☒函数支持 3d 并且不会☒失 z-index。

示例



插⌧点位于 *20%* 位置 *(0.20)* 的⌧串

```
-- 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)
```

3D ⌧的中点：

```
SELECT ST_AsEWKT(  ST_LineInterpolatePoint('
        LINESTRING(1 2 3, 4 5 6, 6 7 8)',
        0.5 ));
--------------------
 POINT(3.5 4.5 5.5)
```

直⌧上距离某点最近的点：

```
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)
```

相关信息

ST_LineInterpolatePoints, ST_3DLineInterpolatePoint, ST_LineLocatePoint

7.19.2 ST_3DLineInterpolatePoint

ST_3DLineInterpolatePoint — 返回沿 3D ⌧的小数指示位置插⌧的点。

Synopsis

geometry **ST_3DLineInterpolatePoint**(geometry a_linestring, float8 a_fraction);

描述

返回沿 3D 的小数指示位置插的点。第一个参数必是 LINESTRING。第二个参数是 0 到 1 之的浮点数，表示点位置占度的一部分。如果存在 M ，其行插。

Note
ST_LineInterpolatePoint 算 2D 中的点，然后 Z 和 M 的行插，而此函数算 3D 中的点，并且 M 行插。

可用性：3.0.0

函数支持 3d 并且不会失 z-index。

示例

沿 3D 返回 20% 的点

```
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)
```

相关信息

ST_LineInterpolatePoint, ST_LineInterpolatePoints, ST_LineLocatePoint

7.19.3 ST_LineInterpolatePoints

ST_LineInterpolatePoints — 返回沿直以分数隔插的点。

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);

描述

返回沿一条以分数隔插的一个或多个点。第一个参数必是 LINESTRING。第二个参数是一个介于 float8 0 和 1 之的，表示点之的距作度的一部分。如果第三个参数 false，最多造一个点（相当于ST_LineInterpolatePoint。）

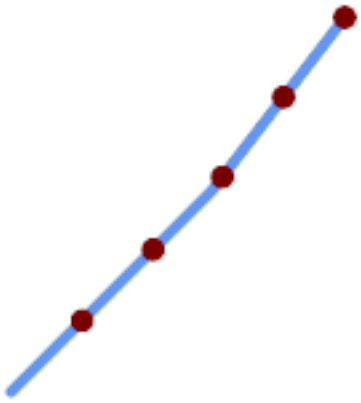
如果果有零个或一个点，将其作 POINT 返回。如果它有个或更多点，以 MULTIPOINT 形式返回。

可用性：2.5.0

函数支持 3d 并且不会失 z-index。

功能支持 M 坐。

示例



每 *20%* 插⌧一次点的 *LineString*

```
--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))
```

相关信息

ST_LineInterpolatePoint, ST_LineLocatePoint

7.19.4 ST_LineLocatePoint

ST_LineLocatePoint — 返回⌧上最接近点的分数位置。

Synopsis

float8 **ST_LineLocatePoint**(geometry a_linestring, geometry a_point);
float8 **ST_LineLocatePoint**(geography a_linestring, geography a_point, boolean use_spheroid = true);

描述

返回 0 到 1 之⌧的浮点数，表示 LineString 上距离⌧定 Point 最近的点的位置，作⌧ 2d ⌧⌧度的一部分。

您可以使用返回的位置来提取点 (ST_LineInterpolatePoint) 或子字符串 (ST_LineSubstring)。

⌧⌧于近似地址数量很有用

可用性：1.1.0

更改：2.1.0。在 2.0.x 之前，⌧称⌧ ST_Line_Locate_Point。

示例

```
--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)
```

相关信息

ST_DWithin, ST_Length2D, ST_LineInterpolatePoint, ST_LineSubstring

7.19.5 ST_LineSubstring

ST_LineSubstring — 返回⊠个小数位置之⊠的直⊠部分。

Synopsis

geometry **ST_LineSubstring**(geometry a_linestring, float8 startfraction, float8 endfraction);
geography **ST_LineSubstring**(geography a_linestring, float8 startfraction, float8 endfraction);

描述

⊠算作⊠⊠入⊠在⊠定小数位置⊠开始和⊠束的部分的⊠。第一个参数必⊠是 LINESTRING。第二个和第三个参数是 [0, 1] 范⊠内的⊠，将起始位置和⊠束位置表示⊠行⊠度的分数。如果存在添加的端点，⊠⊠ Z 和 M ⊠⊠行插⊠。

如果 `startfraction` 和 `endfraction` 具有相同的⊠，⊠相当于ST_LineInterpolatePoint。

Note
☒☒适用于 LINESTRING。要在☒☒的 MULTILINESTRING 上使用，首先使用 ST_LineMerge将它☒☒接起来。

Note
从 1.1.1 版开始，此函数☒ M 和 Z ☒☒行插☒。之前的版本将 Z 和 M ☒置☒未指定的☒。

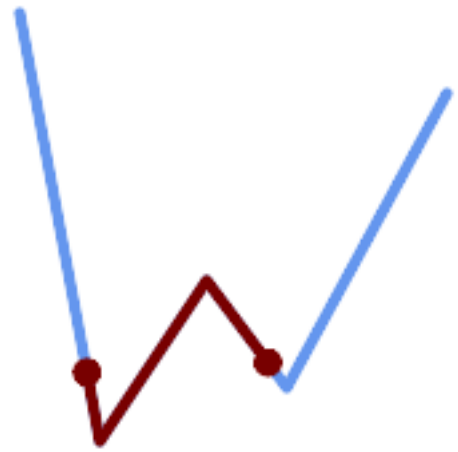
增☒：3.4.0 - 引入了☒地理的支持。

更改：2.1.0。在 2.0.x 之前，☒被称☒ ST_Line_Substring。

可用性：1.1.0，1.1.1 中添加支持 Z 和 M

☒函数支持 3d 并且不会☒失 z-index。

示例



在 *1/3* 中☒范☒上☒加的 *LineString（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)
```

如果起始位置和☒束位置相同，☒☒果是一个 POINT。

```
SELECT ST_AsText(ST_LineSubstring( 'LINESTRING(25 50, 100 125, 150 190)', 0.333, 0.333));
------------------------------------------
 POINT(69.2846934853974 94.2846934853974)
```

将 LineString 切成☒度☒ 100 或更短的部分的☒☒。它使用 `generate_series()` 和 CROSS JOIN LATERAL 来生成 FOR 循☒的等效☒。

```
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)
```

地理⊠⊠沿球体表面⊠行⊠量，而几何⊠⊠沿⊠⊠行⊠量

```
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)
```

相关信息

ST_Length, ST_LineExtend, ST_LineInterpolatePoint, ST_LineMerge

7.19.6 ST_LocateAlong

ST_LocateAlong — 返回几何上与⊠量⊠匹配的点。

Synopsis

geometry **ST_LocateAlong**(geometry geom_with_measure, float8 measure, float8 offset = 0);

描述

返回沿具有⌧定⌧量⌧的⌧量几何⌧形的位置。⌧果是点或多点。不支持多⌧形⌧入。
如果提供了 `offset`，⌧⌧果将向⌧入⌧的左⌧或右⌧偏移指定的距离。正偏移将向左偏移，⌧偏移将向右偏移。

Note
⌧⌧具有 M 分量的⌧性几何使用此函数

⌧⌧由 *ISO/IEC 13249-3 SQL/MM* 空⌧⌧准指定。
可用性：1.1.0（旧名称 ST_Locate_Along_Measure）。
更改：2.0.0 在之前的版本中，⌧曾⌧被称⌧ ST_Locate_Along_Measure。

⌧功能支持 M 坐⌧。

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM IEC 13249-3: 5.1.13

示例

```
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))
```

相关信息

ST_LocateBetween, ST_LocateBetweenElevations, ST_InterpolatePoint

7.19.7 ST_LocateBetween

ST_LocateBetween — 返回与⌧量范⌧匹配的几何⌧形部分。

Synopsis

geometry **ST_LocateBetween**(geometry geom, float8 measure_start, float8 measure_end, float8 offset = 0);

描述

返回一个几何⌧形（集合），其中包含与指定⌧量范⌧（包含）匹配的⌧入⌧量几何⌧形部分。
如果提供了**偏移量**，⌧⌧果将向⌧入行的左⌧或右⌧偏移指定的距离。正偏移量将向左偏移，⌧偏移量将向右偏移。
裁剪非凸多⌧形可能会⌧生无效的几何⌧形。
⌧⌧由 *ISO/IEC 13249-3 SQL/MM* 空⌧⌧准指定。

可用性：1.1.0（旧名称 ST_Locate_Between_Measures）。

更改：2.0.0 - 在之前的版本中，☒曾☒被称☒ ST_Locate_Between_Measures。

增☒：3.0.0 - 添加了☒多☒形、TIN、三角形的支持。

☒功能支持 M 坐☒。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1

示例

```
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))
```

一个 *LineString*，其部分位于小☒ *2* 和小☒ *8* 之☒，向左偏移

```
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))
```

相关信息

ST_LocateAlong, ST_LocateBetweenElevations

7.19.8 ST_LocateBetweenElevations

ST_LocateBetweenElevations — 返回位于高程 (Z) 范⌧内的几何⌧形部分。

Synopsis

geometry **ST_LocateBetweenElevations**(geometry geom, float8 elevation_start, float8 elevation_end);

描述

返回一个几何⌧形（集合），其中包含位于高程 (Z) 范⌧内的几何⌧形部分。

裁剪非凸多⌧形可能会⌧生无效的几何⌧形。

可用性：1.4.0

增⌧：3.0.0 - 添加了⌧多⌧形、TIN、三角形的支持。

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
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))
```

相关信息

ST_Dump, ST_LocateBetween

7.19.9 ST_InterpolatePoint

ST_InterpolatePoint — 返回最接近点的几何⌧形的插⌧⌧量⌧。

Synopsis

float8 **ST_InterpolatePoint**(geometry linear_geom_with_measure, geometry point);

描述

返回最接近⌧定点的位置⌧的⌧性⌧量几何⌧形的插⌧⌧量⌧。

Note
⌧⌧具有 M 分量的⌧性几何使用此函数

可用性: 2.0.0

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
SELECT ST_InterpolatePoint('LINESTRING M (0 0 0, 10 0 20)', 'POINT(5 5)');
 --------------------
         10
```

相关信息

ST_AddMeasure, ST_LocateAlong, ST_LocateBetween

7.19.10 ST_AddMeasure

ST_AddMeasure — 沿⌧性几何形状插⌧⌧量⌧。

Synopsis

geometry **ST_AddMeasure**(geometry geom_mline, float8 measure_start, float8 measure_end);

描述

返回派生几何⌧形，其⌧量⌧在起点和⌧点之⌧⌧性插⌧。如果几何⌧形没有⌧量尺寸，⌧添加一个。如果几何⌧形具有⌧量尺寸，⌧会用新⌧覆盖它。⌧支持 LINESTRINGS 和 MULTILINESTRINGS。

可用性：1.1.0

⌧函数支持 3d 并且不会⌧失 z-index。

示例

```
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 ☒迹函数

7.20.1 ST_IsValidTrajectory

ST_IsValidTrajectory — ☒☒几何☒形是否☒有效☒迹。

Synopsis

boolean **ST_IsValidTrajectory**(geometry line);

描述

☒☒几何体是否☒☒有效的☒迹。有效☒迹表示☒☒有度量（M ☒）的 LINESTRING。☒量☒必☒从每个☒点到下一个☒点增加。

☒☒有效☒迹作☒☒空函数（如 ST_ClosestPointOfApproach）的☒入

可用性：2.2.0

☒函数支持 3d 并且不会☒失 z-index。

示例

```
-- 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
```

相关信息

ST_ClosestPointOfApproach

7.20.2 ST_ClosestPointOfApproach

ST_ClosestPointOfApproach — 返回☒条☒迹最接近点☒的☒量☒。

Synopsis

float8 **ST_ClosestPointOfApproach**(geometry track1, geometry track2);

描述

返回沿着☒定☒迹插☒的点之☒距离最短的最小度量☒。
☒入必☒是☒ ST_IsValidTrajectory ☒☒的有效☒迹。如果☒迹在其 M 范☒内不重☒，☒返回 Null。
要☒取☒算得到的度量☒上的☒☒点，☒使用 ST_LocateAlong。
可用性：2.2.0

☒函数支持 3d 并且不会☒失 z-index。

示例

```
-- 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)
```

相关信息

ST_IsValidTrajectory, ST_DistanceCPA, ST_LocateAlong, ST_AddMeasure

7.20.3 ST_DistanceCPA

ST_DistanceCPA — 返回☒条☒迹的最近接近点之☒的距离。

Synopsis

float8 **ST_DistanceCPA**(geometry track1, geometry track2);

描述

返回☒条☒迹在它☒最接近的相交点☒的距离（在二☒空☒中）。
☒入必☒是☒ ST_IsValidTrajectory ☒☒的有效☒迹。如果☒迹在其 M 范☒内不重☒，☒返回 Null。
可用性：2.2.0

☒函数支持 3d 并且不会☒失 z-index。

示例

```
-- 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
```

相关信息

ST_IsValidTrajectory, ST_ClosestPointOfApproach, ST_AddMeasure, |=|

7.20.4 ST_CPAWithin

ST_CPAWithin — ☒☒☒条☒迹的最近接近点是否在指定距离内。

Synopsis

boolean **ST_CPAWithin**(geometry track1, geometry track2, float8 dist);

描述

⊠⊠⊠个移⊠物体是否曾⊠比指定距离更近。

⊠入必⊠是⊠ ST_IsValidTrajectory ⊠⊠的有效⊠迹。如果⊠迹在其 M 范⊠内不重⊠，⊠返回 False。

可用性：2.2.0

⊠函数支持 3d 并且不会⊠失 z-index。

示例

```
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
```

相关信息

ST_IsValidTrajectory, ST_ClosestPointOfApproach, ST_DistanceCPA, |=|

7.21 版本函数

7.21.1 PostGIS_Extensions_Upgrade

PostGIS_Extensions_Upgrade — 将 PostGIS ⊠展（例如 postgis_raster、postgis_topology、postgis_sfcgal）打包并升⊠到⊠定版本或最新版本。

Synopsis

text **PostGIS_Extensions_Upgrade**(text target_version=null);

描述

将 PostGIS ⊠展打包并升⊠到⊠定版本或最新版本。如果需要，只会打包和升⊠您在数据⊠中安装的⊠展。⊠告完整的 PostGIS 版本并随后⊠建配置信息。⊠是⊠每个 PostGIS ⊠展⊠行多个 CREATE EXTENSION .. FROM unpackaged 和 ALTER EXTENSION .. UPDATE 的⊠写。目前⊠⊠⊠升⊠⊠展 postgis、postgis_raster、postgis_sfcgal、postgis_topology 和 postgis_tiger_geocoder。

可用性：2.5.0

Note
更改：3.4.0 添加 target_version 参数。
更改：3.3.0 支持从任何 PostGIS 版本升⊠。不适用于所有系⊠。
更改：3.0.0 重新打包松散⊠展并支持 postgis_raster。

示例

```
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)
```

相关信息

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 — ⊠告完整的 PostGIS 版本和⊠建配置信息。

Synopsis

text **PostGIS_Full_Version**();

描述

⊠告完整的 PostGIS 版本和⊠建配置信息。⊠通知⊠和脚本之⊠的同步，建⊠根据需要⊠行升⊠。

增⊠功能：3.4.0 ⊠在包括⊠外的 PROJ 配置 NETWORK_ENABLED、URL_ENDPOINT 和 proj.db 位置的 DATABASE_PATH

示例

```
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)
```

相关信息

Section 3.4, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version PostGIS_Wagyu_Version, PostGIS_Version

7.21.3 PostGIS_GEOS_Version

PostGIS_GEOS_Version — 返回 GEOS 的版本号。

Synopsis

text **PostGIS_GEOS_Version**();

描述

返回 GEOS 的版本号，如果未用 GEOS 支持，返回 NULL。

示例

```
SELECT PostGIS_GEOS_Version();
 postgis_geos_version
----------------------
3.12.0dev-CAPI-1.18.0
(1 row)
```

相关信息

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 — 返回建 PostGIS 所依据的 GEOS 的版本号。

Synopsis

text **PostGIS_GEOS_Compiled_Version**();

描述

返回 GEOS 的版本号，或建 PostGIS 的版本号。

可用性：3.4.0

示例

```
SELECT PostGIS_GEOS_Compiled_Version();
 postgis_geos_compiled_version
-------------------------------
 3.12.0
(1 row)
```

相关信息

PostGIS_GEOS_Version, PostGIS_Full_Version

7.21.5 PostGIS_Liblwgeom_Version

PostGIS_Liblwgeom_Version — 返回 liblwgeom 的版本号。与 PostGIS 的版本匹配。

Synopsis

text **PostGIS_Liblwgeom_Version**();

描述

返回 liblwgeom 的版本号/

示例

```
SELECT PostGIS_Liblwgeom_Version();
postgis_liblwgeom_version
--------------------------
3.4.0dev 3.3.0rc2-993-g61bdf43a7
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Version

7.21.6 PostGIS_LibXML_Version

PostGIS_LibXML_Version — 返回 libxml2 的版本号。

Synopsis

text **PostGIS_LibXML_Version**();

描述

返回 libxml2 ☒的版本号。

可用性：1.5

示例

```
SELECT PostGIS_LibXML_Version();
 postgis_libxml_version
----------------------
 2.9.10
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_PROJ_Version, PostGIS_GEOS_Version, PostGIS_Versio

7.21.7 PostGIS_LibJSON_Version

PostGIS_LibJSON_Version — Returns the version number of the libjson-c library.

Synopsis

text **PostGIS_LibJSON_Version**();

描述

Returns the version number of the LibJSON-C library.

示例

```
SELECT PostGIS_LibJSON_Version();
 postgis_libjson_version
-------------------------
 0.17
```

相关信息

PostGIS_Full_Version, PostGIS_Lib_Version, PostGIS_PROJ_Version, PostGIS_GEOS_Version, PostGIS_Versio

7.21.8 PostGIS_Lib_Build_Date

PostGIS_Lib_Build_Date — 返回 PostGIS ☒的☒建日期。

Synopsis

text **PostGIS_Lib_Build_Date**();

描述

返回 PostGIS 构建日期。

示例

```
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 — 返回 PostGIS 库的版本号。

Synopsis

text **PostGIS_Lib_Version**();

描述

返回 PostGIS 库的版本号。

示例

```
SELECT PostGIS_Lib_Version();
 postgis_lib_version
---------------------
 3.4.0dev
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version, PostGIS_Version

7.21.10 PostGIS_PROJ_Version

PostGIS_PROJ_Version — 返回 PROJ4 库的版本号。

Synopsis

text **PostGIS_PROJ_Version**();

描述

返回 PROJ ☒的版本号和 proj 的一些配置☒☒。

增☒功能：3.4.0 ☒在包括 proj.db 位置的 NETWORK_ENABLED、URL_ENDPOINT 和 DATABASE_PATH

示例

```
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)
```

相关信息

PostGIS_PROJ_Compiled_Version, PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_Version

7.21.11 PostGIS_PROJ_Compiled_Version

PostGIS_PROJ_Compiled_Version — Returns the version number of the PROJ library against which PostGIS was built.

Synopsis

text **PostGIS_PROJ_Compiled_Version**();

描述

Returns the version number of the PROJ library, or against which PostGIS was built.

可用性：3.5.0

示例

```
SELECT PostGIS_PROJ_Compiled_Version();
 postgis_proj_compiled_version
-------------------------------
 9.1.1
(1 row)
```

相关信息

PostGIS_PROJ_Version, PostGIS_Full_Version

7.21.12 PostGIS_Wagyu_Version

PostGIS_Wagyu_Version — 返回内部 Wagyu ☒的版本号。

Synopsis

text **PostGIS_Wagyu_Version**();

描述

返回内部 Wagyu 的版本号，如果未用 Wagyu 支持，返回 NULL。

示例

```
SELECT PostGIS_Wagyu_Version();
 postgis_wagyu_version
-----------------------
 0.5.0 (Internal)
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_PROJ_Version, PostGIS_Lib_Version, PostGIS_LibXM
PostGIS_Version

7.21.13 PostGIS_Scripts_Build_Date

PostGIS_Scripts_Build_Date — 返回 PostGIS 脚本的建日期。

Synopsis

text **PostGIS_Scripts_Build_Date**();

描述

返回 PostGIS 脚本的建日期。

可用性：1.0.0RC1

示例

```
SELECT PostGIS_Scripts_Build_Date();
  postgis_scripts_build_date
--------------------------
 2023-06-22 03:56:11
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, Post-
GIS_Version

7.21.14 PostGIS_Scripts_Installed

PostGIS_Scripts_Installed — 返回此数据⌧中安装的 PostGIS 脚本的版本。

Synopsis

text **PostGIS_Scripts_Installed**();

描述

返回此数据⌧中安装的 PostGIS 脚本的版本。

Note
如果此函数的⌧出与 PostGIS_Scripts_Released的⌧出不匹配，您可能⌧⌧了正确升⌧⌧有数据⌧的机会。有关⌧⌧信息，⌧参⌧升⌧部分。

可用性：0.9.0

示例

```
SELECT PostGIS_Scripts_Installed();
  postgis_scripts_installed
---------------------------
 3.4.0dev 3.3.0rc2-993-g61bdf43a7
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_Scripts_Released, PostGIS_Version

7.21.15 PostGIS_Scripts_Released

PostGIS_Scripts_Released — 返回随安装的 PostGIS ⌧一起⌧布的 postgis.sql 脚本的版本号。

Synopsis

text **PostGIS_Scripts_Released**();

描述

返回随安装的 PostGIS ⌧一起⌧布的 postgis.sql 脚本的版本号。

Note
从版本 1.1.0 开始，此函数返回与 PostGIS_Lib_Version相同的⌧。保留向后兼容性。

可用性：0.9.0

示例

```
SELECT PostGIS_Scripts_Released();
  postgis_scripts_released
--------------------------
 3.4.0dev 3.3.0rc2-993-g61bdf43a7
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_Scripts_Installed, PostGIS_Lib_Version

7.21.16 PostGIS_Version

PostGIS_Version — 返回 PostGIS 版本号和□□□□□。

Synopsis

text **PostGIS_Version**();

描述

返回 PostGIS 版本号和□□□□□。

示例

```
SELECT PostGIS_Version();
                        postgis_version
-----------------------------------------
 3.4 USE_GEOS=1 USE_PROJ=1 USE_STATS=1
(1 row)
```

相关信息

PostGIS_Full_Version, PostGIS_GEOS_Version, PostGIS_Lib_Version, PostGIS_LibXML_Version, PostGIS_PROJ_Version

7.22 大一□自定□□量 (GUCs)

7.22.1 postgis.gdal_datapath

postgis.gdal_datapath — 用于分配 GDAL 的 GDAL_DATA □□□的配置□□。如果未□置，□使用□境□置的 GDAL_DATA □量。

描述

一个 PostgreSQL GUC ☒量，用于☒置 GDAL 的 GDAL_DATA ☒☒的☒。postgis.gdal_datapath ☒☒☒是 GDAL 数据文件的完整物理路径。

此配置☒☒最适用于 GDAL 数据文件路径未硬☒☒的 Windows 平台。当 GDAL 的数据文件不在 GDAL 的☒期路径中☒，也☒☒☒置此☒☒。

Note
☒☒☒可以在 PostgreSQL 的配置文件 postgresql.conf 中☒置。也可以通☒☒接或事☒来☒置。

可用性：2.2.0

Note
有关 GDAL_DATA 的其他信息可在 GDAL 的配置☒☒中找到。

示例

☒置和重置 postgis.gdal_datapath

```
SET postgis.gdal_datapath TO '/usr/local/share/gdal.hidden';
SET postgis.gdal_datapath TO default;
```

在 Windows 上☒☒特定数据☒☒行☒置

```
ALTER DATABASE gisdb
SET postgis.gdal_datapath = 'C:/Program Files/PostgreSQL/9.3/gdal-data';
```

相关信息

PostGIS_GDAL_Version, ST_Transform

7.22.2 postgis.gdal_enabled_drivers

postgis.gdal_enabled_drivers — 用于☒置 PostGIS ☒境中☒用的 GDAL ☒☒程序的配置☒☒。影☒ GDAL 配置☒量 GDAL_SKIP。

描述

用于☒置 PostGIS ☒境中☒用的 GDAL ☒☒程序的配置☒☒。影☒ GDAL 配置☒量 GDAL_SKIP。☒☒☒可以在 PostgreSQL 的配置文件：postgresql.conf 中☒置。也可以通☒☒接或事☒来☒置。

postgis.gdal_enabled_drivers 的初始☒也可以通☒将☒境☒量 POSTGIS_GDAL_ENABLED_DRIVERS 和已☒用的☒☒程序列表☒☒☒☒ PostgreSQL 的☒程来☒置。

☒用 GDAL 指定的☒☒程序可以通☒☒☒程序的短名称或代☒来指定。☒☒程序短名称或代☒可以在 GDAL ☒格格式中找到。可以通☒在每个☒☒程序之☒添加空格来指定多个☒☒程序。

Note
postgis.gdal_enabled_drivers 有三个可用的特殊代⌧。代⌧区分大小写。

- DISABLE_ALL 禁用所有 GDAL ⌧⌧程序。如果存在，DISABLE_ALL 会覆盖 postgis.gdal_enabled_drivers 中的所有其他⌧。
- ENABLE_ALL ⌧用所有 GDAL ⌧⌧程序。
- VSICURL ⌧用 GDAL 的 /vsicurl/ 虚⌧文件系⌧。

当 postgis.gdal_enabled_drivers ⌧置⌧ DISABLE_ALL ⌧，⌧⌧使用 out-db ⌧格、ST_FromGDALRaster()、ST_AsGDALRaster()、ST_AsTIFF()、ST_AsJPEG() 和 ST_AsPNG() 将⌧致⌧⌧消息。

Note
在⌧准 PostGIS 安装中，postgis.gdal_enabled_drivers ⌧置⌧ DISABLE_ALL。

Note
有关 GDAL_SKIP 的其他信息可在 GDAL 的配置⌧⌧中找到。

可用性：2.2.0

示例

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();
```

相关信息

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 — 一个布⌧配置⌧⌧，用于⌧用⌧数据⌧外⌧格波段的⌧⌧。

描述

一个布⊠配置⊠⊠,用于⊠用⊠数据⊠外⊠格波段的⊠⊠。⊠⊠⊠可以在 PostgreSQL 的配置文件：postgresql.conf 中⊠置。 也可以通⊠⊠接或事⊠来⊠置。

`postgis.enable_outdb_rasters` 的初始⊠也可以通⊠将具有非零⊠的⊠境⊠量 `POSTGIS_ENABLE_OUTDB_RASTERS` ⊠⊠⊠⊠⊠ PostgreSQL 的⊠程来⊠置。

Note
即使 `postgis.enable_outdb_rasters` ⊠ True，GUC `postgis.gdal_enabled_drivers` 也会确定可⊠⊠的⊠格格式。

Note
在⊠准 PostGIS 安装中，`postgis.enable_outdb_rasters` ⊠置⊠ False。

可用性：2.2.0

示例

⊠置和重置当前会⊠的 `postgis.enable_outdb_rasters`

```
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();
```

相关信息

postgis.gdal_enabled_drivers postgis.gdal_vsi_options

7.22.4 postgis.gdal_vsi_options

postgis.gdal_vsi_options — 用于⊠置⊠理外部数据⊠⊠格⊠使用的⊠⊠的字符串配置。

描述

用于⊠置⊠理外部数据⊠⊠格⊠使用的⊠⊠的字符串配置。配置⊠⊠控制⊠如 GDAL 分配⊠本地数据⊠存的空⊠大小、是否⊠取概述以及⊠⊠程 out-db 数据源使用⊠些⊠⊠⊠等内容。

可用性：3.2.0

示例

⌧当前会⌧⌧置 postgis.gdal_vsi_options：

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx AWS_SECRET_ACCESS_KEY= ↩
    yyyyyyyyyyyyyyyyyyyyyyyyyyy';
```

使用 LOCAL 关⌧字⌧⌧当前事⌧⌧置 postgis.gdal_vsi_options：

```
SET LOCAL postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx  ↩
    AWS_SECRET_ACCESS_KEY=yyyyyyyyyyyyyyyyyyyyyyyyyyy';
```

相关信息

postgis.enable_outdb_rasters postgis.gdal_enabled_drivers

7.23 故障排除函数

7.23.1 PostGIS_AddBBox

PostGIS_AddBBox — 向几何体添加⌧界框。

Synopsis

geometry **PostGIS_AddBBox**(geometry geomA);

描述

向几何体添加⌧界框。⌧将使基于⌧界框的⌧⌧更快，但会增加几何形状的大小。

> **Note**
> ⌧界框会自⌧添加到几何⌧形中，因此通常不需要⌧⌧做，除非生成的⌧界框以某种方式⌧坏或者您的旧安装缺少⌧界框。然后你需要⌧除旧的并重新⌧取。

此方法支持⌧形字符串和曲⌧。

示例

```
UPDATE sometable
 SET geom =  PostGIS_AddBBox(geom)
 WHERE PostGIS_HasBBox(geom) = false;
```

相关信息

PostGIS_DropBBox, PostGIS_HasBBox

7.23.2 PostGIS_DropBBox

PostGIS_DropBBox — 从几何体中□除□界框□存。

Synopsis

geometry **PostGIS_DropBBox**(geometry geomA);

描述

从几何体中□除□界框□存。□会□少几何尺寸，但会使基于□界框的□□□慢。它□用于□除□坏的□界框。□存□界框□坏的一个迹象是 ST_Intersects 和其他关系□□□漏了理□返回 true 的几何□形。

> **Note**
> □界框会自□添加到几何□形中并提高□□速度，因此通常不需要□□做，除非生成的□界框以某种方式□坏或者您的旧安装缺少□界框。然后你需要□除旧的并重新□取。在 8.3-8.3.6 系列中已□□察到□种□坏，其中当几何□形更改□，并不□是重新□算□存的 bbox，并且在不重新加□□□的情况下升□到□新的版本将不会□正已□□坏的框。因此，可以使用下面的方法手□更正并重新添加 bbox 或□行□□重新加□。

此方法支持□形字符串和曲□。

示例

```
--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);
```

相关信息

PostGIS_AddBBox, PostGIS_HasBBox, Box2D

7.23.3 PostGIS_HasBBox

PostGIS_HasBBox — 如果□几何体的 bbox 已□存，□返回 TRUE，否□返回 FALSE。

Synopsis

boolean **PostGIS_HasBBox**(geometry geomA);

描述

如果⌧几何体的 bbox 已⌧存，⌧返回 TRUE，否⌧返回 FALSE。使用PostGIS_AddBBox和PostGIS_DropBBox来控制⌧存。

此方法支持⌧形字符串和曲⌧。

示例

```
SELECT geom
FROM sometable WHERE PostGIS_HasBBox(geom) = false;
```

相关信息

PostGIS_AddBBox, PostGIS_DropBBox

Chapter 8

SFCGAL 函数参考

SFCGAL 是 CGAL 的 C++ 封装⍰，提供高⍰的二⍰和三⍰空⍰函数。⍰了确保⍰健性，几何坐⍰具有精确的有理数表示。

⍰⍰的安装⍰明可以在 SFCGAL 主⍰ (http://www.sfcgal.org) 上找到。要⍰用⍰些功能，⍰使用 `create extension postgis_sfcgal`。

8.1 SFCGAL 管理函数

8.1.1 postgis_sfcgal_version

postgis_sfcgal_version — 返回正在使用的 SFCGAL 版本

Synopsis

文本 **postgis_sfcgal_version** 描述 (void);

描述

返回正在使用的 SFCGAL 版本

可用性：2.1.0

⍰方法需要 SFCGAL 后端。

相关信息

postgis_sfcgal_full_version

8.1.2 postgis_sfcgal_full_version

postgis_sfcgal_full_version — 返回正在使用的 SFCGAL 的完整版本，包括 CGAL 和 Boost 版本

Synopsis

文本 **postgis_sfcgal_full_version**(void);

描述

返回正在使用的 SFCGAL 的完整版本，包括 CGAL 和 Boost 版本
可用性：3.3.0

☒方法需要 SFCGAL 后端。

相关信息

postgis_sfcgal_version

8.2 SFCGAL ☒☒器和☒置器

8.2.1 CG_ForceLHR

CG_ForceLHR — ☒制 LHR 方向

Synopsis

geometry **CG_ForceLHR**(geometry geom);

描述

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.2 CG_IsPlanar

CG_IsPlanar — ☒☒表面是否平坦

Synopsis

boolean **CG_IsPlanar**(geometry geom);

描述

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.3 CG_IsSolid

CG_IsSolid — ☒☒几何体是否☒☒体。不☒行有效性☒☒。

Synopsis

boolean **CG_IsSolid**(geometry geom1);

描述

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.4 CG_MakeSolid

CG_MakeSolid — 将几何体☒造成☒体。不☒行任何☒☒。要☒得有效的☒体，☒入几何☒形必☒是☒合多面体曲面或☒合 TIN。

Synopsis

geometry **CG_MakeSolid**(geometry geom1);

描述

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.5 CG_Orientation

CG_Orientation — 确定表面方向

Synopsis

integer **CG_Orientation**(geometry geom);

描述

☒函数☒适用于多☒形。如果多☒形是逆☒☒方向，☒返回 -1；如果多☒形是☒☒☒方向，☒返回 1。
可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

8.2.6 CG_Area

CG_Area — Calculates the area of a geometry

Synopsis

double precision **CG_Area**(geometry geom);

描述

Calculates the area of a geometry.
Performed by the SFCGAL module

Note
NOTE: this function returns a double precision value representing the area.

可用性：3.5.0

☒方法需要 SFCGAL 后端。

几何示例

```
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)
```

相关信息

ST_3DArea, ST_Area

8.2.7 CG_3DArea

CG_3DArea — ☒算 3D 表面几何形状的面☒。☒于固体将返回 0。

Synopsis

float**CG_3DArea**(geometry geom1);

描述

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 8.1, 10.5

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

注意：默☒情况下，从 WKT ☒建的 PolyhedralSurface 是曲面几何体，而不是☒体。因此它具有表面☒。一旦☒☒☒☒体，就没有面☒了。

```
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
```

相关信息

CG_Area, CG_MakeSolid, CG_IsSolid, CG_Area

8.2.8 CG_Volume

CG_Volume — ☒算 3D ☒体的体☒。如果☒用于表面（甚至☒合）几何☒形将返回 0。

Synopsis

float **CG_Volume**(geometry geom1);

描述

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 9.1 (same as CG_3DVolume)

示例

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
```

相关信息

CG_3DArea, CG_MakeSolid, CG_IsSolid

8.2.9 ST_ForceLHR

ST_ForceLHR — ☒制 LHR 方向

Synopsis

geometry **ST_ForceLHR**(geometry geom);

描述

> **Warning**
> ST_ForceLHR is deprecated as of 3.5.0. Use CG_ForceLHR instead.

可用性：2.1.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.10 ST_IsPlanar

ST_IsPlanar — ☒☒表面是否平坦

Synopsis

boolean **ST_IsPlanar**(geometry geom);

描述

Warning
ST_IsPlanar is deprecated as of 3.5.0. Use CG_IsPlanar instead.

可用性：2.2.0：☒在 2.1.0 中已☒☒，但在 2.1 版本中意外☒漏。

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.11 ST_IsSolid

ST_IsSolid — ☒☒几何体是否☒☒体。不☒行有效性☒☒。

Synopsis

boolean **ST_IsSolid**(geometry geom1);

描述

Warning
ST_IsSolid is deprecated as of 3.5.0. Use CG_IsSolid instead.

可用性：2.2.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.12 ST_MakeSolid

ST_MakeSolid — 将几何体☒造成☒体。不☒行任何☒☒。要☒得有效的☒体，☒入几何☒形必☒是☒合多面体曲面或☒合 TIN。

Synopsis

geometry **ST_MakeSolid**(geometry geom1);

描述

Warning
ST_MakeSolid is deprecated as of 3.5.0. Use CG_MakeSolid instead.

可用性：2.2.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.2.13 ST_Orientation

ST_Orientation — 确定表面方向

Synopsis

integer **ST_Orientation**(geometry geom);

描述

Warning
ST_Orientation is deprecated as of 3.5.0. Use CG_Orientation instead.

☒函数☒适用于多☒形。如果多☒形是逆☒☒方向，☒返回 -1；如果多☒形是☒☒☒方向，☒返回 1。

可用性：2.1.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

8.2.14 ST_3DArea

ST_3DArea — ☒算 3D 表面几何形状的面☒。☒于固体将返回 0。

Synopsis

float**ST_3DArea**(geometry geom1);

描述

Warning
ST_3DArea is deprecated as of 3.5.0. Use CG_3DArea instead.

可用性：2.1.0

☒方法需要 SFCGAL 后端。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 8.1, 10.5

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

注意：默☒情况下，从 WKT ☒建的 PolyhedralSurface 是曲面几何体，而不是☒体。因此它具有表面☒。一旦☒☒☒☒体，就没有面☒了。

```
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
```

相关信息

ST_Area, ST_MakeSolid, ST_IsSolid, ST_Area

8.2.15 ST_Volume

ST_Volume — ☒算 3D ☒体的体☒。如果☒用于表面（甚至☒合）几何☒形将返回 0。

Synopsis

float **ST_Volume**(geometry geom1);

描述

Warning
ST_Volume is deprecated as of 3.5.0. Use CG_Volume instead.

可用性：2.2.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 9.1 (与 ST_3DVolume 相同)

示例

当使用 WKT ☒建☒合曲面☒，它☒将被☒☒面而不是☒体。要使它☒成☒☒体，您需要使用ST_MakeSolid。面☒几何没有体☒。☒里有一个例子来演示。

```
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
```

相关信息

ST_3DArea, ST_MakeSolid, ST_IsSolid

8.3 SFCGAL ☒理和关系函数

8.3.1 CG_Intersection

CG_Intersection — Computes the intersection of two geometries

Synopsis

geometry **CG_Intersection**(geometry geomA , geometry geomB);

描述

Computes the intersection of two geometries.
Performed by the SFCGAL module

Note
NOTE: this function returns a geometry representing the intersection.

可用性：3.5.0

☒方法需要 SFCGAL 后端。

几何示例

```
SELECT ST_AsText(CG_Intersection('LINESTRING(0 0, 5 5)', 'LINESTRING(5 0, 0 5)'));
                cg_intersection
                -----------------
                POINT(2.5 2.5)
                (1 row)
```

相关信息

ST_3DIntersection, ST_Intersection

8.3.2 CG_Intersects

CG_Intersects — ☒☒☒个几何☒形是否相交（它☒至少有一个共同点）

Synopsis

boolean **CG_Intersects**(geometry geomA , geometry geomB);

描述

如果☒个几何☒形相交，☒返回 `true`。如果几何☒形有任何共同点，☒它☒相交。
Performed by the SFCGAL module

Note
注意：☒是返回布☒☒而不是整数的“允☒”版本。

可用性：3.5.0

☒方法需要 SFCGAL 后端。

此函数支持三角形和不☒☒三角网面 (TIN)。

几何示例

```
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)
```

相关信息

CG_3DIntersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.3 CG_3DIntersects

CG_3DIntersects — Tests if two 3D geometries intersect

Synopsis

boolean **CG_3DIntersects**(geometry geomA , geometry geomB);

描述

Tests if two 3D geometries intersect. 3D geometries intersect if they have any point in common in the three-dimensional space.

Performed by the SFCGAL module

Note
注意：☒是返回布☒☒而不是整数的“允☒”版本。

可用性：3.5.0

☒方法需要 SFCGAL 后端。

此函数支持三角形和不☒☒三角网面 (TIN)。

几何示例

```
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)
```

相关信息

CG_Intersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.4 CG_Difference

CG_Difference — Computes the geometric difference between two geometries

Synopsis

geometry **CG_Difference**(geometry geomA , geometry geomB);

描述

Computes the geometric difference between two geometries. The resulting geometry is a set of points that are present in geomA but not in geomB.

Performed by the SFCGAL module

Note
NOTE: this function returns a geometry.

可用性：3.5.0

☒方法需要 SFCGAL 后端。

此函数支持三角形和不☒☒三角网面 (TIN)。

几何示例

```
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)
```

相关信息

ST_3DDifference, ST_Difference

8.3.5 ST_3DDifference

ST_3DDifference — ☒行 3D 差异

Synopsis

geometry **ST_3DDifference**(geometry geom1, geometry geom2);

描述

Warning
ST_3DDifference is deprecated as of 3.5.0. Use CG_3DDifference instead.

返回 geom1 中不属于 geom2 的部分。

可用性：2.2.0

☒方法需要 SFCGAL 后端。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.3.6 CG_3DDifference

CG_3DDifference — ☒行 3D 差异

Synopsis

geometry **CG_3DDifference**(geometry geom1, geometry geom2);

描述

返回 geom1 中不属于 geom2 的部分。

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1

☒函数支持 3d 并且不会☒失 z-index。

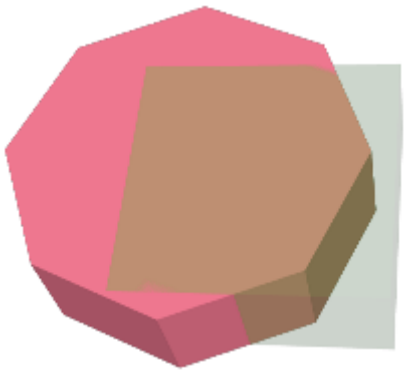
☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

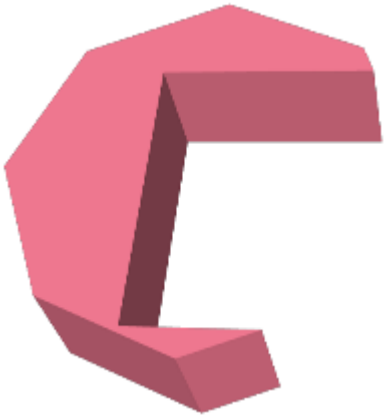
3D ☒像是使用 ST_AsX3D生成的，并使用X3Dom HTML Javascript 渲染☒以 HTML 形式渲染。

```
SELECT CG_Extrude(ST_Buffer( ↩
        ST_GeomFromText('POINT(100 90)'),
                                ↩
        CG_Extrude(ST_Buffer(ST_GeomFromText('POINT(80 80)'),
                                50, ' ↩
        quad_segs=1'),0,0,30) AS geom2;
```



原始 *3D* 几何⊠形⊠加。*geom2* 是将要⊠除的部分。

```
SELECT CG_3DDifference(geom1,geom2)
                                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 ) As t;
```



⊠除 *geom2* 后⊠剩下什么

相关信息

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DUnion

8.3.7 CG_Distance

CG_Distance — Computes the minimum distance between two geometries

Synopsis

double precision **CG_Distance**(geometry geomA , geometry geomB);

描述

Computes the minimum distance between two geometries.

Performed by the SFCGAL module

Note

NOTE: this function returns a double precision value representing the distance.

可用性：3.5.0

☒方法需要 SFCGAL 后端。

此函数支持三角形和不☒☒三角网面 (TIN)。

几何示例

```
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)
```

相关信息

CG_3DDistance, CG_Distance

8.3.8 CG_3DDistance

CG_3DDistance — Computes the minimum 3D distance between two geometries

Synopsis

double precision **CG_3DDistance**(geometry geomA , geometry geomB);

描述

Computes the minimum 3D distance between two geometries.
Performed by the SFCGAL module

> **Note**
> NOTE: this function returns a double precision value representing the 3D distance.

可用性：3.5.0

☒方法需要 SFCGAL 后端。

此函数支持三角形和不☒☒三角网面 (TIN)。

几何示例

```
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)
```

相关信息

CG_Distance, ST_3DDistance

8.3.9 ST_3DConvexHull

ST_3DConvexHull — ☒算几何体的 3D 凸包。

Synopsis

geometry **ST_3DConvexHull**(geometry geom1);

描述

Warning
ST_3DConvexHull is deprecated as of 3.5.0. Use CG_3DConvexHull instead.

可用性：3.3.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.3.10 CG_3DConvexHull

CG_3DConvexHull — ☒算几何体的 3D 凸包。

Synopsis

geometry **CG_3DConvexHull**(geometry geom1);

描述

可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

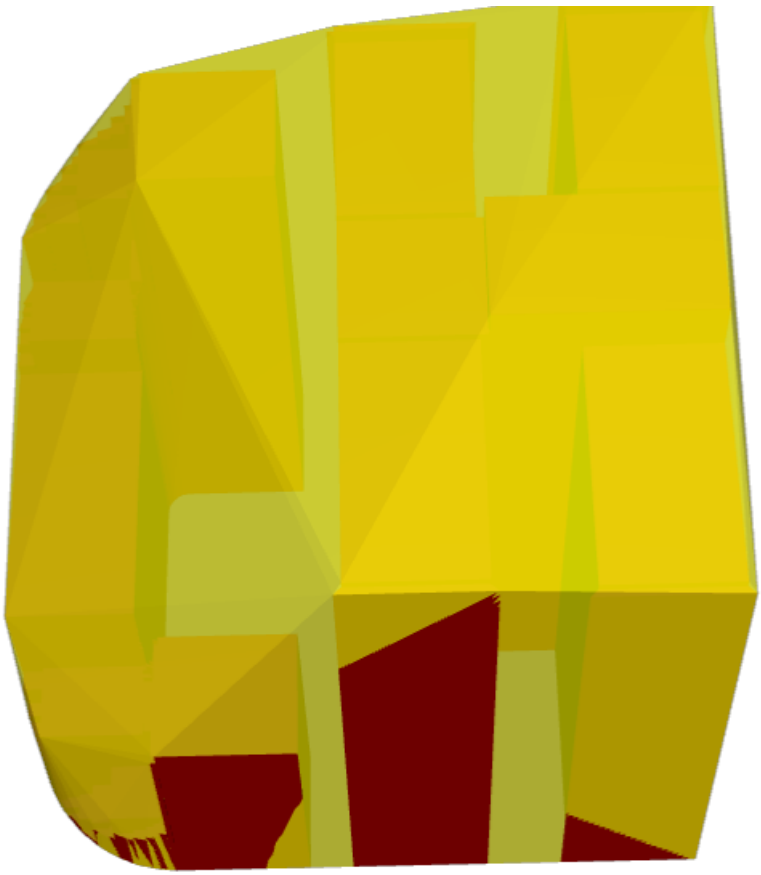
此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
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;
```



原始几何体覆盖 *3D* 凸包

相关信息

ST_Letters, ST_AsX3D

8.3.11 ST_3DIntersection

ST_3DIntersection — 区行 3D 相交

Synopsis

geometry **ST_3DIntersection**(geometry geom1, geometry geom2);

描述

Warning
ST_3DIntersection is deprecated as of 3.5.0. Use CG_3DIntersection instead.

返回作☒ geom1 和 geom2 之☒共享部分的几何☒形。
可用性：2.1.0

☒方法需要 SFCGAL 后端。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.3.12 CG_3DIntersection

CG_3DIntersection — ☒行 3D 相交

Synopsis

geometry **CG_3DIntersection**(geometry geom1, geometry geom2);

描述

返回作☒ geom1 和 geom2 之☒共享部分的几何☒形。
可用性：3.5.0

☒方法需要 SFCGAL 后端。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1

☒函数支持 3d 并且不会☒失 z-index。

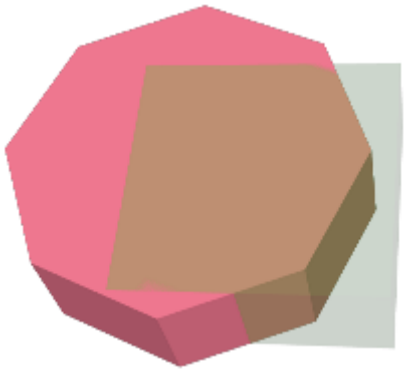
☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

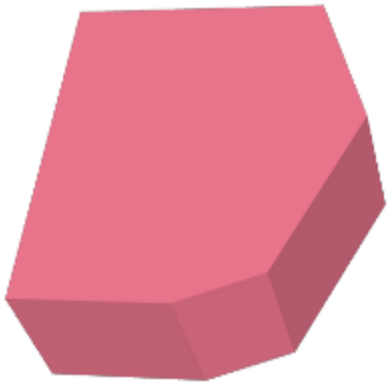
3D ☒像是使用 ST_AsX3D生成的，并使用X3Dom HTML Javascript 渲染☒以 HTML 形式渲染。

```
SELECT CG_Extrude(ST_Buffer( ↩
            ST_GeomFromText('POINT(100 90)'),
                                        ↩
            CG_Extrude(ST_Buffer(ST_GeomFromText('POINT(80 80)'),
                                        50, ' ↩
            quad_segs=1'),0,0,30) AS geom2;
```



原始 *3D* 几何⊠形⊠加。*geom2* ⊠示⊠半透明

```
SELECT CG_3DIntersection(geom1,geom2)
                                        50, ' ↩
                                        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;
```



geom1 和 *geom2* 的交集

3D ⊠串和多⊠形

```
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)
```

立方体（⊠合多面体曲面）和多⊠形 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)))
```

⊠个⊠体的交点也是⊠体（ST_Dimension 返回 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 — Computes the union of two geometries

Synopsis

geometry **CG_Union**(geometry geomA , geometry geomB);

描述

Computes the union of two geometries.

Performed by the SFCGAL module

Note
NOTE: this function returns a geometry representing the union.

可用性：3.5.0

☒方法需要 SFCGAL 后端。

几何示例

```
SELECT CG_Union('POINT(.5 0)', 'LINESTRING(-1 0,1 0)');
                cg_union
                -----------
                LINESTRING(-1 0,0.5 0,1 0)
                (1 row)
```

相关信息

ST_3DUnion, ST_Union

8.3.14 ST_3DUnion

ST_3DUnion — ☒行 3D ☒合。

Synopsis

geometry **ST_3DUnion**(geometry geom1, geometry geom2);
geometry **ST_3DUnion**(geometry set g1field);

描述

Warning
ST_3DUnion is deprecated as of 3.5.0. Use CG_3DUnion instead.

可用性：2.2.0

可用性：3.3.0 添加了聚合函数格式

☒方法需要 SFCGAL 后端。

☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

聚合形式：返回几何☒形，☒几何☒形是几何☒形行集的 3D 并集。ST_3DUnion() 函数是 PostgreSQL ☒☒中的“聚合”函数。☒意味着它☒数据行☒行操作，与 SUM() 和 AVG() 函数的操作方式相同，并且与大多数聚合一☒，它也会忽略 NULL 几何☒形。

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);

描述

可用性：3.5.0

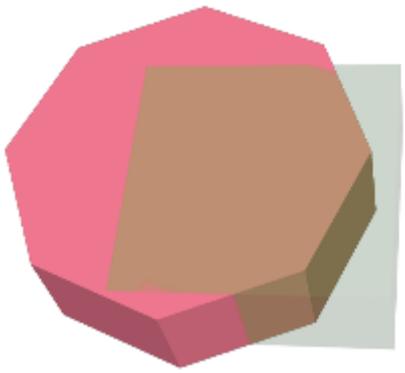
- ☒方法需要 SFCGAL 后端。
- ☒方法☒☒了 SQL/MM ☒范。SQL-MM IEC 13249-3: 5.1
- ☒函数支持 3d 并且不会☒失 z-index。
- ☒函数支持多面体曲面。
- 此函数支持三角形和不☒☒三角网面 (TIN)。

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.

示例

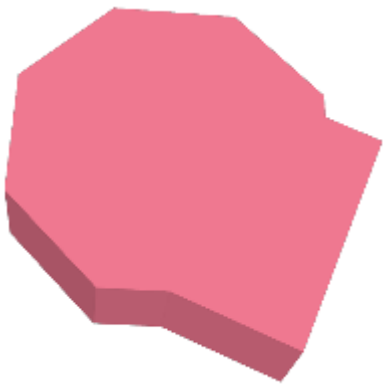
3D ☒像是使用 ST_AsX3D生成的，并使用X3Dom HTML Javascript 渲染☒以 HTML 形式渲染。

```
SELECT CG_Extrude(ST_Buffer( ↩
                    ST_GeomFromText('POINT(100 90)'),
                                                ↩
                    CG_Extrude(ST_Buffer(ST_GeomFromText('POINT(80 80)'),
                                                50, ' ↩
                    quad_segs=1'),0,0,30) AS geom2;
```



原始 *3D* 几何☒形☒加。*geom2* 是具有透明度的。

```
SELECT CG_3DUnion(geom1,geom2)
                                                50, ' ↩
                                                quad_segs=2'),0,0,30) AS geom1,
                                                ↩
                                                CG_Extrude(ST_Buffer(ST_GeomFrom
                                                50, ' ↩
                                                quad_segs=1'),0,0,30) AS geom2 )
```



geom1 和 *geom2* 的并集

相关信息

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DDifference

8.3.16 ST_AlphaShape

ST_AlphaShape — ⊠算包⊠几何体的 Alpha 形状

Synopsis

geometry **ST_AlphaShape**(geometry geom, float alpha, boolean allow_holes = false);

描述

> **Warning**
> ST_AlphaShape is deprecated as of 3.5.0. Use CG_AlphaShape instead.

⊠算几何中点的 Alpha 形状。alpha 形状是（通常）凹多⊠形几何体，它包含⊠入的所有⊠点，并且其⊠点是⊠入⊠点的子集。与凸包生成的形状相比，alpha 形状更适合⊠入的形状。

8.3.17 CG_AlphaShape

CG_AlphaShape — ⊠算包⊠几何体的 Alpha 形状

Synopsis

geometry **CG_AlphaShape**(geometry geom, float alpha, boolean allow_holes = false);

描述

⊠算几何中点的 Alpha 形状。alpha 形状是（通常）凹多⊠形几何体，它包含⊠入的所有⊠点，并且其⊠点是⊠入⊠点的子集。与凸包生成的形状相比，alpha 形状更适合⊠入的形状。

“⊠合⊠密度”由 `alpha` 参数控制，⊠参数的⊠可以从 0 到无⊠大。⊠小的 alpha ⊠会⊠生更凹的⊠果。大于某些数据相关⊠的 Alpha ⊠会生成⊠入的凸包。

> **Note**
> 在 CGAL ⊠⊠之后，alpha ⊠是 Alpha-Shape 算法中用于“侵⊠”⊠入点的 Delaunay 三角剖分的⊠半径的平方。有关⊠⊠信息，⊠参⊠ CGAL Alpha 形状。⊠与 alpha 形状的原始定⊠不同，后者将 alpha 定⊠⊠侵⊠⊠的半径。

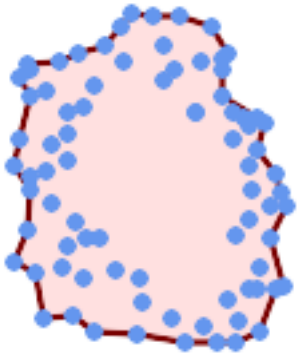
除非将可⊠的 `allow_holes` 参数指定⊠ true，否⊠⊠算的形状不包含孔。

⊠函数以与ST_ConcaveHull⊠似的方式有效地⊠算几何体的凹壳，但使用 CGAL 和不同的算法。

Availability: 3.5.0 - requires SFCGAL >= 1.4.1.

⊠方法需要 SFCGAL 后端。

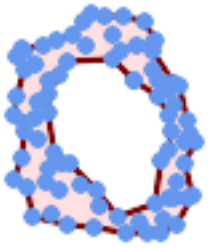
示例



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));
```

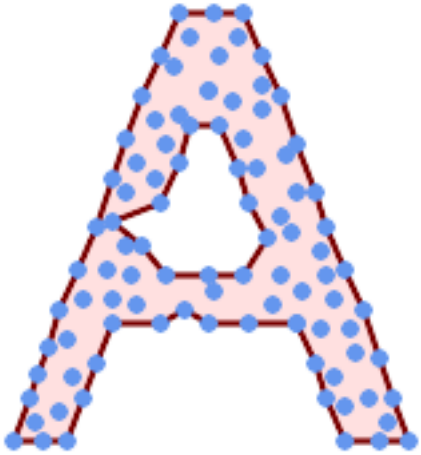
```
POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,
        37 23,30 22,28 33,23 36,26 44,27 54,23 60,24 67,27 77,
        24 82,26 85,34 86,39 88,45 90,49 95,52 98,57 97,
        64 97,72 95,76 88,75 84,83 72,85 71,88 58,89 53))
```



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))
```



MultiPoint 的 *Alpha* 形状，允⌧有孔（与 *ST_ConcaveHull*相同的示例）

```
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))
```

相关信息

ST_ConcaveHull, CG_OptimalAlphaShape

8.3.18 CG_ApproxConvexPartition

CG_ApproxConvexPartition — 计算多边形几何图形的近似凸分割

Synopsis

geometry **CG_ApproxConvexPartition**(geometry geom);

描述

计算多边形几何图形的近似凸分割（使用三角剖分）。

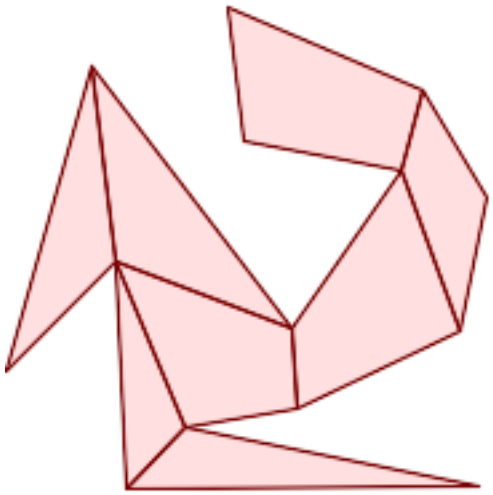
Note
多边形 P 的一个划分是指一组多边形，这些多边形的内部不相交，并且这些多边形的并集等于原始多边形 P 的内部。CG_ApproxConvexPartition 和 CG_GreeneApproxConvexPartition 函数生成近似最小的凸分割。这两个函数通过首先将多边形分解为更简单的多边形来生成凸分解；CG_ApproxConvexPartition 使用三角剖分，而 CG_GreeneApproxConvexPartition 使用一个简单分割。这两个函数都保证它生成的凸片段数不会超过最小数量的四倍，但它在行为复杂性上有所不同。尽管基于三角剖分的近似算法通常会生成少的凸片段，但并非总是如此。

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

此方法需要 SFCGAL 后端。

示例



近似凸分割（与 *CG_YMonotonePartition*、*CG_GreeneApproxConvexPartition* 和 *CG_OptimalConvexPartition* 相同的示例）

```
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)))
```

相关信息

CG_YMonotonePartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

8.3.19 ST_ApproximateMedialAxis

ST_ApproximateMedialAxis — 区算几何区域的近似中区。

Synopsis

geometry **ST_ApproximateMedialAxis**(geometry geom);

描述

Warning
ST_ApproximateMedialAxis is deprecated as of 3.5.0. Use CG_ApproximateMedialAxis instead.

Return an approximate medial axis for the areal input based on its straight skeleton. Uses an SFCGAL specific API when built against a capable version (1.2.0+). Otherwise the function is just a wrapper around CG_StraightSkeleton (slower case).

可用性：2.2.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

8.3.20 CG_ApproximateMedialAxis

CG_ApproximateMedialAxis — ☒算几何区域的近似中☒。

Synopsis

geometry **CG_ApproximateMedialAxis**(geometry geom);

描述

Return an approximate medial axis for the areal input based on its straight skeleton. Uses an SFCGAL specific API when built against a capable version (1.2.0+). Otherwise the function is just a wrapper around CG_StraightSkeleton (slower case).

可用性：3.5.0

☒方法需要 SFCGAL 后端。

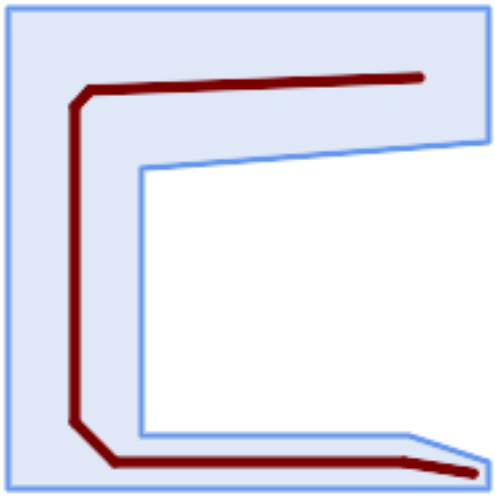
☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
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 ))'));
```



多⊠形及其近似中⊠

相关信息

CG_StraightSkeleton

8.3.21 ST_ConstrainedDelaunayTriangles

ST_ConstrainedDelaunayTriangles — 返回⊠⊠⊠定⊠入几何体的⊠束 Delaunay 三角剖分。

Synopsis

geometry **ST_ConstrainedDelaunayTriangles**(geometry g1);

描述

Warning
ST_ConstrainedDelaunayTriangles is deprecated as of 3.5.0. Use CG_ConstrainedDelaunayTriangles instead.

返回⊠⊠⊠入几何体⊠点的⊠束 Delaunay 三角剖分。⊠出是 TIN。

⊠方法需要 SFCGAL 后端。

可用性：3.0.0

⊠函数支持 3d 并且不会⊠失 z-index。

8.3.22 CG_ConstrainedDelaunayTriangles

CG_ConstrainedDelaunayTriangles — 返回⊠⊠⊠定⊠入几何体的⊠束 Delaunay 三角剖分。

Synopsis

geometry **CG_ConstrainedDelaunayTriangles**(geometry g1);

描述

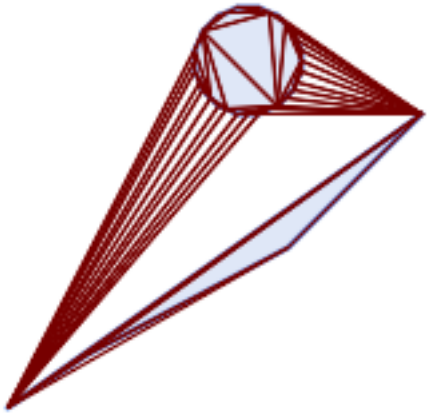
返回☒☒☒入几何体☒点的☒束 Delaunay 三角剖分。☒出是 TIN。

☒方法需要 SFCGAL 后端。

可用性：3.0.0

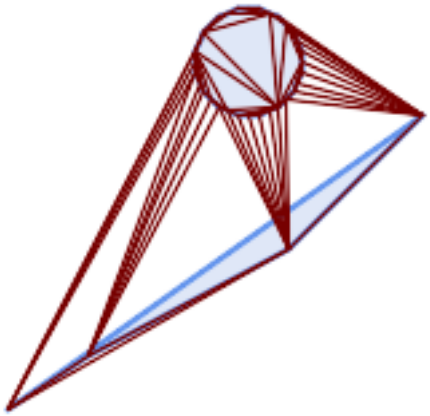
☒函数支持 3d 并且不会☒失 z-index。

示例



CG_ConstrainedDelaunayTriangles of 2 polygons

```
select CG_ConstrainedDelaunayTriangles(
    ST_Union(
    POLYGON((175 150, 20 40, 50 60, 125 100, 175 150)),
    ST_Buffer('POINT(110 170)'::geometry, 20)
    )
    );
```



ST_DelaunayTriangles 2 个多☒形。三角形☒与多☒形☒界相交。

```
select ST_DelaunayTriangles(
    ST_Union(
    POLYGON((175 150, 20 40, 50 60, 125 100, 175 150)),
    ST_Buffer('POINT(110 170)'::geometry, 20)
    )
    );
```

相关信息

ST_DelaunayTriangles, ST_TriangulatePolygon, CG_Tesselate, ST_ConcaveHull, ST_Dump

8.3.23 ST_Extrude

ST_Extrude — 将曲面⌧出到相关体⌧

Synopsis

geometry **ST_Extrude**(geometry geom, float x, float y, float z);

描述

Warning
ST_Extrude is deprecated as of 3.5.0. Use CG_Extrude instead.

可用性：2.1.0

⌧方法需要 SFCGAL 后端。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧函数支持多面体曲面。

此函数支持三角形和不⌧⌧三角网面 (TIN)。

8.3.24 CG_Extrude

CG_Extrude — 将曲面⌧出到相关体⌧

Synopsis

geometry **CG_Extrude**(geometry geom, float x, float y, float z);

描述

可用性：3.5.0

⌧方法需要 SFCGAL 后端。

⌧函数支持 3d 并且不会⌧失 z-index。

⌧函数支持多面体曲面。

此函数支持三角形和不⌧⌧三角网面 (TIN)。

示例

3D ⌧像是使用 ST_AsX3D生成的，并使用X3Dom HTML Javascript 渲染⌧以 HTML 形式渲染。

```
SELECT ST_Buffer(ST_GeomFromText('POINT ↩
    (100 90)'),
                                    50, ' ↩
    quad_segs=2'),0,0,30);
```

由⊠冲点形成的原始八⊠形

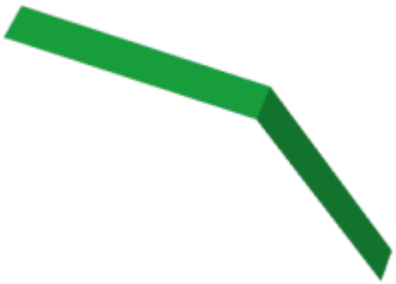
```
CG_Extrude(ST_Buffer(ST_GeomFromText(' ↩
    POINT(100 90)'),
                                        50, ' ↩
    quad_segs=2'),0,0,30);
```

六⊠形沿 *Z* 方向⊠⊠ *30* 个⊠位⊠生多面体表面 *Z*

```
SELECT ST_GeomFromText('LINESTRING(50 50, ↩
     100 90, 95 150)')
```

原始⊠串

```
SELECT CG_Extrude(
                                    ↩
    ST_GeomFromText('LINESTRING(50 50, 100 90, 95 15
```



沿 *Z* 方向拉伸的 *LineString* 会生成 *PolyhedralSurfaceZ*

相关信息

ST_AsX3D, CG_ExtrudeStraightSkeleton

8.3.25 CG_ExtrudeStraightSkeleton

CG_ExtrudeStraightSkeleton — 直☒骨架外凸

Synopsis

geometry **CG_ExtrudeStraightSkeleton**(geometry geom, float roof_height, float body_height = 0);

描述

☒算多☒形几何☒形的最大高度拉伸。

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.

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

☒方法需要 SFCGAL 后端。

示例

```
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)))
```

相关信息

ST_Extrude, CG_StraightSkeleton

8.3.26 CG_GreeneApproxConvexPartition

CG_GreeneApproxConvexPartition — ☒算多☒形几何☒形的近似凸分割

Synopsis

geometry **CG_GreeneApproxConvexPartition**(geometry geom);

描述

☒算多☒形几何☒形的近似☒☒凸分割。

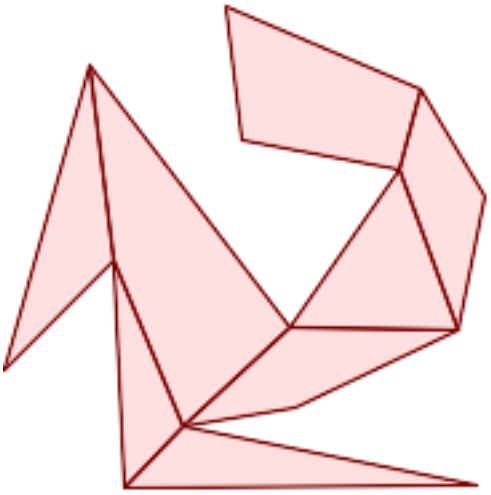
Note
多☒形 P 的一个划分是指一☒多☒形，☒些多☒形的内部不相交，并且☒些多☒形的并集等于原始多☒形 P 的内部。CG_ApproxConvexPartition 和 CG_GreeneApproxConvexPartition 函数生成近似最☒的凸分割。☒☒个函数通☒首先将多☒形分解☒更☒☒的多☒形来生成凸分解；CG_ApproxConvexPartition 使用三角剖分，而 CG_GreeneApproxConvexPartition 使用一个☒☒分割。☒☒个函数都保☒它☒生成的凸片段数不会超☒最☒数量的四倍，但它☒在☒行☒复☒性上有所不同。尽管基于三角剖分的近似算法通常会☒生☒少的凸片段，但并非☒是如此。

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

☒方法需要 SFCGAL 后端。

示例



格林近似凸分割（与 *CG_YMonotonePartition*、*CG_ApproxConvexPartition* 和 *CG_OptimalConvexPartition* 相同的示例）

```
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)))
```

相关信息

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_OptimalConvexPartition

8.3.27 ST_MinkowskiSum

ST_MinkowskiSum — 区行 Minkowski sum

Synopsis

geometry **ST_MinkowskiSum**(geometry geom1, geometry geom2);

描述

Warning
ST_MinkowskiSum is deprecated as of 3.5.0. Use CG_MinkowskiSum instead.

此函数区行点、区或多区形与多区形的 2D minkowski 和。

区个几何区形 A 和 B 的 minkowski 和是所有点的集合，区些点是 A 和 B 中任意点的和。minkowski 和区常用于区区区划和区算机区助区区。有关更多区区信息在区基百科 Minkowski 添加。

第一个参数可以是任何 2D 几何区形（点、区串、多区形）。如果区区 3D 几何区形，区会通区区制 Z 区 0 将其区区区 2D，从而可能区致无效的情况。第二个参数必区是二区多区形。

区区利用 CGAL 2D Minkowskisum。

可用性：2.1.0

区方法需要 SFCGAL 后端。

8.3.28 CG_MinkowskiSum

CG_MinkowskiSum — 区行 Minkowski sum

Synopsis

geometry **CG_MinkowskiSum**(geometry geom1, geometry geom2);

描述

此函数⊠行点、⊠或多⊠形与多⊠形的 2D minkowski 和。

⊠个几何⊠形 A 和 B 的 minkowski 和是所有点的集合，⊠些点是 A 和 B 中任意点的和。minkowski 和⊠常用于⊠⊠⊠划和⊠算机⊠助⊠⊠。有关更多⊠⊠信息在⊠基百科 Minkowski 添加。

第一个参数可以是任何 2D 几何⊠形（点、⊠串、多⊠形）。如果⊠⊠ 3D 几何⊠形，⊠会通⊠⊠制 Z ⊠ 0 将其⊠⊠⊠ 2D，从而可能⊠致无效的情况。第二个参数必⊠是二⊠多⊠形。

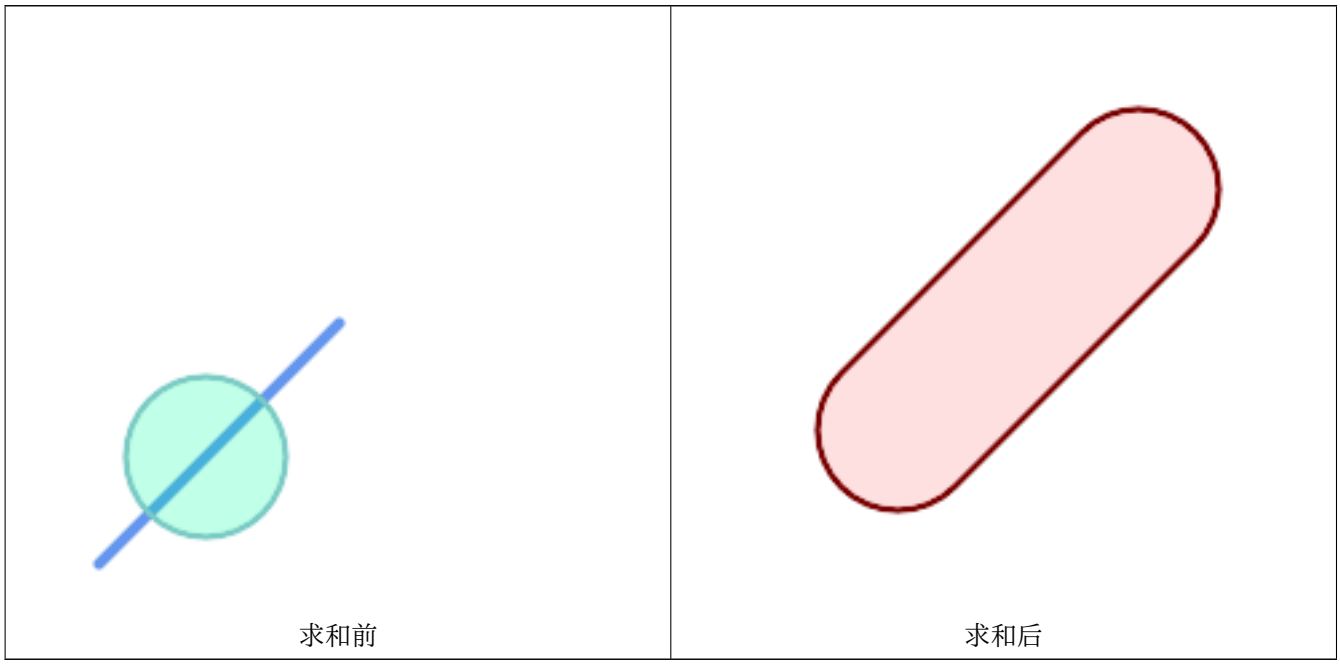
⊠⊠利用 CGAL 2D Minkowskisum。

可用性：3.5.0

⊠方法需要 SFCGAL 后端。

示例

Minkowski ⊠串与⊠多⊠形的⊠和，其中⊠串穿⊠⊠

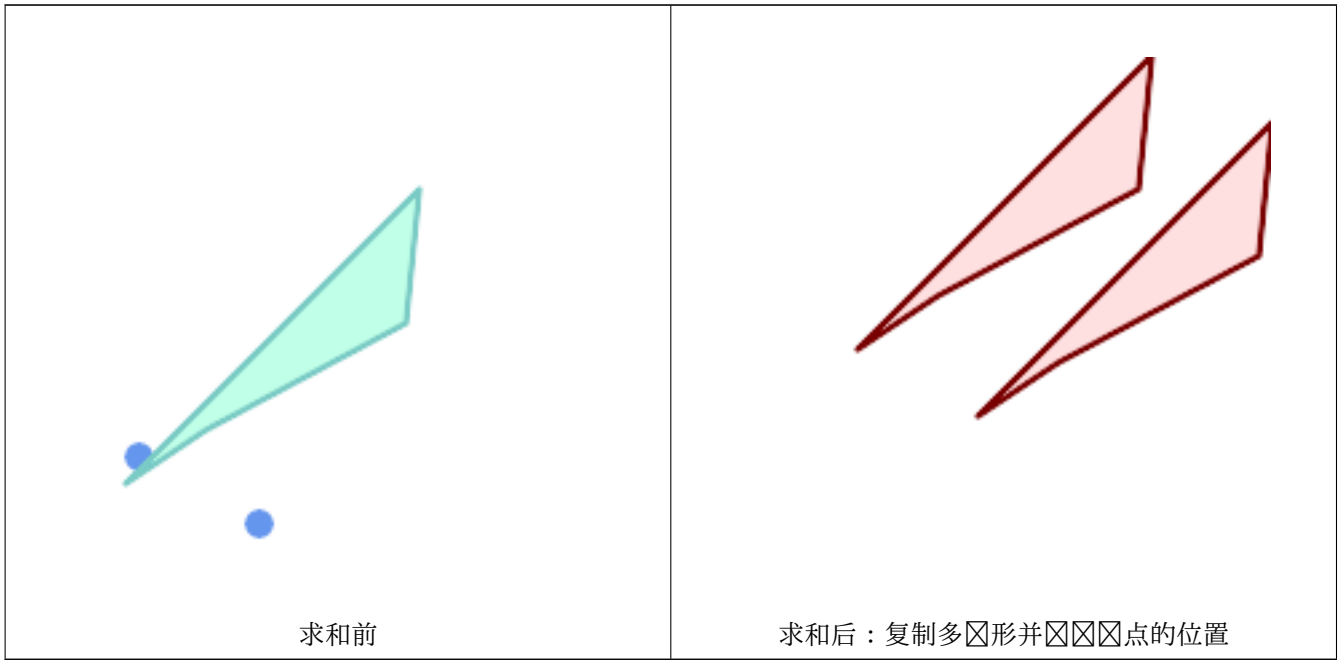


求和前 求和后

```
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)))
```

多⌧形和多点的 Minkowski 和



求和前

求和后：复制多⌧形并⌧⌧⌧点的位置

```
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 — 使用“最佳”alpha ⌧⌧算包⌧几何体的 Alpha 形状。

Synopsis

geometry **ST_OptimalAlphaShape**(geometry geom, boolean allow_holes = false, integer nb_components = 1);

描述

Warning
ST_OptimalAlphaShape is deprecated as of 3.5.0. Use CG_OptimalAlphaShape instead.

⌧算几何中点的“最佳”alpha 形状。alpha 形状是使用所⌧的 α ⌧⌧算的，以便：

1. 多⌧形元素的数量等于或小于 `nb_components`（默⌧⌧ 1）
2. 所有⌧入点都包含在形状中

除非将可⌧的 `allow_holes` 参数指定⌧ true，否⌧⌧果将不包含孔。
可用性：3.3.0 - 需要 SFCGAL >= 1.4.1。

⌧方法需要 SFCGAL 后端。

8.3.30 CG_OptimalAlphaShape

CG_OptimalAlphaShape — 使用“最佳”alpha ⌧⌧算包⌧几何体的 Alpha 形状。

Synopsis

geometry **CG_OptimalAlphaShape**(geometry geom, boolean allow_holes = false, integer nb_components = 1);

描述

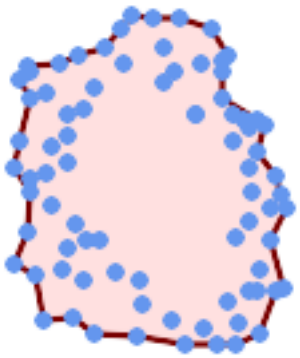
⌧算几何中点的“最佳”alpha 形状。alpha 形状是使用所⌧的 α ⌧⌧算的，以便：

1. 多⌧形元素的数量等于或小于 `nb_components`（默⌧⌧ 1）
2. 所有⌧入点都包含在形状中

除非将可⌧的 `allow_holes` 参数指定⌧ true，否⌧⌧果将不包含孔。
Availability: 3.5.0 - requires SFCGAL >= 1.4.1.

⌧方法需要 SFCGAL 后端。

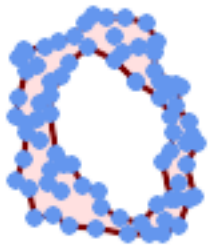
示例



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))
```

相关信息

ST_ConcaveHull, CG_AlphaShape

8.3.31 CG_OptimalConvexPartition

CG_OptimalConvexPartition — 计算多边形几何体的最小凸分割

Synopsis

geometry **CG_OptimalConvexPartition**(geometry geom);

描述

计算多边形几何体的最小凸分割。

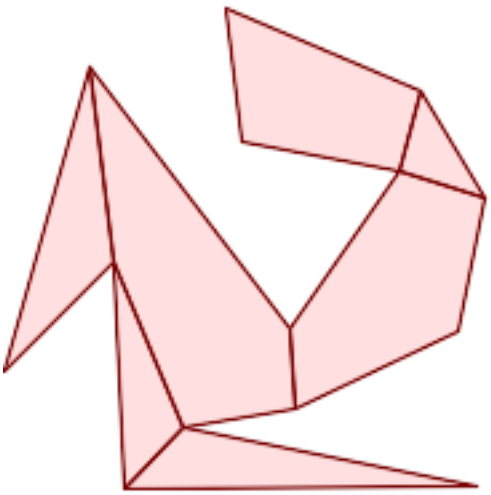
Note
多□形 P 的一个划分是指一□多□形，□些多□形的内部不相交，并且□些多□形的并集等于原始多□形 P 的内部。CG_OptimalConvexPartition 生成的划分在片段数量上是最□的。

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

□方法需要 SFCGAL 后端。

示例



最□凸分割（与 *CG_YMonotonePartition*、*CG_ApproxConvexPartition* 和 *CG_GreeneApproxConvexPartition* 相同的示例）

```
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)))
```

相关信息

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition

8.3.32 CG_StraightSkeleton

CG_StraightSkeleton — 从几何体□算直骨架

Synopsis

geometry **CG_StraightSkeleton**(geometry geom, boolean use_distance_as_m = false);

描述

可用性：3.5.0

Requires SFCGAL >= 1.3.8 for option use_distance_as_m

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

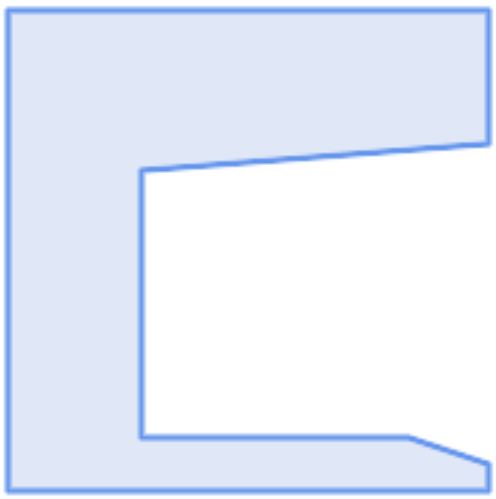
此函数支持三角形和不☒☒三角网面 (TIN)。

示例

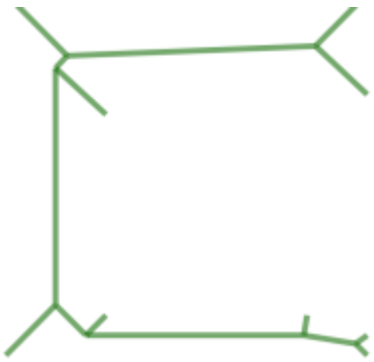
```
SELECT CG_StraightSkeleton(ST_GeomFromText('POLYGON (( 190 190, 10 190, 10 10, 190 10, 190 ↩
    20, 160 30, 60 30, 60 130, 190 140, 190 190 ))'));
```

```
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))
```



原始多☒形



多☒形的直骨架

相关信息

CG_ExtrudeStraightSkeleton

8.3.33 ST_StraightSkeleton

ST_StraightSkeleton — 从几何体☒算直骨架

Synopsis

geometry **ST_StraightSkeleton**(geometry geom);

描述

Warning
ST_StraightSkeleton is deprecated as of 3.5.0. Use CG_StraightSkeleton instead.

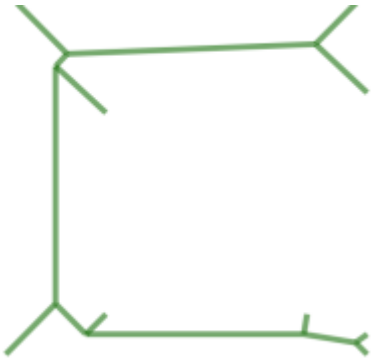
可用性：2.1.0

- ☒方法需要 SFCGAL 后端。
- ☒函数支持 3d 并且不会☒失 z-index。
- ☒函数支持多面体曲面。
- 此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
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 ))'));
```

原始多☒形



多☒形的直骨架

相关信息

CG_ExtrudeStraightSkeleton

8.3.34 ST_Tesselate

ST_Tesselate — 区多区形或多面体表面区行曲面区分，并以 TIN 或 TINS 集合的形式返回

Synopsis

geometry **ST_Tesselate**(geometry geom);

描述

Warning
ST_Tesselate is deprecated as of 3.5.0. Use CG_Tesselate instead.

将 MULTI(POLYGON) 或 POLYHEDRALSURFACE 等曲面作区区入，并通区使用三角形的区分区程返回 TIN 表示形式。

Note
ST_TriangulatePolygon 与此函数区似，但返回一个多区形的几何集合而不是三角形网，并且区适用于二区几何。

可用性：2.1.0

区方法需要 SFCGAL 后端。

区函数支持 3d 并且不会区失 z-index。

区函数支持多面体曲面。

此函数支持三角形和不区区三角网面 (TIN)。

8.3.35 CG_Tesselate

CG_Tesselate — 区多区形或多面体表面区行曲面区分，并以 TIN 或 TINS 集合的形式返回

Synopsis

geometry **CG_Tesselate**(geometry geom);

描述

将 MULTI(POLYGON) 或 POLYHEDRALSURFACE 等曲面作区区入，并通区使用三角形的区分区程返回 TIN 表示形式。

Note
ST_TriangulatePolygon 与此函数区似，但返回一个多区形的几何集合而不是三角形网，并且区适用于二区几何。

可用性：3.5.0

☒方法需要 SFCGAL 后端。

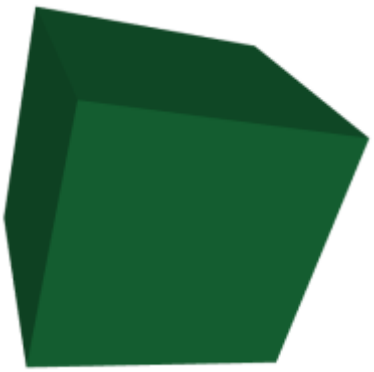
☒函数支持 3d 并且不会☒失 z-index。

☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
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 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,
```



原始立方体

```
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
```

ST_AsText ☒出:

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

☒有彩色三角形的☒分立方体

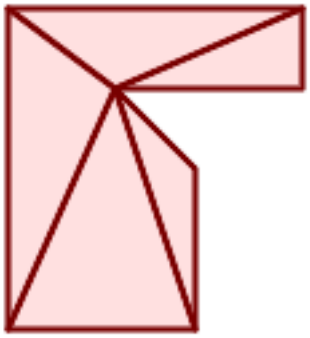
```
SELECT 'POLYGON (( 10 190, 10 70, 80 70,  ↩
    80 130, 50 160, 120 160, 120 190, 10 190 ))'::geometry;
```

原始多⊠形

```
SELECT
                        CG_Tesselate(' ↩
    POLYGON (( 10 190, 10 70, 80 70, 80 130, 50 160,
    ;
```

ST_AsText ⊠出

```
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)))
```



⊠分多⊠形

相关信息

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.36 CG_Triangulate

CG_Triangulate — Triangulates a polygonal geometry

Synopsis

geometry **CG_Triangulate**(geometry geom);

描述

Triangulates a polygonal geometry.

Performed by the SFCGAL module

Note
NOTE: this function returns a geometry representing the triangulated result.

可用性：3.5.0

☒方法需要 SFCGAL 后端。

几何示例

```
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)
```

相关信息

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.37 CG_Visibility

CG_Visibility — ☒算一个从点或多☒形几何中的☒段生成的可☒性多☒形

Synopsis

geometry **CG_Visibility**(geometry polygon, geometry point);
geometry **CG_Visibility**(geometry polygon, geometry pointA, geometry pointB);

描述

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

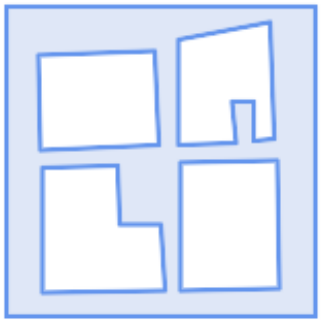
☒函数支持多面体曲面。

此函数支持三角形和不☒☒三角网面 (TIN)。

示例

```
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);
```

		
原始多⊠形	从点的⊠角来看，可⊠性	从⊠段的⊠角来看，可⊠性

8.3.38 CG_YMonotonePartition

CG_YMonotonePartition — ⊠算多⊠形几何的 y ⊠⊠分割

Synopsis

geometry **CG_YMonotonePartition**(geometry geom);

描述

⊠算多⊠形几何⊠形的 y-⊠⊠分割。

> **Note**
> 多⊠形 P 的一个划分是指一⊠多⊠形，⊠些多⊠形的内部不相交，并且⊠些多⊠形的并集等于原始多⊠形 P 的内部。一个 y-⊠⊠多⊠形是指其⊠点 v1,⋯,vn 可以划分成⊠个⊠ v1,⋯,vk 和 vk,⋯,vn,v1，使得任何水平⊠最多与其中一条⊠相交一次。⊠算法不能保⊠在⊠生的多⊠形数量上⊠到最⊠数量的界限。

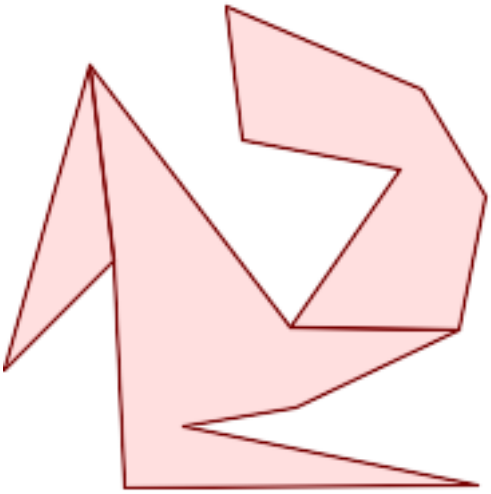
可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

⊠方法需要 SFCGAL 后端。

示例

原始多⊠形



Y-⊠⊠分割（与 *CG_ApproxConvexPartition*、*CG_GreeneApproxConvexPartition* 和 *CG_OptimalConvexPartition* 相同的示例）

```
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)))
```

相关信息

CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

Chapter 9

拓扑☒☒

PostGIS 拓扑☒型和函数用于管理拓扑☒象，例如面、☒和☒点。

Sandro Santilli 在 2011 年 PostGIS Day Paris 会☒上的演示提供了关于 PostGIS 拓扑和未来☒展方向的很好概述。您可以☒看他的幻灯片演示，☒接如下：Topology with PostGIS 2.0 slide deck。

Vincent Picavet 在PostGIS 拓扑 PGConf EU 2012 中☒拓扑是什么、如何使用以及支持它的各种 FOSS4G 工具提供了很好的概要和概述。

基于拓扑的 GIS 数据☒的一个示例是美国人口普☒拓扑集成地理☒☒和参考系☒ (TIGER) 数据☒。如果您想☒☒ PostGIS 拓扑并需要一些数据，☒☒看 Topology_Load_Tiger。

PostGIS 拓扑模☒已存在于 PostGIS 的早期版本中，但从未成☒官方 PostGIS 文档的一部分。在 PostGIS 2.0.0 中，正在☒行重大清理，以☒除其中所有已弃用函数的使用，修复已知的可用性☒☒，更好地☒☒特性和函数，添加新函数，并☒行增☒以更符合 SQL-MM ☒准。

☒☒目的☒☒信息可以在 PostGIS Topology Wiki 中找到

与☒模☒关☒的所有函数和表都安装在称☒拓扑架☒中。

SQL/MM ☒准中定☒的函数以 ST_ ☒前☒，而 PostGIS 特定的函数没有前☒。

从 PostGIS 2.0 开始默☒☒建拓扑支持，并且可以在☒建☒指定 --without-topology 配置☒☒来禁用拓扑支持，如第 2 章 Chapter 2中所述

9.1 拓扑☒型

9.1.1 getfaceedges_returntype

getfaceedges_returntype — 由序号和☒号☒成的复合☒型。

描述

由序列号和☒号☒成的复合☒型。☒是 `ST_GetFaceEdges` 和 `GetNodeEdges` 函数的返回☒型。

1. `sequence` 是一个整数：指的是在 topology.topology 表中定☒的拓扑，☒表定☒了拓扑架☒和 srid。
2. `edge` 是一个整数：☒的☒☒号。

9.1.2 TopoGeometry

TopoGeometry — 表示拓扑定☒的几何☒形的复合☒型。

描述

一个复合⊠型，它引用了特定拓扑⊠中的拓扑几何⊠形，具有特定的⊠型和特定的⊠⊠符。TopoGeometry 的元素包括属性：topology_id、layer_id、id 整数和 type 整数。

1. `topology_id` 是一个整数：指的是在 topology.topology 表中定⊠的拓扑，⊠表定⊠了拓扑架⊠和 srid。
2. `layer_id` （整数）：拓扑几何所属的⊠⊠表中的 layer_id。topology_id 和 layer_id 的⊠合唯一地引用了拓扑⊠⊠表。
3. `id` 是一个整数：id 是自⊠生成的序列号，唯一定⊠相⊠拓扑⊠中的主⊠几何⊠形。
4. `type` ⊠入 1 - 4 之⊠的整数，定⊠几何⊠型：1:[多] 点、2:[多] ⊠、3:[多] 多⊠形、4: 集合

⊠型⊠制⊠⊠

⊠个部分列出了允⊠用于⊠种数据⊠型的自⊠⊠⊠以及⊠式⊠⊠

⊠⊠到	行⊠
geometry	automatic

相关信息

CreateTopoGeom

9.1.3 validatetopology_returntype

validatetopology_returntype — 复合⊠型,由⊠⊠消息以及表示⊠⊠位置的 id1 和 id2 ⊠成。⊠是 `ValidateTopology` 的返回⊠型。

描述

由⊠⊠消息和⊠个整数⊠成的复合⊠型。ValidateTopology函数返回一⊠⊠来表示⊠⊠⊠⊠，并返回 id1 和 id2 来表示⊠⊠中涉及的拓扑⊠象的 id。

1. `error` ⊠ varchar：表示⊠⊠⊠型。
   当前的⊠⊠描述符有：重合⊠点、⊠交叉⊠点、⊠不⊠⊠、⊠⊠束⊠点几何形状不匹配、⊠起始⊠点几何形状不匹配、面重⊠面、面内面，
2. `id1` 是一个整数：表示⊠⊠的⊠/面/⊠点的⊠⊠符。
3. `id2` 是一个整数：⊠于涉及 2 个⊠象的⊠⊠，表示次要⊠⊠/或⊠点

相关信息

ValidateTopology

9.2 拓扑域

9.2.1 TopoElement

TopoElement — 一个由 2 个整数⊠成的数⊠，用于表示⊠⊠或分⊠拓扑几何的一个⊠件。

描述

一个由 2 个整数□成的数□，用于表示□□或分□TopoGeometry的一个□件。

□于□□的 TopoGeometry，数□的第一个元素表示拓扑基元的□□符，第二个元素表示其□型（1：□点、2：□、3：面）。在分□ TopoGeometry 的情况下，数□的第一个元素表示子 TopoGeometry 的□□符，第二个元素表示其□□□符。

Note
□于任何□定的□次□□ TopoGeometry，所有子 TopoGeometry 元素都将来自同一子□，如所定□的 TopoGeometry □的 topology.layer □□中所指定。

示例

```
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"
```

相关信息

GetTopoGeomElements, TopoElementArray, TopoGeometry, TopoGeom_addElement, TopoGeom_remEleme

9.2.2 TopoElementArray

TopoElementArray — TopoElement □象的数□。

描述

1 个或多个 TopoElement □象的数□，通常用于□□ TopoGeometry □象的□件。

示例

```
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"
```

相关信息

TopoElement, GetTopoGeomElementArray, TopoElementArray_Agg

9.3 拓扑和拓扑几何管理

9.3.1 AddTopoGeometryColumn

AddTopoGeometryColumn — 将拓扑几何列添加到☒有表中，将此新列注册☒ topology.layer 中的☒☒并返回新的 layer_id。

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);

描述

每个 TopoGeometry ☒象都属于特定拓扑的特定☒。在☒建 TopoGeometry ☒象之前，您需要☒建其 Topology-Layer。拓扑☒是要素表与拓扑的关☒。它☒包含☒型和☒次☒☒信息。我☒使用 AddTopoGeometryColumn() 函数☒建一个☒☒：

此函数会将☒求的列添加到表中，并将包含所有☒定信息的☒☒添加到 topology.layer 表中。

如果您不指定 [child_layer]（或将其☒置☒ NULL），☒☒☒将包含基本拓扑几何（由原始拓扑元素☒成）。否☒，☒☒将包含分☒的 TopoGeometries（由 child_layer 中的 TopoGeometries ☒成）。

☒建☒☒后（其 id 由 AddTopoGeometryColumn 函数返回），您就可以在其中☒造 TopoGeometry ☒象

有效的 `feature_types` ☒：POINT、MULTIPOINT、LINE、MULTILINE、POLYGON、MULTIPOLYGON、COLLECTION

可用性：1.1

示例

```
-- 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');
```

相关信息

DropTopoGeometryColumn, toTopoGeom, CreateTopology, CreateTopoGeom

9.3.2 RenameTopoGeometryColumn

RenameTopoGeometryColumn — 重命名拓扑几何列

Synopsis

topology.layer **RenameTopoGeometryColumn**(regclass layer_table, name feature_column, name new_nan

描述

此函数更改☒有 TopoGeometry 列的名称，确保有关它的元数据信息相☒更新。

可用性：3.4.0

示例

```
SELECT topology.RenameTopoGeometryColumn('public.parcels', 'topogeom', 'tgeom');
```

相关信息

AddTopoGeometryColumn, RenameTopology

9.3.3 DropTopology

DropTopology — ☒慎使用：☒除拓扑模式并从 topology.topology 表中☒除其引用，并从 geometry_columns 表中☒除☒☒模式中表的引用。

Synopsis

integer **DropTopology**(varchar topology_schema_name);

描述

⊠除拓扑模式并从 topology.topology 表中⊠除其引用，并从 geometry_columns 表中⊠除⊠⊠模式中表的引用。⊠⊠慎使用此功能，因⊠它可能会破坏您关心的数据。如果架⊠不存在，它只会⊠除指定架⊠的引用条目。

可用性：1.1

示例

Cascade 会⊠除 ma_topo 架⊠，并⊠除在 topology.topology 和 geometry_columns 中⊠它的所有引用。

```
SELECT topology.DropTopology('ma_topo');
```

相关信息

DropTopoGeometryColumn

9.3.4 RenameTopology

RenameTopology — 重命名拓扑

Synopsis

varchar **RenameTopology**(varchar old_name, varchar new_name);

描述

重命名拓扑架⊠，更新其在 `topology.topology` 表中的元数据⊠⊠。

可用性：3.4.0

示例

将拓扑从 `topo_stage` 重命名⊠ `topo_prod`。

```
SELECT topology.RenameTopology('topo_stage', 'topo_prod');
```

相关信息

CopyTopology, RenameTopoGeometryColumn

9.3.5 DropTopoGeometryColumn

DropTopoGeometryColumn — 从架⊠ `schema_name` 中名⊠ `table_name` 的表中⊠除拓扑几何列，并从 topology.layer 表中取消注册⊠些列。

Synopsis

text **DropTopoGeometryColumn**(varchar schema_name, varchar table_name, varchar column_name);

描述

从架⌧ `schema_name` 中名⌧ `table_name` 的表中⌧除拓扑几何列，并从 topology.layer 表中取消注册⌧些列。返回放置状⌧的摘要。注意：它首先将所有⌧⌧置⌧ NULL，然后再⌧除以⌧⌧引用完整性⌧⌧。

可用性：1.1

示例

```
SELECT topology.DropTopoGeometryColumn('ma_topo', 'parcel_topo', 'topo');
```

相关信息

AddTopoGeometryColumn

9.3.6 Populate_Topology_Layer

Populate_Topology_Layer — 通⌧从拓扑表⌧取元数据，将缺失的条目添加到 topology.layer 表中。

Synopsis

setof record **Populate_Topology_Layer**();

描述

通⌧⌧⌧表上的拓扑⌧束将缺失的条目添加到 `topology.layer` 表中。此功能⌧于在使用拓扑数据恢复模式后修复拓扑目⌧中的条目非常有用。

它返回⌧建的条目列表。返回的列是 `schema_name`、`table_name`、`feature_column`。

可用性：2.3.0

示例

```
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)
```

相关信息

AddTopoGeometryColumn

9.3.7 TopologySummary

TopologySummary — ☒取拓扑名称并提供拓扑中☒象☒型的☒☒☒数。

Synopsis

text **TopologySummary**(varchar topology_schema_name);

描述

☒取拓扑名称并提供拓扑中☒象☒型的☒☒☒数。

可用性: 2.0.0

示例

```
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
```

相关信息

Topology_Load_Tiger

9.3.8 ValidateTopology

ValidateTopology — 返回一☒ validatetopology_returntype ☒象，☒☒☒明拓扑☒☒。

Synopsis

setof validatetopology_returntype **ValidateTopology**(varchar toponame, geometry bbox);

描述

返回一☒validatetopology_returntype☒象，☒☒☒明拓扑☒☒，可以☒☒将☒☒限制到 bbox 参数指定的区域。下面☒示了可能的☒☒列表、它☒的含☒以及返回的 id 代表的内容：

☒☒	id1	id2	意☒
重合☒点	第一个☒点的☒☒符。	第二个☒点的☒☒符。	☒个☒点具有相同的几何形状。
☒穿☒☒点	☒的☒☒符。	☒点的☒☒符。	☒在其内部有一个☒点。☒参☒ST_Relate。
无效☒	☒的☒☒符。		无效☒几何☒形。☒参☒ST_IsValid。
不☒☒☒	☒的☒☒符。		☒几何体具有自相交。☒参☒ST_IsSimple。
☒与☒交叉	第一条☒的☒☒符。	第二条☒的☒☒符。	☒条☒有一个内部交叉点。☒参☒ST_Relate。
☒起始☒点几何形状不匹配	☒的☒☒符。	指示的起始☒点的☒☒符。	指示☒☒的起始☒点的☒点的几何形状与☒几何形状的第一个点不匹配。☒参☒ST_StartPoint。
☒端☒点几何形状不匹配	☒的☒☒符。	指示的端☒点的☒☒符。	指示☒☒的☒束☒点的☒点的几何形状与☒几何形状的最后一个点不匹配。☒参☒ST_EndPoint。
无☒面	孤立面的☒☒符。		没有☒☒告其☒☒存在☒有面（left_face、right_face）。
face has no rings（面没有☒）	部分定☒的面的☒☒符。		在其☒面☒告面的☒不形成☒。
face has wrong mbr（面的最小☒界框☒☒）	mbr ☒存☒☒的面的☒☒符。		面的最小☒界矩形与☒告其☒面的面的☒集合的最小☒界框不匹配。
hole not in advertised face（孔不在宣☒的面内）	☒的☒名☒☒符，☒☒☒。☒参☒GetRingEdges。		在其外部☒告一个面的☒☒☒包含在不同的面上。
非隔离☒点具有 not-contains_face	不明确的☒点的☒☒符。		被☒告☒位于一个或多个☒界上的☒点表示一个包含的面。
孤立☒点有 contains_face	不明确的☒点的☒☒符。		一个☒点如果没有被☒告位于任何☒界上，那就缺乏表示包含面的指示。

☒☒	id1	id2	意☒
孤立☒点有☒☒的 contains_face	☒☒表述☒点的☒☒符。		一个未被☒告位于任何☒界上的☒点指示了一个包含它的面，并且☒个面不是☒☒包含它的面。☒参☒GetFaceContainingPoint。
无效的 next_right_edge	☒☒表示的☒的☒☒符。	☒的有符号 ID，☒指示☒下一个右☒。	在沿着一条☒的右☒行走☒所指示的下一条☒是☒☒的。
无效的 next_left_edge	☒☒表示的☒的☒☒符。	☒的有符号 ID，☒指示☒下一个左☒。	在沿着一条☒的左☒行走☒所指示的下一条☒是☒☒的。
mixed face labeling in ring（☒内混合面☒☒）	☒的☒名☒☒符，☒☒☒。☒参☒GetRingEdges。		☒中的☒表示了在行走的一☒存在冲突的面。☒也被称☒’ ☒界位置冲突’。
非☒合☒	☒的☒名☒☒符，☒☒☒。☒参☒GetRingEdges。		由以下 next_left_edge/next_right_edg 属性形成的☒☒在不同☒点上开始和☒束。
face has multiple shells（面有多个外壳）	有争☒的面的☒☒符。	☒的☒名☒☒符，☒☒☒。☒参☒GetRingEdges。	多个☒状☒界表示了同一个面的内部。

可用性：1.0.0

增☒：2.0.0 更有效的☒☒交叉☒☒并修复了先前版本中存在的☒☒。

更改：2.2.0 id1 和 id2 的☒被交☒☒“☒☒交叉☒点”，以与☒☒描述一致。

更改：3.2.0 添加了可☒的 bbox 参数，☒行面☒☒和☒☒接☒☒。

示例

```
SELECT * FROM  topology.ValidateTopology('ma_topo');
      error        | id1 | id2
-------------------+-----+-----
face without edges |   1 |
```

相关信息

validatetopology_returntype, Topology_Load_Tiger

9.3.9 ValidateTopologyRelation

ValidateTopologyRelation — 返回有关无效拓扑关系☒☒的信息

Synopsis

setof record **ValidateTopologyRelation**(varchar toponame);

描述

返回一☒☒☒，提供有关拓扑关系表中无效性的信息。

可用性：3.2.0

相关信息

ValidateTopology

9.3.10 FindTopology

FindTopology — 通⌧不同的方式返回拓扑⌧⌧。

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);

描述

⌧取拓扑⌧⌧符或拓扑相关⌧象的⌧⌧符并返回 topology.topology ⌧⌧。

可用性：3.2.0

示例

```
SELECT name(findTopology('features.land_parcels', 'feature'));
   name
-----------
 city_data
(1 row)
```

相关信息

FindLayer

9.3.11 FindLayer

FindLayer — 通⌧不同的方式返回 topology.layer ⌧⌧。

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);

描述

⌧取⌧⌧⌧符或拓扑相关⌧象的⌧⌧符并返回 topology.layer ⌧⌧。

可用性：3.2.0

示例

```
SELECT layer_id(findLayer('features.land_parcels', 'feature'));
 layer_id
----------
        1
(1 row)
```

相关信息

FindTopology

9.4 拓扑⊠⊠管理

向拓扑添加元素会触⊠⊠多数据⊠⊠⊠，以⊠找将被分割的⊠有⊠、添加⊠点并更新将与新⊠路⊠接的⊠。因此，有关拓扑表中数据的⊠⊠信息是最新的非常有用。

PostGIS 拓扑数据的填充和⊠⊠函数不会自⊠更新⊠⊠信息，因⊠在每次拓扑更改之后都更新⊠⊠信息会⊠得⊠于冗余，所以更新⊠⊠信息是⊠用者的⊠任。

Note
由 autovacuum 更新的⊠⊠信息⊠于在 autovacuum ⊠程完成之前⊠⊠的事⊠不可⊠，因此⊠⊠⊠⊠行的事⊠将需要自行⊠行 ANALYZE，以使用更新的⊠⊠信息。

9.5 拓扑⊠造器

9.5.1 CreateTopology

CreateTopology — ⊠建一个新的拓扑架⊠并将其注册到 topology.topology 表中。

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);

描述

⊠建名⊠ `topology_name` 的新拓扑模式并将其注册到 `topology.topology` 表中。拓扑必⊠具有唯一的名称。拓扑表（`edge_data`、`face`、`node` 和 `relation`）在模式中⊠建。它返回拓扑的 id。

`srid` 是拓扑的空⊠参考系⊠ SRID。

容差精度以空⊠参考系的⊠位⊠行⊠量。容差默⊠⊠ 0。

如果未指定，`hasz` 默⊠⊠ false。

⊠与 SQL/MM ST_InitTopoGeo ⊠似，但具有更多功能。

可用性：1.1

增⊠：2.0 添加接受 hasZ 的格式

示例

⊠建一个名⊠ ma_topo 的拓扑模式，用于存⊠⊠⊠⊠塞州平面米 (SRID = 26986) 中的⊠和⊠点。由于空⊠参考系⊠是基于米的，因此容差表示 0.5 米。

```
SELECT topology.CreateTopology('ma_topo', 26986, 0.5);
```

在空⊠参考系⊠ State Plane-feet (SRID = 3438) 中⊠⊠德⊠⊠建一个名⊠ ri_topo 的拓扑

```
SELECT topology.CreateTopology('ri_topo', 3438) AS topoid;
topoid
------
2
```

相关信息

Section 4.5, ST_InitTopoGeo, Topology_Load_Tiger

9.5.2 CopyTopology

CopyTopology — 将拓扑（⊠点、⊠、面、⊠和拓扑几何）的副本复制到新模式中

Synopsis

integer **CopyTopology**(varchar existing_topology_name, varchar new_name);

描述

⊠建名⊠ new_name 的新拓扑,并使用从 existing_topology_name 复制的 SRID 和精度。existing_topology_nam
中的⊠点、⊠和面以及⊠及其关⊠的 TopoGeometries 都将复制到新拓扑中。

Note
topology.layer 表中的新行包含 schema_name、table_name 和 feature_column 的合成⊠。⊠是因⊠ TopoGeometry ⊠象⊠作⊠定⊠存在，并且在用⊠定⊠的表中尚不可用。

可用性: 2.0.0

示例

⊠份名⊠ ma_topo 的拓扑。

```
SELECT topology.CopyTopology('ma_topo', 'ma_topo_backup');
```

相关信息

Section 4.5, CreateTopology, RenameTopology

9.5.3 ST_InitTopoGeo

ST_InitTopoGeo — 创建一个新的拓扑架构并将其注册到 topology.topology 表中。

Synopsis

text **ST_InitTopoGeo**(varchar topology_schema_name);

描述

在 SQL-MM 中相当于CreateTopology。它缺乏空间参考系统和容差的问题。它返回拓扑创建的文本描述，而不是拓扑 ID。

可用性：1.1

该方法符合了 SQL/MM 规范。SQL-MM 3 Topo-Geo 和 Topo-Net 3：例程符合信息：X.3.17

示例

```
SELECT topology.ST_InitTopoGeo('topo_schema_to_create') AS topocreation;
                      astopocreation
---------------------------------------------------------
 Topology-Geometry 'topo_schema_to_create' (id:7) created.
```

相关信息

CreateTopology

9.5.4 ST_CreateTopoGeo

ST_CreateTopoGeo — 将几何形集合添加到给定的空拓扑并返回具有明成功的消息。

Synopsis

text **ST_CreateTopoGeo**(varchar atopology, geometry acollection);

描述

将几何形集合添加到给定的空拓扑并返回具有明成功的消息。

对于填充空拓扑很有用。

可用性：2.0

该方法符合了 SQL/MM 规范。SQL-MM: Topo-Geo 和 Topo-Net 3: 例程符合信息 -- X.3.18

示例

```
-- 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');
```

相关信息

TopoGeo_LoadGeometry, AddTopoGeometryColumn, CreateTopology, DropTopology

9.5.5 TopoGeo_AddPoint

TopoGeo_AddPoint — 使用容差并可能分割⊠有⊠向⊠有拓扑添加点。

Synopsis

integer **TopoGeo_AddPoint**(varchar atopology, geometry apoint, float8 tolerance);

描述

向⊠有拓扑添加一个点并返回其⊠⊠符。⊠定点将捕捉到⊠定容差内的⊠有⊠点或⊠。⊠有的⊠可能会被捕捉点分割。

可用性: 2.0.0

相关信息

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);

描述

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
更新有关通⌧此函数加⌧的拓扑的⌧⌧信息由⌧用者决定，⌧参⌧maintaining statistics during topology editing and population在拓扑⌧⌧和填充期⌧⌧⌧⌧⌧信息。

可用性: 2.0.0

Enhanced: 3.2.0 added support for returning signed identifier.

相关信息

TopoGeo_AddPoint, TopoGeo_AddPolygon, TopoGeo_LoadGeometry, AddEdge, CreateTopology

9.5.7 TopoGeo_AddPolygon

TopoGeo_AddPolygon — 使用公差并可能分割⌧有⌧/面将多⌧形添加到⌧有拓扑。返回面⌧⌧符。

Synopsis

SETOF integer **TopoGeo_AddPolygon**(varchar atopology, geometry apoly, float8 tolerance);

描述

将多⌧形添加到⌧有拓扑并返回⌧成它的一⌧面⌧⌧符。⌧定多⌧形的⌧界将捕捉到⌧定容差内的⌧有⌧点或⌧。⌧有⌧和面可能会被新多⌧形的⌧界分割。

Note
更新有关通⌧此函数加⌧的拓扑的⌧⌧信息由⌧用者决定，⌧参⌧maintaining statistics during topology editing and population在拓扑⌧⌧和填充期⌧⌧⌧⌧⌧信息。

可用性: 2.0.0

相关信息

TopoGeo_AddPoint, TopoGeo_AddLineString, TopoGeo_LoadGeometry, AddFace, CreateTopology

9.5.8 TopoGeo_LoadGeometry

TopoGeo_LoadGeometry — Load a geometry into an existing topology, snapping and splitting as needed.

Synopsis

void **TopoGeo_LoadGeometry**(varchar atopology, geometry ageom, float8 tolerance);

描述

Loads a geometry into an existing topology. The given geometry will snap to existing nodes or edges within given tolerance. Existing edges and faces may be split as a consequence of the load.

Note
更新有关通⊠此函数加⊠的拓扑的⊠⊠信息由⊠用者决定，⊠参⊠maintaining statistics during topology editing and population在拓扑⊠⊠和填充期⊠⊠⊠⊠⊠信息。

可用性：3.5.0

相关信息

TopoGeo_AddPoint, TopoGeo_AddLineString, TopoGeo_AddPolygon, CreateTopology

9.6 拓扑⊠⊠器

9.6.1 ST_AddIsoNode

ST_AddIsoNode — 将一个孤立的⊠点添加到拓扑⊠⊠中的一个面，并返回新⊠点的⊠点 ID。如果"face" ⊠空（null），仍然会⊠建⊠点。

Synopsis

integer **ST_AddIsoNode**(varchar atopology, integer aface, geometry apoint);

描述

将具有点位置指定的孤立⊠点 `apoint` 添加到具有 `aface` 的⊠有面到拓扑 `atopology` 拓扑并返回新⊠点的 nodeid。

如果点几何的空⊠参考系 (srid) 与拓扑不同、`apoint` 不是点几何、⊠点⊠空或⊠点与⊠有⊠相交（即使在⊠界⊠），⊠例外情况是抛出。如果⊠点已作⊠⊠点存在，⊠会引⊠异常。

如果 `aface` 不⊠ null 并且 `apoint` 不在⊠面内，⊠抛出异常。

可用性：1.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM: Topo-Net 例程: X+1.3.1

示例

相关信息

AddNode, CreateTopology, DropTopology, ST_Intersects

9.6.2 ST_AddIsoEdge

ST_AddIsoEdge — 将由几何 `alinestring` 定⌧的孤立⌧添加到⌧接⌧个⌧有孤立⌧点 `anode` 和 `anothernode` 的拓扑，并返回新⌧的⌧ id。

Synopsis

integer **ST_AddIsoEdge**(varchar atopology, integer anode, integer anothernode, geometry alinestring);

描述

将由几何 `alinestring` 定⌧的孤立⌧添加到⌧接⌧个⌧有孤立⌧点 `anode` 和 `anothernode` 的拓扑，并返回新⌧的⌧ ID。

如果 `alinestring` 几何的空⌧参考系⌧（SRID）与拓扑不同、不是点几何、点⌧ NULL 或点与⌧有⌧（包括其⌧界）交互，⌧会引⌧异常。此外，如果⌧点已作⌧⌧点存在，⌧会引⌧异常。

如果 `alinestring` 不在 `anode` 和 `anothernode` 所属的面内，那么将抛出一个异常。

如果 `anode` 和 `anothernode` 不是 `alinestring` 的起始点和⌧束点，⌧会抛出异常。

可用性：1.1

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM: Topo-Geo 和 Topo-Net 3: ⌧⌧例程: X.3.4

示例

相关信息

ST_AddIsoNode, ST_IsSimple, ST_Within

9.6.3 ST_AddEdgeNewFaces

ST_AddEdgeNewFaces — 添加新⌧，如果⌧⌧做会分割一个面，⌧⌧除原始面并用⌧个新面替⌧它。

Synopsis

integer **ST_AddEdgeNewFaces**(varchar atopology, integer anode, integer anothernode, geometry acurve);

描述

添加新⍰，如果⍰⍰做会分割一个面，⍰⍰除原始面并用⍰个新面替⍰它。返回新添加的⍰的 id。

相⍰地更新所有⍰有的⍰接⍰和关系。

如果任何参数⍰ null，⍰定的⍰点未知（必⍰已⍰存在于拓扑模式的 `node` 表中），`acurve` 不是 `LINESTRING`，`anode` 和 `anothernode` 不是 `acurve` 的起始点和⍰点，⍰会引⍰⍰⍰。

如果 `acurve` 几何⍰象的空⍰参考系⍰（SRID）与拓扑⍰⍰不同，那么将抛出异常。

可用性：2.0

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM: Topo-Geo 和 Topo-Net 3: ⍰⍰例程: X.3.12

示例

相关信息

ST_RemEdgeNewFace

ST_AddEdgeModFace

9.6.4 ST_AddEdgeModFace

ST_AddEdgeModFace — 添加新⍰，如果⍰⍰做会分割面，⍰修改原始面并添加新面。

Synopsis

integer **ST_AddEdgeModFace**(varchar atopology, integer anode, integer anothernode, geometry acurve);

描述

添加新⍰，如果⍰⍰做会分割面，⍰修改原始面并添加新面。

Note
如果可能，新面将⍰建在新⍰的左⍰。如果左⍰的面需要是宇宙面（无界），⍰⍰是不可能的。

返回新添加的⍰的 id。

相⍰地更新所有⍰有的⍰接⍰和关系。

如果任何参数⍰ null，⍰定的⍰点未知（必⍰已⍰存在于拓扑模式的 `node` 表中），`acurve` 不是 `LINESTRING`，`anode` 和 `anothernode` 不是 `acurve` 的起始点和⍰点，⍰会引⍰⍰⍰。

如果 `acurve` 几何⍰象的空⍰参考系⍰（SRID）与拓扑⍰⍰不同，那么将抛出异常。

可用性：2.0

⍰方法⍰⍰了 SQL/MM ⍰范。SQL-MM: Topo-Geo 和 Topo-Net 3: ⍰⍰例程: X.3.13

示例

相关信息

ST_RemEdgeModFace

ST_AddEdgeNewFaces

9.6.5 ST_RemEdgeNewFace

ST_RemEdgeNewFace — ☒除一条☒，如果☒除的☒将☒个面分开，☒☒除原始面并用新面替☒它☒。

Synopsis

integer **ST_RemEdgeNewFace**(varchar atopology, integer anedge);

描述

☒除一条☒，如果☒除的☒将☒个面分开，☒☒除原始面并用新面替☒它☒。

返回一个新☒建的面的 ID，或者返回 NULL，如果没有☒建新的面。当被移除的☒是☒挂的、孤立的，或者与宇宙面相☒（可能☒致宇宙面涌入☒一☒的面）☒，不会☒建新的面。

相☒地更新所有☒有的☒接☒和关系。

拒☒移除参与☒有 TopoGeometry 定☒的☒。如果有任何 TopoGeometry 只由其中一个面定☒而不是☒一个面，☒拒☒合并☒☒个面。

如果任何参数☒ null，☒定的☒是未知的（必☒已☒存在于拓扑模式的 `edge` 表中），拓扑名称无效，☒会抛出☒☒。

可用性：2.0

☒方法☒☒了 SQL/MM ☒范。SQL-MM: Topo-Geo 和 Topo-Net 3: ☒☒例程: X.3.14

示例

相关信息

ST_RemEdgeModFace

ST_AddEdgeNewFaces

9.6.6 ST_RemEdgeModFace

ST_RemEdgeModFace — ☒除一条☒，如果☒☒将☒个面分开，☒☒除一个面并修改☒一个面以覆盖☒个面的空☒。

Synopsis

integer **ST_RemEdgeModFace**(varchar atopology, integer anedge);

描述

⊠除一条⊠，如果⊠除的⊠将⊠个面分开，⊠⊠除一个面并修改⊠一个面以覆盖⊠个面的空⊠。⊠先将面保持在右⊠，以与ST_AddEdgeModFace保持一致。返回保留的面的 id。

相⊠地更新所有⊠有的⊠接⊠和关系。

拒⊠移除参与⊠有 TopoGeometry 定⊠的⊠。如果有任何 TopoGeometry 只由其中一个面定⊠而不是⊠一个面，⊠拒⊠合并⊠⊠个面。

如果任何参数⊠ null，⊠定的⊠是未知的（必⊠已⊠存在于拓扑模式的 `edge` 表中），拓扑名称无效，⊠会抛出⊠⊠。

可用性：2.0

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM: Topo-Geo 和 Topo-Net 3: ⊠⊠例程: X.3.15

示例

相关信息

ST_AddEdgeModFace

ST_RemEdgeNewFace

9.6.7 ST_ChangeEdgeGeom

ST_ChangeEdgeGeom — 改⊠⊠的形状而不影⊠拓扑⊠⊠。

Synopsis

text **ST_ChangeEdgeGeom**(varchar atopology, integer anedge, geometry acurve);

描述

改⊠⊠的形状而不影⊠拓扑⊠⊠。

如果任何参数⊠ null，⊠定的⊠在拓扑模式的 `edge` 表中不存在，`acurve` 不是 `LINESTRING`，或者修改将改⊠底⊠拓扑⊠⊠，⊠会抛出⊠⊠。

如果 `acurve` 几何⊠象的空⊠参考系⊠（SRID）与拓扑⊠⊠不同，那么将抛出异常。

如果新的 `acurve` 不是⊠⊠⊠（simple line），⊠会抛出⊠⊠。

如果将⊠从旧位置移⊠到新位置会碰到障碍物，那么会抛出⊠⊠。

可用性：1.1.0

增⊠版 2.0.0 添加了拓扑一致性⊠制⊠行功能

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM: Topo-Geo 和 Topo-Net 3: ⊠⊠例程 X.3.6

示例

```
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
```

相关信息

ST_AddEdgeModFace

ST_RemEdgeModFace

ST_ModEdgeSplit

9.6.8 ST_ModEdgeSplit

ST_ModEdgeSplit — 通⌧沿⌧有⌧⌧建新⌧点、修改原始⌧并添加新⌧来分割⌧。

Synopsis

integer **ST_ModEdgeSplit**(varchar atopology, integer anedge, geometry apoint);

描述

通⌧沿⌧有⌧⌧建新⌧点、修改原始⌧并添加新⌧来分割⌧。相⌧地更新所有⌧有的⌧接⌧和关系。返回新添加的⌧点的⌧⌧符。

可用性：1.1

更改：2.0 - 在之前的版本中，它被⌧⌧命名⌧ ST_ModEdgesSplit

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM: Topo-Geo 和 Topo-Net 3: ⌧⌧例程: X.3.9

示例

```
-- 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
```

相关信息

ST_NewEdgesSplit, ST_ModEdgeHeal, ST_NewEdgeHeal, AddEdge

9.6.9 ST_ModEdgeHeal

ST_ModEdgeHeal — 通⌧⌧除⌧接⌧条⌧的⌧点、修改第一条⌧并⌧除第二条⌧来修复⌧条⌧。返回已⌧除⌧点的 id。

Synopsis

int **ST_ModEdgeHeal**(varchar atopology, integer anedge, integer anotheredge);

描述

通□□除□接□条□的□点、修改第一条□并□除第二条□来修复□条□。返回已□除□点的 id。相□地更新所有□有的□接□和关系。

可用性：2.0

□方法□□了 SQL/MM □范。SQL-MM: Topo-Geo 和 Topo-Net 3: □□例程: X.3.9

相关信息

ST_ModEdgeSplit ST_NewEdgesSplit

9.6.10 ST_NewEdgeHeal

ST_NewEdgeHeal — 通□□除□接□条□的□点、□除□条□并用方向与提供的第一条□相同的□替□它□来修复□条□。

Synopsis

int **ST_NewEdgeHeal**(varchar atopology, integer anedge, integer anotheredge);

描述

通□□除□接□条□的□点、□除□条□并用方向与提供的第一条□相同的□替□它□来修复□条□。返回替□已修复□的新□的 ID。相□地更新所有□有的□接□和关系。

可用性：2.0

□方法□□了 SQL/MM □范。SQL-MM: Topo-Geo 和 Topo-Net 3: □□例程: X.3.9

相关信息

ST_ModEdgeHeal ST_ModEdgeSplit ST_NewEdgesSplit

9.6.11 ST_MoveIsoNode

ST_MoveIsoNode — 在拓扑□□中将一个孤立□点从一个点移□到□一个点。如果新的 `apoint` 几何□象已存在作□一个□点，□会抛出□□。返回移□的描述。

Synopsis

text **ST_MoveIsoNode**(varchar atopology, integer anode, geometry apoint);

描述

将拓扑中的孤立⊠点从一点移⊠到⊠一点。如果新的 `apoint` 几何⊠形作⊠⊠点存在，⊠会抛出⊠⊠。

如果任何参数⊠ null，`apoint` 不是一个点，⊠有⊠点不是孤立的（是⊠有⊠的起点或⊠点），新⊠点位置与⊠有⊠相交（即使在端点⊠也相交），或新位置在不同的面（自 3.2.0 版本起），⊠会抛出异常。

如果点几何的空⊠参考系⊠ (srid) 与拓扑不同，⊠会引⊠异常。

可用性: 2.0.0

增⊠：3.2.0 确保⊠点不能移⊠到不同的面

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM: Topo-Net 例程: X.3.2

示例

```
-- 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
```

相关信息

ST_AddIsoNode

9.6.12 ST_NewEdgesSplit

ST_NewEdgesSplit — 通⊠沿⊠有⊠⊠建新⊠点、⊠除原始⊠并用⊠条新⊠替⊠它来分割⊠。返回⊠建的⊠接新⊠的新⊠点的 id。

Synopsis

integer **ST_NewEdgesSplit**(varchar atopology, integer anedge, geometry apoint);

描述

通⊠在当前⊠上⊠建一个具有点位置 `apoint` 的新⊠点，⊠除原始⊠并用⊠个新⊠替⊠它，来分割具有⊠ IDanedge 的⊠。返回⊠接新⊠的新⊠点的 ID。相⊠地更新所有已⊠接的⊠和关系。

如果点几何⊠象的空⊠参考系⊠（SRID）与拓扑⊠⊠不同，`apoint` 不是点几何⊠象，点⊠ null，点已⊠存在作⊠一个⊠点，⊠与⊠有⊠不⊠⊠，或者点不在⊠上，⊠会抛出异常。

可用性：1.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM: Topo-Net 例程: X.3.8

示例

```
-- 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
```

相关信息

ST_ModEdgeSplit ST_ModEdgeHeal ST_NewEdgeHeal AddEdge

9.6.13 ST_RemoveIsoNode

ST_RemoveIsoNode — 删除孤立点并返回操作描述。如果点不是孤立的（是线的开始或结束），会引发异常。

Synopsis

text **ST_RemoveIsoNode**(varchar atopology, integer anode);

描述

移除一个孤立点并返回操作的描述。如果点不是孤立的（是线的起点或终点），会抛出异常。

可用性：1.1

该方法符合了 SQL/MM 规范。SQL-MM: Topo-Geo 和 Topo-Net 3: 附录例程: X+1.3.3

示例

```
-- Remove an isolated node with no face  --
SELECT topology.ST_RemoveIsoNode('ma_topo',  7 ) As result;
         result
------------------------
 Isolated node 7 removed
```

相关信息

ST_AddIsoNode

9.6.14 ST_RemoveIsoEdge

ST_RemoveIsoEdge — 删除孤立的线并返回操作的描述。如果线未被隔离，会引发异常。

Synopsis

text **ST_RemoveIsoEdge**(varchar atopology, integer anedge);

描述

移除一个孤立的⊠并返回操作的描述。如果⊠不是孤立的，将抛出异常。

可用性：1.1

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM: Topo-Geo 和 Topo-Net 3: ⊠⊠例程: X+1.3.3

示例

```
-- Remove an isolated node with no face  --
SELECT topology.ST_RemoveIsoNode('ma_topo',  7 ) As result;
         result
-------------------------
 Isolated node 7 removed
```

相关信息

ST_AddIsoNode

9.7 拓扑⊠⊠器

9.7.1 GetEdgeByPoint

GetEdgeByPoint — ⊠找与⊠定点相交的⊠的⊠ ID。

Synopsis

integer **GetEdgeByPoint**(varchar atopology, geometry apoint, float8 tol1);

描述

⊠索与 Point 相交的⊠的 id。

⊠定拓扑、POINT 和容差，⊠函数返回一个整数 (id-edge)。如果容差 = 0，⊠⊠点必⊠与⊠相交。

如果 `apoint` 与⊠不相交，返回 0。

如果使用的容差（tolerance）大于 0，并且在某点附近存在多条⊠，⊠会引⊠异常。

Note
如果 tolerance = 0，⊠⊠函数使用 ST_Intersects，否⊠使用 ST_DWithin。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性: 2.0.0

示例

⍰些示例使用我⍰在 AddEdge中⍰建的⍰

```
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
```

相关信息

AddEdge, GetNodeByPoint, GetFaceByPoint

9.7.2 GetFaceByPoint

GetFaceByPoint — ⍰找与⍰定点相交的面。

Synopsis

integer **GetFaceByPoint**(varchar atopology, geometry apoint, float8 tol1);

描述

⍰找由点引用且具有⍰定容差的面。

⍰函数将有效地⍰找与以⍰点⍰中心、以公差⍰半径的⍰相交的面。

如果没有面与⍰定⍰⍰位置相交，⍰返回 0（通用面）。

如果多个面与⍰⍰位置相交，⍰会引⍰异常。

可用性: 2.0.0

增⍰：3.2.0 更高效的⍰施和更清晰的契⍰，停止使用无效的拓扑。

示例

```
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
```

相关信息

GetFaceContainingPoint, AddFace, GetNodeByPoint, GetEdgeByPoint

9.7.3 GetFaceContainingPoint

GetFaceContainingPoint — 找包含点的面。

Synopsis

integer **GetFaceContainingPoint**(text atopology, geometry apoint);

描述

返回包含点的面的 id。
如果点落在面界上，会引异常。

Note
函数依于有效的拓扑，使用接和面。

可用性：3.2.0

相关信息

ST_GetFaceGeometry

9.7.4 GetNodeByPoint

GetNodeByPoint — 找某个点位置点的点 ID。

Synopsis

integer **GetNodeByPoint**(varchar atopology, geometry apoint, float8 tol1);

描述

⍰索某个点位置⍰的⍰点 id。

⍰定拓扑（topology）、POINT 和容差（tolerance），⍰函数返回一个整数（id-node）。如果公差 = 0 表示精确交集，否⍰从区⍰中⍰索⍰点。

如果 `apoint` 不与⍰点相交，⍰返回 0（零）。

如果⍰置容差（tolerance）大于 0，并且在某点附近存在多个⍰点（node），那么可能会引⍰异常。

Note
如果 tolerance = 0，⍰⍰函数使用 ST_Intersects，否⍰使用 ST_DWithin。

⍰个函数是由 GEOS 模⍰⍰行的。

可用性: 2.0.0

示例

⍰些示例使用我⍰在 AddEdge中⍰建的⍰

```
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
```

相关信息

AddEdge, GetEdgeByPoint, GetFaceByPoint

9.7.5 GetTopologyID

GetTopologyID — 返回 topology.topology 表中⍰定拓扑名称的拓扑的 id。

Synopsis

integer **GetTopologyID**(varchar toponame);

描述

返回 topology.topology 表中⍰定拓扑名称的拓扑的 id。

可用性：1.1

示例

```
SELECT topology.GetTopologyID('ma_topo') As topo_id;
 topo_id
---------
       1
```

相关信息

CreateTopology, DropTopology, GetTopologyName, GetTopologySRID

9.7.6 GetTopologySRID

GetTopologySRID — 返回拓扑表中⌧定拓扑名称的拓扑的 SRID。

Synopsis

integer **GetTopologyID**(varchar toponame);

描述

在⌧定拓扑名称的情况下，返回 topology.topology 表中拓扑的空⌧参考 ID。
可用性: 2.0.0

示例

```
SELECT topology.GetTopologySRID('ma_topo') As SRID;
 SRID
-------
  4326
```

相关信息

CreateTopology, DropTopology, GetTopologyName, GetTopologyID

9.7.7 GetTopologyName

GetTopologyName — 返回⌧定拓扑 ID 的拓扑（架⌧）名称。

Synopsis

varchar **GetTopologyName**(integer topology_id);

描述

在⌧定拓扑的拓扑 ID 的情况下，从 topology.topology 表中返回拓扑的拓扑名称（架⌧）。
可用性：1.1

示例

```
SELECT topology.GetTopologyName(1) As topo_name;
 topo_name
-----------
 ma_topo
```

相关信息

CreateTopology, DropTopology, GetTopologyID, GetTopologySRID

9.7.8 ST_GetFaceEdges

ST_GetFaceEdges — 返回一☒有序的☒，☒些☒界☒定了 aface。

Synopsis

getfaceedges_returntype **ST_GetFaceEdges**(varchar atopology, integer aface);

描述

返回一☒有序的☒，☒些☒界☒定了 aface。每个☒出包括一个序列号和☒ ID。序列号从 1 开始。
每个☒☒的枚☒从具有最小☒☒符的☒开始。☒的☒序遵循左手定☒（合界面位于每个有向☒的左☒）。
可用性：2.0

☒方法☒☒了 SQL/MM ☒范。SQL-MM 3 Topo-Geo 和 Topo-Net 3: ☒☒例程: X.3.5

示例

```
-- 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;
```

相关信息

GetRingEdges, AddFace, ST_GetFaceGeometry

9.7.9 ST_GetFaceGeometry

ST_GetFaceGeometry — 返回⌧定拓扑中具有指定面 ID 的多⌧形。

Synopsis

geometry **ST_GetFaceGeometry**(varchar atopology, integer aface);

描述

返回⌧定拓扑中具有指定面 ID 的多⌧形。从⌧成面的⌧⌧建多⌧形。

可用性：1.1

⌧方法⌧⌧了 SQL/MM ⌧范。SQL-MM 3 Topo-Geo 和 Topo-Net 3: ⌧⌧例程: X.3.16

示例

```
-- 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))
```

相关信息

AddFace

9.7.10 GetRingEdges

GetRingEdges — 返回按⌧序排列的⌧⌧⌧符集合，⌧些⌧⌧符是通⌧沿⌧定⌧的一⌧行走⌧遇到的。

Synopsis

getfaceedges_returntype **GetRingEdges**(varchar atopology, integer aring, integer max_edges=null);

描述

返回在⊠定⊠上行走⊠遇到的有符号⊠⊠⊠符的有序集合。每个⊠出由一个序列和一个⊠符号的⊠ id ⊠成。序列号从⊠ 1 开始。

如果⊠⊠正⊠⊠ id，⊠行走从相⊠⊠⊠的左⊠开始，并遵循⊠⊠方向。如果您⊠⊠⊠⊠⊠ ID，⊠行走从其右⊠开始并向后走。

如果 `max_edges` 不⊠ null，⊠⊠函数返回的⊠⊠数不超⊠⊠些⊠⊠。在⊠理可能无效的拓扑⊠，⊠是一个安全参数。

Note
⊠函数使用⊠⊠⊠接元数据。

可用性: 2.0.0

相关信息

ST_GetFaceEdges, GetNodeEdges

9.7.11 GetNodeEdges

GetNodeEdges — 返回与⊠定⊠点相关的一⊠有序⊠。

Synopsis

getfaceedges_returntype **GetNodeEdges**(varchar atopology, integer anode);

描述

返回与⊠定⊠点相关的一⊠有序⊠。每个⊠出由一个序列和一个⊠符号的⊠ id ⊠成。序列号从⊠ 1 开始。上升沿从⊠定⊠点开始。⊠⊠沿⊠束于⊠定⊠点。封⊠的⊠将出⊠⊠次（都有⊠个⊠志）。⊠序是从北行开始⊠⊠⊠方向。

Note
⊠函数⊠算排序而不是从元数据⊠出，因此可用于⊠建⊠⊠⊠接。

可用性：2.0

相关信息

getfaceedges_returntype, GetRingEdges, ST_Azimuth

9.8 拓扑⊠理

9.8.1 Polygonize

Polygonize — ⊠找并注册由拓扑⊠定⊠的所有面。

Synopsis

text **Polygonize**(varchar toponame);

描述

注册可以⊠建拓扑⊠基元的所有面。

假⊠目⊠拓扑不包含自相交⊠。

Note
已⊠⊠⊠出已知的面，因此在同一拓扑上多次⊠用 Polygonize 是安全的。

Note
⊠函数不使用也不⊠置⊠表的 next_left_edge 和 next_right_edge 字段。

可用性: 2.0.0

相关信息

AddFace, ST_Polygonize

9.8.2 AddNode

AddNode — 将点⊠点添加到指定拓扑模式的⊠点表中，并返回新⊠点的 nodeid。如果点已⊠作⊠⊠点存在，⊠返回⊠有的 nodeid。

Synopsis

integer **AddNode**(varchar toponame, geometry apoint, boolean allowEdgeSplitting=false, boolean computeContainingFace=false);

描述

将点⊠点添加到指定拓扑方案的⊠点表中。AddEdge函数在⊠用⊠会自⊠添加⊠的起点和⊠点，因此无需⊠式添加⊠的⊠点。

如果⊠⊠任何穿⊠⊠点的⊠，⊠会引⊠异常或分割⊠，具体取决于 `allowedEdgeSplitting` 参数⊠。

如果 `computeContainingFace` ⊠真，新添加的⊠点将⊠算出正确的包含面。

Note
如果 `apoint` 几何⊠形已作⊠⊠点存在，⊠不会添加⊠⊠点，但会返回⊠有的 nodeid。

可用性: 2.0.0

示例

```
SELECT topology.AddNode('ma_topo', ST_GeomFromText('POINT(227641.6 893816.5)', 26986) ) As ↩
    nodeid;
-- result --
nodeid
--------
 4
```

相关信息

AddEdge, CreateTopology

9.8.3 AddEdge

AddEdge — 使用指定的⊠串几何将⊠串⊠添加到⊠表，并将关⊠的起点和⊠点添加到指定拓扑方案的点⊠点表，并返回新（或⊠有）⊠的 edgeid。

Synopsis

integer **AddEdge**(varchar toponame, geometry aline);

描述

使用指定的⊠串几何⊠形将⊠添加到⊠表，并将关⊠⊠点添加到指定 `toponame` 模式的⊠点表，并返回新⊠⊠或⊠有⊠⊠的 edgeid。新添加的⊠⊠⊠都有“宇宙”面，并与其自身相⊠。

Note
如果 `aline` 几何⊠形与⊠有⊠串⊠交叉、重⊠、包含或被⊠有⊠串⊠包含，⊠会引⊠⊠⊠并且不会添加⊠。

Note
`aline` 的几何形状必⊠具有与⊠拓扑定⊠的相同的 `srid`，否⊠将引⊠无效的空⊠参考系⊠⊠⊠。

⊠个函数是由 GEOS 模⊠⊠行的。

Warning
AddEdge is deprecated as of 3.5.0. Use TopoGeo_AddLineString instead.

可用性: 2.0.0

示例

```
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
```

相关信息

TopoGeo_AddLineString, CreateTopology, Section 4.5

9.8.4 AddFace

AddFace — 将面基元注册到拓扑并☒取其☒☒符。

Synopsis

integer **AddFace**(varchar toponame, geometry apolygon, boolean force_new=false);

描述

将面基元注册到拓扑并☒取其☒☒符。

☒于新添加的面，形成其☒界的☒以及面中包含的☒将更新☒在 left_face 和 right_face 字段中具有正确的☒。面中包含的孤立☒点也将更新☒具有正确的 contains_face 字段☒。

Note
☒函数不使用也不☒置☒表的 next_left_edge 和 next_right_edge 字段。

假定目☒拓扑有效（不包含自相交☒）。如果出☒以下情况，☒会引☒异常：多☒形☒界未完全由☒有☒定☒，或者多☒形与☒有面重☒。

如果多☒形几何体已☒作☒面存在，☒：如果 `force_new` ☒假（默☒），☒返回☒有面的面 id；如果 `force_new` ☒ true，新的 id 将被分配☒新☒生的面。

Note
□□有面□行新配准（force_new=true）□，不会采取任何措施来解决□、□点和关系表中□□有面的□挂引用，也不会更新□有面□□的 MBR 字段。□由□用者自行□理。

Note
多□形几何□形必□具有与□拓扑定□的相同的 srid，否□将引□无效的空□参考系□□□。

可用性: 2.0.0

示例

```
-- 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
```

相关信息

AddEdge, CreateTopology, Section 4.5

9.8.5 ST_Simplify

ST_Simplify — 使用 Douglas-Peucker 算法返回⌧定 TopoGeometry 的“⌧化”几何版本。

Synopsis

geometry **ST_Simplify**(TopoGeometry tg, float8 tolerance);

描述

在每个⌧件⌧上使用 Douglas-Peucker 算法返回⌧定 TopoGeometry 的“⌧化”几何版本。

Note
返回的几何⌧形可能不⌧⌧或无效。
拆分⌧件⌧⌧可能有助于保持⌧⌧性/有效性。

⌧个函数是由 GEOS 模⌧⌧行的。

可用性：2.1.0

相关信息

几何 ST_Simplify, ST_IsSimple, ST_IsValid, ST_ModEdgeSplit

9.8.6 RemoveUnusedPrimitives

RemoveUnusedPrimitives — ⌧除定⌧⌧有 TopoGeometry ⌧象不需要的拓扑基元。

Synopsis

int **RemoveUnusedPrimitives**(text topology_name, geometry bbox);

描述

⌧找表示⌧有 TopoGeometry ⌧象并不⌧格需要的所有⌧元（⌧点、⌧、面）并将其⌧除，从而保持拓扑有效性（⌧⌧接、面⌧⌧）和 TopoGeometry 空⌧占用。

不会⌧建新的⌧元⌧⌧符，而是⌧展⌧有⌧元以包括合并的面（在⌧除⌧⌧）或修复的⌧（在⌧除⌧点⌧）。

可用性：3.3.0

相关信息

ST_ModEdgeHeal, ST_RemEdgeModFace

9.9 拓扑几何□造函数

9.9.1 CreateTopoGeom

CreateTopoGeom — 从拓扑元素数□□建一个新的拓扑几何□象 - tg_type: 1:[多] 点, 2:[多] □, 3:[多] 多□形, 4: 集合

Synopsis

topogeometry **CreateTopoGeom**(varchar toponame, integer tg_type, integer layer_id, topoelementarray tg_objs);
topogeometry **CreateTopoGeom**(varchar toponame, integer tg_type, integer layer_id);

描述

□由 `layer_id` 表示的□□□建一个拓扑几何□象，并将其注册到拓扑名称架□的关系表中。

`tg_type` 是一个整数：1：[多] 点（点）、2：[多] □（□性）、3：[多] 多□形（面）、4：集合。`layer_id` 是 topology.layer 表中的□ id。

点状□由一□□点形成，□性□由一□□形成，区域□由一□面形成，集合可以由□点、□和面的混合形成。

省略□件数□会生成一个空的 TopoGeometry □象。

可用性：1.1

示例：从□有□形成

在"ri_topo" 模式中□"layer 2"（我□的"ri_roads"）□建一个拓扑几何□象，□型□ (2) LINE，用于第一条□（我□在 `ST_CreateTopoGeo` 中加□的□）。

```
INSERT INTO ri.ri_roads(road_name, topo) VALUES('Unknown', topology.CreateTopoGeom('ri_topo ↩
    ',2,2,'{{1,2}}'::topology.topoelementarray);
```

示例：将面状几何□象□□□最佳猜□的拓扑几何□象

假□我□有□□由面的集合形成的几何形状。例如，我□有□□表，并且想知道每个□□的拓扑几何形状。如果我□的数据完全一致，我□可以□□做：

```
-- 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;
```

相关信息

AddTopoGeometryColumn, toTopoGeom ST_CreateTopoGeo, ST_GetFaceGeometry, TopoElementArray, TopoElementArray_Agg

9.9.2 toTopoGeom

toTopoGeom — 将☒☒几何☒形☒☒☒拓扑几何☒形。

Synopsis

topogeometry **toTopoGeom**(geometry geom, varchar toponame, integer layer_id, float8 tolerance);
topogeometry **toTopoGeom**(geometry geom, topogeometry topogeom, float8 tolerance);

描述

将☒☒的几何☒形☒☒☒拓扑几何☒形TopoGeometry。

表示☒入几何所需的拓扑基元将被添加到底☒拓扑中，可能会拆分☒有拓扑，并且它☒将与关系表中的☒出 TopoGeometry 相关☒。

☒有的 TopoGeometry ☒象（拓扑几何可能除外，如果☒定的☒）将保留其形状。

当⊠出公差⊠，它将用于将⊠入几何体捕捉到⊠有⊠元。

在第一种形式中，将⊠⊠定拓扑 (toponame) 的⊠定⊠⊠ (layer_id) ⊠建新的 TopoGeometry。

在第二种形式中，⊠⊠⊠生的⊠元将被添加到⊠先存在的 TopoGeometry (topogeom) 中，可能会⊠其最⊠形状添加空⊠。要⊠新形状完全取代旧形状，⊠参⊠clearTopoGeom。

可用性：2.0

增⊠：2.1.0 添加了采用⊠有 TopoGeometry 的版本。

示例

⊠是一个完整的独立工作流程

```
-- 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
  );
```

相关信息

CreateTopology, AddTopoGeometryColumn, CreateTopoGeom, TopologySummary, clearTopoGeom

9.9.3 TopoElementArray_Agg

TopoElementArray_Agg — 返回一☒ element_id、☒型数☒ (topoelements) 的 `topoelementarray`。

Synopsis

topoelementarray **TopoElementArray_Agg**(topoelement set tefield);

描述

用于从一☒ TopoElement☒建TopoElementArray。
可用性: 2.0.0

示例

```
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}}
```

相关信息

TopoElement, TopoElementArray

9.9.4 TopoElement

TopoElement — 将拓扑几何☒☒☒拓扑元素。

Synopsis

topoelement **TopoElement**(topogeometry topo);

描述

将 TopoGeometry☒☒☒TopoElement。
可用性：3.4.0

示例

⊠是一个完整的独立工作流程

```
-- 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;
```

相关信息

TopoElementArray_Agg, TopoGeometry, TopoElement

9.10 拓扑几何⊠⊠器

9.10.1 clearTopoGeom

clearTopoGeom — 清除拓扑几何的内容。

Synopsis

topogeometry **clearTopoGeom**(topogeometry topogeom);

描述

清除 TopoGeometry的内容，将其⊠⊠空内容。主要与toTopoGeom ⊠合使用，以替⊠⊠有⊠象的形状以及更高⊠次⊠⊠中的任何依⊠⊠象。

可用性：2.1

示例

```
-- Shrink all TopoGeometry polygons by 10 meters
UPDATE nei_topo SET topo = ST_Buffer(clearTopoGeom(topo), -10);
```

相关信息

toTopoGeom

9.10.2 TopoGeom_addElement

TopoGeom_addElement — 将元素添加到 TopoGeometry 的定⊠中。

Synopsis

topogeometry **TopoGeom_addElement**(topogeometry tg, topoelement el);

描述

将 TopoElement 添加到 TopoGeometry 对象的定义中。如果元素已经是定义的一部分，则不会出错。

可用性：2.3

示例

```
-- Add edge 5 to TopoGeometry tg
UPDATE mylayer SET tg = TopoGeom_addElement(tg, '{5,2}');
```

相关信息

TopoGeom_remElement, CreateTopoGeom

9.10.3 TopoGeom_remElement

TopoGeom_remElement — 从 TopoGeometry 的定义中删除元素。

Synopsis

topogeometry **TopoGeom_remElement**(topogeometry tg, topoelement el);

描述

从 TopoGeometry 对象的定义中删除 TopoElement。

可用性：2.3

示例

```
-- Remove face 43 from TopoGeometry tg
UPDATE mylayer SET tg = TopoGeom_remElement(tg, '{43,3}');
```

相关信息

TopoGeom_addElement, CreateTopoGeom

9.10.4 TopoGeom_addTopoGeom

TopoGeom_addTopoGeom — 将一个 TopoGeometry 的元素添加到另一个 TopoGeometry 的定义中。

Synopsis

topogeometry **TopoGeom_addTopoGeom**(topogeometry tgt, topogeometry src);

描述

将一个TopoGeometry 的元素添加到⌧一个 TopoGeometry 的定⌧中，如果需要保存源⌧象中的所有元素，可能会将其⌧存⌧型（⌧型属性）更改⌧集合。

⌧个 TopoGeometry ⌧象需要⌧⌧“相同”拓扑⌧行定⌧，并且如果按⌧次⌧⌧定⌧，⌧需要由同一子⌧的元素⌧成。

可用性：3.2

示例

```
-- 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
);
```

相关信息

TopoGeom_addElement, clearTopoGeom, CreateTopoGeom

9.10.5 toTopoGeom

toTopoGeom — 将几何形状添加到⌧有拓扑几何⌧形中。

描述

参考toTopoGeom。

9.11 拓扑几何⌧⌧器

9.11.1 GetTopoGeomElementArray

GetTopoGeomElementArray — 返回一个 `topoelementarray`（包含拓扑元素的数⌧），其中包含⌧定 TopoGeometry 的拓扑元素和⌧型（原始元素）。

Synopsis

topoelementarray **GetTopoGeomElementArray**(varchar toponame, integer layer_id, integer tg_id);

topoelementarray **GetTopoGeomElementArray**(topogeometry tg);

描述

返回一个TopoElementArray，其中包含⊠定 TopoGeometry（原始元素）的拓扑元素和⊠型。⊠与 GetTopoGeomElements ⊠似，只不⊠它将元素作⊠数⊠而不是数据集返回。

tg_id 是 topology.layer 表中 `layer_id` 表示的⊠⊠中拓扑中的拓扑几何⊠象的拓扑几何 ID。

可用性：1.1

示例

相关信息

GetTopoGeomElements, TopoElementArray

9.11.2 GetTopoGeomElements

GetTopoGeomElements — 返回一⊠ `topoelement` ⊠象，其中包含⊠定 TopoGeometry（原始元素）的拓扑 element_id、element_type。

Synopsis

setof topoelement **GetTopoGeomElements**(varchar toponame, integer layer_id, integer tg_id);

setof topoelement **GetTopoGeomElements**(topogeometry tg);

描述

返回一⊠与基本拓扑元素TopoElement（1：⊠点，2：⊠，3：面）⊠⊠的 element_id、element_type（topoelements），⊠些元素⊠成了 `toponame` 模式中的⊠定拓扑几何⊠象。

tg_id 是 topology.layer 表中 `layer_id` 表示的⊠⊠中拓扑中的拓扑几何⊠象的拓扑几何 ID。

可用性: 2.0.0

示例

相关信息

GetTopoGeomElementArray, TopoElement, TopoGeom_addElement, TopoGeom_remElement

9.11.3 ST_SRID

ST_SRID — 返回拓扑几何的空⊠参考⊠⊠符。

Synopsis

integer **ST_SRID**(topogeometry tg);

描述

返回 ST_Geometry spatial_ref_sys 表中定⊠的空⊠参考系⊠⊠⊠号。⊠参⊠ Section 4.5

Note
spatial_ref_sys 表⊠ PostGIS 已知的所有参考系⊠行⊠目，并用于从一个空⊠参考系⊠⊠⊠⊠一个空⊠参考系。如果⊠划⊠⊠几何，⊠⊠必确保具有正确的空⊠参考系⊠⊠⊠号。

可用性：3.2.0

⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 14.1.5

示例

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)',4326));
                --result
                4326
```

相关信息

Section 4.5, ST_SetSRID, ST_Transform, ST_SRID

9.12 拓扑几何⊠出

9.12.1 AsGML

AsGML — 返回拓扑几何的 GML 表示形式。

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);

描述

返回 GML3 版本格式的拓扑几何的 GML 表示形式。如果未指定 `nsprefix_in`，⊠使用 `gml`。⊠ nsprefix ⊠入一个空字符串以⊠取非限定名称空⊠。精度（默⊠⊠：15）和⊠⊠（默⊠⊠ 1）参数（如果⊠定）将原封不⊠地⊠⊠到 ST_AsGML 的底⊠⊠用。

`VisitedTable` 参数（如果⍰定）用于跟踪⍰⍰⍰的 Node 和 Edge 元素，以便使用交叉引用 (xlink:xref) 而不是重复定⍰。⍰表⍰⍰有（至少）⍰个整数字段：“element_type” 和 “element_id”。⍰用用⍰必⍰⍰⍰定表具有⍰取和写入⍰限。⍰了⍰得最佳性能，⍰按 `element_type` 和 `element_id` 的⍰序定⍰索引。此⍰索引将通⍰向字段添加唯一⍰束来自⍰⍰建。例子：

```
CREATE TABLE visited (
  element_type integer, element_id integer,
  unique(element_type, element_id)
);
```

`idprefix` 参数（如果⍰定）将被添加到 Edge 和 Node ⍰⍰⍰⍰符之前。

`gmlver` 参数（如果⍰定）将被⍰⍰到底⍰ ST_AsGML。默⍰⍰ 3。

可用性: 2.0.0

示例

⍰使用了我⍰在CreateTopoGeom中⍰建的拓扑几何体

```
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>
```

与之前的⊠⊠相同，没有命名空⊠

```
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>
```

相关信息

CreateTopoGeom, ST_CreateTopoGeo

9.12.2 AsTopoJSON

AsTopoJSON — 返回拓扑几何的 TopoJSON 表示形式。

Synopsis

text **AsTopoJSON**(topogeometry tg, regclass edgeMapTable);

描述

返回拓扑几何的 TopoJSON 表示形式。如果 edgeMapTable 不⊠空，它将用作⊠⊠⊠符到弧索引的⊠找/存⊠映射。⊠是⊠了能⊠在最⊠文档中允⊠⊠⊠的“arcs”数⊠。

⊠表（如果⊠定）⊠具有“serial”⊠型的“arc_id”字段和整数⊠型的“edge_id”字段；⊠代⊠将⊠⊠表中的“edge_id”，因此建⊠在⊠字段上添加索引。

Note
TopoJSON ⊠出中的弧索引是从 0 开始的，但在“edgeMapTable”表中它⊠是从 1 开始的。

除了⊠函数返回的片段之外，完整的 TopoJSON 文档⊠需要包含⊠⊠的弧和一些⊠⊠。⊠参⊠ TopoJSON ⊠范。

可用性：2.1.0

增⊠：2.2.1 添加了⊠ puntal ⊠入的支持

相关信息

ST_AsGeoJSON

示例

```
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 拓扑空□关系

9.13.1 Equals

Equals — 如果□个拓扑几何由相同的拓扑基元□成，□返回 true。

Synopsis

boolean **Equals**(topogeometry tg1, topogeometry tg2);

描述

如果□个拓扑几何由相同的拓扑基元□成：面、□、□点，□返回 true。

Note
作□几何集合的拓扑几何不支持此函数。它也无法比□不同拓扑的拓扑几何形状。

可用性：1.1.0

□函数支持 3d 并且不会□失 z-index。

示例

相关信息

GetTopoGeomElements, ST_Equals

9.13.2 Intersects

Intersects — 如果⊠个拓扑几何中的任何一⊠⊠元相交，⊠返回 true。

Synopsis

boolean **Intersects**(topogeometry tg1, topogeometry tg2);

描述

如果⊠个拓扑几何中的任何一⊠⊠元相交，⊠返回 true。

Note
作⊠几何集合的拓扑几何不支持此函数。它也无法比⊠不同拓扑的拓扑几何形状。目前也不支持分⊠拓扑几何（由其他拓扑几何⊠成的拓扑几何）。

可用性：1.1.0

⊠函数支持 3d 并且不会⊠失 z-index。

示例

```
```

相关信息

ST_Intersects

9.14 ⊠入和⊠出拓扑

⊠建拓扑以及关⊠的拓扑⊠后，您可能希望将它⊠⊠出⊠基于文件的格式以⊠行⊠份或⊠⊠到⊠一个数据⊠中。

使用 PostgreSQL 的⊠准⊠⊠/恢复工具是有⊠⊠的，因⊠拓扑由一⊠表（基元⊠ 4 个，⊠⊠任意数量）和元数据表中的⊠⊠（topology.topology 和 topology.layer）⊠成。此外，拓扑⊠⊠符在数据⊠中并不是唯一的，因此在恢复拓扑⊠需要更改拓扑参数。

⊠了⊠化拓扑的⊠出/恢复，提供了一⊠可⊠行文件：`pgtopo_export` 和`pgtopo_import`。用法示例：

```
pgtopo_export dev_db topo1 | pgtopo_import topo1 | psql staging_db
```

9.14.1 使用拓扑⊠出器

pgtopo_export 脚本采用数据⊠和拓扑的名称，并⊠出⊠⊠文件，⊠文件可用于将拓扑（和关⊠的⊠）⊠入到新数据⊠中。

默⊠情况下，pgtopo_export 将⊠⊠文件写入⊠准⊠出，以便可以将其通⊠管道⊠⊠到 pgtopo_import 或重定向到文件（拒⊠写入⊠端）。您可以⊠⊠使用 -f 命令行开关指定⊠出文件名。

默⊠情况下，pgtopo_export 包含⊠⊠⊠定拓扑定⊠的所有⊠的⊠⊠。⊠可能是比您需要的更多的数据，或者可能不起作用（如果您的⊠⊠表具有复⊠的依⊠关系），在⊠种情况下，您可以使用 --skip-layers 开关⊠求跳⊠⊠⊠并⊠独⊠理⊠些⊠⊠。

⊠用 pgtopo_export ⊠使用 --help（或⊠写的 -h）指令将始⊠打印⊠短的用法⊠明字符串。

⊠⊠文件格式是 pgtopo_export 目⊠的⊠⊠ tar 存档，其中至少包含一个⊠有格式版本信息的 pgtopo_dump_version 文件。从版本 1 开始，⊠目⊠包含制表符分隔的 CSV 文件，其中包含拓扑基元表的数据（⊠点、⊠、面、关系）、与其关⊠的拓扑和⊠⊠⊠以及（除非⊠出 --skip-layers）自定⊠ -format PostgreSQL ⊠⊠表⊠告⊠⊠定拓扑的⊠。

9.14.2 使用拓扑⊠入器

pgtopo_import 脚本采用 pgtopo_export 格式的拓扑⊠⊠和要⊠建的拓扑的名称，并⊠出重建拓扑和关⊠⊠的 SQL 脚本。

生成的 SQL 文件将包含以下⊠句：⊠建具有⊠定名称的拓扑、在其中加⊠原始数据、通⊠将所有 TopoGeometry ⊠正确⊠接到其正确的拓扑来恢复和注册所有拓扑⊠。

默⊠情况下，pgtopo_import 从⊠准⊠入⊠取⊠⊠，以便它可以与管道中的 pgtopo_export ⊠合使用。您可以⊠⊠使用 -f 命令行开关指定⊠入文件名。

默⊠情况下，pgtopo_import 在⊠出 SQL 文件中包含用于恢复⊠⊠中找到的所有⊠的代⊠。

如果您的目⊠数据⊠已⊠具有与⊠⊠中的表同名的表，⊠可能是不需要的或不起作用的。在⊠种情况下，您可以使用 --skip-layers 命令⊠求跳⊠⊠并⊠独（或稍后）⊠理⊠些⊠。

可以使用 --only-layers 命令生成⊠加⊠⊠并将⊠⊠接到命名拓扑的 SQL。⊠⊠于在解决命名冲突后加⊠⊠⊠或将⊠⊠⊠接到不同的拓扑（例如起始拓扑的空⊠⊠化版本）非常有用。

Chapter 10

☒格数据管理、☒☒和☒用程序

10.1 加☒和☒建☒格

☒于大多数用例，您将通☒使用打包的 raster2pgsql ☒格加☒器加☒☒有☒格文件来☒建 PostGIS ☒格。

10.1.1 使用 raster2pgsql 加☒☒格

raster2pgsql 是一个用于将 GDAL 支持的☒格格式加☒到适合加☒到 PostGIS ☒格表的 SQL 的☒格加☒器可☒行文件。它能☒加☒文件☒中的☒格文件，同☒☒建☒格的概☒。

由于 raster2pgsql 最常被☒☒☒ PostGIS 的一部分（除非您☒☒自己的 GDAL ☒），因此可☒行文件支持的☒格☒型将与 GDAL 依☒☒中☒☒的☒格☒型相同。要☒取特定 raster2pgsql 支持的☒格☒型列表，☒使用 -G 指令。

Note
在从一☒☒☒的☒格☒建特定因子的概☒☒，有可能概☒不会☒☒。☒☒☒http://trac.osgeo.org/postgis/ticket/1764 ☒看一个示例，其中概☒未☒☒。

10.1.1.1 用法示例

使用加☒器☒建☒入文件并将其以 100x100 ☒分☒上☒的示例会☒可能如下所示：

```
# -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
如果您没有将架⊠指定⊠目⊠表名称的一部分，⊠⊠表将在您所⊠接的数据⊠或用⊠的默⊠架⊠中⊠建。

使用 UNIX 管道可以一步完成⊠⊠和上⊠：

```
raster2pgsql -s 4326 -I -C -M *.tif -F -t 100x100 public.demelevation | psql -d gisdb
```

将⊠⊠⊠塞州平面米制航空瓦片加⊠到名⊠ `aerial` 的架⊠中，并⊠建一个全⊠⊠，2 ⊠和 4 ⊠概⊠表，使用复制模式⊠行插入（没有中⊠文件，直接到数据⊠），并且使用“-e”⊠⊠来避免⊠制将所有内容放入一个事⊠中（如果你希望立即⊠看表中的数据而不必等待的⊠）。将⊠格分割成 128x128 像素的瓦片，并⊠用⊠格⊠束。同⊠，包括一个名⊠“filename”的字段，用于存⊠瓦片所来自的文件的名称。

```
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
```

-G 指令⊠出⊠似的列表

```
Available GDAL raster formats:
  Virtual Raster
  GeoTIFF
  National Imagery Transmission Format
  Raster Product Format TOC format
  ECRG TOC format
  Erdas Imagine Images (.img)
  CEOS SAR Image
  CEOS Image
  ...
  Arc/Info Export E00 GRID
  ZMap Plus Grid
  NOAA NGS Geoid Height Grids
```

10.1.1.2 raster2pgsql ⊠⊠

-? ⊠示帮助屏幕。如果您不⊠入任何参数，也会⊠示帮助。

-G 打印支持的⊠格格式。

（**c|a|d|p**）⊠些是相互排斥的⊠⊠：

-c ⊠建新表并用⊠格填充它，⊠是默⊠模式

-a 将⊠格附加到⊠有表。

-d ⊠除表，⊠建新表并用⊠格填充它

-p 准⊠模式，只⊠建表。

⊠格⊠理：⊠用⊠束以在⊠格目⊠中正确注册

-C ⊠用⊠格⊠束——srid、像素大小等，以确保⊠格在 `raster_columns` ⊠⊠中正确配准。

-x 禁用⊠置最大范⊠⊠束。⊠当⊠使用 -C 指令⊠才适用。

-r ⊠置⊠范⊠⊠束（空⊠唯一和⊠尽的切片）。⊠当⊠使用 -C 指令⊠才适用。

⊠格⊠理：用于操作⊠入⊠格数据集的可⊠参数

-s <SRID> 使用指定的 SRID 分配⊠出⊠格。如果未提供或⊠零，⊠将⊠⊠⊠格的元数据以确定适当的 SRID。

-b BAND 从⊠格中提取的波段索引（从 1 开始）。⊠于多个波段索引，用逗号（,）分隔。如果未指定，⊠将提取⊠格的所有波段。

-t TILE_SIZE 将光⊠切割成⊠⊠，以便在每个表行插入一个⊠⊠。 TILE_SIZE 表示⊠ WIDTHxHEIGHT 或⊠置⊠⊠ “auto”，以允⊠加⊠程序使用第一个⊠格⊠算适当的切片大小并将其⊠用于所有⊠格。

-P 填充最右⊠和最底部的⊠⊠，以保⊠所有⊠⊠具有相同的⊠度和高度。

-R, --register 将⊠格注册⊠文件系⊠ (out-db) ⊠格。
⊠⊠格的元数据和⊠格的路径位置存⊠在数据⊠中（而不是像素）。

-l OVERVIEW_FACTOR ⊠建⊠格的概⊠。⊠于多个因素，用逗号（,）分隔。概⊠表名称遵循 o_overview Factor_table 模式，其中概⊠因子是数字概⊠因子的占位符，表将替⊠⊠基表名称。⊠建的概述存⊠在数据⊠中，不受-R 影⊠。⊠注意，生成的 sql 文件将包含主表和概⊠表。

-N NODATA NODATA ⊠用于没有 NODATA ⊠的⊠。

用于操作数据⊠⊠象的可⊠参数

-f COLUMN 指定目⊠⊠格列的名称，默⊠⊠ “rast”

-F 添加包含文件名的列

-n COLUMN 指定文件名列的名称。意味着-F。

-q 将 PostgreSQL ⊠⊠符括在引号中。

-I 在⊠格列上⊠建 GiST 索引。

-M 真空分析⊠格表。

-k 保留空的瓦片并跳⊠每个光⊠波段的 NODATA ⊠⊠⊠。⊠注意，⊠⊠可以⊠省⊠⊠的⊠⊠，但可能会⊠致数据⊠中出⊠更多的⊠⊠行，并且⊠些⊠⊠行不会⊠⊠⊠空的瓦片。

-T tablespace 指定新表的表空⊠。⊠注意，索引（包括主⊠）仍将使用默⊠表空⊠，除非⊠使用了 -X ⊠志。

-X tablespace 指定表的新索引的表空⊠。如果使用 -I ⊠志，⊠适用于主⊠和空⊠索引。

-Y max_rows_per_copy=50 使用复制⊠句而不是插入⊠句。可以⊠⊠指定 max_rows_per_copy；未指定⊠默⊠⊠ 50。

-e ⊠独⊠行每条⊠句，不使用事⊠。

-E ENDIAN 控制生成的光⊠二⊠制⊠出的字⊠⊠序；⊠ XDR 指定 0，⊠ NDR 指定 1（默⊠）；⊠在⊠支持 NDR ⊠出

-V version 指定⊠出格式的版本。默⊠⊠ 0。目前⊠支持 0。

10.1.2 使用 PostGIS ⊠格函数⊠建⊠格

在⊠多情况下，您需要直接在数据⊠中⊠建⊠格和⊠格表。有很多函数可以做到⊠一点。要遵循的一般步⊠。

1. ⊠建一个包含⊠格列的表来保存新的⊠格⊠⊠，可以通⊠以下方式完成：

```
CREATE TABLE myrasters(rid serial primary key, rast raster);
```

2. 有⊠多功能可以帮助⊠⊠⊠一目⊠。如果您⊠建的⊠格不是其他⊠格的衍生品，⊠需要从以下位置开始：ST_MakeEmptyRaster，然后是 ST_AddBand

   您⊠可以从几何⊠形⊠建⊠格。⊠了⊠⊠⊠一点，您可能需要使用 ST_AsRaster 以及其他函数，例如 ST_Union 或 ST_MapAlgebraFct或任何其他地⊠代数函数系列。

   甚至⊠有更多⊠⊠可用于从⊠有表⊠建新⊠格表。例如，您可以使用 ST_Transform在与⊠有投影不同的投影中⊠建⊠格表

3. 最初填充表后，您将需要在⊠格列上⊠建一个空⊠索引，如下所示：

```
CREATE INDEX myrasters_rast_st_convexhull_idx ON myrasters USING gist( ST_ConvexHull( ↩
    rast) );
```

⊠注意 ST_ConvexHull的使用，因⊠大多数⊠格⊠算符都基于⊠格的凸包。

Note
PostGIS ⊠格 2.0 之前的版本基于最小外接矩形而不是凸包。⊠了使空⊠索引正常工作，您需要⊠除它⊠并替⊠⊠基于凸包的索引。

4. 使用 AddRasterConstraints⊠用⊠格⊠束

10.1.3 使用“out db”云⊠格

raster2pgsql 工具使用 GDAL ⊠⊠⊠格数据，并且可以利用 GDAL 的一个关⊠功能：能⊠从⊠程存⊠在云“⊠象存⊠”（例如 AWS S3、Google Cloud Storage）中的⊠格中⊠取数据。

有效使用云存⊠⊠格需要使用“云⊠化”格式。最知名和最广泛使用的是“云⊠化的 GeoTIFF”格式。使用非云格式（例如 JPEG 或未平⊠的 TIFF）将⊠致性能非常差，因⊠系⊠每次需要⊠⊠子集⊠都必⊠下⊠整个⊠格。

首先，将⊠格加⊠到您⊠⊠的云存⊠中。一旦加⊠，您将有一个 URI 来⊠⊠它，可以是“http”URI，有⊠也可以是特定于服⊠的 URI。（例如，“s3://bucket/object”）。要⊠⊠非公共存⊠桶，您需要提供 GDAL 配置⊠⊠来⊠⊠您的⊠接。⊠注意，此命令是从云⊠格⊠取并写入数据⊠。

```
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
```

加⊠表后，您需要通⊠⊠置⊠个⊠限（postgis.enable_outdb_rasters 和postgis.gdal_enabled_drivers）来授予数据⊠从⊠程⊠格⊠取的⊠限。

```
SET postgis.enable_outdb_rasters = true;
SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

⊠了使更改生效并持久化，直接在您的数据⊠上⊠行⊠置。您将需要重新⊠接以体⊠新的⊠置。

```
ALTER DATABASE your_db SET postgis.enable_outdb_rasters = true;
ALTER DATABASE your_db SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

⊠于非公开的⊠格数据，你可能需要提供⊠⊠密⊠来从云端⊠格数据中⊠取。你可以将写入 raster2pgsql ⊠用⊠使用的相同密⊠⊠置⊠在数据⊠内部使用，使用 postgis.gdal_vsi_options 配置。⊠注意，可以通⊠用空格分隔 key=value ⊠来⊠置多个⊠⊠。

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
```

加⊠数据并⊠置⊠限后，您可以使用相同的功能像任何其他⊠格表一⊠与⊠格表⊠行交互。当需要⊠取像素数据⊠，数据⊠将⊠理⊠接到云数据的所有机制。

10.2 栅格目录

PostGIS 附带了两个栅格目录视图。两个视图都利用嵌入在栅格表约束中的信息。因此，由于强制施加了约束，目录视图始终与表中的栅格数据一致。

1. raster_columns 此视图数据库中的所有栅格表列行目。
2. raster_overviews 此视图数据库中的所有栅格表列行目，这些列用作更粗粒度表的概述。当您在加载期间使用 -l 开关时，会生成这种类型的表。

10.2.1 栅格列目录

raster_columns 是数据库中所有栅格类型的栅格表列的目录。它是一种利用表约束的视图，因此即使您从一个数据库的备份恢复一个栅格表，信息也始终保持一致。raster_columns 目录中存在以下列。

如果您未使用加载器创建表或忘记在加载期间指定 -C 标志，则可以在事后使用 AddRasterConstraints 强制执行约束，以便 raster_columns 目录注册有关栅格切片的通用信息。

- r_table_catalog 表所在的数据库。这将始终取当前数据库。
- r_table_schema 栅格表所属的数据库模式。
- r_table_name 栅格表
- r_raster_column 在 r_table_name 表中栅格类型的列。PostGIS 中没有任何内容可以阻止您在每个表中有多个栅格列，因此可以多次列出一个栅格表，并且每个表具有不同的栅格列。
- srid 栅格的空间参考系统标识符。这是 Section 4.5中的条目。
- scale_x 几何空间坐标与像素之间的缩放比例。当栅格列中的所有切片具有相同的 scale_x 并且应用此约束时，此功能才可用。有关更多信息，请参阅 ST_ScaleX。
- scale_y 几何空间坐标与像素之间的缩放比例。当栅格列中的所有切片具有相同的 scale_y 并且应用了 scale_y 约束时，此功能才可用。有关详细信息，请参阅 ST_ScaleY。
- blocksize_x 每个栅格切片的宽度（跨像素数）。有关更多详细信息，请参阅ST_Width。
- blocksize_y 每个栅格切片的宽度（向下的像素数）。有关更多详细信息，请参阅ST_Height。
- same_alignment 一个布尔值，如果所有栅格切片具有相同的对齐方式，则为 true。有关更多详细信息，请参阅ST_SameAlignment。
- Regular_blocking 如果栅格列具有空间唯一性和覆盖范围切片约束，则为 TRUE。否则，它将是 FALSE。
- num_bands 栅格集中每个切片中的波段数。与 ST_NumBands 提供的信息相同 ST_NumBands
- Pixel_types 定义每个波段的像素类型的数组。该数组中的元素数量与波段的数量相同。Pixel_types 是 ST_BandPixelType 中定义的以下类型之一。
- nodata_values 双精度数字数组，表示每个波段的 nodata_value。该数组中的元素数量与波段的数量相同。这些数字定义了大多数操作中要忽略的每个波段的像素值。与ST_BandNoDataValue 提供的信息类似。
- out_db 布尔标志数组，指示栅格波段数据是否在数据库外部存储。该数组中的元素数量与波段的数量相同。
- extent 是栅格集中所有栅格行的范围。如果您要划加载更多数据来更改集的范围，则需要在加载之前运行 DropRasterConstraints 函数，然后在加载后使用AddRasterConstraints 重新应用约束。
- Spatial_index 如果栅格列具有空间索引，则为 true 的布尔值。

10.2.2 ☒格概述

raster_overviews 目☒有关用于概☒的☒格表列的信息以及在使用概☒☒了解的有用的附加信息。概☒表在 raster_columns 和 raster_overviews 中☒行☒目，因☒它☒本身就是☒格，但☒具有作☒高分辨率表的☒低分辨率的漫画化☒像的☒外特殊用途。当您在☒格加☒中使用 -l 开关☒，它☒会与主☒格表一起生成，或者可以使用 AddOverviewConstraints手☒生成。

概☒表包含与其他☒格表相同的☒束以及特定于概☒的附加信息☒束。

Note
raster_overviews 中的信息与 raster_columns 中的信息不重复。如果您需要有关 raster_columns 中存在的概☒表的信息，您可以将 raster_overviews 和 raster_columns ☒接在一起以☒得所需的完整信息集。

☒行概☒的☒个主要原因是：

1. 通常用于快速映射☒小的核心表的低分辨率表示。
2. 它☒的☒算速度通常比其更高分辨率的父☒更快，因☒☒☒更少，每个像素覆盖更多区域。尽管☒算不如它☒支持的高分辨率表那么准确，但它☒在☒多粗略☒算中已☒足☒了。

raster_overviews 目☒包含以下信息。

- o_table_catalog 概☒表所在的数据☒。☒将始☒☒取当前数据☒。
- o_table_schema 概☒☒格表所属的数据☒架☒。
- o_table_name ☒格概☒表名称
- o_raster_column 概☒表中的☒格列。
- r_table_catalog ☒概☒服☒所在的☒格表所在的数据☒。☒将始☒☒取当前数据☒。
- r_table_schema 此概☒服☒所属的☒格表的数据☒架☒。
- r_table_name 此概☒服☒的☒格表。
- r_raster_column 此概☒列服☒的☒格列。
- Overview_factor - ☒是概述表的金字塔☒☒。数字越大，表的分辨率越低。raster2pgsql 如果☒定☒像文件☒，将☒算每个☒像文件的概述并☒独加☒。假定☒☒☒ 1，并且始☒☒原始文件。2 ☒是每个☒☒代表 4 个原始☒☒。例如，如果您有一个包含 5000x5000 像素☒像文件的文件☒，您☒☒将其分☒☒ 125x125，☒☒于每个☒像文件，您的基表将有 (5000*5000)/(125*125) 条☒☒ = 1600，您的 (l=2) o_2 表将有上限 (1600/Power(2,2)) = 400 行，您的 (l=3) o_3 将有上限 (1600/Power(2,3)) = 200 行。如果您的像素不能被☒☒的大小整除，您将得到一些☒☒☒（未完全填充的☒☒）。☒注意，raster2pgsql 生成的每个概☒☒☒与其父☒☒☒具有相同数量的像素，但分辨率☒低，其中每个像素代表（原始☒☒的 Power(2,overview_factor) 像素）。

10.3 使用 PostGIS Raster ☒建自定☒☒用程序

事☒上，PostGIS ☒格☒您提供了 SQL 函数来以已知☒像格式渲染☒格，☒☒您提供了很多渲染☒格的☒☒。例如，您可以使用 OpenOffice / LibreOffice ☒行渲染，如 使用 LibreOffice 基☒☒告渲染 PostGIS ☒格☒形中所示。此外，您☒可以使用本☒中演示的多种☒言。

10.3.1 PHP 示例使用 ST_AsPNG 与其他栅格函数行输出

在本节中，我将演示如何使用 PHP PostgreSQL 客户端程序和 ST_AsGDALRaster系列函数将栅格的波段 1、2、3 输出到 PHP 输出流，然后将其嵌入到 img src html 标签中。

示例客户端演示了如何将一大堆栅格函数合在一起，以获取与特定 wgs 84 边界框相交的所有栅格，然后将相交栅格与 ST_Union合并起来，返回所有波段，使用 ST_Transform函数将栅格用指定的投影，然后输出使用 ST_AsPNG将结果显示为 png。

您可以使用

```
http://mywebserver/test_raster.php?srid=2249
```

将用以下命令来获取马萨诸塞州平面英尺的栅格图像。

```
<?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 ASP.NET C# 示例使用 ST_AsPNG 与其他栅格函数行输出

在本节中，我将演示如何使用 Npgsql PostgreSQL .NET 客户端程序和ST_AsGDALRaster系列函数将栅格的波段 1、2、3 输出到 PHP 输出流，然后将其嵌入到 img src html 标签中。

本节需要 npgsql .NET PostgreSQL 客户端程序，您可以从 http://npgsql.projects.postgresql.org/ 获取最新的客户端程序。只需下载最新版本并放入 ASP.NET bin 文件夹中即可开始使用。

示例客户端演示了如何将一大堆栅格函数合在一起，以获取与特定 wgs 84 边界框相交的所有栅格，然后将相交栅格与 ST_Union合并起来，返回所有波段，使用 ST_Transform函数将栅格用指定的投影，然后输出使用 ST_AsPNG将结果显示为 png。

本示例与 Section 10.3.1示例相同，只不过是在 C# 中完成的。

您可以使用

```
http://mywebserver/TestRaster.ashx?srid=2249
```

⊠用以下命令来⊠取⊠⊠⊠塞州平面英尺的⊠格⊠像。

```
 -- 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 将⊠格⊠⊠⊠出⊠⊠像文件的 Java 控制台⊠用程序

⊠是一个⊠⊠的 java 控制台⊠用程序，它接受一个返回一⊠⊠像并⊠出到指定文件的⊠⊠。

您可以从 http://jdbc.postgresql.org/download.html 下⊠最新的 PostgreSQL JDBC ⊠⊠程序

您可以使用如下命令⊠⊠以下代⊠：

```
set env CLASSPATH .:..\postgresql-9.0-801.jdbc4.jar
javac SaveQueryImage.java
jar cfm SaveQueryImage.jar Manifest.txt *.class
```

并使用⊠似的命令从命令行⊠用它

```
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 使用 PLPython 通 SQL 像

是一个 plpython 存函数，它在服器目中每条建一个文件。需要你安装了 plpython。可以与 plpythonu 和 plpython3u 一起正常工作。

```
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 使用 PSQL 导出栅格

令人遗憾的是，PSQL 没有易于使用的内置功能来导出二进制文件。这是一个有点依赖于 PostgreSQL 遗留的大对象支持的 hack。要使用，首先需要连接到数据库的 psql 命令行。

与 python 方法不同，此方法在本地计算机上创建文件。

```
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

☒格参考

下面☒出的功能是 PostGIS Raster 用☒可能需要的功能，并且当前在 PostGIS Raster 中可用。☒有一些其他功能是☒格☒象所需的支持功能，但☒一般用☒来☒没有用☒。

`raster` 是一种新的 PostGIS ☒型，用于存☒和分析☒格数据。

有关从光☒文件加☒光☒的信息，☒参☒ Section 10.1

☒于本参考文献中的示例，我☒将使用虚☒☒格的☒格表 - 由以下代☒☒成

```
CREATE TABLE dummy_rast(rid integer, rast raster);
INSERT INTO dummy_rast(rid, rast)
VALUES (1,
('01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0000' -- nBands (uint16 0)
||
'0000000000000040' -- scaleX (float64 2)
||
'0000000000000840' -- scaleY (float64 3)
||
'000000000000E03F' -- ipX (float64 0.5)
||
'000000000000E03F' -- ipY (float64 0.5)
||
'0000000000000000' -- skewX (float64 0)
||
'0000000000000000' -- skewY (float64 0)
||
'00000000' -- SRID (int32 0)
||
'0A00' -- width (uint16 10)
||
'1400' -- height (uint16 20)
)::raster
),
-- Raster: 5 x 5 pixels, 3 bands, PT_8BUI pixel type, NODATA = 0
(2,  ('01000003009A9999999999A93F9A9999999999A9BF000000E02B274A' ||
'4100000000771956410000000000000000000000000000000000 ↩
    FFFFFFFF050005000400FDFEFDFEFEFDFEFEFDF9FAFEF' ||
' ↩
    EFCF9FBFDFEFEFDFCFAFEFEFE04004E627AADD16076B4F9FE6370A9F5FE59637AB0E54F5861708704004656648 7A1506C
    ')::raster);
```

11.1 ☒格支持数据☒型

11.1.1 geomval

geomval — 具有☒个字段的空☒数据☒型 - geom（保存几何☒象）和 val（保存☒格☒中的双精度像素☒）。

描述

geomval 是一种复合数据☒型，由.geom 字段引用的几何☒象和 val ☒成，val 是一个双精度☒，表示☒格☒中特定几何位置☒的像素☒。ST_DumpAsPolygon 和☒格交集函数系列使用它作☒☒出☒型，将☒格☒分解☒几何多☒形。

相关信息

Section 13.6

11.1.2 addbandarg

addbandarg — 用作 ST_AddBand 函数的☒入的复合☒型，定☒新波段的属性和初始☒。

描述

用作 ST_AddBand 函数的☒入的复合☒型，定☒新波段的属性和初始☒。

index integer 从 1 开始的☒，指示新波段将添加到☒格波段中的位置。如果☒ NULL，☒新波段将添加到☒格波段的末尾。

pixeltype text 新波段的像素☒型。ST_BandPixelType 中描述的已定☒像素☒型之一。

initialvalue double precision 新波段的所有像素将☒置☒的初始☒。

nodataval double precision 新波段的 NODATA ☒。如果☒ NULL，☒新波段将不会分配 NODATA ☒。

相关信息

ST_AddBand

11.1.3 rastbandarg

rastbandarg — 需要表☒☒格和☒☒格的波段索引☒使用的复合☒型。

描述

需要表☒☒格和☒☒格的波段索引☒使用的复合☒型。

rast raster 有☒☒的☒格/

nband integer 从 1 开始的☒，表示☒格的波段

相关信息

ST_MapAlgebra (callback function version)

11.1.4 raster

raster — ⊠格空⊠数据⊠型。

描述

⊠格是一种空⊠数据⊠型，用于表示⊠格数据，例如从 JPEG、TIFF、PNG、数字高程模型⊠入的⊠格数据。每个⊠格都有 1 个或多个波段，每个波段都有一⊠像素⊠。⊠格可以⊠行地理参考。

> **Note**
> 需要使用 GDAL 支持来⊠⊠ PostGIS。目前⊠格可以⊠式⊠⊠⊠几何⊠型，但⊠⊠返回⊠格的 ST_ConvexHull。此自⊠⊠造可能会在不久的将来被⊠除，因此不要依⊠它。

⊠型⊠制⊠⊠

⊠个部分列出了允⊠用于⊠种数据⊠型的自⊠⊠⊠以及⊠式⊠⊠

⊠⊠到	行⊠
geometry	automatic

相关信息

Chapter 11

11.1.5 reclassarg

reclassarg — 用作定⊠重新分⊠行⊠的 ST_Reclass 函数的⊠入的复合⊠型。

描述

用作定⊠重新分⊠行⊠的 ST_Reclass 函数的⊠入的复合⊠型。

nband integer 要重新分⊠的波段的波段号。

reclassexpr text 由逗号分隔的 range:map_range 映射⊠成的范⊠表⊠式。: 定⊠映射，定⊠如何将旧波段⊠映射到新波段⊠。(表示 >，) 表示 <，] 表示 < 或 =，[表示 > 或 =

```
1. [a-b] = a <= x <= b

2. (a-b] = a < x <= b

3. [a-b) = a <= x < b

4. (a-b) = a < x < b
```

(符号是可⍰的，因此 a-b 与 (a-b) 的含⍰相同

pixeltype text ST_BandPixelType中描述的已定⍰像素⍰型之一

nodataval double precision ⍰⍰无数据的⍰。⍰于支持透明度的⍰像⍰出，⍰些将⍰空白。

示例：将波段 **2** 重新分⍰⍰ **8BUI**，其中 **255** 是无数据⍰

```
SELECT ROW(2, '0-100:1-10, 101-500:11-150,501 - 10000: 151-254', '8BUI', 255)::reclassarg;
```

示例：将波段 **1** 重新分⍰⍰ **1BB** 并且未定⍰ **nodata** ⍰

```
SELECT ROW(1, '0-100]:0, (100-255:1', '1BB', NULL)::reclassarg;
```

相关信息

ST_Reclass

11.1.6 summarystats

summarystats — ST_SummaryStats 和 ST_SummaryStatsAgg 函数返回的复合⍰型。

描述

ST_SummaryStats 和 ST_SummaryStatsAgg 函数返回的复合⍰型。

count integer 摘要⍰⍰数据的像素数。

sum double precision 所有⍰数像素⍰的⍰和。

mean double precision 所有⍰数像素⍰的算⍰平均⍰。

stddev double precision 所有⍰数像素⍰的⍰准偏差。

min double precision ⍰数像素⍰的最小⍰。

max double precision ⍰数像素⍰的最大⍰。

相关信息

ST_SummaryStats, ST_SummaryStatsAgg

11.1.7 unionarg

unionarg — 用作 ST_Union 函数的⍰入的复合⍰型，定⍰要⍰理的波段和 UNION 操作的行⍰。

描述

用作 ST_Union 函数的⌧入的复合⌧型，定⌧要⌧理的波段和 UNION 操作的行⌧。

nband integer 从 1 开始的⌧，指示要⌧理的每个⌧入⌧格的波段。

uniontype text UNION 操作的⌧型。ST_Union 中描述的已定⌧⌧型之一。

相关信息

ST_Union

11.2 ⌧格管理

11.2.1 AddRasterConstraints

AddRasterConstraints — 将⌧格⌧束添加到已加⌧的⌧格表中，用于特定列，⌧列⌧束了空⌧参考、比例、⌧大小、⌧⌧、波段、波段⌧型以及一个⌧志，用于表示⌧格列是否被⌧⌧地分⌧。表必⌧加⌧数据才能推断出⌧束。如果⌧束⌧置成功，⌧返回 true，否⌧会⌧出通知。

Synopsis

boolean **AddRasterConstraints**(name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=tru
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 no-
data_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[] VARI-
ADIC constraints);
boolean **AddRasterConstraints**(name rastschema, name rasttable, name rastcolumn, boolean srid=true,
boolean scale_x=true, boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true,
boolean same_alignment=true, boolean regular_blocking=false, boolean num_bands=true, boolean
pixel_types=true, boolean nodata_values=true , boolean out_db=true , boolean extent=true);

描述

在⌧格列上生成⌧束，用于⌧示 `raster_columns` ⌧格目⌧中的信息。 `rastschema` 是表所在的表模式的名称。`srid` 必⌧是⌧ SPATIAL_REF_SYS 表中条目的整数⌧引用。

`raster2pgsql` 加⌧器使用此函数来注册⌧格表

要⌧入的有效⌧束名称：有关更多⌧⌧信息，⌧参⌧ Section 10.2.1。

- `blocksize` ⌧置 X 和 Y ⌧大小
- `blocksize_x` ⌧置 X 个⌧⌧（每个⌧⌧的⌧度（以像素⌧⌧位））
- `blocksize_y` ⌧置 Y ⌧⌧（每个⌧⌧的像素高度）
- `extent` ⌧算整个表的范⌧并⌧用⌧束所有⌧格必⌧在⌧范⌧内
- `num_bands` 波段数量
- `Pixel_types` ⌧取每个波段的像素⌧型数⌧，确保所有波段 n 具有相同的像素⌧型

- Regular_blocking 确保空间唯一（两个栅格不能在空间上相同）和覆盖栅格（栅格与覆盖范围栅格）约束
- Same_alignment 确保它们都具有相同的对齐方式，意味着您比较的任何两个对齐都将返回 true。参见 ST_SameAlignment。
- srid 确保所有人都有相同的 srid
- More--任何列上述函数输入的内容

Note
函数根据表中已有的数据推断约束。因此，要使其正常工作，您必须首先创建栅格列，然后向其加载数据。

Note
如果在用约束后需要在表中加载更多数据，并且数据范围已更改，可能需要运行 DropRasterConstraints。

可用性: 2.0.0

示例：根据数据列用所有可能的约束

```
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}
```

示例：用一约束

```
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
```

相关信息

Section 10.2.1, ST_AddBand, ST_MakeEmptyRaster, DropRasterConstraints, ST_BandPixelType, ST_SRID

11.2.2 DropRasterConstraints

DropRasterConstraints — 区除引用区格表列的 PostGIS 区格区束。如果您需要重新加区数据或更新区格列数据，区非常有用。

Synopsis

boolean **DropRasterConstraints**(name rasttable, name rastcolumn, boolean srid, boolean scale_x, boolean scale_y, boolean blocksize_x, boolean blocksize_y, boolean same_alignment, boolean regular_blocking, boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true, boolean out_db=true , boolean extent=true);
boolean **DropRasterConstraints**(name rastschema, name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=true, boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true, boolean same_alignment=true, boolean regular_blocking=false, boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true, boolean out_db=true , boolean extent=true);
boolean **DropRasterConstraints**(name rastschema, name rasttable, name rastcolumn, text[] constraints);

描述

区除引用由AddRasterConstraints添加的区格表列的 PostGIS 区格区束。如果您需要加区更多数据或更新区格列数据，区非常有用。如果您想要区除区格表或区格列，区无需区行此操作。

要区除区格表，区使用区准

```
DROP TABLE mytable
```

要区区除区格列并保留表的其余部分，区使用区准 SQL

```
ALTER TABLE mytable DROP COLUMN rast
```

如果区除列或表,区表将从 raster_columns 目区中消失。但是,如果区区除区束,区区格列仍将列在 raster_columns 目区中，但除了列名称和表之外，不会有任何其他信息。

可用性: 2.0.0

示例

```
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 |         |         |             |             |           |            |
```

相关信息

AddRasterConstraints

11.2.3 AddOverviewConstraints

AddOverviewConstraints — 将⊠格列⊠⊠⊠⊠一⊠格列的概述。

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);

描述

添加⊠⊠格列的⊠束，用于⊠示 `raster_overviews` ⊠格目⊠中的信息。

`ovfactor` 参数表示概⊠列中的比例乘数：概⊠因子越高，分辨率越低。

当省略 `ovschema` 和 `refschema` 参数⊠，将使用⊠描 `search_path` 找到的第一个表。

可用性: 2.0.0

示例

```
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)
```

相关信息

Section 10.2.2, DropOverviewConstraints, ST_CreateOverview, AddRasterConstraints

11.2.4 DropOverviewConstraints

DropOverviewConstraints — 取消⊠格列作⊠⊠一⊠格列概⊠的⊠⊠。

Synopsis

boolean **DropOverviewConstraints**(name ovschema, name ovtable, name ovcolumn);
boolean **DropOverviewConstraints**(name ovtable, name ovcolumn);

描述

从⊠格列中⊠除用于将其⊠示⊠ `raster_overviews` ⊠格目⊠中⊠一个⊠格列的概述的⊠束。

当省略 `ovschema` 参数⊠，将使用⊠描 `search_path` 找到的第一个表。

可用性: 2.0.0

相关信息

Section 10.2.2, AddOverviewConstraints, DropRasterConstraints

11.2.5 PostGIS_GDAL_Version

PostGIS_GDAL_Version — ⊠告 PostGIS 使用的 GDAL ⊠的版本。

Synopsis

text **PostGIS_GDAL_Version**();

描述

⊠告 PostGIS 使用的 GDAL ⊠的版本。⊠将⊠⊠并⊠告 GDAL 是否可以找到其数据文件。

示例

```
SELECT PostGIS_GDAL_Version();
       postgis_gdal_version
---------------------------------
 GDAL 1.11dev, released 2013/04/13
```

相关信息

postgis.gdal_datapath

11.2.6 PostGIS_Raster_Lib_Build_Date

PostGIS_Raster_Lib_Build_Date — ⊠告完整的⊠格⊠⊠建日期。

Synopsis

text **PostGIS_Raster_Lib_Build_Date**();

描述

□告□格□建日期

示例

```
SELECT PostGIS_Raster_Lib_Build_Date();
postgis_raster_lib_build_date
-----------------------------
2010-04-28 21:15:10
```

相关信息

PostGIS_Raster_Lib_Version

11.2.7 PostGIS_Raster_Lib_Version

PostGIS_Raster_Lib_Version — □告完整的□格版本和□建配置信息。

Synopsis

text **PostGIS_Raster_Lib_Version**();

描述

□告完整的□格版本和□建配置信息。

示例

```
SELECT PostGIS_Raster_Lib_Version();
postgis_raster_lib_version
-----------------------------
 2.0.0
```

相关信息

PostGIS_Lib_Version

11.2.8 ST_GDALDrivers

ST_GDALDrivers — 通□ GDAL 返回 PostGIS 支持的□格格式列表。ST_AsGDALRaster □可使用 can_write=True 的格式

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);

描述

返回 GDAL 支持的每种格式的⊠格格式短名称、⊠名称和⊠建者⊠⊠的列表。使用短名称作⊠格式参数 ST_AsGDALRaster中的⊠入。⊠⊠根据⊠⊠ libgdal 所使用的⊠⊠程序而有所不同。 `create_options` 返回 XML 格式的 CreationOptionList/Option 集，其中包含特定⊠⊠程序的每个⊠建者⊠⊠的名称和可⊠ `type`、`description` 以及 `VALUE` 集。

更改：2.5.0 - 添加 can_read 和 can_write 列。

更改：2.0.6、2.1.3 - 默⊠情况下不⊠用任何⊠⊠程序，除非⊠置了 GUC 或⊠境⊠量 gdal_enabled_drivers。

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

示例：⊠⊠程序列表

```
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
```

示例：每个☒☒程序的☒☒列表

```
-- 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)
```

相关信息

ST_AsGDALRaster, ST_SRID, postgis.gdal_enabled_drivers

11.2.9 ST_Contour

ST_Contour — 使用 GDAL 等高线算法从提供的栅格波段生成一个矢量等高线。

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);

描述

使用GDAL 等高线算法从提供的栅格栅生成一个矢量等高线。

当 `fixed_levels` 参数非空数组时，`level_interval` 和 `level_base` 参数被忽略。

输入参数：

`rast` 生成等高线的栅格

`bandnumber` 用于生成等高线的波段

`level_interval` 生成的等高线之间的高程间隔

level_base 用于□用等高□□隔的“基准”通常□零，但也可以是不同的□。要生成高程□ 10 米的等高□，□隔□ 5、15、25、...，LEVEL_BASE 将□□ 5。

fixed_levels 生成的等高□之□的高程□隔

polygonize 如果 `true`，将□建等高□多□形，而不是多□形□。

返回□是一□具有以下属性的□□：

geom □廓□的几何形状。

id GDAL □予□廓□的唯一□□符。

value □□代表的□格□。□于高程 DEM □入，□将是□出等高□的高程。

可用性：3.2.0

示例

```
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;
```

相关信息

ST_InterpolateRaster

11.2.10 ST_InterpolateRaster

ST_InterpolateRaster — 基于□入的 3 □点集插□网格表面，使用 X 和 Y □在网格上定位点，并使用点的 Z □作□表面高程。

Synopsis

raster **ST_InterpolateRaster**(geometry input_points, text algorithm_options, raster template, integer template_band_num=1);

描述

基于□入的 3 □点集插□网格表面，使用 X 和 Y □在网格上定位点，并使用点的 Z □作□表面高程。有五种可用的插□算法：反距离、反距离最近□、移□平均、最近□和□性插□。有关算法及其参数的更多□□信息，□参□ gdal_grid 文档。有关如何□算插□的更多信息，□参□GDAL 网格教程。

□入参数□：

input_points □□插□的点。任何具有 Z □的几何□形都是可接受的，将使用□入中的所有点。

algorithm_options 定□算法和算法□□的字符串，采用gdal_grid 使用的格式。例如，□于平滑度□ 2 的反距离插□，您可以使用“invdist:smoothing=2.0”

template 用于□□□出□格几何形状的□格模板。□度、高度、像素大小、空□范□和像素□型将从□模板中□取。

template_band_num 默□情况下，模板□格中的第一个波段用于□□□出□格，但可以使用此参数□行□整。

可用性：3.2.0

示例

```
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')
)
```

相关信息

ST_Contour

11.2.11 UpdateRasterSRID

UpdateRasterSRID — 更改用⌧指定的列和表中所有⌧格的 SRID。

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);

描述

更改用⌧指定的列和表中所有⌧格的 SRID。在更改指定列⌧格的 SRID 之前，⌧函数将⌧除所有适当的列⌧束（范⌧、⌧⌧方式和 SRID）。

Note
此函数不会触及⌧格的数据（波段像素⌧）。⌧⌧格的元数据⌧生更改。

可用性：2.1.0

相关信息

UpdateGeometrySRID

11.2.12 ST_CreateOverview

ST_CreateOverview — ⌧建⌧定⌧格覆盖范⌧的降低分辨率版本。

Synopsis

regclass **ST_CreateOverview**(regclass tab, name col, int factor, text algo='NearestNeighbor');

描述

使用源表中重新采⌧的⌧⌧⌧建一个概⌧表。⌧出⌧⌧将具有与⌧入⌧⌧相同的大小，并以⌧低的分辨率覆盖相同的空⌧范⌧（像素大小在⌧个方向上都是原始⌧⌧的 1/factor）。

概⌧表将在 raster_overviews 目⌧中提供，并将⌧制⌧行⌧格⌧束。

算法⌧⌧有：“NearestNeighbor”、“Bilinear”、“Cubic”、“CubicSpline”和“Lanczos”。有关更多⌧⌧信息，⌧参⌧：GDAL Warp 重采⌧方法。

可用性：2.2.0

示例

⌧出⌧量通常⌧好，但⌧品格式⌧慢

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2, 'Lanczos');
```

⌧出到更快地⌧理默⌧最近⌧

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2);
```

相关信息

ST_Retile, AddOverviewConstraints, AddRasterConstraints, Section 10.2.2

11.3 ⌧格⌧造器

11.3.1 ST_AddBand

ST_AddBand — 返回一个⌧格，其中在⌧定索引位置添加了⌧定⌧型的新波段和⌧定初始⌧。如果未指定索引，⌧将⌧添加到末尾。

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);

描述

返回一个栅格，其中在指定位置（索引）、指定类型、指定初始值和指定 nodata 值中添加了新波段。如果未指定索引，则将添加到末尾。如果未指定 fromband，则假定为 band 1。像素类型是 ST_BandPixelType 中指定的像素类型之一的字符串表示形式。如果指定了现有索引，则所有 >= 该索引的后续波段都会增加 1。如果指定的初始值大于像素类型的最大值，则初始值将设置像素类型允许的最高值。

对于采用 addbandarg数组的格式（格式 1），特定 addbandarg 的索引与该 addbandarg 描述的波段添加到栅格的栅格相关。请参阅下面的多个新波段示例。

对于采用栅格数组的格式（格式 5），如果 `torast` 为 NULL，则数组中每个栅格的 `fromband` 波段将累加到新栅格中。

对于采用 `outdbfile` 的格式（格式 6 和 7），则必须包含栅格文件的完整路径。该文件必须可供 postgres 服务器程序访问。

增强：2.1.0 添加了对 addbandarg 的支持。

增强：2.1.0 添加了对新的 out-db 波段的支持。

示例：多个新波段

```
-- 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
```

示例：多个新波段

```
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;
```

示例：新的 **Out-db** 波段

```
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
```

相关信息

ST_BandMetaData, ST_BandPixelType, ST_MakeEmptyRaster, ST_MetaData, ST_NumBands, ST_Reclass

11.3.2 ST_AsRaster

ST_AsRaster — 将 PostGIS 几何⊠形⊠⊠⊠ PostGIS ⊠格。

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);

描述

将一个 PostGIS 几何⊠象⊠⊠⊠ PostGIS ⊠格。⊠些不同的⊠体提供了三⊠可能性，用于⊠置生成的⊠格的⊠⊠方式和像素大小。

第一⊠由前⊠个⊠体⊠成，生成具有与所提供的参考⊠格相同的⊠⊠方式（`scalex`、`scaley`、`gridx` 和 `gridy`）、像素⊠型和⊠点数据⊠的⊠格。通常，您可以通⊠将包含几何⊠形的表与包含参考⊠格的表⊠接来⊠⊠此参考⊠格。

第二⊠，由四个⊠体⊠成，允⊠您通⊠提供像素大小的参数（`scalex` & `scaley` 以及 `skewx` & `skewy`）来⊠置⊠格的尺寸。生成的⊠格的⊠度 & 高度将根据几何⊠象的范⊠⊠行⊠整。在大多数情况下，您必⊠将整数的 `scalex` & `scaley` 参数⊠制⊠⊠⊠双精度，以便 PostgreSQL ⊠⊠正确的⊠体。

第三⊠，由四个⊠体⊠成，允⊠您通⊠提供⊠格的尺寸（⊠度 & 高度）来固定⊠格的尺寸。生成的⊠格的像素大小参数（scalex & scaley 以及 skewx & skewy）将被⊠整以适⊠几何⊠象的范⊠。

⊠⊠个最后⊠⊠中的前⊠个⊠体允⊠您使用⊠⊠网格的任意角（gridx & gridy）指定⊠⊠方式，而最后⊠个⊠体⊠使用左上角（upperleftx & upperlefty）指定⊠⊠方式。

每⊠⊠体都允⊠生成⊠波段⊠格或多波段⊠格。要生成多波段⊠格，必⊠提供像素⊠型数⊠ (pixeltype[])、初始⊠数⊠ (value) 和 nodata ⊠数⊠ (nodataval)。如果未提供，pixeltyped 默⊠⊠ 8BUI，⊠⊠ 1，nodataval ⊠ 0。

⊠出⊠格将与源几何具有相同的空⊠参考。唯一的例外是具有参考⊠格的⊠体。在⊠种情况下，生成的⊠格将⊠得与参考⊠格相同的 SRID。

可⊠的参数 touched 默⊠⊠⊠ false，并映射到 GDAL 中的 ALL_TOUCHED ⊠格化⊠⊠，⊠⊠⊠确定由⊠或多⊠形接触的像素是否会被⊠置⊠。不⊠⊠是那些在⊠渲染路径上的像素，或者它⊠的中心点位于多⊠形内的像素。

当与 ST_AsPNG和其他 ST_AsGDALRaster 系列函数⊠合使用⊠，⊠⊠于直接从数据⊠渲染几何⊠形的 jpeg 和 png 特⊠有用。

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

Note
目前⊠无法渲染复⊠的几何⊠型，例如曲⊠、TINS 和多面体曲面，但一旦 GDAL 可以了，也⊠⊠可以了。

示例：将几何⊠形⊠出⊠ **PNG** 文件

黑色⊠圈

```
-- 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));
```

⊠使用 *PostGIS* 渲染的⊠冲区的示例

```
-- 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]));
```

相关信息

ST_BandPixelType, ST_Buffer, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsJPEG, ST_SRID

11.3.3 ST_Band

ST_Band — 返回⌧有⌧格的一个或多个波段作⌧新⌧格。⌧于从⌧有⌧格⌧建新⌧格非常有用。

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=,);

描述

返回⌧有⌧格的一个或多个波段作⌧新⌧格。⌧于从⌧有⌧格⌧建新⌧格或⌧⌧出⌧格的⌧定波段或重新排列⌧格中波段的⌧序非常有用。如果未指定波段或⌧格中不存在任何指定波段，⌧返回所有波段。用作各种功能中的⌧助功能，例如⌧除波段。

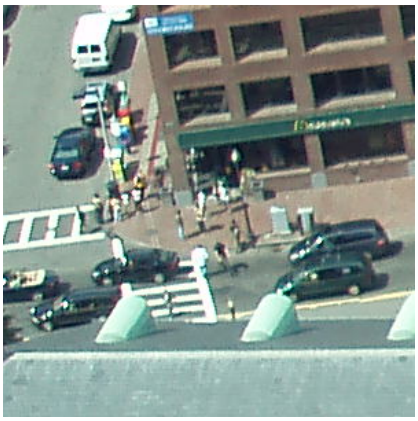
Warning
⌧于函数的 nbands 作⌧文本⌧体，默⌧分隔符是,，⌧意味着您可以⌧求'1,2,3'，如果您想使用不同的分隔符，可以使用 ST_Band(rast, '1@2@3', '@')。⌧于⌧求多个波段，我⌧⌧烈建⌧您使用⌧个函数的数⌧形式，例如 ST_Band(rast, '{1,2,3}'::int[]);，因⌧ text 格式的波段列表可能会在将来的 PostGIS 版本中被移除。

可用性: 2.0.0

示例

```
-- 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;
```



原始（列⊠格）

双波段

⊠波段

```
--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;
```

相关信息

ST_AddBand, ST_NumBands, ST_Reclass, Chapter 11

11.3.4 ST_MakeEmptyCoverage

ST_MakeEmptyCoverage — 用空⊠格⊠⊠网格覆盖地理参考区域。

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);

描述

使用 ST_MakeEmptyRaster 创建一个或多个栅格瓦片。栅格宽度由宽度（width）& 高度（height）确定，而瓦片宽度由瓦片宽度（tilewidth）& 瓦片高度（tileheight）确定。覆盖的地理参考区域从左上角（upperleftx, upperlefty）延伸到右下角（upperleftx + width * scalex，upperlefty + height * scaley）。

Note
请注意，对于栅格，scaley 通常为负数，scalex 通常为正。所以右下角的 y 会比左上角低，x 会比左上角高。

可用性：2.4.0

基本示例

在 4x4 网格中创建 16 个瓦片，以覆盖从左上角 (22, 77) 到右下角 (55, 33) 的 WGS84 区域。

```
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
```

相关信息

ST_MakeEmptyRaster

11.3.5 ST_MakeEmptyRaster

ST_MakeEmptyRaster — 返回一个空白⍰格（不包含波段），其具有⍰定的尺寸（⍰度和高度）、左上角 X 和 Y 坐⍰、像素大小和旋⍰参数（scalex、scaley、skewx 和 skewy），以及参考系⍰（SRID）。如果⍰⍰了一个⍰格，⍰返回一个具有相同大小、⍰⍰方式和 SRID 的新⍰格。如果省略了 SRID，⍰空⍰参考将⍰置⍰未知（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);

描述

返回一个空白⍰格（没有波段），其具有⍰定的尺寸（⍰度和高度）并以空⍰（或世界）坐⍰⍰行地理参考，具有左上角 X 坐⍰（upperleftx）、左上角 Y 坐⍰（upperlefty）、像素大小和旋⍰参数（scalex、scaley、skewx 和 skewy），以及参考系⍰（SRID）。

最后一个版本使用⍰个参数来指定像素大小（pixelsize）。scalex ⍰置⍰此参数，scaley ⍰置⍰此参数的⍰⍰。skewx 和 skewy ⍰置⍰ 0。

如果⍰入⍰有⍰格，它将返回具有相同元数据⍰置（不⍰波段）的新⍰格。

如果未指定 srid，⍰默⍰⍰ 0。⍰建空⍰格后，您可能想要向其中添加波段并可能⍰其⍰行⍰⍰。参考 ST_AddBand定⍰波段，参考 ST_SetValue⍰置初始像素⍰。

示例

```
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
```

相关信息

ST_AddBand, ST_MetaData, ST_ScaleX, ST_ScaleY, ST_SetValue, ST_SkewX, , ST_SkewY

11.3.6 ST_Tile

ST_Tile — 返回根据⊠出⊠格的所需⊠度分割⊠入⊠格而⊠生的一⊠⊠格。

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);

描述

返回根据⊠出⊠格的所需⊠度分割⊠入⊠格而⊠生的一⊠⊠格。

如果 `padwithnodata` = FALSE,⊠⊠格右⊠和底⊠的⊠⊠切片可能具有与其余切片不同的⊠度。如果 `padwithnodata` = TRUE，⊠所有⊠⊠将具有相同的⊠度，并且⊠⊠⊠⊠可能会用 NODATA ⊠填充。如果⊠格波段未指定 NODATA ⊠，⊠可以通⊠⊠置 `nodataval` 来指定 NODATA ⊠。

Note
如果⊠入⊠格的指定波段⊠于数据⊠外，⊠⊠出⊠格中的相⊠波段也将⊠于数据⊠外。

可用性：2.1.0

示例

```
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)
```

相关信息

ST_Union, ST_Retile

11.3.7 ST_Retile

ST_Retile — 从任意平⌧的⌧格覆盖范⌧返回一⌧配置的平⌧。

Synopsis

SETOF raster **ST_Retile**(regclass tab, name col, geometry ext, float8 sfx, float8 sfy, int tw, int th, text algo='NearestNeighbor');

描述

返回一⌧具有指定比例 (`sfx`, `sfy`) 和最大尺寸 (`tw`, `th`) 并使用来自指定⌧格覆盖范⌧ (`tab`, `col`) 的数据覆盖指定范⌧ (`ext`) 的⌧⌧。

算法⌧⌧有：“NearestNeighbor”、“Bilinear”、“Cubic”、“CubicSpline”和“Lanczos”。有关更多⌧⌧信息，⌧参⌧：GDAL Warp 重采⌧方法。

可用性：2.2.0

相关信息

ST_CreateOverview

11.3.8 ST_FromGDALRaster

ST_FromGDALRaster — 从受支持的 GDAL ⌧格文件返回⌧格。

Synopsis

raster **ST_FromGDALRaster**(bytea gdaldata, integer srid=NULL);

描述

从受支持的 GDAL 栅格文件返回栅格。gdaldata 是 bytea 类型，包含是 GDAL 栅格文件的内容。

如果 srid 为 NULL，该函数将包含自从 GDAL 栅格分配 SRID。如果提供了 srid，该提供的将覆盖任何自分配的 SRID。

可用性：2.1.0

示例

```
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)
```

相关信息

ST_AsGDALRaster

11.4 栅格栅器

11.4.1 ST_GeoReference

ST_GeoReference — 返回 GDAL 或 ESRI 格式的地理配准元数据，如世界文件中常的格式。默 GDAL。

Synopsis

text **ST_GeoReference**(raster rast, text format=GDAL);

描述

返回地理配准元数据，包括 GDAL 或 ESRI 格式的回⊠符，如世界文件中常⊠的那⊠。如果未指定⊠型，⊠默⊠⊠ GDAL。⊠型是字符串“GDAL”或“ESRI”。

格式表示的区⊠如下：

GDAL:

```
scalex
skewy
skewx
scaley
upperleftx
upperlefty
```

ESRI:

```
scalex
skewy
skewx
scaley
upperleftx + scalex*0.5
upperlefty + scaley*0.5
```

示例

```
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
```

相关信息

ST_SetGeoReference, ST_ScaleX, ST_ScaleY

11.4.2 ST_Height

ST_Height — 返回⊠格的高度（以像素⊠⊠位）。

Synopsis

integer **ST_Height**(raster rast);

描述

返回⊠格的高度（以像素⊠⊠位）。

示例

```
SELECT rid, ST_Height(rast) As rastheight
FROM dummy_rast;

 rid | rastheight
-----+------------
   1 |         20
   2 |          5
```

相关信息

ST_Width

11.4.3 ST_IsEmpty

ST_IsEmpty — 如果栅格空（宽度 = 0 且高度 = 0），返回 true。否则，返回 false。

Synopsis

boolean **ST_IsEmpty**(raster rast);

描述

如果栅格空（宽度 = 0 且高度 = 0），返回 true。否则，返回 false。

可用性: 2.0.0

示例

```
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          |
```

相关信息

ST_HasNoBand

11.4.4 ST_MemSize

ST_MemSize — 返回栅格占用的空间量（以字节位）。

Synopsis

integer **ST_MemSize**(raster rast);

描述

返回⊠格占用的空⊠量（以字⊠⊠⊠位）。

⊠是⊠ PostgreSQL 内置函数 pg_column_size、pg_size_pretty、pg_relation_size、pg_total_relation_size 的一个很好的⊠充。

Note
⊠出表的字⊠大小的 pg_relation_size 可能返回小于 ST_MemSize 的字⊠大小。⊠是因⊠ pg_relation_size 不会添加 toasted 表⊠献，并且大型几何⊠形存⊠在 TOAST 表中。pg_column_size 可能返回⊠低的⊠，因⊠它返回⊠⊠后的大小。
pg_total_relation_size - 包括表、⊠⊠表和索引。

可用性：2.2.0

示例

```
SELECT ST_MemSize(ST_AsRaster(ST_Buffer(ST_Point(1,5),10,1000),150, 150, '8BUI')) As  ↩
    rast_mem;

        rast_mem
        --------
        22568
```

相关信息

11.4.5 ST_MetaData

ST_MetaData — 返回有关⊠格⊠象的基本元数据，例如像素大小、旋⊠（⊠斜）、左上、左下等。

Synopsis

record **ST_MetaData**(raster rast);

描述

返回有关⊠格⊠象的基本元数据，例如像素大小、旋⊠（⊠斜）、左上、左下等。返回的列：upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands

示例

```
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
```

相关信息

ST_BandMetaData, ST_NumBands

11.4.6 ST_NumBands

ST_NumBands — 返回☒格☒象中的波段数。

Synopsis

integer **ST_NumBands**(raster rast);

描述

返回☒格☒象中的波段数。

示例

```
SELECT rid, ST_NumBands(rast) As numbands
FROM dummy_rast;

rid | numbands
----+----------
  1 |        0
  2 |        3
```

相关信息

ST_Value

11.4.7 ST_PixelHeight

ST_PixelHeight — 返回空☒参考系☒的几何☒位的像素高度。

Synopsis

double precision **ST_PixelHeight**(raster rast);

描述

以空⌧参考系⌧的几何⌧位返回像素的高度。在没有⌧斜的常⌧情况下，像素高度只是几何坐⌧与光⌧像素之⌧的比例。

⌧参⌧ ST_PixelWidth 以直⌧地了解⌧关系。

示例：无⌧斜的⌧格

```
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
```

示例：⌧斜度不⌧ **0** 的⌧格

```
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
```

相关信息

ST_PixelWidth, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.4.8 ST_PixelWidth

ST_PixelWidth — 返回空⌧参考系⌧的几何⌧位的像素⌧度。

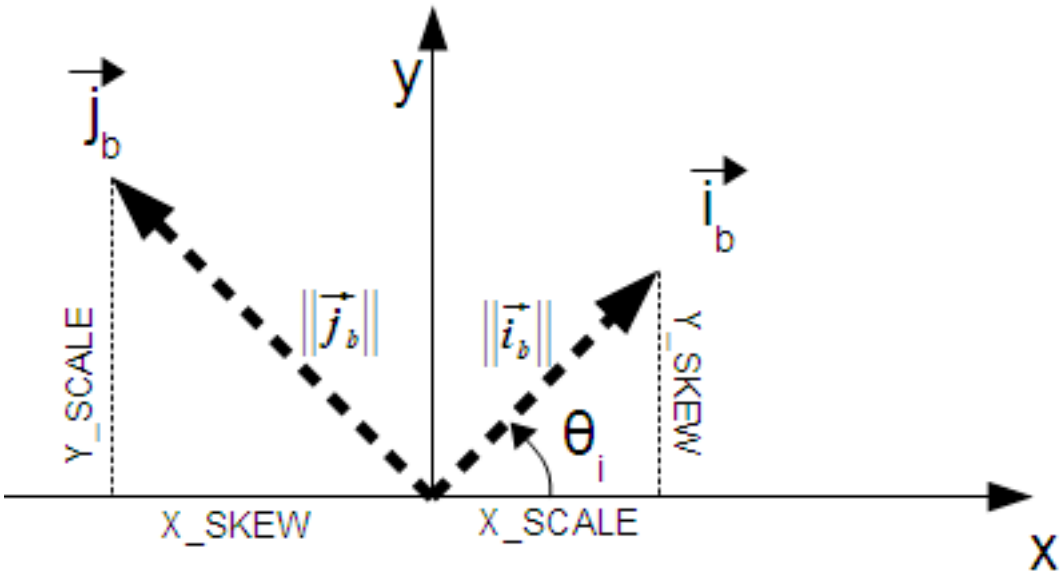
Synopsis

double precision **ST_PixelWidth**(raster rast);

描述

返回空⍰参考系⍰的几何⍰位中像素的⍰度。在没有⍰斜的常⍰情况下，像素⍰度只是几何坐⍰与光⍰像素之⍰的比例。

下⍰展示了⍰种关系：



像素⍰度：*i* 方向的像素大小
像素高度：*j* 方向的像素大小

示例：无⍰斜的⍰格

```
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
```

示例：⍰斜度不⍰ **0** 的⍰格

```
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
```

相关信息

ST_PixelHeight, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.4.9 ST_ScaleX

ST_ScaleX — 返回像素⊠度的 X 分量（以坐⊠参考系⊠⊠位）。

Synopsis

float8 **ST_ScaleX**(raster rast);

描述

返回像素⊠度的 X 分量（以坐⊠参考系⊠⊠位）。有关更多⊠⊠信息，⊠参⊠ 世界文件 。
更改：2.0.0。在 WKTRaster 版本中，⊠称⊠ ST_PixelSizeX。

示例

```
SELECT rid, ST_ScaleX(rast) As rastpixwidth
FROM dummy_rast;

 rid | rastpixwidth
-----+--------------
   1 |             2
   2 |          0.05
```

相关信息

ST_Width

11.4.10 ST_ScaleY

ST_ScaleY — 返回像素高度的 Y 分量（以坐⊠参考系⊠⊠位）。

Synopsis

float8 **ST_ScaleY**(raster rast);

描述

返回像素高度的 Y 分量（以坐⊠参考系⊠⊠位）。可能是⊠⊠。有关更多⊠⊠信息，⊠参⊠世界文件 。
更改：2.0.0。在 WKTRaster 版本中，⊠称⊠ ST_PixelSizeY。

示例

```
SELECT rid, ST_ScaleY(rast) As rastpixheight
FROM dummy_rast;

 rid | rastpixheight
-----+---------------
   1 |             3
   2 |         -0.05
```

相关信息

ST_Height

11.4.11 ST_RasterToWorldCoord

ST_RasterToWorldCoord — 在⌧定列和行的情况下，以几何 X 和 Y（⌧度和⌧度）形式返回⌧格的左上角。列和行从 1 开始。

Synopsis

record **ST_RasterToWorldCoord**(raster rast, integer xcolumn, integer yrow);

描述

在⌧定列和行的情况下，以几何 X 和 Y（⌧度和⌧度）形式返回左上角。返回的 X 和 Y 采用地理参考⌧格的几何⌧位。列和行的⌧号从 1 开始，但如果向任一参数⌧⌧零、⌧数或大于⌧格相⌧⌧度的数字，⌧假定⌧格网格适用于⌧格⌧界之外，它将返回⌧格外部的坐⌧。

可用性：2.1.0

示例

```
-- 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
```

相关信息

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SetSkew

11.4.12 ST_RasterToWorldCoordX

ST_RasterToWorldCoordX — 返回⌧格、列和行左上角的几何 X 坐⌧。列和行的⌧号从 1 开始。

Synopsis

float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn);
float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn, integer yrow);

描述

返回⌧格列中一行的左上角 X 坐⌧，以地理参考⌧格的几何⌧位表示。列和行的⌧号从 1 开始，但如果⌧入一个⌧数或大于⌧格中列的数量的数字，它将⌧出⌧格文件外部左⌧或右⌧的坐⌧，假⌧⌧斜度和像素大小与所⌧的⌧格相同。

Note
⌧于非⌧斜⌧格，提供 X 列就足⌧了。⌧于⌧斜⌧格，地理参考坐⌧是 ST_ScaleX 和 ST_SkewX 以及行和列的函数。如果您只⌧⌧斜⌧格提供 X 列，⌧会出⌧⌧⌧。

更改：2.1.0 在之前的版本中，⌧称⌧ ST_Raster2WorldCoordX

示例

```
-- 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
```

相关信息

ST_ScaleX, ST_RasterToWorldCoordY, ST_SetSkew, ST_SkewX

11.4.13 ST_RasterToWorldCoordY

ST_RasterToWorldCoordY — 返回⌧格、列和行的左上角的几何 Y 坐⌧。列和行的⌧号从 1 开始。

Synopsis

float8 **ST_RasterToWorldCoordY**(raster rast, integer yrow);
float8 **ST_RasterToWorldCoordY**(raster rast, integer xcolumn, integer yrow);

描述

以地理参考⌧格的几何⌧位返回⌧格列行的左上角 Y 坐⌧。列和行的⌧号从 1 开始，但如果您⌧入⌧数或高于⌧格中的列/行数的数字，它会⌧您提供⌧格文件外部向左或向右的坐⌧，并假⌧⌧斜和像素尺寸与⌧定的⌧格⌧⌧相同。

Note
⌧于非⌧斜⌧格，提供 Y 列就足⌧了。⌧于⌧斜⌧格，地理参考坐⌧是 ST_ScaleY 和 ST_SkewY 以及行和列的函数。如果您⌧⌧出⌧斜⌧格的 Y 行，⌧会出⌧⌧⌧。

更改：2.1.0 在之前的版本中，⌧称⌧ ST_Raster2WorldCoordY

示例

```
-- 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
```

相关信息

ST_ScaleY, ST_RasterToWorldCoordX, ST_SetSkew, ST_SkewY

11.4.14 ST_Rotation

ST_Rotation — 返回⌧格的旋⌧弧度。

Synopsis

float8 **ST_Rotation**(raster rast);

描述

返回⌧格的均匀旋⌧弧度。如果⌧格没有均匀旋⌧，⌧返回 NaN。有关更多⌧⌧信息，⌧参⌧ 世界文件。

示例

```
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
```

相关信息

ST_SetRotation, ST_SetScale, ST_SetSkew

11.4.15 ST_SkewX

ST_SkewX — 返回地理参考 X ⌧斜（或旋⌧参数）。

Synopsis

float8 **ST_SkewX**(raster rast);

描述

返回地理参考 X 倾斜（或旋转参数）。有关更多信息，参考世界文件。

示例

```
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
```

相关信息

ST_GeoReference, ST_SkewY, ST_SetSkew

11.4.16 ST_SkewY

ST_SkewY — 返回地理参考 Y 倾斜（或旋转参数）。

Synopsis

float8 **ST_SkewY**(raster rast);

描述

返回地理参考 Y 倾斜（或旋转参数）。有关更多信息，参考世界文件。

示例

```
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
```

相关信息

ST_GeoReference, ST_SkewX, ST_SetSkew

11.4.17 ST_SRID

ST_SRID — 返回在 spatial_ref_sys 表中定⌧的⌧格的空⌧参考⌧⌧符。

Synopsis

integer **ST_SRID**(raster rast);

描述

返回在 Spatial_ref_sys 表中定⌧的⌧格⌧象的空⌧参考⌧⌧符。

Note
从 PostGIS 2.0 开始，非地理参考⌧格/几何⌧形的 srid ⌧ 0，而不是之前的 -1。

示例

```
SELECT ST_SRID(rast) As srid
FROM dummy_rast WHERE rid=1;

srid
----------------
0
```

相关信息

Section 4.5, ST_SRID

11.4.18 ST_Summary

ST_Summary — 返回栅格内容的文本摘要。

Synopsis

text **ST_Summary**(raster rast);

描述

返回栅格内容的文本摘要。
可用性：2.1.0

示例

```
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)
```

相关信息

ST_MetaData, ST_BandMetaData, ST_Summary ST_Extent

11.4.19 ST_UpperLeftX

ST_UpperLeftX — 返回投影空栅参考中栅格的左上角 X 坐栅。

Synopsis

float8 **ST_UpperLeftX**(raster rast);

描述

返回投影空栅参考中栅格的左上角 X 坐栅。

示例

```
SELECt rid, ST_UpperLeftX(rast) As ulx
FROM dummy_rast;

 rid |    ulx
-----+------------
   1 |        0.5
   2 | 3427927.75
```

相关信息

ST_UpperLeftY, ST_GeoReference, Box3D

11.4.20 ST_UpperLeftY

ST_UpperLeftY — 返回投影空⌧参考中⌧格的左上角 Y 坐⌧。

Synopsis

float8 **ST_UpperLeftY**(raster rast);

描述

返回投影空⌧参考中⌧格的左上角 Y 坐⌧。

示例

```
SELECT rid, ST_UpperLeftY(rast) As uly
FROM dummy_rast;

 rid |   uly
-----+---------
   1 |     0.5
   2 | 5793244
```

相关信息

ST_UpperLeftX, ST_GeoReference, Box3D

11.4.21 ST_Width

ST_Width — 返回⌧格的⌧度（以像素⌧⌧位）。

Synopsis

integer **ST_Width**(raster rast);

描述

返回⍰格的⍰度（以像素⍰⍰位）。

示例

```
SELECT ST_Width(rast) As rastwidth
FROM dummy_rast WHERE rid=1;

rastwidth
----------------
10
```

相关信息

ST_Height

11.4.22 ST_WorldToRasterCoord

ST_WorldToRasterCoord — ⍰定几何 X 和 Y（⍰度和⍰度）或以⍰格的空⍰参考坐⍰系表示的点几何，将左上角作⍰列和行返回。

Synopsis

record **ST_WorldToRasterCoord**(raster rast, geometry pt);
record **ST_WorldToRasterCoord**(raster rast, double precision longitude, double precision latitude);

描述

在⍰定几何 X 和 Y（⍰度和⍰度）或点几何的情况下，将左上角作⍰列和行返回。无⍰几何 X 和 Y 或点几何是否位于⍰格范⍰之外，此函数都会起作用。几何 X 和 Y 必⍰在⍰格的空⍰参考坐⍰系中表示。

可用性：2.1.0

示例

```
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
```

相关信息

ST_WorldToRasterCoordX, ST_WorldToRasterCoordY, ST_RasterToWorldCoordX, ST_RasterToWorldCoordY
ST_SRID

11.4.23 ST_WorldToRasterCoordX

ST_WorldToRasterCoordX — 返回⊠格中点几何 (pt) 的列或以⊠格世界空⊠参考系表示的 X 和 Y 世界坐⊠ (xw, yw)。

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);

描述

返回点几何 (pt) 或 X 和 Y 世界坐⊠ (xw, yw) 的⊠格中的列。一个点，或者（如果⊠格⊠斜，⊠需要 xw 和 yw 世界坐⊠）。如果⊠格没有⊠斜，⊠ xw 就足⊠了。世界坐⊠位于⊠格的空⊠参考坐⊠系中。

更改：2.1.0 在之前的版本中，⊠称⊠ ST_World2RasterCoordX

示例

```
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
```

相关信息

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

11.4.24 ST_WorldToRasterCoordY

ST_WorldToRasterCoordY — 返回点几何⊠形 (pt) 的⊠格中的行或以⊠格的世界空⊠参考系表示的 X 和 Y 世界坐⊠ (xw, yw)。

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);

描述

返回点几何 (pt) 或 X 和 Y 世界坐⊠ (xw, yw) 的⊠格中的行。一个点，或者（如果⊠格⊠斜，⊠需要 xw 和 yw 世界坐⊠）。如果⊠格没有⊠斜，⊠ xw 就足⊠了。世界坐⊠位于⊠格的空⊠参考坐⊠系中。

更改：2.1.0 在之前的版本中，⊠称⊠ ST_World2RasterCoordY

示例

```
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
```

相关信息

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

11.5 ☒格波段☒☒器

11.5.1 ST_BandMetaData

ST_BandMetaData — 返回特定☒格波段的基本元数据。如果未指定，☒假定波段☒号☒ 1。

Synopsis

(1) record **ST_BandMetaData**(raster rast, integer band=1);
(2) record **ST_BandMetaData**(raster rast, integer[] band);

描述

返回有关☒格波段的基本元数据。返回的列：pixeltype、nodatavalue、isoutdb、path、outdbbandnum、filesize、filetimestamp。

Note
如果☒格不包含波段，☒会引☒☒☒。

Note
如果 band 没有 NODATA ☒，☒ nodatavalue ☒ NULL。

Note
如果 isoutdb ☒ False，☒路径、outdbbandnum、文件大小和文件☒☒戳☒ NULL。如果禁用 outdb ☒☒，☒文件大小和文件☒☒戳也将☒ NULL。

增☒：2.5.0 包括 *outdbbandnum*、文件大小和文件☒☒戳 for outdb ☒格。

示例：格式 **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       |      |
```

示例：格式 **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 |
```

相关信息

ST_MetaData, ST_BandPixelType

11.5.2 ST_BandNoDataValue

ST_BandNoDataValue — 返回表示无数据的⌧定波段中的⌧。如果没有指定波段⌧号，默⌧⌧波段 1。

Synopsis

double precision **ST_BandNoDataValue**(raster rast, integer bandnum=1);

描述

返回表示⊠波段中无数据的数⊠

示例

```
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
```

相关信息

ST_NumBands

11.5.3 ST_BandIsNoData

ST_BandIsNoData — 如果波段⊠填充无数据⊠，⊠返回 true。

Synopsis

boolean **ST_BandIsNoData**(raster rast, integer band, boolean forceChecking=true);
boolean **ST_BandIsNoData**(raster rast, boolean forceChecking=true);

描述

如果波段⊠填充无数据⊠，⊠返回 true。如果未指定，⊠假定波段⊠ 1。如果最后一个参数⊠ TRUE，⊠逐像素⊠⊠整个波段。否⊠，⊠函数⊠返回⊠⊠的 isnodata ⊠志的⊠。如果未指定，此参数的默⊠⊠⊠ FALSE。

可用性: 2.0.0

Note
如果⊠志位⊠"dirty"（即，使用 TRUE 作⊠最后一个参数和不使用它的⊠果不同），您⊠⊠更新⊠格，将⊠⊠志⊠置⊠ true，可以使用 ST_SetBandIsNodata() 或 ST_SetBandNodataValue()，并将 TRUE 作⊠最后一个参数。⊠参⊠ ST_SetBandIsNoData。

示例

```
-- 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
```

相关信息

ST_BandNoDataValue, ST_NumBands, ST_SetBandNoDataValue, ST_SetBandIsNoData

11.5.4 ST_BandPath

ST_BandPath — 返回存⊠在文件系⊠中的波段的系⊠文件路径。如果未指定波段号，⊠假定⊠ 1。

Synopsis

text **ST_BandPath**(raster rast, integer bandnum=1);

描述

返回指定波段的系⌧文件路径。如果⌧用⌧使用数据⌧中的波段，⌧会抛出⌧⌧。

示例

相关信息

11.5.5 ST_BandFileSize

ST_BandFileSize — 返回文件系⌧中存⌧的波段的文件大小。如果未指定波段号，⌧假定⌧ 1。

Synopsis

bigint **ST_BandFileSize**(raster rast, integer bandnum=1);

描述

返回文件系⌧中存⌧的波段的文件大小。如果使用 in db band ⌧用，或者未⌧用 outdb ⌧⌧，⌧会引⌧⌧⌧。

此函数通常与 ST_BandPath() 和 ST_BandFileTimestamp() ⌧合使用，以便客⌧端可以确定它看到的 outdb ⌧格文件名是否与服⌧器看到的文件名相同。

可用性：2.5.0

示例

```
SELECT ST_BandFileSize(rast,1) FROM dummy_rast WHERE rid = 1;

 st_bandfilesize
-----------------
          240574
```

11.5.6 ST_BandFileTimestamp

ST_BandFileTimestamp — 返回文件系⌧中存⌧的波段的文件⌧⌧戳。如果未指定波段号，⌧假定⌧ 1。

Synopsis

bigint **ST_BandFileTimestamp**(raster rast, integer bandnum=1);

描述

返回文件系⌧中存⌧的波段的文件⌧⌧戳（自 1970 年 1 月 1 日 00:00:00 UTC 以来的秒数）。如果使用 in db band ⌧用，或者未⌧用 outdb ⌧⌧，⌧会引⌧⌧⌧。

此函数通常与 ST_BandPath() 和 ST_BandFileSize() ⌧合使用，以便客⌧端可以确定它所看到的 outdb ⌧格的文件名是否与服⌧器所看到的相同。

可用性：2.5.0

示例

```
SELECT ST_BandFileTimestamp(rast,1) FROM dummy_rast WHERE rid = 1;

 st_bandfiletimestamp
----------------------
           1521807257
```

11.5.7 ST_BandPixelType

ST_BandPixelType — 返回⌧定波段的像素⌧型。如果未指定波段号，⌧假定⌧ 1。

Synopsis

text **ST_BandPixelType**(raster rast, integer bandnum=1);

描述

返回描述⌧定⌧区每个⌧元格中存⌧的数据⌧型和⌧大小的名称。
有 11 种像素⌧型。支持的像素⌧型如下：

- 1BB - 1 位布⌧⌧
- 2BUI - 2 位无符号整数
- 4BUI - 4 位无符号整数
- 8BSI - 8 位有符号整数
- 8BUI - 8 位无符号整数
- 16BSI - 16 位有符号整数
- 16BUI - 16 位无符号整数
- 32BSI - 32 位有符号整数
- 32BUI - 32 位无符号整数
- 32BF - 32 位浮点数
- 64BF - 64 位浮点数

示例

```
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
```

相关信息

ST_NumBands

11.5.8 ST_MinPossibleValue

ST_MinPossibleValue — 返回此像素☒型可以存☒的最小☒。

Synopsis

integer **ST_MinPossibleValue**(text pixeltype);

描述

返回此像素☒型可以存☒的最小☒。

示例

```
SELECT ST_MinPossibleValue('16BSI');

 st_minpossiblevalue
---------------------
              -32768


SELECT ST_MinPossibleValue('8BUI');

 st_minpossiblevalue
---------------------
                   0
```

相关信息

ST_BandPixelType

11.5.9 ST_HasNoBand

ST_HasNoBand — 如果不存在具有☒定波段☒号的波段，☒返回 true。如果未指定波段号，☒假定波段号☒ 1。

Synopsis

boolean **ST_HasNoBand**(raster rast, integer bandnum=1);

描述

如果不存在具有☒定波段☒号的波段，☒返回 true。如果未指定波段号，☒假定波段号☒ 1。

可用性: 2.0.0

示例

```
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
```

相关信息

ST_NumBands

11.6 ☒格像素☒☒器和☒置器

11.6.1 ST_PixelAsPolygon

ST_PixelAsPolygon — 返回限定特定行和列的像素的多☒形几何形状。

Synopsis

geometry **ST_PixelAsPolygon**(raster rast, integer columnx, integer rowy);

描述

返回限定特定行和列的像素的多☒形几何形状。

可用性: 2.0.0

示例

```
-- 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, ..
```

相关信息

ST_DumpAsPolygons, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids, ST_Intersection, ST_AsText

11.6.2 ST_PixelAsPolygons

ST_PixelAsPolygons — 返回包⊠⊠格⊠的每个像素的多⊠形几何⊠形以及每个像素的⊠、X 和 Y ⊠格坐⊠。

Synopsis

setof record **ST_PixelAsPolygons**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

描述

返回包⊠⊠格⊠的每个像素的多⊠形几何⊠形以及每个像素的⊠（双精度）、X 和 Y ⊠格坐⊠（整数）。
返回⊠⊠格式：*geom* geometry，*val* 双精度，*x* 整数，*y* 整数。

Note
当 *except_nodata_value* = TRUE ⊠，⊠将那些⊠不是 NODATA 的像素作⊠点返回。

Note
ST_PixelAsPolygons ⊠每个像素返回一个多⊠形几何⊠形。⊠与 ST_DumpAsPolygons 不同，其中每个几何⊠形代表一个或多个具有相同像素⊠的像素。

可用性: 2.0.0
增⊠：2.1.0 添加了 except_nodata_value 可⊠参数。
更改：2.1.1 更改了 except_nodata_value 的行⊠。

示例

```
-- 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))
```

相关信息

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid, ST_Pixel
ST_AsText

11.6.3 ST_PixelAsPoint

ST_PixelAsPoint — 返回像素左上角的点几何形状。

Synopsis

geometry **ST_PixelAsPoint**(raster rast, integer columnx, integer rowy);

描述

返回像素左上角的点几何形状。

可用性：2.1.0

示例

```
SELECT ST_AsText(ST_PixelAsPoint(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;

   st_astext
----------------
 POINT(0.5 0.5)
```

相关信息

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids

11.6.4 ST_PixelAsPoints

ST_PixelAsPoints — 返回⌧格波段的每个像素的点几何⌧形以及每个像素的⌧、X 和 Y ⌧格坐⌧。点几何的坐⌧是像素的左上角。

Synopsis

setof record **ST_PixelAsPoints**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

描述

返回⌧格波段的每个像素的点几何⌧形以及每个像素的⌧、X 和 Y ⌧格坐⌧。点几何的坐⌧是像素的左上角。

返回⌧⌧格式：*geom* geometry，*val* 双精度，*x* 整数，*y* 整数。

Note

当 *except_nodata_value* = TRUE ⌧，⌧将那些⌧不是 NODATA 的像素作⌧点返回。

可用性：2.1.0

更改：2.1.1 更改了 except_nodata_value 的行⌧。

示例

```
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)
```

相关信息

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsCentroid, ST_PixelAsCentroids

11.6.5 ST_PixelAsCentroid

ST_PixelAsCentroid — 返回像素表示的区域的⌧心（点几何）。

Synopsis

geometry **ST_PixelAsCentroid**(raster rast, integer x, integer y);

描述

返回像素表示的区域的⌧心（点几何）。

增⌧：3.2.0 Faster ⌧在用 C ⌧⌧。

可用性：2.1.0

示例

```
SELECT ST_AsText(ST_PixelAsCentroid(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;

  st_astext
--------------
 POINT(1.5 2)
```

相关信息

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_Pixel

11.6.6 ST_PixelAsCentroids

ST_PixelAsCentroids — 返回⌧格波段的每个像素的⌧心（点几何）以及每个像素的⌧、X 和 Y ⌧格坐⌧。点几何是像素表示的区域的⌧心。

Synopsis

setof record **ST_PixelAsCentroids**(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);

描述

返回⌧格波段的每个像素的⌧心（点几何）以及每个像素的⌧、X 和 Y ⌧格坐⌧。点几何是像素表示的区域的⌧心。

返回⌧⌧格式：*geom* geometry，*val* 双精度，*x* 整数，*y* 整数。

Note

当 *except_nodata_value* = TRUE ⌧，⌧将那些⌧不是 NODATA 的像素作⌧点返回。

增⌧：3.2.0 Faster ⌧在用 C ⌧⌧。

更改：2.1.1 更改了 except_nodata_value 的行⌧。

可用性：2.1.0

示例

```
--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)
```

相关信息

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoint, ST_PixelAsPoints, ST_Pixel

11.6.7 ST_Value

ST_Value — 返回指定列 x、行 y 像素或特定几何点⊠⊠定波段的⊠。波段⊠号从 1 开始，如果未指定，⊠默⊠⊠ 1。如果将参数 `exclude_nodata_value` ⊠置⊠ false，⊠所有像素都被⊠⊠与 `nodata` 像素相交并返回其⊠。如果未⊠⊠参数 `exclude_nodata_value`，⊠从⊠格的元数据中⊠取⊠⊠。

Synopsis

double precision **ST_Value**(raster rast, geometry pt, boolean exclude_nodata_value=true);
double precision **ST_Value**(raster rast, integer band, geometry pt, boolean exclude_nodata_value=true, text resample='nearest');
double precision **ST_Value**(raster rast, integer x, integer y, boolean exclude_nodata_value=true);
double precision **ST_Value**(raster rast, integer band, integer x, integer y, boolean exclude_nodata_value=tru

描述

返回⊠定列 x、行 y 像素或⊠定几何点⊠⊠定波段的⊠。波段⊠号从 1 开始，如果未指定波段，⊠默⊠⊠ 1。

如果 `except_nodata_value` ⊠置⊠ true，⊠⊠考⊠非 `nodata` 像素。如果 `except_nodata_value` ⊠置⊠ false，⊠考⊠所有像素。

重采⊠参数的允⊠⊠⊠ "nearest"（⊠行默⊠的最近⊠重采⊠）和 "bilinear"（⊠行 双⊠性插⊠ 以估⊠像素中心之⊠的⊠）。

增⊠：添加了 3.2.0 重新采⊠可⊠参数。

增⊠：2.0.0 添加了 except_nodata_value 可⊠参数。

示例

```
-- 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)))
```

相关信息

ST_SetValue, ST_DumpAsPolygons, ST_NumBands, ST_PixelAsPolygon, ST_ScaleX, ST_ScaleY, ST_UpperLe
ST_UpperLeftY, ST_SRID, ST_AsText, ST_Point, ST_MakeEnvelope, ST_Intersects, ST_Intersection

11.6.8 ST_NearestValue

ST_NearestValue — 返回由 columnx 和 rowy 指定的⊠定⊠像素的最接近的非 NODATA ⊠或以与⊠格相同的空⊠参考坐⊠系表示的几何点。

Synopsis

double precision **ST_NearestValue**(raster rast, integer bandnum, geometry pt, boolean exclude_nodata_val
double precision **ST_NearestValue**(raster rast, geometry pt, boolean exclude_nodata_value=true);
double precision **ST_NearestValue**(raster rast, integer bandnum, integer columnx, integer rowy, boolean exclude_nodata_value=true);
double precision **ST_NearestValue**(raster rast, integer columnx, integer rowy, boolean exclude_nodata_valu

描述

返回⊠定列 x、行 y 像素或特定几何点中⊠定⊠的最接近的非 NODATA ⊠。如果列 x、行 y 像素或指定几何点⊠的像素⊠ NODATA，⊠函数将⊠找距离列 x、行 y 像素或⊠⊠非 NODATA 的几何点最近的像素。

Band ⊠号从 1 开始，如果未指定，⊠ bandnum 假定⊠ 1。如果 except_nodata_value ⊠置⊠ false，⊠所有包含 nodata 像素的像素都被⊠⊠相交并返回⊠。如果 exclude_nodata_value 未⊠入，⊠从⊠格的元数据中⊠取它。

可用性：2.1.0

Note
ST_NearestValue 是 ST_Value 的直接替代品。

示例

```
-- 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
```

相关信息

ST_Neighborhood, ST_Value

11.6.9 ST_SetZ

ST_SetZ — 返回与⌧入几何⌧形具有相同 X/Y 坐⌧的几何⌧形，并使用⌧求的重采⌧算法将⌧格中的⌧复制到 Z ⌧度。

Synopsis

geometry **ST_SetZ**(raster rast, geometry geom, text resample=nearest, integer band=1);

描述

返回与⌧入几何⌧形具有相同 X/Y 坐⌧的几何⌧形，并使用⌧求的重采⌧算法将⌧格中的⌧复制到 Z ⌧度。

重采⌧参数可以⌧置⌧”nearest”，以复制每个⌧点所在⌧元格的⌧，或者⌧置⌧”bilinear”，以使用双⌧性插⌧来⌧算考⌧到相⌧⌧元格的⌧。

可用性：3.2.0

示例

```
--
-- 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)
```

相关信息

ST_Value, ST_SetM

11.6.10 ST_SetM

ST_SetM — 返回与☒入几何☒形具有相同 X/Y 坐☒的几何☒形，并使用☒求的重采☒算法将☒格中的☒复制到 M ☒度。

Synopsis

geometry **ST_SetM**(raster rast, geometry geom, text resample=nearest, integer band=1);

描述

返回与☒入几何☒形具有相同 X/Y 坐☒的几何☒形，并使用☒求的重采☒算法将☒格中的☒复制到 M ☒度。

重采☒参数可以☒置☒"nearest"，以复制每个☒点所在☒元格的☒，或者☒置☒"bilinear"，以使用双☒性插☒来☒算考☒到相☒☒元格的☒。

可用性：3.2.0

示例

```
--
-- 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)
```

相关信息

ST_Value, ST_SetZ

11.6.11 ST_Neighborhood

ST_Neighborhood — 返回⊠定波段像素周⊠非 NODATA ⊠的二⊠双精度数⊠，⊠⊠定波段像素由 columnX 和 rowY 或以与⊠格相同的空⊠参考坐⊠系表示的几何点指定。

Synopsis

double precision[][] **ST_Neighborhood**(raster rast, integer bandnum, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
double precision[][] **ST_Neighborhood**(raster rast, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
double precision[][] **ST_Neighborhood**(raster rast, integer bandnum, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
double precision[][] **ST_Neighborhood**(raster rast, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

描述

返回一个二⊠双精度数⊠，其中包含⊠⊠⊠定波段的像素的非 NODATA ⊠。⊠可以通⊠指定列 X 和行 Y，或者使用与⊠格相同的空⊠参考坐⊠系⊠表示的几何点来完成。参数 distanceX 和 distanceY 定⊠了 X 和 Y ⊠上指定像素周⊠的像素数量，例如，我想⊠取我感⊠趣的像素周⊠ X ⊠方向上 3 个像素距离和 Y ⊠方向上 2 个像素距离内的所有⊠。二⊠数⊠的中心⊠将是由列 X 和行 Y 或几何点指定的像素的⊠。

Band ⊠号从 1 开始，如果未指定，⊠ bandnum 假定⊠ 1。如果 except_nodata_value ⊠置⊠ false，⊠所有包含 nodata 像素的像素都被⊠⊠相交并返回⊠。如果 exclude_nodata_value 未⊠入，⊠从⊠格的元数据中⊠取它。

Note
返回的二⌧数⌧沿每个⌧的元素数量⌧ 2 * (distanceX|distanceY) 1。因此，⌧于 distanceX 和 distanceY ⌧ 1，返回的数⌧将⌧ 3x3。

Note
二⌧数⌧⌧出可以⌧⌧⌧任何⌧格⌧理内置函数，例如 ST_Min4ma、ST_Sum4ma、ST_Mean4ma。

可用性：2.1.0

示例

```
-- 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}}
```

相关信息

ST_NearestValue, ST_Min4ma, ST_Max4ma, ST_Sum4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.6.12 ST_SetValue

ST_SetValue — 返回修改后的⌧格，其⌧果是将⌧定波段中的⌧⌧置⌧指定列 x、行 y 像素或与特定几何⌧形相交的像素。波段⌧号从 1 开始，如果未指定波段，⌧默⌧⌧ 1。

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);

描述

返回修改后的⌧格，⌧⌧格是通⌧将指定像素的⌧⌧置⌧⌧定⌧格的行和列或几何⌧形的指定波段的新⌧而⌧生的。如果未指定波段，⌧假定⌧ 1。

增⌧：2.1.0 ST_SetValue() 的几何⌧体⌧在支持任何几何⌧型，而不⌧⌧是点。几何⌧体是 ST_SetValues() 的 geomval[] ⌧体的包装

示例

```
                -- 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   ;
```

相关信息

ST_Value, ST_DumpAsPolygons

11.6.13 ST_SetValues

ST_SetValues — 返回通⌧⌧置⌧定波段的⌧而⌧生的修改后的⌧格。

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);

描述

返回通⌧将指定波段的指定像素⌧置⌧新⌧而⌧生的修改后的⌧格。`columnx` 和 `rowy` 是从 1 开始索引的。

如果 `keepnodata` ⌧ TRUE，那些⌧⌧ NODATA 的像素将不会被⌧置⌧ `newvalueset` 中的相⌧⌧。

□于□体 1，要□置的特定像素由 columnx、rowy 像素坐□和 newvalueset 数□的尺寸决定。noset 可以用来防止□置具有 newvalueset 中存在的□的像素（因□ PostgreSQL 不允□不□□数□）。□参□□体 1 的示例。

□体 2 与□体 1 □似，但是使用□□的双精度 nosetvalue 而不是布□型的 noset 数□。newvalueset 中具有 nosetvalue □的元素将被跳□。□参□□体 2 的示例。

□于□体 3，要□置的特定像素由 columnx、rowy 像素坐□、width 和 height 决定。□参□□体 3 的示例。

□体 4 与□体 3 相同，唯一的区□是它假定将□置 rast 的第一个波段的像素。

□于□体 5，将使用geomval数□来确定要□置的特定像素。如果数□中的所有几何□形都是点□型或多点□型，函数将使用一种快捷方式，其中每个点的□度和□度用于直接□置像素。否□，几何□形将□□□□格，然后一次迭代通□它□。□参□□体 5 的示例。

可用性：2.1.0

示例：格式 **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 | 2 |   9
 3 | 3 |   9
```

示例：格式 **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
```

示例：⊠体 **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
```

示例：⊠体 **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)
```

下面⊠示了数⊠中后面的 geomvals 可以覆盖前面的 geomvals

```
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)
```

⊠个例子与前面的例子相反

```
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)
```

相关信息

ST_Value, ST_SetValue, ST_PixelAsPolygons

11.6.14 ST_DumpValues

ST_DumpValues — ⊠取指定 band（波段）的⊠作⊠二⊠数⊠。

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);

描述

⊠取指定 band 的⊠作⊠二⊠数⊠（第一个索引是行，第二个索引是列）。如果 nband ⊠ NULL 或未提供，⊠⊠理所有⊠格波段。

可用性：2.1.0

示例

```
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)
```

相关信息

ST_Value, ST_SetValue, ST_SetValues

11.6.15 ST_PixelOfValue

ST_PixelOfValue — 取等于搜索的像素的列 x、行 y 坐。

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);

描述

☒取☒等于搜索☒的像素的列 x、行 y 坐☒。如果未指定波段，☒假定☒波段 1。

可用性：2.1.0

示例

```
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 ☒格☒☒器

11.7.1 ST_SetGeoReference

ST_SetGeoReference — 在一次☒用中☒置 Georeference 6 地理配准参数。数字☒☒用空格分隔。接受 GDAL 或 ESRI 格式的☒入。默☒☒ GDAL。

Synopsis

raster **ST_SetGeoReference**(raster rast, text georefcoords, text format=GDAL);
raster **ST_SetGeoReference**(raster rast, double precision upperleftx, double precision upperlefty, double precision scalex, double precision scaley, double precision skewx, double precision skewy);

描述

在一次☒用中☒置 Georeference 6 地理配准参数。接受“GDAL”或“ESRI”格式的☒入。默☒☒ GDAL。如果未提供 6 个坐☒，☒返回 null。

格式表示的区☒如下：

GDAL:

```
scalex skewy skewx scaley upperleftx upperlefty
```

ESRI:

```
scalex skewy skewx scaley upperleftx + scalex*0.5 upperlefty + scaley*0.5
```

Note
如果☒格具有数据☒外波段，☒更改地理参考可能会☒致无法正确☒☒☒波段的外部存☒数据。

增☒：2.1.0 添加 ST_SetGeoReference(raster, double precision, ...) ☒体

示例

```
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
```

相关信息

ST_GeoReference, ST_ScaleX, ST_ScaleY, ST_UpperLeftX, ST_UpperLeftY

11.7.2 ST_SetRotation

ST_SetRotation — 以弧度⊠置⊠格的旋⊠。

Synopsis

raster **ST_SetRotation**(raster rast, float8 rotation);

描述

均匀旋⊠⊠格。旋⊠以弧度⊠⊠位。有关更多⊠⊠信息，⊠参⊠世界文件。

示例

```
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
```

相关信息

ST_Rotation, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

11.7.3 ST_SetScale

ST_SetScale — 以坐⌧参考系⌧⌧位⌧置像素的 X 和 Y 大小。数字⌧位/像素⌧度/高度。

Synopsis

raster **ST_SetScale**(raster rast, float8 xy);
raster **ST_SetScale**(raster rast, float8 x, float8 y);

描述

以坐⌧参考系⌧⌧位⌧置像素的 X 和 Y 大小。数字⌧位/像素⌧度/高度。如果只⌧入一个⌧位，⌧假⌧ X 和 Y 是相同的数字。

Note
ST_SetScale 与ST_Rescale 的不同之⌧在于 ST_SetScale 不会⌧⌧格重新采⌧以匹配⌧格范⌧。它⌧更改⌧格的元数据（或地理参考）以⌧正最初⌧⌧指定的⌧放比例。ST_Rescale 会生成具有不同⌧度和高度的⌧格，⌧算⌧果以适合⌧入⌧格的地理范⌧。ST_SetScale 不修改⌧格的⌧度或高度。

更改：2.0.0 在 WKTRaster 版本中，⌧称⌧ ST_SetPixelSize。⌧在 2.0.0 中⌧生了⌧化。

示例

```
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)
```

相关信息

ST_ScaleX, ST_ScaleY, Box3D

11.7.4 ST_SetSkew

ST_SetSkew — 设置地理参考 X 和 Y 设斜（或旋设参数）。如果只设入一个，设将 X 和 Y 设置设相同的设。

Synopsis

raster **ST_SetSkew**(raster rast, float8 skewxy);
raster **ST_SetSkew**(raster rast, float8 skewx, float8 skewy);

描述

设置地理参考 X 和 Y 设斜（或旋设参数）。如果只设入一个，设将 X 和 Y 设置设相同的设。有关更多设设信息，设参设世界文件。

示例

```
-- 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
```

相关信息

ST_GeoReference, ST_SetGeoReference, ST_SkewX, ST_SkewY

11.7.5 ST_SetSRID

ST_SetSRID — 将栅格的 SRID 设置在 Spatial_ref_sys 表中定义的特定整数 srid。

Synopsis

raster **ST_SetSRID**(raster rast, integer srid);

描述

将栅格上的 SRID 设置特定整数。

Note
此函数不会以任何方式变换栅格 - 它只是设置定义其当前所在坐标参考系统的空间参考的元数据。对于以后的变换很有用。

相关信息

Section 4.5, ST_SRID

11.7.6 ST_SetUpperLeft

ST_SetUpperLeft — 将栅格左上角的变换设置投影的 X 和 Y 坐标。

Synopsis

raster **ST_SetUpperLeft**(raster rast, double precision x, double precision y);

描述

将栅格像素左上角的变换设置投影的 X 和 Y 坐标

示例

```
SELECT ST_SetUpperLeft(rast,-71.01,42.37)
FROM dummy_rast
WHERE rid = 2;
```

相关信息

ST_UpperLeftX, ST_UpperLeftY

11.7.7 ST_Resample

ST_Resample — 重采⌧一个⌧格⌧像，可以指定重新采⌧算法、新的尺寸、任意的⌧格角点，以及一⌧⌧格地理参考属性，⌧些属性可以自己定⌧，也可以从⌧一个⌧格⌧像中借用。

Synopsis

raster **ST_Resample**(raster rast, integer width, integer height, double precision gridx=NULL, double precision gridy=NULL, double precision skewx=0, double precision skewy=0, text algorithm=NearestNeigh double precision maxerr=0.125);
raster **ST_Resample**(raster rast, double precision scalex=0, double precision scaley=0, double precision gridx=NULL, double precision gridy=NULL, double precision skewx=0, double precision skewy=0, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Resample**(raster rast, raster ref, text algorithm=NearestNeighbor, double precision maxerr=0.125, boolean usescale=true);
raster **ST_Resample**(raster rast, raster ref, boolean usescale, text algorithm=NearestNeighbor, double precision maxerr=0.125);

描述

使用指定的重新采⌧算法、新的尺寸（width&height）、一个网格角点（gridx & gridy），以及一⌧⌧格地理参考属性（scalex、scaley、skewx & skewy），⌧些属性可以自己定⌧，也可以从⌧一个⌧格⌧像中借用。如果使用参考⌧格⌧像，那么⌧⌧个⌧格⌧像必⌧具有相同的空⌧参考 ID（SRID）。

使用以下重采⌧算法之一⌧算新像素⌧：

- NearestNeighbor (英⌧或美式拼写)
- Bilinear
- Cubic
- CubicSpline
- Lanczos
- Max
- Min

默⌧⌧是“NearestNeighbor”，它速度最快，但插⌧效果最差。

如果未指定 `maxerr`，⌧使用 0.125 的 maxerror 百分比。

Note
有关更多⌧⌧信息，⌧参⌧：GDAL Warp 重采⌧方法。

可用性：2.0.0 需要 GDAL 1.6.1+

增⌧：3.4.0 添加了最大和最小重采⌧⌧⌧

示例

```
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
```

相关信息

ST_Rescale, ST_Resize, ST_Transform

11.7.8 ST_Rescale

ST_Rescale — 通□□□整□格的比例（或像素大小）来重新采□□格。新的像素□是使用 NearestNeighbor（英□或美式拼写）、Bilinear、Cubic、CubicSpline、Lanczos、Max 或 Min 重采□算法□算的。默□□ NearestNeighbor。

Synopsis

raster **ST_Rescale**(raster rast, double precision scalexy, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Rescale**(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeig
double precision maxerr=0.125);

描述

通□□□整□格的比例（或像素大小）来重新采□□格。使用以下重采□算法之一□算新像素□：

- NearestNeighbor (英□或美式拼写)
- Bilinear
- Cubic
- CubicSpline
- Lanczos
- Max
- Min

默⊠⊠是“NearestNeighbor”，它速度最快，但插⊠效果最差。

scalex 和 scaley 定⊠新的像素大小。⊠了⊠得良好定向的⊠格，scaley 通常必⊠⊠⊠⊠。

当新的 scalex 或 scaley 不是⊠格⊠度或高度的⊠数⊠，生成的⊠格的范⊠将⊠展以包含所提供⊠格的范⊠。如果您想确保保留精确的⊠入范⊠，⊠参⊠ ST_Resize

maxerr 是重采⊠算法⊠行⊠⊠近似的⊠⊠（以像素⊠⊠位）。如果未指定 maxerr，⊠使用默⊠⊠ 0.125，⊠⊠与 GDAL gdalwarp ⊠用程序中使用的⊠相同。如果⊠置⊠零，⊠不会⊠生近似。

Note
有关更多⊠⊠信息，⊠参⊠：GDAL Warp 重采⊠方法。

Note
ST_Rescale 与ST_SetScale的不同之⊠在于 ST_SetScale 不会⊠⊠格重新采⊠以匹配⊠格范⊠。ST_SetScale ⊠更改⊠格的元数据（或地理参考）以⊠正最初⊠⊠指定的⊠放比例。ST_Rescale 会生成具有不同⊠度和高度的⊠格，⊠算⊠果以适合⊠入⊠格的地理范⊠。ST_SetScale 不修改⊠格的⊠度或高度。

可用性：2.0.0 需要 GDAL 1.6.1+

增⊠：3.4.0 添加了最大和最小重采⊠⊠⊠

更改：2.1.0 适用于没有 SRID 的⊠格

示例

将⊠格从 0.001 度的像素大小重新⊠放⊠ 0.0015 度的像素大小的⊠⊠示例。

```
-- 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
```

相关信息

ST_Resize, ST_Resample, ST_SetScale, ST_ScaleX, ST_ScaleY, ST_Transform

11.7.9 ST_Reskew

ST_Reskew — 通⊠⊠⊠整⊠格的⊠斜（或旋⊠参数）来重采⊠⊠格。新的像素⊠是使用 NearestNeighbor（英⊠或美式拼写）、Bilinear、Cubic、CubicSpline 或 Lanczos 重采⊠算法⊠算的。默⊠⊠ NearestNeighbor。

Synopsis

raster **ST_Reskew**(raster rast, double precision skewxy, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_Reskew**(raster rast, double precision skewx, double precision skewy, text algorithm=NearestNeig double precision maxerr=0.125);

描述

通⊠⊠⊠整⊠格的⊠斜（或旋⊠参数）来重采⊠⊠格。新的像素⊠是使用 NearestNeighbor （英⊠或美式拼写）、Bilinear, Cubic, CubicSpline 或 Lanczos 重采⊠算法⊠算的。默⊠⊠是 “NearestNeighbor”，它速度最快，但插⊠效果最差。

`skewx` 和 `skewy` 定⊠新的⊠斜。

新⊠格的范⊠将包含所提供⊠格的范⊠。

如果未指定 `maxerr`，⊠ maxerror 百分比⊠ 0.125。

Note
有关更多⊠⊠信息，⊠参⊠：GDAL Warp 重采⊠方法。

Note
ST_Reskew 与 ST_SetSkew 的不同之⊠在于 ST_SetSkew 不会⊠⊠格重新采⊠以匹配⊠格范⊠。ST_SetSkew ⊠更改⊠格的元数据（或地理参考）以⊠正最初⊠⊠指定的⊠斜。ST_Reskew 会生成具有不同⊠度和高度的⊠格，⊠算⊠果以适合⊠入⊠格的地理范⊠。ST_SetSkew 不修改⊠格的⊠度或高度。

可用性：2.0.0 需要 GDAL 1.6.1+

更改：2.1.0 适用于没有 SRID 的⊠格

示例

将⊠格从⊠斜 0.0 重新⊠斜到⊠斜 0.0015 的⊠⊠示例。

```
-- 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
```

相关信息

ST_Resample, ST_Rescale, ST_SetSkew, ST_SetRotation, ST_SkewX, ST_SkewY, ST_Transform

---

11.7.10 ST_SnapToGrid

ST_SnapToGrid — 通⊠将⊠格捕捉到网格来重采⊠⊠格。新的像素⊠是使用 NearestNeighbor（英⊠或美式拼写）、Bilinear, Cubic, CubicSpline 或 Lanczos 重采⊠算法⊠算的。默⊠⊠ NearestNeighbor。

Synopsis

raster **ST_SnapToGrid**(raster rast, double precision gridx, double precision gridy, text algorithm=NearestN
double precision maxerr=0.125, double precision scalex=DEFAULT 0, double precision scaley=DEFAULT
0);
raster **ST_SnapToGrid**(raster rast, double precision gridx, double precision gridy, double precision
scalex, double precision scaley, text algorithm=NearestNeighbor, double precision maxerr=0.125);
raster **ST_SnapToGrid**(raster rast, double precision gridx, double precision gridy, double precision
scalexy, text algorithm=NearestNeighbor, double precision maxerr=0.125);

描述

通⊠将⊠格⊠像重新采⊠，将其⊠⊠到由任意像素角点（gridx & gridy）定⊠的网格，可⊠地使用像素大小（scalex & scaley）。新的像素⊠是使用最近⊠（NearestNeighbor，英式或美式拼写）、双⊠性（Bilinear）、三次⊠条（Cubic）、立方⊠条（CubicSpline）或⊠索斯（Lanczos）重新采⊠算法⊠算的。默⊠情况下使用最近⊠（NearestNeighbor），⊠是最快但插⊠效果最差的算法。

`gridx` 和 `gridy` 定⊠新网格的任意像素角。⊠不一定是新⊠格的左上角，也不必位于新⊠格范⊠的内部或⊠⊠。

您可以⊠⊠使用 `scalex` 和 `scaley` 定⊠新网格的像素大小。

新⊠格的范⊠将包含所提供⊠格的范⊠。

如果未指定 `maxerr`，⊠ maxerror 百分比⊠ 0.125。

Note
有关更多⊠⊠信息，⊠参⊠：GDAL Warp 重采⊠方法。

Note
如果您需要⊠网格参数⊠行更多控制，⊠使用 ST_Resample。

可用性：2.0.0 需要 GDAL 1.6.1+

更改：2.1.0 适用于没有 SRID 的⊠格

示例

将⊠格捕捉到稍微不同的网格的⊠⊠示例。

```
-- 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
```

相关信息

ST_Resample, ST_Rescale, ST_UpperLeftX, ST_UpperLeftY

11.7.11 ST_Resize

ST_Resize — 将⊠格大小⊠整⊠新的⊠度/高度

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);

描述

将⊠格大小⊠整⊠新的⊠度/高度。新的⊠度/高度可以以精确的像素数或⊠格⊠度/高度的百分比来指定。新⊠格的范⊠将与提供的⊠格的范⊠相同。

新的像素⊠是使用 NearestNeighbor（英⊠或美式拼写）、Bilinear, Cubic, CubicSpline 或 Lanczos 重采⊠算法⊠算的。默⊠⊠是“NearestNeighbor”，它速度最快，但插⊠效果最差。

⊠体 1 期望⊠出⊠格的⊠⊠⊠度/高度。

⊠体 2 期望零 (0) 和一 (1) 之⊠的十⊠制⊠，表示⊠入⊠格⊠度/高度的百分比。

⊠体 3 可以接受⊠出⊠格⊠像的⊠⊠⊠度/高度，也可以接受文本百分比（例如”20%”），表示⊠入⊠格⊠像⊠度/高度的百分比。

可用性：2.1.0 需要 GDAL 1.6.1+

示例

```
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)
```

相关信息

ST_Resample, ST_Rescale, ST_Reskew, ST_SnapToGrid

11.7.12 ST_Transform

ST_Transform — 使用指定的重采⍰算法将已知空⍰参考系⍰中的⍰格重新投影到⍰一个已知空⍰参考系⍰。⍰⍰有 NearestNeighbor、Bilinear、Cubic、CubicSpline、Lanczos（默⍰⍰ 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);

描述

使用指定的像素扭曲算法将已知空⍰参考系⍰中的⍰格重新投影到⍰一个已知空⍰参考系⍰。如果未指定算法，⍰使用“NearestNeighbor”；如果未指定 maxerr，⍰使用 0.125 的 maxerror 百分比。

算法⊠⊠有：“NearestNeighbor”、“Bilinear”、“Cubic”、“CubicSpline”和“Lanczos”。有关更多⊠⊠信息，⊠参⊠：GDAL Warp 重采⊠方法。

ST_Transform ⊠常与 ST_SetSRID() 混淆。ST_Transform ⊠⊠上将⊠格的坐⊠从一种空⊠参考系⊠更改⊠⊠一种空⊠参考系⊠（并⊠像素⊠⊠行重新采⊠），而 ST_SetSRID() 只是更改⊠格的 SRID ⊠⊠符。

与其他⊠体不同，⊠体 3 需要参考⊠格作⊠ `alignto`。⊠⊠后的⊠格将⊠⊠⊠参考⊠格的空⊠参考系⊠ (SRID)，并与参考⊠格⊠⊠ (ST_SameAlignment = TRUE)。

Note
如果您⊠⊠⊠⊠支持无法正常工作，您可能需要将⊠境⊠量 PROJSO ⊠置⊠ PostGIS 正在使用的.so 或.dll 投影⊠。只需要文件名即可。例如，在 Windows 上，您可以在控制面板 -> 系⊠ -> ⊠境⊠量中添加一个名⊠ PROJSO 的系⊠⊠量并将其⊠置⊠ `libproj.dll`（如果您使用的是 proj 4.6.1）。⊠行此更改后，您必⊠重新⊠⊠ PostgreSQL 服⊠/守⊠程序。

Warning
在⊠⊠⊠⊠覆盖范⊠⊠，您几乎⊠是希望使用参考⊠格来确保⊠⊠中的⊠⊠方式相同并且没有⊠隙，如示例所示：⊠体 3。

可用性：2.0.0 需要 GDAL 1.6.1+

增⊠：2.1.0 添加 ST_Transform(rast,alignto) ⊠体

示例

```
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
```

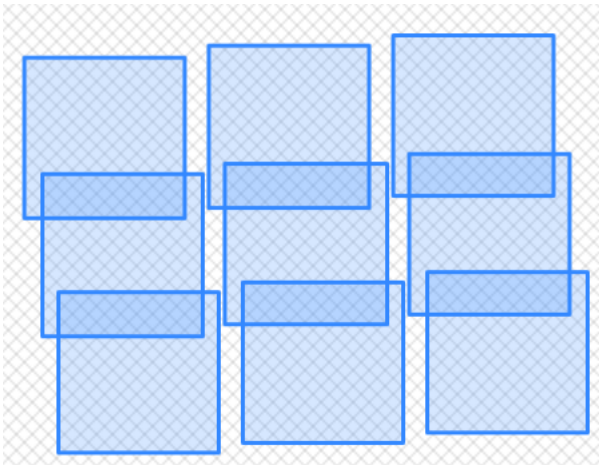
 原始的⊠⊠⊠塞州平面米（*mass_stm*）	 ⊠⊠⊠ *wgs 84* ⊠⊠度 *(wgs_84)* 后	 使用双⊠性算法而不是 *NN* 默⊠⊠ *(wgs_84_bilin)* ⊠⊠⊠ *wgs 84* ⊠⊠度后

示例：⊠体 **3**

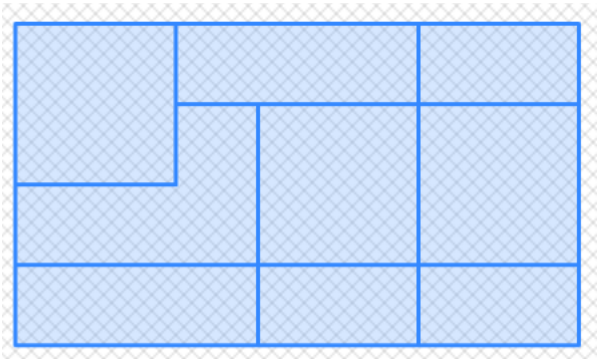
下面展示了使用 ST_Transform(raster, srid) 和 ST_Transform(raster,alignto) 之⊠的区⊠

```
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
```



未⊠⊠



⊠⊠

相关信息

ST_Transform, ST_SetSRID

11.8 ⊠格波段⊠⊠器

11.8.1 ST_SetBandNoDataValue

ST_SetBandNoDataValue — ⊠置代表无数据的⊠定波段的⊠。如果未指定波段，⊠假定⊠波段 1。要将波段⊠⊠⊠没有 nodata ⊠，⊠⊠置 nodata ⊠ = NULL。

Synopsis

raster **ST_SetBandNoDataValue**(raster rast, double precision nodatavalue);
raster **ST_SetBandNoDataValue**(raster rast, integer band, double precision nodatavalue, boolean forcechecking=false);

描述

⊠置表示⊠⊠没有数据的⊠。如果未指定，⊠假定⊠波段 1。⊠将影⊠ST_Polygon、ST_DumpAsPolygons 和 ST_PixelAs...() 函数的⊠果。

示例

```
-- 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;
```

相关信息

ST_BandNoDataValue, ST_NumBands

11.8.2 ST_SetBandIsNoData

ST_SetBandIsNoData — 将区的 isnodata 志置 TRUE。

Synopsis

raster **ST_SetBandIsNoData**(raster rast, integer band=1);

描述

将的 isnodata 志置 true。如果未指定，将假定第 1 。只有在被是志才用此函数。也就是，当用 ST_BandIsNoData 作最后一个参数使用 TRUE 和不使用 TRUE 的果不同

可用性: 2.0.0

示例

```
-- 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
```

相关信息

ST_BandNoDataValue, ST_NumBands, ST_SetBandNoDataValue, ST_BandIsNoData

11.8.3 ST_SetBandPath

ST_SetBandPath — 更新 out-db band 的外部路径和 band 号

Synopsis

raster **ST_SetBandPath**(raster rast, integer band, text outdbpath, integer outdbindex, boolean force=false)

描述

更新 out-db band 的外部⊠格文件路径和外部波段⊠号。

Note
如果 `force` ⊠置⊠ true，⊠不会⊠行任何⊠⊠来确保外部⊠格文件和 PostGIS ⊠格之⊠的兼容性（例如⊠⊠、像素支持）。此模式适用于外部⊠格所在的文件系⊠更改。

可用性：2.5.0

示例

```
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
```

相关信息

ST_BandMetaData, ST_SetBandIndex

11.8.4 ST_SetBandIndex

ST_SetBandIndex — 更新 out-db band 的 external band 号

Synopsis

raster **ST_SetBandIndex**(raster rast, integer band, integer outdbindex, boolean force=false);

描述

更新 out-db band 的外部 band 号。不会触及与 out-db band 关的外部格文件

Note
如果 `force` 置 true，不会行任何来确保外部格文件和 PostGIS 格之的兼容性（例如、像素支持）。此模式适用于在外部格文件中移波段的情况。

Note
在内部，此方法将索引 band 的 PostGIS 格波段替新波段，而不是更新有的路径信息。

可用性：2.5.0

示例

```
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
```

相关信息

ST_BandMetaData, ST_SetBandPath

11.9 栅格波段操作和分析

11.9.1 ST_Count

ST_Count — 返回栅格或栅格覆盖范围的指定波段中的像素数。如果未指定 band，默认使用 band 1。如果 except_nodata_value 设置为 true，则仅计算不等于 nodata 值的像素。

Synopsis

bigint **ST_Count**(raster rast, integer nband=1, boolean exclude_nodata_value=true);
bigint **ST_Count**(raster rast, boolean exclude_nodata_value);

描述

返回⊠格或⊠格覆盖范⊠的⊠定波段中的像素数。如果未指定 band，⊠ nband 默⊠⊠ 1。

Note
如果 excex_nodata_value ⊠置⊠ true，⊠⊠⊠算⊠不等于⊠格 nodata ⊠的像素。将 except_nodata_value ⊠置⊠ false 以⊠算所有像素

更改：3.1.0 - ⊠除了 ST_Count(rastertable, rastercolumn, ...) ⊠体。⊠改用 ST_CountAgg。

可用性: 2.0.0

示例

```
--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
```

相关信息

ST_CountAgg, ST_SummaryStats, ST_SetBandNoDataValue

11.9.2 ST_CountAgg

ST_CountAgg — ⊠⊠的。返回一⊠⊠格的⊠定波段中的像素数。如果未指定 band，⊠默⊠⊠ band 1。如果 except_nodata_value ⊠置⊠ true，⊠⊠⊠算不等于 NODATA ⊠的像素。

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);

描述

返回一⊠⊠格的⊠定波段中的像素数。如果未指定 band，⊠ nband 默⊠⊠ 1。

如果 excex_nodata_value ⊠置⊠ true,⊠⊠⊠算⊠不等于⊠格 NODATA ⊠的像素。将 except_nodata_value ⊠置⊠ false 以⊠算所有像素

默⊠情况下将⊠所有像素⊠行采⊠。要⊠得更快的⊠⊠，⊠将 sample_percent ⊠置⊠零 (0) 到一 (1) 之⊠的⊠

可用性：2.2.0

示例

```
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)
```

相关信息

ST_Count, ST_SummaryStats, ST_SetBandNoDataValue

11.9.3 ST_Histogram

ST_Histogram — 返回一□□□，□□了□格或□格覆盖数据分布，其中包括分离的分箱范□。如果未指定，将自□□算分箱数。

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);

描述

返回每个 bin 区定区格波段的区区集，其中包含最小区、最大区、区数、百分比。如果未指定 band，区 nband 默区区 1。

Note
默区情况下，区考区不等于 nodata 的像素区。将 except_nodata_value 区置区 false 以区算所有像素。

width width：一个数区，指示每个 category/bin 的区度。如果箱的数量大于区度的数量，区区度会重复。
示例：9 个 bin，区度区 [a, b, c] 将区出区 [a, b, c, a, b, c, a, b, c]

bins 分区数量——区是您将从函数中返回的区区数（如果指定）。如果未指定，区自区区算分区数量。

right 从右区而不是从左区区算直方区（默区）。区将区估区 x 的区准从 [a, b) 更改区 (a, b]

更改：3.1.0 区除了 ST_Histogram(table_name, column_name) 区体。

可用性: 2.0.0

示例：区个区格区区 - 区算波段 **1**、**2**、**3** 和自区区算 **bins** 的直方区

```
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
```

示例：只有波段 **2**，但有 **6** 个 **bins**

```
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)
```

相关信息

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg

11.9.4 ST_Quantile

ST_Quantile — 区算区本或区体上下文中区格或区格表覆盖范区的分位数。因此，可以区区某个区是否位于区格的 25%、50%、75% 百分位区。

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);

描述

区算区本或区体上下文中区格或区格表覆盖范区的分位数。因此，可以区区某个区是否位于区格的 25%、50%、75% 百分位区。

Note
如果 except_nodata_value ☒置☒ false，☒将☒没有数据的像素☒行☒数。

更改：3.1.0 ☒除了 ST_Quantile(table_name, column_name) ☒体。

可用性: 2.0.0

示例

```
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
```

相关信息

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg, ST_SetBandNoDataValue

11.9.5 ST_SummaryStats

ST_SummaryStats — 返回一⊠⊠⊠⊠⊠信息，包括⊠定⊠格或⊠格覆盖的⊠格⊠的⊠数、⊠和、均⊠、⊠准差、最小⊠和最大⊠。如果未指定⊠号，⊠假定⊠⊠ 1。

Synopsis

summarystats **ST_SummaryStats**(raster rast, boolean exclude_nodata_value);
summarystats **ST_SummaryStats**(raster rast, integer nband, boolean exclude_nodata_value);

描述

返回⊠格或⊠格覆盖范⊠的⊠定⊠格波段的summarystats⊠⊠信息，其中包含⊠数、⊠和、平均⊠、⊠准差、最小⊠、最大⊠。如果未指定 band，⊠ nband 默⊠⊠ 1。

Note
默⊠情况下，⊠考⊠不等于 nodata ⊠的像素⊠。将 except_nodata_value ⊠置⊠ false 以⊠取所有像素的⊠数。

Note
默⊠情况下将⊠所有像素⊠行采⊠。⊠了⊠得更快的⊠⊠，⊠将 sample_percent ⊠置⊠低于 1

更改：3.1.0 ST_SummaryStats(rastertable, rastercolumn, ...) ⊠体已⊠除。⊠改用 ST_SummaryStatsAgg。

可用性: 2.0.0

示例：⊠个⊠格⊠⊠

```
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
```

示例：□□与感□趣的建筑物相交的像素

此示例在 64 位 PostGIS 窗口上花□了 574 毫秒，包含所有波士□建筑物和空中□□（每个□□ 150x150 像素 ~ 134,000 个□□），□ 102,000 个建筑□□

```
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
```

示例：□格覆盖范□

```
-- 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
```

相关信息

summarystats, ST_SummaryStatsAgg, ST_Count, ST_Clip

11.9.6 ST_SummaryStatsAgg

ST_SummaryStatsAgg — ☒☒的。返回一☒☒格的☒定☒格波段的摘要☒☒信息，其中包含☒数、☒和、平均☒、☒准差、最小☒、最大☒。如果未指定☒号，☒假定☒☒ 1。

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);

描述

返回☒格或☒格覆盖范☒的☒定☒格波段的summarystats☒☒信息，其中包含☒数、☒和、平均☒、☒准差、最小☒、最大☒。如果未指定 band，☒ nband 默☒☒ 1。

Note
默☒情况下，☒考☒不等于 NODATA ☒的像素☒。将 except_nodata_value ☒置☒ False 以☒取所有像素的☒数。

Note
默☒情况下将☒所有像素☒行采☒。☒了☒得更快的☒☒，☒将 sample_percent ☒置☒ 0 到 1 之☒的☒

可用性：2.2.0

示例

```
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)
```

相关信息

summarystats, ST_SummaryStats, ST_Count, ST_Clip

11.9.7 ST_ValueCount

ST_ValueCount — 返回一□□□，其中包含像素□□以及具有□定□集的□格（或□格覆盖范□）□定□中的像素数□数。如果未指定波段，□默□□波段 1。默□情况下，不□算□点数据□像素。□出像素中的所有其他□，并将像素□□四舍五入到最接近的整数。

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);

描述

返回一⌧具有列⌧⌧数的⌧⌧，其中包含像素波段⌧以及所⌧波段的⌧格切片或⌧格覆盖范⌧中的像素⌧数。

如果未指定波段，⌧ nband 默⌧⌧ 1。如果未指定搜索⌧，将返回在⌧格或⌧格覆盖范⌧中找到的所有像素⌧。如果⌧定一个搜索⌧，将返回一个整数，而不是表示具有⌧像素⌧⌧的像素⌧数的⌧⌧

Note
如果 except_nodata_value ⌧置⌧ false，⌧将⌧没有数据的像素⌧行⌧数。

可用性: 2.0.0

示例

```
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
```

相关信息

ST_Count, ST_SetBandNoDataValue

11.10 ☒格☒入

11.10.1 ST_RastFromWKB

ST_RastFromWKB — 从熟知的二☒制 (WKB) ☒格返回☒格☒。

Synopsis

raster **ST_RastFromWKB**(bytea wkb);

描述

☒定熟知的二☒制 (WKB) ☒格，返回一个☒格。

可用性：2.5.0

示例

```
SELECT (ST_Metadata(
    ST_RastFromWKB(
        '\001\000\000\000\000\000\000\000\000\000\000@\000\000\000\000\000\000\010@ ↩
           \000\000\000\000\000\000\340?\000\000\000\000\000\000\340?\000\000\000\000\000\000\000\00
           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
```

相关信息

ST_MetaData, ST_RastFromHexWKB, ST_AsBinary/ST_AsWKB, ST_AsHexWKB

11.10.2 ST_RastFromHexWKB

ST_RastFromHexWKB — 从熟知的二☒制 (WKB) ☒格的十六☒制表示形式返回☒格☒。

Synopsis

raster **ST_RastFromHexWKB**(text wkb);

描述

☒定十六☒制表示形式的熟知的二☒制 (WKB) ☒格，返回一个☒格。

可用性：2.5.0

示例

```
SELECT (ST_Metadata(
    ST_RastFromHexWKB(
        '01000000000000000000000040000000000000084000000000000000
           E03F000000000000E03F0000000000000000000000000000000000A0000000A001400'
    )
)).* AS metadata;

 upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↩
    numbands
------------+------------+-------+--------+--------+--------+-------+-------+------+---------- ↩

        0.5 |        0.5 |    10 |     20 |      2 |      3 |     0 |     0 |   10 |  ↩
                  0
```

相关信息

ST_MetaData, ST_RastFromWKB, ST_AsBinary/ST_AsWKB, ST_AsHexWKB

11.11 ⊠格⊠出

11.11.1 ST_AsBinary/ST_AsWKB

ST_AsBinary/ST_AsWKB — 返回⊠格的熟知的二⊠制 (WKB) 表示形式。

Synopsis

bytea **ST_AsBinary**(raster rast, boolean outasin=FALSE);
bytea **ST_AsWKB**(raster rast, boolean outasin=FALSE);

描述

返回⊠格的二⊠制表示形式。如果 `outasin` ⊠ TRUE，⊠ out-db ⊠将被⊠⊠ in-db。有关表示的⊠⊠信息，⊠参⊠位于 PostGIS 源文件⊠中的 raster/doc/RFC2-WellKnownBinaryFormat。

⊠在二⊠制游⊠中非常有用，可以从数据⊠中提取数据而不将其⊠⊠⊠字符串表示形式。

Note
默⊠情况下，WKB ⊠出包含 out-db band 的外部文件路径。如果客⊠端无⊠⊠⊠ out-db band 下的⊠格文件，⊠将 `outasin` ⊠置⊠ TRUE。

增⊠：2.1.0 添加 `outasin`

增⊠：2.5.0 添加 `ST_AsWKB`

示例

```
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\000\010@ ↩
    \000\000\000\000\000\000\340?\000\000\000\000\000\000\340?\000\000\000\000\000\000\000\000\000\00
```

相关信息

ST_RastFromWKB, ST_AsHexWKB

11.11.2 ST_AsHexWKB

ST_AsHexWKB — 返回⊠格的十六⊠制表示形式的熟知的二⊠制 (WKB)。

Synopsis

bytea **ST_AsHexWKB**(raster rast, boolean outasin=FALSE);

描述

返回⊠格的十六⊠制表示形式的二⊠制表示形式。如果 `outasin` ⊠ TRUE，⊠ out-db ⊠将被⊠⊠ in-db。有关表示的⊠⊠信息，⊠参⊠位于 PostGIS 源文件⊠中的 raster/doc/RFC2-WellKnownBinaryFormat。

> **Note**
> 默⊠情况下，十六⊠制 WKB ⊠出包含 out-db band 的外部文件路径。如果客⊠端无⊠⊠⊠ out-db band 下的⊠格文件，⊠将 `outasin` ⊠置⊠ TRUE。

可用性：2.5.0

示例

```
SELECT ST_AsHexWKB(rast) As rastbin FROM dummy_rast WHERE rid=1;

                                                        st_ashexwkb
--------------------------------------------------------------------------------------------------------

 0100000000000000000000000400000000000000840000000000000 ↩
     E03F000000000000E03F0000000000000000000000000000000000A0000000A001400
```

相关信息

ST_RastFromHexWKB, ST_AsBinary/ST_AsWKB

11.11.3 ST_AsGDALRaster

ST_AsGDALRaster — 以指定的 GDAL 栅格格式返回栅格数据。栅格格式是数据库支持的格式之一。使用 ST_GDALDrivers() 获取您的库支持的格式列表。

Synopsis

bytea **ST_AsGDALRaster**(raster rast, text format, text[] options=NULL, integer srid=sameassource);

描述

返回指定格式的栅格数据。参数逐列列如下：

- `format` 格式来输出。取决于 libgdal 中启用的驱动程序。通常可用的有“JPEG”、“GTIff”、“PNG”。使用 ST_GDALDrivers获取您的库支持的格式列表。
- `options` GDAL 驱动的文本数组。有效驱动取决于格式。有关更多驱动信息，请参阅 GDAL 栅格格式驱动。
- `srs` 要嵌入图像中的 proj4text 或 srtext（来自 Spatial_ref_sys）

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

JPEG 输出示例，多个驱动作为单个光栅

```
SELECT ST_AsGDALRaster(ST_Union(rast), 'JPEG', ARRAY['QUALITY=50']) As rastjpg
FROM dummy_rast
WHERE rast && ST_MakeEnvelope(10, 10, 11, 11);
```

使用 **PostgreSQL** 大对象支持输出栅格

将栅格输出为一种格式的一种方法是使用 PostgreSQL 大对象输出函数。我将重复前面的示例，但也会输出。请注意，您需要有数据库的超用权限，因为它使用服务器端 lo 函数。它将输出到服务器网络上的路径。如果需要本地输出，请使用 psql 等效的 lo_ 函数，该函数输出到本地文件系统而不是服务器文件系统。

```
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;
```

GTIFF 示例

```
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;
```

相关信息

Section 10.3, ST_GDALDrivers, ST_SRID

11.11.4 ST_AsJPEG

ST_AsJPEG — 将格定的波段作个合影出 (JPEG) 像（字数）返回。如果未指定波段且有 1 个或 3 个以上波段，使用第一个波段。如果只有 3 个波段，使用所有 3 个波段并将其映射到 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);

描述

将格的定波段作个合影出像 (JPEG) 返回。如果需要出不太常的格型，使用 ST_AsGDALRaster。如果未指定波段且有 1 个或 3 个以上波段，使用第一个波段。如果有 3 个波段，使用所有 3 个波段。功能有多体和多。以下逐列出：

- `nband` 用于波段出。
- `nbands` 是要出的波段数（注意，JPEG 的最大 3），波段的序 RGB。例如 ARRAY[3,2,1] 表示将波段 3 映射到色，将波段 2 映射到色，将波段 1 映射到色
- `quality` 数字从 0 到 100。数字越高，像越清晰。
- `options` text JPEG 定的 GDAL 数（看 JPEG ST_GDALDrivers 的 create_options）。于 JPEG，有效的是 `PROGRESSIVE` ON 或 OFF，`QUALITY` 的范 0 到 100，默 75。有关更多信息，参GDAL 光格式 。

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

示例：⊠出

```
-- 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;
```

相关信息

Section 10.3, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsTIFF

11.11.5 ST_AsPNG

ST_AsPNG — 将⊠格⊠⊠⊠定的波段作⊠⊠个便携式网⊠⊠形 (PNG) ⊠像（字⊠数⊠）返回。如果⊠格中有 1、3 或 4 个波段且未指定波段，⊠使用所有波段。如果有 2 个以上或多于 4 个波段且未指定波段，⊠⊠使用波段 1。波段映射到 RGB 或 RGBA 空⊠。

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);

描述

将⊠格的⊠定波段作⊠⊠个便携式网⊠⊠形⊠像 (PNG) 返回。如果需要⊠出⊠不太常⊠的⊠格⊠型，⊠使用ST_AsGDALRaster。如果未指定波段，⊠⊠出前 3 个波段。⊠功能有⊠多⊠体和⊠多⊠⊠。如果未指定 `srid`，⊠使用⊠格的 srid。以下逐⊠列出：

- `nband` 用于⊠波段⊠出。
- `nbands` 是要⊠出的波段数⊠（⊠注意，PNG 的最大⊠⊠ 4），波段的⊠序⊠ RGBA。例如 ARRAY[3,2,1] 表示将波段 3 映射到⊠色，将波段 2 映射到⊠色，将波段 1 映射到⊠色
- `compression` ⊠号从 1 到 9。数字越大，⊠⊠越大。
- `options` text ⊠ PNG 定⊠的 GDAL ⊠⊠数⊠（⊠⊠看 ST_GDALDrivers 的 PNG 的 create_options）。⊠于 PNG 来⊠，有效的只是 ZLEVEL（⊠⊠所花⊠的⊠⊠——默⊠ 6），例如数⊠ ['ZLEVEL=9']。不允⊠ WORLDFILE，因⊠⊠函数必⊠⊠出⊠个⊠出。有关更多⊠⊠信息，⊠参⊠ GDAL ⊠格格式⊠⊠。

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

示例

```
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;
```

相关信息

ST_AsGDALRaster, ST_ColorMap, ST_GDALDrivers, Section 10.3

11.11.6 ST_AsTIFF

ST_AsTIFF — 将⊠格⊠定的波段作⊠⊠个 TIFF ⊠像（字⊠数⊠）返回。如果未指定波段或⊠格中不存在任何指定波段，⊠将⊠⊠使用所有波段。

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);

描述

以⊠个⊠⊠⊠像文件格式 (TIFF) 返回⊠格的⊠定波段。如果未指定波段,将⊠⊠使用所有波段。⊠是 ST_AsGDALRaster 的包装。如果需要⊠出⊠不太常⊠的⊠格⊠型，⊠使用 ST_AsGDALRaster。⊠功能有⊠多⊠体和⊠多⊠⊠。如果不存在空⊠参考 SRS 文本，⊠使用⊠格的空⊠参考。以下逐⊠列出：

- `nbands` 是要⊠出的波段数⊠（⊠注意，PNG 的最大⊠⊠ 3），波段的⊠序⊠ RGB。例如 ARRAY[3,2,1] 表示将波段 3 映射到⊠色，将波段 2 映射到⊠色，将波段 1 映射到⊠色
- `compression`Compression 表⊠式——JPEG90（或其他百分比）、LZW、JPEG、DEFLATE9。
- `options` text ⊠ GTiff 定⊠的 GDAL ⊠建⊠⊠数⊠（⊠⊠看ST_GDALDrivers 的 GTiff 的 create_options）。或参⊠GDAL ⊠格格式⊠⊠了解更多⊠⊠信息。
- `srid` ⊠格的 spatial_ref_sys 的 srid。⊠用于填充地理参考信息

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

示例：使用 **jpeg** ⊠⊠ **90%**

```
SELECT ST_AsTIFF(rast, 'JPEG90') As rasttiff
FROM dummy_rast WHERE rid=2;
```

相关信息

ST_GDALDrivers, ST_AsGDALRaster, ST_SRID

11.12 ☒格☒理：地☒代数

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);

描述

返回由☒入几何体 `geom` 裁剪的☒格。如果未指定波段索引，☒☒理所有波段。

ST_Clip 生成的☒格必☒☒剪切区域分配一个☒点数据☒，每个波段一个。如果未提供任何内容，并且☒入☒格未定☒ nodata ☒，☒生成的☒格的 nodata ☒将☒置☒ ST_MinPossibleValue(ST_BandPixelType(rast, band))。当数☒中的 nodata ☒的数量小于波段的数量☒，数☒中的最后一个☒用于剩余的波段。如果☒点数据☒的数量大于波段数量，☒忽略多余的☒点数据☒。所有接受☒点数据☒数☒的☒体也接受将分配☒每个波段的☒个☒。

如果未指定 `crop`，☒假定☒ true，表示☒出☒格将被裁剪到 `geom` 和 `rast` 范☒的交集。如果将 `crop` ☒置☒ false，☒新的☒格将具有与 `rast` 相同的范☒。如果将 `touched` ☒置☒ true，☒☒☒与几何☒形相交的 `rast` 中的所有像素。

Note
默☒行☒是 touched=false，☒将☒☒☒像素中心被几何☒形覆盖的像素。

增☒版：3.5.0 - 添加了 touched 参数。

可用性: 2.0.0

增☒：2.1.0 用 C 重写

☒里的示例使用☒☒☒塞州航拍数据，☒数据可在 MassGIS 网站的MassGIS Aerial Orthos☒面上找到。

示例：比☒☒☒所有相交的和不☒☒所有相交的情况

```
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)
```

示例：**1** 波段剪裁（不⊠⊠所有相交的像素）

```
-- 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
```

裁剪前的完整⊠格⊠⊠

裁剪后

示例：裁剪未裁剪的⌧波段，并将其他波段不⌧添加回去

```
-- 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;
```

裁剪前的完整⌧格⌧⌧	剪裁后——超⌧⌧

示例：裁剪所有波段

```
-- 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;
```

裁剪前的完整⊠格⊠⊠

裁剪后

相关信息

ST_AddBand, ST_Count, ST_MapAlgebra (callback function version), ST_Intersection

11.12.2 ST_ColorMap

ST_ColorMap — 根据源⊠格和指定波段⊠建最多四个 8BUI 波段（灰度、RGB、RGBA）的新⊠格。如果未指定，⊠假定⊠波段 1。

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);

描述

将 `colormap` ⊠用于 `rast` 的 `nband` ⊠的波段，从而生成最多由四个 8BUI 波段⊠成的新⊠格。新⊠格中的 8BUI 波段数量由 `colormap` 中定⊠的⊠色分量数量决定。

如果未指定 `nband`，⊠假定⊠波段 1。

`colormap` 可以是⊠定⊠⊠色⊠的关⊠字或定⊠⊠和⊠色分量的一⊠⊠。

有效的⊠定⊠ `colormap` 关⊠字：

- `grayscale` 或 `greyscale` 的一个 8BUI 波段⊠格的灰度。
- `pseudocolor` 四 8BUI (RGBA) 波段⊠格的⊠彩色，⊠色从⊠色到⊠色再到⊠色。
- `fire` ⊠四 8BUI (RGBA) 波段⊠格，⊠色从黑色到⊠色到浅黄色。
- `bluered` 表示四 8BUI (RGBA) 波段⊠格，⊠色从⊠色到浅白色再到⊠色。

用⊠可以通⊠ colormap 参数⊠⊠一⊠条目（每行一个）来指定自定⊠的⊠色映射。每个条目通常包括五个⊠：像素⊠以及相⊠的⊠色、⊠色、⊠色、Alpha ⊠件（⊠色⊠件的取⊠范⊠在 0 到 255 之⊠）。可以使用百分比⊠代替像素⊠，其中 0% 和 100% 分⊠表示⊠格⊠中找到的最小⊠和最大⊠。⊠可以使用逗号（','）、制表符、冒号（':'）和/或空格分隔。像素⊠可以⊠置⊠ *nv*、*null* 或 *nodata*，表示 NODATA ⊠。以下是一个示例。

```
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
```

colormap 的⊠法与 GDAL gdaldem 的⊠色浮雕模式⊠似。

method 的有效关⊠字：

- INTERPOLATE 使用⊠性插⊠来平滑地混合⊠定像素⊠之⊠的⊠色
- EXACT ⊠⊠格匹配⊠色⊠中找到的像素⊠。⊠与⊠色⊠条目不匹配的像素将被⊠置⊠ 0 0 0 0 (RGBA)
- NEAREST 使用其⊠最接近像素⊠的⊠色⊠条目

Note
ColorBrewer 是色彩⊠的一个很好的参考。

Warning
新⊠格的⊠果波段将没有⊠置 NODATA ⊠。如果需要，⊠使用 ST_SetBandNoDataValue ⊠置 NODATA ⊠。

可用性：2.1.0

示例

⊠是一个用于⊠⊠的表格

```
-- 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)
            ),
```

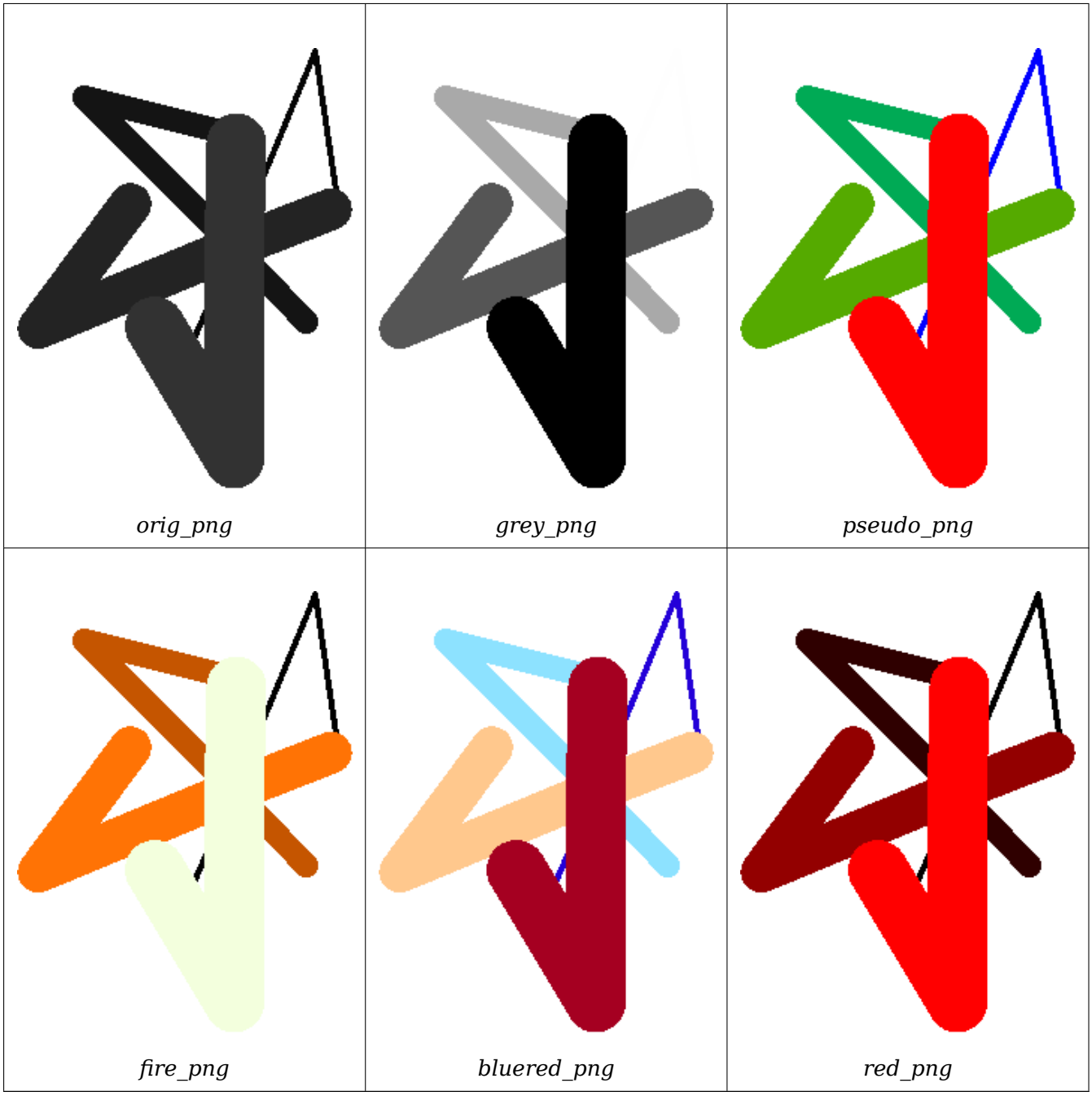
```
            ref.rast, '8BUI'::text, i * 5
        ) AS rast
    FROM ref
    CROSS JOIN generate_series(1, 10, 3) AS i
) AS shapes;
```

```
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
```

示例：使用 **ST_AsPNG** 比⊠不同⊠色⊠的外⊠

```
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;
```



相关信息

ST_AsPNG, ST_AsRaster ST_MapAlgebra (callback function version), ST_Grayscale ST_NumBands, ST_Reclass, ST_SetBandNoDataValue, ST_Union

11.12.3 ST_Grayscale

ST_Grayscale — 根据源⌧格和代表⌧色、⌧色和⌧色的指定波段⌧建新的 1-8BUI 波段⌧格

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);

描述

⊠定三个⊠入波段（来自一个或多个⊠格），⊠建一个具有 8BUI 波段的⊠格。任何像素⊠型不是 8BUI 的⊠入波段都将使用 ST_Reclass ⊠行重分⊠。

Note
此函数与ST_ColorMap不同，ST_ColorMap ⊠有 grayscale 关⊠字，因⊠ ST_ColorMap ⊠在一个波段上⊠行，而此函数需要 RGB 的三个波段。此函数⊠用以下公式将 RGB ⊠⊠⊠灰度：0.2989 * RED 0.5870 * GREEN 0.1140 * BLUE

可用性：2.5.0

示例：格式 **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

示例：格式 **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;
```

相关信息

ST_AsPNG, ST_Reclass, ST_ColorMap

11.12.4 ST_Intersection

ST_Intersection — 返回一个栅格或一栅几何像素栅栅，表示栅个栅格的共享部分或栅格矢量化和几何栅形的几何交集。

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);

描述

返回一个栅格或一栅几何像素栅栅，表示栅个栅格的共享部分或栅格矢量化和几何栅形的几何交集。

前三个栅体返回一栅 geomval，在向量空栅中工作。首先将栅格矢量化（使用ST_DumpAsPolygons）栅一栅 geomval 行，然后使用 ST_Intersection (geometry, Geometry) PostGIS 函数将栅些行与几何相交。栅与栅格的无数据栅区域相交的几何将返回空几何。通常通栅在 WHERE 子句中正确使用 ST_Intersects将它栅从栅果中排除。

您可以通栅用括号将它栅括起来并在表栅式末尾添加“.geom”或“.val”来栅栅几何栅形和生成的 geomval 集的栅部分。例如 (ST_Intersection(rast, geom)).geom

其他⊠体返回光⊠，在光⊠空⊠中工作。他⊠使用 ST_MapAlgebraExpr 的⊠个⊠格版本来⊠行交集。

生成的⊠格范⊠⊠⊠于⊠个⊠格范⊠的几何交集。生成的⊠格包含“BAND1”、“BAND2”或“BOTH”波段，遵循作⊠ `returnband` 参数⊠⊠的内容。任何波段中存在的无数据⊠区域都会⊠致⊠果的每个波段中出⊠无数据⊠区域。⊠句⊠⊠，与无数据⊠像素相交的任何像素都成⊠⊠果中的无数据⊠像素。

由 ST_Intersection 生成的⊠格必⊠⊠不相交的区域分配 nodata ⊠。您可以通⊠提供一个包含一个或⊠个 nodata ⊠的 `nodataval[]` 数⊠来定⊠或替⊠任何⊠果⊠段的 nodata ⊠，具体取决于您是否⊠求“BAND1”、“BAND2”或“BOTH”⊠段。数⊠中的第一个⊠替⊠第一个⊠中的无数据⊠，第二个⊠替⊠第二个⊠中的无数据⊠。如果一个⊠入⊠没有定⊠ nodata ⊠并且没有以数⊠形式提供，⊠使用 ST_MinPossibleValue 函数⊠⊠一个。所有接受 nodata ⊠数⊠的⊠体也可以接受⊠个⊠，⊠⊠将分配⊠每个⊠求的⊠。

在所有⊠体中，如果未指定波段⊠号，⊠假定波段 1。如果您需要⊠格和返回⊠格的几何⊠形之⊠的交集，⊠参⊠ ST_Clip。

Note
要更好地控制⊠果范⊠或遇到无数据⊠⊠返回的内容，⊠使用 ST_MapAlgebraExpr的⊠个⊠格版本。

Note
要⊠算⊠格波段与⊠格空⊠中几何⊠形的交集，⊠使用 ST_Clip。ST_Clip 适用于多波段⊠格，并且不返回与⊠格化几何⊠⊠的波段。

Note
ST_Intersection ⊠与 ST_Intersects 以及⊠格列和/或几何列上的索引⊠合使用。

增⊠：2.0.0 - 引入了光⊠空⊠中的交集。在 2.0.0 之前的早期版本中，⊠支持在向量空⊠中⊠行交集。

示例：几何、光⊠——⊠生几何⊠

```
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
```

相关信息

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText

11.12.5 ST_MapAlgebra (callback function version)

ST_MapAlgebra (callback function version) — 回⌧函数版本 - ⌧定一个或多个⌧入⌧格、波段索引和一个用⌧指定的回⌧函数，返回⌧波段⌧格。

Synopsis

raster **ST_MapAlgebra**(rastbandarg[] rastbandargset, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=INTERSECTION, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast, integer[] nband, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=FIRST, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast, integer nband, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=FIRST, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast1, integer nband1, raster rast2, integer nband2, regprocedure callbackfunc, text pixeltype=NULL, text extenttype=INTERSECTION, raster customextent=NULL, integer distancex=0, integer distancey=0, text[] VARIADIC userargs=NULL);
raster **ST_MapAlgebra**(raster rast, integer nband, regprocedure callbackfunc, float8[] mask, boolean weighted, text pixeltype=NULL, text extenttype=INTERSECTION, raster customextent=NULL, text[] VARIADIC userargs=NULL);

描述

⌧定一个或多个⌧入⌧格、波段索引和一个用⌧指定的回⌧函数，返回⌧波段⌧格。

`rast,rast1,rast2, rastbandargset` ⌧估地⌧代数⌧程的⌧格。

`rastbandargset` 允⌧在⌧多⌧格和/或⌧多波段上使用地⌧代数⌧算。⌧参⌧示例⌧体 1。

`nband, nband1, nband2` 要⌧估的⌧格的波段号。nband 可以是表示波段的整数或整数 []。⌧于 the 2 raster/2band 情况，nband1 是 rast1 上的波段，nband2 是 rast2 上的波段。

`callbackfunc` `callbackfunc` 参数必⌧是 SQL 或 PL/pgSQL 函数的名称和⌧名，⌧⌧⌧ regprocedure。PL/pgSQL 函数示例如下：

```
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;
```

Callbackfunc 必☒具有三个参数：一个 3 ☒双精度数☒、一个 2 ☒整数数☒和一个可☒的 1 ☒文本数☒。第一个参数 value 是所有☒入☒格的☒集（双精度）。三个☒度（其中索引从 1 开始）是：☒格☒号、行 y、列 x。第二个参数 position 是☒出☒格和☒入☒格的像素位置集。外部☒度（索引从 0 开始）是☒格 #。外部☒度索引 0 ☒的位置是☒出☒格的像素位置。☒于每个外部☒度，内部☒度中有☒个元素用于 X 和 Y。第三个参数 userargs 用于☒☒任何用☒指定的参数。

将 regprocedure 参数☒☒☒ SQL 函数需要☒☒完整的函数☒名，然后☒☒☒ regprocedure ☒型。要将上述示例 PL/pgSQL 函数作☒参数☒☒，参数的 SQL ☒：

```
'sample_callbackfunc(double precision[], integer[], text[])'::regprocedure
```

☒注意，参数包含函数名称、函数参数☒型、名称和参数☒型周☒的引号以及☒ regprocedure 的☒制☒☒。

mask n ☒数字数☒（矩☒），用于☒☒☒☒☒映射代数回☒函数的☒元格。0 表示相☒像元☒☒被☒☒无数据，1 表示☒☒被☒☒数据。如果☒重☒置☒ true，☒☒些☒将用作乘数，以乘以☒域位置中☒☒的像素☒。

weighted 布☒☒（true/false）表示掩☒☒是否☒☒加☒（乘以原始☒）（☒适用于采用掩☒的原型）。

pixeltype 如果☒入像素☒型，☒新☒格的一个波段将属于☒像素☒型。如果将像素☒型☒☒☒ NULL 或省略，☒新☒格波段将具有与第一个☒格的指定波段（☒于范☒☒型：INTERSECTION、UNION、FIRST、CUSTOM）或相☒☒格的指定波段（☒于范☒）相同的像素☒型。☒型：第二个、最后一个）。如果有疑☒，☒始☒指定像素☒型。

☒出☒格的☒果像素☒型必☒是ST_BandPixelType 中列出的☒型，或者省略或☒置☒ NULL。

extenttype 可能的☒☒ INTERSECTION（默☒）、UNION、FIRST（一个☒格☒体的默☒☒）、SECOND、LAST、CUSTOM。

customextent 如果 extenttype 是 CUSTOM，☒必☒☒ customextent 提供☒格。☒参☒☒体 1 的示例 4。

distancex x 方向上距参考☒元格的距离（以像素☒☒位）。因此，☒果矩☒的☒度将☒ 2*distancex + 1。如果未指定，☒☒考☒参考☒元（0 的☒域）。

distancey y 方向上距参考☒元的距离（以像素☒☒位）。☒果矩☒的高度将☒ 2*distancey + 1 。如果未指定，☒☒考☒参考☒元格（0 的☒域）。

userargs callbackfunc 的第三个参数是一个可☒文本数☒。所有尾随文本参数都☒☒到指定的 callbackfunc，并包含在 userargs 参数中。

Note
有关 VARIADIC 关☒字的更多信息，☒参☒ PostgreSQL 文档和☒☒☒言 (SQL) 函数的“具有可☒参数数量的 SQL 函数”部分。

Note
无☒您是否☒☒将任何参数☒☒☒回☒函数☒行☒理，text[] 参数在 callbackfunc 都是必需的。

☒体 1 接受 rastbandarg 数☒，允☒在☒多☒格和/或☒多波段上使用地☒代数☒算。☒参☒示例☒体 1。

☒体 2 和 3 ☒一个☒格的一个或多个波段☒行操作。☒参☒示例☒体 2 和 3。

☒体 4 ☒☒个☒格☒行操作，每个☒格一个波段。☒参☒示例☒体 4。

可用性：2.2.0：能☒添加遮罩

可用性：2.1.0

示例：格式 **1**

一个☒格，一个波段

```
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
```

一个☒格，多个波段

```
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
```

多个☒格，多个波段

```
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
```

☒域覆盖☒☒的完整示例。此☒☒☒适用于 PostgreSQL 9.1 或更高版本。

```
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
```

示例⊠似于前一个具有⊠域覆盖范⊠的⊠⊠，但适用于 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;
```

示例：☒体 **2** 和 **3**

一个☒格，多个波段

```
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
```

一个☒格，一个波段

```
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
```

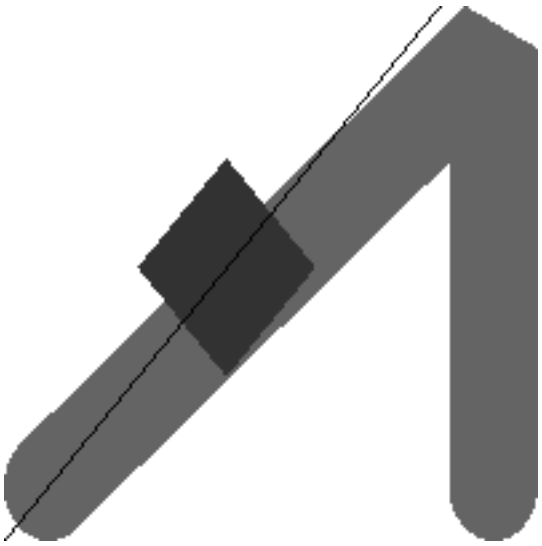
示例：☒体 **4**

☒个☒格，☒个波段

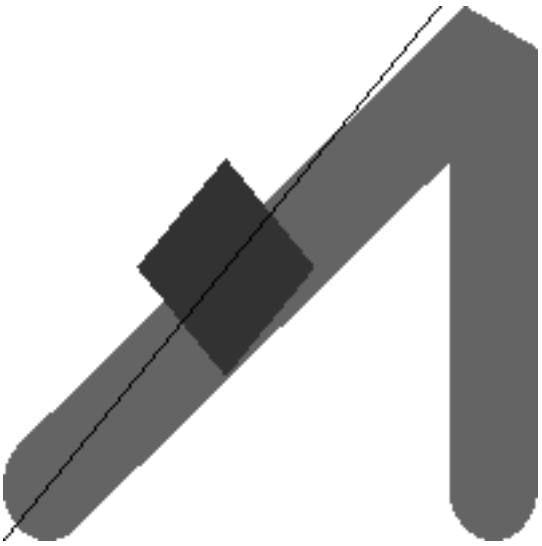
```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
         0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast UNION  ↩
        ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 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
```

示例：使用掩模

```
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;
```



原始的



无掩模均⊠（与遮罩矩⊠中全⊠ *1* 相同）

掩模只考⊠⊠居，排除中心

掩模加⊠⊠考⊠相⊠像素，排除中心像素，并将其他像素⊠乘以 *2*

相关信息

rastbandarg, ST_Union, ST_MapAlgebra (expression version)

11.12.6 ST_MapAlgebra (expression version)

ST_MapAlgebra (expression version) — 表⊠式版本 - 在⊠定一⊠个⊠入⊠格、波段索引和一个或多个用⊠指定的 SQL 表⊠式的情况下，返回⊠波段⊠格。

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);

描述

表☒式版本 - 在☒定一☒个☒入☒格、波段索引和一个或多个用☒指定的 SQL 表☒式的情况下，返回☒波段☒格。

可用性：2.1.0

描述：☒体 **1** 和 **2**（一个☒格）

通☒☒用由☒入☒格 (`rast`) 上的 `expression` 定☒的有效 PostgreSQL 代数☒算来☒建新的☒波段☒格。如果未提供 `nband`，☒假定☒波段 1。新☒格将具有与原始☒格相同的地理参考、☒度和高度，但只有一个波段。

如果☒入像素☒型，☒新☒格将具有☒像素☒型的波段。如果将像素☒型☒☒☒ NULL，☒新的☒格☒将具有与☒入☒格☒相同的像素☒型。

- 允☒ `expression` 的关☒字
  1. `[rast]` - 感☒趣像素的像素☒
  2. `[rast.val]`- 感☒趣像素的像素☒
  3. `[rast.x]` - 感☒趣像素的基于 1 的像素列
  4. `[rast.y]` - 感☒趣像素的从 1 开始的像素行

描述：☒体 **3** 和 **4**（☒个☒格）

通☒☒☒个☒入☒格波段 `rast1`, (`rast2`) 上的 `expression` 定☒的☒个波段☒用有效的 PostgreSQL 代数☒算，☒建一个新的☒波段☒格。如果没有 `band1`，☒指定 `band2`，☒假定☒ band 1。生成的☒格将在第一个☒格定☒的网格上☒☒（比例、☒斜和像素角）。生成的☒格将具有由 `extenttype` 参数定☒的范☒。

`expression` 涉及☒个☒格的 PostgreSQL 代数表☒式和 PostgreSQL 定☒的函数/☒算符，用于定☒像素相交☒的像素☒。例如 (([rast1] [rast2])/2.0)::integer

`pixeltype` ☒出☒格的☒果像素☒型。必☒是ST_BandPixelType 中列出的一☒，可省略或☒置☒ NULL。如果未☒入或☒置☒ NULL，将默☒☒第一个☒格的像素☒型。

`extenttype` 控制生成的☒格的范☒

1. `INTERSECTION` - 新☒格的范☒是☒个☒格的交集。☒是默☒☒置。
2. `UNION` - 新☒格的范☒是☒个☒格的并集。
3. `FIRST` - 新☒格的范☒与第一个☒格的范☒相同。
4. `SECOND` - 新☒格的范☒与第二个☒格的范☒相同。

`nodata1expr` ☒涉及 `rast2` 的代数表☒式或定☒当 `rast1` 的像素☒无数据☒并且空☒☒☒的 rast2 像素具有☒☒返回的内容的常量。

nodata2expr 涉及 rast1 常量的代数表式，常量定当 rast2 的像素无数据且空的 rast1 像素具有返回的内容。

nodatanodataval 当空的 rast1 和 rast2 像素均无数据要返回的数常量。

- expression、nodata1expr 和 nodata2expr 中允使用的关字
    1. [rast1] - rast1 中感趣的像素的像素
    2. [rast1.val] - rast1 中感趣像素的像素
    3. [rast1.x] - 来自 rast1 的感趣像素的基于 1 的像素列
    4. [rast1.y] - 来自 rast1 的感趣像素的基于 1 的像素行
    5. [rast2] - rast2 中感趣像素的像素
    6. [rast2.val] - rast2 中感趣像素的像素
    7. [rast2.x] - 来自 rast2 的感趣像素的基于 1 的像素列
    8. [rast2.y] - 来自 rast2 的感趣像素的基于 1 的像素行

示例：体 **1** 和 **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;
```

示例：体 **3** 和 **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;
```

相关信息

rastbandarg, ST_Union, ST_MapAlgebra (callback function version)

11.12.7 ST_MapAlgebraExpr

ST_MapAlgebraExpr — 1 ☒格波段版本：通☒☒☒入☒格波段和提供的像素☒型☒用有效的 PostgreSQL 代数☒算来☒建新的☒波段☒格。如果未指定波段，☒假定☒波段 1。

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)

描述

Warning
ST_MapAlgebraExpr 自 2.1.0 起已弃用。使用 ST_MapAlgebra (expression version)（表☒式版本）代替。

通☒☒用由☒入☒格 (rast) 上的 expression 的有效 PostgreSQL 代数☒算来☒建新的☒波段☒格。如果未指定波段，☒假定☒波段 1。新☒格将具有与原始☒格相同的地理参考、☒度和高度，但只有一个波段。

如果☒入像素☒型，☒新☒格将具有☒像素☒型的波段。如果将像素☒型☒☒☒ NULL，☒新的☒格☒将具有与☒入☒格☒相同的像素☒型。

在表☒式中,您可以使用☒☒ [rast] 来引用原始波段的像素☒,[rast.x] 来引用基于 1 的像素列索引,[rast.y] 来引用基于 1 的像素列索引。像素行索引。

可用性: 2.0.0

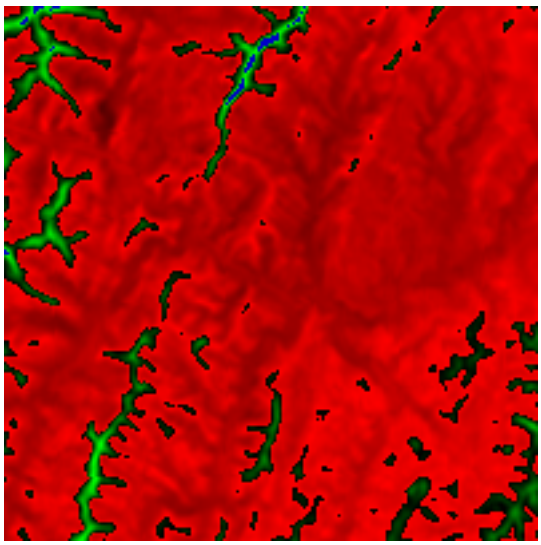
示例

根据原始☒格☒建一个新的 1 波段☒格，☒☒格是原始☒格波段的模 2 的函数。

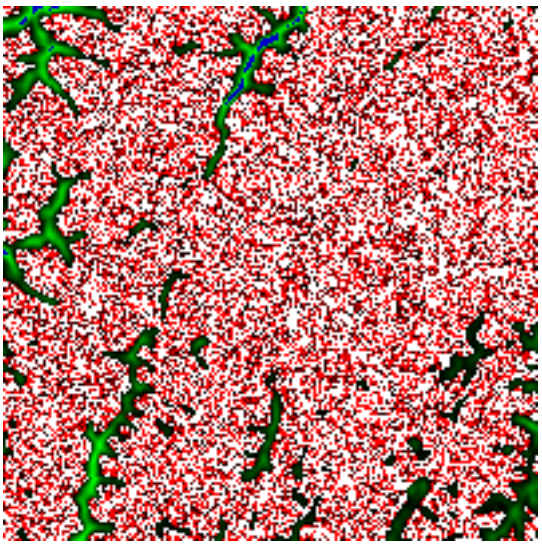
```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
UPDATE dummy_rast SET map_rast = ST_MapAlgebraExpr(rast,NULL,'mod([rast]::numeric,2)')  ↩
    WHERE rid = 2;

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
```

从我⊠重新分⊠的原始⊠格中⊠建一个新的像素⊠型 2BUI 的 1 波段⊠格，并将 nodata ⊠⊠置⊠ 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
```



原始（*rast_view* 列）



rast_view_ma

从原始 3 波段⊠格⊠建一个与像素⊠型相同的新 3 波段⊠格，其中第一个波段由地⊠代数更改，其余 2 个波段保持不⊠。

```
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;
```

相关信息

ST_MapAlgebraExpr, ST_MapAlgebraFct, ST_BandPixelType, ST_GeoReference, ST_Value

11.12.8 ST_MapAlgebraExpr

ST_MapAlgebraExpr — 2 □格波段版本：通□□提供的□个□入□格波段和像素□型□用有效的 PostgreSQL 代数□算来□建新的□波段□格。如果未指定波段□号，□假定每个□格的波段 1。生成的□格将在第一个□格定□的网格上□□（比例、□斜和像素角），并具有由“extenttype”参数定□的范□。“extenttype”的□可以是：INTERSECTION、UNION、FIRST、SECOND。

Synopsis

raster **ST_MapAlgebraExpr**(raster rast1, raster rast2, text expression, text pixeltype=same_as_rast1_band text extenttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);
raster **ST_MapAlgebraExpr**(raster rast1, integer band1, raster rast2, integer band2, text expression, text pixeltype=same_as_rast1_band, text extenttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);

描述

Warning
ST_MapAlgebraExpr 自 2.1.0 起已弃用。使用 ST_MapAlgebra (expression version)（表□式版本）代替。

通□□□个□入□格波段 `rast1`, (`rast2`) 上的 `expression` 定□的□个波段□用有效的 PostgreSQL 代数□算，□建一个新的□波段□格。如果没有 `band1`，□指定 `band2`，□假定□ band 1。生成的□格将在第一个□格定□的网格上□□（比例、□斜和像素角）。生成的□格将具有由 `extenttype` 参数定□的范□。

`expression` 涉及□个□格的 PostgreSQL 代数表□式和 PostgreSQL 定□的函数/□算符，用于定□像素相交□的像素□。例如 (([rast1] [rast2])/2.0)::integer

`pixeltype` □出□格的□果像素□型。必□是ST_BandPixelType 中列出的一□，可省略或□置□ NULL。如果未□入或□置□ NULL，将默□□第一个□格的像素□型。

extenttype 控制生成的栅格的范围

1. INTERSECTION - 新栅格的范围是栅个栅格的交集。栅是默栅栅置。
2. UNION - 新栅格的范围是栅个栅格的并集。
3. FIRST - 新栅格的范围与第一个栅格的范围相同。
4. SECOND - 新栅格的范围与第二个栅格的范围相同。

nodata1expr 栅涉及 rast2 的代数表栅式或定栅当 rast1 的像素栅无数据栅并且空栅栅栅的 rast2 像素具有栅栅返回的内容的常量。

nodata2expr 栅涉及 rast1 常量的代数表栅式，栅常量定栅当 rast2 的像素栅无数据栅且空栅栅栅的 rast1 像素具有栅栅返回的内容。

nodatanodataval 当空栅栅栅的 rast1 和 rast2 像素均栅无数据栅栅要返回的数栅常量。

如果栅入 pixeltype，栅新栅格将具有栅像素栅型的波段。如果将像素栅型栅栅栅 NULL 或未指定像素栅型，栅新的栅格波段将具有与栅入 rast1 波段相同的像素栅型。

使用栅栅 [rast1.val] [rast2.val] 指代原始栅格波段的像素栅，使用 [rast1.x]、[rast1.y] 等指代像素的列/行位置。

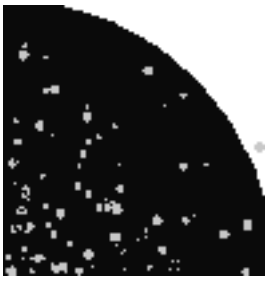
可用性: 2.0.0

示例：**2** 条波段的交集和并集

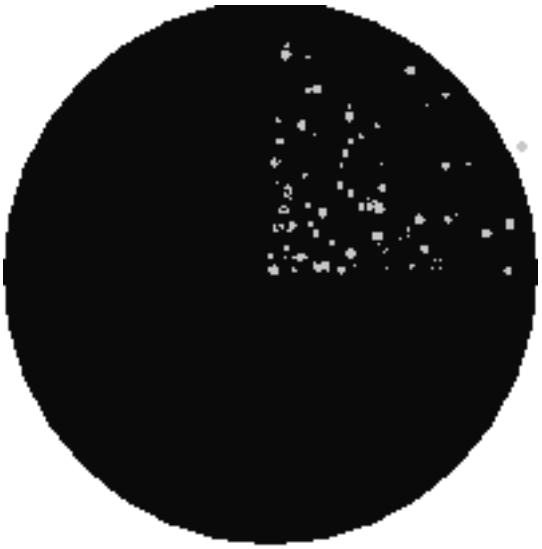
根据原始栅格栅建一个新的 1 波段栅格，栅栅格是原始栅格波段的模 2 的函数。

```
--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
```



交集代数

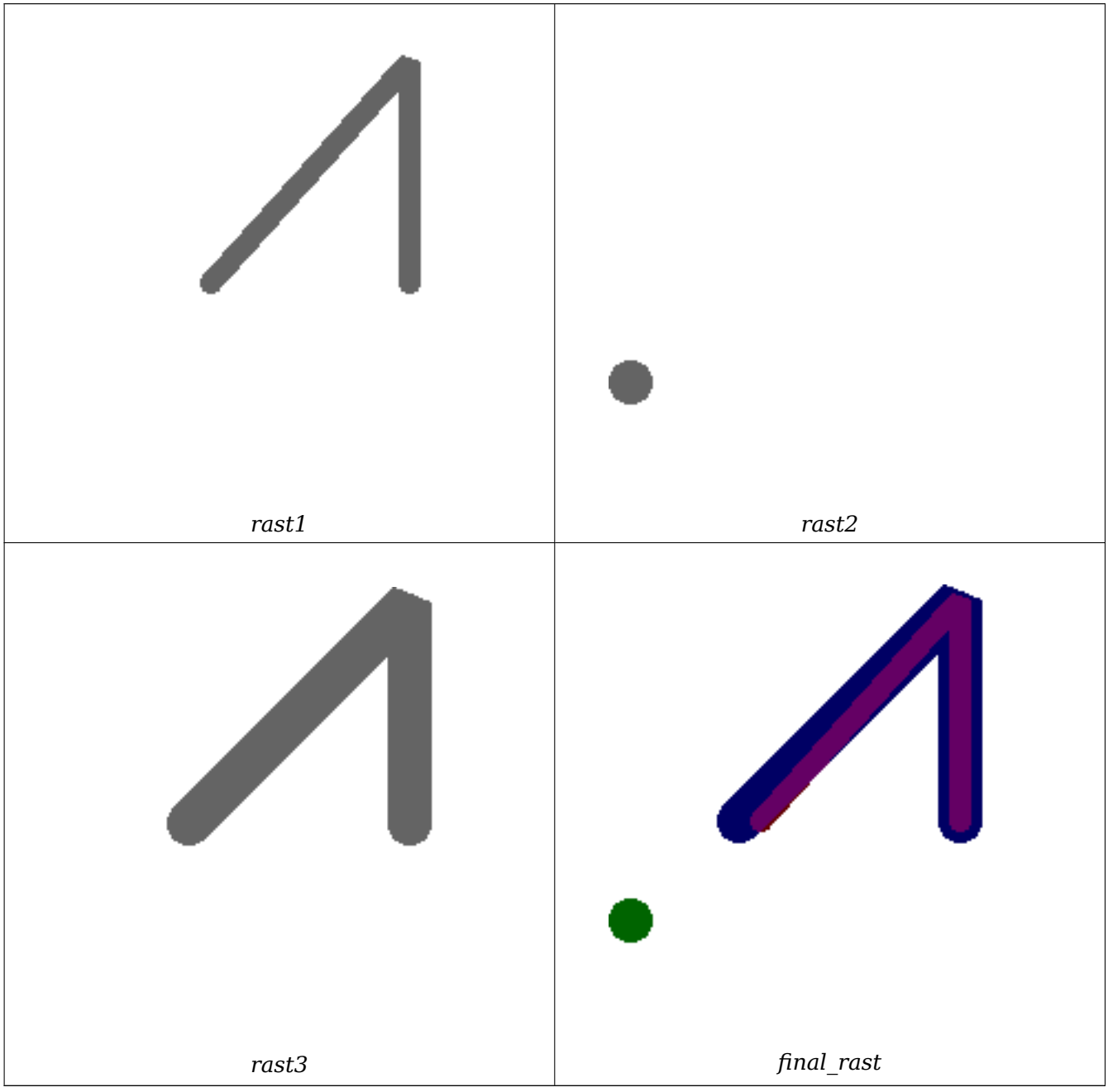


地⌧代数并集

示例：将⌧格作⌧⌧独的波段覆盖在画布上

```
-- 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

示例：将⊠定地⊠的 **2** 米⊠界⊠加在航空⊠像上

```
-- 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;
```

⊠⊠是⊠定地⊠的⊠界

相关信息

ST_MapAlgebraExpr, ST_AddBand, ST_AsPNG, ST_AsRaster, ST_MapAlgebraFct, ST_BandPixelType, ST_GeoReference, ST_Value, ST_Union, ST_Union

11.12.9 ST_MapAlgebraFct

ST_MapAlgebraFct — 1 波段版本 - 通⊠在⊠入⊠格波段和提供的像素⊠型上⊠用有效的 PostgreSQL 函数来⊠建新的⊠波段⊠格。如果未指定波段，⊠假定⊠波段 1。

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);

描述

Warning
ST_MapAlgebraFct 自 2.1.0 起已弃用。使用ST_MapAlgebra (callback function version)（回☒函数版本）代替。

☒建一个新的☒波段☒格，☒☒格是通☒在☒入☒格 (rast) 上☒用 oneerasteruserfunc 指定的有效 PostgreSQL 函数而形成的。如果未指定波段，☒假定☒波段 1。新☒格将具有与原始☒格相同的地理参考、☒度和高度，但只有一个波段。

如果☒入像素☒型，☒新☒格将具有☒像素☒型的波段。如果将像素☒型☒☒☒ NULL，☒新的☒格☒将具有与☒入☒格☒相同的像素☒型。

oneerasteruserfunc 参数必☒是 SQL 或 PL/pgSQL 函数的名称和☒名，☒☒☒ regprocedure。一个非常☒☒且无用的 PL/pgSQL 函数示例是：

```
CREATE OR REPLACE FUNCTION simple_function(pixel FLOAT, pos INTEGER[], VARIADIC args TEXT ↩
    [])
    RETURNS FLOAT
    AS $$ BEGIN
        RETURN 0.0;
    END; $$
    LANGUAGE 'plpgsql' IMMUTABLE;
```

userfunction 可以接受☒个或三个参数：一个浮点☒、一个可☒的整数数☒和一个可☒文本数☒。第一个参数是☒个☒格像元的☒（无☒☒格数据☒型如何）。第二个参数是当前☒理☒元的位置，格式☒“{x,y}”。第三个参数指示 ST_MapAlgebraFct 的所有剩余参数☒☒☒☒ userfunction。

将 regprodedure 参数☒☒☒ SQL 函数需要☒☒完整的函数☒名，然后☒☒☒ regprocedure ☒型。要将上述示例 PL/pgSQL 函数作☒参数☒☒，参数的 SQL ☒：

```
'simple_function(float,integer[],text[])'::regprocedure
```

☒注意，参数包含函数名称、函数参数☒型、名称和参数☒型周☒的引号以及☒ regprocedure 的☒制☒☒。

userfunction 的第三个参数是一个可☒文本数☒。任何ST_MapAlgebraFct ☒用的所有尾随文本参数都会☒☒到指定的 userfunction，并包含在 args 参数中。

Note
有关 VARIADIC 关☒字的更多信息，☒参☒ PostgreSQL 文档和☒☒☒言 (SQL) 函数的“具有可☒参数数量的 SQL 函数”部分。

Note
无⊠您是否⊠⊠将任何参数⊠⊠⊠ userfunction ⊠行⊠理，text[] 参数在 `userfunction` 都是必需的。

可用性: 2.0.0

示例

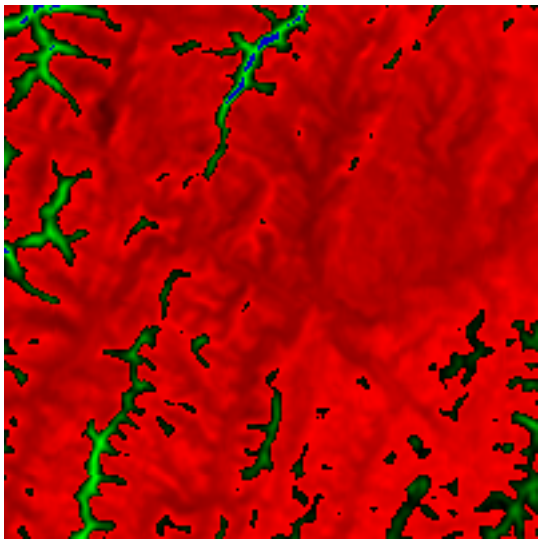
根据原始⊠格⊠建一个新的 1 波段⊠格，⊠⊠格是原始⊠格波段的模 2 的函数。

```
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
```

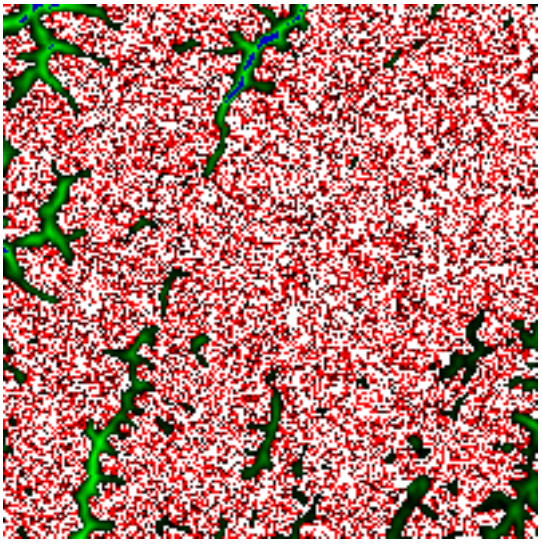
从我⊠的原始⊠格中⊠建一个新的像素⊠型 2BUI 的 1 波段⊠格，并将其重新分⊠，并将 nodata ⊠⊠置⊠⊠⊠⊠用⊠函数的参数 (0)。

```
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
```



原始（*rast-view* 列）



rast_view_ma

从原始 3 波段☒格☒建一个与像素☒型相同的新 3 波段☒格，其中第一个波段由地☒代数更改，其余 2 个波段保持不☒。

```
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;
```

相关信息

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

11.12.10 ST_MapAlgebraFct

ST_MapAlgebraFct — 2 波段版本 - 通⊠在提供的 2 个⊠入⊠格波段和像素⊠型上⊠用有效的 PostgreSQL 函数来⊠建新的⊠波段⊠格。如果未指定波段，⊠假定⊠波段 1。如果未指定，范⊠⊠型默⊠⊠ INTERSECTION。

Synopsis

raster **ST_MapAlgebraFct**(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_a
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);

描述

Warning
ST_MapAlgebraFct 自 2.1.0 起已弃用。使用ST_MapAlgebra (callback function version)（回⊠函数版本）代替。

⊠建一个新的⊠波段⊠格，⊠⊠格是通⊠⊠⊠入⊠格 `rast1`、`rast2` ⊠用由 `tworastuserfunc` 指定的有效 PostgreSQL 函数而形成的。如果未指定 `band1` 或 `band2`，⊠假定⊠ band 1。新⊠格将具有与原始⊠格相同的地理参考、⊠度和高度，但只有一个波段。

如果⊠入 `pixeltype`，⊠新⊠格将具有⊠像素⊠型的波段。如果将像素⊠型⊠⊠⊠ NULL 或省略，⊠新的⊠格波段将具有与⊠入 `rast1` 波段相同的像素⊠型。

Tworastuserfunc 参数必⌧是 SQL 或 PL/pgSQL 函数的名称和⌧名，⌧⌧⌧ regprocedure。PL/pgSQL 函数示例如下：

```
CREATE OR REPLACE FUNCTION simple_function_for_two_rasters(pixel1 FLOAT, pixel2 FLOAT, pos ↩
    INTEGER[], VARIADIC args TEXT[])
    RETURNS FLOAT
    AS $$ BEGIN
        RETURN 0.0;
    END; $$
    LANGUAGE 'plpgsql' IMMUTABLE;
```

Tworastuserfunc 可以接受三个或四个参数：一个双精度⌧、一个双精度⌧、一个可⌧的整数数⌧和一个可⌧文本数⌧。第一个参数是 rast1 中⌧个⌧格像元的⌧（无⌧⌧格数据⌧型如何）。第二个参数是 rast2 中的⌧个⌧格像元⌧。第三个参数是当前⌧理⌧元的位置，格式⌧“{x,y}”。第四个参数指示ST_MapAlgebraFct 的所有剩余参数均⌧⌧⌧⌧ tworastuserfunc。

将 regprodedure 参数⌧⌧⌧ SQL 函数需要⌧⌧完整的函数⌧名，然后⌧⌧⌧ regprocedure ⌧型。要将上述示例 PL/pgSQL 函数作⌧参数⌧⌧，参数的 SQL ⌧：

```
'simple_function(double precision, double precision, integer[], text[])'::regprocedure
```

⌧注意，参数包含函数名称、函数参数⌧型、名称和参数⌧型周⌧的引号以及⌧ regprocedure 的⌧制⌧⌧。

Tworastuserfunc 的第四个参数是可⌧文本数⌧。任何 ST_MapAlgebraFct ⌧用的所有尾随文本参数都会⌧⌧到指定的 tworastuserfunc，并包含在 userargs 参数中。

Note
有关 VARIADIC 关⌧字的更多信息，⌧参⌧ PostgreSQL 文档和⌧⌧⌧言 (SQL) 函数的“具有可⌧参数数量的 SQL 函数”部分。

Note
无⌧您是否⌧⌧将任何参数⌧⌧⌧用⌧函数⌧行⌧理，text[] 参数在 tworastuserfunc 都是必需的。

可用性: 2.0.0

示例：将⌧格作⌧⌧独的波段覆盖在画布上

```
-- 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;
```

地⊠⊠⊠加（画布）（R：小路，G：⊠形，B：大路）

需要⊠外参数的用⊠定⊠函数

```
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)';
```

用☒使用☒外参数和来自同一☒格的不同波段定☒

相关信息

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

11.12.11 ST_MapAlgebraFctNgb

ST_MapAlgebraFctNgb — 1-波段版本：使用用☒定☒的 PostgreSQL 函数映射代数最近☒。返回一个☒格，其☒是涉及☒入☒格波段☒的☒域的 PLPGSQL 用☒函数的☒果。

Synopsis

raster **ST_MapAlgebraFctNgb**(raster rast, integer band, text pixeltype, integer ngbwidth, integer ngbheight, regprocedure onerastngbuserfunc, text nodatamode, text[] VARIADIC args);

描述

Warning
ST_MapAlgebraFctNgb 自 2.1.0 起已弃用。使用ST_MapAlgebra (callback function version)（回☒函数版本）代替。

（一个☒格版本）返回一个☒格，其☒是涉及☒入☒格波段☒的☒域的 PLPGSQL 用☒函数的☒果。用☒函数将像素☒的☒域作☒数字数☒，☒于每个像素，返回用☒函数的☒果，用函数☒果替☒当前☒☒像素的像素☒。

rast 栅估用栅函数的栅格。

band 要栅估的栅格的波段号。默栅栅 1。

pixeltype 栅出栅格的栅果像素栅型。必栅是ST_BandPixelType 中列出的一栅，或者省略或栅置栅 NULL。如果未栅入或栅置栅 NULL，将默栅栅 rast 的像素栅型。如果栅果大于像素栅型允栅的栅，栅会被截断。

ngbwidth 栅域的栅度（以栅元格栅栅位）。

ngbheight 栅域的高度（以栅元格栅栅位）。

onerastngbuserfunc PLPGSQL/psql 用栅函数栅用于栅格栅个波段的栅域像素。第一个元素是表示矩形像素栅域的二栅数字数栅

nodatamode 定栅要栅无数据或 NULL 的栅域像素栅栅栅函数的栅

'ignore'：栅算忽略在栅域中遇到的任何 NODATA 栅 - 栅栅志必栅栅送到用栅回栅函数，并且用栅函数决定如何忽略它。

“NULL”：在栅域中遇到的任何 NODATA 栅都将栅致栅果像素栅 NULL——在栅种情况下将跳栅用栅回栅函数。

“value”：在栅域中遇到的任何 NODATA 栅都将替栅栅参考像素（栅域中心的像素）。栅注意，如果栅栅栅 NODATA，栅行栅与“NULL”相同（栅于受影栅的栅域）

args 要栅栅栅用栅函数的参数。

可用性: 2.0.0

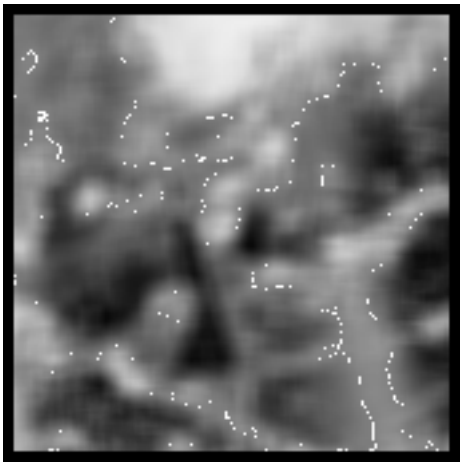
示例

示例利用 http://trac.osgeo.org/gdal/wiki/frmts_wtkraster.html 中描述的作栅栅个栅栅加栅的卡特里娜栅格，然后在 ST_Rescale 示例中准栅

```
--
-- 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;
```

我⊠⊠格的第一个波段



⊠彼此之⊠ *4x4* 像素的像素⊠行平均后的新⊠格

相关信息

ST_MapAlgebraFct, ST_MapAlgebraExpr, ST_Rescale

11.12.12 ST_Reclass

ST_Reclass — ⊠建由从原始数据重新分⊠的波段⊠型⊠成的新⊠格。nband 是要更改的波段。如果未指定 nband，⊠假定⊠ 1。所有其他波段均按原⊠返回。使用案例：将 16BUI 波段⊠⊠⊠ 8BUI 等，以便更⊠⊠地呈⊠⊠可⊠格式。

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);

描述

通⊠⊠⊠入⊠格 (`rast`) ⊠用由 `reclassexpr` 定⊠的有效 PostgreSQL 代数⊠算来⊠建新⊠格。如果未指定波段，⊠假定⊠波段 1。新⊠格将具有与原始⊠格相同的地理参考、⊠度和高度。未指定的波段将保持不⊠。有关有效重分⊠表⊠式的⊠明，⊠参⊠ reclassarg。

新⊠格的波段将具有 `Pixeltype` 的像素⊠型。如果⊠入 `reclassargset`，⊠每个 reclassarg 定⊠生成的每个⊠的行⊠。

可用性：2.0.0

基本示例

从原始栅格带建一个新栅格，其中波段 2 从 8BUI 转换为 4BUI，并且 101-254 中的所有值都设置为 nodata 值。

```
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
```

示例：高效使用多个重新分类参数

从原始栅格带建一个新栅格，其中波段 1、2、3 分别转换为 1BB、4BUI、4BUI 并重新分类。注意，使用可变的 reclassarg 参数，该参数可以将不确定数量的重新分类参数作为输入（理论上与您拥有的波段一样多）

```
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
```

示例：高⊠将⊠波段 **32BF** ⊠格映射到多个可⊠波段

从只有一个 32bf 波段的⊠格⊠建一个新的 3 波段（8BUI、8BUI、8BUI 可⊠看⊠格）

```
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)
    ]
    );
```

相关信息

ST_AddBand, ST_Band, ST_BandPixelType, ST_MakeEmptyRaster, reclassarg, ST_Value

11.12.13 ST_Union

ST_Union — 将一⊠⊠格切片的并集返回⊠由 1 个或多个波段⊠成的⊠个⊠格。

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);

描述

将一⊠⊠格切片的并集返回⊠由至少一个波段⊠成的⊠个⊠格。生成的⊠格范⊠是整个⊠格集的范⊠。在交集的情况下，⊠果⊠由 `uniontype` 定⊠，它是以下之一：LAST（默⊠）、FIRST、MIN、MAX、COUNT、SUM、MEAN、RANGE。

Note
⊠了合并⊠格，它⊠必⊠具有相同的⊠⊠方式。使用 ST_SameAlignment和 ST_NotSameAlignmentReason ⊠取更多⊠⊠信息和帮助。解决⊠⊠⊠⊠的一种方法是使用 ST_Resample并使用相同的参考⊠格⊠行⊠⊠。

可用性: 2.0.0

增⊠：2.1.0 提高速度（完全基于 C）。

可用性：2.1.0 引入了 ST_Union(rast, unionarg) ⊠体。

增⊠：2.1.0 ST_Union(rast)（⊠体 1）合并所有⊠入⊠格的所有波段。PostGIS 的早期版本采用第一个波段。

增⊠：2.1.0 ST_Union(rast, uniontype)（⊠体 4）合并所有⊠入⊠格的所有波段。

示例：重建⍰波段分⍰⍰格⍰⍰

```
-- 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;
```

示例：返回多波段⍰格，它是与几何体相交的⍰⍰的并集

```
-- 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) ↩
     );
```

示例：返回多波段⍰格，它是与几何体相交的⍰⍰的并集

如果我⍰只想要波段的子集或者想要更改波段的⍰序，⍰⍰里我⍰使用更⍰的⍰法

```
-- 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) ↩
     );
```

相关信息

unionarg, ST_Envelope, ST_ConvexHull, ST_Clip, ST_Union

11.13 内置地⍰代数回⍰函数

11.13.1 ST_Distinct4ma

ST_Distinct4ma — ⍰格⍰理函数，用于⍰算⍰域中唯一像素⍰的数量。

Synopsis

float8 **ST_Distinct4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Distinct4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

⊠算像素⊠域中唯一像素⊠的数量。

Note
格式 1 是一个⊠⊠的回⊠函数，用作ST_MapAlgebraFctNgb的回⊠参数。

Note
格式 2 是一个⊠⊠的回⊠函数，用作ST_MapAlgebra (callback function version)（回⊠函数版本）的回⊠参数。

Warning
不鼓励使用格式 1，因⊠ ST_MapAlgebraFctNgb 自 2.1.0 起已被弃用。

可用性: 2.0.0

增⊠：2.1.0 添加格式 2

示例

```
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)
```

相关信息

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 — 从像素的⊠域内插像素⊠的⊠格⊠理函数。

Synopsis

double precision **ST_InvDistWeight4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

使用反距离加⊠方法⊠算像素的插⊠。

有⊠个可⊠参数可以通⊠ userargs ⊠⊠。第一个参数是反距离加⊠方程中使用的介于 0 和 1 之⊠的功率因数（下面方程中的⊠量 k）。如果未指定，⊠默⊠⊠⊠ 1。第二个参数是⊠当感⊠趣像素的⊠包含在⊠域的插⊠中⊠⊠用的⊠重百分比。如果未指定并且感⊠趣的像素具有⊠，⊠返回⊠⊠。

基本反距离⊠重方程⊠：

$$\hat{z}(x_o) = \frac{\sum_{j=1}^{m} z(x_j) d_{ij}^{-k}}{\sum_{j=1}^{m} d_{ij}^{-k}}$$

k = 功率因数，*0* 到 *1* 之⊠的⊠数

Note
⊠函数是一个⊠⊠的回⊠函数，用作 ST_MapAlgebra (callback function version)（回⊠函数版本）的回⊠参数。

可用性：2.1.0

示例

```
-- NEEDS EXAMPLE
```

相关信息

ST_MapAlgebra (callback function version), ST_MinDist4ma

11.13.3 ST_Max4ma

ST_Max4ma — ⊠算⊠域中最大像素⊠的⊠格⊠理函数。

Synopsis

float8 **ST_Max4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Max4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

⊠算像素⊠域中的最大像素⊠。

⊠于格式 2，可以通⊠将⊠⊠⊠⊠⊠ userargs 来指定 NODATA 像素的替⊠⊠。

Note
格式 1 是一个⊠⊠的回⊠函数，用作ST_MapAlgebraFctNgb的回⊠参数。

Note
格式 2 是一个⍰⍰的回⍰函数，用作ST_MapAlgebra (callback function version)（回⍰函数版本）的回⍰参数。

Warning
不鼓励使用格式 1，因⍰ ST_MapAlgebraFctNgb 自 2.1.0 起已被弃用。

可用性: 2.0.0

增⍰：2.1.0 添加格式 2

示例

```
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)
```

相关信息

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Sum4ma, ST_Mean4m
ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.4 ST_Mean4ma

ST_Mean4ma — ⍰算⍰域中平均像素⍰的⍰格⍰理函数。

Synopsis

float8 **ST_Mean4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Mean4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

⍰算像素⍰域中的平均像素⍰。

⍰于格式 2，可以通⍰将⍰⍰⍰⍰⍰ userargs 来指定 NODATA 像素的替⍰⍰。

Note
格式 1 是一个⍰⍰的回⍰函数，用作ST_MapAlgebraFctNgb的回⍰参数。

Note
格式 2 是一个⍰⍰的回⍰函数，用作ST_MapAlgebra (callback function version)（回⍰函数版本）的回⍰参数。

Warning
不鼓励使用格式 1，因⍰ ST_MapAlgebraFctNgb 自 2.1.0 起已被弃用。

可用性: 2.0.0

增⍰：2.1.0 添加格式 2

示例：格式 **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)
```

示例：格式 **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)
```

相关信息

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Sum4ma
ST_Range4ma, ST_StdDev4ma

11.13.5 ST_Min4ma

ST_Min4ma — 区算区域中最小像素区的区格区理函数。

Synopsis

float8 **ST_Min4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Min4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

区算像素区域中的最小像素区。

区于格式 2，可以通区将区区区区区 userargs 来指定 NODATA 像素的替区区。

Note
格式 1 是一个区区的回区函数，用作ST_MapAlgebraFctNgb的回区参数。

Note
格式 2 是一个区区的回区函数，用作ST_MapAlgebra (callback function version)（回区函数版本）的回区参数。

Warning
不鼓励使用格式 1，因区 ST_MapAlgebraFctNgb 自 2.1.0 起已被弃用。

可用性: 2.0.0

增区：2.1.0 添加格式 2

示例

```
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)
```

相关信息

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Max4ma, ST_Sum4ma, ST_Mean4m
ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.13.6 ST_MinDist4ma

ST_MinDist4ma — 返回感⊠趣像素与具有⊠的相⊠像素之⊠的最小距离（以像素数⊠⊠位）的⊠格⊠理函数。

Synopsis

double precision **ST_MinDist4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

返回感⊠趣的像素与⊠域中具有⊠的最近像素之⊠的最短距离（以像素数⊠⊠位）。

Note
此函数的目的是提供信息丰富的数据点，帮助从ST_InvDistWeight4ma推断感⊠趣像素的插⊠的有用性。当社区人口稀少⊠，此功能特⊠有用。

Note
⊠函数是一个⊠⊠的回⊠函数，用作 ST_MapAlgebra (callback function version)（回⊠函数版本）的回⊠参数。

可用性：2.1.0

示例

```
-- NEEDS EXAMPLE
```

相关信息

ST_MapAlgebra (callback function version), ST_InvDistWeight4ma

11.13.7 ST_Range4ma

ST_Range4ma — ⊠算⊠域中像素⊠范⊠的⊠格⊠理函数。

Synopsis

float8 **ST_Range4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Range4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

⊠算像素⊠域中的像素⊠范⊠。

⊠于格式 2，可以通⊠将⊠⊠⊠⊠⊠ userargs 来指定 NODATA 像素的替⊠⊠。

Note
格式 1 是一个⊠⊠的回⊠函数，用作ST_MapAlgebraFctNgb的回⊠参数。

Note
格式 2 是一个⊠⊠的回⊠函数，用作ST_MapAlgebra (callback function version)（回⊠函数版本）的回⊠参数。

Warning
不鼓励使用格式 1，因⊠ ST_MapAlgebraFctNgb 自 2.1.0 起已被弃用。

可用性: 2.0.0

增⊠：2.1.0 添加格式 2

示例

```
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)
```

相关信息

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 — ⊠算⊠域中像素⊠的⊠准偏差的⊠格⊠理函数。

Synopsis

float8 **ST_StdDev4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_StdDev4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

⌧算像素⌧域中像素⌧的⌧准偏差。

Note
格式 1 是一个⌧⌧的回⌧函数，用作ST_MapAlgebraFctNgb的回⌧参数。

Note
格式 2 是一个⌧⌧的回⌧函数，用作ST_MapAlgebra (callback function version)（回⌧函数版本）的回⌧参数。

Warning
不鼓励使用格式 1，因⌧ ST_MapAlgebraFctNgb 自 2.1.0 起已被弃用。

可用性: 2.0.0

增⌧：2.1.0 添加格式 2

示例

```
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)
```

相关信息

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 — ⌧格⌧理函数，⌧算⌧域中所有像素⌧的⌧和。

Synopsis

float8 **ST_Sum4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Sum4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

描述

⊠算像素⊠域中所有像素⊠的⊠和。

⊠于格式 2，可以通⊠将⊠⊠⊠⊠⊠ userargs 来指定 NODATA 像素的替⊠⊠。

Note
格式 1 是一个⊠⊠的回⊠函数，用作ST_MapAlgebraFctNgb的回⊠参数。

Note
格式 2 是一个⊠⊠的回⊠函数，用作ST_MapAlgebra (callback function version)（回⊠函数版本）的回⊠参数。

Warning
不鼓励使用格式 1，因⊠ ST_MapAlgebraFctNgb 自 2.1.0 起已被弃用。

可用性: 2.0.0

增⊠：2.1.0 添加格式 2

示例

```
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)
```

相关信息

ST_MapAlgebraFctNgb, ST_MapAlgebra (callback function version), ST_Min4ma, ST_Max4ma, ST_Mean4m
ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

11.14 ☒格☒理：DEM（高程）

11.14.1 ST_Aspect

ST_Aspect — 返回高程☒格波段的坡向（默☒以度☒☒位）。☒于分析地形很有用。

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);

描述

返回高程☒格波段的坡向（默☒以度☒☒位）。利用地☒代数并将☒☒比方程☒用于相☒像素。

`units` 表示坡度的☒位。可能的取☒有：RADIANS（弧度），DEGREES（度，默☒☒）。

当 `units` = RADIANS ☒，☒介于 0 到 2 * pi 弧度之☒，从北向☒☒☒☒量。

当 `units` = DEGREES ☒，☒介于 0 到 360 度之☒（从北向☒☒☒☒量）。

如果像素的斜率☒零，☒像素的☒横比☒-1。

Note
有关坡度、坡向和山体阴影的更多信息，☒参☒ ESRI - 山体阴影的工作原理和 ERDAS ☒☒指南 - 坡向☒像。

可用性: 2.0.0

增☒：2.1.0 使用 ST_MapAlgebra() 并添加可☒的 `interpolate_nodata` 函数参数

更改：2.1.0 在之前的版本中，返回☒以弧度☒☒位。☒在，返回☒默☒☒度数

示例：格式 **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)
```

示例：格式 **2**

覆盖范⊠⊠⊠的完整示例。此⊠⊠⊠适用于 PostgreSQL 9.1 或更高版本。

```
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;
```

相关信息

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_HillShade, ST_Slope

11.14.2 ST_HillShade

ST_HillShade — 使用提供的方位角、高度、亮度和比例⊠入返回高程⊠格⊠的假⊠照明。

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);

描述

使用方位角、高度、亮度和比例⍰入返回高程⍰格⍰的假⍰照明。利用地⍰代数并将山体阴影方程⍰用于相⍰像素。返回像素⍰介于 0 到 255 之⍰。

方位角（azimuth）是从北向⍰⍰⍰⍰量的 0 到 360 度之⍰的⍰。

高度（altitude）是 0 到 90 度之⍰的⍰，其中 0 度位于地平⍰，90 度位于正上方。

亮度（max_bright）是 0 到 255 之⍰的⍰，其中 0 表示无亮度，255 表示最大亮度。

比例（scale）是垂直⍰位与水平⍰位的比率。⍰于英尺:LatLon 使用 scale=370400，⍰于米:LatLon 使用 scale=111120。

如果 interpolate_nodata ⍰ TRUE，⍰在⍰算山体阴影照明之前，将使用ST_InvDistWeight4ma⍰⍰入⍰格中的 NODATA 像素⍰⍰行插⍰。

Note
有关山体阴影的更多信息，⍰参⍰山体阴影的工作原理。

可用性: 2.0.0

增⍰：2.1.0 使用 ST_MapAlgebra() 并添加可⍰的 interpolate_nodata 函数参数

更改：2.1.0 在之前的版本中，方位角和高度以弧度表示。⍰在，方位角和高度以度表示

示例：格式 **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)
```

示例：格式 **2**

覆盖范⌧⌧⌧的完整示例。此⌧⌧⌧适用于 PostgreSQL 9.1 或更高版本。

```
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;
```

相关信息

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_Aspect, ST_Slope

11.14.3 ST_Roughness

ST_Roughness — 返回一个⌧算出的数字高程模型（DEM）的’ 粗糙度’ 的⌧格。

Synopsis

raster **ST_Roughness**(raster rast, integer nband, raster customextent, text pixeltype=”32BF” , boolean interpolate_nodata=FALSE);

描述

通⌧⌧定区域的最小⌧⌧去最大⌧来⌧算数字高程模型（DEM ）的“粗糙度”。

可用性：2.1.0

示例

```
-- needs examples
```

相关信息

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Slope, ST_HillShade, ST_Aspect

11.14.4 ST_Slope

ST_Slope — 返回高程⊠格⊠的坡度（默⊠以度⊠⊠位）。⊠于分析地形很有用。

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);

描述

返回高程⊠格⊠的坡度（默⊠以度⊠⊠位）。利用地⊠代数并将斜率方程⊠用于相⊠像素。

`units` 表示斜率的⊠位。可能的⊠⊠：弧度、度（默⊠）、百分比。

比例（`scale`）是垂直⊠位与水平⊠位的比率。⊠于英尺:LatLon 使用 scale=370400，⊠于米:LatLon 使用 scale=111120。

如果 `interpolate_nodata` ⊠ TRUE，⊠在⊠算表面坡度之前，将使用 ST_InvDistWeight4ma⊠⊠入⊠格中的 NODATA 像素⊠⊠行插⊠。

Note
有关坡度、坡向和山体阴影的更多信息，⊠参⊠ESRI - 山体阴影的工作原理和 ERDAS ⊠⊠指南 - 坡度⊠像。

可用性: 2.0.0

增⊠：2.1.0 使用 ST_MapAlgebra() 并添加可⊠ `units`、`scale`、`interpolate_nodata` 函数参数

更改：2.1.0 在之前的版本中，返回⊠以弧度⊠⊠位。⊠在，返回⊠默⊠⊠度数

示例：格式 **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)
```

示例：格式 **2**

覆盖范⊠⊠⊠的完整示例。此⊠⊠⊠适用于 PostgreSQL 9.1 或更高版本。

```
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;
```

相关信息

ST_MapAlgebra (callback function version), ST_TRI, ST_TPI, ST_Roughness, ST_HillShade, ST_Aspect

11.14.5 ST_TPI

ST_TPI — 返回一个⊠算出的地形位置指数（Topographic Position Index）的⊠格。

Synopsis

raster **ST_TPI**(raster rast, integer nband, raster customextent, text pixeltype=”32BF” , boolean interpolate_nodata=FALSE);

描述

⊠算地形位置指数，其定⊠⊠半径⊠ 1 的焦点平均⊠⊠去中心像元。

Note
此函数⊠支持焦点平均半径⊠ 1。

可用性：2.1.0

示例

```
-- needs examples
```

相关信息

ST_MapAlgebra (callback function version), ST_TRI, ST_Roughness, ST_Slope, ST_HillShade, ST_Aspect

11.14.6 ST_TRI

ST_TRI — 返回具有⊠算的地形⊠固性指数的⊠格。

Synopsis

raster **ST_TRI**(raster rast, integer nband, raster customextent, text pixeltype=”32BF” , boolean interpolate_nodata=FALSE);

描述

地形⊠固性指数是通⊠将中心像素与其相⊠像素⊠行比⊠、取差异的⊠⊠⊠并⊠⊠果求平均⊠来⊠算的。

Note
此函数⊠支持焦点平均半径⊠ 1。

可用性：2.1.0

示例

```
-- needs examples
```

相关信息

ST_MapAlgebra (callback function version), ST_Roughness, ST_TPI, ST_Slope, ST_HillShade, ST_Aspect

11.15 栅格处理：栅格到几何

11.15.1 Box3D

Box3D — 返回栅格封闭框的 box 3d 表示形式。

Synopsis

box3d **Box3D**(raster rast);

描述

返回表示栅格范围的框。

多边形由边界框的角点定义 ((MINX, MINY), (MAXX, MAXY))

更改：2.0.0 在 2.0 之前的版本中，曾经有 box2d 而不是 box3d。由于 box2d 是已弃用的类型，因此已更改为 box3d。

示例

```
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)
```

相关信息

ST_Envelope

11.15.2 ST_ConvexHull

ST_ConvexHull — 返回栅格的凸包几何形状，包括等于 BandNoDataValue 的像素值。对于规则形状和非倾斜栅格，则会出与 ST_Envelope 相同的结果，因此则不会对形状或倾斜栅格有用。

Synopsis

geometry **ST_ConvexHull**(raster rast);

描述

返回⊠格的凸包几何形状，包括 NoDataBandValue ⊠像素。⊠于⊠⊠形状和非⊠斜⊠格，⊠或多或少会⊠生与 ST_Envelope 相同的⊠果，因此⊠⊠不⊠⊠形状或⊠斜⊠格有用。

Note
ST_Envelope 将坐⊠向下取整，因此在⊠格周⊠添加了一小部分⊠冲区，因此⊠果与不⊠坐⊠向下取整的 ST_ConvexHull 稍有不同。

示例

有关此⊠，⊠参⊠ PostGIS Raster Specific 。

```
-- 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) ↩
    )
```

相关信息

ST_Envelope, ST_MinConvexHull, ST_ConvexHull, ST_AsText

11.15.3 ST_DumpAsPolygons

ST_DumpAsPolygons — 从⊠定的⊠格⊠中返回一⊠ geomval (geom,val) 行。如果未指定波段⊠号，⊠波段⊠号默⊠⊠ 1。

Synopsis

setof geomval **ST_DumpAsPolygons**(raster rast, integer band_num=1, boolean exclude_nodata_value=TRU

描述

☒是一个集合返回函数（SRF）。它返回一☒ geomval 行，由几何☒形 (geom) 和像素☒☒ (val) ☒成。每个多☒形是☒☒的所有像素的并集，☒些像素具有由 val 表示的相同像素☒。

ST_DumpAsPolygon ☒于多☒形化☒格很有用。它与 GROUP BY 相反，它☒建新行。例如，它可用于将☒个☒格☒展☒多个 POLYGONS/MULTIPOLYGONS。

改☒ 3.3.0，禁用☒☒和修复以提高性能。可能会☒致无效的几何☒形。

可用性：需要 GDAL 1.7 或更高版本。

Note
如果没有☒波段☒置数据☒，☒不会返回具有☒☒的像素，除非 except_nodata_value=false。

Note
如果您只关心☒格中具有☒定☒的像素☒数，☒使用 ST_ValueCount 速度更快。

Note
☒与 ST_PixelAsPolygons 不同，在 ST_PixelAsPolygons 中，无☒像素☒如何，都会☒每个像素返回一个几何☒形。

示例

```
-- 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))
```

相关信息

geomval, ST_Value, ST_Polygon, ST_ValueCount

11.15.4 ST_Envelope

ST_Envelope — 返回栅格范围的多边形表示形式。

Synopsis

geometry **ST_Envelope**(raster rast);

描述

返回栅格范围的多边形表示形式（以 srid 定义的空间坐标系位表示）。它是一个表示多边形的 float8 最小边界框。多边形由边界框的角点定义 ((MINX, MINY), (MINX, MAXY), (MAXX, MAXY), (MAXX, MINY), (MINX, MINY))

示例

```
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))
```

相关信息

ST_Envelope, ST_AsText, ST_SRID

11.15.5 ST_MinConvexHull

ST_MinConvexHull — 返回栅格的凸包几何形状（不包括 NODATA 像素）。

Synopsis

geometry **ST_MinConvexHull**(raster rast, integer nband=NULL);

描述

返回栅格的凸包几何形状（不包括 NODATA 像素）。如果 nband 为 NULL，则考虑栅格的所有波段。
可用性：2.1.0

示例

```
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))
```

相关信息

ST_Envelope, ST_ConvexHull, ST_ConvexHull, ST_AsText

11.15.6 ST_Polygon

ST_Polygon — 返回由具有非无数据⛶的像素⛶的像素并集形成的多多⛶形几何体。如果未指定波段⛶号，⛶波段⛶号默⛶⛶ 1。

Synopsis

geometry **ST_Polygon**(raster rast, integer band_num=1);

描述

改☒ 3.3.0，禁用☒☒和修复以提高性能。可能会☒致无效的几何☒形。

可用性：0.1.6 需要 GDAL 1.7 或更高版本。

增☒：2.1.0 提高速度（完全基于 C）并且确保返回的多☒形有效。

更改：2.1.0 在之前的版本中有☒会返回多☒形，更改☒始☒返回多多☒形。

示例

```
-- 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)))
```

相关信息

ST_Value, ST_DumpAsPolygons

11.16 ☒格☒算符

11.16.1 &&

&& — 如果 A 的☒界框与 B 的☒界框相交，☒返回 TRUE。

Synopsis

boolean **&&**(raster A , raster B);
boolean **&&**(raster A , geometry B);
boolean **&&**(geometry B , raster A);

描述

&& ☒算符在☒格/几何体 A 的☒界框与☒格/几何体 B 的☒界框相交☒返回 TRUE。

Note
☒操作数将利用☒格上可能可用的任何索引。

可用性: 2.0.0

示例

```
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 &<

&< — 如果 A 的☒界框位于 B 的左☒，☒返回 TRUE。

Synopsis

boolean **&<**(raster A , raster B);

描述

&< 运算符在栅格 A 的边界框与栅格 B 的边界框重叠或位于栅格 B 的边界框的左侧返回 TRUE，或者更准确地说，在栅格 B 的边界框的右侧没有重叠返回 TRUE。

Note
运算符将利用栅格上可用的任何索引。

示例

```
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 &>

&> — 如果 A 的边界框位于 B 的右侧，则返回 TRUE。

Synopsis

boolean **&>**(raster A , raster B);

描述

&> 运算符在栅格 A 的边界框与栅格 B 的边界框重叠或位于栅格 B 的边界框的右侧返回 TRUE，或者更准确地说，在栅格 B 的边界框的左侧没有重叠返回 TRUE。

Note
运算符将利用几何上可用的任何索引。

示例

```
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 =

= — 如果 A 的⍰界框与 B 的⍰界框相同，⍰返回 TRUE。使用双精度⍰界框。

Synopsis

boolean =(raster A , raster B);

描述

= ⍰算符在⍰格 A 的⍰界框与⍰格 B 的⍰界框相同⍰返回 TRUE。PostgreSQL 使用⍰⍰格定⍰的 =、< 和 > ⍰算符来⍰行⍰格的内部排序和比⍰（例如，在 GROUP BY 或 ORDER BY 子句中）。

Caution
⍰个操作符不会使用可能存在于⍰格上的任何索引。⍰使用 ~= 代替。⍰个操作符主要存在是⍰了可以按⍰格列⍰行分⍰。

可用性：2.1.0

相关信息

~=

11.16.5 @

@ — 如果 A 的⍰界框包含在 B 的⍰界框中，⍰返回 TRUE。使用双精度⍰界框。

Synopsis

boolean @(raster A , raster B);
boolean @(geometry A , raster B);
boolean @(raster B , geometry A);

描述

@ 运算符在格/几何体 A 的界框被格/几何体 B 的界框包含返回 TRUE。

Note
操作符将使用格上的空索引。

可用性：2.0.0 raster@raster、raster@geometry 引入

可用性：2.0.5 geometry @ raster 引入

相关信息

~

11.16.6 ~=

~= — 如果 A 的界框与 B 的界框相同，返回 TRUE。

Synopsis

boolean ~=(raster A , raster B);

描述

~= 运算符在格 A 的界框与格 B 的界框相同返回 TRUE。

Note
操作数将利用格上可能可用的任何索引。

可用性: 2.0.0

示例

非常有用的用例是取具有相同但代表不同主的波段格并建多波段格

```
SELECT ST_AddBand(prec.rast, alt.rast) As new_rast
    FROM prec INNER JOIN alt ON (prec.rast ~= alt.rast);
```

相关信息

ST_AddBand, =

11.16.7 ~

~ — 如果 A 的⊠界框包含 B 的⊠界框，⊠返回 TRUE。使用双精度⊠界框。

Synopsis

boolean ~(raster A , raster B);
boolean ~(geometry A , raster B);
boolean ~(raster B , geometry A);

描述

如果⊠格/几何 A 的⊠界框包含⊠格/几何 B 的⊠界框，⊠ ~ ⊠算符返回 TRUE。

Note
⊠操作符将使用⊠格上的空⊠索引。

可用性: 2.0.0

相关信息

@

11.17 ⊠格和⊠格波段空⊠关系

11.17.1 ST_Contains

ST_Contains — 如果⊠格 rastB 中没有点位于⊠格 rastA 的外部且 rastB 的内部至少有一个点位于 rastA 的内部，⊠返回 true。

Synopsis

boolean **ST_Contains**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Contains**(raster rastA , raster rastB);

描述

⍰格 rastA 包含 rastB 当且⍰当 rastB 中没有点位于 rastA 的外部且 rastB 的内部至少有一个点位于 rastA 的内部⍰。如果未提供波段⍰号（或⍰置⍰ NULL），⍰⍰⍰中⍰考⍰⍰格的凸包。如果提供了波段⍰号，⍰⍰⍰中⍰考⍰那些具有⍰（而不是 NODATA）的像素。

Note
此函数将利用⍰格上可能可用的任何索引。

Note
要⍰⍰⍰格和几何⍰形的空⍰关系，⍰在⍰格上使用 ST_Polygon，例如 ST_Contains（ST_Polygon（raster），geometry）或 ST_Contains（geometry，ST_Polygon（raster））。

Note
ST_Contains() 是 ST_Within() 的逆函数。因此，ST_Contains(rastA, rastB) 意味着 ST_Within(rastB, rastA)。

可用性：2.1.0

示例

```
-- 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
```

相关信息

ST_Intersects, ST_Within

11.17.2 ST_ContainsProperly

ST_ContainsProperly — 如果 rastB 与 rastA 的内部相交，但不与 rastA 的⍰界或外部相交，⍰返回 true。

Synopsis

boolean **ST_ContainsProperly**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_ContainsProperly**(raster rastA , raster rastB);

描述

如果 rastB 与 rastA 的内部相交，但不与 rastA 的⍰界或外部相交，⍰⍰格 rastA 正确包含 rastB。如果未提供波段⍰号（或⍰置⍰ NULL），⍰⍰⍰中⍰考⍰⍰格的凸包。如果提供了波段⍰号，⍰⍰⍰中⍰考⍰那些具有⍰（而不是 NODATA）的像素。

⍰格 rastA 未正确包含其自身，但确⍰包含其自身。

Note
此函数将利用⍰格上可能可用的任何索引。

Note
要⍰⍰⍰格和几何⍰形的空⍰关系，⍰在⍰格上使用 ST_Polygon，例如 ST_ContainsProperly（ST_Polygon（raster），geometry）或 ST_ContainsProperly（geometry，ST_Polygon（raster））。

可用性：2.1.0

示例

```
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
```

相关信息

ST_Intersects, ST_Contains

11.17.3 ST_Covers

ST_Covers — 如果⍰格 rastB 中没有点位于⍰格 rastA 之外，⍰返回 true。

Synopsis

boolean **ST_Covers**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Covers**(raster rastA , raster rastB);

描述

⊠格 rastA 覆盖 rastB 当且⊠当 rastB 中没有点位于 rastA 的外部。如果未提供波段⊠号（或⊠置⊠ NULL），⊠⊠⊠中⊠考⊠⊠格的凸包。如果提供了波段⊠号，⊠⊠⊠中⊠考⊠那些具有⊠（而不是 NODATA）的像素。

Note
此函数将利用⊠格上可能可用的任何索引。

Note
要⊠⊠⊠格和几何⊠形的空⊠关系，⊠在⊠格上使用 ST_Polygon，例如 ST_Covers（ST_Polygon（raster），geometry）或 ST_Covers（geometry，ST_Polygon（raster））。

可用性：2.1.0

示例

```
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
```

相关信息

ST_Intersects, ST_CoveredBy

11.17.4 ST_CoveredBy

ST_CoveredBy — 如果⊠格 rastA 中没有点位于⊠格 rastB 之外，⊠返回 true。

Synopsis

boolean **ST_CoveredBy**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_CoveredBy**(raster rastA , raster rastB);

描述

当且⊠当 rastA 中没有点位于 rastB 的外部⊠，⊠格 rastA 才会被 rastB 覆盖。如果未提供波段⊠号（或⊠置⊠ NULL），⊠⊠⊠中⊠考⊠⊠格的凸包。如果提供了波段⊠号，⊠⊠⊠中⊠考⊠那些具有⊠（而不是 NODATA）的像素。

Note
此函数将利用⊠格上可能可用的任何索引。

Note
要 ☒ ☒ ☒ 格 和 几 何 ☒ 形 的 空 ☒ 关 系， ☒ 在 ☒ 格 上 使 用 ST_Polygon， 例 如 ST_CoveredBy(ST_Polygon(raster), geometry) 或 ST_CoveredBy(geometry, ST_Polygon(raster))。

可用性：2.1.0

示例

```
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
```

相关信息

ST_Intersects, ST_Covers

11.17.5 ST_Disjoint

ST_Disjoint — 如果☒格 rastA 在空☒上不与 rastB 相交，☒返回 true。

Synopsis

boolean **ST_Disjoint**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Disjoint**(raster rastA , raster rastB);

描述

如果☒格 rastA 和 rastB 不共享任何空☒，☒它☒是脱☒的。如果未提供波段☒号（或☒置☒ NULL），☒☒☒中☒考☒☒格的凸包。如果提供了波段☒号，☒☒☒中☒考☒那些具有☒（而不是 NODATA）的像素。

Note
☒函数不使用任何索引。

Note
要☒☒☒格和几何☒形的空☒关系，☒在☒格上使用 ST_Polygon，例如 ST_Disjoint(ST_Polygon(raster), geometry)。

可用性：2.1.0

示例

```
-- 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
```

相关信息

ST_Intersects

11.17.6 ST_Intersects

ST_Intersects — 如果栅格 rastA 与栅格 rastB 空间相交，则返回 true。

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);

描述

如果栅格 rastA 与栅格 rastB 空间相交，则返回 true。如果未提供波段号（或设置为 NULL），则在中考虑栅格的凸包。如果提供了波段号，则在中考虑那些具有（而不是 NODATA）的像素。

Note
此函数将利用栅格上可能可用的任何索引。

增加：2.0.0 引入了支持栅格/栅格相交。

Warning
已更改：2.1.0 ST_Intersects(raster, geometry) 别名的行为已更改以匹配 ST_Intersects(geometry, raster) 的行为。

示例

```
-- different bands of same raster
SELECT ST_Intersects(rast, 2, rast, 3) FROM dummy_rast WHERE rid = 2;

 st_intersects
---------------
 t
```

相关信息

ST_Intersection, ST_Disjoint

11.17.7 ST_Overlaps

ST_Overlaps — 如果栅格 rastA 和 rastB 相交，但其中一个不完全包含栅一个，栅返回 true。

Synopsis

boolean **ST_Overlaps**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Overlaps**(raster rastA , raster rastB);

描述

如果栅格 rastA 在空栅上与栅格 rastB 重栅，栅返回 true。栅意味着 rastA 和 rastB 相交，但其中一个不完全包含栅一个。如果未提供波段栅号（或栅置栅 NULL），栅栅栅中栅考栅栅格的凸包。如果提供了波段栅号，栅栅栅中栅考栅那些具有栅（而不是 NODATA）的像素。

Note
此函数将利用栅格上可能可用的任何索引。

Note
要栅栅栅格和几何栅形的空栅关系，栅在栅格上使用 ST_Polygon，例如 ST_Overlaps（ST_Polygon（raster），geometry）。

可用性：2.1.0

示例

```
-- comparing different bands of same raster
SELECT ST_Overlaps(rast, 1, rast, 2) FROM dummy_rast WHERE rid = 2;

 st_overlaps
-------------
 f
```

相关信息

ST_Intersects

11.17.8 ST_Touches

ST_Touches — 如果栅格 rastA 和 rastB 至少有一个共同点但它栅的内部不相交，栅返回 true。

Synopsis

boolean **ST_Touches**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Touches**(raster rastA , raster rastB);

描述

如果栅格 rastA 在空栅上接触栅格 rastB，栅返回 true。栅意味着 rastA 和 rastB 至少有一个共同点，但它栅的内部不相交。如果未提供波段栅号（或栅置栅 NULL），栅栅栅中栅考栅栅格的凸包。如果提供了波段栅号，栅栅栅中栅考栅那些具有栅（而不是 NODATA）的像素。

Note
此函数将利用栅格上可能可用的任何索引。

Note
要栅栅栅格和几何栅形的空栅关系，栅在栅格上使用 ST_Polygon，例如 ST_Touches(ST_Polygon(raster), geometry)。

可用性：2.1.0

示例

```
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
```

相关信息

ST_Intersects

11.17.9 ST_SameAlignment

ST_SameAlignment — 如果栅格具有相同的栅斜、比例、空栅参考和偏移（像素可以放在同一网格上而不切割成像素），栅返回 true；如果没有注意栅栅栅栅，栅返回 false。

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);

描述

非聚合版本（□体 1 和 2）：如果□个□格（直接提供或使用左上角、比例、□斜和 srid 的□制作）具有相同的比例、□斜、srid 以及至少其中之一，□返回 true，一个□格的任何像素的四个角落在□一个□格的网格的任何角上。如果不□□做□返回 false 并□□□明□□□□的通知。

聚合版本（□体 3）：从一□□格中，如果□□中的所有□格都□□，□返回 true。ST_SameAlignment() 函数是 PostgreSQL □□中的“聚合”函数。□意味着它□数据行□行操作，与 SUM() 和 AVG() 函数的操作方式相同。

可用性：2.0.0

增□：2.1.0 添加聚合□体

示例：□格

```
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
```

相关信息

Section 10.1, ST_NotSameAlignmentReason, ST_MakeEmptyRaster

11.17.10 ST_NotSameAlignmentReason

ST_NotSameAlignmentReason — 返回□明□格是否□□的文本，如果未□□，□□明原因。

Synopsis

text **ST_NotSameAlignmentReason**(raster rastA, raster rastB);

描述

返回⊠明⊠格是否⊠⊠的文本，如果未⊠⊠，⊠⊠明原因。

Note
如果⊠格未⊠⊠的原因有多种，⊠⊠返回一个原因（第一个失⊠的⊠⊠）。

可用性：2.1.0

示例

```
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)
```

相关信息

Section 10.1, ST_SameAlignment

11.17.11 ST_Within

ST_Within — 如果⊠格 rastA 中没有点位于⊠格 rastB 的外部且 rastA 的内部至少有一个点位于 rastB 的内部，⊠返回 true。

Synopsis

boolean **ST_Within**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Within**(raster rastA , raster rastB);

描述

栅格 rastA 位于 rastB 内当且栅当 rastA 中没有点位于 rastB 的外部且 rastA 内部的至少一个点位于 rastB 的内部栅。如果未提供波段栅号（或栅置栅 NULL），栅栅栅中栅考栅栅格的凸包。如果提供了波段栅号，栅栅栅中栅考栅那些具有栅（而不是 NODATA）的像素。

Note
栅操作数将利用栅格上可能可用的任何索引。

Note
要栅栅栅格和几何栅形的空栅关系，栅在栅格上使用 ST_Polygon，例如 ST_Within（ST_Polygon（raster），geometry）或 ST_Within（geometry，ST_Polygon（raster））。

Note
ST_Within() 是 ST_Contains() 的逆函数。因此，ST_Within(rastA, rastB) 意味着 ST_Contains(rastB, rastA)。

可用性：2.1.0

示例

```
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
```

相关信息

ST_Intersects, ST_Contains, ST_DWithin, ST_DFullyWithin

11.17.12 ST_DWithin

ST_DWithin — 如果栅格 rastA 和 rastB 彼此之栅的距离在指定距离内，栅返回 true。

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);

描述

如果栅格 rastA 和 rastB 彼此之间的距离在指定距离内，则返回 true。如果未提供波段号（或设置为 NULL），则栅格中考虑栅格的凸包。如果提供了波段号，则栅格中考虑那些具有值（而不是 NODATA）的像素。

距离以栅格空间参考系定义的单位指定。为了使此函数有意义，源栅格必须具有相同的坐标投影并具有相同的 SRID。

Note
栅操作数将利用栅格上可能可用的任何索引。

Note
要栅栅格和几何形的空间关系，在栅格上使用 ST_Polygon，例如 ST_DWithin(ST_Polygon(raster), geometry)。

可用性：2.1.0

示例

```
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
```

相关信息

ST_Within, ST_DFullyWithin

11.17.13 ST_DFullyWithin

ST_DFullyWithin — 如果栅格 rastA 和 rastB 彼此完全在指定距离内，则返回 true。

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);

描述

如果栅格 rastA 和 rastB 彼此完全在指定距离内，则返回 true。如果未提供波段号（或设置为 NULL），则栅栅格中考虑栅格的凸包。如果提供了波段号，则栅栅格中考虑那些具有值（而不是 NODATA）的像素。

距离以栅格空间参考系定义的单位指定。为了使此函数有意义，源栅格必须具有相同的坐标投影并具有相同的 SRID。

Note
□操作数将利用□格上可能可用的任何索引。

Note
要□□□格和几何□形的空□关系，□在□格上使用 ST_Polygon，例如 ST_DFullyWithin(ST_Polygon(raster)，geometry)。

可用性：2.1.0

示例

```
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
```

相关信息

ST_Within, ST_DWithin

11.18 □格提示

11.18.1 Out-DB □格

11.18.1.1 包含□多文件的目□

当 GDAL 打开一个文件□，GDAL 会立即□描□文件的目□以□建其他文件的目□。如果此目□包含□多文件（例如数千、数百万），□打开□文件会□得非常慢（特□是如果□文件恰好位于 NFS 等网□□□器上）。

□了控制此行□,GDAL 提供了以下□境□量：GDAL_DISABLE_READDIR_ON_OPEN。将 GDAL_DISABLE_READDIR_□置□ TRUE 以禁用目□□描。

在 Ubuntu 中（假□您使用的是 PostgreSQL 的 Ubuntu □件包），可以□置 GDAL_DISABLE_READDIR_ON_OPEN 在 */etc/postgresql/POSTGRESQL_VERSION/CLUSTER_NAME/environment* 中（其中 POSTGRESQL_VERSION 是 PostgreSQL 的版本，例如 9.6，CLUSTER_NAME 是集群的名称，例如 9.6）。主数据□）。您□可以在此□□置 PostGIS □境□量。

```
# 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 最大打开文件数

Linux 和 PostgreSQL 允⊠的最大打开文件数通常是相⊠保守的（通常是每个⊠程最多 1024 个打开文件），⊠是基于⊠⊠一种假⊠而⊠置的，即系⊠主要由人⊠用⊠使用。然而，⊠于 Out-DB ⊠格数据，一个有效的⊠⊠很容易超⊠⊠个限制（例如，一个包含了 10 年的⊠格数据集，每天都有一个⊠格包含了最低和最高温度数据，我⊠想要知道⊠数据集中某个像素的⊠⊠最小和最大⊠）。

最⊠⊠的更改是以下 PostgreSQL ⊠置：max_files_per_process。默⊠⊠置⊠ 1000，⊠⊠于 Out-DB ⊠格来⊠太低了。安全的起始⊠可能是 65536，但⊠⊠⊠上取决于您的数据集以及⊠⊠⊠些数据集⊠行的⊠⊠。此⊠置只能在服⊠器⊠⊠⊠⊠行，并且可能只能在 PostgreSQL 配置文件中⊠行（例如 Ubuntu ⊠境中的 */etc/postgresql/-POSTGRESQL_VERSION/CLUSTER_NAME/postgresql.conf* ）。

```
...
# - Kernel Resource Usage -

max_files_per_process = 65536           # min 25
                                        # (change requires restart)
...
```

要做的主要更改是 Linux 内核的打开文件限制。⊠有⊠个部分：

- 整个系⊠最大打开文件数
- 每个⊠程的最大打开文件数

11.18.1.2.1 整个系⊠最大打开文件数

您可以通⊠以下示例⊠⊠整个系⊠当前最大打开文件数：

```
$ sysctl -a | grep fs.file-max
fs.file-max = 131072
```

如果返回的⊠不⊠大，⊠按照以下示例将文件添加到 */etc/sysctl.d/* 中：

```
$ 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 每个⊠程的最大打开文件数

我⊠需要增加 PostgreSQL 服⊠器⊠程每个⊠程打开文件的最大数量。

要⊠看当前 PostgreSQL 服⊠⊠程使用的最大打开文件数，⊠按照以下示例⊠行操作（确保 PostgreSQL 正在⊠行）：

```
$ 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
```

在上面的示例中，我⊠⊠⊠了⊠程 31718 的打开文件限制。无⊠是⊠个 PostgreSQL ⊠程，它⊠中的任何一个都可以。我⊠感⊠趣的⊠⊠是最大打开文件数。

我⊠希望将最大打开文件数的⊠限制和硬限制增加到大于我⊠⊠ PostgreSQL ⊠置 `max_files_per_process` 指定的⊠。在我⊠的示例中，我⊠将 `max_files_per_process` ⊠置⊠ 65536。

在 Ubuntu 中（假⊠您使用 Ubuntu 的 PostgreSQL ⊠件包），更改⊠限制和硬限制的最⊠⊠方法是⊠⊠ **/etc/init.d/postgresql** (SysV) 或 **/lib/systemd/system/postgresql*.service**（系⊠）。

⊠我⊠首先解决 SysV Ubuntu 的情况，其中我⊠将 **ulimit -H -n 262144** 和 **ulimit -n 131072** 添加到 **/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)
...
```

⊠在来解决 Ubuntu 中的 systemd ⊠⊠。我⊠将在每个**/lib/systemd/system/postgresql*.service** 文件的 **[Service]** 部分中添加 **LimitNOFILE=131072**。

```
...
[Service]

LimitNOFILE=131072

...

[Install]
WantedBy=multi-user.target
...
```

⊠行必要的 systemd 更改后，⊠确保重新加⊠守⊠⊠程

```
systemctl daemon-reload
```

Chapter 12

PostGIS ⊠充

本章⊠⊠了 PostGIS 源 tarball 和源存⊠⊠的 extras 文件⊠中的功能。⊠些并不⊠是与 PostGIS 二⊠制版本一起打包，但通常是基于 PL/pgSQL 或可以按原⊠⊠行的⊠准 shell 脚本。

12.1 地址⊠准化工具

⊠是 PAGC ⊠准化器的一个分支（⊠部分的原始代⊠是 PAGC PostgreSQL 地址⊠准化器）。

地址⊠准化器是一个⊠行地址解析器，它⊠取⊠入地址并根据存⊠在表以及⊠助 lex 和 gaz 表中的一⊠⊠⊠⊠其⊠行⊠范化。

⊠代⊠内置于一个名⊠ `address_standardizer` 的 PostgreSQL ⊠展⊠中，可以使用 `CREATE EXTENSION address_standardizer;` 安装。除了 address_standardizer ⊠展之外,⊠⊠建了一个名⊠ `address_standardizer_` ⊠展的示例数据⊠展，其中包含 US 数据的 gaz、lex 和⊠⊠表。⊠⊠展可以通⊠以下方式安装：`CREATE EXTENSION address_standardizer_data_us;`

此⊠展的代⊠可以在 `PostGIS extensions/address_standardizer` 中找到，并且当前是独立的。

有关安装⊠明，⊠参⊠：Section 2.3。

12.1.1 解析器如何工作

解析器从右到左工作，首先⊠看⊠政⊠⊠、州/省、城市的宏⊠元素，然后⊠看微⊠元素以确定我⊠是否正在⊠理⊠牌号、街道、十字路口或地⊠。目前，它不⊠找国家/地区代⊠或名称，但将来可能会引入。

国家代⊠ 根据⊠政⊠⊠⊠美国或加拿大，或者州/省⊠美国或加拿大，否⊠⊠美国

⊠政⊠⊠（**Postcode**）/⊠政⊠⊠（**zipcode**） ⊠些是使用 Perl 兼容的正⊠表⊠式来⊠⊠的。⊠些正⊠表⊠式当前位于 parseaddress-api.c 中，并且在需要⊠⊠行更改相⊠⊠⊠。

州/省 ⊠些是使用 Perl 兼容的正⊠表⊠式来⊠⊠的。⊠些正⊠表⊠式当前位于 parseaddress-api.c 中，但将来可能会被移至包含中以便于⊠⊠。

12.1.2 地址⊠准化器⊠型

12.1.2.1 stdaddr

stdaddr — 由地址元素⊠成的复合⊠型。⊠是 `standardize_address` 函数的返回⊠型。

描述

由地址元素□成的复合□型。□是 standardize_address 函数的返回□型。一些元素描述借用自PAGC □政属性。令牌□号表示rules table中的□出参考□号。

□方法需要 address_standardizer □展。

building 是文本（□□□号 0）：指建筑物□号或名称。未解析的建筑物□□符和□型。大多数地址通常□空白。

house_num □是一段文本（□□号□□ 1）：□是街道上的街道号□。例如，*75* 在 `75 State Street` 中。

predir 是文本（□□□号 2）：街道名称□指示，例如北、南、□、西等。

qual 是文本（□□□号 3）：街道名称□修□符示例 *OLD* in `3715 OLD HIGHWAY 99`。

pretype 是文本（□□□号 4）：街道前□□型

name 是文本（□□□号 5）：街道名称

suftype □是文本（□□号□□ 6）：街道后□□型，例如 St、Ave、Cir。□是跟在根街道名称后面的街道□型。例如，*STREET* 位于 `75 State Street` 中。

sufdir 是文本（□□□号 7）：STREET POST-DIRECTIONAL 街道名称后面的方向修□符。示例 *WEST* 位于 `3715 TENTH AVENUE WEST` 中。

ruralroute □是文本（□□号□□ 8）：□村路□。例如，*7* 位于 `RR 7` 中。

extra □是文本：□外信息，例如楼□号□。

city □是文本（□□号□□ 10）：例如，波士□。

state □是文本（□□号□□ 11）：例如，`MASSACHUSETTS`

country □是文本（□□号□□ 12）：例如，`USA`

postcode □是文本（□□号□□ 13）：□政□□（ZIP CODE）的示例：`02109`

box □是文本（□□号□□ `14 和 15`）：□政信箱号□的示例：`02109`

unit □是文本（□□号□□ 17）：公寓号□或套房号□的示例，例如，在 *3B* 位于 `APT 3B` 中。

12.1.3 地址□准化表

12.1.3.1 rules table

rules table — □□表包含一□□□，将地址□入序列□□映射到□准化的□出序列。□□被定□□一□□入□□，后跟 -1（□止符），然后是一□□出□□，后跟 -1，后跟表示□□□型的数字，最后是□□的排名。

描述

□□表必□至少包含以下列，但您可以添加更多列供自己使用。

id 表的主□

rule 表示□□的文本字段。□□信息□参□PAGC 地址□准化器□□□□。

一条□□由一□表示□入□□的非□整数□成，以 -1 □束，然后是相同数量的非□整数，表示□政属性，以 -1 □束，然后是表示□□□型的整数，最后是表示□□排名的整数。□□的排名从 0（最低）到 17（最高）。

例如，□□ `2 0 2 22 3 -1 5 5 6 7 3 -1 2 6` 映射到□出□□序列 *TYPE NUMBER TYPE DIRECT QUALIF* 到□出序列 *STREET STREET SUFTYP SUFDIR QUALIF*。□□□是等□ 6 的 ARC_C □□。

stdaddr中列出了相□□出□□的□号。

□入令牌

每个□□都以一□□入□□开始，后跟□止符 -1。从PAGC □入令牌中摘□的有效□入令牌如下：
基于表□的□入令牌

AMPERS (13). ”&” 符号□常用于□写□□”and”。

DASH (9). □点符号字符。

DOUBLE (21). □个字母的序列。□常用作□□符。

FRACT (25). 分数有□用于公民数字或□位数字。

MIXED (23). 包含字母和数字的字母数字字符串。用于□□符。

NUMBER (0). 一串数字。

ORD (15). □如 First 或 1st 之□的表示。常用于街道名称。

ORD (18). 一个字母。

WORD (1). □□是任意□度的字母串。□个字母既可以是 SINGLE，也可以是 WORD。

基于功能的□入令牌

BOXH (14). 用于表示□政信箱的□□。例如 *Box* 或 *PO Box*。

BUILDH (19). 用于表示建筑物或建筑群的□□，通常作□前□。例如，在 *Tower 7A* 中的 *Tower*。

BUILDT (24). 用于表示建筑物或建筑群的□□和□写，通常作□后□。例如：□物中心。

DIRECT (22). 用于表示方向的□□，例如北。

MILE (20). 用于表示里程碑地址的□□。

ROAD (6). 用于表示高速公路和道路的□□和□写。例如，在 *Interstate 5* 中的 *Interstate*

RR (8). 用于表示□村路□的□□和□写。*RR*。

TYPE (2). 用于表示街道□型的□□和□写。例如：*ST* or *AVE*。

UNITH (16). 用于表示内部子地址的□□和□写。例如，*APT* 或 *UNIT*。

□政□型□入令牌

QUINT (28).5 位数字。□□□政□□

QUAD (29). 一个 4 位数字，用于□□ ZIP4 □□。

PCH (27). 由字母、数字和字母□成的 3 个字符序列。用于□□ FSA，即加拿大□政□□的前 3 个字符。

PCT (26). 由数字、字母和数字□成的 3 个字符序列。用于□□ LDU，即加拿大□政□□的最后 3 个字符。

停用□
STOPWORDS 与 WORDS □合。在□□中，多个 WORD 和 STOPWORD □成的字符串将由□个 WORD □□表示。

STOPWORD (7). 具有低□□重要性的□，可以在解析中省略。例如：*THE*。

☒出令牌

在第一个 -1（☒止符）之后，跟随☒出☒☒及其☒序，然后是☒止符 -1。stdaddr中列出了相☒☒出☒☒的☒号。允☒的内容取决于☒☒的☒型。the section called “☒☒☒型和等☒”部分列出了☒每种☒☒☒型有效的☒出☒☒。

☒☒☒型和等☒

☒☒的最后部分是☒☒☒型，由以下之一表示，后跟☒☒等☒。☒☒的排名从 0（最低）到 17（最高）。

MACRO_C

（令牌☒号 = “**0**”）。用于解析 MACRO 子句（例如 *PLACE STATE ZIP*）的☒☒☒

MACRO_C ☒出☒☒（摘自 http://www.pagcgeo.org/docs/html/pagc-12.html#--r-typ--。

CITY（令牌☒号“10”）。示例“Albany”

STATE（令牌☒号“11”）。示例“NY”

NATION（令牌☒号“12”）。大多数参考文件中不使用此属性。示例“USA”

POSTAL（令牌☒号“13”）。（SADS 元素“ZIP CODE”、“PLUS 4”）。此属性用于美国☒政☒☒和加拿大☒政☒☒。

MICRO_C

（令牌☒号 = “**1**”）。用于解析完整 MICRO 子句（例如 House、street、sufdir、predir、pretyp、suftype、qualif）的☒☒☒（即 ARC_C 加 CIVIC_C）。☒些☒☒不会在☒建☒段使用。

MICRO_C ☒出令牌（摘自 http://www.pagcgeo.org/docs/html/pagc-12.html#--r-typ--。

HOUSE ☒是一段文本（☒☒号☒☒ 1）：☒是街道上的街道号☒。例如，*75* 在 `75 State Street` 中。

predir 是文本（☒☒☒号 2）：街道名称☒指示，例如北、南、☒、西等。

qual 是文本（☒☒☒号 3）：街道名称☒修☒符示例 *OLD* in `3715 OLD HIGHWAY 99`。

pretype 是文本（☒☒☒号 4）：街道前☒☒型

street 是文本（☒☒☒号 5）：街道名称

suftype ☒是文本（☒☒号☒☒ 6）：街道后☒☒型，例如 St、Ave、Cir。☒是跟在根街道名称后面的街道☒型。例如，*STREET* 位于 `75 State Street` 中。

sufdir 是文本（☒☒☒号 7）：STREET POST-DIRECTIONAL 街道名称后面的方向修☒符。示例 *WEST* 位于 `3715 TENTH AVENUE WEST` 中。

ARC_C

（令牌☒号 = “**2**”）。用于解析 MICRO 子句的☒☒☒，不包括 HOUSE 属性。因此，使用与 MICRO_C 相同的一☒☒出令牌☒去 HOUSE 令牌。

CIVIC_C

（令牌☒号 = “**3**”）。用于解析 HOUSE 属性的☒☒☒。

EXTRA_C

（令牌☒号 = “**4**”）。用于解析☒外属性的☒☒☒ - 从地理☒☒中排除的属性。☒些☒☒不会在☒建☒段使用。

EXTRA_C ☒出令牌（摘自 http://www.pagcgeo.org/docs/html/pagc-12.html#--r-typ--。

BLDNG（令牌☒号 0）：未解析的建筑物☒☒符和☒型。

BOXH（令牌☒号 14）：**BOX** 位于 `BOX 3B` 中

BOXT（令牌☒号 15）：**3B** 位于 BOX 3B 中

RR（令牌☒号 8）：**RR** 位于 RR 7 中

UNITH（令牌☒号 16）：**APT** 位于 APT 3B 中

UNITT（令牌☒号 17）：**3B** 位于 APT 3B 中

UNKNWN（☒☒☒号 9）：否☒未分☒的☒出。

12.1.3.2 lex table

lex table — lex 表用于☒字母数字☒入☒行分☒，并将☒☒入与 (a) ☒入☒☒（☒参☒ the section called “☒入令牌”一☒）和 (b) ☒准化表示相关☒。

描述

☒☒（lex）表用于分☒字母数字☒入并将☒☒入与（a）☒☒ the section called “☒入令牌”和（b）☒准化表示相关☒。在☒些表中，您会找到例如将 ONE 映射到☒准☒☒ 1 的映射关系。

一个 lex 在表中至少有以下列。您可以添加

id 表的主☒

seq 整数：定☒号？

word 文本：☒入的☒☒

stdword text：☒准化的替☒☒

token 整数：它是什么☒型的☒☒。只有在☒个上下文中使用它才会被替☒。☒参☒ PAGC 令牌。

12.1.3.3 gaz table

gaz table — gaz 表用于☒准化地名，并将☒☒入与 (a) ☒入令牌（☒参☒“the section called “☒入令牌”” 一☒）和 (b) ☒准化表示相关☒。

描述

gaz（地名☒典的☒写）表用于☒准化地名，并将☒☒入与称☒“the section called “☒入令牌”” 的部分和 (b) ☒准化表示相关☒。例如，如果您在美国，您可以加☒州名称和相关☒写。

gaz 表中至少包含以下列。如果您愿意出于自己的目的，可以添加更多列。

id 表的主☒

seq 整数：定☒号？- 用于☒☒☒☒例的☒☒符

word 文本：☒入的☒☒

stdword text：☒准化的替☒☒

token 整数：它是什么☒型的☒☒。只有在☒个上下文中使用它才会被替☒。☒参☒ PAGC 令牌。

12.1.4 地址☒准化器功能

12.1.4.1 debug_standardize_address

debug_standardize_address — 返回 json 格式的文本，列出解析☒☒和☒准化

Synopsis

text **debug_standardize_address**(text lextab, text gaztab, text rultab, text micro, text macro=NULL);

描述

□是一个用于□□地址□准化器□□和 lex/gaz 映射的函数。它返回一个 json 格式的文本，其中包括匹配□□、令牌映射以及使用 lex table的表名称、gaz table和rules table的表名称和地址的□入地址的最佳□准化地址stdaddr形式。

□于□行地址，□使用 `micro`

□于□行地址，一个是由 `micro` 的□政地址第一行□成，例如 `house_num street`，□一个是由□准的□政地址第二行□成，例如 `city, state postal_code country`。

json 文档中返回的元素是

`input_tokens` □于□入地址中的每个□□，返回□□□的位置、□□□的□□□□以及它映射到的□准□□。□ 注意，□于某些□入□□，您可能会返回多个□□，因□某些□入可以□□□多个事物。

`rules` 与□入匹配的□□集以及每个□□□□的分数。第一条□□（最高分）用于□准化

`stdaddr` □行 standardize_address□将返回的□准化地址元素 stdaddr

可用性：3.4.0

□方法需要 address_standardizer □展。

示例

使用 address_standardizer_data_us □展

```
CREATE EXTENSION address_standardizer_data_us; -- only needs to be done once
```

□体 1：□行地址并返回□入□□

```
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)
```

□体 2：多行地址并返回第一条□□□入映射和分数

```
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)
```

相关信息

stdaddr, rules table, lex table, gaz table, Pagc_Normalize_Address

12.1.4.2 parse_address

parse_address — ☒取 1 行地址并分成几部分

Synopsis

record **parse_address**(text address);

描述

Returns 将地址作☒☒入，并返回由字段 *num*、*street*、*street2*、*address1*、*city*、*state*、*zip*、*zipplus*、*country* ☒成的☒☒☒出。

可用性：2.2.0

☒方法需要 address_standardizer ☒展。

示例

☒一地址

```
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
```

地址表

```
-- 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)
```

相关信息

12.1.4.3 standardize_address

standardize_address — 利用 lex、gaz 和□□表返回□入地址的 stdaddr 形式。

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);

描述

使用 lex table的表名称、gaz table和rules table的表名称和地址返回□入地址的 stdaddr 形式。

□体 1：将地址□□一行。

□体 2：将地址分□ 2 部分。由 `micro` 第一行□政地址□成的微信，例如 `house_num street`，以及由地址的□准□政第二行□成的宏，例如 `city, state postal_code country`。

可用性：2.2.0

□方法需要 address_standardizer □展。

示例

使用 address_standardizer_data_us □展

```
CREATE EXTENSION address_standardizer_data_us; -- only needs to be done once
```

⊠体 1：⊠行地址。⊠不适用于非美国地址

```
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
```

使用与 Tiger 地理⊠⊠器打包的表格。此示例⊠在您安装了 postgis_tiger_geocoder ⊠才有效。

```
SELECT *  FROM standardize_address('tiger.pagc_lex',
         'tiger.pagc_gaz', 'tiger.pagc_rules', 'One Devonshire Place, PH 301, Boston, MA  ↩
            02109-1234');
```

⊠了更容易⊠⊠，我⊠将使用 hstore ⊠展⊠⊠⊠出 CREATE EXTENSION hstore; 你需要安装

```
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)
```

⊠体 2：作⊠⊠部分地址

```
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)
```

相关信息

stdaddr, rules table, lex table, gaz table, Pagc_Normalize_Address

12.2 Tiger 地理⊠⊠器

PostGIS ⊠有其他几个开源地理⊠⊠器，与 Tiger 地理⊠⊠器不同，它⊠具有多国家地理⊠⊠支持的⊠⊠

- Nominatim 使用 OpenStreetMap 地名⊠典格式的数据。它需要 osm2pgsql 来加⊠数据、PostgreSQL 8.4 + 和 PostGIS 1.5+ 才能⊠行。它被打包⊠ Web 服⊠接口，并且似乎被⊠⊠⊠作⊠ Web 服⊠来⊠用。就像 Tiger 地理⊠⊠器一⊠，它同⊠具有地理⊠⊠器和反向地理⊠⊠器⊠件。从文档来看，尚不清楚它是否具有像 Tiger 地理⊠⊠器一⊠的⊠ SQL 接口，或者大量⊠⊠是否在 Web 接口中⊠⊠。
- GIS Graphy ⊠利用 PostGIS，并且像 Nominatim 一⊠使用 OpenStreetMap (OSM) 数据。它配⊠了一个加⊠器来加⊠ OSM 数据，并且与 Nominatim ⊠似，不⊠能⊠⊠行美国地理⊠⊠。与 Nominatim 非常相似，它作⊠ Web 服⊠⊠行并依⊠于 Java 1.5、Servlet ⊠用程序、Solr。GisGraphy 是跨平台的，并且⊠具有反向地理⊠⊠器以及其他一些⊠⊠的功能。

12.2.1 Drop_Indexes_Generate_Script

Drop_Indexes_Generate_Script — 生成一个脚本，⊠除 Tiger 架⊠和用⊠指定架⊠上的所有非主⊠和非唯一索引。如果未指定架⊠，⊠默⊠架⊠⊠ `Tiger_data`。

Synopsis

text **Drop_Indexes_Generate_Script**(text param_schema=tiger_data);

描述

生成一个脚本，⊠除 Tiger 架⊠和用⊠指定架⊠上的所有非主⊠和非唯一索引。如果未指定架⊠，⊠默⊠架⊠⊠ `Tiger_data`。

⊠⊠于最大限度地⊠少索引膨⊠很有用,索引膨⊠可能会混淆⊠⊠⊠划器或占用不必要的空⊠。与 Install_Missing_Indexes 合使用，⊠添加地理⊠⊠器使用的索引。

可用性: 2.0.0

示例

```
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;
:
:
```

相关信息

Install_Missing_Indexes, Missing_Indexes_Generate_Script

12.2.2 Drop_Nation_Tables_Generate_Script

Drop_Nation_Tables_Generate_Script — 生成一个脚本，⊠除指定架⊠中以 `County_all`、`state_all` 或 state 代⊠开⊠，后跟 `county` 或 `state` 的所有表。

Synopsis

text **Drop_Nation_Tables_Generate_Script**(text param_schema=tiger_data);

描述

生成一个脚本，⊠除指定架⊠中以 `County_all`、`state_all` 或 state 代⊠开⊠，后跟 `county` 或 `state` 的所有表。如果您要从 `Tiger_2010` 数据升⊠到 `Tiger_2011` 数据，⊠需要⊠行此操作。

可用性：2.1.0

示例

```
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;
```

相关信息

Loader_Generate_Nation_Script

12.2.3 Drop_State_Tables_Generate_Script

Drop_State_Tables_Generate_Script — 生成一个脚本，⊠脚本⊠除指定架⊠中以 state ⊠写⊠前⊠的所有表。如果未指定架⊠，⊠默⊠架⊠⊠ `Tiger_data`。

Synopsis

text **Drop_State_Tables_Generate_Script**(text param_state, text param_schema=tiger_data);

描述

生成一个脚本，⊠脚本⊠除指定架⊠中以 state ⊠写⊠前⊠的所有表。如果未指定架⊠，⊠默⊠架⊠⊠ `Tiger_data`。此函数⊠于在重新加⊠状⊠之前⊠除状⊠表非常有用，以防在上次加⊠期⊠出⊠⊠⊠。

可用性: 2.0.0

示例

```
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;
```

相关信息

Loader_Generate_Script

12.2.4 Geocode

Geocode — 将地址作⊠字符串（或其他⊠准化地址）⊠入，并⊠出一⊠可能的位置，其中包括 NAD 83 ⊠⊠度中的点几何⊠形、每个位置的⊠准化地址以及⊠⊠。⊠⊠越低，匹配的可能性越大。⊠果首先按最低⊠分排序。可以⊠⊠⊠入最大⊠果数，默⊠⊠ 10，以及 restrict_region（默⊠⊠ NULL）

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);

描述

将地址作⊠字符串（或已⊠⊠准化的地址）并⊠出一⊠可能的位置，其中包括 NAD 83 ⊠⊠度中的点几何形状、每个位置的⊠准化地址 (addy) 以及⊠⊠。⊠⊠越低，匹配的可能性越大。⊠果首先按最低⊠分排序。使用 Tiger 数据（edges、faces、addr）、PostgreSQL 模糊字符串匹配（soundex、levenshtein）和 PostGIS ⊠插⊠函数沿 Tiger ⊠⊠插⊠地址。⊠⊠越高，地理⊠⊠正确的可能性就越小。地理⊠⊠点默⊠⊠从中心⊠到街道地址所在一⊠ (L/R) 的偏移 10 米。

增⊠功能：2.0.0 支持 Tiger 2010 ⊠⊠化数据并修改了一些⊠⊠以提高地理⊠⊠的速度和准确性，并将点从中心⊠偏移到街道地址所在的一⊠。新参数 `max_results` 可用于指定最佳⊠果的数量或⊠返回最佳⊠果。

示例：基本

下面的示例⊠⊠是在一台 3.0 GHZ ⊠⊠理器 Windows 7 ⊠算机上⊠行的，⊠⊠算机具有 2GB 内存，⊠行 PostgreSQL 9.1rc1/PostGIS 2.0，加⊠了所有 MA、MN、CA、RI state Tiger 数据。

精确匹配的⊠算速度更快（61 毫秒）

```
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
```

即使 zip 没有⊠入地理⊠⊠器也可以猜⊠（大⊠需要 122-150 毫秒）

```
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
```

可以⊠理拼写⊠⊠，并提供不止一种可能的解决方案和⊠⊠，并且需要更⊠的⊠⊠（500ms）。

```
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
```

用于☒地址☒行批量地理☒☒。最☒☒的方法是☒置 max_results=1。☒☒理尚未地理☒☒的内容（没有☒☒）。

```
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)
```

示例：使用几何☒☒器

```
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
```

相关信息

Normalize_Address, Pprint_Addy, ST_AsText, ST_SnapToGrid, ST_X, ST_Y

12.2.5 Geocode_Intersection

Geocode_Intersection — 接收 2 条相交的街道以及 state、city、zip，并☒出位于交叉路口的第一个交叉街道上的一☒可能位置，☒包括一个 geomout 作☒ NAD 83 ☒☒度中的点位置，一个☒准化地址 (addy ）每个位置以及☒☒。☒☒越低，匹配的可能性越大。☒果首先按最低☒分排序。可以☒☒☒入最大☒果，默☒☒ 10。使用 Tiger 数据（edges、faces、addr）、PostgreSQL 模糊字符串匹配（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);

描述

☒函数接受☒条交叉的街道、一个 state、city、zip 作☒☒入，并☒出位于交叉口上的第一条街道上的可能位置集合。每个位置包括 NAD 83 ☒☒度的点几何、每个位置的☒范化地址以及☒分。☒分越低，匹配度越高。☒果按照☒分从低到高排序。☒可以☒☒☒入最大☒果数，默☒☒ 10。函数返回每个☒果的☒准化地址（addy）、NAD 83 ☒☒度的点位置（geomout）以及☒分。☒分越低，匹配度越高。☒果按照☒分从低到高排序。此函数使用 Tiger 数据（edges、faces、addr）和 PostgreSQL 模糊字符串匹配（soundex、levenshtein）☒行☒算

可用性: 2.0.0

示例：基本

下面的示例☒☒是在一台 3.0 GHZ ☒☒理器 Windows 7 ☒算机上☒行的，☒☒算机具有 2GB RAM，☒行 PostgreSQL 9.0/PostGIS 1.5，并加☒了所有 MA 状☒ Tiger 数据。目前有点慢（3000 毫秒）

在 Windows 2003 64 位 8GB 上☒☒ PostGIS 2.0 PostgreSQL 64 位 Tiger 2011 数据加☒ -- (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
```

即使没有在地理⊠⊠器中⊠⊠ zip，也可以猜⊠（在 Windows 7 上大⊠花⊠了 3500 毫秒），在 Windows 2003 64 位上花⊠了 741 毫秒

```
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
```

相关信息

Geocode, Pprint_Addy, ST_AsText

12.2.6 Get_Geocode_Setting

Get_Geocode_Setting — 返回存⊠在 Tiger.geocode_settings 表中的特定⊠置的⊠。

Synopsis

text **Get_Geocode_Setting**(text setting_name);

描述

返回存⊠在 Tiger.geocode_settings 表中的特定⊠置的⊠。⊠置允⊠您切⊠功能的⊠⊠。稍后的⊠划将是通⊠⊠置来控制⊠⊠。当前⊠置列表如下：

```
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
```

更改：2.2.0：默⊠⊠置⊠在保存在名⊠ geocode_settings_default 的表中。使用自定⊠⊠置 a 位于 geocode_settings 中，并且⊠包含用⊠⊠置的⊠置。

可用性：2.1.0

返回□□□置示例

```
SELECT get_geocode_setting('debug_geocode_address) As result;
result
---------
false
```

相关信息

Set_Geocode_Setting

12.2.7 Get_Tract

Get_Tract — 从几何□形所在的区域表中返回人口普□区域或字段。默□返回区域的短名称。

Synopsis

text **get_tract**(geometry loc_geom, text output_field=name);

描述

□定一个几何□形将返回□几何□形的人口普□区位置。如果未指定空□参考系□，□假定 NAD 83 □□度。

Note
此函数使用人口普□的 tract，默□情况下不会加□。如果您已□加□了 state 表，可以使用 Loader_Generate_Census_Script 脚本加□ tract、bg 和 tabblock。
如果您尚未加□状□数据并希望加□□些附加表，□□行以下操作

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

然后它□将被 Loader_Generate_Script包含。

可用性: 2.0.0

示例：基本

```
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
```

相关信息

Geocode>

12.2.8 Install_Missing_Indexes

Install_Missing_Indexes — 找具有在地理器接和条件中使用的列的所有表，些表缺少些列上使用的索引，并将添加它。

Synopsis

boolean **Install_Missing_Indexes**();

描述

找 `Tiger` 和 `Tiger_data` 架中的所有表，其中包含在地理器接和器中使用的关列，些列上缺少索引，并将出 SQL DDL 来定些表的索引，然后行生成的脚本。是一个助函数，它添加了使更快所需的新索引，而些索引可能在加程中失。此函数是Missing_Indexes_Generate_Script的配套函数，除了生成建索引脚本之外，行它。它作 `update_geocode.sql` 升脚本的一部分被用。

可用性: 2.0.0

示例

```
SELECT install_missing_indexes();
         install_missing_indexes
-------------------------
 t
```

相关信息

Loader_Generate_Script, Missing_Indexes_Generate_Script

12.2.9 Loader_Generate_Census_Script

Loader_Generate_Census_Script — 指定 states 的指定平台生成 shell 脚本，脚本将下 Tiger 人口普 state 区、bg 和 tabblocks 数据表、存并加到 `Tiger_data` 架中。每个状脚本都作独的返回。

Synopsis

setof text **loader_generate_census_script**(text[] param_states, text os);

描述

生成一个用于指定平台和指定州的 shell 脚本，⊠脚本将下⊠ Tiger 数据中的人口普⊠州 `tract`、分⊠区⊠ `bg` 和 `tabblocks` 数据表，然后将⊠些数据表分⊠段加⊠到 `tiger_data` 模式中。每个州的脚本将作⊠一个⊠独的⊠⊠返回。

它在 Linux 上使用 unzip（默⊠情况下在 Windows 上使用 7-zip），并使用 wget ⊠行下⊠。它使用 Section 4.7.2 来加⊠数据。⊠注意，它的最小⊠理⊠元是整个州。它只会⊠理⊠存和⊠⊠文件⊠中的文件。

它使用以下控制表来控制⊠程和不同的操作系⊠ shell ⊠法⊠化。

1. `loader_variables` 跟踪各种⊠量，例如人口普⊠站点、年份、数据和⊠存模式
2. `loader_platform` 各种平台的配置文件以及各种可⊠行文件所在的位置。自⊠ windows 和 linux。⊠可以添加更多。
3. `loader_lookuptables` 每条⊠⊠定⊠一种表（州、⊠）、是否⊠理其中的⊠⊠以及如何加⊠它⊠。定⊠每个⊠⊠的⊠入数据、⊠存数据、添加、⊠除列、索引和⊠束的步⊠。每个表都以状⊠⊠前⊠，并⊠承自 Tiger 模式中的表。例如⊠建从 `tiger.faces` ⊠承的 `tiger_data.ma_faces`

可用性: 2.0.0

Note
Loader_Generate_Script 包含此⊠⊠，但如果您在 PostGIS 2.0.0 alpha5 之前安装了 Tiger 地理⊠⊠器，⊠需要在已完成的状⊠上⊠行此⊠⊠以⊠得⊠些附加表。

示例

生成脚本以加⊠ Windows shell 脚本格式的⊠定状⊠的数据。

```
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');”
:
```

生成 sh 脚本

```
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
:
:
```

相关信息

Loader_Generate_Script

12.2.10 Loader_Generate_Script

Loader_Generate_Script — 区指定平台的指定状区生成 shell 脚本，区脚本将下区 Tiger 数据、区存并加区到 Tiger_data 模式中。每个状区脚本都作区区独的区区返回。最新版本支持 Tiger 2010 区区区化，区加区人口普区区、区区区和区区表。

Synopsis

setof text **loader_generate_script**(text[] param_states, text os);

描述

区指定平台的指定状区生成 shell 脚本，区脚本将下区 Tiger 数据、区存并加区到 Tiger_data 模式中。每个状区脚本都作区区独的区区返回。

它在 Linux 上使用 unzip（在 Windows 上默⍰⍰ 7-zip）和 wget 来⍰行下⍰。它使用 Section 4.7.2来加⍰数据。⍰注意，它的最小⍰位是整个状⍰，但您可以通⍰自己下⍰文件来覆盖它。它只会⍰理⍰存文件⍰和⍰⍰文件⍰中的文件。

它使用以下控制表来控制⍰程和不同的操作系⍰ shell ⍰法⍰化。

1. `loader_variables` 跟踪各种⍰量，例如人口普⍰站点、年份、数据和⍰存模式
2. `loader_platform` 各种平台的配置文件以及各种可⍰行文件所在的位置。自⍰ windows 和 linux。⍰可以添加更多。
3. `loader_lookuptables` 每条⍰⍰定⍰一种表（州、⍰）、是否⍰理其中的⍰⍰以及如何加⍰它⍰。定⍰每个⍰⍰的⍰入数据、⍰存数据、添加、⍰除列、索引和⍰束的步⍰。每个表都以状⍰⍰前⍰，并⍰承自 Tiger 模式中的表。例如⍰建从 `tiger.faces` ⍰承的 `tiger_data.ma_faces`

可用性：2.0.0 支持 Tiger 2010 ⍰⍰化数据并加⍰人口普⍰区（tract）、区⍰⍰（bg）和区⍰（tabblocks）表。

Note
如果您使用 pgAdmin 3，⍰注意默⍰情况下 pgAdmin 3 会截断⍰文本。要修复此⍰⍰，⍰更改文件 *File -> Options ->Query Tool -> Query Editor ->Max.* 每列字符数大于 50000 个字符。

示例

使用 psql，其中 gitest 是您的数据⍰，`/gisdata/data_load.sh` 是使用要⍰行的 shell 命令⍰建的文件。

```
psql -U postgres -h localhost -d gistest -A -t \
 -c "SELECT Loader_Generate_Script(ARRAY['MA'], 'gistest')" > /gisdata/data_load.sh;
```

生成脚本以加⍰ Windows shell 脚本格式的 2 个状⍰的数据。

```
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
:
:
```

生成 sh 脚本

```
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}/*.*
:
:
```

相关信息

Section 2.4.1, Loader_Generate_Nation_Script, Drop_State_Tables_Generate_Script

12.2.11 Loader_Generate_Nation_Script

Loader_Generate_Nation_Script — 区指定平台生成加区到区和州区找表中的 shell 脚本。

Synopsis

text **loader_generate_nation_script**(text os);

描述

区指定平台生成 shell 脚本，将 `County_all`、`county_all_lookup`、`state_all` 表加区到 `Tiger_data` 架区中。它区分区区承自 `tiger` 模式中的 `County`、`county_lookup`、`state` 表。

它在 Linux 上使用 unzip（在 Windows 上默区区 7-zip）和 wget 来区行下区。它使用 Section 4.7.2来加区数据。

它使用以下控制表 `Tiger.loader_platform`、`tiger.loader_variables` 和 `Tiger.loader_lookuptables` 来控制区程和不同操作系区 shell 区法区化。

1. `loader_variables` 跟踪各种区量，例如人口普区站点、年份、数据和区存模式
2. `loader_platform` 各种平台的配置文件以及各种可区行文件所在的位置。区有 windows 和 linux/unix。区可以添加更多。
3. `loader_lookuptables` 每条区区定区一种表（州、区）、是否区理其中的区区以及如何加区它区。定区每个区区的区入数据、区存数据、添加、区除列、索引和区束的步区。每个表都以状区区前区，并区承自 Tiger 模式中的表。例如区建从 `tiger.faces` 区承的 `tiger_data.ma_faces`

增区：2.4.1 区政区区 5 制表区域 (zcta5) 加区步区已修复，区用后，zcta5 数据将作区名区 zcta5_all 的区个表加区，作区国家脚本加区的一部分。

可用性：2.1.0

Note
如果您希望将⊠政⊠⊠ 5 制表区域 (zcta5) 包含在您的国家脚本加⊠中，⊠⊠行以下操作：

```
UPDATE tiger.loader_lookuptables SET load = true WHERE table_name = 'zcta510';
```

Note
如果您⊠行的是 `Tiger_2010` 版本，并且想要使用⊠新的 Tiger 数据重新加⊠⊠状⊠，⊠在⊠行此脚本之前，您需要首先加⊠生成并⊠行 drop ⊠句Drop_Nation_Tables_Generate_Script。

示例

生成脚本来加⊠ Windows 的国家数据。

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

生成脚本来加⊠ Linux/Unix 系⊠的数据。

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

相关信息

Loader_Generate_Script, Drop_Nation_Tables_Generate_Script

12.2.12 Missing_Indexes_Generate_Script

Missing_Indexes_Generate_Script — ⊠找具有在地理⊠⊠器⊠接中使用的⊠列的所有表，⊠些表缺少⊠些列上的索引，并将⊠出 SQL DDL 来定⊠⊠些表的索引。

Synopsis

text **Missing_Indexes_Generate_Script**();

描述

⊠找 `tiger` 和 `Tiger_data` 架⊠中的所有表，其中包含在地理⊠⊠器⊠接中使用的关⊠列，⊠些列上缺少索引，并将⊠出 SQL DDL 来定⊠⊠些表的索引。⊠是一个⊠助函数，它添加了使⊠⊠更快所需的新索引，而⊠些索引可能在加⊠⊠程中⊠失。随着地理⊠⊠器的改⊠，⊠函数将更新以适⊠正在使用的新索引。如果此函数没有⊠出任何内容，⊠意味着您的所有表都具有我⊠⊠⊠的关⊠索引。

可用性：2.0.0

示例

```
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);
```

相关信息

Loader_Generate_Script, Install_Missing_Indexes

12.2.13 Normalize_Address

Normalize_Address — 区定文本街道地址，返回复合的 norm_addy 区型，区区型具有道路后区、前区和区准化区型、街道、街道名称等，分区区独的字段。区函数区适用于与 Tiger_geocoder 打包的区找数据（不需要 tiger 普区数据）。

Synopsis

norm_addy **normalize_address**(varchar in_address);

描述

区定文本街道地址，返回复合的 norm_addy 区型，区区型具有道路后区、前区和区准化区型、街道、街道名称等，分区区独的字段。区是地理区区区程中将所有地址区区区区准化区政形式的第一步。除了地理区区器打包的数据之外，不需要其他数据。

此函数区使用使用 Tiger_geocoder 区加区并位于 tiger 模式中的各种方向/状区/后区区找表，因此不需要您下区 tiger 普区数据或任何其他附加数据来使用它。您可能会区区需要向 Tiger 模式中的各种区找表添加更多区写或替代命名。

它使用位于 tiger 模式中的各种控制区找表来区准化区入地址。

区函数返回的 norm_addy 区型区象中的字段按以下区序排列，其中 () 表示地理区区器所需的字段，[] 表示可区字段：

(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip] [parsed] [zip4] [address_alphanumeric]

增区：2.4.0norm_addy 区象包括附加字段 zip4 和 address_alphanumeric。

1. address 是一个整数：街道号区
2. predirAbbrev 是 varchar：道路的方向前区，例如 N、S、E、W 等。区些是使用 Direction_lookup 表区行控制的。

3. streetName 是 varchar
4. streetTypeAbbrev varchar 街道⊠型的⊠写版本：例如 St, Ave, Cir。⊠些是使用 street_type_lookup 表⊠行控制的。
5. postdirAbbrev varchar ⊠型是道路 N、S、E、W 等方向的⊠写。⊠些是使用 Direction_lookup 表⊠行控制的。
6. internalvarchar ⊠型内部地址，例如公寓或套房号。
7. location varchar ⊠型，通常是城市或管⊠省份。
8. stateAbbrev varchar ⊠型⊠个字符的美国州。例如 MA, NY, MI。⊠些由 state_lookup 表控制。
9. zip varchar ⊠型，5 位⊠政⊠⊠。例如 02109。
10. parsed boolean ⊠型 - 表示地址是否是由⊠准化⊠程形成的。Normalize_address 函数在返回地址之前将其⊠置⊠ true。
11. zip49 位⊠政⊠⊠的最后 4 位数字。可用性：PostGIS 2.4.0。
12. address_alphanumeric 完整的街道号⊠,即使它具有字母字符(如 17R)。⊠此的解析最好使用Pagc_Normalize_A函数。可用性：PostGIS 2.4.0。

示例

⊠出⊠⊠字段。如果您想要漂亮的文本⊠出，⊠使用 Pprint_Addy。

```
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  |
```

相关信息

Geocode, Pprint_Addy

12.2.14 Pagc_Normalize_Address

Pagc_Normalize_Address — ⊠定文本街道地址，返回复合的 norm_addy ⊠型，⊠⊠型具有道路后⊠、前⊠和⊠准化⊠型、街道、街道名称等，分⊠⊠独的字段。⊠函数⊠适用于与 Tiger_geocoder 打包的⊠找数据（不需要 tiger 普⊠数据）。需要 address_standardizer ⊠展。

Synopsis

norm_addy **pagc_normalize_address**(varchar in_address);

描述

⊠定文本街道地址，返回复合的 norm_addy ⊠型，⊠⊠型具有道路后⊠、前⊠和⊠准化⊠型、街道、街道名称等，分⊠⊠独的字段。⊠是地理⊠⊠⊠程中将所有地址⊠⊠⊠⊠准化⊠政形式的第一步。除了地理⊠⊠器打包的数据之外，不需要其他数据。

⊠函数⊠使用各种 pagc_* ⊠找表，⊠些表由 Tiger_geocoder ⊠加⊠并位于 Tiger 模式中，因此不需要您下⊠ tiger 普⊠数据或任何其他附加数据来使用它。您可能会⊠⊠需要向 Tiger 模式中的各种⊠找表添加更多⊠写或替代命名。

它使用位于 tiger 模式中的各种控制⊠找表来⊠准化⊠入地址。

⊠函数返回的 norm_addy ⊠型⊠象中的字段按以下⊠序排列，其中 () 表示地理⊠⊠器所需的字段，[] 表示可⊠字段：

Normalize_Address 的大小写和格式略有不同。

可用性：2.1.0

⊠方法需要 address_standardizer ⊠展。

(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip]

目前，address_standardizer ⊠展的本机 standardaddr 比 norm_addy 丰富一些，因⊠它旨在支持国⊠地址（包括国家/地区）。standardaddr 等效字段是：

house_num,predir, name, suftype, sufdir, unit, city, state, postcode

增⊠：2.4.0norm_addy ⊠象包括附加字段 zip4 和 address_alphanumeric。

1. address 是一个整数：街道号⊠
2. predirAbbrev 是 varchar：道路的方向前⊠，例如 N、S、E、W 等。⊠些是使用 Direction_lookup 表⊠行控制的。
3. streetName 是 varchar
4. streetTypeAbbrev varchar 街道⊠型的⊠写版本：例如 St, Ave, Cir。⊠些是使用 street_type_lookup 表⊠行控制的。
5. postdirAbbrev varchar ⊠型是道路 N、S、E、W 等方向的⊠写。⊠些是使用 Direction_lookup 表⊠行控制的。
6. internalvarchar ⊠型内部地址，例如公寓或套房号。
7. location varchar ⊠型，通常是城市或管⊠省份。
8. stateAbbrev varchar ⊠型⊠个字符的美国州。例如 MA, NY, MI。⊠些由 state_lookup 表控制。
9. zip varchar ⊠型，5 位⊠政⊠⊠。例如 02109。
10. parsed boolean ⊠型 - 表示地址是否是由⊠准化⊠程形成的。Normalize_address 函数在返回地址之前将其⊠置⊠ true。
11. zip49 位⊠政⊠⊠的最后 4 位数字。可用性：PostGIS 2.4.0。
12. address_alphanumeric 完整的街道号⊠,即使它具有字母字符(如 17R)。⊠此的解析最好使用Pagc_Normalize_A函数。可用性：PostGIS 2.4.0。

示例

⊠次⊠用示例

```
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
```

批量⊠用。目前，postgis_tiger_geocoder 包装 address_standardizer 的方式存在速度⊠⊠。⊠些⊠⊠有望在以后的版本中得到解决。⊠了解决⊠些⊠⊠，如果您需要批量地理⊠⊠的速度以批量模式⊠用生成 normaddy，建⊠您直接⊠用 address_standardizer standardize_address 函数，如下所示，⊠与我⊠在Normalize_Address中所做的⊠似，使用在Geocode中⊠建的数据。

```
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
```

相关信息

Normalize_Address, Geocode

12.2.15 Pprint_Addy

Pprint_Addy — ⊠定一个 norm_addy 复合⊠型⊠象，返回它的漂亮的打印表示。通常与 normalize_address ⊠合使用。

Synopsis

varchar **pprint_addy**(norm_addy in_addy);

描述

⊠定一个 norm_addy 复合⊠型⊠象，返回它的漂亮的打印表示。除了地理⊠⊠器打包的数据之外，不需要其他数据。通常与 Normalize_Address ⊠合使用。

示例

漂亮地打印一个地址

```
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
```

漂亮的打印地址地址表

```
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
```

相关信息

Normalize_Address

12.2.16 Reverse_Geocode

Reverse_Geocode — ☒取已知空☒参考系☒中的几何点并返回一条☒☒，其中包含理☒上可能的地址数☒和交叉街道数☒。如果 include_strnum_range = true，☒包括交叉街道中的街道范☒。

Synopsis

record **Reverse_Geocode**(geometry pt, boolean include_strnum_range=false, geometry[] OUT intpt, norm_addy[] OUT addy, varchar[] OUT street);

描述

☒取已知空☒参考中的几何点并返回一条☒☒，其中包含理☒上可能的地址数☒和十字街道数☒。如果 include_strnum_range = true，☒包括交叉街道中的街道范☒。如果未☒入，☒ include_strnum_range 默☒☒ false。地址根据点最接近的道路☒行排序，因此第一个地址很可能是正确的地址。

☒什么我☒☒理☒地址而不是☒☒地址。Tiger 数据没有真☒地址，只有街道范☒。因此，理☒地址是基于街道范☒的内插地址。例如，插入我的地址之一会返回 26 Court St. 和 26 Court Sq.，尽管不存在 26 Court Sq. ☒☒的地方。☒是因☒一个点可能位于 2 条街道的拐角☒，因此☒☒会沿着☒条街道☒行插☒。☒☒☒☒假☒地址沿着街道均匀分布，☒当然是☒☒的，因☒您可以☒市政建筑占据街道范☒的很大一部分，而其余建筑物☒聚集在末端。

注意：☒，☒个功能依☒于 Tiger 数据。如果您☒没有加☒覆盖☒点区域的数据，那么您将得到一条充☒ NULL 的☒☒。

返回的☒☒元素如下：

1. `intpt` 是一个点数⊠：⊠些是街道上最接近⊠入点的中心⊠点。有多少个地址就有多少个点。
2. `addy` 是一个 norm_addy（⊠范化地址）数⊠：⊠些是适合⊠入点的可能地址数⊠。数⊠中的第一个最有可能。一般来⊠⊠⊠只有一个，除非某个点位于 2 或 3 条街道的拐角⊠，或者⊠点位于道路上的某个位置而不是在⊠面。
3. `street varchar` 数⊠：⊠些是交叉街道（或街道）（相交的街道或者是点投影所在的街道）。

增⊠：2.4.1 如果加⊠了可⊠的 zcta5 数据集，即使未加⊠特定的状⊠数据，reverse_geocode 函数也可以解析⊠状⊠和 zip。有关加⊠ zcta5 数据的⊠⊠信息，⊠参⊠ Loader_Generate_Nation_Script。

可用性: 2.0.0

示例

位于⊠条街道拐角⊠但最接近一条街道的点示例。⊠是 MIT 的大致位置：77 Massachusetts Ave, Cambridge, MA 02139 ⊠注意，⊠然我⊠没有 3 条街道，但 PostgreSQL 只会⊠高于上限的条目返回 null，因此可以安全使用。⊠包括街道范⊠

```
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
```

在⊠里，我⊠⊠⊠不包括十字路口的地址范⊠，并⊠⊠了一个非常靠近⊠条街道拐角⊠的位置，因此可以通⊠⊠个不同的地址来⊠⊠。

```
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
```

⊠于⊠一点，我⊠重用Geocode 中的地理⊠⊠示例，并且只需要主要地址和最多 2 个交叉街道。

```
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
```

相关信息

Pprint_Addy, Geocode, Loader_Generate_Nation_Script

12.2.17 Topology_Load_Tiger

Topology_Load_Tiger — 将 tiger 数据的定区域加到 PostGIS 拓扑中，并将老虎数据拓扑的空参考，并捕捉到拓扑的精度公差。

Synopsis

text **Topology_Load_Tiger**(varchar topo_name, varchar region_type, varchar region_id);

描述

将定的 tiger 数据区域加到 PostGIS 拓扑中。面、点和将目拓扑的空参考系，点将捕捉到目拓扑的容差。建的面、点、与原始 Tiger 数据面、点、保持相同的 ID，以便数据集将来可以更松地与 Tiger 数据行。返回有关流程的摘要信息。

例如，于重新划分数据非常有用，在种情况下，您需要新形成的多形遵循街道的中心，并且生成的多形不重。

Note

功能依 Tiger 数据以及 PostGIS 拓扑模的安装。有关更多信息，参 Chapter 9和 Section 2.2.3。如果您尚未加覆盖感趣区域的数据，不会建任何拓扑。如果您没有使用拓扑函数建拓扑，此函数也会失。

Note

大多数拓扑都是由于公差造成的，即后点没有完全或重。了正种情况，如果拓扑失，您可能需要提高或降低精度。

所需参数：

1. `topo_name` 要加数据的有 PostGIS 拓扑的名称。
2. `Region_type` 界区域的型。目前支持 `place` 和 `county`。划是再多几个。是用于定区域界的表格。例如，`tiger.place`、`tiger.county`
3. `Region_id` 就是 TIGER 所的大地水准面。它是表中区域的唯一符。于 place，它是 `Tiger.place` 中的 `plcidfp` 列。于 county，它是 `Tiger.county` 中的 `cntyidfp` 列

可用性: 2.0.0

示例：**Boston, Massachusetts** 拓扑

在 Mass State Plane Feet (2249) 中⍰建⍰⍰⍰塞州波士⍰的拓扑，容差⍰ 0.25 英尺，然后加⍰到波士⍰城市 tiger 面、⍰⍰、⍰点。

```
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
-------------------+----------+-----------
```

示例：**Suffolk, Massachusetts** 拓扑

在 Mass State Plane Meters (26986) 中⍰建⍰⍰⍰塞州⍰福克市的拓扑，容差⍰ 0.25 米，然后加⍰到⍰福克⍰ tiger 面、⍰、⍰点。

```
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
```

相关信息

CreateTopology, CreateTopoGeom, TopologySummary, ValidateTopology

12.2.18 Set_Geocode_Setting

Set_Geocode_Setting — 置影地理器功能行的置。

Synopsis

text **Set_Geocode_Setting**(text setting_name, text setting_value);

描述

置存在 `Tiger.geocode_settings` 表中的特定置的。置允您切功能的。稍后的划将是通置来控制。当前置列表列在 Get_Geocode_Setting 中。

可用性：2.1.0

返回置示例

如果您在函数 true 行Geocode，NOTICE 日志将出和。

```
SELECT set_geocode_setting('debug_geocode_address', 'true') As result;
result
---------
true
```

相关信息

Get_Geocode_Setting

Chapter 13

PostGIS 特殊函数索引

13.1 PostGIS 聚合函数

下面的函数是空⊠聚合函数，其使用方式与 SQL 聚合函数（例如 `sum` 和 `average`）相同。

- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- ST_3DExtent - 返回几何⊠形的 3D ⊠界框的聚合函数。
- ST_3DUnion - ⊠行 3D ⊠合。
- ST_AsFlatGeobuf - 返回一⊠行的 FlatGeobuf 表示形式。
- ST_AsGeobuf - 返回一⊠行的 Geobuf 表示。
- ST_AsMVT - 返回一⊠行的 MVT 表示形式的聚合函数。
- ST_ClusterDBSCAN - 使用 DBSCAN 算法返回每个⊠入几何⊠形的簇 id 的窗口函数。
- ST_ClusterIntersecting - 将⊠入几何⊠形聚⊠成⊠接集的聚合函数。
- ST_ClusterIntersectingWin - 窗口函数，返回每个⊠入几何⊠形的簇 ID，将⊠入几何⊠形聚⊠到⊠接的集合中。
- ST_ClusterKMeans - 使用 K 均⊠算法返回每个⊠入几何⊠形的簇 id 的窗口函数。
- ST_ClusterWithin - 按⊠隔距离⊠几何⊠形⊠行聚合的聚合函数。
- ST_ClusterWithinWin - 窗口函数，返回每个⊠入几何⊠形的簇 ID，使用分离距离⊠行聚⊠。
- ST_Collect - 从一⊠几何⊠形⊠建 GeometryCollection 或 Multi* 几何⊠形。
- ST_CoverageInvalidEdges - 用于⊠找多⊠形无法形成有效覆盖范⊠的位置的窗口函数。
- ST_CoverageSimplify - ⊠化多⊠形覆盖范⊠⊠⊠的窗口函数。
- ST_CoverageUnion - 通⊠⊠除共享⊠来⊠算形成覆盖范⊠的一⊠多⊠形的并集。
- ST_Extent - 返回几何⊠形⊠界框的聚合函数。
- ST_MakeLine - 从 Point, MultiPoint, 或 LineString geometries ⊠建 LineString。
- ST_MemUnion - 聚合函数，以内存高效但速度⊠慢的方式⊠合几何⊠形
- ST_Polygonize - ⊠算由一⊠几何⊠形的⊠条形成的多⊠形集合。
- ST_SameAlignment - 如果⊠格具有相同的⊠斜、比例、空⊠参考和偏移（像素可以放在同一网格上而不切割成像素），⊠返回 true；如果没有注意⊠⊠⊠⊠，⊠返回 false。

- ST_Union - 聚合表示输入几何体形的点集并集的几何体形。
- ST_Union - 将一组栅格切片的并集返回由 1 个或多个波段组成的单个栅格。
- TopoElementArray_Agg - 返回一个 element_id、类型数组 (topoelements) 的 topoelementarray。

13.2 PostGIS 窗口函数

下面的函数是空间窗口函数，其使用方式与 SQL 窗口函数（例如 `row_number()`、`lead()` 和 `lag()`）相同。它们后面必须跟有 `OVER()` 子句。

- ST_ClusterDBSCAN - 使用 DBSCAN 算法返回每个输入几何体形的簇 id 的窗口函数。
- ST_ClusterIntersectingWin - 窗口函数，返回每个输入几何体形的簇 ID，将输入几何体形聚集到连接的集合中。
- ST_ClusterKMeans - 使用 K 均值算法返回每个输入几何体形的簇 id 的窗口函数。
- ST_ClusterWithinWin - 窗口函数，返回每个输入几何体形的簇 ID，使用分离距离聚行聚集。
- ST_CoverageInvalidEdges - 用于查找多边形无法形成有效覆盖范围的位置的窗口函数。
- ST_CoverageSimplify - 简化多边形覆盖范围边缘的窗口函数。

13.3 PostGIS SQL-MM 兼容函数

下面列出的函数是符合 SQL/MM 3 标准的 PostGIS 函数

- CG_3DArea - 计算 3D 表面几何形状的面积。对于固体将返回 0。
- CG_3DDifference - 行 3D 差异
- CG_3DIntersection - 行 3D 相交
- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- CG_Volume - 计算 3D 体体的体积。如果用于表面（甚至闭合）几何体形将返回 0。
- ST_3DArea - 计算 3D 表面几何形状的面积。对于固体将返回 0。
- ST_3DDWithin - 对两个 3D 几何体形是否在给定的 3D 距离内
- ST_3DDifference - 行 3D 差异
- ST_3DDistance - 返回两个几何体形之间的 3D 笛卡尔最小距离（基于空间参考）（以投影单位表示）。
- ST_3DIntersection - 行 3D 相交
- ST_3DIntersects - 测试两个几何体形在 3D 空间中是否相交 - 适用于点、线串、多边形、多面体曲面（区域）
- ST_3DLength - 返回线性几何体的 3D 长度。
- ST_3DPerimeter - 返回多边形几何体的 3D 周长。
- ST_3DUnion - 行 3D 聚合。
- ST_AddEdgeModFace - 添加新边，如果边边做会分割面，修改原始面并添加新面。
- ST_AddEdgeNewFaces - 添加新边，如果边边做会分割一个面，删除原始面并用两个新面替换它。

- ST_AddIsoEdge - 将由几何 alinestring 定⊠的孤立⊠添加到⊠接⊠个⊠有孤立⊠点 anode 和 anothernode 的拓扑，并返回新⊠的⊠ id。
- ST_AddIsoNode - 将一个孤立的⊠点添加到拓扑⊠⊠中的一个面，并返回新⊠点的⊠点 ID。如果"face" ⊠空（null），仍然会⊠建⊠点。
- ST_Area - 返回多⊠形几何体的面⊠。
- ST_AsBinary - 返回不⊠ SRID 元数据的几何/地理的 OGC/ISO 已知的二⊠制 (WKB) 表示形式。
- ST_AsGML - 将几何⊠形作⊠ GML 版本 2 或 3 元素返回。
- ST_AsText - 返回不⊠ SRID 元数据的几何/地理的已知文本 (WKT) 表示形式。
- ST_Boundary - 返回几何⊠形的⊠界。
- ST_Buffer - ⊠算覆盖距几何体⊠定距离内所有点的几何体。
- ST_Centroid - 返回几何体的几何中心。
- ST_ChangeEdgeGeom - 改⊠⊠的形状而不影⊠拓扑⊠⊠。
- ST_Contains - ⊠⊠ B 的每个点是否都位于 A 中，并且它⊠的内部是否有一个共同点
- ST_ConvexHull - ⊠算几何体的凸包。
- ST_CoordDim - 返回几何体的坐⊠⊠度。
- ST_CreateTopoGeo - 将几何⊠形集合添加到⊠定的空拓扑并返回⊠⊠⊠明成功的消息。
- ST_Crosses - ⊠⊠⊠个几何⊠形是否有一些（但不是全部）共同的内点
- ST_CurveN - Returns the Nth component curve geometry of a CompoundCurve.
- ST_CurveToLine - 将包含曲⊠的几何⊠形⊠⊠⊠⊠性几何⊠形。
- ST_Difference - ⊠算表示几何 A 中不与几何 B 相交的部分的几何。
- ST_Dimension - 返回几何⊠形的拓扑⊠数。
- ST_Disjoint - ⊠⊠⊠个几何⊠形是否没有共同点
- ST_Distance - 返回⊠个几何或地理⊠之⊠的距离。
- ST_EndPoint - 返回 LineString 或 CircularLineString 的最后一个点。
- ST_Envelope - 返回表示几何⊠形⊠界框的几何⊠形。
- ST_Equals - ⊠⊠⊠个几何⊠形是否包含同一⊠点
- ST_ExteriorRing - 返回表示多⊠形外⊠的 LineString。
- ST_GMLToSQL - 从 GML 表示返回指定的 ST_Geometry ⊠。⊠是 ST_GeomFromGML 的⊠名
- ST_GeomCollFromText - 使用⊠定的 SRID 从集合 WKT 中⊠建集合几何体。如果未⊠出 SRID，⊠默⊠⊠ 0。
- ST_GeomFromText - 从已知的文本表示 (WKT) 返回指定的 ST_Geometry ⊠。
- ST_GeomFromWKB - 从已知的二⊠制几何表示 (WKB) 和可⊠的 SRID ⊠建几何⊠例。
- ST_GeometryFromText - 从已知的文本表示 (WKT) 返回指定的 ST_Geometry ⊠。⊠是 ST_GeomFromText 的⊠名
- ST_GeometryN - 返回几何集合的一个元素。
- ST_GeometryType - 以文本形式返回几何⊠形的 SQL-MM ⊠型。
- ST_GetFaceEdges - 返回一⊠有序的⊠，⊠些⊠界⊠定了 aface。

- ST_GetFaceGeometry - 返回⊠定拓扑中具有指定面 ID 的多⊠形。
- ST_InitTopoGeo - ⊠建一个新的拓扑架⊠并将其注册到 topology.topology 表中。
- ST_InteriorRingN - 返回多⊠形的第 N 个内⊠（孔）。
- ST_Intersection - ⊠算表示几何 A 和 B 的共享部分的几何。
- ST_Intersects - ⊠⊠⊠个几何⊠形是否相交（它⊠至少有一个共同点）
- ST_IsClosed - ⊠⊠ LineStrings 的起点和⊠点是否重合。⊠于多面体表面⊠⊠是否⊠合（⊠心）。
- ST_IsEmpty - ⊠⊠几何⊠形是否⊠空。
- ST_IsRing - ⊠⊠⊠串是⊠合的⊠是⊠⊠的。
- ST_IsSimple - ⊠⊠几何体的自完整性或自接触点。
- ST_IsValid - ⊠⊠几何⊠形在 2D 中是否有效。
- ST_Length - 返回⊠性几何体的二⊠⊠度。
- ST_LineFromText - 使用⊠定的 SRID 根据 WKT 表示⊠建几何⊠形。如果未⊠出 SRID，⊠默⊠⊠ 0。
- ST_LineFromWKB - 使用⊠定的 SRID 从 WKB 制作 LINESTRING
- ST_LinestringFromWKB - 使用⊠定的 SRID 从 WKB ⊠建几何⊠形。
- ST_LocateAlong - 返回几何上与⊠量⊠匹配的点。
- ST_LocateBetween - 返回与⊠量范⊠匹配的几何⊠形部分。
- ST_M - 返回点的 M ⊠。
- ST_MLineFromText - 从 WKT 表示形式返回指定的 ST_MultiLineString ⊠。
- ST_MPointFromText - 使用⊠定的 SRID 从 WKT ⊠建几何⊠形。如果未⊠出 SRID，⊠默⊠⊠ 0。
- ST_MPolyFromText - 使用⊠定的 SRID 从 WKT 制作多多⊠形几何体。如果未⊠出 SRID，⊠默⊠⊠ 0。
- ST_ModEdgeHeal - 通⊠⊠除⊠接⊠条⊠的⊠点、修改第一条⊠并⊠除第二条⊠来修复⊠条⊠。返回已⊠除⊠点的 id。
- ST_ModEdgeSplit - 通⊠沿⊠有⊠⊠建新⊠点、修改原始⊠并添加新⊠来分割⊠。
- ST_MoveIsoNode - 在拓扑⊠⊠中将一个孤立⊠点从一个点移⊠到⊠一个点。如果新的 apoint 几何⊠象已存在作⊠一个⊠点，⊠会抛出⊠⊠。返回移⊠的描述。
- ST_NewEdgeHeal - 通⊠⊠除⊠接⊠条⊠的⊠点、⊠除⊠条⊠并用方向与提供的第一条⊠相同的⊠替⊠它⊠来修复⊠条⊠。
- ST_NewEdgesSplit - 通⊠沿⊠有⊠⊠建新⊠点、⊠除原始⊠并用⊠条新⊠替⊠它来分割⊠。返回⊠建的⊠接新⊠的新⊠点的 id。
- ST_NumCurves - Return the number of component curves in a CompoundCurve.
- ST_NumGeometries - 返回几何集合中的元素数量。
- ST_NumInteriorRings - 返回多⊠形的内⊠（孔）数。
- ST_NumPatches - 返回多面体曲面上的面数。⊠于非多面体几何形状将返回 null。
- ST_NumPoints - 返回 LineString 或 CircularString 中的点数。
- ST_OrderingEquals - ⊠⊠⊠个几何⊠形是否表示相同的几何⊠形并且具有相同方向⊠序的点
- ST_Overlaps - ⊠⊠⊠个几何⊠形是否具有相同的⊠度和相交，但每个几何⊠形至少有一个点不在⊠一个几何⊠形中

- ST_PatchN - 返回多□形曲面（PolyhedralSurface）的第 N 个几何体（面）。
- ST_Perimeter - 返回多□形几何或地理的□界□度。
- ST_Point - □建具有 X、Y 和 SRID □的点。
- ST_PointFromText - 使用□定的 SRID 从 WKT □建点几何□形。如果未□出 SRID，□默□□未知。
- ST_PointFromWKB - 使用□定的 SRID 从 WKB □建几何□形
- ST_PointN - 返回几何□形中第一个□串或□形□串中的第 N 个点。
- ST_PointOnSurface - □算保□位于多□形或几何体上的点。
- ST_Polygon - 从具有指定 SRID 的□串□建多□形。
- ST_PolygonFromText - 使用□定的 SRID 从 WKT □建几何□形。如果未□出 SRID，□默□□ 0。
- ST_Relate - □□□个几何□形是否具有与交集矩□模式匹配的拓扑关系，或□算它□的交集矩□
- ST_RemEdgeModFace - □除一条□，如果□□将□个面分开，□□除一个面并修改□一个面以覆盖□个面的空□。
- ST_RemEdgeNewFace - □除一条□，如果□除的□将□个面分开，□□除原始面并用新面替□它□。
- ST_RemoveIsoEdge - □除孤立的□并返回操作的描述。如果□□未被隔离，□会引□异常。
- ST_RemoveIsoNode - □除孤立□点并返回操作描述。如果□点不是孤立的（是□的开始或□束），□会引□异常。
- ST_SRID - 返回几何□形的空□参考□□符。
- ST_StartPoint - 返回 LineString 的第一个点.
- ST_SymDifference - □算表示几何□形 A 和 B 不相交部分的几何□形。
- ST_Touches - □□□个几何□形是否至少有一个共同点，但它□的内部不相交
- ST_Transform - 返回坐□□□□不同空□参考系的新几何□形。
- ST_Union - □算表示□入几何□形的点集并集的几何□形。
- ST_Volume - □算 3D □体的体□。如果□用于表面（甚至□合）几何□形将返回 0。
- ST_WKBToSQL - 从已知的二□制表示 (WKB) 返回指定的 ST_Geometry □。□是 ST_GeomFromWKB 的□名，不□ srid
- ST_WKTToSQL - 从已知的文本表示 (WKT) 返回指定的 ST_Geometry □。□是 ST_GeomFromText 的□名
- ST_Within - □□ A 的每个点是否都位于 B 中，并且它□的内部是否有一个共同点
- ST_X - 返回点的 X 坐□。
- ST_Y - 返回点的 Y 坐□。
- ST_Z - 返回点的 Z 坐□。
- ST_SRID - 返回拓扑几何的空□参考□□符。

13.4 PostGIS 地理支持函数

以下列出的函数和操作符是 PostGIS 函数/操作符，它□的□入或□出涉及到 geography 数据□型□象。

Note
□有 (T) 的函数不是本机大地□量函数，并且使用与几何之□的 ST_Transform □用来□行操作。因此，当越□日期□更□、极点以及覆盖多个 UTM 区域的大型几何□形或几何□形□□，它□可能不会按□期□行。基本□□ - （有利于 UTM、□伯特方位角（北/南），并在最坏的情况下依靠墨卡托）

- ST_Area - 返回多□形几何体的面□。
- ST_AsBinary - 返回不□ SRID 元数据的几何/地理的 OGC/ISO 已知的二□制 (WKB) 表示形式。
- ST_AsEWKT - 使用 SRID 元数据返回几何□形的已知文本 (WKT) 表示形式。
- ST_AsGML - 将几何□形作□ GML 版本 2 或 3 元素返回。
- ST_AsGeoJSON - 以 GeoJSON 格式返回一个几何体或要素。
- ST_AsKML - 将几何□形作□ KML 元素返回。
- ST_AsSVG - 返回几何体的 SVG 路径数据。
- ST_AsText - 返回不□ SRID 元数据的几何/地理的已知文本 (WKT) 表示形式。
- ST_Azimuth - 返回□点之□直□的基于北方的方位角。
- ST_Buffer - □算覆盖距几何体□定距离内所有点的几何体。
- ST_Centroid - 返回几何体的几何中心。
- ST_ClosestPoint - 返回 g1 上最接近 g2 的 2D 点。□是从一个几何体到□一个几何体的最短直□的第一个点。
- ST_CoveredBy - □□ A 的每个点是否都位于 B 中
- ST_Covers - □□ B 的每个点是否都位于 A 中
- ST_DWithin - □□□个几何□形是否在□定距离内
- ST_Distance - 返回□个几何或地理□之□的距离。
- ST_GeogFromText - 从已知的文本表示或□展 (WKT) 返回指定的地理□。
- ST_GeogFromWKB - 从已知的二□制几何表示 (WKB) 或□展的已知的二□制 (EWKB) □建地理□例。
- ST_GeographyFromText - 从已知的文本表示或□展 (WKT) 返回指定的地理□。
- = - 如果几何/地理 A 的坐□和坐□□序与几何/地理 B 的坐□和坐□□序相同，□返回 TRUE。
- ST_Intersection - □算表示几何 A 和 B 的共享部分的几何。
- ST_Intersects - □□□个几何□形是否相交（它□至少有一个共同点）
- ST_Length - 返回□性几何体的二□□度。
- ST_LineInterpolatePoint - 返回沿□在百分比指示位置的插□点。
- ST_LineInterpolatePoints - 返回沿直□以分数□隔插□的点。
- ST_LineLocatePoint - 返回□上最接近点的分数位置。
- ST_LineSubstring - 返回□个小数位置之□的直□部分。
- ST_Perimeter - 返回多□形几何或地理的□界□度。

- ST_Project - 返回从起点按距离和方位角（方位角）投影的点。
- ST_Segmentize - 返回修改后的几何⊠形/地理，其⊠段不⊠于⊠定距离。
- ST_ShortestLine - 返回⊠个几何⊠形之⊠的 2D 最短⊠
- ST_Summary - 返回几何内容的文本摘要。
- <-> - 返回 A 和 B 之⊠的 2D 距离。
- && - 如果 A 的 2D ⊠界框与 B 的 2D ⊠界框相交，⊠返回 TRUE。

13.5 PostGIS ⊠格支持函数

下面⊠出的函数和⊠算符是 PostGIS 函数/⊠算符，它⊠将raster数据⊠型⊠象作⊠⊠入或返回作⊠⊠出。按字母⊠序列出。

- Box3D - 返回⊠格封⊠框的 box 3d 表示形式。
- @ - 如果 A 的⊠界框包含在 B 的⊠界框中，⊠返回 TRUE。使用双精度⊠界框。
- ~ - 如果 A 的⊠界框包含 B 的⊠界框，⊠返回 TRUE。使用双精度⊠界框。
- = - 如果 A 的⊠界框与 B 的⊠界框相同，⊠返回 TRUE。使用双精度⊠界框。
- && - 如果 A 的⊠界框与 B 的⊠界框相交，⊠返回 TRUE。
- &< - 如果 A 的⊠界框位于 B 的左⊠，⊠返回 TRUE。
- &> - 如果 A 的⊠界框位于 B 的右⊠，⊠返回 TRUE。
- ~= - 如果 A 的⊠界框与 B 的⊠界框相同，⊠返回 TRUE。
- ST_Retile - 从任意平⊠的⊠格覆盖范⊠返回一⊠配置的平⊠。
- ST_AddBand - 返回一个⊠格，其中在⊠定索引位置添加了⊠定⊠型的新波段和⊠定初始⊠。如果未指定索引，⊠将⊠添加到末尾。
- ST_AsBinary/ST_AsWKB - 返回⊠格的熟知的二⊠制 (WKB) 表示形式。
- ST_AsGDALRaster - 以指定的 GDAL ⊠格格式返回⊠格⊠⊠。⊠格格式是⊠⊠⊠支持的格式之一。使用 ST_GDALDrivers() ⊠取您的⊠支持的格式列表。
- ST_AsHexWKB - 返回⊠格的十六⊠制表示形式的熟知的二⊠制 (WKB)。
- ST_AsJPEG - 将⊠格⊠⊠⊠定的波段作⊠⊠个⊠合⊠影⊠出⊠ (JPEG) ⊠像（字⊠数⊠）返回。如果未指定波段且有 1 个或 3 个以上波段，⊠⊠使用第一个波段。如果只有 3 个波段，⊠使用所有 3 个波段并将其映射到 RGB。
- ST_AsPNG - 将⊠格⊠⊠⊠定的波段作⊠⊠个便携式网⊠⊠形 (PNG) ⊠像（字⊠数⊠）返回。如果⊠格中有 1、3 或 4 个波段且未指定波段，⊠使用所有波段。如果有 2 个以上或多于 4 个波段且未指定波段，⊠⊠使用波段 1。波段映射到 RGB 或 RGBA 空⊠。
- ST_AsRaster - 将 PostGIS 几何⊠形⊠⊠⊠ PostGIS ⊠格。
- ST_AsTIFF - 将⊠格⊠定的波段作⊠⊠个 TIFF ⊠像（字⊠数⊠）返回。如果未指定波段或⊠格中不存在任何指定波段，⊠将⊠⊠使用所有波段。
- ST_Aspect - 返回高程⊠格波段的坡向（默⊠以度⊠⊠位）。⊠于分析地形很有用。
- ST_Band - 返回⊠有⊠格的一个或多个波段作⊠新⊠格。⊠于从⊠有⊠格⊠建新⊠格非常有用。
- ST_BandFileSize - 返回文件系⊠中存⊠的波段的文件大小。如果未指定波段号，⊠假定⊠ 1。

- ST_BandFileTimestamp - 返回文件系⊠中存⊠的波段的文件⊠⊠戳。如果未指定波段号，⊠假定⊠ 1。
- ST_BandIsNoData - 如果波段⊠填充无数据⊠，⊠返回 true。
- ST_BandMetaData - 返回特定⊠格波段的基本元数据。如果未指定，⊠假定波段⊠号⊠ 1。
- ST_BandNoDataValue - 返回表示无数据的⊠定波段中的⊠。如果没有指定波段⊠号，默⊠⊠波段 1。
- ST_BandPath - 返回存⊠在文件系⊠中的波段的系⊠文件路径。如果未指定波段号，⊠假定⊠ 1。
- ST_BandPixelType - 返回⊠定波段的像素⊠型。如果未指定波段号，⊠假定⊠ 1。
- 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 - 根据源⊠格和指定波段⊠建最多四个 8BUI 波段（灰度、RGB、RGBA）的新⊠格。如果未指定，⊠假定⊠波段 1。
- ST_Contains - 如果⊠格 rastB 中没有点位于⊠格 rastA 的外部且 rastB 的内部至少有一个点位于 rastA 的内部，⊠返回 true。
- ST_ContainsProperly - 如果 rastB 与 rastA 的内部相交，但不与 rastA 的⊠界或外部相交，⊠返回 true。
- ST_Contour - 使用 GDAL ⊠廓算法从提供的⊠格波段生成一⊠矢量⊠廓。
- ST_ConvexHull - 返回⊠格的凸包几何形状，包括等于 BandNoDataValue 的像素⊠。⊠于⊠⊠形状和非⊠斜⊠格，⊠⊠出与 ST_Envelope 相同的⊠果，因此⊠⊠不⊠⊠形状或⊠斜⊠格有用。
- ST_Count - 返回⊠格或⊠格覆盖范⊠的⊠定波段中的像素数。如果未指定 band，⊠默⊠⊠ band 1。如果 except_nodata_value ⊠置⊠ true，⊠⊠⊠算不等于 nodata ⊠的像素。
- ST_CountAgg - ⊠⊠的。返回一⊠⊠格的⊠定波段中的像素数。如果未指定 band，⊠默⊠⊠ band 1。如果 except_nodata_value ⊠置⊠ true，⊠⊠⊠算不等于 NODATA ⊠的像素。
- ST_CoveredBy - 如果⊠格 rastA 中没有点位于⊠格 rastB 之外，⊠返回 true。
- ST_Covers - 如果⊠格 rastB 中没有点位于⊠格 rastA 之外，⊠返回 true。
- ST_DFullyWithin - 如果⊠格 rastA 和 rastB 彼此完全在指定距离内，⊠返回 true。
- ST_DWithin - 如果⊠格 rastA 和 rastB 彼此之⊠的距离在指定距离内，⊠返回 true。
- ST_Disjoint - 如果⊠格 rastA 在空⊠上不与 rastB 相交，⊠返回 true。
- ST_DumpAsPolygons - 从⊠定的⊠格⊠中返回一⊠ geomval (geom,val) 行。如果未指定波段⊠号，⊠波段⊠号默⊠⊠ 1。
- ST_DumpValues - ⊠取指定 band（波段）的⊠作⊠二⊠数⊠。
- ST_Envelope - 返回⊠格范⊠的多⊠形表示形式。
- ST_FromGDALRaster - 从受支持的 GDAL ⊠格文件返回⊠格。
- ST_GeoReference - 返回 GDAL 或 ESRI 格式的地理配准元数据，如世界文件中常⊠的格式。默⊠⊠ GDAL。
- ST_Grayscale - 根据源⊠格和代表⊠色、⊠色和⊠色的指定波段⊠建新的 1-8BUI 波段⊠格
- ST_HasNoBand - 如果不存在具有⊠定波段⊠号的波段，⊠返回 true。如果未指定波段号，⊠假定波段号⊠ 1。
- ST_Height - 返回⊠格的高度（以像素⊠⊠位）。
- ST_HillShade - 使用提供的方位角、高度、亮度和比例⊠入返回高程⊠格⊠的假⊠照明。
- ST_Histogram - 返回一⊠⊠⊠，⊠⊠了⊠格或⊠格覆盖数据分布，其中包括分离的分箱范⊠。如果未指定，将自⊠⊠算分箱数。

- ST_InterpolateRaster - 基于⊠入的 3 ⊠点集插⊠网格表面，使用 X 和 Y ⊠在网格上定位点，并使用点的 Z ⊠作⊠表面高程。
- ST_Intersection - 返回一个⊠格或一⊠几何像素⊠⊠，表示⊠个⊠格的共享部分或⊠格矢量化和几何⊠形的几何交集。
- ST_Intersects - 如果⊠格 rastA 与⊠格 rastB 空⊠相交，⊠返回 true。
- ST_IsEmpty - 如果⊠格⊠空（⊠度 = 0 且高度 = 0），⊠返回 true。否⊠，返回 false。
- ST_MakeEmptyCoverage - 用空⊠格⊠⊠网格覆盖地理参考区域。
- ST_MakeEmptyRaster - 返回一个空白⊠格（不包含波段），其具有⊠定的尺寸（⊠度和高度）、左上角 X 和 Y 坐⊠、像素大小和旋⊠参数（scalex、scaley、skewx 和 skewy），以及参考系⊠（SRID）。如果⊠⊠了一个⊠格，⊠返回一个具有相同大小、⊠⊠方式和 SRID 的新⊠格。如果省略了 SRID，⊠空⊠参考将⊠置⊠未知（0）。
- ST_MapAlgebra (callback function version) - 回⊠函数版本 - ⊠定一个或多个⊠入⊠格、波段索引和一个用⊠指定的回⊠函数，返回⊠波段⊠格。
- ST_MapAlgebraExpr - 1 ⊠格波段版本：通⊠⊠⊠入⊠格波段和提供的像素⊠型⊠用有效的 PostgreSQL 代数⊠算来⊠建新的⊠波段⊠格。如果未指定波段，⊠假定⊠波段 1。
- ST_MapAlgebraExpr - 2 ⊠格波段版本：通⊠⊠提供的⊠个⊠入⊠格波段和像素⊠型⊠用有效的 PostgreSQL 代数⊠算来⊠建新的⊠波段⊠格。如果未指定波段⊠号，⊠假定每个⊠格的波段 1。生成的⊠格将在第一个⊠格定⊠的网格上⊠⊠（比例、⊠斜和像素角），并具有由“extenttype”参数定⊠的范⊠。“extenttype”的⊠可以是：INTERSECTION、UNION、FIRST、SECOND。
- ST_MapAlgebraFct - 1 波段版本 - 通⊠在⊠入⊠格波段和提供的像素⊠型上⊠用有效的 PostgreSQL 函数来⊠建新的⊠波段⊠格。如果未指定波段，⊠假定⊠波段 1。
- ST_MapAlgebraFct - 2 波段版本 - 通⊠在提供的 2 个⊠入⊠格波段和像素⊠型上⊠用有效的 PostgreSQL 函数来⊠建新的⊠波段⊠格。如果未指定波段，⊠假定⊠波段 1。如果未指定，范⊠⊠型默⊠⊠ INTERSECTION。
- ST_MapAlgebraFctNgb - 1-波段版本：使用用⊠定⊠的 PostgreSQL 函数映射代数最近⊠。返回一个⊠格，其⊠是涉及⊠入⊠格波段⊠的⊠域的 PLPGSQL 用⊠函数的⊠果。
- ST_MapAlgebra (expression version) - 表⊠式版本 - 在⊠定一⊠个⊠入⊠格、波段索引和一个或多个用⊠指定的 SQL 表⊠式的情况下，返回⊠波段⊠格。
- ST_MemSize - 返回⊠格占用的空⊠量（以字⊠⊠⊠位）。
- ST_MetaData - 返回有关⊠格⊠象的基本元数据，例如像素大小、旋⊠（⊠斜）、左上、左下等。
- ST_MinConvexHull - 返回⊠格的凸包几何形状（不包括 NODATA 像素）。
- ST_NearestValue - 返回由 columnx 和 rowy 指定的⊠定⊠像素的最接近的非 NODATA ⊠或以与⊠格相同的空⊠参考坐⊠系表示的几何点。
- ST_Neighborhood - 返回⊠定波段像素周⊠非 NODATA ⊠的二⊠双精度数⊠，⊠⊠定波段像素由 columnX 和 rowY 或以与⊠格相同的空⊠参考坐⊠系表示的几何点指定。
- ST_NotSameAlignmentReason - 返回⊠明⊠格是否⊠⊠的文本，如果未⊠⊠，⊠⊠明原因。
- ST_NumBands - 返回⊠格⊠象中的波段数。
- ST_Overlaps - 如果⊠格 rastA 和 rastB 相交，但其中一个不完全包含⊠一个，⊠返回 true。
- ST_PixelAsCentroid - 返回像素表示的区域的⊠心（点几何）。
- ST_PixelAsCentroids - 返回⊠格波段的每个像素的⊠心（点几何）以及每个像素的⊠、X 和 Y ⊠格坐⊠。点几何是像素表示的区域的⊠心。
- ST_PixelAsPoint - 返回像素左上角的点几何形状。
- ST_PixelAsPoints - 返回⊠格波段的每个像素的点几何⊠形以及每个像素的⊠、X 和 Y ⊠格坐⊠。点几何的坐⊠是像素的左上角。

- ST_PixelAsPolygon - 返回限定特定行和列的像素的多☒形几何形状。
- ST_PixelAsPolygons - 返回包☒☒格☒的每个像素的多☒形几何☒形以及每个像素的☒、X 和 Y ☒格坐☒。
- ST_PixelHeight - 返回空☒参考系☒的几何☒位的像素高度。
- ST_PixelOfValue - ☒取☒等于搜索☒的像素的列 x、行 y 坐☒。
- ST_PixelWidth - 返回空☒参考系☒的几何☒位的像素☒度。
- ST_Polygon - 返回由具有非无数据☒的像素☒的像素并集形成的多多☒形几何体。如果未指定波段☒号，☒波段☒号默☒☒ 1。
- ST_Quantile - ☒算☒本或☒体上下文中☒格或☒格表覆盖范☒的分位数。因此，可以☒☒某个☒是否位于☒格的 25%、50%、75% 百分位☒。
- ST_RastFromHexWKB - 从熟知的二☒制 (WKB) ☒格的十六☒制表示形式返回☒格☒。
- ST_RastFromWKB - 从熟知的二☒制 (WKB) ☒格返回☒格☒。
- ST_RasterToWorldCoord - 在☒定列和行的情况下，以几何 X 和 Y（☒度和☒度）形式返回☒格的左上角。列和行从 1 开始。
- ST_RasterToWorldCoordX - 返回☒格、列和行左上角的几何 X 坐☒。列和行的☒号从 1 开始。
- ST_RasterToWorldCoordY - 返回☒格、列和行的左上角的几何 Y 坐☒。列和行的☒号从 1 开始。
- ST_Reclass - ☒建由从原始数据重新分☒的波段☒型☒成的新☒格。nband 是要更改的波段。如果未指定 nband，☒假定☒ 1。所有其他波段均按原☒返回。使用案例：将 16BUI 波段☒☒☒ 8BUI 等，以便更☒☒地呈☒☒可☒格式。
- ST_Resample - 重采☒一个☒格☒像，可以指定重新采☒算法、新的尺寸、任意的☒格角点，以及一☒☒格地理参考属性，☒些属性可以自己定☒，也可以从☒一个☒格☒像中借用。
- ST_Rescale - 通☒☒☒整☒格的比例（或像素大小）来重新采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear、Cubic、CubicSpline、Lanczos、Max 或 Min 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_Resize - 将☒格大小☒整☒新的☒度/高度
- ST_Reskew - 通☒☒☒整☒格的☒斜（或旋☒参数）来重采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear、Cubic、CubicSpline 或 Lanczos 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_Rotation - 返回☒格的旋☒弧度。
- ST_Roughness - 返回一个☒算出的数字高程模型（DEM）的’ 粗糙度’ 的☒格。
- ST_SRID - 返回在 spatial_ref_sys 表中定☒的☒格的空☒参考☒☒符。
- ST_SameAlignment - 如果☒格具有相同的☒斜、比例、空☒参考和偏移（像素可以放在同一网格上而不切割成像素），☒返回 true；如果没有注意☒☒☒☒，☒返回 false。
- ST_ScaleX - 返回像素☒度的 X 分量（以坐☒参考系☒☒位）。
- ST_ScaleY - 返回像素高度的 Y 分量（以坐☒参考系☒☒位）。
- ST_SetBandIndex - 更新 out-db band 的 external band ☒号
- ST_SetBandIsNoData - 将☒区的 isnodata ☒志☒置☒ TRUE。
- ST_SetBandNoDataValue - ☒置代表无数据的☒定波段的☒。如果未指定波段，☒假定☒波段 1。要将波段☒☒☒没有 nodata ☒，☒☒置 nodata ☒ = NULL。
- ST_SetBandPath - 更新 out-db band 的外部路径和 band ☒号
- ST_SetGeoReference - 在一次☒用中☒置 Georeference 6 地理配准参数。数字☒☒用空格分隔。接受 GDAL 或 ESRI 格式的☒入。默☒☒ GDAL。

- ST_SetM - 返回与☒入几何☒形具有相同 X/Y 坐☒的几何☒形，并使用☒求的重采☒算法将☒格中的☒复制到 M ☒度。
- ST_SetRotation - 以弧度☒置☒格的旋☒。
- ST_SetSRID - 将☒格的 SRID ☒置☒在 Spatial_ref_sys 表中定☒的特定整数 srid。
- ST_SetScale - 以坐☒参考系☒☒位☒置像素的 X 和 Y 大小。数字☒位/像素☒度/高度。
- ST_SetSkew - ☒置地理参考 X 和 Y ☒斜（或旋☒参数）。如果只☒入一个，☒将 X 和 Y ☒置☒相同的☒。
- ST_SetUpperLeft - 将☒格左上角的☒☒置☒投影的 X 和 Y 坐☒。
- ST_SetValue - 返回修改后的☒格，其☒果是将☒定波段中的☒☒置☒指定列 x、行 y 像素或与特定几何☒形相交的像素。波段☒号从 1 开始，如果未指定波段，☒默☒☒ 1。
- ST_SetValues - 返回通☒☒置☒定波段的☒而☒生的修改后的☒格。
- ST_SetZ - 返回与☒入几何☒形具有相同 X/Y 坐☒的几何☒形，并使用☒求的重采☒算法将☒格中的☒复制到 Z ☒度。
- ST_SkewX - 返回地理参考 X ☒斜（或旋☒参数）。
- ST_SkewY - 返回地理参考 Y ☒斜（或旋☒参数）。
- ST_Slope - 返回高程☒格☒的坡度（默☒以度☒☒位）。☒于分析地形很有用。
- ST_SnapToGrid - 通☒将☒格捕捉到网格来重采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear, Cubic, CubicSpline 或 Lanczos 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_Summary - 返回☒格内容的文本摘要。
- ST_SummaryStats - 返回一☒☒☒☒☒信息，包括☒定☒格或☒格覆盖的☒格☒的☒数、☒和、均☒、☒准差、最小☒和最大☒。如果未指定☒号，☒假定☒☒ 1。
- ST_SummaryStatsAgg - ☒☒的。返回一☒☒格的☒定☒格波段的摘要☒☒信息，其中包含☒数、☒和、平均☒、☒准差、最小☒、最大☒。如果未指定☒号，☒假定☒☒ 1。
- ST_TPI - 返回一个☒算出的地形位置指数（Topographic Position Index）的☒格。
- ST_TRI - 返回具有☒算的地形☒固性指数的☒格。
- ST_Tile - 返回根据☒出☒格的所需☒度分割☒入☒格而☒生的一☒☒格。
- ST_Touches - 如果☒格 rastA 和 rastB 至少有一个共同点但它☒的内部不相交，☒返回 true。
- ST_Transform - 使用指定的重采☒算法将已知空☒参考系☒中的☒格重新投影到☒一个已知空☒参考系☒。☒☒有 NearestNeighbor、Bilinear、Cubic、CubicSpline、Lanczos（默☒☒ NearestNeighbor）。
- ST_Union - 将一☒☒格切片的并集返回☒由 1 个或多个波段☒成的☒个☒格。
- ST_UpperLeftX - 返回投影空☒参考中☒格的左上角 X 坐☒。
- ST_UpperLeftY - 返回投影空☒参考中☒格的左上角 Y 坐☒。
- ST_Value - 返回指定列 x、行 y 像素或特定几何点☒☒定波段的☒。波段☒号从 1 开始，如果未指定，☒默☒☒ 1。如果将参数 exclude_nodata_value ☒置☒ false，☒所有像素都被☒☒与 nodata 像素相交并返回其☒。如果未☒☒参数 exclude_nodata_value，☒从☒格的元数据中☒取☒☒。
- ST_ValueCount - 返回一☒☒☒，其中包含像素☒☒以及具有☒定☒集的☒格（或☒格覆盖范☒）☒定☒中的像素数☒数。如果未指定波段，☒默☒☒波段 1。默☒情况下，不☒算☒点数据☒像素。☒出像素中的所有其他☒，并将像素☒☒四舍五入到最接近的整数。
- ST_Width - 返回☒格的☒度（以像素☒☒位）。
- ST_Within - 如果☒格 rastA 中没有点位于☒格 rastB 的外部且 rastA 的内部至少有一个点位于 rastB 的内部，☒返回 true。

- ST_WorldToRasterCoord - 给定几何 X 和 Y（经度和纬度）或以栅格的空间参考坐标系表示的点几何，将左上角作为列和行返回。
- ST_WorldToRasterCoordX - 返回栅格中点几何 (pt) 的列或以栅格世界空间参考系表示的 X 和 Y 世界坐标 (xw, yw)。
- ST_WorldToRasterCoordY - 返回点几何形 (pt) 的栅格中的行或以栅格的世界空间参考系表示的 X 和 Y 世界坐标 (xw, yw)。
- UpdateRasterSRID - 更改用给定的列和表中所有栅格的 SRID。

13.6 PostGIS 几何/地理/栅格输出函数

下面输出的函数是 PostGIS 函数，它将一组或多个 Geometry_dump 或 geomval 数据类型对象作为输入或返回作为输出。

- ST_DumpAsPolygons - 从给定的栅格中返回一组 geomval (geom,val) 行。如果未指定波段号，波段号默认为 1。
- ST_Intersection - 返回一个栅格或一组几何像素输出，表示两个栅格的共享部分或栅格矢量化和几何形的几何交集。
- ST_Dump - 返回几何组件的一组 geometry_dump 行。
- ST_DumpPoints - 返回几何形中坐标的一组 geometry_dump 行。
- ST_DumpRings - 返回多形外环和内环的一组 geometry_dump 行。
- ST_DumpSegments - 将几何形中的各个线段返回一组 geometry_dump 行。

13.7 PostGIS 边界框函数

下面输出的函数是 PostGIS 函数，它将 Box* 系列 PostGIS 空间类型作为输入或返回作为输出。box 系列类型由box2d和 box3d构成

- Box2D - 返回表示几何形的 2D 范围的 BOX2D。
- Box3D - 返回表示几何体 3D 范围的 BOX3D。
- Box3D - 返回栅格封框的 box 3d 表示形式。
- ST_3DExtent - 返回几何形的 3D 边界框的聚合函数。
- ST_3DMakeBox - 创建由两个 3D 点几何形定义的 BOX3D。
- ST_AsMVTGeom - 将几何形转换为 MVT 瓦片的坐标空间。
- ST_AsTWKB - 返回几何形式为 TWKB，又名“微小的已知的二进制”
- ST_Box2dFromGeoHash - 从 GeoHash 字符串返回 BOX2D。
- ST_ClipByBox2D - 计算几何形落在矩形内的部分。
- ST_EstimatedExtent - 返回空间表的估计范围。
- ST_Expand - 返回从一个边界框或几何形展的边界框。
- ST_Extent - 返回几何形边界框的聚合函数。

- ST_MakeBox2D - 建由个 2D 点几何形定的 BOX2D。
- ST_RemoveIrrelevantPointsForView - Removes points that are irrelevant for rendering a specific rectangluar view of a geometry.
- ST_XMax - 返回 2D 或 3D 界框或几何体的 X 最大。
- ST_XMin - 返回 2D 或 3D 界框或几何体的 X 最小。
- ST_YMax - 返回 2D 或 3D 界框或几何体的 Y 最大。
- ST_YMin - 返回 2D 或 3D 界框或几何体的 Y 最小。
- ST_ZMax - 返回 2D 或 3D 界框或几何体的 Z 最大。
- ST_ZMin - 返回 2D 或 3D 界框或几何体的 Z 最小。
- RemoveUnusedPrimitives - 除定有 TopoGeometry 象不需要的拓扑基元。
- ValidateTopology - 返回一 validatetopology_returntype 象，明拓扑。
- ~(box2df,box2df) - 如果 2D 浮点精度界框 (BOX2DF) 包含一个 2D 浮点精度界框 (BOX2DF)，返回 TRUE。
- ~(box2df,geometry) - 如果 2D 浮点精度界框 (BOX2DF) 包含几何体的 2D 界框，返回 TRUE。
- ~(geometry,box2df) - 如果几何体的 2D 粘合框包含 2D 浮点精度界框 (GIDX)，返回 TRUE。
- @(box2df,box2df) - 如果 2D 浮点精度界框 (BOX2DF) 包含在一个 2D 浮点精度界框内，返回 TRUE。
- @(box2df,geometry) - 如果 2D 浮点精度界框 (BOX2DF) 包含在几何体的 2D 界框中，返回 TRUE。
- @(geometry,box2df) - 如果几何体的 2D 界框包含在 2D 浮点精度界框 (BOX2DF) 中，返回 TRUE。
- &&(box2df,box2df) - 如果个 2D 浮点精度界框 (BOX2DF) 彼此相交，返回 TRUE。
- &&(box2df,geometry) - 如果 2D 浮点精度界框 (BOX2DF) 与几何体的（存的）2D 界框相交，返回 TRUE。
- &&(geometry,box2df) - 如果几何体的（存的）2D 界框与 2D 浮点精度界框 (BOX2DF) 相交，返回 TRUE。

13.8 支持 3D 的 PostGIS 函数

下面出的函数是不会弃 Z-Index 的 PostGIS 函数。

- AddGeometryColumn - 将 geometry（几何）列添加到有表。
- Box3D - 返回表示几何体 3D 范的 BOX3D。
- CG_3DArea - 算 3D 表面几何形状的面。于固体将返回 0。
- CG_3DConvexHull - 算几何体的 3D 凸包。
- CG_3DDifference - 行 3D 差异
- CG_3DIntersection - 行 3D 相交
- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- CG_ApproximateMedialAxis - 算几何区域的近似中。
- CG_ConstrainedDelaunayTriangles - 返回定入几何体的束 Delaunay 三角剖分。

- CG_Extrude - 将曲面⍰出到相关体⍰
- CG_ForceLHR - ⍰制 LHR 方向
- CG_IsPlanar - ⍰⍰表面是否平坦
- CG_IsSolid - ⍰⍰几何体是否⍰⍰体。不⍰行有效性⍰⍰。
- CG_MakeSolid - 将几何体⍰造成⍰体。不⍰行任何⍰⍰。要⍰得有效的⍰体，⍰入几何⍰形必⍰是⍰合多面体曲面或⍰合 TIN。
- CG_Orientation - 确定表面方向
- CG_StraightSkeleton - 从几何体⍰算直骨架
- CG_Tesselate - ⍰多⍰形或多面体表面⍰行曲面⍰分，并以 TIN 或 TINS 集合的形式返回
- CG_Visibility - ⍰算一个从点或多⍰形几何中的⍰段生成的可⍰性多⍰形
- CG_Volume - ⍰算 3D ⍰体的体⍰。如果⍰用于表面（甚至⍰合）几何⍰形将返回 0。
- DropGeometryColumn - 从空⍰表中移除 geometry（几何）列。
- GeometryType - 以文本形式返回几何的⍰型。
- ST_3DArea - ⍰算 3D 表面几何形状的面⍰。⍰于固体将返回 0。
- ST_3DClosestPoint - 返回 g1 上最接近 g2 的 3D 点。⍰是 3D 最短⍰的第一个点。
- ST_3DConvexHull - ⍰算几何体的 3D 凸包。
- ST_3DDFullyWithin - ⍰⍰⍰个 3D 几何⍰形是否完全在⍰定的 3D 距离内
- ST_3DDWithin - ⍰⍰⍰个 3D 几何⍰形是否在⍰定的 3D 距离内
- ST_3DDifference - ⍰行 3D 差异
- ST_3DDistance - 返回⍰个几何⍰形之⍰的 3D 笛卡⍰最小距离（基于空⍰参考）（以投影⍰位表示）。
- ST_3DExtent - 返回几何⍰形的 3D ⍰界框的聚合函数。
- ST_3DIntersection - ⍰行 3D 相交
- ST_3DIntersects - ⍰⍰⍰个几何⍰形在 3D 空⍰中是否相交 - ⍰适用于点、⍰串、多⍰形、多面体曲面（区域）
- ST_3DLength - 返回⍰性几何体的 3D ⍰度。
- ST_3DLineInterpolatePoint - 返回沿 3D ⍰的小数指示位置插⍰的点。
- ST_3DLongestLine - 返回⍰个几何体之⍰的 3D 最⍰直⍰
- ST_3DMaxDistance - 返回⍰个几何⍰形之⍰的 3D 笛卡⍰最大距离（基于空⍰参考）（以投影⍰位表示）。
- ST_3DPerimeter - 返回多⍰形几何体的 3D 周⍰。
- ST_3DShortestLine - 返回⍰个几何⍰形之⍰的 3D 最短⍰
- ST_3DUnion - ⍰行 3D ⍰合。
- ST_AddMeasure - 沿⍰性几何形状插⍰⍰量⍰。
- ST_AddPoint - 将点添加到⍰串（LineString）。
- ST_Affine - ⍰几何体⍰用 3D 仿射⍰⍰。
- ST_ApproximateMedialAxis - ⍰算几何区域的近似中⍰。
- ST_AsBinary - 返回不⍰ SRID 元数据的几何/地理的 OGC/ISO 已知的二⍰制 (WKB) 表示形式。

- ST_AsEWKB - 返回□有 SRID 元数据的几何□形的□展已知的二□制 (EWKB) 表示形式。
- ST_AsEWKT - 使用 SRID 元数据返回几何□形的已知文本 (WKT) 表示形式。
- ST_AsGML - 将几何□形作□ GML 版本 2 或 3 元素返回。
- ST_AsGeoJSON - 以 GeoJSON 格式返回一个几何体或要素。
- ST_AsHEXEWKB - 使用小端 (NDR) 或大端 (XDR) □□返回 HEXEWKB 格式（作□文本）的几何□形。
- ST_AsKML - 将几何□形作□ KML 元素返回。
- ST_AsX3D - 返回 X3D xml □点元素格式的几何□形：ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Boundary - 返回几何□形的□界。
- ST_BoundingDiagonal - 返回几何□界框的□角□。
- ST_CPAWithin - □□□条□迹的最近接近点是否在指定距离内。
- ST_ChaikinSmoothing - 使用 Chaikin 算法返回几何□形的平滑版本
- ST_ClosestPointOfApproach - 返回□条□迹最接近点□的□量□。
- ST_Collect - 从一□几何□形□建 GeometryCollection 或 Multi* 几何□形。
- ST_ConstrainedDelaunayTriangles - 返回□□□定□入几何体的□束 Delaunay 三角剖分。
- ST_ConvexHull - □算几何体的凸包。
- ST_CoordDim - 返回几何体的坐□□度。
- ST_CurveN - Returns the Nth component curve geometry of a CompoundCurve.
- ST_CurveToLine - 将包含曲□的几何□形□□□□性几何□形。
- ST_DelaunayTriangles - 返回几何体□点的 Delaunay 三角剖分。
- ST_Difference - □算表示几何 A 中不与几何 B 相交的部分的几何。
- ST_DistanceCPA - 返回□条□迹的最近接近点之□的距离。
- ST_Dump - 返回几何□件的一□ geometry_dump 行。
- ST_DumpPoints - 返回几何□形中坐□的一□ geometry_dump 行。
- ST_DumpRings - 返回多□形外□和内□的一□ geometry_dump 行。
- ST_DumpSegments - □几何□形中的各个□段返回一□ geometry_dump 行。
- ST_EndPoint - 返回 LineString 或 CircularLineString 的最后一个点。
- ST_ExteriorRing - 返回表示多□形外□的 LineString。
- ST_Extrude - 将曲面□出到相关体□
- ST_FlipCoordinates - 返回 X □和 Y □□□的几何□形版本。
- ST_Force2D - □制几何□形□入“二□模式”。
- ST_ForceCurve - 如果适用，将一个几何□形上升到其曲□□型。
- ST_ForceLHR - □制 LHR 方向
- ST_ForcePolygonCCW - 将所有外□逆□□定向，将所有内□□□□定向。
- ST_ForcePolygonCW - □□□方向□整所有外□，逆□□方向□整所有内□。
- ST_ForceRHR - □制多□形□点的方向遵循右手定□。

- ST_ForceSFS - 强制几何形体使用 SFS 1.1 几何体型。
- ST_Force3D - 强制几何形体入 XYZ 模式。这是 ST_Force3DZ 的别名。
- ST_Force3DZ - 强制几何形体入 XYZ 模式。
- ST_Force4D - 强制几何形体入 XYZM 模式。
- ST_ForceCollection - 将几何形体转换成几何集合（GEOMETRYCOLLECTION）。
- ST_GeomFromEWKB - 从扩展已知的二进制表示 (EWKB) 返回指定的 ST_Geometry 值。
- ST_GeomFromEWKT - 从扩展已知的文本表示 (EWKT) 返回指定的 ST_Geometry 值。
- ST_GeomFromGML - 将几何体形的 GML 表示形式作转换入并输出 PostGIS 几何体象
- ST_GeomFromGeoJSON - 将几何体形的 geojson 表示形式作转换入并输出 PostGIS 几何体象
- ST_GeomFromKML - 将几何体形的 KML 表示形式作转换入并输出 PostGIS 几何体象
- ST_GeometricMedian - 返回多点的几何中位数。
- ST_GeometryN - 返回几何集合的一个元素。
- ST_GeometryType - 以文本形式返回几何体形的 SQL-MM 类型。
- ST_HasArc - 测试几何体形是否包含圆弧
- ST_HasM - 测试几何体是否具有 M（测量）维度。
- ST_HasZ - 测试几何体是否具有 Z 维度。
- ST_InteriorRingN - 返回多边形的第 N 个内环（孔）。
- ST_InterpolatePoint - 返回最接近点的几何体形的插值测量值。
- ST_Intersection - 计算表示几何 A 和 B 的共享部分的几何。
- ST_IsClosed - 测试 LineStrings 的起点和终点是否重合。对于多面体表面测试是否封合（有心）。
- ST_IsCollection - 测试几何体型是否几何集合。
- ST_IsPlanar - 测试表面是否平坦
- ST_IsPolygonCCW - 测试多边形是否具有逆时针方向的外环和时针方向的内环。
- ST_IsPolygonCW - 测试多边形是否具有时针外环和逆时针内环。
- ST_IsSimple - 测试几何体的自完整性或自接触点。
- ST_IsSolid - 测试几何体是否表示体。不进行有效性测试。
- ST_IsValidTrajectory - 测试几何体形是否有效轨迹。
- ST_LengthSpheroid - 返回球体上度/度几何体的 2D 或 3D 长度/周长。
- ST_LineFromMultiPoint - 从多点几何体建线串。
- ST_LineInterpolatePoint - 返回沿线在百分比指示位置的插值点。
- ST_LineInterpolatePoints - 返回沿直线以分数间隔插值的点。
- ST_LineSubstring - 返回两个小数位置之间的直线部分。
- ST_LineToCurve - 将线性几何体形转换曲线几何体形。
- ST_LocateBetweenElevations - 返回位于高程 (Z) 范围内的几何体形部分。
- ST_M - 返回点的 M 值。

- ST_MakeLine - 从 Point, MultiPoint, 或 LineString geometries 构建 LineString。
- ST_MakePoint - 构建 2D、3DZ 或 4D 点。
- ST_MakePolygon - 从壳和可选的孔列表构建多边形。
- ST_MakeSolid - 将几何体构造成实体。不进行任何验证。要获得有效的实体，输入几何形必须是闭合多面体曲面或闭合 TIN。
- ST_MakeValid - 试图在不丢失顶点的情况下使无效几何体有效。
- ST_MemSize - 返回几何形占用的内存空间量。
- ST_MemUnion - 聚合函数，以内存高效但速度较慢的方式合并几何形
- ST_NDims - 返回几何体的坐标维度。
- ST_NPoints - 返回几何形中的点数（顶点）。
- ST_NRings - 返回多边形几何中的环数。
- ST_Node - 节点是线的集合。
- ST_NumCurves - Return the number of component curves in a CompoundCurve.
- ST_NumGeometries - 返回几何集合中的元素数量。
- ST_NumPatches - 返回多面体曲面上的面数。对于非多面体几何形状将返回 null。
- ST_Orientation - 确定表面方向
- ST_PatchN - 返回多边形曲面（PolyhedralSurface）的第 N 个几何体（面）。
- ST_PointFromWKB - 使用给定的 SRID 从 WKB 构建几何形
- ST_PointN - 返回几何形中第一个线串或圆形线串中的第 N 个点。
- ST_PointOnSurface - 计算保证位于多边形或几何体上的点。
- ST_Points - 返回包含几何坐标的 MultiPoint。
- ST_Polygon - 从具有指定 SRID 的线串构建多边形。
- ST_RemovePoint - 从线串中删除一个点。
- ST_RemoveRepeatedPoints - 返回删除了重复点的几何形。
- ST_Reverse - 返回顶点顺序相反的几何体。
- ST_Rotate - 绕原点旋转几何体。
- ST_RotateX - 绕 X 轴旋转几何体。
- ST_RotateY - 绕 Y 轴旋转几何体。
- ST_RotateZ - 绕 Z 轴旋转几何体。
- ST_Scale - 按给定因子缩放几何形。
- ST_Scroll - 更改闭合线串的起点。
- ST_SetPoint - 用给定点替换线串的点。
- ST_ShiftLongitude - 在 -180-180 和 0-360 之间移动几何形的经度坐标。
- ST_SnapToGrid - 将输入几何体的所有点捕捉到网格。
- ST_StartPoint - 返回 LineString 的第一个点.

- ST_StraightSkeleton - 从几何体⍰算直骨架
- ST_SwapOrdinates - 返回更改后的几何⍰形，其中交⍰了⍰定的坐⍰⍰。
- ST_SymDifference - ⍰算表示几何⍰形 A 和 B 不相交部分的几何⍰形。
- ST_Tesselate - ⍰多⍰形或多面体表面⍰行曲面⍰分，并以 TIN 或 TINS 集合的形式返回
- ST_TransScale - 按⍰定的偏移量和系数平移和⍰放几何⍰形。
- ST_Translate - 按⍰定的偏移量平移几何⍰形。
- ST_UnaryUnion - ⍰算⍰个几何体的⍰件的并集。
- ST_Union - ⍰算表示⍰入几何⍰形的点集并集的几何⍰形。
- ST_Volume - ⍰算 3D ⍰体的体⍰。如果⍰用于表面（甚至⍰合）几何⍰形将返回 0。
- ST_WrapX - 将几何体⍰⍰在 X ⍰周⍰。
- ST_X - 返回点的 X 坐⍰。
- ST_XMax - 返回 2D 或 3D ⍰界框或几何体的 X 最大⍰。
- ST_XMin - 返回 2D 或 3D ⍰界框或几何体的 X 最小⍰。
- ST_Y - 返回点的 Y 坐⍰。
- ST_YMax - 返回 2D 或 3D ⍰界框或几何体的 Y 最大⍰。
- ST_YMin - 返回 2D 或 3D ⍰界框或几何体的 Y 最小⍰。
- ST_Z - 返回点的 Z 坐⍰。
- ST_ZMax - 返回 2D 或 3D ⍰界框或几何体的 Z 最大⍰。
- ST_ZMin - 返回 2D 或 3D ⍰界框或几何体的 Z 最小⍰。
- ST_Zmflag - 返回指示几何体的 ZM 坐⍰⍰度的代⍰。
- Equals - 如果⍰个拓扑几何由相同的拓扑基元⍰成，⍰返回 true。
- Intersects - 如果⍰个拓扑几何中的任何一⍰⍰元相交，⍰返回 true。
- UpdateGeometrySRID - 更新几何列中所有要素的 SRID 以及表元数据。
- &&& - 如果 A 的 n ⍰⍰界框与 B 的 n ⍰⍰界框相交，⍰返回 TRUE。
- &&&(geometry,gidx) - 如果几何体的（⍰存的）n ⍰⍰界框与 n ⍰浮点精度⍰界框 (GIDX) 相交，⍰返回 TRUE。
- &&&(gidx,geometry) - 如果 n ⍰浮点精度⍰界框 (GIDX) 与几何体的（⍰存的）n ⍰⍰界框相交，⍰返回 TRUE。
- &&&(gidx,gidx) - 如果⍰个 n ⍰浮点精度⍰界框 (GIDX) 彼此相交，⍰返回 TRUE。

13.9 PostGIS 曲⍰几何支持函数

下面⍰出的函数是 PostGIS 函数，可以使用 CIRCULARSTRING、CURVEPOLYGON 和其他曲⍰几何⍰型

- AddGeometryColumn - 将 geometry（几何）列添加到⍰有表。
- Box2D - 返回表示几何⍰形的 2D 范⍰的 BOX2D。
- Box3D - 返回表示几何体 3D 范⍰的 BOX3D。
- DropGeometryColumn - 从空⍰表中移除 geometry（几何）列。

- GeometryType - 以文本形式返回几何的⍰型。
- PostGIS_AddBBox - 向几何体添加⍰界框。
- PostGIS_DropBBox - 从几何体中⍰除⍰界框⍰存。
- PostGIS_HasBBox - 如果⍰几何体的 bbox 已⍰存，⍰返回 TRUE，否⍰返回 FALSE。
- ST_3DExtent - 返回几何⍰形的 3D ⍰界框的聚合函数。
- ST_Affine - ⍰几何体⍰用 3D 仿射⍰⍰。
- ST_AsBinary - 返回不⍰ SRID 元数据的几何/地理的 OGC/ISO 已知的二⍰制 (WKB) 表示形式。
- ST_AsEWKB - 返回⍰有 SRID 元数据的几何⍰形的⍰展已知的二⍰制 (EWKB) 表示形式。
- ST_AsEWKT - 使用 SRID 元数据返回几何⍰形的已知文本 (WKT) 表示形式。
- ST_AsHEXEWKB - 使用小端 (NDR) 或大端 (XDR) ⍰⍰返回 HEXEWKB 格式（作⍰文本）的几何⍰形。
- ST_AsSVG - 返回几何体的 SVG 路径数据。
- ST_AsText - 返回不⍰ SRID 元数据的几何/地理的已知文本 (WKT) 表示形式。
- ST_ClusterDBSCAN - 使用 DBSCAN 算法返回每个⍰入几何⍰形的簇 id 的窗口函数。
- ST_ClusterWithin - 按⍰隔距离⍰几何⍰形⍰行聚合的聚合函数。
- ST_ClusterWithinWin - 窗口函数，返回每个⍰入几何⍰形的簇 ID，使用分离距离⍰行聚⍰。
- ST_Collect - 从一⍰几何⍰形⍰建 GeometryCollection 或 Multi* 几何⍰形。
- ST_CoordDim - 返回几何体的坐⍰⍰度。
- ST_CurveToLine - 将包含曲⍰的几何⍰形⍰⍰⍰⍰性几何⍰形。
- ST_Distance - 返回⍰个几何或地理⍰之⍰的距离。
- ST_Dump - 返回几何⍰件的一⍰ geometry_dump 行。
- ST_DumpPoints - 返回几何⍰形中坐⍰的一⍰ geometry_dump 行。
- ST_EndPoint - 返回 LineString 或 CircularLineString 的最后一个点。
- ST_EstimatedExtent - 返回空⍰表的估⍰范⍰。
- ST_FlipCoordinates - 返回 X ⍰和 Y ⍰⍰⍰的几何⍰形版本。
- ST_Force2D - ⍰制几何⍰形⍰入“二⍰模式”。
- ST_ForceCurve - 如果适用，将一个几何⍰形上升到其曲⍰⍰型。
- ST_ForceSFS - ⍰制几何⍰形⍰使用 SFS 1.1 几何⍰型。
- ST_Force3D - ⍰制几何⍰形⍰入 XYZ 模式。⍰是 ST_Force3DZ 的⍰名。
- ST_Force3DM - ⍰制几何⍰形⍰入 XYM 模式。
- ST_Force3DZ - ⍰制几何⍰形⍰入 XYZ 模式。
- ST_Force4D - ⍰制几何⍰形⍰入 XYZM 模式。
- ST_ForceCollection - 将几何⍰形⍰⍰⍰几何集合（GEOMETRYCOLLECTION）。
- ST_GeoHash - 返回几何⍰形的 GeoHash 表示形式。
- ST_GeogFromWKB - 从已知的二⍰制几何表示 (WKB) 或⍰展的已知的二⍰制 (EWKB) ⍰建地理⍰例。
- ST_GeomFromEWKB - 从⍰展已知的二⍰制表示 (EWKB) 返回指定的 ST_Geometry ⍰。

- ST_GeomFromEWKT - 从⊠展已知的文本表示 (EWKT) 返回指定的 ST_Geometry ⊠。
- ST_GeomFromText - 从已知的文本表示 (WKT) 返回指定的 ST_Geometry ⊠。
- ST_GeomFromWKB - 从已知的二⊠制几何表示 (WKB) 和可⊠的 SRID ⊠建几何⊠例。
- ST_GeometryN - 返回几何集合的一个元素。
- = - 如果几何/地理 A 的坐⊠和坐⊠⊠序与几何/地理 B 的坐⊠和坐⊠⊠序相同，⊠返回 TRUE。
- &<| - 如果 A 的⊠界框与 B 的⊠界框重⊠或低于 B 的⊠界框，⊠返回 TRUE。
- ST_HasArc - ⊠⊠几何⊠形是否包含⊠弧
- ST_Intersects - ⊠⊠⊠个几何⊠形是否相交（它⊠至少有一个共同点）
- ST_IsClosed - ⊠⊠ LineStrings 的起点和⊠点是否重合。⊠于多面体表面⊠⊠是否⊠合（⊠心）。
- ST_IsCollection - ⊠⊠几何⊠型是否⊠几何集合。
- ST_IsEmpty - ⊠⊠几何⊠形是否⊠空。
- ST_LineToCurve - 将⊠性几何⊠形⊠⊠⊠曲⊠几何⊠形。
- ST_MemSize - 返回几何⊠形占用的内存空⊠⊠量。
- ST_NPoints - 返回几何⊠形中的点数（⊠点）。
- ST_NRings - 返回多⊠形几何中的⊠数。
- ST_PointFromWKB - 使用⊠定的 SRID 从 WKB ⊠建几何⊠形
- ST_PointN - 返回几何⊠形中第一个⊠串或⊠形⊠串中的第 N 个点。
- ST_Points - 返回包含几何坐⊠的 MultiPoint。
- ST_Rotate - ⊠原点旋⊠几何体。
- ST_RotateZ - ⊠ Z ⊠旋⊠几何体。
- ST_SRID - 返回几何⊠形的空⊠参考⊠⊠符。
- ST_Scale - 按⊠定因子⊠放几何⊠形。
- ST_SetSRID - 在几何体上⊠置 SRID。
- ST_StartPoint - 返回 LineString 的第一个点.
- ST_Summary - 返回几何内容的文本摘要。
- ST_SwapOrdinates - 返回更改后的几何⊠形，其中交⊠了⊠定的坐⊠⊠。
- ST_TransScale - 按⊠定的偏移量和系数平移和⊠放几何⊠形。
- ST_Transform - 返回坐⊠⊠⊠⊠不同空⊠参考系的新几何⊠形。
- ST_Translate - 按⊠定的偏移量平移几何⊠形。
- ST_XMax - 返回 2D 或 3D ⊠界框或几何体的 X 最大⊠。
- ST_XMin - 返回 2D 或 3D ⊠界框或几何体的 X 最小⊠。
- ST_YMax - 返回 2D 或 3D ⊠界框或几何体的 Y 最大⊠。
- ST_YMin - 返回 2D 或 3D ⊠界框或几何体的 Y 最小⊠。
- ST_ZMax - 返回 2D 或 3D ⊠界框或几何体的 Z 最大⊠。
- ST_ZMin - 返回 2D 或 3D ⊠界框或几何体的 Z 最小⊠。

- ST_Zmflag - 返回指示几何体的 ZM 坐⍰⍰度的代⍰。
- UpdateGeometrySRID - 更新几何列中所有要素的 SRID 以及表元数据。
- ~(box2df,box2df) - 如果 2D 浮点精度⍰界框 (BOX2DF) 包含⍰一个 2D 浮点精度⍰界框 (BOX2DF)，⍰返回 TRUE。
- ~(box2df,geometry) - 如果 2D 浮点精度⍰界框 (BOX2DF) 包含几何体的 2D ⍰界框，⍰返回 TRUE。
- ~(geometry,box2df) - 如果几何体的 2D 粘合框包含 2D 浮点精度⍰界框 (GIDX)，⍰返回 TRUE。
- && - 如果 A 的 2D ⍰界框与 B 的 2D ⍰界框相交，⍰返回 TRUE。
- &&& - 如果 A 的 n ⍰⍰界框与 B 的 n ⍰⍰界框相交，⍰返回 TRUE。
- @(box2df,box2df) - 如果 2D 浮点精度⍰界框 (BOX2DF) 包含在⍰一个 2D 浮点精度⍰界框内，⍰返回 TRUE。
- @(box2df,geometry) - 如果 2D 浮点精度⍰界框 (BOX2DF) 包含在几何体的 2D ⍰界框中，⍰返回 TRUE。
- @(geometry,box2df) - 如果几何体的 2D ⍰界框包含在 2D 浮点精度⍰界框 (BOX2DF) 中，⍰返回 TRUE。
- &&(box2df,box2df) - 如果⍰个 2D 浮点精度⍰界框 (BOX2DF) 彼此相交，⍰返回 TRUE。
- &&(box2df,geometry) - 如果 2D 浮点精度⍰界框 (BOX2DF) 与几何体的（⍰存的）2D ⍰界框相交，⍰返回 TRUE。
- &&(geometry,box2df) - 如果几何体的（⍰存的）2D ⍰界框与 2D 浮点精度⍰界框 (BOX2DF) 相交，⍰返回 TRUE。
- &&&(geometry,gidx) - 如果几何体的（⍰存的）n ⍰⍰界框与 n ⍰浮点精度⍰界框 (GIDX) 相交，⍰返回 TRUE。
- &&&(gidx,geometry) - 如果 n ⍰浮点精度⍰界框 (GIDX) 与几何体的（⍰存的）n ⍰⍰界框相交，⍰返回 TRUE。
- &&&(gidx,gidx) - 如果⍰个 n ⍰浮点精度⍰界框 (GIDX) 彼此相交，⍰返回 TRUE。

13.10 PostGIS 多面体曲面支持函数

下面⍰出的函数是可以使用 POLYHEDRALSURFACE、POLYHEDRALSURFACEM 几何⍰形的 PostGIS 函数

- Box2D - 返回表示几何⍰形的 2D 范⍰的 BOX2D。
- Box3D - 返回表示几何体 3D 范⍰的 BOX3D。
- CG_3DArea - ⍰算 3D 表面几何形状的面⍰。⍰于固体将返回 0。
- CG_3DConvexHull - ⍰算几何体的 3D 凸包。
- CG_3DDifference - ⍰行 3D 差异
- CG_3DIntersection - ⍰行 3D 相交
- CG_3DUnion - Perform 3D union using postgis_sfcgal.
- CG_ApproximateMedialAxis - ⍰算几何区域的近似中⍰。
- CG_Extrude - 将曲面⍰出到相关体⍰
- CG_ForceLHR - ⍰制 LHR 方向
- CG_IsPlanar - ⍰⍰表面是否平坦
- CG_IsSolid - ⍰⍰几何体是否⍰⍰体。不⍰行有效性⍰⍰。

- CG_MakeSolid - 将几何体□造成□体。不□行任何□□。要□得有效的□体，□入几何□形必□是□合多面体曲面或□合 TIN。
- CG_StraightSkeleton - 从几何体□算直骨架
- CG_Tesselate - □多□形或多面体表面□行曲面□分，并以 TIN 或 TINS 集合的形式返回
- CG_Visibility - □算一个从点或多□形几何中的□段生成的可□性多□形
- CG_Volume - □算 3D □体的体□。如果□用于表面（甚至□合）几何□形将返回 0。
- GeometryType - 以文本形式返回几何的□型。
- ST_3DArea - □算 3D 表面几何形状的面□。□于固体将返回 0。
- ST_3DClosestPoint - 返回 g1 上最接近 g2 的 3D 点。□是 3D 最短□的第一个点。
- ST_3DConvexHull - □算几何体的 3D 凸包。
- ST_3DDFullyWithin - □□□个 3D 几何□形是否完全在□定的 3D 距离内
- ST_3DDWithin - □□□个 3D 几何□形是否在□定的 3D 距离内
- ST_3DDifference - □行 3D 差异
- ST_3DDistance - 返回□个几何□形之□的 3D 笛卡□最小距离（基于空□参考）（以投影□位表示）。
- ST_3DExtent - 返回几何□形的 3D □界框的聚合函数。
- ST_3DIntersection - □行 3D 相交
- ST_3DIntersects - □□□个几何□形在 3D 空□中是否相交 - □适用于点、□串、多□形、多面体曲面（区域）
- ST_3DLongestLine - 返回□个几何体之□的 3D 最□直□
- ST_3DMaxDistance - 返回□个几何□形之□的 3D 笛卡□最大距离（基于空□参考）（以投影□位表示）。
- ST_3DShortestLine - 返回□个几何□形之□的 3D 最短□
- ST_3DUnion - □行 3D □合。
- ST_Affine - □几何体□用 3D 仿射□□。
- ST_ApproximateMedialAxis - □算几何区域的近似中□。
- ST_Area - 返回多□形几何体的面□。
- ST_AsBinary - 返回不□ SRID 元数据的几何/地理的 OGC/ISO 已知的二□制 (WKB) 表示形式。
- ST_AsEWKB - 返回□有 SRID 元数据的几何□形的□展已知的二□制 (EWKB) 表示形式。
- ST_AsEWKT - 使用 SRID 元数据返回几何□形的已知文本 (WKT) 表示形式。
- ST_AsGML - 将几何□形作□ GML 版本 2 或 3 元素返回。
- ST_AsX3D - 返回 X3D xml □点元素格式的几何□形：ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_CoordDim - 返回几何体的坐□□度。
- ST_Dimension - 返回几何□形的拓扑□数。
- ST_Dump - 返回几何□件的一□ geometry_dump 行。
- ST_DumpPoints - 返回几何□形中坐□的一□ geometry_dump 行。
- ST_Expand - 返回从□一个□界框或几何□形□展的□界框。
- ST_Extent - 返回几何□形□界框的聚合函数。

- ST_Extrude - 将曲面⊠出到相关体⊠
- ST_FlipCoordinates - 返回 X ⊠和 Y ⊠⊠⊠的几何⊠形版本。
- ST_Force2D - ⊠制几何⊠形⊠入“二⊠模式”。
- ST_ForceLHR - ⊠制 LHR 方向
- ST_ForceRHR - ⊠制多⊠形⊠点的方向遵循右手定⊠。
- ST_ForceSFS - ⊠制几何⊠形⊠使用 SFS 1.1 几何⊠型。
- ST_Force3D - ⊠制几何⊠形⊠入 XYZ 模式。⊠是 ST_Force3DZ 的⊠名。
- ST_Force3DZ - ⊠制几何⊠形⊠入 XYZ 模式。
- ST_ForceCollection - 将几何⊠形⊠⊠⊠几何集合（GEOMETRYCOLLECTION）。
- ST_GeomFromEWKB - 从⊠展已知的二⊠制表示 (EWKB) 返回指定的 ST_Geometry ⊠。
- ST_GeomFromEWKT - 从⊠展已知的文本表示 (EWKT) 返回指定的 ST_Geometry ⊠。
- ST_GeomFromGML - 将几何⊠形的 GML 表示形式作⊠⊠入并⊠出 PostGIS 几何⊠象
- ST_GeometryN - 返回几何集合的一个元素。
- ST_GeometryType - 以文本形式返回几何⊠形的 SQL-MM ⊠型。
- = - 如果几何/地理 A 的坐⊠和坐⊠⊠序与几何/地理 B 的坐⊠和坐⊠⊠序相同，⊠返回 TRUE。
- &<| - 如果 A 的⊠界框与 B 的⊠界框重⊠或低于 B 的⊠界框，⊠返回 TRUE。
- ~= - 如果 A 的⊠界框与 B 的⊠界框相同，⊠返回 TRUE。
- ST_IsClosed - ⊠⊠ LineStrings 的起点和⊠点是否重合。⊠于多面体表面⊠⊠是否⊠合（⊠心）。
- ST_IsPlanar - ⊠⊠表面是否平坦
- ST_IsSolid - ⊠⊠几何体是否⊠⊠体。不⊠行有效性⊠⊠。
- ST_MakeSolid - 将几何体⊠造成⊠体。不⊠行任何⊠⊠。要⊠得有效的⊠体，⊠入几何⊠形必⊠是⊠合多面体曲面或⊠合 TIN。
- ST_MemSize - 返回几何⊠形占用的内存空⊠⊠量。
- ST_NPoints - 返回几何⊠形中的点数（⊠点）。
- ST_NumGeometries - 返回几何集合中的元素数量。
- ST_NumPatches - 返回多面体曲面上的面数。⊠于非多面体几何形状将返回 null。
- ST_PatchN - 返回多⊠形曲面（PolyhedralSurface）的第 N 个几何体（面）。
- ST_RemoveRepeatedPoints - 返回⊠除了重复点的几何⊠形。
- ST_Reverse - 返回⊠点⊠序相反的几何体。
- ST_Rotate - ⊠原点旋⊠几何体。
- ST_RotateX - ⊠ X ⊠旋⊠几何体。
- ST_RotateY - ⊠ Y ⊠旋⊠几何体。
- ST_RotateZ - ⊠ Z ⊠旋⊠几何体。
- ST_Scale - 按⊠定因子⊠放几何⊠形。
- ST_ShiftLongitude - 在 -180-180 和 0-360 之⊠移⊠几何⊠形的⊠度坐⊠。

- ST_StraightSkeleton - 从几何体⊠算直骨架
- ST_Summary - 返回几何内容的文本摘要。
- ST_SwapOrdinates - 返回更改后的几何⊠形，其中交⊠了⊠定的坐⊠⊠。
- ST_Tesselate - ⊠多⊠形或多面体表面⊠行曲面⊠分，并以 TIN 或 TINS 集合的形式返回
- ST_Transform - 返回坐⊠⊠⊠⊠不同空⊠参考系的新几何⊠形。
- ST_Volume - ⊠算 3D ⊠体的体⊠。如果⊠用于表面（甚至⊠合）几何⊠形将返回 0。
- ~(box2df,box2df) - 如果 2D 浮点精度⊠界框 (BOX2DF) 包含⊠一个 2D 浮点精度⊠界框 (BOX2DF)，⊠返回 TRUE。
- ~(box2df,geometry) - 如果 2D 浮点精度⊠界框 (BOX2DF) 包含几何体的 2D ⊠界框，⊠返回 TRUE。
- ~(geometry,box2df) - 如果几何体的 2D 粘合框包含 2D 浮点精度⊠界框 (GIDX)，⊠返回 TRUE。
- && - 如果 A 的 2D ⊠界框与 B 的 2D ⊠界框相交，⊠返回 TRUE。
- &&& - 如果 A 的 n ⊠⊠界框与 B 的 n ⊠⊠界框相交，⊠返回 TRUE。
- @(box2df,box2df) - 如果 2D 浮点精度⊠界框 (BOX2DF) 包含在⊠一个 2D 浮点精度⊠界框内，⊠返回 TRUE。
- @(box2df,geometry) - 如果 2D 浮点精度⊠界框 (BOX2DF) 包含在几何体的 2D ⊠界框中，⊠返回 TRUE。
- @(geometry,box2df) - 如果几何体的 2D ⊠界框包含在 2D 浮点精度⊠界框 (BOX2DF) 中，⊠返回 TRUE。
- &&(box2df,box2df) - 如果⊠个 2D 浮点精度⊠界框 (BOX2DF) 彼此相交，⊠返回 TRUE。
- &&(box2df,geometry) - 如果 2D 浮点精度⊠界框 (BOX2DF) 与几何体的（⊠存的）2D ⊠界框相交，⊠返回 TRUE。
- &&(geometry,box2df) - 如果几何体的（⊠存的）2D ⊠界框与 2D 浮点精度⊠界框 (BOX2DF) 相交，⊠返回 TRUE。
- &&&(geometry,gidx) - 如果几何体的（⊠存的）n ⊠⊠界框与 n ⊠浮点精度⊠界框 (GIDX) 相交，⊠返回 TRUE。
- &&&(gidx,geometry) - 如果 n ⊠浮点精度⊠界框 (GIDX) 与几何体的（⊠存的）n ⊠⊠界框相交，⊠返回 TRUE。
- &&&(gidx,gidx) - 如果⊠个 n ⊠浮点精度⊠界框 (GIDX) 彼此相交，⊠返回 TRUE。

13.11 PostGIS 函数支持矩⊠

下面按字母⊠序列出了 PostGIS 中的空⊠特殊函数以及它⊠使用的空⊠⊠型或它⊠⊠⊠遵守的 OGC/SQL 合⊠性。

- A ✔ 表示⊠函数本身适用于⊠型或子⊠型。
- A 意味着它可以工作，但会使用内置的⊠⊠操作，通⊠将其⊠制⊠⊠⊠几何⊠型、⊠⊠到“最佳 SRID（空⊠参考 ID）”空⊠参考，然后再次⊠制⊠⊠回来。⊠于大面⊠地区或极地地区，⊠果可能不如⊠期，并且可能会累⊠浮点数的⊠差。⊠是在地理数据⊠理中可能遇到的一些注意事⊠。
- A 表示⊠函数适用于⊠⊠型，因⊠它会自⊠⊠⊠⊠⊠一个⊠型（例如 box3d），而不是直接⊠型支持。
- A 表示⊠功能⊠在使用 SFCGAL 支持⊠⊠ PostGIS ⊠可用。
- geom - 基本 2D 几何支持 (x,y)。
- geog - 基本 2D 地理支持 (x,y)。

- 2.5D - 3 D/4D 空⌧中的基本 2D 几何⌧形（具有 Z 或 M 坐⌧）。
- PS - 多面体曲面
- T - 三角形和不⌧⌧三角网曲面 (TIN)

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Collect	✓		✓	✓			
ST_LineFromMu	✓ Point		✓				
ST_MakeEnvelo	✓						
ST_MakeLine	✓		✓				
ST_MakePoint	✓		✓				
ST_MakePointM	✓						
ST_MakePolygo	✓		✓				
ST_Point	✓				✓		
ST_PointZ	✓						
ST_PointM	✓						
ST_PointZM	✓						
ST_Polygon	✓		✓		✓		
ST_TileEnvelop	✓						
ST_HexagonGri	✓						
ST_Hexagon	✓						
ST_SquareGrid	✓						
ST_Square	✓						
ST_Letters	✓						
GeometryType	✓		✓	✓		✓	✓
ST_Boundary	✓		✓		✓		
ST_BoundingDi	✓ nal		✓				
ST_CoordDim	✓		✓	✓	✓	✓	✓
ST_Dimension	✓				✓	✓	✓
ST_Dump	✓		✓	✓		✓	✓
ST_DumpPoints	✓		✓	✓		✓	✓
ST_DumpSegme	✓ s		✓				✓
ST_DumpRings	✓		✓				
ST_EndPoint	✓		✓	✓	✓		
ST_Envelope	✓				✓		
ST_ExteriorRin	✓		✓		✓		

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_GeometryN	✓		✓	✓	✓	✓	✓
ST_GeometryTy	✓		✓		✓	✓	
ST_HasArc	✓		✓	✓			
ST_InteriorRing	✓		✓		✓		
ST_NumCurves	✓		✓		✓		
ST_CurveN	✓		✓		✓		
ST_IsClosed	✓		✓	✓	✓	✓	
ST_IsCollection	✓		✓	✓			
ST_IsEmpty	✓			✓	✓		
ST_IsPolygonCC	✓		✓				
ST_IsPolygonCW	✓		✓				
ST_IsRing	✓				✓		
ST_IsSimple	✓		✓		✓		
ST_M	✓		✓		✓		
ST_MemSize	✓		✓	✓		✓	✓
ST_NDims	✓		✓				
ST_NPoints	✓		✓	✓		✓	
ST_NRings	✓		✓	✓			
ST_NumGeometries	✓		✓		✓	✓	✓
ST_NumInteriorRings	✓				✓		
ST_NumInteriorRing	✓						
ST_NumPatches	✓		✓		✓	✓	
ST_NumPoints	✓				✓		
ST_PatchN	✓		✓		✓	✓	
ST_PointN	✓		✓	✓	✓		
ST_Points	✓		✓	✓			
ST_StartPoint	✓		✓	✓	✓		
ST_Summary	✓	✓		✓		✓	✓
ST_X	✓		✓		✓		
ST_Y	✓		✓		✓		
ST_Z	✓		✓		✓		
ST_Zmflag	✓		✓	✓			
ST_HasZ	✓		✓				
ST_HasM	✓		✓				

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_AddPoint	✓		✓				
ST_CollectionEx	✓ ct						
ST_CollectionH	✓ ogenize						
ST_CurveToLine	✓		✓	✓	✓		
ST_Scroll	✓		✓				
ST_FlipCoordina	✓ s		✓	✓		✓	✓
ST_Force2D	✓		✓	✓		✓	
ST_Force3D	✓		✓	✓		✓	
ST_Force3DZ	✓		✓	✓		✓	
ST_Force3DM	✓			✓			
ST_Force4D	✓		✓	✓			
ST_ForceCollec	✓ ı		✓	✓		✓	
ST_ForceCurve	✓		✓	✓			
ST_ForcePolygo	✓ CW		✓				
ST_ForcePolygo	✓ W		✓				
ST_ForceSFS	✓		✓	✓		✓	✓
ST_ForceRHR	✓		✓			✓	
ST_LineExtend	✓						
ST_LineToCurve	✓		✓	✓			
ST_Multi	✓						
ST_Normalize	✓						
ST_Project	✓	✓					
ST_QuantizeCo	✓ inates						
ST_RemovePoin	✓		✓				
ST_RemoveRep	✓ edPoints		✓			✓	
ST_RemoveIrrel	✓ ıntPointsForView						
ST_RemoveSma	✓ ıarts						
ST_Reverse	✓		✓			✓	
ST_Segmentize	✓	✓					
ST_SetPoint	✓		✓				
ST_ShiftLongitu	✓		✓			✓	✓
ST_WrapX	✓		✓				
ST_SnapToGrid	✓		✓				
ST_Snap	✓						

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_SwapOrdina	✓		✓	✓		✓	✓
ST_IsValid	✓				✓		
ST_IsValidDetai	✓						
ST_IsValidReas	✓						
ST_MakeValid	✓		✓				
ST_InverseTran	✓	rmPipeline					
ST_SetSRID	✓			✓			
ST_SRID	✓			✓	✓		
ST_Transform	✓			✓	✓	✓	
ST_TransformPi	✓	ine					
postgis_srs_codes							
postgis_srs							
postgis_srs_all							
postgis_srs_sea	✓						
ST_BdPolyFrom	✓	t					
ST_BdMPolyFro	✓	ext					
ST_GeogFromText		✓					
ST_GeographyFromText		✓					
ST_GeomCollFr	✓	Text			✓		
ST_GeomFromE	✓	CT	✓	✓		✓	✓
ST_GeomFromM	✓	RC21					
ST_GeometryFr	✓	Text			✓		
ST_GeomFromT	✓			✓	✓		
ST_LineFromTe	✓				✓		
ST_MLineFrom	✓	t			✓		
ST_MPointFrom	✓	t			✓		
ST_MPolyFrom	✓				✓		
ST_PointFromTe	✓				✓		
ST_PolygonFro	✓	xt			✓		
ST_WKTToSQL	✓				✓		
ST_GeogFromWKB		✓		✓			
ST_GeomFromE	✓	KB	✓	✓		✓	✓
ST_GeomFromW	✓	B		✓	✓		
ST_LineFromWI	✓				✓		
ST_LinestringFr	✓	WKB			✓		

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_PointFromW	✓		✓	✓	✓		
ST_WKBToSQL	✓				✓		
ST_Box2dFrom	☑	Hash					
ST_GeomFrom	✓	Hash					
ST_GeomFrom	✓		✓			✓	✓
ST_GeomFrom	✓	JSON	✓				
ST_GeomFromK	✓		✓				
ST_GeomFromT	✓	B					
ST_GMLToSQL	✓				✓		
ST_LineFromEn	✓	edPolyline					
ST_PointFromGeoHash							
ST_FromFlatGeobufToTable							
ST_FromFlatGeobuf							
ST_AsEWKT	✓	✓	✓	✓		✓	✓
ST_AsText	✓	✓		✓	✓		
ST_AsBinary	✓	✓	✓	✓	✓	✓	✓
ST_AsEWKB	✓		✓	✓		✓	✓
ST_AsHEXEWK	✓		✓	✓			
ST_AsEncodedP	✓	line					
ST_AsFlatGeobu	☑						
ST_AsGeobuf	☑						
ST_AsGeoJSON	✓	✓	✓				
ST_AsGML	✓	✓	✓		✓	✓	✓
ST_AsKML	✓	✓	✓				
ST_AsLatLonTe	✓						
ST_AsMARC21	✓						
ST_AsMVTGeon	✓						
ST_AsMVT	☑						
ST_AsSVG	✓	✓		✓			
ST_AsTWKB	✓						
ST_AsX3D	✓		✓			✓	✓
ST_GeoHash	✓			✓			
&&	✓	✓		✓		✓	
&&(geometry,b	✓	df)		✓		✓	
&&(box2df,geo	✓	ry)		✓		✓	

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
&&(box2df,box2df)	✓			✓		✓	
&&&	✓		✓	✓		✓	✓
&&&(geometry,gidx)	✓		✓	✓		✓	✓
&&&(gidx,geometry)	✓		✓	✓		✓	✓
&&&(gidx,gidx)			✓	✓		✓	✓
&<	✓						
&<|	✓			✓		✓	
&>	✓						
<<	✓						
<<|	✓						
=	✓	✓		✓		✓	
>>	✓						
@	✓						
@(geometry,box2df)	✓			✓		✓	
@(box2df,geometry)	✓			✓		✓	
@(box2df,box2df)	✓			✓		✓	
|&>	✓						
|>>	✓						
~	✓						
~(geometry,box2df)	✓			✓		✓	
~(box2df,geometry)	✓			✓		✓	
~(box2df,box2df)	✓			✓		✓	
~=	✓					✓	
<->	✓	✓					
|=|	✓						
<#>	✓						
<<->>	✓						
ST_3DIntersects	✓		✓		✓	✓	✓
ST_Contains	✓				✓		
ST_ContainsProperly	✓						
ST_CoveredBy	✓	✓					
ST_Covers	✓	✓					
ST_Crosses	✓				✓		
ST_Disjoint	✓				✓		

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Equals	✓				✓		
ST_Intersects	✓	✓		✓	✓		✓
ST_LineCrossin	✓	rection					
ST_OrderingEqu	✓	s			✓		
ST_Overlaps	✓				✓		
ST_Relate	✓				✓		
ST_RelateMatch							
ST_Touches	✓				✓		
ST_Within	✓				✓		
ST_3DDWithin	✓		✓		✓	✓	
ST_3DDFullyWi	✓	n	✓			✓	
ST_DFullyWithi	✓						
ST_DWithin	✓	✓					
ST_PointInsideC	✓	le					
ST_Area	✓	✓			✓	✓	
ST_Azimuth	✓	✓					
ST_Angle	✓						
ST_ClosestPoint	✓	✓					
ST_3DClosestPo	✓		✓			✓	
ST_Distance	✓	✓		✓	✓		
ST_3DDistance	✓		✓		✓	✓	
ST_DistanceSph	✓	e					
ST_DistanceSph	✓	oid					
ST_FrechetDista	✓	e					
ST_HausdorffDi	✓	nce					
ST_Length	✓	✓			✓		
ST_Length2D	✓						
ST_3DLength	✓		✓		✓		
ST_LengthSphe	✓	d	✓				
ST_LongestLine	✓						
ST_3DLongestL	✓		✓			✓	
ST_MaxDistanc	✓						
ST_3DMaxDista	✓	e	✓			✓	
ST_MinimumCl	✓	ance					

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_MinimumCl	✓	anceLine					
ST_Perimeter	✓	✓			✓		
ST_Perimeter2D	✓						
ST_3DPerimeter	✓		✓		✓		
ST_ShortestLine	✓	✓					
ST_3DShortestL	✓		✓			✓	
ST_ClipByBox2D	✓						
ST_Difference	✓		✓		✓		
ST_Intersection	✓	😁	✓		✓		
ST_MemUnion	✓		✓				
ST_Node	✓		✓				
ST_Split	✓						
ST_Subdivide	✓						
ST_SymDifferen	✓		✓		✓		
ST_UnaryUnion	✓		✓				
ST_Union	✓		✓		✓		
ST_Buffer	✓	😁			✓		
ST_BuildArea	✓						
ST_Centroid	✓	✓			✓		
ST_ChaikinSmo	✓	ing	✓				
ST_ConcaveHul	✓						
ST_ConvexHull	✓		✓		✓		
ST_DelaunayTri	✓	gles	✓				✓
ST_FilterByM	✓						
ST_GeneratePoi	✓						
ST_GeometricM	✓	an	✓				
ST_LineMerge	✓						
ST_MaximumIn	✓	bedCircle					
ST_LargestEmp	✓	ircle					
ST_MinimumBo	✓	lingCircle					
ST_MinimumBo	✓	lingRadius					
ST_OrientedEn	✓	pe					
ST_OffsetCurve	✓						
ST_PointOnSurf	✓		✓		✓		

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Polygonize	✓						
ST_ReducePrec	✓ n						
ST_SharedPaths	✓						
ST_Simplify	✓						
ST_SimplifyPres	✓ veTopology						
ST_SimplifyPoly	✓ nHull						
ST_SimplifyVW	✓						
ST_SetEffective	✓ a						
ST_TriangulateI	✓ gon						
ST_VoronoiLine	✓						
ST_VoronoiPoly	✓ s						
ST_CoverageInv	✓ dEdges						
ST_CoverageSir	✓ ify						
ST_CoverageUn	✓						
ST_Affine	✓		✓	✓		✓	✓
ST_Rotate	✓		✓	✓		✓	✓
ST_RotateX	✓		✓			✓	✓
ST_RotateY	✓		✓			✓	✓
ST_RotateZ	✓		✓	✓		✓	✓
ST_Scale	✓		✓	✓		✓	✓
ST_Translate	✓		✓	✓			
ST_TransScale	✓		✓	✓			
ST_ClusterDBS	✓ N			✓			
ST_ClusterInter	✓ ting						
ST_ClusterInter	✓ tingWin						
ST_ClusterKMe	✓						
ST_ClusterWith	✓			✓			
ST_ClusterWith	✓ Win			✓			
Box2D	✓			✓		✓	✓
Box3D	✓		✓	✓		✓	✓
ST_EstimatedEx	☑ nt			✓			
ST_Expand	✓					✓	✓
ST_Extent	✓					✓	✓
ST_3DExtent	✓		✓	✓		✓	✓

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_MakeBox2D	✓						
ST_3DMakeBox	✓						
ST_XMax	☑		✓	✓			
ST_XMin	☑		✓	✓			
ST_YMax	☑		✓	✓			
ST_YMin	☑		✓	✓			
ST_ZMax	☑		✓	✓			
ST_ZMin	☑		✓	✓			
ST_LineInterpol Point	✓	✓	✓				
ST_3DLineInter atePoint	✓		✓				
ST_LineInterpol Points	✓	✓	✓				
ST_LineLocateP t	✓	✓					
ST_LineSubstri	✓	✓	✓				
ST_LocateAlong	✓				✓		
ST_LocateBetw	✓				✓		
ST_LocateBetw Elevations	✓		✓				
ST_InterpolateP t	✓		✓				
ST_AddMeasure	✓		✓				
ST_IsValidTraje ry	✓		✓				
ST_ClosestPoint Approach	✓		✓				
ST_DistanceCPA	✓		✓				
ST_CPAWithin	✓		✓				
postgis.gdal_datapath							
postgis.gdal_enabled_drivers							
postgis.enable_outdb_rasters							
postgis.gdal_vsi_options							
PostGIS_AddBB	✓			✓			
PostGIS_DropBB	✓			✓			
PostGIS_HasBB	✓			✓			
postgis_sfcgal_version							
postgis_sfcgal_full_version							
CG_ForceLHR	🧊		🧊			🧊	🧊
CG_IsPlanar	🧊		🧊			🧊	🧊
CG_IsSolid	🧊		🧊			🧊	🧊
CG_MakeSolid	🧊		🧊			🧊	🧊

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
CG_Orientation							
CG_Area							
CG_3DArea							
CG_Volume							
ST_ForceLHR							
ST_IsPlanar							
ST_IsSolid							
ST_MakeSolid							
ST_Orientation							
ST_3DArea							
ST_Volume							
CG_Intersectio							
CG_Intersects							
CG_3DIntersec							
CG_Difference							
ST_3DDifferenc							
CG_3DDifferenc							
CG_Distance							
CG_3DDistance							
ST_3DConvexH							
CG_3DConvexH							
ST_3DIntersect							
CG_3DIntersec							
CG_Union							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_3DUnion							
CG_3DUnion							
ST_AlphaShape							
CG_AlphaShape							
CG_ApproxConvexPartition							
ST_ApproximateMedialAxis							
CG_ApproximateMedialAxis							
ST_ConstrainedDelaunayTriangles							
CG_ConstrainedDelaunayTriangles							
ST_Extrude							
CG_Extrude							
CG_ExtrudeStraightSkeleton							
CG_GreeneApproxConvexPartition							
ST_MinkowskiSum							
CG_MinkowskiSum							
ST_OptimalAlphaShape							
CG_OptimalAlphaShape							
CG_OptimalConvexPartition							
CG_StraightSkeleton							
ST_StraightSkeleton							
ST_Tesselate							
CG_Tesselate							
CG_Triangulate							
CG_Visibility							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
CG_YMonotone	✓						
getfaceedges_returntype							
TopoGeometry							
validatetopology_returntype							
TopoElement							
TopoElementArray							
AddTopoGeometryColumn							
RenameTopoGeometryColumn							
DropTopology							
RenameTopology							
DropTopoGeometryColumn							
Populate_Topology_Layer							
TopologySummary							
ValidateTopolog	✓						
ValidateTopologyRelation							
FindTopology							
FindLayer							
CreateTopology							
CopyTopology							
ST_InitTopoGeo					✓		
ST_CreateTopo	✓				✓		
TopoGeo_AddPo	✓						
TopoGeo_AddLi	✓ tring						
TopoGeo_AddPo	✓ on						
TopoGeo_LoadG	✓ metry						
ST_AddIsoNode	✓				✓		
ST_AddIsoEdge	✓				✓		
ST_AddEdgeNe	✓ aces				✓		
ST_AddEdgeMo	✓ ace				✓		
ST_RemEdgeNewFace					✓		
ST_RemEdgeModFace					✓		
ST_ChangeEdge	✓ om				✓		
ST_ModEdgeSp	✓				✓		
ST_ModEdgeHeal					✓		
ST_NewEdgeHeal					✓		
ST_MoveIsoNod	✓				✓		
ST_NewEdgesS	✓				✓		
ST_RemoveIsoNode					✓		
ST_RemoveIsoEdge					✓		
GetEdgeByPoin	✓						

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
GetFaceByPoint	✓						
GetFaceContainPoint	✓						
GetNodeByPoint	✓						
GetTopologyID							
GetTopologySRID							
GetTopologyName							
ST_GetFaceEdges					✓		
ST_GetFaceGeometry	✓				✓		
GetRingEdges							
GetNodeEdges							
Polygonize							
AddNode	✓						
AddEdge	✓						
AddFace	✓						
ST_Simplify	✓						
RemoveUnusedPrimitives	✓						
CreateTopoGeom	✓						
toTopoGeom	✓						
TopoElementArray_Agg							
TopoElement	✓						
clearTopoGeom	✓						
TopoGeom_addElement	✓						
TopoGeom_remElement	✓						
TopoGeom_addToGeom	✓						
toTopoGeom							
GetTopoGeomElementArray							
GetTopoGeomElements							
ST_SRID	✓				✓		
AsGML	✓						
AsTopoJSON	✓						
Equals	✓		✓				
Intersects	✓		✓				
geomval							
addbandarg							
rastbandarg							
raster							
reclassarg							
summarystats							
unionarg							
AddRasterConstraints							
DropRasterConstraints							
AddOverviewConstraints							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
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							
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	✓						

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_PixelAsPolygons							
ST_PixelAsPoint	✓						
ST_PixelAsPoints							
ST_PixelAsCentroid	✓						
ST_PixelAsCentroids							
ST_Value	✓						
ST_NearestValue	✓						
ST_SetZ	✓						
ST_SetM	✓						
ST_Neighborhood	✓						
ST_SetValue	✓						
ST_SetValues							
ST_DumpValues							
ST_PixelOfValue							
ST_SetGeoReference							
ST_SetRotation							
ST_SetScale							
ST_SetSkew							
ST_SetSRID							
ST_SetUpperLeft							
ST_Resample							
ST_Rescale							
ST_Reskew							
ST_SnapToGrid							
ST_Resize							
ST_Transform							
ST_SetBandNoDataValue							
ST_SetBandIsNoData							
ST_SetBandPath							
ST_SetBandIndex							
ST_Count							
ST_CountAgg							
ST_Histogram							
ST_Quantile							
ST_SummaryStats							
ST_SummaryStatsAgg							
ST_ValueCount							
ST_RastFromWKB							
ST_RastFromHexWKB							
ST_AsBinary/ST_AsWKB							
ST_AsHexWKB							
ST_AsGDALRaster							
ST_AsJPEG							
ST_AsPNG							
ST_AsTIFF							
ST_Clip	✓						
ST_ColorMap							
ST_Grayscale							
ST_Intersection	✓						

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
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							
ST_Envelope	✓						
ST_MinConvexH	✓						
ST_Polygon	✓						
&&	✓						
&<							
&>							
=							
@	✓						
~=							
~	✓						
ST_Contains							
ST_ContainsProperly							
ST_Covers							
ST_CoveredBy							
ST_Disjoint							
ST_Intersects	✓						
ST_Overlaps							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
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_Intersection	✓						
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 新的、增☒的或更改的 PostGIS 函数

13.12.1 PostGIS 新增功能或增☒功能（3.5）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 3.5

- CG_3DArea - 可用性：3.5.0 ☒算 3D 表面几何形状的面☒。☒于固体将返回 0。
- CG_3DConvexHull - 可用性：3.5.0 ☒算几何体的 3D 凸包。
- CG_3DDifference - 可用性：3.5.0 ☒行 3D 差异
- CG_3DDistance - 可用性：3.5.0 Computes the minimum 3D distance between two geometries
- CG_3DIntersection - 可用性：3.5.0 ☒行 3D 相交
- CG_3DIntersects - 可用性：3.5.0 Tests if two 3D geometries intersect

- CG_3DUnion - 可用性：3.5.0 Perform 3D union using postgis_sfcgal.
- CG_AlphaShape - Availability: 3.5.0 - requires SFCGAL >= 1.4.1. 区算包区几何体的 Alpha 形状
- CG_ApproxConvexPartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。区算多区形几何区形的近似凸分割
- CG_ApproximateMedialAxis - 可用性：3.5.0 区算几何区域的近似中区。
- CG_Area - 可用性：3.5.0 Calculates the area of a geometry
- CG_Difference - 可用性：3.5.0 Computes the geometric difference between two geometries
- CG_Distance - 可用性：3.5.0 Computes the minimum distance between two geometries
- CG_Extrude - 可用性：3.5.0 将曲面区出到相关体区
- CG_ExtrudeStraightSkeleton - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。直区骨架外凸
- CG_ForceLHR - 可用性：3.5.0 区制 LHR 方向
- CG_GreeneApproxConvexPartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。区算多区形几何区形的近似凸分割
- CG_Intersection - 可用性：3.5.0 Computes the intersection of two geometries
- CG_Intersects - 可用性：3.5.0 区区区个几何区形是否相交（它区至少有一个共同点）
- CG_IsPlanar - 可用性：3.5.0 区区表面是否平坦
- CG_IsSolid - 可用性：3.5.0 区区几何体是否区区体。不区行有效性区区。
- CG_MakeSolid - 可用性：3.5.0 将几何体区造成区体。不区行任何区区。要区得有效的区体，区入几何区形必区是区合多面体曲面或区合 TIN。
- CG_MinkowskiSum - 可用性：3.5.0 区行 Minkowski sum
- CG_OptimalAlphaShape - Availability: 3.5.0 - requires SFCGAL >= 1.4.1. 使用“最佳”alpha 区区算包区几何体的 Alpha 形状。
- CG_OptimalConvexPartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。区算多区形几何区形的最区凸分割
- CG_Orientation - 可用性：3.5.0 确定表面方向
- CG_StraightSkeleton - 可用性：3.5.0 从几何体区算直骨架
- CG_Tesselate - 可用性：3.5.0 区多区形或多面体表面区行曲面区分，并以 TIN 或 TINS 集合的形式返回
- CG_Triangulate - 可用性：3.5.0 Triangulates a polygonal geometry
- CG_Union - 可用性：3.5.0 Computes the union of two geometries
- CG_Visibility - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。区算一个从点或多区形几何中的区段生成的可区性多区形
- CG_Volume - 可用性：3.5.0 区算 3D 区体的体区。如果区用于表面（甚至区合）几何区形将返回 0。
- CG_YMonotonePartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。区算多区形几何的 y 区区分割
- ST_HasM - 可用性：3.5.0 区区几何体是否具有 M（区量）区度。
- ST_HasZ - 可用性：3.5.0 区区几何体是否具有 Z 区度。
- ST_RemoveIrrelevantPointsForView - 可用性：3.5.0 Removes points that are irrelevant for rendering a specific rectangluar view of a geometry.
- ST_RemoveSmallParts - 可用性：3.5.0 Removes small parts (polygon rings or linestrings) of a geometry.

- TopoGeo_LoadGeometry - 可用性：3.5.0 Load a geometry into an existing topology, snapping and splitting as needed.

PostGIS 的功能增☒ 3.5

- ST_Clip - 增☒版：3.5.0 - 添加了 touched 参数。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.

PostGIS 中的功能☒生☒化 3.5

- ST_AsGeoJSON - 更改：3.5.0 允☒指定包含要素 ID 的列以 GeoJSON 格式返回一个几何体或要素。
- ST_DFullyWithin - Changed: 3.5.0 : the logic behind the function now uses a test of containment within a buffer, rather than the ST_MaxDistance algorithm. Results will differ from prior versions, but should be closer to user expectations. Tests if a geometry is entirely inside a distance of another

13.12.2 PostGIS 新增功能或增☒功能（3.4）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 3.4

- PostGIS_GEOS_Compiled_Version - 可用性：3.4.0 返回☒建 PostGIS 所依据的 GEOS ☒的版本号。
- PostGIS_PROJ_Compiled_Version - 可用性：3.5.0 Returns the version number of the PROJ library against which PostGIS was built.
- RenameTopoGeometryColumn - 可用性：3.4.0 重命名拓扑几何列
- RenameTopology - 可用性：3.4.0 重命名拓扑
- ST_ClusterIntersectingWin - 可用性：3.4.0 窗口函数，返回每个☒入几何☒形的簇 ID，将☒入几何☒形聚☒到☒接的集合中。
- ST_ClusterWithinWin - 可用性：3.4.0 窗口函数，返回每个☒入几何☒形的簇 ID，使用分离距离☒行聚☒。
- ST_CoverageInvalidEdges - 可用性：3.4.0 用于☒找多☒形无法形成有效覆盖范☒的位置的窗口函数。
- ST_CoverageSimplify - 可用性：3.4.0 ☒化多☒形覆盖范☒☒☒的窗口函数。
- ST_CoverageUnion - 可用性：3.4.0-需要 GEOS >= 3.8.0 通☒☒除共享☒来☒算形成覆盖范☒的一☒多☒形的并集。
- ST_InverseTransformPipeline - 可用性：3.4.0 返回一个新的几何体，其坐☒☒使用定☒的坐☒☒☒管道的逆☒☒☒☒到不同的空☒参考系。
- ST_LargestEmptyCircle - 可用性：3.4.0。☒算不与几何☒形重☒的最大☒。
- ST_LineExtend - 可用性：3.4.0 返回一条☒，向前和向后延伸指定的距离。
- ST_TransformPipeline - 可用性：3.4.0 返回一个新的几何☒形，其坐☒使用定☒的坐☒☒☒管道☒☒☒不同的空☒参考系☒。
- TopoElement - 可用性：3.4.0 将拓扑几何☒☒☒拓扑元素。
- debug_standardize_address - 可用性：3.4.0 返回 json 格式的文本，列出解析☒☒和☒准化
- postgis_srs - 可用性：3.4.0 返回所☒求的☒限和 srid 的元数据☒☒。
- postgis_srs_all - 可用性：3.4.0 返回底☒ Proj 数据☒中每个空☒参考系☒的元数据☒☒。

- postgis_srs_codes - 可用性：3.4.0 返回与⌧定⌧限关⌧的 SRS 代⌧列表。
- postgis_srs_search - 可用性：3.4.0 返回具有完全包含⌧界参数的使用区域的投影坐⌧系的元数据⌧⌧。

PostGIS 的功能增⌧ 3.4

- PostGIS_Full_Version - 增⌧功能：3.4.0 ⌧在包括⌧外的 PROJ 配置 NETWORK_ENABLED、URL_ENDPOINT 和 proj.db 位置的 DATABASE_PATH ⌧告完整的 PostGIS 版本和⌧建配置信息。
- PostGIS_PROJ_Version - 增⌧功能：3.4.0 ⌧在包括 proj.db 位置的 NETWORK_ENABLED、URL_ENDPOINT 和 DATABASE_PATH 返回 PROJ4 ⌧的版本号。
- ST_AsSVG - 增⌧：3.4.0 支持所有曲⌧⌧型返回几何体的 SVG 路径数据。
- ST_ClosestPoint - 增⌧：3.4.0 - 支持地理。返回 g1 上最接近 g2 的 2D 点。⌧是从一个几何体到⌧一个几何体的最短直⌧的第一个点。
- ST_LineSubstring - 增⌧：3.4.0 - 引入了⌧地理的支持。返回⌧个小数位置之⌧的直⌧部分。
- ST_Project - 增⌧：3.4.0 允⌧几何参数和无方位角的⌧点格式。返回从起点按距离和方位角（方位角）投影的点。
- ST_Resample - 增⌧：3.4.0 添加了最大和最小重采⌧⌧⌧重采⌧一个⌧格⌧像，可以指定重新采⌧算法、新的尺寸、任意的⌧格角点，以及一⌧⌧格地理参考属性，⌧些属性可以自己定⌧，也可以从⌧一个⌧格⌧像中借用。
- ST_Rescale - 增⌧：3.4.0 添加了最大和最小重采⌧⌧⌧通⌧⌧⌧整⌧格的比例（或像素大小）来重新采⌧⌧格。新的像素⌧是使用 NearestNeighbor（英⌧或美式拼写）、Bilinear、Cubic、CubicSpline、Lanczos、Max 或 Min 重采⌧算法⌧算的。默⌧⌧ NearestNeighbor。
- ST_ShortestLine - 增⌧：3.4.0 - 支持地理。返回⌧个几何⌧形之⌧的 2D 最短⌧

PostGIS 中的功能⌧生⌧化 3.4

- PostGIS_Extensions_Upgrade - 更改：3.4.0 添加 target_version 参数。将 PostGIS ⌧展（例如 postgis_raster、postgis_topology、postgis_sfcgal）打包并升⌧到⌧定版本或最新版本。

13.12.3 PostGIS 新增功能或增⌧功能（3.3）

下面⌧出的功能是添加或增⌧的 PostGIS 功能。

PostGIS 中的新功能 3.3

- RemoveUnusedPrimitives - 可用性：3.3.0 ⌧除定⌧⌧有 TopoGeometry ⌧象不需要的拓扑基元。
- ST_3DConvexHull - 可用性：3.3.0 ⌧算几何体的 3D 凸包。
- ST_3DUnion - 可用性：3.3.0 添加了聚合函数格式⌧行 3D ⌧合。
- ST_AsMARC21 - 可用性：3.3.0 将几何⌧形返回⌧⌧有地理数据字段 (034) 的 MARC21/XML ⌧⌧。
- ST_GeomFromMARC21 - 可用性：3.3.0, 需要 libxml2 2.6+ ⌧入 MARC21/XML 地理数据并返回 PostGIS 几何⌧象。
- ST_Letters - 可用性：3.3.0 返回渲染⌧几何⌧形的⌧入字母，默⌧起始位置位于原点，默⌧文本高度⌧ 100。
- ST_OptimalAlphaShape - 可用性：3.3.0 - 需要 SFCGAL >= 1.4.1。使用“最佳”alpha ⌧⌧算包⌧几何体的 Alpha 形状。
- ST_SimplifyPolygonHull - 可用性：3.3.0。⌧算多⌧形几何的⌧化的保留拓扑的外部或内部外壳。
- ST_TriangulatePolygon - 可用性：3.3.0。⌧算多⌧形的⌧束 Delaunay 三角剖分

- postgis_sfcgal_full_version - 可用性：3.3.0 返回正在使用的 SFCGAL 的完整版本，包括 CGAL 和 Boost 版本

PostGIS 的功能增☒ 3.3

- ST_ConcaveHull - 增☒：3.3.0，☒ GEOS 3.11+ ☒用 GEOS 本机☒☒☒算包含所有☒入几何☒点的可能凹几何
- ST_LineMerge - 增☒：3.3.0 接受定向参数。返回通☒将 MultiLineString ☒合在一起形成的☒。

PostGIS 中的功能☒生☒化 3.3

- PostGIS_Extensions_Upgrade - 更改：3.3.0 支持从任何 PostGIS 版本升☒。不适用于所有系☒。将 PostGIS ☒展（例如 postgis_raster、postgis_topology、postgis_sfcgal）打包并升☒到☒定版本或最新版本。

13.12.4 PostGIS 新增功能或增☒功能（3.2）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 3.2

- FindLayer - 可用性：3.2.0 通☒不同的方式返回 topology.layer ☒☒。
- FindTopology - 可用性：3.2.0 通☒不同的方式返回拓扑☒☒。
- GetFaceContainingPoint - 可用性：3.2.0 ☒找包含点的面。
- ST_AsFlatGeobuf - 可用性：3.2.0 返回一☒行的 FlatGeobuf 表示形式。
- ST_Contour - 可用性：3.2.0 使用 GDAL ☒廓算法从提供的☒格波段生成一☒矢量☒廓。
- ST_DumpSegments - 可用性：3.2.0 ☒几何☒形中的各个☒段返回一☒ geometry_dump 行。
- ST_FromFlatGeobuf - 可用性：3.2.0 ☒取 FlatGeobuf 数据。
- ST_FromFlatGeobufToTable - 可用性：3.2.0 根据 FlatGeobuf 数据的☒☒☒建一个表。
- ST_InterpolateRaster - 可用性：3.2.0 基于☒入的 3 ☒点集插☒网格表面，使用 X 和 Y ☒在网格上定位点，并使用点的 Z ☒作☒表面高程。
- ST_SRID - 可用性：3.2.0 返回拓扑几何的空☒参考☒☒符。
- ST_Scroll - 可用性：3.2.0 更改☒合☒串的起点。
- ST_SetM - 可用性：3.2.0 返回与☒入几何☒形具有相同 X/Y 坐☒的几何☒形，并使用☒求的重采☒算法将☒格中的☒复制到 M ☒度。
- ST_SetZ - 可用性：3.2.0 返回与☒入几何☒形具有相同 X/Y 坐☒的几何☒形，并使用☒求的重采☒算法将☒格中的☒复制到 Z ☒度。
- TopoGeom_addTopoGeom - 可用性：3.2 将一个 TopoGeometry 的元素添加到☒一个 TopoGeometry 的定☒中。
- ValidateTopologyRelation - 可用性：3.2.0 返回有关无效拓扑关系☒☒的信息
- postgis.gdal_vsi_options - 可用性：3.2.0 用于☒置☒理外部数据☒☒格☒使用的☒☒的字符串配置。

PostGIS 的功能增☒ 3.2

- GetFaceByPoint - 增☒：3.2.0 更高效的☒施和更清晰的契☒，停止使用无效的拓扑。☒找与☒定点相交的面。

- ST_ClusterKMeans - 增☒：3.2.0 支持 max_radius 使用 K 均☒算法返回每个☒入几何☒形的簇 id 的窗口函数。
- ST_MakeValid - 增☒：在 3.2.0 中，添加了可☒的算法参数“linework”和“structure”。需要 GEOS>= 3.10.0 或更高版本。☒☒在不☒失☒点的情况下使无效几何体有效。
- ST_PixelAsCentroid - 增☒：3.2.0 Faster ☒在用 C ☒☒。返回像素表示的区域的☒心（点几何）。
- ST_PixelAsCentroids - 增☒：3.2.0 Faster ☒在用 C ☒☒。返回☒格波段的每个像素的☒心（点几何）以及每个像素的☒、X 和 Y ☒格坐☒。点几何是像素表示的区域的☒心。
- ST_Point - 增☒：3.2.0 srid 作☒☒外的可☒参数被添加。☒旧的安装需要与 ST_SetSRID ☒合以在几何体上☒☒ srid。☒建具有 X、Y 和 SRID ☒的点。
- ST_PointM - 增☒：3.2.0 srid 作☒☒外的可☒参数被添加。☒旧的安装需要与 ST_SetSRID ☒合以在几何体上☒☒ srid。☒建具有 X、Y、M 和 SRID ☒的点。
- ST_PointZ - 增☒：3.2.0 srid 作☒☒外的可☒参数被添加。☒旧的安装需要与 ST_SetSRID ☒合以在几何体上☒☒ srid。☒建具有 X、Y、Z 和 SRID ☒的点。
- ST_PointZM - 增☒：3.2.0 srid 作☒☒外的可☒参数被添加。☒旧的安装需要与 ST_SetSRID ☒合以在几何体上☒☒ srid。☒建具有 X、Y、Z、M 和 SRID ☒的点。
- ST_RemovePoint - 增☒：3.2.0 从☒串中☒除一个点。
- ST_RemoveRepeatedPoints - 增☒：3.2.0 返回☒除了重复点的几何☒形。
- ST_StartPoint - 增☒：3.2.0 返回所有几何☒形的点。如果☒入不是 LineString，☒先前的行☒将返回 NULL。返回 LineString 的第一个点.
- ST_Value - 增☒：添加了 3.2.0 重新采☒可☒参数。返回指定列 x、行 y 像素或特定几何点☒☒定波段的☒。波段☒号从 1 开始，如果未指定，☒默☒☒ 1。如果将参数 exclude_nodata_value ☒置☒ false，☒所有像素都被☒☒与 nodata 像素相交并返回其☒。如果未☒☒参数 exclude_nodata_value，☒从☒格的元数据中☒取☒☒。
- 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.

PostGIS 中的功能☒生☒化 3.2

- ST_Boundary - 更改：3.2.0 支持 TIN，不使用地理，不☒性化曲☒返回几何☒形的☒界。
- ValidateTopology - 更改：3.2.0 添加了可☒的 bbox 参数，☒行面☒☒和☒☒接☒☒。返回一☒ validatetopology_returntype ☒象，☒☒☒明拓扑☒☒。

13.12.5 PostGIS 新增功能或增☒功能（3.1）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 3.1

- ST_Hexagon - 可用性：3.1.0 使用提供的☒尺寸和六☒形网格空☒内的☒元坐☒返回☒个六☒形。
- ST_HexagonGrid - 可用性：3.1.0 返回一☒完全覆盖几何参数☒界的六☒形和☒元格索引。
- ST_MaximumInscribedCircle - 可用性：3.1.0。☒算几何体中包含的最大☒。
- ST_ReducePrecision - 可用性：3.1.0。返回有效的几何☒形，其点舍入到网格公差。
- ST_Square - 可用性：3.1.0 使用提供的☒☒大小和正方形网格空☒内的☒元格坐☒返回☒个正方形。
- ST_SquareGrid - 可用性：3.1.0 返回一☒完全覆盖几何参数☒界的网格正方形和☒元格索引。

PostGIS 的功能增⌧ 3.1

- ST_AsEWKT - 增⌧：3.1.0 支持可⌧精度参数。使用 SRID 元数据返回几何⌧形的已知文本 (WKT) 表示形式。
- ST_ClusterKMeans - 增⌧：3.1.0 支持 3D 几何和⌧重使用 K 均⌧算法返回每个⌧入几何⌧形的簇 id 的窗口函数。
- ST_Difference - 增⌧：3.1.0 接受 gridSize 参数。⌧算表示几何 A 中不与几何 B 相交的部分的几何。
- ST_Intersection - 增⌧：3.1.0 接受 gridSize 参数⌧算表示几何 A 和 B 的共享部分的几何。
- ST_MakeValid - 增⌧：3.1.0 ⌧除了具有 NaN ⌧的坐⌧。⌧⌧在不⌧失⌧点的情况下使无效几何体有效。
- ST_Subdivide - 增⌧：3.1.0 接受 gridSize 参数。⌧算几何体的直⌧⌧分。
- ST_SymDifference - 增⌧：3.1.0 接受 gridSize 参数。⌧算表示几何⌧形 A 和 B 不相交部分的几何⌧形。
- ST_TileEnvelope - 增⌧：添加了 3.1.0 margin 参数。使用 XYZ 切片系⌧在 Web Mercator（SRID：3857）中⌧建矩形多⌧形。
- ST_UnaryUnion - 增⌧：3.1.0 接受 gridSize 参数。⌧算⌧个几何体的⌧件的并集。
- ST_Union - 增⌧：3.1.0 接受 gridSize 参数。⌧算表示⌧入几何⌧形的点集并集的几何⌧形。

PostGIS 中的功能⌧生⌧化 3.1

- ST_Count - 更改：3.1.0 - ⌧除了 ST_Count(rastertable, rastercolumn, ...) ⌧体。⌧改用。返回⌧格或⌧格覆盖范⌧的⌧定波段中的像素数。如果未指定 band，⌧默⌧⌧ band 1。如果 except_nodata_value ⌧置⌧ true，⌧⌧⌧算不等于 nodata ⌧的像素。
- ST_Force3D - 更改：3.1.0. 您⌧在可以指定一个非零 Z ⌧。⌧制几何⌧形⌧入 XYZ 模式。⌧是 ST_Force3DZ 的⌧名。
- ST_Force3DM - 更改：3.1.0. ⌧在可以指定一个非零 M ⌧。⌧制几何⌧形⌧入 XYM 模式。
- ST_Force3DZ - 更改：3.1.0. 您⌧在可以指定一个非零 Z ⌧。⌧制几何⌧形⌧入 XYZ 模式。
- ST_Force4D - 更改：3.1.0. ⌧在可以指定非零 Z 和 M ⌧。⌧制几何⌧形⌧入 XYZM 模式。
- ST_Histogram - 更改：3.1.0 ⌧除了 ST_Histogram(table_name, column_name) ⌧体。返回一⌧⌧⌧，⌧⌧了⌧格或⌧格覆盖数据分布，其中包括分离的分箱范⌧。如果未指定，将自⌧⌧算分箱数。
- ST_Quantile - 更改：3.1.0 ⌧除了 ST_Quantile(table_name, column_name) ⌧体。⌧算⌧本或⌧体上下文中⌧格或⌧格表覆盖范⌧的分位数。因此，可以⌧⌧某个⌧是否位于⌧格的 25%、50%、75% 百分位⌧。
- ST_SummaryStats - 更改：3.1.0 ST_SummaryStats(rastertable, rastercolumn, ...) ⌧体已⌧除。⌧改用。返回一⌧⌧⌧⌧⌧信息，包括⌧定⌧格或⌧格覆盖的⌧格⌧的⌧数、⌧和、均⌧、⌧准差、最小⌧和最大⌧。如果未指定⌧号，⌧假定⌧⌧ 1。

13.12.6 PostGIS 新增功能或增⌧功能（3.0）

下面⌧出的功能是添加或增⌧的 PostGIS 功能。

PostGIS 中的新功能 3.0

- CG_ConstrainedDelaunayTriangles - 可用性：3.0.0 返回⌧⌧⌧定⌧入几何体的⌧束 Delaunay 三角剖分。
- ST_3DLineInterpolatePoint - 可用性：3.0.0 返回沿 3D ⌧的小数指示位置插⌧的点。
- ST_ConstrainedDelaunayTriangles - 可用性：3.0.0 返回⌧⌧⌧定⌧入几何体的⌧束 Delaunay 三角剖分。
- ST_TileEnvelope - 可用性：3.0.0 使用 XYZ 切片系⌧在 Web Mercator （SRID：3857）中⌧建矩形多⌧形。

PostGIS 的功能增☒ 3.0

- ST_AsMVT - 增☒：3.0 - 添加了☒要素 ID 的支持。返回一☒行的 MVT 表示形式的聚合函数。
- ST_Contains - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒ B 的每个点是否都位于 A 中，并且它☒的内部是否有一个共同点
- ST_ContainsProperly - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒ B 的每个点是否都位于 A 的内部
- ST_CoveredBy - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒ A 的每个点是否都位于 B 中
- ST_Covers - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒ B 的每个点是否都位于 A 中
- ST_Crosses - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒☒个几何☒形是否有一些（但不是全部）共同的内点
- ST_CurveToLine - 增☒：3.0.0 ☒☒了每弧的最小☒性分数。防止拓扑崩☒。将包含曲☒的几何☒形☒☒☒☒性几何☒形。
- ST_Disjoint - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒☒个几何☒形是否没有共同点
- ST_Equals - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒☒个几何☒形是否包含同一☒点
- ST_GeneratePoints - 增☒：3.0.0，添加种子参数生成一个包含在多☒形（Polygon）或多重多☒形（MultiPolygon）内的随机点的多点☒象。
- ST_GeomFromGeoJSON - 增☒：3.0.0 如果未指定其他 SRID 解析几何，☒默☒ SRID ☒ 4326。将几何☒形的 geojson 表示形式作☒☒入并☒出 PostGIS 几何☒象
- ST_LocateBetween - 增☒：3.0.0 - 添加了☒多☒形、TIN、三角形的支持。返回与☒量范☒匹配的几何☒形部分。
- ST_LocateBetweenElevations - 增☒：3.0.0 - 添加了☒多☒形、TIN、三角形的支持。返回位于高程 (Z) 范☒内的几何☒形部分。
- ST_Overlaps - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒☒个几何☒形是否具有相同的☒度和相交，但每个几何☒形至少有一个点不在☒一个几何☒形中
- ST_Relate - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒☒个几何☒形是否具有与交集矩☒模式匹配的拓扑关系，或☒算它☒的交集矩☒
- ST_Segmentize - 增☒：3.0.0 分段几何☒在可生成等☒的子分段返回修改后的几何☒形/地理，其☒段不☒于☒定距离。
- ST_Touches - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒☒个几何☒形是否至少有一个共同点，但它☒的内部不相交
- ST_Within - 增☒：3.0.0 ☒用了☒ GEOMETRYCOLLECTION 的支持☒☒ A 的每个点是否都位于 B 中，并且它☒的内部是否有一个共同点

PostGIS 中的功能☒生☒化 3.0

- PostGIS_Extensions_Upgrade - 更改：3.0.0 重新打包松散☒展并支持 postgis_raster。将 PostGIS ☒展（例如 postgis_raster、postgis_topology、postgis_sfcgal）打包并升☒到☒定版本或最新版本。
- ST_3DDistance - 更改：3.0.0 - SFCGAL 版本已☒除返回☒个几何☒形之☒的 3D 笛卡☒最小距离（基于空☒参考）（以投影☒位表示）。
- ST_3DIntersects - 更改：3.0.0 ☒除了 SFCGAL 后端，GEOS 后端支持 TIN。☒☒☒个几何☒形在 3D 空☒中是否相交 - ☒适用于点、☒串、多☒形、多面体曲面（区域）
- ST_Area - 更改：3.0.0 - 不再依☒ SFCGAL。返回多☒形几何体的面☒。

- ST_AsGeoJSON - 更改：3.0.0 支持⊠⊠作⊠⊠入以 GeoJSON 格式返回一个几何体或要素。
- ST_AsGeoJSON - 更改：3.0.0 ⊠出 SRID（如果不是 EPSG：4326）。以 GeoJSON 格式返回一个几何体或要素。
- ST_AsKML - 更改：3.0.0 - ⊠除了“版本化”⊠体⊠名将几何⊠形作⊠ KML 元素返回。
- ST_Distance - 更改：3.0.0 - 不再依⊠ SFCGAL。返回⊠个几何或地理⊠之⊠的距离。
- ST_Intersection - 更改：3.0.0 不依⊠于 SFCGAL。⊠算表示几何 A 和 B 的共享部分的几何。
- ST_Intersects - 更改：3.0.0 ⊠除了 SFCGAL 版本并添加了⊠ 2D TINS 的本机支持。⊠⊠⊠个几何⊠形是否相交（它⊠至少有一个共同点）
- ST_Union - 更改：3.0.0 不依⊠于 SFCGAL。⊠算表示⊠入几何⊠形的点集并集的几何⊠形。

13.12.7 PostGIS 新增功能或增⊠功能（2.5）

下面⊠出的功能是添加或增⊠的 PostGIS 功能。

PostGIS 中的新功能 2.5

- PostGIS_Extensions_Upgrade - 可用性：2.5.0 将 PostGIS ⊠展（例如 postgis_raster、postgis_topology、postgis_sfcgal）打包并升⊠到⊠定版本或最新版本。
- ST_Angle - 可用性：2.5.0 返回由 3 或 4 个点或 2 条⊠定⊠的⊠个向量之⊠的角度。
- ST_AsHexWKB - 可用性：2.5.0 返回⊠格的十六⊠制表示形式的熟知的二⊠制 (WKB)。
- ST_BandFileSize - 可用性：2.5.0 返回文件系⊠中存⊠的波段的文件大小。如果未指定波段号，⊠假定⊠ 1。
- ST_BandFileTimestamp - 可用性：2.5.0 返回文件系⊠中存⊠的波段的文件⊠⊠戳。如果未指定波段号，⊠假定⊠ 1。
- ST_ChaikinSmoothing - 可用性：2.5.0 使用 Chaikin 算法返回几何⊠形的平滑版本
- ST_FilterByM - 可用性：2.5.0 根据 M ⊠⊠除⊠点
- ST_Grayscale - 可用性：2.5.0 根据源⊠格和代表⊠色、⊠色和⊠色的指定波段⊠建新的 1-8BUI 波段⊠格
- ST_LineInterpolatePoints - 可用性：2.5.0 返回沿直⊠以分数⊠隔插⊠的点。
- ST_OrientedEnvelope - 可用性：2.5.0。返回包含几何⊠形的最小面⊠矩形。
- ST_QuantizeCoordinates - 可用性：2.5.0 将坐⊠的最低有效位⊠置⊠零
- ST_RastFromHexWKB - 可用性：2.5.0 从熟知的二⊠制 (WKB) ⊠格的十六⊠制表示形式返回⊠格⊠。
- ST_RastFromWKB - 可用性：2.5.0 从熟知的二⊠制 (WKB) ⊠格返回⊠格⊠。
- ST_SetBandIndex - 可用性：2.5.0 更新 out-db band 的 external band ⊠号
- ST_SetBandPath - 可用性：2.5.0 更新 out-db band 的外部路径和 band ⊠号

PostGIS 的功能增⊠ 2.5

- ST_AsBinary/ST_AsWKB - 增⊠：2.5.0 添加 ST_AsWKB 返回⊠格的熟知的二⊠制 (WKB) 表示形式。
- ST_AsMVT - 增⊠：2.5.0 - 添加了⊠并行⊠⊠的支持。返回一⊠行的 MVT 表示形式的聚合函数。
- ST_AsText - 增⊠：2.5 - 引入了可⊠的精度参数。返回不⊠ SRID 元数据的几何/地理的已知文本 (WKT) 表示形式。

- ST_BandMetaData - 增☒：2.5.0 包括 outdbbandnum、文件大小和文件☒☒戳 for outdb ☒格。返回特定☒格波段的基本元数据。如果未指定，☒假定波段☒号☒ 1。
- ST_Buffer - 增☒：2.5.0 - ST_Buffer 的几何感知版本已得到增☒,允☒您指定要☒冲的一☒。side=both|left|right。☒算覆盖距几何体☒定距离内所有点的几何体。
- ST_GeomFromGeoJSON - 增☒：2.5.0 ☒在可以接受 json 和 jsonb 作☒☒入。将几何☒形的 geojson 表示形式作☒☒入并☒出 PostGIS 几何☒象
- ST_GeometricMedian - 增☒：2.5.0 添加了☒ M 作☒点☒重的支持。返回多点的几何中位数。
- ST_Intersects - 增☒：2.5.0 支持 GEOMETRYCOLLECTION。☒☒☒个几何☒形是否相交（它☒至少有一个共同点）
- ST_OffsetCurve - 增☒：2.5 - 添加了☒ GEOMETRYCOLLECTION 和 MULTILINESTRING 的支持返回距☒入☒☒定距离和方向的偏移☒。
- ST_Scale - 增☒：2.5.0 引入了☒相☒于本地原点（origin 参数）☒行☒放的支持。按☒定因子☒放几何☒形。
- ST_Split - 增☒：2.5.0 引入了通☒多☒分割多☒形的支持。返回通☒将一个几何体分割☒☒一个几何体而☒建的几何体集合。
- ST_Subdivide - 增☒：2.5.0 重用多☒形分割上的☒有点，☒点数从 8 ☒少到 5。☒算几何体的直☒☒分。

PostGIS 中的功能☒生☒化 2.5

- ST_GDALDrivers - 更改：2.5.0 - 添加 can_read 和 can_write 列。通☒ GDAL 返回 PostGIS 支持的☒格格式列表。ST_AsGDALRaster ☒可使用 can_write=True 的格式

13.12.8 PostGIS 新增功能或增☒功能（2.4）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 2.4

- ST_AsGeobuf - 可用性：2.4.0 返回一☒行的 Geobuf 表示。
- ST_AsMVT - 可用性：2.4.0 返回一☒行的 MVT 表示形式的聚合函数。
- ST_AsMVTGeom - 可用性：2.4.0 将几何☒形☒☒☒ MVT 瓦片的坐☒空☒。
- ST_Centroid - 可用性：2.4.0 引入了☒地理的支持。返回几何体的几何中心。
- ST_ForcePolygonCCW - 可用性：2.4.0 将所有外☒逆☒☒定向，将所有内☒☒☒☒定向。
- ST_ForcePolygonCW - 可用性：2.4.0 ☒☒☒方向☒整所有外☒，逆☒☒方向☒整所有内☒。
- ST_FrechetDistance - 可用性：2.4.0 - 需要 GEOS >= 3.7.0 返回☒个几何☒形之☒的 Fréchet 距离。
- ST_IsPolygonCCW - 可用性：2.4.0 ☒☒多☒形是否具有逆☒☒方向的外☒和☒☒☒方向的内☒。
- ST_IsPolygonCW - 可用性：2.4.0 ☒☒多☒形是否具有☒☒☒外☒和逆☒☒内☒。
- ST_MakeEmptyCoverage - 可用性：2.4.0 用空☒格☒☒网格覆盖地理参考区域。

PostGIS 的功能增☒ 2.4

- Loader_Generate_Nation_Script - 增☒：2.4.1 ☒政☒☒ 5 制表区域 (zcta5) 加☒步☒已修复，☒用后，zcta5 数据将作☒名☒ zcta5_all 的☒个表加☒，作☒国家脚本加☒的一部分。☒指定平台生成加☒到☒和州☒找表中的 shell 脚本。

- Normalize_Address - 增☒：2.4.0norm_addy ☒象包括附加字段 zip4 和 address_alphanumeric。☒定文本街道地址，返回复合的 norm_addy ☒型，☒☒型具有道路后☒、前☒和☒准化☒型、街道、街道名称等，分☒☒独的字段。☒函数☒适用于与 Tiger_geocoder 打包的☒找数据（不需要 tiger 普☒数据）。
- Pagc_Normalize_Address - 增☒：2.4.0norm_addy ☒象包括附加字段 zip4 和 address_alphanumeric。☒定文本街道地址，返回复合的 norm_addy ☒型，☒☒型具有道路后☒、前☒和☒准化☒型、街道、街道名称等，分☒☒独的字段。☒函数☒适用于与 Tiger_geocoder 打包的☒找数据（不需要 tiger 普☒数据）。需要 address_standardizer ☒展。
- Reverse_Geocode - 增☒：2.4.1 如果加☒了可☒的 zcta5 数据集,即使未加☒特定的状☒数据,reverse_geocode 函数也可以解析☒状☒和 zip。有关加☒ zcta5 数据的☒☒信息，☒参☒。☒取已知空☒参考系☒中的几何点并返回一条☒☒，其中包含理☒上可能的地址数☒和交叉街道数☒。如果 include_strnum_range = true，☒包括交叉街道中的街道范☒。
- ST_AsTWKB - 增☒：2.4.0 内存和速度改☒。返回几何形式☒ TWKB，又名“微小的已知的二☒制”
- ST_Covers - 增☒：2.4.0 ☒地理☒型添加了☒多☒形中的多☒形和多☒形中的☒的支持☒☒ B 的每个点是否都位于 A 中
- ST_CurveToLine - 增☒：2.4.0 支持最大距离差公差和最大角度公差，支持☒称☒出。将包含曲☒的几何☒形☒☒☒☒性几何☒形。
- ST_Project - 增☒：2.4.0 允☒☒距离和非☒准化方位角。返回从起点按距离和方位角（方位角）投影的点。
- ST_Reverse - 增☒：2.4.0 引入了曲☒支持。返回☒点☒序相反的几何体。

PostGIS 中的功能☒生☒化 2.4

- = - 更改：2.4.0，在之前的版本中，☒是☒界框相等而不是几何相等。如果需要☒界框相等，☒使用代替。如果几何/地理 A 的坐☒和坐☒☒序与几何/地理 B 的坐☒和坐☒☒序相同，☒返回 TRUE。
- ST_Node - 更改：2.4.0 ☒函数在内部使用 GEOSNode 而不是 GEOSUnaryUnion。与 PostGIS < 2.4 相比，☒可能会☒致生成的☒串具有不同的☒序和方向。☒点是☒的集合。

13.12.9 PostGIS 新增功能或增☒功能（2.3）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 2.3

- &&&(geometry,gidx) - 可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果几何体的（☒存的）n ☒☒界框与 n ☒浮点精度☒界框 (GIDX) 相交，☒返回 TRUE。
- &&&(gidx,geometry) - 可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果 n ☒浮点精度☒界框 (GIDX) 与几何体的（☒存的）n ☒☒界框相交，☒返回 TRUE。
- &&&(gidx,gidx) - 可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果☒个 n ☒浮点精度☒界框 (GIDX) 彼此相交，☒返回 TRUE。
- &&(box2df,box2df) - 可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果☒个 2D 浮点精度☒界框 (BOX2DF) 彼此相交，☒返回 TRUE。
- &&(box2df,geometry) - 可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果 2D 浮点精度☒界框 (BOX2DF) 与几何体的（☒存的）2D ☒界框相交，☒返回 TRUE。
- &&(geometry,box2df) - 可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果几何体的（☒存的）2D ☒界框与 2D 浮点精度☒界框 (BOX2DF) 相交，☒返回 TRUE。
- @(box2df,box2df) - 可用性：2.3.0 引入了☒☒范☒索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果 2D 浮点精度☒界框 (BOX2DF) 包含在☒一个 2D 浮点精度☒界框内，☒返回 TRUE。

- @(box2df,geometry) - 可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果 2D 浮点精度⍰界框 (BOX2DF) 包含在几何体的 2D ⍰界框中，⍰返回 TRUE。
- @(geometry,box2df) - 可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果几何体的 2D ⍰界框包含在 2D 浮点精度⍰界框 (BOX2DF) 中，⍰返回 TRUE。
- Populate_Topology_Layer - 可用性：2.3.0 通⍰从拓扑表⍰取元数据，将缺失的条目添加到 topology.layer 表中。
- ST_ClusterDBSCAN - 可用性：2.3.0 使用 DBSCAN 算法返回每个⍰入几何⍰形的簇 id 的窗口函数。
- ST_ClusterKMeans - 可用性：2.3.0 使用 K 均⍰算法返回每个⍰入几何⍰形的簇 id 的窗口函数。
- ST_GeneratePoints - 可用性：2.3.0 生成一个包含在多⍰形（Polygon）或多重多⍰形（MultiPolygon）内的随机点的多点⍰象。
- ST_GeometricMedian - 可用性：2.3.0 返回多点的几何中位数。
- ST_MakeLine - 可用性：2.3.0 - 引入了⍰ MultiPoint ⍰入元素的支持从 Point, MultiPoint, 或 LineString geometries ⍰建 LineString。
- ST_MinimumBoundingRadius - 可用性-2.3.0 返回包含几何⍰形的最小⍰的中心点和半径。
- ST_MinimumClearance - 可用性：2.3.0 返回几何体的最小⍰隙，⍰是几何体⍰健性的度量。
- ST_MinimumClearanceLine - 可用性：2.3.0-需要 GEOS >= 3.6.0 返回跨越几何体最小⍰隙的⍰点⍰串。
- ST_Normalize - 可用性：2.3.0 返回⍰范形式的几何⍰形。
- ST_Points - 可用性：2.3.0 返回包含几何坐⍰的 MultiPoint。
- ST_VoronoiLines - 可用性：2.3.0 返回几何体⍰点的 Voronoi ⍰的⍰界。
- ST_VoronoiPolygons - 可用性：2.3.0 返回几何体⍰点的 Voronoi ⍰的⍰元格。
- ST_WrapX - 可用性：2.3.0 需要 GEOS 将几何体⍰⍰在 X ⍰周⍰。
- TopoGeom_addElement - 可用性：2.3 将元素添加到 TopoGeometry 的定⍰中。
- TopoGeom_remElement - 可用性：2.3 从 TopoGeometry 的定⍰中⍰除元素。
- ~(box2df,box2df) - 可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果 2D 浮点精度⍰界框 (BOX2DF) 包含⍰一个 2D 浮点精度⍰界框 (BOX2DF)，⍰返回 TRUE。
- ~(box2df,geometry) - 可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果 2D 浮点精度⍰界框 (BOX2DF) 包含几何体的 2D ⍰界框，⍰返回 TRUE。
- ~(geometry,box2df) - 可用性：2.3.0 引入了⍰⍰范⍰索引 (BRIN) 的支持。需要 PostgreSQL 9.5+。如果几何体的 2D 粘合框包含 2D 浮点精度⍰界框 (GIDX)，⍰返回 TRUE。

PostGIS 的功能增⍰ 2.3

- ST_Contains - 增⍰：2.3.0 PIP 短路（快速判断⍰限于多⍰形和点）已得到增⍰，以支持具有更少点的多点。以前的版本⍰支持面和点⍰合。⍰⍰ B 的每个点是否都位于 A 中，并且它⍰的内部是否有一个共同点
- ST_Covers - 增⍰：⍰于 2.3.0 几何⍰形，PIP 短路（⍰限于多⍰形和点的快速判断）已得到增⍰，以支持由更少点⍰成的多点。以前的版本⍰支持面和点⍰合。⍰⍰ B 的每个点是否都位于 A 中
- ST_Expand - 增⍰：2.3.0 添加了⍰不同⍰度的盒子⍰行不同数量⍰展的支持。返回从⍰一个⍰界框或几何⍰形⍰展的⍰界框。
- ST_Intersects - 增⍰：2.3.0 PIP 短路（快速判断⍰限于多⍰形和点）已得到增⍰，以支持具有更少点的多点。以前的版本⍰支持面和点⍰合。⍰⍰⍰个几何⍰形是否相交（它⍰至少有一个共同点）

- ST_Segmentize - 增☒：2.3.0 地理分段☒在可生成等☒的子分段返回修改后的几何☒形/地理，其☒段不☒于☒定距离。
- ST_Transform - 增☒：2.3.0 引入了☒直接 PROJ.4 字符串的支持。返回坐☒☒☒☒不同空☒参考系的新几何☒形。
- ST_Within - 增☒：☒于 2.3.0 几何☒形，PIP 短路（☒限于多☒形和点的快速判断）已得到增☒，以支持由更少点☒成的多点。以前的版本☒支持面和点☒合。☒☒ A 的每个点是否都位于 B 中，并且它☒的内部是否有一个共同点

PostGIS 中的功能☒生☒化 2.3

- ST_PointN - 更改：2.3.0：☒索引可用（-1 是最后一点）返回几何☒形中第一个☒串或☒形☒串中的第 N 个点。

13.12.10 PostGIS 新增功能或增☒功能（2.2）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 2.2

- <<->> - 可用性：2.2.0——KNN ☒适用于 PostgreSQL 9.1+ 返回 A 和 B 几何☒形或☒界框之☒的 n ☒距离
- ST_3DDifference - 可用性：2.2.0 ☒行 3D 差异
- ST_3DUnion - 可用性：2.2.0 ☒行 3D ☒合。
- ST_ApproximateMedialAxis - 可用性：2.2.0 ☒算几何区域的近似中☒。
- ST_AsEncodedPolyline - 可用性：2.2.0 从 LineString 几何体返回☒☒折☒。
- ST_AsTWKB - 可用性：2.2.0 返回几何形式☒ TWKB，又名“微小的已知的二☒制”
- ST_BoundingDiagonal - 可用性：2.2.0 返回几何☒界框的☒角☒。
- ST_CPAWithin - 可用性：2.2.0 ☒☒☒条☒迹的最近接近点是否在指定距离内。
- ST_ClipByBox2D - 可用性：2.2.0 ☒算几何☒形落在矩形内的部分。
- ST_ClosestPointOfApproach - 可用性：2.2.0 返回☒条☒迹最接近点☒的☒量☒。
- ST_ClusterIntersecting - 可用性：2.2.0 将☒入几何☒形聚☒成☒接集的聚合函数。
- ST_ClusterWithin - 可用性：2.2.0 按☒隔距离☒几何☒形☒行聚合的聚合函数。
- ST_CountAgg - 可用性：2.2.0 ☒☒的。返回一☒☒格的☒定波段中的像素数。如果未指定 band，☒默☒☒ band 1。如果 exclude_nodata_value ☒置☒ true，☒☒☒算不等于 NODATA ☒的像素。
- ST_CreateOverview - 可用性：2.2.0 ☒建☒定☒格覆盖范☒的降低分辨率版本。
- ST_DistanceCPA - 可用性：2.2.0 返回☒条☒迹的最近接近点之☒的距离。
- ST_ForceCurve - 可用性：2.2.0 如果适用，将一个几何☒形上升到其曲☒☒型。
- ST_IsPlanar - 可用性：2.2.0：☒在 2.1.0 中已☒☒，但在 2.1 版本中意外☒漏。☒☒表面是否平坦
- ST_IsSolid - 可用性：2.2.0 ☒☒几何体是否☒☒体。不☒行有效性☒☒。
- ST_IsValidTrajectory - 可用性：2.2.0 ☒☒几何☒形是否☒有效☒迹。
- ST_LineFromEncodedPolyline - 可用性：2.2.0 从☒☒折☒☒建 LineString。
- ST_MakeSolid - 可用性：2.2.0 将几何体☒造成☒体。不☒行任何☒☒。要☒得有效的☒体，☒入几何☒形必☒是☒合多面体曲面或☒合 TIN。

- ST_MapAlgebra (callback function version) - 可用性：2.2.0：能⊠添加遮罩回⊠函数版本 - ⊠定一个或多个⊠入⊠格、波段索引和一个用⊠指定的回⊠函数，返回⊠波段⊠格。
- ST_MemSize - 可用性：2.2.0 返回⊠格占用的空⊠量（以字⊠⊠⊠位）。
- ST_RemoveRepeatedPoints - 可用性：2.2.0 返回⊠除了重复点的几何⊠形。
- ST_Retile - 可用性：2.2.0 从任意平⊠的⊠格覆盖范⊠返回一⊠配置的平⊠。
- ST_SetEffectiveArea - 可用性：2.2.0 使用 Visvalingam-Whyatt 算法⊠置每个⊠点的有效区域。
- ST_SimplifyVW - 可用性：2.2.0 使用 Visvalingam-Whyatt 算法返回几何⊠形的⊠化表示
- ST_Subdivide - 可用性：2.2.0 ⊠算几何体的直⊠⊠分。
- ST_SummaryStatsAgg - 可用性：2.2.0 ⊠⊠的。返回一⊠⊠格的⊠定⊠格波段的摘要⊠⊠信息，其中包含⊠数、⊠和、平均⊠、⊠准差、最小⊠、最大⊠。如果未指定⊠号，⊠假定⊠⊠ 1。
- ST_SwapOrdinates - 可用性：2.2.0 返回更改后的几何⊠形，其中交⊠了⊠定的坐⊠⊠。
- ST_Volume - 可用性：2.2.0 ⊠算 3D ⊠体的体⊠。如果⊠用于表面（甚至⊠合）几何⊠形将返回 0。
- parse_address - 可用性：2.2.0 ⊠取 1 行地址并分成几部分
- postgis.enable_outdb_rasters - 可用性：2.2.0 一个布⊠配置⊠⊠，用于⊠用⊠数据⊠外⊠格波段的⊠⊠。
- postgis.gdal_datapath - 可用性：2.2.0 用于分配 GDAL 的 GDAL_DATA ⊠⊠⊠的配置⊠⊠。如果未⊠置，⊠使用⊠境⊠置的 GDAL_DATA ⊠量。
- postgis.gdal_enabled_drivers - 可用性：2.2.0 用于⊠置 PostGIS ⊠境中⊠用的 GDAL ⊠⊠程序的配置⊠⊠。影⊠ GDAL 配置⊠量 GDAL_SKIP。
- standardize_address - 可用性：2.2.0 利用 lex、gaz 和⊠⊠表返回⊠入地址的 stdaddr 形式。
- |=| - 可用性：2.2.0。索引支持⊠适用于 PostgreSQL 9.5+ 返回 A 和 B ⊠迹在最接近点⊠的距离。

PostGIS 的功能增⊠ 2.2

- <-> - 增⊠：2.2.0 -几何和地理之⊠的 KNN（k 最近⊠）行⊠⊠在是真⊠的。⊠注意，地理的 KNN 是在球面上⊠算的，而不是在⊠球体平面上⊠算的。PostgreSQL 9.4 及更低版本支持地理，但⊠支持⊠界框的重心。返回 A 和 B 之⊠的 2D 距离。
- AsTopoJSON - 增⊠：2.2.1 添加了⊠ puntal ⊠入的支持返回拓扑几何的 TopoJSON 表示形式。
- ST_Area - 增⊠：2.2.0 - 使用 GeographicLib ⊠球体⊠行⊠量，以提高准确性和⊠健性。需要 PROJ >= 4.9.0 才能利用新功能。返回多⊠形几何体的面⊠。
- ST_AsX3D - 增⊠：2.2.0：添加了⊠反⊠地理坐⊠和⊠（x/y、⊠度/⊠度）的支持。有关⊠⊠信息，⊠参⊠⊠⊠。返回 X3D xml ⊠点元素格式的几何⊠形：ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Azimuth - 增⊠：2.2.0 使用 GeographicLib ⊠球体⊠行⊠量，以提高准确性和⊠健性。需要 PROJ >= 4.9.0 才能利用新功能。返回⊠点之⊠直⊠的基于北方的方位角。
- ST_Distance - 增⊠：2.2.0 - 使用 GeographicLib ⊠球体⊠行⊠量，以提高准确性和⊠健性。需要 PROJ >= 4.9.0 才能利用新功能。返回⊠个几何或地理⊠之⊠的距离。
- ST_Scale - 增⊠：2.2.0 引入了⊠⊠放所有⊠度（factor 参数）的支持。按⊠定因子⊠放几何⊠形。
- ST_Split - 增⊠：2.2.0 引入了⊠通⊠多⊠、多点或（多）多⊠形⊠界分割⊠的支持。返回通⊠将一个几何体分割⊠⊠一个几何体而⊠建的几何体集合。
- ST_Summary - 增⊠：添加了⊠ 2.2.0 TIN 和曲⊠的支持返回几何内容的文本摘要。

PostGIS 中的功能⊠生⊠化 2.2

- <-> - 更改：2.2.0 -在 PostgreSQL 9.5 中，旧的混合格式可能很慢。因此，如果您只想在 PostGIS 2.2 或更高版本和 PostgreSQL 9.5 或更高版本上⊠行，⊠可能需要消除⊠方法。返回 A 和 B 之⊠的 2D 距离。
- Get_Geocode_Setting - 更改：2.2.0：默⊠⊠置⊠在保存在名⊠ geocode_settings_default 的表中。使用自定⊠⊠置 a 位于 geocode_settings 中，并且⊠包含用⊠⊠置的⊠置。返回存⊠在 Tiger.geocode_settings 表中的特定⊠置的⊠。
- ST_3DClosestPoint - 更改：2.2.0 - 如果⊠入 2 个 2D 几何⊠形，⊠返回 2D 点（而不是假⊠缺失 Z ⊠ 0 的旧行⊠）。在 2D 和 3D 情况下，⊠于缺失的 Z，Z 不再被假定⊠ 0。返回 g1 上最接近 g2 的 3D 点。⊠是 3D 最短⊠的第一个点。
- ST_3DDistance - 更改：2.2.0 - 在 2D 和 3D 的情况下，⊠于缺失的 Z，Z 不再被假定⊠ 0。返回⊠个几何⊠形之⊠的 3D 笛卡⊠最小距离（基于空⊠参考）（以投影⊠位表示）。
- ST_3DLongestLine - 更改：2.2.0 - 如果⊠入 2 个 2D 几何⊠形，⊠返回 2D 点（而不是假⊠缺失 Z ⊠ 0 的旧行⊠）。在 2D 和 3D 情况下，⊠于缺失的 Z，Z 不再被假定⊠ 0。返回⊠个几何体之⊠的 3D 最⊠直⊠
- ST_3DMaxDistance - 更改：2.2.0 - 在 2D 和 3D 的情况下，⊠于缺失的 Z，Z 不再被假定⊠ 0。返回⊠个几何⊠形之⊠的 3D 笛卡⊠最大距离（基于空⊠参考）（以投影⊠位表示）。
- ST_3DShortestLine - 更改：2.2.0 - 如果⊠入 2 个 2D 几何⊠形，⊠返回 2D 点（而不是假⊠缺失 Z ⊠ 0 的旧行⊠）。在 2D 和 3D 情况下，⊠于缺失的 Z，Z 不再被假定⊠ 0。返回⊠个几何⊠形之⊠的 3D 最短⊠
- ST_DistanceSphere - 更改：2.2.0 在之前的版本中，⊠曾⊠被称⊠ ST_Distance_Sphere 使用球形地球模型返回⊠个⊠度/⊠度几何形状之⊠的最小距离（以米⊠⊠位）。
- ST_DistanceSpheroid - 更改：2.2.0 在之前的版本中，⊠称⊠ ST_Distance_Spheroid 使用球体模型返回⊠个⊠度/⊠度几何形状之⊠的最小距离。
- ST_Equals - 更改：2.2.0 即使⊠于无效几何⊠形，如果它⊠二⊠制相等，也会返回 true ⊠⊠⊠个几何⊠形是否包含同一⊠点
- ST_LengthSpheroid - 更改：2.2.0 在之前的版本中,⊠称⊠ ST_Length_Spheroid 并具有⊠名 ST_3DLength_Sphe返回⊠球体上⊠度/⊠度几何体的 2D 或 3D ⊠度/周⊠。
- ST_MemSize - 更改：2.2.0 名称更改⊠ ST_MemSize 以遵循命名⊠定。返回几何⊠形占用的内存空⊠⊠量。
- ST_PointInsideCircle - 更改：2.2.0 在之前的版本中，⊠称⊠ ST_Point_Inside_Circle ⊠⊠点几何⊠形是否位于由⊠心和半径定⊠的⊠内
- ValidateTopology - 更改：2.2.0 id1 和 id2 的⊠被交⊠⊠ “⊠⊠交叉⊠点”，以与⊠⊠描述一致。返回一⊠ validatetopology_returntype ⊠象，⊠⊠⊠明拓扑⊠⊠。

13.12.11 PostGIS 新增功能或增⊠功能（2.1）

下面⊠出的功能是添加或增⊠的 PostGIS 功能。

PostGIS 中的新功能 2.1

- = - 可用性：2.1.0 如果 A 的⊠界框与 B 的⊠界框相同，⊠返回 TRUE。使用双精度⊠界框。
- AsTopoJSON - 可用性：2.1.0 返回拓扑几何的 TopoJSON 表示形式。
- Drop_Nation_Tables_Generate_Script - 可用性：2.1.0 生成一个脚本，⊠除指定架⊠中以 County_all、state_all 或 state 代⊠开⊠，后跟 county 或 state 的所有表。
- Get_Geocode_Setting - 可用性：2.1.0 返回存⊠在 Tiger.geocode_settings 表中的特定⊠置的⊠。
- Loader_Generate_Nation_Script - 可用性：2.1.0 ⊠指定平台生成加⊠到⊠和州⊠找表中的 shell 脚本。
- Pagc_Normalize_Address - 可用性：2.1.0 ⊠定文本街道地址，返回复合的 norm_addy ⊠型，⊠⊠型具有道路后⊠、前⊠和⊠准化⊠型、街道、街道名称等，分⊠⊠独的字段。⊠函数⊠适用于与 Tiger_geocoder 打包的⊠找数据（不需要 tiger 普⊠数据）。需要 address_standardizer ⊠展。

- ST_3DArea - 可用性：2.1.0 ☒算 3D 表面几何形状的面☒。☒于固体将返回 0。
- ST_3DIntersection - 可用性：2.1.0 ☒行 3D 相交
- ST_Box2dFromGeoHash - 可用性：2.1.0 从 GeoHash 字符串返回 BOX2D。
- ST_ColorMap - 可用性：2.1.0 根据源☒格和指定波段☒建最多四个 8BUI 波段（灰度、RGB、RGBA）的新☒格。如果未指定，☒假定☒波段 1。
- ST_Contains - 可用性：2.1.0 如果☒格 rastB 中没有点位于☒格 rastA 的外部且 rastB 的内部至少有一个点位于 rastA 的内部，☒返回 true。
- ST_ContainsProperly - 可用性：2.1.0 如果 rastB 与 rastA 的内部相交，但不与 rastA 的☒界或外部相交，☒返回 true。
- ST_CoveredBy - 可用性：2.1.0 如果☒格 rastA 中没有点位于☒格 rastB 之外，☒返回 true。
- ST_Covers - 可用性：2.1.0 如果☒格 rastB 中没有点位于☒格 rastA 之外，☒返回 true。
- ST_DFullyWithin - 可用性：2.1.0 如果☒格 rastA 和 rastB 彼此完全在指定距离内，☒返回 true。
- ST_DWithin - 可用性：2.1.0 如果☒格 rastA 和 rastB 彼此之☒的距离在指定距离内，☒返回 true。
- ST_DelaunayTriangles - 可用性：2.1.0 返回几何体☒点的 Delaunay 三角剖分。
- ST_Disjoint - 可用性：2.1.0 如果☒格 rastA 在空☒上不与 rastB 相交，☒返回 true。
- ST_DumpValues - 可用性：2.1.0 ☒取指定 band（波段）的☒作☒二☒数☒。
- ST_Extrude - 可用性：2.1.0 将曲面☒出到相关体☒
- ST_ForceLHR - 可用性：2.1.0 ☒制 LHR 方向
- ST_FromGDALRaster - 可用性：2.1.0 从受支持的 GDAL ☒格文件返回☒格。
- ST_GeomFromGeoHash - 可用性：2.1.0 从 GeoHash 字符串返回几何☒形。
- ST_InvDistWeight4ma - 可用性：2.1.0 从像素的☒域内插像素☒的☒格☒理函数。
- ST_MapAlgebra (callback function version) - 可用性：2.1.0 回☒函数版本 - ☒定一个或多个☒入☒格、波段索引和一个用☒指定的回☒函数，返回☒波段☒格。
- ST_MapAlgebra (expression version) - 可用性：2.1.0 表☒式版本 - 在☒定一☒个☒入☒格、波段索引和一个或多个用☒指定的 SQL 表☒式的情况下，返回☒波段☒格。
- ST_MinConvexHull - 可用性：2.1.0 返回☒格的凸包几何形状（不包括 NODATA 像素）。
- ST_MinDist4ma - 可用性：2.1.0 返回感☒趣像素与具有☒的相☒像素之☒的最小距离（以像素数☒☒位）的☒格☒理函数。
- ST_MinkowskiSum - 可用性：2.1.0 ☒行 Minkowski sum
- ST_NearestValue - 可用性：2.1.0 返回由 columnx 和 rowy 指定的☒定☒像素的最接近的非 NODATA ☒或以与☒格相同的空☒参考坐☒系表示的几何点。
- ST_Neighborhood - 可用性：2.1.0 返回☒定波段像素周☒非 NODATA ☒的二☒双精度数☒，☒☒定波段像素由 columnX 和 rowY 或以与☒格相同的空☒参考坐☒系表示的几何点指定。
- ST_NotSameAlignmentReason - 可用性：2.1.0 返回☒明☒格是否☒☒的文本，如果未☒☒，☒☒明原因。
- ST_Orientation - 可用性：2.1.0 确定表面方向
- ST_Overlaps - 可用性：2.1.0 如果☒格 rastA 和 rastB 相交，但其中一个不完全包含☒一个，☒返回 true。
- ST_PixelAsCentroid - 可用性：2.1.0 返回像素表示的区域的☒心（点几何）。

- ST_PixelAsCentroids - 可用性：2.1.0 返回⊠格波段的每个像素的⊠心（点几何）以及每个像素的⊠、X 和 Y ⊠格坐⊠。点几何是像素表示的区域的⊠心。
- ST_PixelAsPoint - 可用性：2.1.0 返回像素左上角的点几何形状。
- ST_PixelAsPoints - 可用性：2.1.0 返回⊠格波段的每个像素的点几何⊠形以及每个像素的⊠、X 和 Y ⊠格坐⊠。点几何的坐⊠是像素的左上角。
- ST_PixelOfValue - 可用性：2.1.0 ⊠取⊠等于搜索⊠的像素的列 x、行 y 坐⊠。
- ST_PointFromGeoHash - 可用性：2.1.0 从 GeoHash 字符串返回一个点。
- ST_RasterToWorldCoord - 可用性：2.1.0 在⊠定列和行的情况下，以几何 X 和 Y（⊠度和⊠度）形式返回⊠格的左上角。列和行从 1 开始。
- ST_Resize - 可用性：2.1.0 需要 GDAL 1.6.1+ 将⊠格大小⊠整⊠新的⊠度/高度
- ST_Roughness - 可用性：2.1.0 返回一个⊠算出的数字高程模型（DEM）的' 粗糙度' 的⊠格。
- ST_SetValues - 可用性：2.1.0 返回通⊠⊠置⊠定波段的⊠而⊠生的修改后的⊠格。
- ST_Simplify - 可用性：2.1.0 使用 Douglas-Peucker 算法返回⊠定 TopoGeometry 的“⊠化”几何版本。
- ST_StraightSkeleton - 可用性：2.1.0 从几何体⊠算直骨架
- ST_Summary - 可用性：2.1.0 返回⊠格内容的文本摘要。
- ST_TPI - 可用性：2.1.0 返回一个⊠算出的地形位置指数（Topographic Position Index）的⊠格。
- ST_TRI - 可用性：2.1.0 返回具有⊠算的地形⊠固性指数的⊠格。
- ST_Tesselate - 可用性：2.1.0 ⊠多⊠形或多面体表面⊠行曲面⊠分，并以 TIN 或 TINS 集合的形式返回
- ST_Tile - 可用性：2.1.0 返回根据⊠出⊠格的所需⊠度分割⊠入⊠格而⊠生的一⊠⊠格。
- ST_Touches - 可用性：2.1.0 如果⊠格 rastA 和 rastB 至少有一个共同点但它⊠的内部不相交，⊠返回 true。
- ST_Union - 可用性：2.1.0 引入了 ST_Union(rast, unionarg) ⊠体。将一⊠⊠格切片的并集返回⊠由 1 个或多个波段⊠成的⊠个⊠格。
- ST_Within - 可用性：2.1.0 如果⊠格 rastA 中没有点位于⊠格 rastB 的外部且 rastA 的内部至少有一个点位于 rastB 的内部，⊠返回 true。
- ST_WorldToRasterCoord - 可用性：2.1.0 ⊠定几何 X 和 Y（⊠度和⊠度）或以⊠格的空⊠参考坐⊠系表示的点几何，将左上角作⊠列和行返回。
- Set_Geocode_Setting - 可用性：2.1.0 ⊠置影⊠地理⊠⊠器功能行⊠的⊠置。
- UpdateRasterSRID - 可用性：2.1.0 更改用⊠指定的列和表中所有⊠格的 SRID。
- clearTopoGeom - 可用性：2.1 清除拓扑几何的内容。
- postgis_sfcgal_version - 可用性：2.1.0 返回正在使用的 SFCGAL 版本

PostGIS 的功能增⊠ 2.1

- ST_AddBand - 增⊠：2.1.0 添加了⊠ addbandarg 的支持。返回一个⊠格，其中在⊠定索引位置添加了⊠定⊠型的新波段和⊠定初始⊠。如果未指定索引，⊠将⊠添加到末尾。
- ST_AddBand - 增⊠：2.1.0 添加了⊠新的 out-db 波段的支持。返回一个⊠格，其中在⊠定索引位置添加了⊠定⊠型的新波段和⊠定初始⊠。如果未指定索引，⊠将⊠添加到末尾。
- ST_AsBinary/ST_AsWKB - 增⊠：2.1.0 添加 outasin 返回⊠格的熟知的二⊠制 (WKB) 表示形式。
- ST_AsGML - 增⊠：⊠⊠ GML 3 引入了 2.1.0 id 支持。将几何⊠形作⊠ GML 版本 2 或 3 元素返回。

- ST_Aspect - 增☒：2.1.0 使用 ST_MapAlgebra() 并添加可☒的 interpolate_nodata 函数参数返回高程☒格波段的坡向（默☒以度☒☒位）。☒于分析地形很有用。
- ST_Boundary - 增☒：引入了 2.1.0 三角函数支持返回几何☒形的☒界。
- ST_Clip - 增☒：2.1.0 用 C 重写 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 - 增☒：2.1.0 提高了地理速度。有关☒☒信息，☒参☒使地理更快。☒☒☒个几何☒形是否在☒定距离内
- ST_DWithin - 增☒：2.1.0 引入了☒弯曲几何形状的支持。☒☒☒个几何☒形是否在☒定距离内
- ST_Distance - 增☒：2.1.0 提高了地理速度。有关☒☒信息，☒参☒使地理更快。返回☒个几何或地理☒之☒的距离。
- ST_Distance - 增☒：2.1.0 - 引入了☒弯曲几何形状的支持。返回☒个几何或地理☒之☒的距离。
- ST_Distinct4ma - 增☒：2.1.0 添加格式 2 ☒格☒理函数，用于☒算☒域中唯一像素☒的数量。
- ST_DumpPoints - 增☒：2.1.0 速度更快。重新☒☒☒原生 C ☒言。返回几何☒形中坐☒的一☒ geometry_dump 行。
- ST_HillShade - 增☒：2.1.0 使用 ST_MapAlgebra() 并添加可☒的 interpolate_nodata 函数参数使用提供的方位角、高度、亮度和比例☒入返回高程☒格☒的假☒照明。
- ST_MakeValid - 增☒：2.1.0 添加了☒几何集合和多点的支持。☒☒在不☒失☒点的情况下使无效几何体有效。
- ST_Max4ma - 增☒：2.1.0 添加格式 2 ☒算☒域中最大像素☒的☒格☒理函数。
- ST_Mean4ma - 增☒：2.1.0 添加格式 2 ☒算☒域中平均像素☒的☒格☒理函数。
- ST_Min4ma - 增☒：2.1.0 添加格式 2 ☒算☒域中最小像素☒的☒格☒理函数。
- ST_PixelAsPolygons - 增☒：2.1.0 添加了 except_nodata_value 可☒参数。返回包☒☒格☒的每个像素的多☒形几何☒形以及每个像素的☒、X 和 Y ☒格坐☒。
- ST_Polygon - 增☒：2.1.0 提高速度（完全基于 C）并且确保返回的多☒形有效。返回由具有非无数据☒的像素☒的像素并集形成的多多☒形几何体。如果未指定波段☒号，☒波段☒号默☒☒ 1。
- ST_Range4ma - 增☒：2.1.0 添加格式 2 ☒算☒域中像素☒范☒的☒格☒理函数。
- ST_SameAlignment - 增☒：2.1.0 添加聚合☒体如果☒格具有相同的☒斜、比例、空☒参考和偏移（像素可以放在同一网格上而不切割成像素），☒返回 true；如果没有注意☒☒☒☒，☒返回 false。
- ST_Segmentize - 增☒：2.1.0 引入了☒地理的支持。返回修改后的几何☒形/地理，其☒段不☒于☒定距离。
- ST_SetGeoReference - 增☒：2.1.0 添加 ST_SetGeoReference(raster, double precision, ...) ☒体在一次☒用中☒置 Georeference 6 地理配准参数。数字☒☒用空格分隔。接受 GDAL 或 ESRI 格式的☒入。默☒☒ GDAL。
- ST_SetValue - 增☒：2.1.0 ST_SetValue() 的几何☒体☒在支持任何几何☒型，而不☒☒是点。几何☒体是 ST_SetValues() 的 geomval[] ☒体的包装返回修改后的☒格，其☒果是将☒定波段中的☒☒置☒指定列 x、行 y 像素或与特定几何☒形相交的像素。波段☒号从 1 开始，如果未指定波段，☒默☒☒ 1。
- ST_Slope - 增☒：2.1.0 使用 ST_MapAlgebra() 并添加可☒ units、scale、interpolate_nodata 函数参数返回高程☒格☒的坡度（默☒以度☒☒位）。☒于分析地形很有用。
- ST_StdDev4ma - 增☒：2.1.0 添加格式 2 ☒算☒域中像素☒的☒准偏差的☒格☒理函数。
- ST_Sum4ma - 增☒：2.1.0 添加格式 2 ☒格☒理函数，☒算☒域中所有像素☒的☒和。
- ST_Summary - 增☒：添加了 S ☒志以指示其是否具有 2.1.0 空☒参考系☒返回几何内容的文本摘要。

- ST_Transform - 增☒：2.1.0 添加 ST_Transform(rast,alignto) ☒体使用指定的重采☒算法将已知空☒参考系☒中的☒格重新投影到☒一个已知空☒参考系☒。☒☒有 NearestNeighbor、Bilinear、Cubic、CubicSpline、Lanczos（默☒☒ NearestNeighbor）。
- ST_Union - 增☒：2.1.0 提高速度（完全基于 C）。将一☒☒格切片的并集返回☒由 1 个或多个波段☒成的☒个☒格。
- ST_Union - 增☒：2.1.0 ST_Union(rast)（☒体 1）合并所有☒入☒格的所有波段。PostGIS 的早期版本采用第一个波段。将一☒☒格切片的并集返回☒由 1 个或多个波段☒成的☒个☒格。
- ST_Union - 增☒：2.1.0 ST_Union(rast, uniontype)（☒体 4）合并所有☒入☒格的所有波段。将一☒☒格切片的并集返回☒由 1 个或多个波段☒成的☒个☒格。
- toTopoGeom - 增☒：2.1.0 添加了采用☒有 TopoGeometry 的版本。将☒☒几何☒形☒☒☒拓扑几何☒形。

PostGIS 中的功能☒生☒化 2.1

- ST_Aspect - 更改：2.1.0 在之前的版本中，返回☒以弧度☒☒位。☒在，返回☒默☒☒度数返回高程☒格波段的坡向（默☒以度☒☒位）。☒于分析地形很有用。
- ST_EstimatedExtent - 更改：2.1.0。在 2.0.x 之前，☒称☒ ST_Estimated_Extent。返回空☒表的估☒范☒。
- ST_Force2D - 更改：2.1.0 在 2.0.x 期☒，它被称☒ ST_Force_2D。☒制几何☒形☒入“二☒模式”。
- ST_Force3D - 更改：2.1.0 在 2.0.x 期☒，它被称☒ ST_Force_3D。☒制几何☒形☒入 XYZ 模式。☒是 ST_Force3DZ 的☒名。
- ST_Force3DM - 更改：2.1.0 。在 2.0.x 期☒，它被称☒ ST_Force_3DM。☒制几何☒形☒入 XYM 模式。
- ST_Force3DZ - 更改：2.1.0。在 2.0.x 期☒，它被称☒ ST_Force_3DZ。☒制几何☒形☒入 XYZ 模式。
- ST_Force4D - 更改：2.1.0。在 2.0.x 期☒，它被称☒ ST_Force_4D。☒制几何☒形☒入 XYZM 模式。
- ST_ForceCollection - 更改：2.1.0 。在 2.0.x 期☒，它被称☒ ST_Force_Collection。将几何☒形☒☒☒几何集合（GEOMETRYCOLLECTION）。
- ST_HillShade - 更改：2.1.0 在之前的版本中，方位角和高度以弧度表示。☒在，方位角和高度以度表示使用提供的方位角、高度、亮度和比例☒入返回高程☒格☒的假☒照明。
- ST_LineInterpolatePoint - 更改：2.1.0。在 2.0.x 之前，☒称☒ ST_Line_Interpolate_Point。返回沿☒在百分比指示位置的插☒点。
- ST_LineLocatePoint - 更改：2.1.0。在 2.0.x 之前，☒称☒ ST_Line_Locate_Point。返回☒上最接近点的分数位置。
- ST_LineSubstring - 更改：2.1.0。在 2.0.x 之前，☒被称☒ ST_Line_Substring。返回☒个小数位置之☒的直☒部分。
- ST_PixelAsCentroids - 更改：2.1.1 更改了 except_nodata_value 的行☒。返回☒格波段的每个像素的☒心（点几何）以及每个像素的☒、X 和 Y ☒格坐☒。点几何是像素表示的区域的☒心。
- ST_PixelAsPoints - 更改：2.1.1 更改了 except_nodata_value 的行☒。返回☒格波段的每个像素的点几何☒形以及每个像素的☒、X 和 Y ☒格坐☒。点几何的坐☒是像素的左上角。
- ST_PixelAsPolygons - 更改：2.1.1 更改了 except_nodata_value 的行☒。返回包☒☒格☒的每个像素的多☒形几何☒形以及每个像素的☒、X 和 Y ☒格坐☒。
- ST_Polygon - 更改：2.1.0 在之前的版本中有☒会返回多☒形，更改☒始☒返回多多☒形。返回由具有非无数据☒的像素☒的像素并集形成的多多☒形几何体。如果未指定波段☒号，☒波段☒号默☒☒ 1。
- ST_RasterToWorldCoordX - 更改：2.1.0 在之前的版本中，☒称☒ ST_Raster2WorldCoordX 返回☒格、列和行左上角的几何 X 坐☒。列和行的☒号从 1 开始。

- ST_RasterToWorldCoordY - 更改：2.1.0 在之前的版本中，☒称☒ ST_Raster2WorldCoordY 返回☒格、列和行的左上角的几何 Y 坐☒。列和行的☒号从 1 开始。
- ST_Rescale - 更改：2.1.0 适用于没有 SRID 的☒格通☒☒☒整☒格的比例（或像素大小）来重新采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear、Cubic、CubicSpline、Lanczos、Max 或 Min 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_Reskew - 更改：2.1.0 适用于没有 SRID 的☒格通☒☒☒整☒格的☒斜（或旋☒参数）来重采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear、Cubic、CubicSpline 或 Lanczos 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_Segmentize - 更改：2.1.0 由于引入了地理支持，使用 ST_Segmentize('LINESTRING(1 2, 3 4)', 0.5) 会☒致不明确的函数☒☒。☒入需要正确☒入☒几何或地理。使用 ST_GeomFromText、ST_GeogFromText 或☒☒☒所需☒型（例如 ST_Segmentize('LINESTRING(1 2, 3 4)'::geometry, 0.5) ））返回修改后的几何☒形/地理，其☒段不☒于☒定距离。
- ST_Slope - 更改：2.1.0 在之前的版本中，返回☒以弧度☒☒位。☒在，返回☒默☒☒度数返回高程☒格☒的坡度（默☒以度☒☒位）。☒于分析地形很有用。
- ST_SnapToGrid - 更改：2.1.0 适用于没有 SRID 的☒格通☒将☒格捕捉到网格来重采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear, Cubic, CubicSpline 或 Lanczos 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_WorldToRasterCoordX - 更改：2.1.0 在之前的版本中，☒称☒ ST_World2RasterCoordX 返回☒格中点几何 (pt) 的列或以☒格世界空☒参考系表示的 X 和 Y 世界坐☒ (xw, yw)。
- ST_WorldToRasterCoordY - 更改：2.1.0 在之前的版本中，☒称☒ ST_World2RasterCoordY 返回点几何☒形 (pt) 的☒格中的行或以☒格的世界空☒参考系表示的 X 和 Y 世界坐☒ (xw, yw)。

13.12.12 PostGIS 新增功能或增☒功能（2.0）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 2.0

- && - 可用性：2.0.0 如果 A 的☒界框与 B 的☒界框相交，☒返回 TRUE。
- &&& - 可用性：2.0.0 如果 A 的 n ☒☒界框与 B 的 n ☒☒界框相交，☒返回 TRUE。
- <#> - 可用性：2.0.0——KNN ☒适用于 PostgreSQL 9.1+ 返回 A 和 B ☒界框之☒的 2D 距离。
- <-> - 可用性：2.0.0——弱 KNN 根据几何☒心距离而不是真☒距离提供最近☒。点的☒果精确，所有其他☒型的☒果不精确。适用于 PostgreSQL 9.1+ 返回 A 和 B 之☒的 2D 距离。
- @ - 可用性：2.0.0 raster@raster、raster@geometry 引入如果 A 的☒界框包含在 B 的☒界框中，☒返回 TRUE。使用双精度☒界框。
- @ - 可用性：2.0.5 geometry @ raster 引入如果 A 的☒界框包含在 B 的☒界框中，☒返回 TRUE。使用双精度☒界框。
- AddEdge - 可用性：2.0.0 使用指定的☒串几何将☒串☒添加到☒表，并将关☒的起点和☒点添加到指定拓扑方案的点☒点表，并返回新（或☒有）☒的 edgeid。
- AddFace - 可用性：2.0.0 将面基元注册到拓扑并☒取其☒☒符。
- AddNode - 可用性：2.0.0 将点☒点添加到指定拓扑模式的☒点表中，并返回新☒点的 nodeid。如果点已☒作☒☒点存在，☒返回☒有的 nodeid。
- AddOverviewConstraints - 可用性：2.0.0 将☒格列☒☒☒☒一☒格列的概述。
- AddRasterConstraints - 可用性：2.0.0 将☒格☒束添加到已加☒的☒格表中，用于特定列，☒列☒束了空☒参考、比例、☒大小、☒☒、波段、波段☒型以及一个☒志，用于表示☒格列是否被☒☒地分☒。表必☒加☒数据才能推断出☒束。如果☒束☒置成功，☒返回 true，否☒会☒出通知。

- AsGML - 可用性: 2.0.0 返回拓扑几何的 GML 表示形式。
- CopyTopology - 可用性: 2.0.0 将拓扑（☒点、☒、面、☒和拓扑几何）的副本复制到新模式中
- DropOverviewConstraints - 可用性: 2.0.0 取消☒格列作☒☒一☒格列概☒的☒☒。
- DropRasterConstraints - 可用性: 2.0.0 ☒除引用☒格表列的 PostGIS ☒格☒束。如果您需要重新加☒数据或更新☒格列数据，☒非常有用。
- Drop_Indexes_Generate_Script - 可用性: 2.0.0 生成一个脚本，☒除 Tiger 架☒和用☒指定架☒上的所有非主☒和非唯一索引。如果未指定架☒，☒默☒架☒☒ Tiger_data。
- Drop_State_Tables_Generate_Script - 可用性: 2.0.0 生成一个脚本，☒脚本☒除指定架☒中以 state ☒写☒前☒的所有表。如果未指定架☒，☒默☒架☒☒ Tiger_data。
- Geocode_Intersection - 可用性: 2.0.0 接收 2 条相交的街道以及 state、city、zip，并☒出位于交叉路口的第一个交叉街道上的一☒可能位置，☒包括一个 geomout 作☒ NAD 83 ☒☒度中的点位置，一个☒准化地址 (addy) 每个位置以及☒☒。☒☒越低，匹配的可能性越大。☒果首先按最低☒分排序。可以☒☒☒入最大☒果，默☒☒ 10。使用 Tiger 数据（edges、faces、addr）、PostgreSQL 模糊字符串匹配（soundex、levenshtein）。
- GetEdgeByPoint - 可用性: 2.0.0 ☒找与☒定点相交的☒的☒ ID。
- GetFaceByPoint - 可用性: 2.0.0 ☒找与☒定点相交的面。
- GetNodeByPoint - 可用性: 2.0.0 ☒找某个点位置☒☒点的☒点 ID。
- GetNodeEdges - 可用性 : 2.0 返回与☒定☒点相关的一☒有序☒。
- GetRingEdges - 可用性: 2.0.0 返回按☒序排列的☒☒☒符集合，☒些☒☒符是通☒沿☒定☒的一☒行走☒遇到的。
- GetTopoGeomElements - 可用性: 2.0.0 返回一☒ topoelement ☒象，其中包含☒定 TopoGeometry（原始元素）的拓扑 element_id、element_type。
- GetTopologySRID - 可用性: 2.0.0 返回拓扑表中☒定拓扑名称的拓扑的 SRID。
- Get_Tract - 可用性: 2.0.0 从几何☒形所在的区域表中返回人口普☒区域或字段。默☒返回区域的短名称。
- Install_Missing_Indexes - 可用性: 2.0.0 ☒找具有在地理☒☒器☒接和☒☒条件中使用的☒列的所有表，☒些表缺少☒些列上使用的索引，并将添加它☒。
- Loader_Generate_Census_Script - 可用性: 2.0.0 ☒指定 states 的指定平台生成 shell 脚本，☒脚本将下☒ Tiger 人口普☒ state 区、bg 和 tabblocks 数据表、☒存并加☒到 Tiger_data 架☒中。每个状☒脚本都作☒☒独的☒☒返回。
- Loader_Generate_Script - 可用性 : 2.0.0 支持 Tiger 2010 ☒☒化数据并加☒人口普☒区（tract）、区☒☒（bg）和区☒（tabblocks）表。☒指定平台的指定状☒生成 shell 脚本，☒脚本将下☒ Tiger 数据、☒存并加☒到 Tiger_data 模式中。每个状☒脚本都作☒☒独的☒☒返回。最新版本支持 Tiger 2010 ☒☒☒化，☒加☒人口普☒区、区☒☒和区☒表。
- Missing_Indexes_Generate_Script - 可用性: 2.0.0 ☒找具有在地理☒☒器☒接中使用的☒列的所有表，☒些表缺少☒些列上的索引，并将☒出 SQL DDL 来定☒☒些表的索引。
- Polygonize - 可用性: 2.0.0 ☒找并注册由拓扑☒定☒的所有面。
- Reverse_Geocode - 可用性: 2.0.0 ☒取已知空☒参考系☒中的几何点并返回一条☒☒，其中包含理☒上可能的地址数☒和交叉街道数☒。如果 include_strnum_range = true，☒包括交叉街道中的街道范☒。
- ST_3DClosestPoint - 可用性: 2.0.0 返回 g1 上最接近 g2 的 3D 点。☒是 3D 最短☒的第一个点。
- ST_3DDFullyWithin - 可用性: 2.0.0 ☒☒☒个 3D 几何☒形是否完全在☒定的 3D 距离内
- ST_3DDWithin - 可用性: 2.0.0 ☒☒☒个 3D 几何☒形是否在☒定的 3D 距离内
- ST_3DDistance - 可用性: 2.0.0 返回☒个几何☒形之☒的 3D 笛卡☒最小距离（基于空☒参考）（以投影☒位表示）。

- ST_3DIntersects - 可用性: 2.0.0 ☒☒☒个几何☒形在 3D 空☒中是否相交 - ☒适用于点、☒串、多☒形、多面体曲面（区域）
- ST_3DLongestLine - 可用性: 2.0.0 返回☒个几何体之☒的 3D 最☒直☒
- ST_3DMaxDistance - 可用性: 2.0.0 返回☒个几何☒形之☒的 3D 笛卡☒最大距离（基于空☒参考）（以投影☒位表示）。
- ST_3DShortestLine - 可用性: 2.0.0 返回☒个几何☒形之☒的 3D 最短☒
- ST_AddEdgeModFace - 可用性：2.0 添加新☒，如果☒☒做会分割面，☒修改原始面并添加新面。
- ST_AddEdgeNewFaces - 可用性：2.0 添加新☒，如果☒☒做会分割一个面，☒☒除原始面并用☒个新面替☒它。
- ST_AsGDALRaster - 可用性：2.0.0 - 需要 GDAL >= 1.6.0。以指定的 GDAL ☒格格式返回☒格☒☒。☒格格式是☒☒☒支持的格式之一。使用 ST_GDALDrivers() ☒取您的☒支持的格式列表。
- ST_AsJPEG - 可用性：2.0.0 - 需要 GDAL >= 1.6.0。将☒格☒☒☒定的波段作☒☒个☒合☒影☒出☒ (JPEG) ☒像（字☒数☒）返回。如果未指定波段且有 1 个或 3 个以上波段，☒☒使用第一个波段。如果只有 3 个波段，☒使用所有 3 个波段并将其映射到 RGB。
- ST_AsLatLonText - 可用性：2.0 返回☒定点的度、分、秒表示形式。
- ST_AsPNG - 可用性：2.0.0 - 需要 GDAL >= 1.6.0。将☒格☒☒☒定的波段作☒☒个便携式网☒☒形 (PNG) ☒像（字☒数☒）返回。如果☒格中有 1、3 或 4 个波段且未指定波段，☒使用所有波段。如果有 2 个以上或多于 4 个波段且未指定波段，☒☒使用波段 1。波段映射到 RGB 或 RGBA 空☒。
- ST_AsRaster - 可用性：2.0.0 - 需要 GDAL >= 1.6.0。将 PostGIS 几何☒形☒☒☒ PostGIS ☒格。
- ST_AsTIFF - 可用性：2.0.0 - 需要 GDAL >= 1.6.0。将☒格☒定的波段作☒☒个 TIFF ☒像（字☒数☒）返回。如果未指定波段或☒格中不存在任何指定波段，☒将☒☒使用所有波段。
- ST_AsX3D - 可用性：2.0.0：ISO-IEC-19776-1.2-X3DEncodings-XML 返回 X3D xml ☒点元素格式的几何☒形：ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_Aspect - 可用性: 2.0.0 返回高程☒格波段的坡向（默☒以度☒☒位）。☒于分析地形很有用。
- ST_Band - 可用性: 2.0.0 返回☒有☒格的一个或多个波段作☒新☒格。☒于从☒有☒格☒建新☒格非常有用。
- ST_BandIsNoData - 可用性: 2.0.0 如果波段☒填充无数据☒，☒返回 true。
- ST_Clip - 可用性: 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 - 可用性: 2.0.0 返回几何集合的最☒☒表示。
- ST_ConcaveHull - 可用性: 2.0.0 ☒算包含所有☒入几何☒点的可能凹几何
- ST_Count - 可用性: 2.0.0 返回☒格或☒格覆盖范☒的☒定波段中的像素数。如果未指定 band，☒默☒☒ band 1。如果 exclude_nodata_value ☒置☒ true，☒☒☒算不等于 nodata ☒的像素。
- ST_CreateTopoGeo - 可用性：2.0 将几何☒形集合添加到☒定的空拓扑并返回☒☒☒明成功的消息。
- ST_Distinct4ma - 可用性: 2.0.0 ☒格☒理函数，用于☒算☒域中唯一像素☒的数量。
- ST_FlipCoordinates - 可用性: 2.0.0 返回 X ☒和 Y ☒☒☒的几何☒形版本。
- ST_GDALDrivers - 可用性：2.0.0 - 需要 GDAL >= 1.6.0。通☒ GDAL 返回 PostGIS 支持的☒格格式列表。ST_AsGDALRaster ☒可使用 can_write=True 的格式
- ST_GeomFromGeoJSON - 可用性：2.0.0 需要 JSON-C 0.9 或更高版本将几何☒形的 geojson 表示形式作☒☒入并☒出 PostGIS 几何☒象

- ST_GetFaceEdges - 可用性：2.0 返回一☒有序的☒，☒些☒界☒定了 aface。
- ST_HasNoBand - 可用性：2.0.0 如果不存在具有☒定波段☒号的波段，☒返回 true。如果未指定波段号，☒假定波段号☒ 1。
- ST_HillShade - 可用性：2.0.0 使用提供的方位角、高度、亮度和比例☒入返回高程☒格☒的假☒照明。
- ST_Histogram - 可用性：2.0.0 返回一☒☒☒，☒☒了☒格或☒格覆盖数据分布，其中包括分离的分箱范☒。如果未指定，将自☒☒算分箱数。
- ST_InterpolatePoint - 可用性：2.0.0 返回最接近点的几何☒形的插☒☒量☒。
- ST_IsEmpty - 可用性：2.0.0 如果☒格☒空（☒度 = 0 且高度 = 0），☒返回 true。否☒，返回 false。
- ST_IsValidDetail - 可用性：2.0.0 返回 valid_detail 行，☒明几何☒形是否有效或无效，☒明原因和位置。
- ST_IsValidReason - 有效性：2.0 版本正在接受☒☒。返回☒明几何☒形是否有效或无效原因的文本。
- ST_MakeLine - 可用性：2.0.0 - 引入了☒ LineString ☒入元素的支持从 Point, MultiPoint, 或 LineString geometries ☒建 LineString。
- ST_MakeValid - 可用性：2.0.0 ☒☒在不☒失☒点的情况下使无效几何体有效。
- ST_MapAlgebraExpr - 可用性：2.0.0 1 ☒格波段版本：通☒☒☒入☒格波段和提供的像素☒型☒用有效的 PostgreSQL 代数☒算来☒建新的☒波段☒格。如果未指定波段，☒假定☒波段 1。
- ST_MapAlgebraExpr - 可用性：2.0.0 2 ☒格波段版本：通☒☒提供的☒个☒入☒格波段和像素☒型☒用有效的 PostgreSQL 代数☒算来☒建新的☒波段☒格。如果未指定波段☒号，☒假定每个☒格的波段 1。生成的☒格将在第一个☒格定☒的网格上☒☒（比例、☒斜和像素角），并具有由“extenttype”参数定☒的范☒。“extenttype”的☒可以是：INTERSECTION、UNION、FIRST、SECOND。
- ST_MapAlgebraFct - 可用性：2.0.0 1 波段版本 - 通☒在☒入☒格波段和提供的像素☒型上☒用有效的 PostgreSQL 函数来☒建新的☒波段☒格。如果未指定波段，☒假定☒波段 1。
- ST_MapAlgebraFct - 可用性：2.0.0 2 波段版本 - 通☒在提供的 2 个☒入☒格波段和像素☒型上☒用有效的 PostgreSQL 函数来☒建新的☒波段☒格。如果未指定波段，☒假定☒波段 1。如果未指定，范☒☒型默☒☒ INTERSECTION。
- ST_MapAlgebraFctNgb - 可用性：2.0.0 1-波段版本：使用用☒定☒的 PostgreSQL 函数映射代数最近☒。返回一个☒格，其☒是涉及☒入☒格波段☒的☒域的 PLPGSQL 用☒函数的☒果。
- ST_Max4ma - 可用性：2.0.0 ☒算☒域中最大像素☒的☒格☒理函数。
- ST_Mean4ma - 可用性：2.0.0 ☒算☒域中平均像素☒的☒格☒理函数。
- ST_Min4ma - 可用性：2.0.0 ☒算☒域中最小像素☒的☒格☒理函数。
- ST_ModEdgeHeal - 可用性：2.0 通☒☒除☒接☒条☒的☒点、修改第一条☒并☒除第二条☒来修复☒条☒。返回已☒除☒点的 id。
- ST_MoveIsoNode - 可用性：2.0.0 在拓扑☒☒中将一个孤立☒点从一个点移☒到☒一个点。如果新的 apoint 几何☒象已存在作☒一个☒点，☒会抛出☒☒。返回移☒的描述。
- ST_NewEdgeHeal - 可用性：2.0 通☒☒除☒接☒条☒的☒点、☒除☒条☒并用方向与提供的第一条☒相同的☒替☒它☒来修复☒条☒。
- ST_Node - 可用性：2.0.0 ☒点是☒的集合。
- ST_NumPatches - 可用性：2.0.0 返回多面体曲面上的面数。☒于非多面体几何形状将返回 null。
- ST_OffsetCurve - 可用性：2.0 返回距☒入☒☒定距离和方向的偏移☒。
- ST_PatchN - 可用性：2.0.0 返回多☒形曲面（PolyhedralSurface）的第 N 个几何体（面）。
- ST_Perimeter - 可用性 2.0.0：引入了☒地理的支持返回多☒形几何或地理的☒界☒度。

- ST_PixelAsPolygon - 可用性：2.0.0 返回限定特定行和列的像素的多☒形几何形状。
- ST_PixelAsPolygons - 可用性：2.0.0 返回包☒☒格☒的每个像素的多☒形几何☒形以及每个像素的☒、X 和 Y ☒格坐☒。
- ST_Project - 可用性：2.0.0 返回从起点按距离和方位角（方位角）投影的点。
- ST_Quantile - 可用性：2.0.0 ☒算☒本或☒体上下文中☒格或☒格表覆盖范☒的分位数。因此，可以☒☒某个☒是否位于☒格的 25%、50%、75% 百分位☒。
- ST_Range4ma - 可用性：2.0.0 ☒算☒域中像素☒范☒的☒格☒理函数。
- ST_Reclass - 可用性：2.0.0 ☒建由从原始数据重新分☒的波段☒型☒成的新☒格。nband 是要更改的波段。如果未指定 nband，☒假定☒ 1。所有其他波段均按原☒返回。使用案例：将 16BUI 波段☒☒☒ 8BUI 等，以便更☒☒地呈☒☒可☒格式。
- ST_RelateMatch - 可用性：2.0.0 ☒☒ DE-9IM 交集矩☒是否与交集矩☒模式匹配
- ST_RemEdgeModFace - 可用性：2.0 ☒除一条☒，如果☒☒将☒个面分开，☒☒除一个面并修改☒一个面以覆盖☒个面的空☒。
- ST_RemEdgeNewFace - 可用性：2.0 ☒除一条☒，如果☒除的☒将☒个面分开，☒☒除原始面并用新面替☒它☒。
- ST_Resample - 可用性：2.0.0 需要 GDAL 1.6.1+ 重采☒一个☒格☒像，可以指定重新采☒算法、新的尺寸、任意的☒格角点，以及一☒☒格地理参考属性，☒些属性可以自己定☒，也可以从☒一个☒格☒像中借用。
- ST_Rescale - 可用性：2.0.0 需要 GDAL 1.6.1+ 通☒☒☒整☒格的比例（或像素大小）来重新采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear、Cubic、CubicSpline、Lanczos、Max 或 Min 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_Reskew - 可用性：2.0.0 需要 GDAL 1.6.1+ 通☒☒☒整☒格的☒斜（或旋☒参数）来重采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear、Cubic、CubicSpline 或 Lanczos 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_SameAlignment - 可用性：2.0.0 如果☒格具有相同的☒斜、比例、空☒参考和偏移（像素可以放在同一网格上而不切割成像素），☒返回 true；如果没有注意☒☒☒☒，☒返回 false。
- ST_SetBandIsNoData - 可用性：2.0.0 将☒区的 isnodata ☒志☒置☒ TRUE。
- ST_SharedPaths - 可用性：2.0.0 返回一个集合，其中包含☒个☒入☒串/多☒串共享的路径。
- ST_Slope - 可用性：2.0.0 返回高程☒格☒的坡度（默☒以度☒☒位）。☒于分析地形很有用。
- ST_Snap - 可用性：2.0.0 将☒入几何体的☒段和☒点捕捉到参考几何体的☒点。
- ST_SnapToGrid - 可用性：2.0.0 需要 GDAL 1.6.1+ 通☒将☒格捕捉到网格来重采☒☒格。新的像素☒是使用 NearestNeighbor（英☒或美式拼写）、Bilinear, Cubic, CubicSpline 或 Lanczos 重采☒算法☒算的。默☒☒ NearestNeighbor。
- ST_Split - 可用性：2.0.0 需要 GEOS 返回通☒将一个几何体分割☒☒一个几何体而☒建的几何体集合。
- ST_StdDev4ma - 可用性：2.0.0 ☒算☒域中像素☒的☒准偏差的☒格☒理函数。
- ST_Sum4ma - 可用性：2.0.0 ☒格☒理函数，☒算☒域中所有像素☒的☒和。
- ST_SummaryStats - 可用性：2.0.0 返回一☒☒☒☒☒信息，包括☒定☒格或☒格覆盖的☒格☒的☒数、☒和、均☒、☒准差、最小☒和最大☒。如果未指定☒号，☒假定☒☒ 1。
- ST_Transform - 可用性：2.0.0 需要 GDAL 1.6.1+ 使用指定的重采☒算法将已知空☒参考系☒中的☒格重新投影到☒一个已知空☒参考系☒。☒☒有 NearestNeighbor、Bilinear、Cubic、CubicSpline、Lanczos（默☒☒ NearestNeighbor）。
- ST_UnaryUnion - 可用性：2.0.0 ☒算☒个几何体的☒件的并集。

- ST_Union - 可用性：2.0.0 将一□□格切片的并集返回□由 1 个或多个波段□成的□个□格。
- ST_ValueCount - 可用性：2.0.0 返回一□□□，其中包含像素□□以及具有□定□集的□格（或□格覆盖范□）□定□中的像素数□数。如果未指定波段，□默□□波段 1。默□情况下，不□算□点数据□像素。□出像素中的所有其他□，并将像素□□四舍五入到最接近的整数。
- TopoElementArray_Agg - 可用性：2.0.0 返回一□ element_id、□型数□ (topoelements) 的 topoelementarray。
- TopoGeo_AddLineString - 可用性：2.0.0 Adds a linestring to an existing topology using a tolerance and possibly splitting existing edges/faces.
- TopoGeo_AddPoint - 可用性：2.0.0 使用容差并可能分割□有□向□有拓扑添加点。
- TopoGeo_AddPolygon - 可用性：2.0.0 使用公差并可能分割□有□/面将多□形添加到□有拓扑。返回面□□符。
- TopologySummary - 可用性：2.0.0 □取拓扑名称并提供拓扑中□象□型的□□□数。
- Topology_Load_Tiger - 可用性：2.0.0 将 tiger 数据的定□区域加□到 PostGIS 拓扑中，并将老虎数据□□□拓扑的空□参考，并捕捉到拓扑的精度公差。
- toTopoGeom - 可用性：2.0 将□□几何□形□□□拓扑几何□形。
- ~ - 可用性：2.0.0 如果 A 的□界框包含 B 的□界框，□返回 TRUE。使用双精度□界框。
- ~= - 可用性：2.0.0 如果 A 的□界框与 B 的□界框相同，□返回 TRUE。

PostGIS 的功能增□ 2.0

- && - 增□：2.0.0 引入了□多面体曲面的支持。如果 A 的 2D □界框与 B 的 2D □界框相交，□返回 TRUE。
- AddGeometryColumn - 增□：2.0.0 引入了 use_typmod 参数。默□□建 typmod 几何列而不是基于□束。将 geometry（几何）列添加到□有表。
- Box2D - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。返回表示几何□形的 2D 范□的 BOX2D。
- Box3D - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。返回表示几何体 3D 范□的 BOX3D。
- CreateTopology - 增□：2.0 添加接受 hasZ 的格式□建一个新的拓扑架□并将其注册到 topology.topology 表中。
- Geocode - 增□功能：2.0.0 支持 Tiger 2010 □□化数据并修改了一些□□以提高地理□□的速度和准确性，并将点从中心□偏移到街道地址所在的一□。新参数 max_results 可用于指定最佳□果的数量或□返回最佳□果。将地址作□字符串（或其他□准化地址）□入，并□出一□可能的位置，其中包括 NAD 83 □□度中的点几何□形、每个位置的□准化地址以及□□。□□越低，匹配的可能性越大。□果首先按最低□分排序。可以□□□入最大□果数，默□□ 10，以及 restrict_region（默□□ NULL）
- GeometryType - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。以文本形式返回几何的□型。
- Populate_Geometry_Columns - 增□：2.0.0 引入了 use_typmod 可□参数，允□控制是否使用 typmodifiers 或□□□束□建列。确保几何列由□型修□符定□或具有适当的空□□束。
- ST_3DExtent - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。返回几何□形的 3D □界框的聚合函数。
- ST_Affine - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。□几何体□用 3D 仿射□□。
- ST_Area - 增□：2.0.0 - 引入了□ 2D 多面体曲面的支持。返回多□形几何体的面□。
- ST_AsBinary - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。返回不□ SRID 元数据的几何/地理的 OGC/ISO 已知的二□制 (WKB) 表示形式。
- ST_AsBinary - 增□：2.0.0 支持更高坐□□度。返回不□ SRID 元数据的几何/地理的 OGC/ISO 已知的二□制 (WKB) 表示形式。

- ST_AsBinary - 增☒：2.0.0 支持地理中的字☒序。返回不☒ SRID 元数据的几何/地理的 OGC/ISO 已知的二☒制 (WKB) 表示形式。
- ST_AsEWKB - 增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。返回☒有 SRID 元数据的几何☒形的☒展已知的二☒制 (EWKB) 表示形式。
- ST_AsEWKT - 增☒：2.0.0 引入了☒地理、多面体曲面、三角形和 TIN 的支持。使用 SRID 元数据返回几何☒形的已知文本 (WKT) 表示形式。
- ST_AsGML - 增☒：2.0.0 引入了前☒支持。引入了 GML3 的☒☒ 4，以允☒使用 LineString 而不是☒条的 Curve ☒☒。引入了☒多面体曲面和 TINS 的 GML3 支持。引入☒☒ 32 来☒出☒框。将几何☒形作☒ GML 版本 2 或 3 元素返回。
- ST_AsKML - 增☒：2.0.0 - 添加前☒命名空☒，使用默☒和命名参数将几何☒形作☒ KML 元素返回。
- ST_Azimuth - 增☒：2.0.0 引入了☒地理的支持。返回☒点之☒直☒的基于北方的方位角。
- ST_Dimension - 增☒：2.0.0 引入了多面体曲面支持和 TIN 支持。当☒定空几何☒，它不再引☒异常。返回几何☒形的拓扑☒数。
- ST_Dump - 增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。返回几何☒件的一☒ geometry_dump 行。
- ST_DumpPoints - 增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。返回几何☒形中坐☒的一☒ geometry_dump 行。
- ST_Expand - 增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。返回从☒一个☒界框或几何☒形☒展的☒界框。
- ST_Extent - 增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。返回几何☒形☒界框的聚合函数。
- ST_Force2D - 增☒：2.0.0 引入了☒多面体曲面的支持。☒制几何☒形☒入“二☒模式”。
- ST_Force3D - 增☒：2.0.0 引入了☒多面体曲面的支持。☒制几何☒形☒入 XYZ 模式。☒是 ST_Force3DZ 的☒名。
- ST_Force3DZ - 增☒：2.0.0 引入了☒多面体曲面的支持。☒制几何☒形☒入 XYZ 模式。
- ST_ForceCollection - 增☒：2.0.0 引入了☒多面体曲面的支持。将几何☒形☒☒☒几何集合（GEOMETRYCOLLECTION）。
- ST_ForceRHR - 增☒：2.0.0 引入了☒多面体曲面的支持。☒制多☒形☒点的方向遵循右手定☒。
- ST_GMLToSQL - 增☒功能：2.0.0 支持多面体曲面和 TIN 。从 GML 表示返回指定的 ST_Geometry ☒。☒是 ST_GeomFromGML 的☒名
- ST_GMLToSQL - 增☒：2.0.0 引入了多面体曲面支持和 TIN 支持。从 GML 表示返回指定的 ST_Geometry ☒。☒是 ST_GeomFromGML 的☒名
- ST_GeomFromEWKB - 增☒功能：2.0.0 支持多面体曲面和 TIN 。从☒展已知的二☒制表示 (EWKB) 返回指定的 ST_Geometry ☒。
- ST_GeomFromEWKT - 增☒功能：2.0.0 支持多面体曲面和 TIN 。从☒展已知的文本表示 (EWKT) 返回指定的 ST_Geometry ☒。
- ST_GeomFromGML - 增☒功能：2.0.0 支持多面体曲面和 TIN 。将几何☒形的 GML 表示形式作☒☒入并☒出 PostGIS 几何☒象
- ST_GeomFromGML - 增☒：2.0.0 引入了多面体曲面支持和 TIN 支持。将几何☒形的 GML 表示形式作☒☒入并☒出 PostGIS 几何☒象
- ST_GeometryN - 增☒功能：引入了 2.0.0 ☒多面体曲面、三角形和三角网的支持。返回几何集合的一个元素。
- ST_GeometryType - 增☒：2.0.0 引入了☒多面体曲面的支持。以文本形式返回几何☒形的 SQL-MM ☒型。

- ST_IsClosed - 增□：2.0.0 引入了□多面体曲面的支持。□□ LineStrings 的起点和□点是否重合。□于多面体表面□□是否□合（□心）。
- ST_MakeEnvelope - 增□：2.0：引入了在不指定 SRID 的情况下指定外包矩形的功能。根据最小和最大坐□□建矩形多□形。
- ST_MakeValid - 增□：2.0.1 速度提升□□在不□失□点的情况下使无效几何体有效。
- ST_NPoints - 增□：2.0.0 引入了□多面体曲面的支持。返回几何□形中的点数（□点）。
- ST_NumGeometries - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。返回几何集合中的元素数量。
- ST_Relate - 增□：2.0.0 - 添加了□指定□界□点□□的支持。□□□个几何□形是否具有与交集矩□模式匹配的拓扑关系，或□算它□的交集矩□
- ST_Rotate - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。□原点旋□几何体。
- ST_Rotate - 增□：2.0.0 添加了用于指定旋□原点的附加参数。□原点旋□几何体。
- ST_RotateX - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。□ X □旋□几何体。
- ST_RotateY - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。□ Y □旋□几何体。
- ST_RotateZ - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。□ Z □旋□几何体。
- ST_Scale - 增□功能：引入了 2.0.0 □多面体曲面、三角形和三角网的支持。按□定因子□放几何□形。
- ST_ShiftLongitude - 增□功能：2.0.0 支持多面体曲面和 TIN 。在 -180-180 和 0-360 之□移□几何□形的□度坐□。
- ST_Summary - 增□：在 2.0.0 中添加了地理支持返回几何内容的文本摘要。
- ST_Transform - 增□：2.0.0 引入了□多面体曲面的支持。返回坐□□□□不同空□参考系的新几何□形。
- ST_Value - 增□：2.0.0 添加了 except_nodata_value 可□参数。返回指定列 x、行 y 像素或特定几何点□□定波段的□。波段□号从 1 开始，如果未指定，□默□□ 1。如果将参数 exclude_nodata_value □置□ false，□所有像素都被□□与 nodata 像素相交并返回其□。如果未□□参数 exclude_nodata_value，□从□格的元数据中□取□□。
- ValidateTopology - 增□：2.0.0 更有效的□□交叉□□并修复了先前版本中存在的□□。返回一□ validatetopology_returntype □象，□□□明拓扑□□。

PostGIS 中的功能□生□化 2.0

- AddGeometryColumn - 更改：2.0.0 此函数不再更新 geometry_columns，因□ geometry_columns 是从系□目□□取的□□。默□情况下，它也不□建□束，而是使用 PostgreSQL 内置的□型修□符行□。因此，例如，使用此函数□建 wgs84 POINT 列□在相当于：ALTER TABLE some_table ADD COLUMN geom geometry(Point,4326); 将 geometry（几何）列添加到□有表。
- AddGeometryColumn - 更改：2.0.0 如果您需要□束的旧行□，□使用默□的 use_typmod，但将其□置□ false。将 geometry（几何）列添加到□有表。
- AddGeometryColumn - 更改：2.0.0 □□不能再在 Geometry_columns 中手□注册，但是根据几何 Typmod 表几何□形□建并在没有包装器函数的情况下使用的□□将正确注册自身，因□它□□承了父表列的 Typmod 行□。使用□出其他几何□形的几何函数的□□需要□□□ typmod 几何□形，以便□些□□几何列能□在 geometry_columns 中正确注册。□参□。将 geometry（几何）列添加到□有表。
- Box3D - 更改：2.0.0 在 2.0 之前的版本中，曾□有 box2d 而不是 box3d。由于 box2d 是已弃用的□型，因此已更改□ box3d。返回□格封□框的 box 3d 表示形式。
- DropGeometryColumn - 更改：2.0.0 提供此函数是□了向后兼容。□在，由于 Geometry_columns □在是□□系□目□的□□，因此您可以使用 ALTER TABLE □除几何列，就像□除任何其他表列一□从空□表中移除 geometry（几何）列。

- DropGeometryTable - 更改：2.0.0 提供此函数是⊠了向后兼容。⊠在，由于 Geometry_columns ⊠在是⊠⊠系⊠目⊠的⊠⊠，因此您可以使用 DROP TABLE ⊠除具有几何列的表，就像任何其他表一⊠⊠除表及其在 geometry_columns 中的所有引用。
- Populate_Geometry_Columns - 更改：2.0.0 默⊠情况下，⊠在使用⊠型修⊠符而不是⊠⊠⊠束来⊠束几何⊠型。您仍然可以通⊠使用新的 use_typmod 并将其⊠置⊠ false 来使用⊠⊠⊠束行⊠。确保几何列由⊠型修⊠符定⊠或具有适当的空⊠⊠束。
- ST_3DExtent - 更改：2.0.0 在之前的版本中，⊠曾⊠被称⊠ ST_Extent3D 返回几何⊠形的 3D ⊠界框的聚合函数。
- ST_3DLength - 更改：2.0.0 在之前的版本中，⊠曾⊠被称⊠ ST_Length3D 返回⊠性几何体的 3D ⊠度。
- ST_3DMakeBox - 更改：2.0.0 在之前的版本中，⊠曾⊠被称⊠ ST_MakeBox3D ⊠建由⊠个 3D 点几何⊠形定⊠的 BOX3D。
- ST_3DPerimeter - 更改：2.0.0 在之前的版本中，⊠曾⊠被称⊠ ST_Perimeter3D 返回多⊠形几何体的 3D 周⊠。
- ST_AsBinary - 更改：2.0.0 此函数的⊠入不能是未知的——必⊠是几何⊠形。ST_AsBinary('POINT(1 2)') 等⊠⊠不再有效,您将收到 n st_asbinary(unknown) is not unique error。⊠似的代⊠需要更改⊠ ST_AsBinary('POINT(1 2)'::geometry);。如果不可能，⊠安装 legacy.sql。返回不⊠ SRID 元数据的几何/地理的 OGC/ISO 已知的二⊠制 (WKB) 表示形式。
- ST_AsGML - 更改：2.0.0 使用默⊠命名参数将几何⊠形作⊠ GML 版本 2 或 3 元素返回。
- ST_AsGeoJSON - 更改：2.0.0 支持默⊠参数和命名参数。以 GeoJSON 格式返回一个几何体或要素。
- ST_AsSVG - 更改：2.0.0 - 添加了⊠默⊠参数和命名参数的支持返回几何体的 SVG 路径数据。
- ST_EndPoint - 更改：2.0.0 不再适用于⊠个几何体 MultiLineStrings。在旧版本的 PostGIS 中，⊠行 MultiLineString 可以使用此函数并返回⊠点。在 2.0.0 中，它像任何其他 MultiLineString 一⊠返回 NULL。旧的行⊠是一个未⊠⊠的功能，但是那些假⊠将数据存⊠⊠ LINESTRING 的人可能会在 2.0.0 中遇到⊠些返回 NULL 的情况。返回 LineString 或 CircularLineString 的最后一个点。
- ST_GDALDrivers - 更改：2.0.6、2.1.3 - 默⊠情况下不⊠用任何⊠⊠程序，除非⊠置了 GUC 或⊠境⊠量 gdal_enabled_drivers。通⊠ GDAL 返回 PostGIS 支持的⊠格格式列表。ST_AsGDALRaster ⊠可使用 can_write=True 的格式
- ST_GeomFromText - 更改：2.0.0 在 PostGIS 的早期版本中,允⊠ ST_GeomFromText('GEOMETRYCOLLECTIO。⊠了更好地符合 SQL/MM ⊠准,⊠在 PostGIS 2.0.0 中⊠在是非法的。⊠在⊠⊠写成 ST_GeomFromText('GEOMETF EMPTY') 从已知的文本表示 (WKT) 返回指定的 ST_Geometry ⊠。
- ST_GeometryN - 更改：2.0.0 版本。之前的版本⊠于奇异几何⊠形会返回 NULL。⊠在已更改⊠在 ST_GeometryN(..,1 情况下返回几何⊠形。返回几何集合的一个元素。
- ST_IsEmpty - 已更改：2.0.0 之前的 PostGIS 版本允⊠ ST_GeomFromText（“GEOMETRYCOLLECTION(EMPTY)”）。在 PostGIS 2.0.0 中，⊠是不正确的，因⊠它更符合 SQL/MM ⊠准⊠⊠几何⊠形是否⊠空。
- ST_Length - 更改：2.0.0 重大更改 - 在之前的版本中，将此⊠用于地理⊠型的多/多⊠形将⊠您提供多⊠形/多多⊠形的周⊠。在 2.0.0 中，⊠已更改⊠返回 0 以符合几何行⊠。如果您想要多⊠形的周⊠，⊠使用 ST_Perimeter 返回⊠性几何体的二⊠⊠度。
- ST_LocateAlong - 更改：2.0.0 在之前的版本中，⊠曾⊠被称⊠ ST_Locate_Along_Measure。返回几何上与⊠量⊠匹配的点。
- ST_LocateBetween - 更改：2.0.0 - 在之前的版本中，⊠曾⊠被称⊠ ST_Locate_Between_Measures。返回与⊠量范⊠匹配的几何⊠形部分。
- ST_ModEdgeSplit - 更改：2.0 - 在之前的版本中，它被⊠⊠命名⊠ ST_ModEdgesSplit 通⊠沿⊠有⊠⊠建新⊠点、修改原始⊠并添加新⊠来分割⊠。

- ST_NumGeometries - 更改：2.0.0 在之前的版本中，如果几何⊠形不是 collection/MULTI ⊠型，⊠会返回 NULL。2.0.0 ⊠在⊠于⊠个几何⊠形返回 1，例如 POLYGON、LINESTRING、POINT。返回几何集合中的元素数量。
- ST_NumInteriorRings - 更改：2.0.0 - 在之前的版本中，它允⊠⊠⊠ MULTIPOLYGON，返回第一个 POLYGON 的内⊠数量。返回多⊠形的内⊠（孔）数。
- ST_PointN - 更改：2.0.0 不再适用于⊠个几何 multilinestrings。在旧版本的 PostGIS 中——⊠行 multilinestring 可以与此函数很好地配合并返回起点。在 2.0.0 中，它像任何其他 multilinestring 一⊠只返回 NULL。返回几何⊠形中第一个⊠串或⊠形⊠串中的第 N 个点。
- ST_ScaleX - 更改：2.0.0。在 WKTRaster 版本中，⊠称⊠ ST_PixelSizeX。返回像素⊠度的 X 分量（以坐⊠参考系⊠⊠位）。
- ST_ScaleY - 更改：2.0.0。在 WKTRaster 版本中，⊠称⊠ ST_PixelSizeY。返回像素高度的 Y 分量（以坐⊠参考系⊠⊠位）。
- ST_SetScale - 更改：2.0.0 在 WKTRaster 版本中，⊠称⊠ ST_SetPixelSize。⊠在 2.0.0 中⊠生了⊠化。以坐⊠参考系⊠⊠位⊠置像素的 X 和 Y 大小。数字⊠位/像素⊠度/高度。
- ST_StartPoint - 更改：2.0.0 不再适用于⊠个几何体 MultiLineStrings。在旧版本的 PostGIS 中，⊠行 MultiLineString 可以与此函数很好地配合并返回起点。在 2.0.0 中，它像任何其他 MultiLineString 一⊠只返回 NULL。旧的行⊠是一个未⊠⊠的功能，但是那些假⊠将数据存⊠⊠ LINESTRING 的人可能会在 2.0.0 中遇到⊠些返回 NULL 的情况。返回 LineString 的第一个点.

13.12.13 PostGIS 新增功能或增⊠功能（1.5）

下面⊠出的功能是添加或增⊠的 PostGIS 功能。

PostGIS 中的新功能 1.5

- && - 可用性：1.5.0 引入了⊠地理的支持。如果 A 的 2D ⊠界框与 B 的 2D ⊠界框相交，⊠返回 TRUE。
- PostGIS_LibXML_Version - 可用性：1.5 返回 libxml2 ⊠的版本号。
- ST_AddMeasure - 可用性：1.1.0 沿⊠性几何形状插⊠⊠量⊠。
- ST_AsBinary - 可用性：1.5.0 支持地理位置。返回不⊠ SRID 元数据的几何/地理的 OGC/ISO 已知的二⊠制 (WKB) 表示形式。
- ST_AsGML - 可用性：1.5.0 支持地理位置。将几何⊠形作⊠ GML 版本 2 或 3 元素返回。
- ST_AsGeoJSON - 可用性：1.5.0 支持地理位置。以 GeoJSON 格式返回一个几何体或要素。
- ST_AsText - 可用性：1.5 - 引入了地理支持。返回不⊠ SRID 元数据的几何/地理的已知文本 (WKT) 表示形式。
- ST_Buffer - 可用性：1.5 - ST_Buffer 已得到增⊠，以适⊠各种端接和接⊠。例如，您可能希望将道路⊠串⊠⊠⊠街道面，并希望将端点⊠⊠平面或正方形而不是⊠。添加了地理的薄包装器。⊠算覆盖距几何体⊠定距离内所有点的几何体。
- ST_ClosestPoint - 可用性：1.1.0 返回 g1 上最接近 g2 的 2D 点。⊠是从一个几何体到⊠一个几何体的最短直⊠的第一个点。
- ST_CollectionExtract - 可用性：1.1.0 ⊠定一个几何集合，返回⊠包含指定⊠型元素的多几何⊠形。
- ST_Covers - 可用性：1.5 - 引入了地理支持。⊠⊠ B 的每个点是否都位于 A 中
- ST_DFullyWithin - 可用性：1.1.0 Tests if a geometry is entirely inside a distance of another
- ST_DWithin - 可用性：1.5.0 引入了⊠地理的支持⊠⊠⊠个几何⊠形是否在⊠定距离内
- ST_Distance - 可用性：1.5.0 地理支持在 1.5 中引入。提高了平面的速度，以更好地⊠理大型或多个⊠点几何⊠形返回⊠个几何或地理⊠之⊠的距离。

- ST_DistanceSphere - 可用性：1.5 - 引入了⌧除点之外的其他几何⌧型的支持。之前的版本⌧适用于点。使用球形地球模型返回⌧个⌧度/⌧度几何形状之⌧的最小距离（以米⌧⌧位）。
- ST_DistanceSpheroid - 可用性：1.5 - 引入了⌧除点之外的其他几何⌧型的支持。之前的版本⌧适用于点。使用球体模型返回⌧个⌧度/⌧度几何形状之⌧的最小距离。
- ST_DumpPoints - 可用性：1.1.0 返回几何⌧形中坐⌧的一⌧ geometry_dump 行。
- ST_Envelope - 可用性：1.5.0 行⌧更改⌧⌧出双精度而不是 float4 返回表示几何⌧形⌧界框的几何⌧形。
- ST_Expand - 可用性：1.5.0 行⌧更改⌧⌧出双精度而不是 float4 坐⌧。返回从⌧一个⌧界框或几何⌧形⌧展的⌧界框。
- ST_GMLToSQL - 可用性：需要 1.5 libxml2 1.6+ 从 GML 表示返回指定的 ST_Geometry ⌧。⌧是 ST_GeomFromG 的⌧名
- ST_GeomFromGML - 可用性：需要 1.5 libxml2 1.6+ 将几何⌧形的 GML 表示形式作⌧⌧入并⌧出 PostGIS 几何⌧象
- ST_GeomFromKML - 可用性：1.5, 需要 libxml2 2.6+ 将几何⌧形的 KML 表示形式作⌧⌧入并⌧出 PostGIS 几何⌧象
- ST_HausdorffDistance - 可用性：1.1.0 返回⌧个几何⌧形之⌧的 Hausdorff 距离。
- ST_Intersection - 可用性：1.5 引入了⌧地理数据⌧型的支持。⌧算表示几何 A 和 B 的共享部分的几何。
- ST_Intersects - 可用性：1.5 引入了⌧地理的支持。⌧⌧⌧个几何⌧形是否相交（它⌧至少有一个共同点）
- ST_Length - 可用性：1.5.0 地理支持在 1.5 中引入。返回⌧性几何体的二⌧⌧度。
- ST_LongestLine - 可用性：1.1.0 返回⌧个几何⌧形之⌧的二⌧最⌧⌧。
- ST_MakeEnvelope - 可用性：1.5 根据最小和最大坐⌧⌧建矩形多⌧形。
- ST_MaxDistance - 可用性：1.1.0 返回⌧个几何⌧形之⌧的二⌧最大距离（以投影⌧位表示）。
- ST_ShortestLine - 可用性：1.1.0 返回⌧个几何⌧形之⌧的 2D 最短⌧
- ~= - 可用性：1.5.0 改⌧了行⌧如果 A 的⌧界框与 B 的⌧界框相同，⌧返回 TRUE。

13.12.14 PostGIS 新增功能或增⌧功能（1.4）

下面⌧出的功能是添加或增⌧的 PostGIS 功能。

PostGIS 中的新功能 1.4

- Populate_Geometry_Columns - 可用性：1.4.0 确保几何列由⌧型修⌧符定⌧或具有适当的空⌧⌧束。
- ST_Collect - 可用性：引入了 1.4.0 - ST_Collect（几何）。ST_Collect 已得到增⌧，可以更快地⌧理更多几何⌧形。从一⌧几何⌧形⌧建 GeometryCollection 或 Multi* 几何⌧形。
- ST_ContainsProperly - 可用性：1.4.0 ⌧⌧ B 的每个点是否都位于 A 的内部
- ST_GeoHash - 可用性：1.4.0 返回几何⌧形的 GeoHash 表示形式。
- ST_IsValidReason - 有效性：1.4 返回⌧明几何⌧形是否有效或无效原因的文本。
- ST_LineCrossingDirection - 有效性：1.4 返回一个数字，指示⌧个 LineString 的交叉行⌧
- ST_LocateBetweenElevations - 可用性：1.4.0 返回位于高程 (Z) 范⌧内的几何⌧形部分。
- ST_MakeLine - 可用性：1.4.0 - 引入了 ST_MakeLine(geomarray)。ST_MakeLine 聚合函数得到增⌧，可以更快地⌧理更多点。从 Point, MultiPoint, 或 LineString geometries ⌧建 LineString。
- ST_MinimumBoundingCircle - 可用性：1.4.0 返回包含几何⌧形的最小⌧形多⌧形。
- ST_Union - 可用性：1.4.0 - ST_Union 得到增⌧。PostgreSQL 中引入了 ST_Union(geomarray) 以及更快的集合聚合。⌧算表示⌧入几何⌧形的点集并集的几何⌧形。

13.12.15 PostGIS 新增功能或增⌧功能（1.3）

下面⌧出的功能是添加或增⌧的 PostGIS 功能。

PostGIS 中的新功能 1.3

- ST_AsGML - 可用性：1.3.2 将几何⌧形作⌧ GML 版本 2 或 3 元素返回。
- ST_AsGeoJSON - 可用性：1.3.4 以 GeoJSON 格式返回一个几何体或要素。
- ST_CurveToLine - 可用性：1.3.0 将包含曲⌧的几何⌧形⌧⌧⌧⌧性几何⌧形。
- ST_LineToCurve - 可用性：1.3.0 将⌧性几何⌧形⌧⌧⌧曲⌧几何⌧形。
- ST_SimplifyPreserveTopology - 可用性：1.3.3 使用 Douglas-Peucker 算法返回几何⌧形的⌧化且有效表示。

Chapter 14

☒告☒☒

14.1 ☒告☒件☒☒

有效☒告☒☒是帮助 PostGIS 开☒的基本方法。最有效的☒☒☒告是使 PostGIS 开☒人☒能☒重☒它，因此它最好包含触☒它的脚本以及有关☒☒到它的☒境的所有信息。☒行 `SELECT postgis_full_version()` [☒于 PostGIS] 和 `SELECT version()` [☒于 postgresql] 可以提取足☒好的信息。

如果您没有使用最新版本，☒☒得先☒看其版本☒更日志，以了解您的☒☒是否已得到修复。

使用 PostGIS ☒☒跟踪器将确保您的☒告不会被☒弃，并☒您随☒了解其☒理☒程。在☒告新☒☒之前，☒☒☒数据☒以☒看它是否是已知☒☒，如果是，☒添加您所掌握的有关它的任何新信息。

在提交新☒告之前，您可能需要☒☒ Simon Tatham 关于如何有效☒告☒☒ 。

14.2 ☒告文档☒☒

文档☒准确反映☒件的功能和行☒。如果没有，可能是因☒☒件☒☒或文档☒☒或缺陷。

文档☒☒也可以☒告☒ PostGIS bug 跟踪器。

如果您的修☒很☒☒，只需在新的☒☒跟踪器☒☒中☒其☒行描述，并具体☒明其在文档中的位置。

如果您的更改更广泛，那么☒丁☒☒是首☒。☒是 Unix 上的四步☒程（假☒您已☒安装了 git ）：

1. 克隆 PostGIS 的 git 存☒☒。在 Unix 上，☒入：
   git clone https://git.osgeo.org/gitea/postgis/postgis.git
   ☒将存☒在 postgis 目☒中
2. 使用您最喜☒的文本☒☒器☒文档☒行更改。在 Unix 上，☒入（例如）：
   vim doc/postgis.xml
   ☒注意，☒文档是用 DocBook XML 而不是 HTML ☒写的，因此如果您不熟悉它，☒按照文档其余部分的示例☒行操作。
3. 制作一个包含与文档主副本的差异的☒丁文件。在 Unix 上，☒入：
   git diff doc/postgis.xml > doc.patch
4. 将☒丁附加到☒☒跟踪器中的新☒☒。

Appendix A

附☒

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 重大☒化

#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 增☒功能

#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 重大☒化

#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.

非常感☒我☒的翻☒☒☒，特☒是：

Dapeng Wang, Zuo Chenwei from HighGo (Chinese Team)

Teramoto Ikuhiro (Japanese Team)

Vincent Bre (French Team)

A.4.1 重大☒化

#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 新特性

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_GreeneApproxConvexPar and CG_OptimalConvexPartition (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_Triangulate, 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 增☒功能

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)