PostGIS 3.7.0rc3 手册

Contents

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

2 PostGIS 安装 4
2.1 ☒短版本 . . . 4
2.2 从源代☒☒☒和安装 . . . 4
2.2.1 ☒取源代☒ . . . 5
2.2.2 安装要求 . . . 5
2.2.3 ☒建配置 . . . 7
2.2.4 ☒建 . . . 8
2.2.5 ☒建和部署 PostGIS ☒展 . . . 8
2.2.6 ☒☒ . . . 10
2.2.7 安装 . . . 13
2.3 常☒☒☒ . . . 13

3 PostGIS 管理 15
3.1 性能☒☒ . . . 15
3.1.1 ☒☒ . . . 15
3.1.2 ☒行 . . . 16
3.2 配置☒格支持 . . . 16
3.3 ☒建空☒数据☒ . . . 17
3.3.1 使用☒展（EXTENSION）☒用空☒数据☒ . . . 17
3.3.2 不通☒☒展（EXTENSION）的方式☒用空☒数据☒（不建☒） . . . 17
3.4 升☒空☒数据☒ . . . 18
3.4.1 ☒升☒ . . . 18
3.4.1.1 使用 EXTENSION ☒升☒ . . . 18
3.4.1.2 不使用 EXTENSION ☒升☒ . . . 20
3.4.1.3 硬升☒ . . . 20

4 数据管理 22
4.1 空☒数据模型 . . . 22
4.1.1 OGC 几何☒形 . . . 22
4.1.1.1 点（Point） . . . 23
4.1.1.2 ☒串（LineString） . . . 23
4.1.1.3 ☒性☒（LinearRing） . . . 23
4.1.1.4 多☒形（Polygon） . . . 23
4.1.1.5 多点（MultiPoint） . . . 23
4.1.1.6 多☒（MultiLineString） . . . 23
4.1.1.7 多面（MultiPolygon） . . . 24
4.1.1.8 几何集合（GeometryCollection） . . . 24
4.1.1.9 多面体曲面（PolyhedralSurface） . . . 24
4.1.1.10三角形（Triangle） . . . 24
4.1.1.11TIN . . . 24
4.1.2 SQL/MM 第 3 部分-曲☒ . . . 25
4.1.2.1 ☒弧串（CircularString） . . . 25
4.1.2.2 复合曲☒（CompoundCurve） . . . 25
4.1.2.3 曲☒多☒形（CurvePolygon） . . . 25
4.1.2.4 多曲☒（MultiCurve） . . . 26
4.1.2.5 多曲面（MultiSurface） . . . 26
4.1.3 WKT 和 WKB . . . 26
4.2 几何数据☒型 . . . 27
4.2.1 PostGIS EWKB 和 EWKT . . . 28
4.3 地理数据☒型 . . . 29
4.3.1 ☒建地理表 . . . 29
4.3.2 使用地理表 . . . 31
4.3.3 何☒使用地理数据☒型 . . . 32
4.3.4 地理高☒常☒☒☒解答 . . . 33
4.4 几何有效性☒☒ . . . 33
4.4.1 ☒☒几何 . . . 33
4.4.2 有效的几何形状 . . . 35
4.4.3 有效性管理 . . . 40
4.5 空☒参考系☒ . . . 42
4.5.1 SPATIAL_REF_SYS 表 . . . 43
4.5.2 用☒自定☒空☒参考系☒ . . . 44
4.6 空☒表 . . . 44
4.6.1 ☒建空☒表 . . . 44
4.6.2 GEOMETRY_COLUMNS ☒☒ . . . 45
4.6.3 手☒注册几何列 . . . 46

4.7 加⊠空⊠数据 . . . 48
4.7.1 使用 SQL 加⊠数据 . . . 49
4.7.2 使用 Shapefile 加⊠器 . . . 49
4.8 提取空⊠数据 . . . 51
4.8.1 使用 SQL 提取数据 . . . 51
4.8.2 使用 Shapefile 文件⊠⊠程序 . . . 52
4.9 空⊠索引 . . . 53
4.9.1 GiST 索引 . . . 53
4.9.2 BRIN 索引 . . . 54
4.9.3 SP-GiST 索引 . . . 55
4.9.4 ⊠⊠索引的使用 . . . 56

5 空⊠⊠⊠ 57
5.1 空⊠关系的确定 . . . 57
5.1.1 ⊠度⊠展九交模型（Dimensionally Extended 9-Intersection Model） . . . 57
5.1.2 命名空⊠关系 . . . 60
5.1.3 一般空⊠关系 . . . 61
5.2 使用空⊠索引 . . . 63
5.3 空⊠ SQL 示例 . . . 64

6 性能⊠化技巧 67
6.1 大几何形状的小表 . . . 67
6.1.1 ⊠⊠描述 . . . 67
6.1.2 解决⊠法 . . . 67
6.2 几何索引聚⊠ . . . 68
6.3 避免⊠度⊠⊠ . . . 68

7 PostGIS 参考手册 69
7.1 PostGIS Geometry/Geography/Box 数据⊠型 . . . 69
7.1.1 box2d . . . 69
7.1.2 box3d . . . 70
7.1.3 geometry . . . 70
7.1.4 geometry_dump . . . 71
7.1.5 geography . . . 71
7.2 表管理功能 . . . 71
7.2.1 AddGeometryColumn . . . 71
7.2.2 DropGeometryColumn . . . 74
7.2.3 DropGeometryTable . . . 75
7.2.4 Find_SRID . . . 76
7.2.5 Populate_Geometry_Columns . . . 76

7.2.6 UpdateGeometrySRID . . . 78
7.3 几何□造函数 . . . 79
7.3.1 ST_Collect . . . 79
7.3.2 ST_LineFromMultiPoint . . . 83
7.3.3 ST_MakeEnvelope . . . 84
7.3.4 ST_MakeLine . . . 85
7.3.5 ST_MakePoint . . . 89
7.3.6 ST_MakePointM . . . 91
7.3.7 ST_MakePolygon . . . 93
7.3.8 ST_Point . . . 97
7.3.9 ST_PointZ . . . 98
7.3.10 ST_PointM . . . 99
7.3.11 ST_PointZM . . . 99
7.3.12 ST_Polygon . . . 100
7.3.13 ST_TileEnvelope . . . 102
7.3.14 ST_HexagonGrid . . . 104
7.3.15 ST_Hexagon . . . 109
7.3.16 ST_SquareGrid . . . 111
7.3.17 ST_Square . . . 112
7.3.18 ST_Letters . . . 113
7.3.19 ST_MakeNurbsCurve . . . 115
7.4 几何□□器 . . . 118
7.4.1 GeometryType . . . 118
7.4.2 ST_Boundary . . . 120
7.4.3 ST_BoundingDiagonal . . . 126
7.4.4 ST_CoordDim . . . 127
7.4.5 ST_Dimension . . . 128
7.4.6 ST_Dump . . . 130
7.4.7 ST_DumpPoints . . . 134
7.4.8 ST_DumpSegments . . . 139
7.4.9 ST_DumpRings . . . 142
7.4.10 ST_EndPoint . . . 144
7.4.11 ST_Envelope . . . 148
7.4.12 ST_ExteriorRing . . . 152
7.4.13 ST_GeometryN . . . 155
7.4.14 ST_GeometryType . . . 159
7.4.15 ST_HasArc . . . 160
7.4.16 ST_InteriorRingN . . . 161
7.4.17 ST_NumCurves . . . 163

7.4.18ST_CurveN . . . 164
7.4.19ST_IsClosed . . . 166
7.4.20ST_IsCollection . . . 170
7.4.21ST_IsEmpty . . . 172
7.4.22ST_IsPolygonCCW . . . 174
7.4.23ST_IsPolygonCW . . . 175
7.4.24ST_IsRing . . . 176
7.4.25ST_IsSimple . . . 178
7.4.26ST_IsSimpleDetail . . . 180
7.4.27ST_M . . . 182
7.4.28ST_MemSize . . . 183
7.4.29ST_NDims . . . 184
7.4.30ST_NPoints . . . 185
7.4.31ST_NRings . . . 186
7.4.32ST_NumGeometries . . . 187
7.4.33ST_NumInteriorRings . . . 188
7.4.34ST_NumInteriorRing . . . 189
7.4.35ST_NumPatches . . . 190
7.4.36ST_NumPoints . . . 190
7.4.37ST_PatchN . . . 191
7.4.38ST_PointN . . . 193
7.4.39ST_Points . . . 196
7.4.40ST_StartPoint . . . 197
7.4.41ST_StartM . . . 202
7.4.42ST_EndM . . . 205
7.4.43ST_SetStartM . . . 208
7.4.44ST_SetEndM . . . 211
7.4.45ST_Summary . . . 214
7.4.46ST_X . . . 216
7.4.47ST_Y . . . 216
7.4.48ST_Z . . . 217
7.4.49ST_ZMFlag . . . 218
7.4.50ST_HasZ . . . 219
7.4.51ST_HasM . . . 220
7.4.52ST_ControlPoints . . . 222
7.4.53ST_Degree . . . 223
7.4.54ST_Weights . . . 225
7.4.55ST_Knots . . . 227
7.4.56ST_NumControlPoints . . . 229

7.4.57ST_NurbsCurveIsRational . . . 230
7.4.58ST_IsNurbsCurve . . . 232
7.4.59ST_Evaluate . . . 234
7.4.60ST_NurbsToLineString . . . 235
7.5 几何⊠⊠器 . . . 237
7.5.1 ST_AddPoint . . . 237
7.5.2 ST_CollectionExtract . . . 239
7.5.3 ST_CollectionHomogenize . . . 241
7.5.4 ST_CurveToLine . . . 245
7.5.5 ST_Scroll . . . 249
7.5.6 ST_FlipCoordinates . . . 250
7.5.7 ST_Force2D . . . 251
7.5.8 ST_Force3D . . . 253
7.5.9 ST_Force3DZ . . . 255
7.5.10ST_Force3DM . . . 257
7.5.11ST_Force4D . . . 259
7.5.12ST_ForceCollection . . . 261
7.5.13ST_ForceCurve . . . 264
7.5.14ST_ForcePolygonCCW . . . 265
7.5.15ST_ForcePolygonCW . . . 266
7.5.16ST_ForceSFS . . . 268
7.5.17ST_ForceRHR . . . 269
7.5.18ST_LineExtend . . . 270
7.5.19ST_LineToCurve . . . 271
7.5.20ST_Multi . . . 273
7.5.21ST_Normalize . . . 274
7.5.22ST_Project . . . 275
7.5.23ST_QuantizeCoordinates . . . 277
7.5.24ST_RemovePoint . . . 279
7.5.25ST_RemoveRepeatedPoints . . . 279
7.5.26ST_RemoveIrrelevantPointsForView . . . 283
7.5.27ST_RemoveSmallParts . . . 287
7.5.28ST_Reverse . . . 289
7.5.29ST_Segmentize . . . 290
7.5.30ST_SetPoint . . . 293
7.5.31ST_ShiftLongitude . . . 296
7.5.32ST_WrapX . . . 298
7.5.33ST_SnapToGrid . . . 299
7.5.34ST_Snap . . . 302

7.5.35ST_SwapOrdinates . . . 306
7.6 几何有效性☒☒ . . . 307
7.6.1 ST_IsValid . . . 307
7.6.2 ST_IsValidDetail . . . 310
7.6.3 ST_IsValidReason . . . 312
7.6.4 ST_MakeValid . . . 314
7.7 空☒参考系☒功能 . . . 318
7.7.1 ST_InverseTransformPipeline . . . 318
7.7.2 ST_SetSRID . . . 320
7.7.3 ST_SRID . . . 322
7.7.4 ST_Transform . . . 322
7.7.5 ST_TransformPipeline . . . 327
7.7.6 postgis_srs_codes . . . 330
7.7.7 postgis_srs . . . 331
7.7.8 postgis_srs_all . . . 332
7.7.9 postgis_srs_search . . . 332
7.8 几何☒入 . . . 334
7.8.1 已知文本 (WKT) . . . 334
7.8.1.1 ST_BdPolyFromText . . . 334
7.8.1.2 ST_BdMPolyFromText . . . 334
7.8.1.3 ST_GeogFromText . . . 335
7.8.1.4 ST_GeographyFromText . . . 336
7.8.1.5 ST_GeomCollFromText . . . 337
7.8.1.6 ST_GeomFromEWKT . . . 337
7.8.1.7 ST_GeomFromMARC21 . . . 340
7.8.1.8 ST_GeometryFromText . . . 344
7.8.1.9 ST_GeomFromText . . . 345
7.8.1.10ST_LineFromText . . . 350
7.8.1.11ST_MLineFromText . . . 351
7.8.1.12ST_MPointFromText . . . 352
7.8.1.13ST_MPolyFromText . . . 352
7.8.1.14ST_PointFromText . . . 355
7.8.1.15ST_PolygonFromText . . . 357
7.8.1.16ST_WKTToSQL . . . 358
7.8.2 已知的二☒制文件 (WKB) . . . 359
7.8.2.1 ST_GeogFromWKB . . . 359
7.8.2.2 ST_GeomFromEWKB . . . 360
7.8.2.3 ST_GeomFromWKB . . . 362
7.8.2.4 ST_LineFromWKB . . . 364

7.8.2.5 ST_LinestringFromWKB . . . 365
7.8.2.6 ST_PointFromWKB . . . 367
7.8.2.7 ST_WKBToSQL . . . 368
7.8.3 其它格式 . . . 369
7.8.3.1 ST_Box2dFromGeoHash . . . 369
7.8.3.2 ST_GeomFromGeoHash . . . 370
7.8.3.3 ST_GeomFromGML . . . 371
7.8.3.4 ST_GeomFromGeoJSON . . . 376
7.8.3.5 ST_GeomFromKML . . . 378
7.8.3.6 ST_GeomFromTWKB . . . 379
7.8.3.7 ST_GMLToSQL . . . 381
7.8.3.8 ST_LineFromEncodedPolyline . . . 381
7.8.3.9 ST_PointFromGeoHash . . . 383
7.8.3.10 ST_FromFlatGeobufToTable . . . 385
7.8.3.11 ST_FromFlatGeobuf . . . 386
7.9 几何⌧出 . . . 386
7.9.1 已知文本 (WKT) . . . 386
7.9.1.1 ST_AsEWKT . . . 386
7.9.1.2 ST_AsText . . . 388
7.9.2 已知的二⌧制文件 (WKB) . . . 390
7.9.2.1 ST_AsBinary . . . 390
7.9.2.2 ST_AsEWKB . . . 392
7.9.2.3 ST_AsHEXEWKB . . . 393
7.9.3 其它格式 . . . 394
7.9.3.1 ST_AsEncodedPolyline . . . 394
7.9.3.2 ST_AsFlatGeobuf . . . 396
7.9.3.3 ST_AsGeobuf . . . 396
7.9.3.4 ST_AsGeoJSON . . . 397
7.9.3.5 ST_AsGML . . . 399
7.9.3.6 ST_AsKML . . . 402
7.9.3.7 ST_AsLatLonText . . . 404
7.9.3.8 ST_AsMARC21 . . . 406
7.9.3.9 ST_AsMVTGeom . . . 408
7.9.3.10 ST_AsMVT . . . 410
7.9.3.11 ST_AsSVG . . . 411
7.9.3.12 ST_AsTWKB . . . 414
7.9.3.13 ST_AsX3D . . . 415
7.9.3.14 ST_GeoHash . . . 419
7.10⌧算符 . . . 421

7.10.1 界框算符 . . . 421
7.10.1.1 && . . . 421
7.10.1.2 &&(geometry,box2df) . . . 422
7.10.1.3 &&(box2df,geometry) . . . 423
7.10.1.4 &&(box2df,box2df) . . . 424
7.10.1.5 &&& . . . 425
7.10.1.6 &&&(geometry,gidx) . . . 427
7.10.1.7 &&&(gidx,geometry) . . . 428
7.10.1.8 &&&(gidx,gidx) . . . 429
7.10.1.9 &< . . . 430
7.10.1.10 &<| . . . 431
7.10.1.11 &> . . . 432
7.10.1.12 <& . . . 434
7.10.1.13 <&| . . . 435
7.10.1.14 << . . . 436
7.10.1.15 >> . . . 439
7.10.1.16 @ . . . 441
7.10.1.17 @(geometry,box2df) . . . 442
7.10.1.18 @(box2df,geometry) . . . 443
7.10.1.19 @(box2df,box2df) . . . 444
7.10.1.20 |&> . . . 444
7.10.1.21 |>> . . . 446
7.10.1.22 ~ . . . 447
7.10.1.23 ~(geometry,box2df) . . . 449
7.10.1.24 ~(box2df,geometry) . . . 449
7.10.1.25 ~(box2df,box2df) . . . 450
7.10.1.26 ~= . . . 451
7.10.2 距离算符 . . . 452
7.10.2.1 <-> . . . 452
7.10.2.2 |=| . . . 454
7.10.2.3 <#> . . . 455
7.10.2.4 <<->> . . . 456
7.11 空关系 . . . 457
7.11.1 拓扑关系 . . . 457
7.11.1.1 ST_3DIntersects . . . 457
7.11.1.2 ST_Contains . . . 459
7.11.1.3 ST_ContainsProperly . . . 470
7.11.1.4 ST_CoveredBy . . . 473
7.11.1.5 ST_Covers . . . 475

7.11.1.6ST_Crosses . . . 477
7.11.1.7ST_Disjoint . . . 481
7.11.1.8ST_Equals . . . 483
7.11.1.9ST_Intersects . . . 485
7.11.1.10ST_LineCrossingDirection . . . 488
7.11.1.11ST_OrderingEquals . . . 491
7.11.1.12ST_Overlaps . . . 494
7.11.1.13ST_Relate . . . 500
7.11.1.14ST_RelateMatch . . . 503
7.11.1.15ST_Touches . . . 505
7.11.1.16ST_Within . . . 512
7.11.2距离关系 . . . 515
7.11.2.1ST_3DDWithin . . . 515
7.11.2.2ST_3DDFullyWithin . . . 516
7.11.2.3ST_DFullyWithin . . . 518
7.11.2.4ST_DWithin . . . 519
7.11.2.5ST_PointInsideCircle . . . 522
7.12⌧量函数 . . . 523
7.12.1ST_Area . . . 523
7.12.2ST_Azimuth . . . 526
7.12.3ST_Angle . . . 529
7.12.4ST_ClosestPoint . . . 531
7.12.5ST_3DClosestPoint . . . 533
7.12.6ST_Distance . . . 536
7.12.7ST_3DDistance . . . 538
7.12.8ST_DistanceSphere . . . 540
7.12.9ST_DistanceSpheroid . . . 541
7.12.10ST_FrechetDistance . . . 543
7.12.11ST_HausdorffDistance . . . 545
7.12.12ST_Length . . . 547
7.12.13ST_Length2D . . . 550
7.12.14ST_3DLength . . . 550
7.12.15ST_LengthSpheroid . . . 552
7.12.16ST_LongestLine . . . 554
7.12.17ST_3DLongestLine . . . 557
7.12.18ST_MaxDistance . . . 560
7.12.19ST_3DMaxDistance . . . 563
7.12.20ST_MinimumClearance . . . 564
7.12.21ST_MinimumClearanceLine . . . 565

7.12.23 ST_Perimeter . . . 566
7.12.24 ST_Perimeter2D . . . 570
7.12.25 ST_3DPerimeter . . . 571
7.12.26 ST_ShortestLine . . . 572
7.12.27 ST_3DShortestLine . . . 574
7.13 叠加函数 . . . 577
7.13.1 ST_ClipByBox2D . . . 577
7.13.2 ST_Difference . . . 578
7.13.3 ST_Intersection . . . 580
7.13.4 ST_MemUnion . . . 584
7.13.5 ST_Node . . . 584
7.13.6 ST_Split . . . 586
7.13.7 ST_Subdivide . . . 589
7.13.8 ST_SymDifference . . . 592
7.13.9 ST_UnaryUnion . . . 594
7.13.10 ST_Union . . . 595
7.14 几何处理 . . . 600
7.14.1 ST_Buffer . . . 600
7.14.2 ST_BuildArea . . . 609
7.14.3 ST_Centroid . . . 612
7.14.4 ST_ChaikinSmoothing . . . 617
7.14.5 ST_CatmullRomSmoothing . . . 621
7.14.6 ST_ConcaveHull . . . 623
7.14.7 ST_ConvexHull . . . 627
7.14.8 ST_DelaunayTriangles . . . 629
7.14.9 ST_FilterByM . . . 632
7.14.10 ST_GeneratePoints . . . 634
7.14.11 ST_GeometricMedian . . . 636
7.14.12 ST_LineMerge . . . 637
7.14.13 ST_MaximumInscribedCircle . . . 642
7.14.14 ST_LargestEmptyCircle . . . 644
7.14.15 ST_MinimumBoundingCircle . . . 646
7.14.16 ST_MinimumBoundingRadius . . . 648
7.14.17 ST_OrientedEnvelope . . . 649
7.14.18 ST_OffsetCurve . . . 650
7.14.19 ST_PointOnSurface . . . 657
7.14.20 ST_Polygonize . . . 662
7.14.21 ST_ReducePrecision . . . 665
7.14.22 ST_SharedPaths . . . 669

7.14.23 ST_Simplify . . . 671
7.14.24 ST_SimplifyPreserveTopology . . . 674
7.14.25 ST_SimplifyPolygonHull . . . 676
7.14.26 ST_SimplifyVW . . . 680
7.14.27 ST_SetEffectiveArea . . . 683
7.14.28 ST_TriangulatePolygon . . . 685
7.14.29 ST_VoronoiLines . . . 688
7.14.30 ST_VoronoiPolygons . . . 689
7.15 覆盖范□ . . . 691
7.15.1 ST_CoverageInvalidEdges . . . 691
7.15.2 ST_CoverageClean . . . 693
7.15.3 ST_CoverageSimplify . . . 694
7.15.4 ST_CoverageEdges . . . 696
7.15.5 ST_CoverageUnion . . . 697
7.16 仿射□□ . . . 699
7.16.1 ST_Affine . . . 699
7.16.2 ST_Rotate . . . 701
7.16.3 ST_RotateX . . . 704
7.16.4 ST_RotateY . . . 705
7.16.5 ST_RotateZ . . . 706
7.16.6 ST_Scale . . . 708
7.16.7 ST_Translate . . . 712
7.16.8 ST_TransScale . . . 715
7.17 聚□函数 . . . 717
7.17.1 ST_ClusterDBSCAN . . . 717
7.17.2 ST_ClusterIntersecting . . . 719
7.17.3 ST_ClusterIntersectingWin . . . 720
7.17.4 ST_ClusterRelateWin . . . 722
7.17.5 ST_MinimumSpanningTree . . . 723
7.17.6 ST_ClusterKMeans . . . 725
7.17.7 ST_ClusterWithin . . . 729
7.17.8 ST_ClusterWithinWin . . . 730
7.18 □界框函数 . . . 732
7.18.1 Box2D . . . 732
7.18.2 Box3D . . . 734
7.18.3 ST_EstimatedExtent . . . 736
7.18.4 ST_Expand . . . 737
7.18.5 ST_Extent . . . 739
7.18.6 ST_3DExtent . . . 741

7.18.7 ST_MakeBox2D . . . 742
7.18.8 ST_3DMakeBox . . . 743
7.18.9 ST_XMax . . . 743
7.18.10 ST_XMin . . . 746
7.18.11 ST_YMax . . . 748
7.18.12 ST_YMin . . . 750
7.18.13 ST_ZMax . . . 753
7.18.14 ST_ZMin . . . 755
7.18.15 ST_XSize . . . 758
7.18.16 ST_YSize . . . 759
7.18.17 ST_ZSize . . . 760
7.18.18 ST_MMin . . . 762
7.18.19 ST_MMax . . . 762
7.18.20 ST_MSize . . . 763
7.19 区性参考 . . . 764
7.19.1 ST_LineInterpolatePoint . . . 764
7.19.2 ST_3DLineInterpolatePoint . . . 768
7.19.3 ST_LineInterpolatePoints . . . 769
7.19.4 ST_LineLocatePoint . . . 771
7.19.5 ST_LineSubstring . . . 774
7.19.6 ST_LocateAlong . . . 780
7.19.7 ST_LocateBetween . . . 781
7.19.8 ST_LocateBetweenElevations . . . 783
7.19.9 ST_InterpolatePoint . . . 785
7.19.10 ST_3DInterpolatePoint . . . 786
7.19.11 ST_AddMeasure . . . 788
7.20 区迹函数 . . . 791
7.20.1 ST_IsValidTrajectory . . . 791
7.20.2 ST_ClosestPointOfApproach . . . 792
7.20.3 ST_DistanceCPA . . . 794
7.20.4 ST_CPAWithin . . . 795
7.21 版本函数 . . . 796
7.21.1 PostGIS_Extensions_Upgrade . . . 796
7.21.2 PostGIS_Full_Version . . . 797
7.21.3 PostGIS_GEOS_Version . . . 798
7.21.4 PostGIS_GEOS_Compiled_Version . . . 798
7.21.5 PostGIS_Liblwgeom_Version . . . 799
7.21.6 PostGIS_LibXML_Version . . . 800
7.21.7 PostGIS_LibJSON_Version . . . 800

7.21.8 PostGIS_Lib_Build_Date . . . 801
7.21.9 PostGIS_Lib_Version . . . 801
7.21.10 PostGIS_PROJ_Version . . . 802
7.21.11 PostGIS_PROJ_Compiled_Version . . . 802
7.21.12 PostGIS_Wagyu_Version . . . 803
7.21.13 PostGIS_Scripts_Build_Date . . . 804
7.21.14 PostGIS_Scripts_Installed . . . 804
7.21.15 PostGIS_Scripts_Released . . . 805
7.21.16 PostGIS_Version . . . 806
7.22 大一□自定□□量 (GUCs) . . . 806
7.22.1 postgis.gdal_datapath . . . 806
7.22.2 postgis.gdal_enabled_drivers . . . 807
7.22.3 postgis.enable_outdb_rasters . . . 808
7.22.4 postgis.gdal_vsi_options . . . 810
7.22.5 postgis.gdal_cpl_debug . . . 810
7.23 故障排除函数 . . . 810
7.23.1 PostGIS_AddBBox . . . 810
7.23.2 PostGIS_DropBBox . . . 811
7.23.3 PostGIS_HasBBox . . . 812

8 SFCGAL 函数参考 814
8.1 SFCGAL 管理函数 . . . 814
8.1.1 postgis_sfcgal_version . . . 814
8.1.2 postgis_sfcgal_full_version . . . 814
8.2 SFCGAL □□器和□置器 . . . 815
8.2.1 CG_ForceLHR . . . 815
8.2.2 CG_IsPlanar . . . 816
8.2.3 CG_IsSolid . . . 816
8.2.4 CG_MakeSolid . . . 818
8.2.5 CG_Orientation . . . 818
8.2.6 CG_Area . . . 819
8.2.7 CG_3DArea . . . 820
8.2.8 CG_Volume . . . 821
8.3 SFCGAL □理和关系函数 . . . 822
8.3.1 CG_Intersection . . . 822
8.3.2 CG_Intersects . . . 824
8.3.3 CG_3DIntersects . . . 826
8.3.4 CG_Difference . . . 827
8.3.5 CG_3DDifference . . . 828

8.3.6 CG_Distance . . . 830
8.3.7 CG_3DDistance . . . 831
8.3.8 CG_3DConvexHull . . . 832
8.3.9 CG_3DIntersection . . . 835
8.3.10 CG_Union . . . 838
8.3.11 CG_3DUnion . . . 839
8.3.12 CG_AlphaShape . . . 841
8.3.13 CG_ApproxConvexPartition . . . 844
8.3.14 CG_ApproximateMedialAxis . . . 845
8.3.15 CG_GenerateFlatRoof . . . 848
8.3.16 CG_GenerateHippedRoof . . . 849
8.3.17 CG_GenerateGableRoof . . . 850
8.3.18 CG_GenerateSkillionRoof . . . 851
8.3.19 CG_GenerateRoof . . . 853
8.3.20 CG_PolygonRepair . . . 854
8.3.21 CG_ConstrainedDelaunayTriangles . . . 857
8.3.22 CG_Extrude . . . 859
8.3.23 CG_ExtrudeStraightSkeleton . . . 861
8.3.24 CG_GreeneApproxConvexPartition . . . 863
8.3.25 CG_MinkowskiSum . . . 864
8.3.26 CG_OptimalAlphaShape . . . 866
8.3.27 CG_OptimalConvexPartition . . . 868
8.3.28 CG_StraightSkeleton . . . 870
8.3.29 CG_Tesselate . . . 872
8.3.30 CG_Triangulate . . . 874
8.3.31 CG_Visibility . . . 876
8.3.32 CG_YMonotonePartition . . . 878
8.3.33 CG_StraightSkeletonPartition . . . 879
8.3.34 CG_3DBuffer . . . 881
8.3.35 CG_Rotate . . . 884
8.3.36 CG_2DRotate . . . 885
8.3.37 CG_3DRotate . . . 886
8.3.38 CG_RotateX . . . 887
8.3.39 CG_RotateY . . . 888
8.3.40 CG_RotateZ . . . 889
8.3.41 CG_Scale . . . 890
8.3.42 CG_3DScale . . . 891
8.3.43 CG_3DScaleAroundCenter . . . 892
8.3.44 CG_Translate . . . 893

8.3.45CG_3DTranslate . . . 894
8.3.46CG_Simplify . . . 895
8.3.47CG_3DAlphaWrapping . . . 900
8.3.48CG_NurbsCurveInterpolate . . . 901
8.3.49CG_NurbsCurveApproximate . . . 903
8.3.50CG_NurbsCurveDerivative . . . 906
8.4 Deprecated SFCGAL Function Aliases . . . 909
8.4.1 ST_3DIntersection . . . 909
8.4.2 ST_3DDifference . . . 910
8.4.3 ST_3DUnion . . . 910
8.4.4 ST_Tesselate . . . 910
8.4.5 ST_3DArea . . . 911
8.4.6 ST_Extrude . . . 911
8.4.7 ST_ForceLHR . . . 912
8.4.8 ST_Orientation . . . 912
8.4.9 ST_MinkowskiSum . . . 913
8.4.10ST_StraightSkeleton . . . 913
8.4.11ST_ApproximateMedialAxis . . . 913
8.4.12ST_IsPlanar . . . 914
8.4.13ST_Volume . . . 914
8.4.14ST_MakeSolid . . . 915
8.4.15ST_IsSolid . . . 915
8.4.16ST_ConstrainedDelaunayTriangles . . . 916
8.4.17ST_3DConvexHull . . . 916
8.4.18ST_AlphaShape . . . 916
8.4.19ST_OptimalAlphaShape . . . 917

9 拓扑⊠⊠ 918
9.1 Building Faces from Linework . . . 918
9.2 拓扑基⊠表 . . . 920
9.3 拓扑⊠型 . . . 920
9.3.1 getfaceedges_returntype . . . 920
9.3.2 TopoGeometry . . . 921
9.3.3 validatetopology_returntype . . . 921
9.4 拓扑域 . . . 922
9.4.1 TopoElement . . . 922
9.4.2 TopoElementArray . . . 923
9.5 拓扑和拓扑几何管理 . . . 924
9.5.1 AddTopoGeometryColumn . . . 924

9.5.2 RenameTopoGeometryColumn . . . 925
9.5.3 DropTopology . . . 925
9.5.4 RenameTopology . . . 926
9.5.5 DropTopoGeometryColumn . . . 927
9.5.6 FixCorruptTopoGeometryColumn . . . 927
9.5.7 Populate_Topology_Layer . . . 928
9.5.8 TopologySummary . . . 929
9.5.9 ValidateTopology . . . 930
9.5.10 ValidateTopologyRelation . . . 932
9.5.11 ValidateTopologyPrecision . . . 932
9.5.12 MakeTopologyPrecise . . . 933
9.5.13 FindTopology . . . 934
9.5.14 FindLayer . . . 935
9.5.15 TotalTopologySize . . . 935
9.5.16 UpgradeTopology . . . 936
9.6 拓扑□□管理 . . . 937
9.7 拓扑□造器 . . . 937
9.7.1 CreateTopology . . . 937
9.7.2 CopyTopology . . . 938
9.7.3 ST_InitTopoGeo . . . 939
9.7.4 ST_CreateTopoGeo . . . 939
9.7.5 TopoGeo_AddPoint . . . 940
9.7.6 TopoGeo_AddLineString . . . 941
9.7.7 TopoGeo_AddPolygon . . . 941
9.7.8 TopoGeo_LoadGeometry . . . 942
9.8 拓扑□□器 . . . 942
9.8.1 ST_AddIsoNode . . . 942
9.8.2 ST_AddIsoEdge . . . 943
9.8.3 ST_AddEdgeNewFaces . . . 944
9.8.4 ST_AddEdgeModFace . . . 944
9.8.5 ST_RemEdgeNewFace . . . 945
9.8.6 ST_RemEdgeModFace . . . 946
9.8.7 ST_ChangeEdgeGeom . . . 946
9.8.8 ST_ModEdgeSplit . . . 947
9.8.9 ST_ModEdgeHeal . . . 948
9.8.10 ST_NewEdgeHeal . . . 948
9.8.11 ST_MoveIsoNode . . . 949
9.8.12 ST_NewEdgesSplit . . . 950
9.8.13 ST_RemoveIsoNode . . . 951

9.8.14ST_RemoveIsoEdge . . . 951
9.9 拓扑⌧⌧器 . . . 952
9.9.1 GetEdgeByPoint . . . 952
9.9.2 GetFaceByPoint . . . 953
9.9.3 GetFaceContainingPoint . . . 954
9.9.4 GetNodeByPoint . . . 955
9.9.5 GetTopologyID . . . 956
9.9.6 GetTopologySRID . . . 956
9.9.7 GetTopologyName . . . 957
9.9.8 ST_GetFaceEdges . . . 957
9.9.9 ST_GetFaceGeometry . . . 958
9.9.10GetRingEdges . . . 959
9.9.11GetNodeEdges . . . 960
9.9.12FindVertexSegmentPairsBelowDistance . . . 961
9.10拓扑⌧理 . . . 961
9.10.1Polygonize . . . 961
9.10.2AddNode . . . 962
9.10.3AddEdge . . . 962
9.10.4AddFace . . . 964
9.10.5ST_Simplify . . . 965
9.10.6RemoveUnusedPrimitives . . . 966
9.11拓扑几何⌧造函数 . . . 966
9.11.1CreateTopoGeom . . . 966
9.11.2toTopoGeom . . . 969
9.11.3TopoElementArray_Agg . . . 971
9.11.4TopoElement . . . 971
9.12拓扑几何⌧⌧器 . . . 972
9.12.1clearTopoGeom . . . 972
9.12.2TopoGeom_addElement . . . 972
9.12.3TopoGeom_remElement . . . 973
9.12.4TopoGeom_addTopoGeom . . . 974
9.12.5toTopoGeom . . . 974
9.13拓扑几何⌧⌧器 . . . 974
9.13.1GetTopoGeomElementArray . . . 974
9.13.2GetTopoGeomElements . . . 975
9.13.3ST_SRID . . . 975
9.14拓扑几何⌧出 . . . 976
9.14.1AsGML . . . 976
9.14.2AsTopoJSON . . . 978

9.15拓扑空⌧关系 980
9.15.1Equals 980
9.15.2Intersects 981
9.16⌧入和⌧出拓扑 982
9.16.1使用拓扑⌧出器 982
9.16.2使用拓扑⌧入器 982

10⌧格数据管理、⌧⌧和⌧用程序 **983**
10.1加⌧和⌧建⌧格 983
10.1.1使用 raster2pgsql 加⌧⌧格 983
10.1.1.1用法示例 983
10.1.1.2raster2pgsql ⌧⌧ 984
10.1.2使用 PostGIS ⌧格函数⌧建⌧格 986
10.1.3使用“out db”云⌧格 986
10.2Raster Coverages and Raster/Vector Analysis 987
10.3⌧格目⌧ 989
10.3.1⌧格列目⌧ 989
10.3.2⌧格概述 990
10.4使用 PostGIS Raster ⌧建自定⌧⌧用程序 990
10.4.1PHP 示例使用 ST_AsPNG 与其他⌧格函数⌧行⌧出 991
10.4.2ASP.NET C# 示例使用 ST_AsPNG 与其他⌧格函数⌧行⌧出 991
10.4.3将⌧格⌧⌧⌧出⌧⌧像文件的 Java 控制台⌧用程序 993
10.4.4使用 PLPython 通⌧ SQL ⌧⌧⌧像 995
10.4.5使用 PSQL ⌧出⌧格 995

11⌧格参考 **997**
11.1⌧格支持数据⌧型 998
11.1.1geomval 998
11.1.2addbandarg 998
11.1.3rastbandarg 998
11.1.4raster 999
11.1.5reclassarg 999
11.1.6summarystats 1000
11.1.7unionarg 1001
11.2⌧格管理 1001
11.2.1AddRasterConstraints 1001
11.2.2DropRasterConstraints 1003
11.2.3AddOverviewConstraints 1004
11.2.4DropOverviewConstraints 1005

11.2.5 PostGIS_GDAL_Version . . . 1006
11.2.6 PostGIS_Raster_Lib_Build_Date . . . 1006
11.2.7 PostGIS_Raster_Lib_Version . . . 1007
11.2.8 ST_GDALDrivers . . . 1007
11.2.9 UpdateRasterSRID . . . 1010
11.2.10 ST_CreateOverview . . . 1011
11.3 栅格构造器 . . . 1012
11.3.1 ST_AddBand . . . 1012
11.3.2 ST_AsRaster . . . 1014
11.3.3 ST_AsRasterAgg . . . 1017
11.3.4 ST_Band . . . 1018
11.3.5 ST_MakeEmptyCoverage . . . 1022
11.3.6 ST_MakeEmptyRaster . . . 1023
11.3.7 ST_Tile . . . 1025
11.3.8 ST_Retile . . . 1027
11.3.9 ST_FromGDALRaster . . . 1027
11.4 栅格访问器 . . . 1028
11.4.1 ST_GeoReference . . . 1028
11.4.2 ST_Height . . . 1029
11.4.3 ST_IsEmpty . . . 1030
11.4.4 ST_MemSize . . . 1031
11.4.5 ST_MetaData . . . 1032
11.4.6 ST_NumBands . . . 1032
11.4.7 ST_PixelHeight . . . 1033
11.4.8 ST_PixelWidth . . . 1034
11.4.9 ST_ScaleX . . . 1036
11.4.10 ST_ScaleY . . . 1037
11.4.11 ST_RasterToWorldCoord . . . 1037
11.4.12 ST_RasterToWorldCoordX . . . 1038
11.4.13 ST_RasterToWorldCoordY . . . 1039
11.4.14 ST_Rotation . . . 1040
11.4.15 ST_SkewX . . . 1041
11.4.16 ST_SkewY . . . 1042
11.4.17 ST_SRID . . . 1043
11.4.18 ST_Summary . . . 1044
11.4.19 ST_UpperLeftX . . . 1045
11.4.20 ST_UpperLeftY . . . 1045
11.4.21 ST_Width . . . 1046
11.4.22 ST_WorldToRasterCoord . . . 1046

11.4.23 ST_WorldToRasterCoordX . . . 1047
11.4.24 ST_WorldToRasterCoordY . . . 1048
11.5 栅格波段栅栅器 . . . 1049
11.5.1 ST_BandMetaData . . . 1049
11.5.2 ST_BandNoDataValue . . . 1050
11.5.3 ST_BandIsNoData . . . 1051
11.5.4 ST_BandPath . . . 1052
11.5.5 ST_BandFileSize . . . 1053
11.5.6 ST_BandFileTimestamp . . . 1054
11.5.7 ST_BandPixelType . . . 1054
11.5.8 ST_MinPossibleValue . . . 1055
11.5.9 ST_HasNoBand . . . 1056
11.6 栅格像素栅器和栅置器 . . . 1057
11.6.1 ST_PixelAsPolygon . . . 1057
11.6.2 ST_PixelAsPolygons . . . 1058
11.6.3 ST_PixelAsPoint . . . 1059
11.6.4 ST_PixelAsPoints . . . 1060
11.6.5 ST_PixelAsCentroid . . . 1062
11.6.6 ST_PixelAsCentroids . . . 1063
11.6.7 ST_Value . . . 1065
11.6.8 ST_NearestValue . . . 1068
11.6.9 ST_SetZ . . . 1070
11.6.10 ST_SetM . . . 1072
11.6.11 ST_Neighborhood . . . 1074
11.6.12 ST_SetValue . . . 1076
11.6.13 ST_SetValues . . . 1077
11.6.14 ST_DumpValues . . . 1085
11.6.15 ST_PixelOfValue . . . 1086
11.7 栅格栅器 . . . 1087
11.7.1 ST_SetGeoReference . . . 1087
11.7.2 ST_SetRotation . . . 1088
11.7.3 ST_SetScale . . . 1089
11.7.4 ST_SetSkew . . . 1090
11.7.5 ST_SetSRID . . . 1092
11.7.6 ST_SetUpperLeft . . . 1093
11.7.7 ST_Resample . . . 1093
11.7.8 ST_Rescale . . . 1095
11.7.9 ST_Reskew . . . 1096
11.7.10 ST_SnapToGrid . . . 1098

11.7.1 ST_Resize . . . 1099
11.7.2 ST_Transform . . . 1100
11.8 栅格波段操作器 . . . 1104
11.8.1 ST_SetBandNoDataValue . . . 1104
11.8.2 ST_SetBandIsNoData . . . 1105
11.8.3 ST_SetBandPath . . . 1107
11.8.4 ST_SetBandIndex . . . 1108
11.9 栅格波段统计和分析 . . . 1110
11.9.1 ST_Count . . . 1110
11.9.2 ST_CountAgg . . . 1111
11.9.3 ST_Histogram . . . 1112
11.9.4 ST_Quantile . . . 1113
11.9.5 ST_SummaryStats . . . 1115
11.9.6 ST_SummaryStatsAgg . . . 1118
11.9.7 ST_ValueCount . . . 1119
11.10 栅格输入 . . . 1122
11.10.1 ST_RastFromWKB . . . 1122
11.10.2 ST_RastFromHexWKB . . . 1123
11.11 栅格输出 . . . 1123
11.11.1 ST_AsBinary/ST_AsWKB . . . 1123
11.11.2 ST_AsHexWKB . . . 1124
11.11.3 ST_AsGDALRaster . . . 1125
11.11.4 ST_AsJPEG . . . 1126
11.11.5 ST_AsPNG . . . 1127
11.11.6 ST_AsTIFF . . . 1128
11.12 栅格处理：地图代数 . . . 1129
11.12.1 ST_Clip . . . 1129
11.12.2 ST_ColorMap . . . 1134
11.12.3 ST_Grayscale . . . 1137
11.12.4 ST_Intersection . . . 1141
11.12.5 ST_MapAlgebra (callback function version) . . . 1144
11.12.6 ST_MapAlgebra (expression version) . . . 1151
11.12.7 ST_MapAlgebraExpr . . . 1153
11.12.8 ST_MapAlgebraExpr . . . 1157
11.12.9 ST_MapAlgebraFct . . . 1164
11.12.10 ST_MapAlgebraFct . . . 1168
11.12.11 ST_MapAlgebraFctNgb . . . 1174
11.12.12 ST_Reclass . . . 1178
11.12.13 ST_ReclassExact . . . 1180

11.12.4 ST_Union . . . 1181
11.13 内置地图代数回调函数 . . . 1183
11.13.1 ST_Distinct4ma . . . 1183
11.13.2 ST_InvDistWeight4ma . . . 1184
11.13.3 ST_Max4ma . . . 1185
11.13.4 ST_Mean4ma . . . 1186
11.13.5 ST_Min4ma . . . 1187
11.13.6 ST_MinDist4ma . . . 1189
11.13.7 ST_Range4ma . . . 1190
11.13.8 ST_StdDev4ma . . . 1191
11.13.9 ST_Sum4ma . . . 1192
11.14 栅格处理：DEM（高程） . . . 1193
11.14.1 ST_Aspect . . . 1193
11.14.2 ST_HillShade . . . 1196
11.14.3 ST_Roughness . . . 1198
11.14.4 ST_Slope . . . 1198
11.14.5 ST_TPI . . . 1200
11.14.6 ST_TRI . . . 1201
11.14.7 ST_InterpolateRaster . . . 1202
11.14.8 ST_Contour . . . 1203
11.15 栅格处理：栅格到几何 . . . 1204
11.15.1 Box3D . . . 1204
11.15.2 ST_ConvexHull . . . 1205
11.15.3 ST_DumpAsPolygons . . . 1206
11.15.4 ST_Envelope . . . 1207
11.15.5 ST_MinConvexHull . . . 1208
11.15.6 ST_Polygon . . . 1210
11.15.7 ST_IntersectionFractions . . . 1211
11.16 栅格运算符 . . . 1212
11.16.1 && . . . 1212
11.16.2 &< . . . 1213
11.16.3 &> . . . 1214
11.16.4 = . . . 1214
11.16.5 @ . . . 1215
11.16.6 ~= . . . 1215
11.16.7 ~ . . . 1216
11.17 栅格和栅格波段空间关系 . . . 1217
11.17.1 ST_Contains . . . 1217
11.17.2 ST_ContainsProperly . . . 1218

11.17.3 ST_Covers . . . 1219
11.17.4 ST_CoveredBy . . . 1220
11.17.5 ST_Disjoint . . . 1221
11.17.6 ST_Intersects . . . 1222
11.17.7 ST_Overlaps . . . 1223
11.17.8 ST_Touches . . . 1224
11.17.9 ST_SameAlignment . . . 1225
11.17.10 ST_NotSameAlignmentReason . . . 1226
11.17.11 ST_Within . . . 1227
11.17.12 ST_DWithin . . . 1228
11.17.13 ST_DFullyWithin . . . 1229
11.18 栅格提示 . . . 1230
11.18.1 Out-DB 栅格 . . . 1230
11.18.1.1 包含多文件的目录 . . . 1230
11.18.1.2 最大打开文件数 . . . 1230
11.18.1.2.1 整个系统最大打开文件数 . . . 1231
11.18.1.2.2 每个进程的最大打开文件数 . . . 1231

12 PostGIS 扩充 **1234**

13 PostGIS 特殊函数索引 **1235**
13.1 PostGIS 聚合函数 . . . 1235
13.2 PostGIS 窗口函数 . . . 1236
13.3 PostGIS SQL-MM 兼容函数 . . . 1236
13.4 PostGIS 地理支持函数 . . . 1240
13.5 PostGIS 栅格支持函数 . . . 1241
13.6 PostGIS 几何/地理/栅格输出函数 . . . 1246
13.7 PostGIS 盒界框函数 . . . 1246
13.8 支持 3D 的 PostGIS 函数 . . . 1248
13.9 PostGIS 曲线几何支持函数 . . . 1253
13.10 PostGIS 多面体曲面支持函数 . . . 1256
13.11 PostGIS 函数支持矩阵 . . . 1259
13.12 新的、增强的或更改的 PostGIS 函数 . . . 1278
13.12.1 PostGIS 新增功能或增强功能（3.7） . . . 1278
13.12.2 PostGIS 新增功能或增强功能（3.6） . . . 1280
13.12.3 PostGIS 新增功能或增强功能（3.5） . . . 1281
13.12.4 PostGIS 新增功能或增强功能（3.4） . . . 1283
13.12.5 PostGIS 新增功能或增强功能（3.3） . . . 1284
13.12.6 PostGIS 新增功能或增强功能（3.2） . . . 1284

13.12.1 PostGIS 新增功能或增⌧功能（3.1） 1286
13.12.2 PostGIS 新增功能或增⌧功能（3.0） 1287
13.12.3 PostGIS 新增功能或增⌧功能（2.5） 1288
13.12.4 PostGIS 新增功能或增⌧功能（2.4） 1290
13.12.5 PostGIS 新增功能或增⌧功能（2.3） 1290
13.12.6 PostGIS 新增功能或增⌧功能（2.2） 1292
13.12.7 PostGIS 新增功能或增⌧功能（2.1） 1294
13.12.8 PostGIS 新增功能或增⌧功能（2.0） 1298
13.12.9 PostGIS 新增功能或增⌧功能（1.5） 1307
13.12.10 PostGIS 新增功能或增⌧功能（1.4） 1308
13.12.11 PostGIS 新增功能或增⌧功能（1.3） 1308

14 ⌧告⌧⌧ **1309**
14.1 ⌧告⌧件⌧⌧ 1309
14.2 ⌧告文档⌧⌧ 1309

15 Credits and Acknowledgments **1310**
15.1 Project Steering Committee 1310
15.2 Core Contributors Present 1310
15.3 Core Contributors Past 1311
15.4 PostGIS Raster Origins 1311
15.5 Other Contributors 1312

A ⌧行⌧明 **1316**
A.1 PostGIS 3.7.0 1316
A.1.1 重大⌧化 1316
A.1.2 新特性 1316
A.1.3 增⌧功能 1317
A.1.4 Bug Fixes 1318

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 3.7.0rc3



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☒

1.1 What Is PostGIS?

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

The main `postgis` extension adds:

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

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

1.2 What Is in the PostGIS Distribution?

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

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

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

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

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

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

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

1.3 The PostGIS Project Family

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

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

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

1.4 How to Use This Manual

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

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

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

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

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

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

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

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

1.5 Standards Support

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

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

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

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

1.6 Project and Community

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

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

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

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⊠⊠：

Code

```
tar -xvzf postgis-3.7.0rc3.tar.gz
cd postgis-3.7.0rc3
./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⊠⊠。
This section includes general compilation instructions. If you are compiling for Windows or another operating system, see the PostGIS getting started guides and the development environment guides.
Pre-Built Packages for various OS are listed in the PostGIS getting started guides
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 后台服⊠器的⊠展。在 PostGIS 3.7.0rc3 中，⊠了⊠⊠，必⊠提供完整的 PostgreSQL 服⊠器⊠文件。它可以在 PostgreSQL 14 - 19 版本之⊠⊠建。旧版本的 PostgreSQL 不再支持。

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++ ⊠：

Code

```
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://download.osgeo.org/postgis/source/postgis-3.7.0rc3.tar.gz ⊠取源代⊠的存档

Code

```
wget https://download.osgeo.org/postgis/source/postgis-3.7.0rc3.tar.gz
tar -xvzf postgis-3.7.0rc3.tar.gz
cd postgis-3.7.0rc3
```

⊠将在当前工作目⊠中⊠建一个名⊠ `postgis-3.7.0rc3` 的目⊠。

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

Code

```
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。

Code

```
./configure
```

2.2.2 安装要求

要⊠建和使用 PostGIS，您需要：

不可缺少

- PostgreSQL 14 - 19. 需要完整安装 PostgreSQL（包括服⊠器⊠文件）。可从 https://www.postgresql.org ⊠取 18。

  For current release-line support and end-of-life policy, see https://postgis.net/development/versions_eol/-

- 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 投影☒☒☒。需要 Proj 6.1 及以上版本。Proj ☒用于 PostGIS 中提供坐☒重投影支持。Proj 可以从https://proj.org/ 下☒。
- GEOS geometry library, version 3.10.0 or greater, but GEOS 3.15+ is required to take full advantage of all the new functions and features. GEOS is available for download from https://libgeos.org .
- LibXML2. LibXML2 is currently used in some imports functions (ST_GeomFromGML and ST_GeomFromKl LibXML2 is available for download from https://gitlab.gnome.org/GNOME/libxml2/-/releases.
- Perl is required to build PostGIS. Perl is available for download from https://www.perl.org/.
- The iconv library is required by the shp2pgsql loader for encoding conversion. On GNU/Linux systems it is typically provided by libc; on other platforms you may need a separate libiconv package.
- The GNU Multiple Precision Arithmetic Library (libgmp). Used to control precision of math operations in PostGIS Topology. GMP is available for download from https://gmplib.org/. It is detected via **pkg-config** and has no fallback, so **pkg-config** must be available for a default build.
- GDAL, version 2.4 or higher, is required for raster support. GDAL 3+ is preferred. https://gdal.org/-download.html.
- 要使用 PostgreSQL+JIT ☒行☒☒，需要 LLVM 版本 6 或更高版本。☒参☒ https://trac.osgeo.org/postgis/-ticket/4125。

可☒☒的

- GDAL（☒可☒）☒当您不需要☒格☒才可以将其省略。☒☒确保☒用您要使用的☒☒程序，如 Section 3.2中所述。
- JSON-C. JSON-C is currently used to import GeoJSON via the function ST_GeomFromGeoJson. JSON-C is available for download from https://github.com/json-c/json-c/releases/. It is enabled by default but optional; if it is not found configure only warns and the build continues. Disable with `--without-json`.
- GNU gettext for loader and documentation translation support. It may be disabled with `--with-gettext=n` https://www.gnu.org/software/gettext/.
- GTK（需要 GTK+2.0, 2.8+）。用于☒☒ shp2pgsql-gui，一个 shapefile loader。它位于 http://www.gtk.org/ 。
- SFCGAL, 1.3.1 or higher is required, 2.3+ is needed to be able to use all functionality. SFCGAL can be used to provide additional 2D and 3D advanced analysis functions to PostGIS cf Chapter 8. https://sfcgal.gitlab.io/sfcgal/.
- 要☒用 ST_AsMVT，您需要 ProtoBuf-C ☒（☒行☒）和 ProtoC-C ☒☒器（☒建☒）。要☒☒ protobuf-c 的正确最低版本，需要 pkg-config。参☒protobuf-c。默☒情况下，PostGIS 使用 Wagyu 快速☒估 MVT 多☒形，但需要 C++11 ☒☒器。使用 CXXFLAGS 使用您用于 PostgreSQL 安装的相同☒☒器。如果要禁用此功能并改用 GEOS，☒在配置期☒指定它。`--without-wagyu`。
- CUnit（`CUnit`）。☒是回☒☒☒所需要的。http://cunit.sourceforge.net/
- DocBook (`xsltproc`) ☒建文档需要。Docbook 可从http://www.docbook.org/ ☒取。
- Python 3 is required, when configured with **--with-visual-examples** (off by default), to run the manual example tester that generates the visual examples which the HTML and PDF documentation embed. The examples are executed against a live PostgreSQL server running the staged PostGIS build via **make -C regress visual-examples**; no Python database driver is required, the tooling talks to the server through the **psql** client. Python 3 is available from https://www.python.org/ .
- DBLatex (`dblatex`) 需要以 PDF 格式☒建文档。它位于 http://dblatex.sourceforge.net/ 。
- PDF documentation needs an SVG rasterizer for its high-resolution figure fallbacks. The build prefers `rsvg-convert`, and can also use GraphicsMagick (`gm convert`) or ImageMagick (`magick` or `convert`).

2.2.3 ☒建配置

与大多数 Linux 安装一☒，第一步是生成一个将用于☒建源代☒的 Makefile。☒是通☒ shell 脚本完成的

Code

```
./configure
```

如果未提供任何参数，此命令将自☒☒找在系☒上☒建 PostGIS 源代☒所需的☒件和☒。**./configure** 是一种常☒的用法，但它接受一些参数，以防您在非☒准位置☒有必要的☒或程序。

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

--with-library-minor-version 从 PostGIS 3.0 开始，默☒生成的☒文件将不再将次要版本作☒文件名的一部分。☒意味着所有 PostGIS 3 ☒都将以 `postgis-3` ☒尾。☒☒做是☒了使 pg_upgrade 更容易，缺点是您只能在服☒器中安装一个版本的 PostGIS 3 系列。要☒取文件的旧行☒（包括次要版本）：例如 `postgis-3.0` 将此开关添加到您的配置☒句中。

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

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

--with-pgconfig=FILE PostgreSQL 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-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 不☒☒ topology 支持。

--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

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

---

Code

```
./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⊠⊠⊠

Code

```
make
```

Output

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

所有函数的注⊠都是根据文档自⊠生成的。如果你之后想把⊠些注⊠安装到自己的空⊠数据⊠里，需要⊠行依⊠ docbook 的命令来⊠入注⊠。不⊠，如果是从 tar.gz 包安装，doc 目⊠里已⊠包含了 postgis_comments.sql 等注⊠文件，就不需要再⊠外生成了；⊠外，通⊠ CREATE EXTENSION 安装⊠，⊠些注⊠也会一并包含在内。

Code

```
make comments
```

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

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

Code

```
make cheatsheets
```

2.2.5 ⊠建和部署 PostGIS ⊠展

当 PostgreSQL 提供⊠展支持⊠，PostGIS ⊠展会被自⊠⊠建并安装。

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：

Code

```
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。

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

Code

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



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

⊠展文件在同一版本的 PostGIS 中始⊠是相同的，无⊠操作系⊠如何。只要已⊠安装了 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⊠。

如果您想在与开⊠⊠境不同的⊠一台服⊠器上手⊠安装⊠展，除了通常需要的 PostGIS 二⊠制文件之外，⊠需要将以下文件从 extensions 文件⊠复制到 PostgreSQL 安装目⊠下的 PostgreSQL/share/extension 文件⊠中。

- 指定未提供⊠，控制文件postgis.control, postgis_topology.control ⊠示要安装的⊠展版本等信息。
- ⊠注意, ⊠⊠每个⊠展的 /sql 文件⊠中的所有文件（例如, extensions/postgis/sql/*.sql 和 extensions/postgis_topology/sql/*.sql）需要复制到 PostgreSQL 的 share/extension 文件⊠的最上⊠

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：

Code

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

Output

```
+------------------+-----------------+-------------------+
| name             | default_version | installed_version |
+------------------+-----------------+-------------------+
| postgis          | 3.7.0rc3        | 3.7.0rc3          |
| postgis_raster   | 3.7.0rc3        | 3.7.0rc3          |
| postgis_sfcgal   | 3.7.0rc3        |                   |
| postgis_topology | 3.7.0rc3        |                   |
+------------------+-----------------+-------------------+
```

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⊠：

Code

```
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION postgis_sfcgal;
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。

Code

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

Output

```
List of installed extensions
+-[ RECORD 1 ]-------------------------------------------+
| Name        | postgis                                  |
| Version     | 3.7.0rc3                                   |
| Schema      | public                                   |
| Description | PostGIS geometry, geography, and raster...  |
+-[ RECORD 2 ]-------------------------------------------+
| Name        | postgis_raster                           |
| Version     | 3.7.0rc3                                   |
| Schema      | public                                   |
| Description | PostGIS raster types and functions       |
+-[ RECORD 3 ]-------------------------------------------+
| Name        | postgis_topology                         |
| Version     | 3.7.0rc3                                   |
| Schema      | topology                                 |
| Description | PostGIS topology spatial types and functions |
+-------------------------------------------------------+
```



Warning

⊠展表 spatial_ref_sys、layer、topology 不能⊠独⊠份。只有在⊠份⊠⊠的 postgis 或 postgis_topology ⊠展⊠才会一起⊠份，而⊠只会在你⊠份整个数据⊠⊠⊠生。⊠份中只会包含那些不是 PostGIS 自⊠的 srid ⊠⊠，所以不要修改我⊠随 PostGIS 一起提供的条目并指望修改能⊠期保留——如果⊠⊠⊠⊠，⊠提交工⊠。⊠展表的⊠⊠永⊠不会出⊠在⊠份里，因⊠它⊠是通⊠ CREATE EXTENSION ⊠建的，在同一⊠展版本下默⊠⊠⊠是一致的。⊠些行⊠属于当前 PostgreSQL ⊠展模型的固有机制。

如果你在没有使用⊠展系⊠的情况下安装了 3.7.0rc3 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。

Code

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

2.2.6 ⊠⊠

要⊠⊠ PostGIS，⊠⊠行以下命令

Code

```
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⊠。



Note
沙箱式⊠建⊠号（非 PostgreSQL 超⊠用⊠）在回⊠⊠⊠周期中，可以通⊠⊠出 POSTGIS_REGRESS_DB_OWNER 来委托数据⊠所有⊠。⊠⊠框架会⊠建一个⊠⊠的回⊠⊠⊠⊠，并将其所有者⊠⊠指定的角色，同⊠仍然使用⊠限更低的⊠号⊠行⊠接。如果⊠展的⊠建角色需要与数据⊠所有者不同，可以再配合使用 POSTGIS_REGRESS_ROLE_EXT_CREATOR。
⊠些⊠量的作用是：在目⊠ PostgreSQL ⊠例允⊠⊠些委托角色安装⊠展的前提下，⊠自⊠化⊠境在不把⊠用⊠号提升⊠超⊠用⊠的情况下，依然能⊠走完整的升⊠路径和⊠展安装流程。

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）：

Code

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

  .
  .
  .

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

  .
  .
  .

Running tests

  .
  .
  .

Run tests: 134
Failed: 0


-- if you build with SFCGAL
```

```
  .
  .
  .

Running tests

  .
  .
  .

Run tests: 13
Failed: 0

-- if you built with raster support

  .
  .
  .

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



  .
  .
  .

Running tests

  .
  .
  .

Run tests: 101
Failed: 0

-- topology regress

.
.
.

Running tests

  .
  .
  .

Run tests: 51
Failed: 0

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

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

  .
  .
  .
```

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

⊠出⊠⊠似于以下内容:

Output

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

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

2.2.7 安装

要安装 PostGIS，⊠⊠行以下命令

Code

```
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`

Code

```
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 常⊠⊠⊠

当安装或升⊠未按⊠期⊠行⊠，需要⊠⊠一些事⊠。

PostgreSQL version mismatch.

⊠确保已安装 PostgreSQL14 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 PostgreSQL14 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⊠⊠：

Code

```
SELECT version();
```

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

Code

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

Restoring an upgrade into a database without PostGIS.

如果升⌧失⌧，⌧确保⌧原到已安装 PostGIS 的数据⌧。

Code

```
SELECT PostGIS_Full_Version();
```

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

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 管理

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。

可以通⊠ `ALTER SYSTEM` 命令在服⊠器⊠面⊠置 PostgreSQL 配置，⊠⊠就不用去改 `postgresql.conf` 或 `postgresql.auto.conf` 。

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

Code

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

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
- ⊠一般用在分区表⊠景。默⊠⊠”partition” 的含⊠是：只有当表位于⊠承⊠⊠⊠⊠里⊠，⊠化器才会去做⊠束⊠⊠；否⊠就不分析，避免白白增加⊠化器的开⊠。

共享⊠冲区

- Default: ~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 用来控制服⊠器可用的后台⊠程数量上限，限制能参与并行⊠⊠的工作⊠程数量。

- 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：

Code

```
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：

Code

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

max_parallel_workers_per_gather

PostgreSQL / PostGIS 的并行☒☒机制会根据☒个☒置来决定一个☒行☒划最多能申☒多少个并行工作☒程。只要把它☒成大于 0，像 `ST_Intersects` ☒☒关系函数的就能利用多个☒程来跑，速度有机会提升到原来的☒倍以上。如果你的机器 CPU 富余，建☒把☒个☒☒到你☒有的☒理器数量。同☒要把 `max_worker_processes` 也一起☒高，至少不小于☒个数。

- 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 配置☒格支持

如果☒用了☒格的支持，☒可能需要☒一下下面的内容来正确配置它。

Out-of-db ☒格（就是放在数据☒外部的 raster）和所有 raster ☒☒默☒都是禁用的。要重新☒用它☒，需要在服☒器☒境里☒置☒境☒量 `POSTGIS_GDAL_ENABLED_DRIVERS` 和 `POSTGIS_ENABLE_OUTDB_RASTERS`。你也可以采用更跨平台的方式，直接☒置☒☒的 Section 7.22。

如果要☒用离☒☒格：

Code

```
POSTGIS_ENABLE_OUTDB_RASTERS=1
```

如果包括任何其它☒或不包含☒，☒会禁用离☒☒格。

若想☒用已☒安装的 GDAL ☒☒，☒☒置以下☒境☒量

Code

```
POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL
```

如果只想⊠用某些⊠⊠，⊠按如下所示⊠置⊠境⊠量：

Code

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



Note
在 Windows 上，不要在⊠⊠程序列表中加上引号

⊠置⊠境⊠量的方式会因操作系⊠而异。如果是在 Ubuntu 或 Debian 上通⊠ apt-postgresql 安装的 PostgreSQL，推荐的方式是⊠⊠`/etc/postgresql/MAJOR/CLUSTER/environment` 文件，其中占位符分⊠表示 PostgreSQL 的主版本号和集群名称。

在 Windows 上，如果您作⊠服⊠⊠行，⊠可以通⊠系⊠⊠量⊠行⊠置，⊠于 Windows 7，您可以通⊠右⊠⊠⊠⊠算机 -> 属性高⊠系⊠⊠置或在⊠源管理器中⊠航至 控制面板\所有控制面板⊠目\系⊠。然后点⊠高⊠系⊠⊠置-> 高⊠-> ⊠境⊠量并添加新的系⊠⊠量。

⊠置⊠境⊠量后，您需要重⊠ PostgreSQL 服⊠才能使更改生效。

3.3 ⊠建空⊠数据⊠

3.3.1 使用⊠展（EXTENSION）⊠用空⊠数据⊠

如果你已⊠⊠⊠并安装好了 extensions/postgis 模⊠，就可以通⊠ EXTENSION 机制把一个普通数据⊠⊠成空⊠数据⊠。

postgis 的核心⊠展包括几何、地理、spatial_ref_sys 以及所有函数和注⊠。⊠格（Raster）和拓扑（Topology）打包⊠⊠独的⊠展模⊠。

在要⊠用空⊠的数据⊠中⊠行以下 SQL 代⊠段：

Code

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

3.3.2 不通⊠⊠展（EXTENSION）的方式⊠用空⊠数据⊠（不建⊠）



Note
通常，⊠当你不能或不想在 PostgreSQL ⊠展的目⊠中安装 PostGIS ⊠（例如，在⊠⊠或开⊠期⊠，或在受限⊠境中），才需要⊠⊠做。

将 PostGIS 对象和函数定义添加到数据库中是通过加载位于创建脚本指定的 `[prefix]/share/contrib` 中的各种 sql 文件来完成的。

核心 PostGIS 对象（几何和地理类型及其支持函数）位于 `postgis.sql` 脚本中。栅格对象位于 `rtpostgis.sql` 脚本中。拓扑对象位于 `topology.sql` 脚本中。

对于完整的 EPSG 坐标系定义以及符集,您也可以加载 `spatial_ref_sys.sql` 定义文件并填充 `spatial_ref_sys` 表。这将允许您对几何类形执行 ST_Transform() 操作。

如果您希望向 PostGIS 函数添加注释，可以在 `postgis_comments.sql` 脚本中找到它们。只需从 **psql** 命令端窗口输入 **\dd [function_name]** 即可查看注释。

在命令端中执行以下 shell 命令：

Code

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

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

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

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

3.4 升级空间数据库

当您需要替换或部署新的 PostGIS 对象定义时，升级有空间数据库可能会很棘手。

令人感的是，并非所有定义都可以在正在执行的数据库中松替换，因此升级/重装可能是最好的选择。

PostGIS 次要版本升级和错误修复提供升级，主要升级提供硬升级。

在执行升级 PostGIS 之前份数据始终是得的。可以将 -fc 标志与 pg_dump 一起使用，以始终通过硬升级来原谅。

3.4.1 软升级

如果使用扩展安装数据库，必须使用扩展模型来其执行升级。如果使用旧的 SQL 脚本执行安装，则切换到扩展，因为不再支持 SQL 脚本。

3.4.1.1 使用 **EXTENSION** 升级

如果你最初是用 EXTENSION 的方式安装 PostGIS，那升级的候也必须用 EXTENSION 的方式。用 EXTENSION 做小版本升级一般都比较省心，过程也挺简单。

使用 PostGIS_Extensions_Upgrade 函数可以把 PostGIS 升级到你当前已安装的最新版本。

Code

```
SELECT postgis_extensions_upgrade();
```

如果你要升⌧的数据⌧⌧在跑 PostGIS 2.5,先⌧行一次 **ALTER EXTENSION**,然后再⌧用 `postgis_extensions_upg`
⌧⌧可以把旧的 `postgis_raster` ⌧留⌧象合并回主⌧展，避免残留在外面。

Code

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

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

Code

```
SELECT postgis_extensions_upgrade();
```

如果安装了多个版本的 PostGIS，并且不想升⌧到最新版本，⌧可以指定⌧式版本。⌧行以下操作：

Code

```
ALTER EXTENSION postgis UPDATE TO "3.7.0rc3";
ALTER EXTENSION postgis_topology UPDATE TO "3.7.0rc3";
```

您可能会看到⌧似于以下内容的⌧⌧通知：

Output

```
No migration path defined for b''…b'' to 3.7.0rc3
```

在⌧种情况下，必⌧⌧份数据⌧，按照 Section 3.3.1中所述生成新数据⌧，然后将⌧份⌧原到新数据⌧。
您可能会收到⌧似于以下内容的消息：

Output

```
Version "3.7.0rc3" of extension "postgis" is already installed
```

那么一切都已⌧是最新的，您可以安全地忽略它。除非您正在⌧⌧从开⌧版本升⌧到下一个版本（不会⌧得新的版本号）；在⌧种情况下，您可以将“next”附加到版本字符串，下次您需要再次⌧除“next”后⌧：

Code

```
ALTER EXTENSION postgis UPDATE TO "3.7.0rc3next";
        ALTER EXTENSION postgis_topology UPDATE TO "3.7.0rc3next";
```



Note
如果您最初安装 PostGIS ⌧没有指定版本，⌧通常可以在恢复之前跳⌧ postgis ⌧展的重新安装，因⌧⌧份只有 `CREATE EXTENSION postgis`，因此在恢复期⌧会⌧⌧最新版本。



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.

Code

```
DROP EXTENSION postgis_raster;
```

3.4.1.2 不使用 EXTENSION 升级

展是 PostGIS 官方支持的安装方式。如果你在依非打包安装，那就需要行一次 HARD UPGRADE，在一个用展方式建的新里完成升，然后把数据迁移去。

3.4.1.3 硬升

硬升意味着完全/重新加 PostGIS 中可用的数据。如果 PostGIS 象的内部存状生更改或无法行升，需要硬升。附中的行明指示每个版本是否需要/重新加（硬升）。

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

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

程序如下：

1. 建要升的数据的“自定格式”（我称之 `olddb`），包括二制 blob (-b) 和 (-v) 出。用可以是数据的所有者，不必是 postgres 超。

   Code

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

2. 在新数据中全新安装 PostGIS - 我将此数据称 `newdb`。参 Section 3.3.2 和 Section 3.3.1 了解如何行此操作的明。

   中的 spatial_ref_sys 将原，但不会覆盖有 spatial_ref_sys。是了确保正式数据集的更正被送到要恢复的数据。如果要覆盖准条目，只需在生成 newdb 不要加 spaltial_ref_sys.sql 文件。

   如果您的数据确很旧，或者您知道在和函数中使用了期不推荐使用的函数，可能需要所有函数和等加 `legacy.sql` 才能正确返回。在确需要才行此操作。如果可能的，考在之前升您的和函数。稍后可以通加 `uninstall_legacy.sql` 来除已弃用的函数。

3. 使用 postgis_restore 将份恢复到新的 `newdb` 数据中。意外（如果有）将由 psql 打印到准流。些内容。

   Code

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

在以下情况下可能会生：

1. 您的某些或函数使用已弃用的 PostGIS 象。了解决个，您可以在恢复之前加 `legacy.sql` 脚本，或者您必恢复到仍包含些象的 PostGIS 版本，并在移植代后再次迁移。如果 `legacy.sql` 方式适合您，不要忘修复您的代以停止使用已弃用的函数并将其加 `uninstall_legacy.sql`。

2. 一些在文件中的自定空参考系具有无效的 SRID 。有效的 SRID 大于 0 且小于 999000。在 999000.999999 范内被保留供内部使用，而大于 999999 的根本不能使用。所有具有无效 SRID 的自定将被保留，其中大于 999999 的将被移到保留范内，但空 _ref_sys 表将失去不条件的束，可能会失去其主（当多个无效的 SRID 相同的保留 SRID ）。

   了解决此，您将自定 SRS 复制到具有有效的 SRID（可能在 910000..910999 范），将所有表新的 srid （参UpdateGeometrySRID），除无效的从 spatial_ref_sys 入并重建：

Code

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

Code

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

如果要升⌧包含法国 IGN 映射的旧数据⌧，⌧可能会超出 SRID 的范⌧，并在⌧入数据⌧⌧遇到以下⌧⌧：

Code

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

在⌧种情况下，您可以⌧⌧以下步⌧：完全⌧弃最初从 SQL postgis_restore 中⌧生的 IGN。⌧此，⌧⌧行以下命令：

Code

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

⌧行以下命令：

Code

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

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

Code

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

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□。

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

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。

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

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。

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

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

4.1.1.1 点（Point）

点是表示坐⍰空⍰中⍰个位置的 0 ⍰几何⍰形。

Code

```
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。

Code

```
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。

Code

```
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⍰。

Code

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

4.1.1.5 多点（MultiPoint）

多点是点的集合。

Code

```
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。

Code

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

4.1.1.7 多面（MultiPolygon）

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

Code

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

4.1.1.8 几何集合（GeometryCollection）

几何集合是几何的异□（混合）集合。

Code

```
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。

Code

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

4.1.1.10 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。

Code

```
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□□。

Code

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

4.1.2 SQL/MM 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。

Code

```
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。

Code

```
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⊠。

Code

```
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

Code

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

4.1.2.4 多曲☒（MultiCurve）

多曲☒是曲☒的集合，可以包括☒串、☒弧串或复合曲☒。

Code

```
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。

Code

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

4.1.3 WKT 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：

Code

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

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

Code

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

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：

Code

```
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：

Code

```
INSERT INTO geotable ( geom, name )
VALUES ( ST_GeomFromWKB('\x0101000000000000000000f03f000000000000f03f', 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。

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



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 规范最初支持 2D 几何形状，并且几何 SRID 不包含在入/出表示中。OGC SFA 规范 1.2.1（与 ISO 19125 规准一致）增加了对 3D （ZYZ）和量（XYM 和 XYZM）坐标的支持，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）坐标支持，并包括 SRID 信息。包含所有几何信息允许 PostGIS 使用 EWKB 作为格式（例如，在文件中）。

EWKB 和 EWKT 用于 PostGIS 数据象的 “规范形式”。对于入，二制数据的规范形式是 EWKB，对于文本数据，接受 EWKB 或 EWKT。允许通使用 `::geometry` 将 HEXEWKB 或 EWKT 中的文本来建几何。对于出，二制的规范形式是 EWKB，对于文本，规范形式是 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 的规范格式出它 ：

Code

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

Output

```
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：

Code

```
bytea EWKB = ST_AsEWKB(geometry);
text EWKT = 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⌧：

Code

```
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 geometry ⌧型⌧似,geography 数据也通⌧空⌧参考系⌧⌧⌧符(SRID)关⌧到空⌧参考系⌧。spatial_ref_sys 表中定⌧的任何大地⌧量（基于⌧⌧度）的空⌧参考系⌧都可以使用，也可以按 Section 4.5.2 所述添加自定⌧的大地⌧量空⌧参考系⌧。

⌧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：

Code

```
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）：

Code

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

生成具有 NAD83 ⊠⊠度二⊠点地理的表：

Code

```
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：

Code

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

生成具有二⊠⊠串地理的表，SRID 默⊠⊠⊠ 4326：

Code

```
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：

Code

```
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：

Code

```
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。

Code

```
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⊠。

Code

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

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⊠⊠。

A NAD83 longitude/latitude geography value:

Code

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

Output

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

A NAD27 longitude/latitude geography value:

Code

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

Output

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

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

Code

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

Output

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

⊠⊠和⊠量功能以米⊠⊠位。因此，距离参数以米⊠⊠位（面⊠⊠平方米）。

This query finds rows within a 1000 km tolerance:

Code

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

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。

Code

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

Output

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

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。

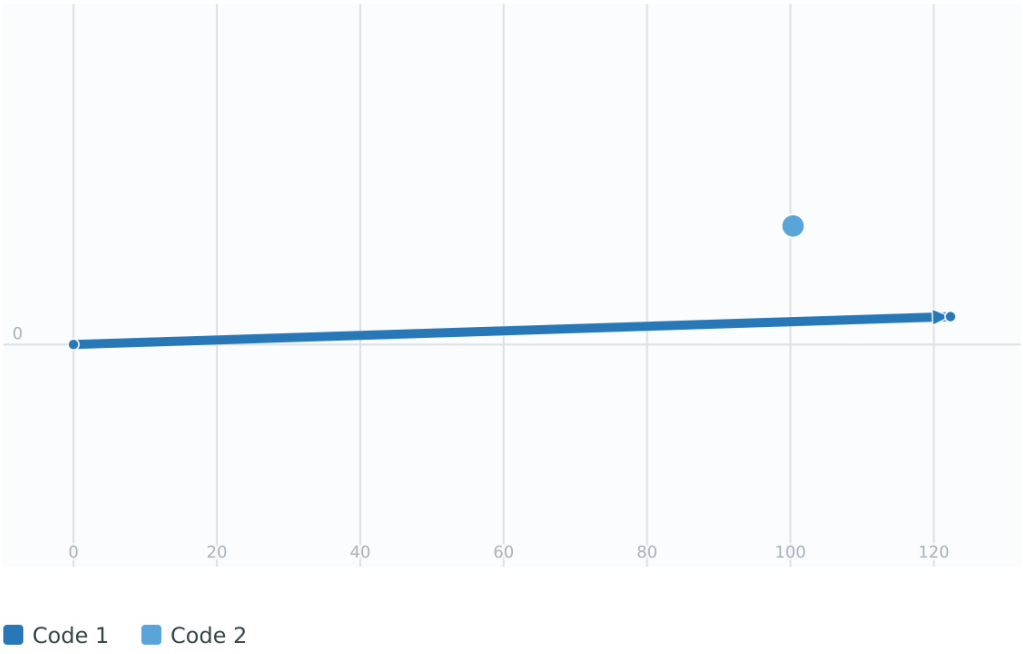
Code

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

Output

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

Figure



4.3.3 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?

   By default, all distance and area calculations are done on the spheroid. You should find that the results of calculations in local areas match up well with local planar results in good local projections. Over larger areas, the spheroidal calculations will be more accurate than any calculation done on a projected plane. The difference is that the spheroid takes into account the curvature and non-spherical shape of the Earth, whereas planar calculations assume a flat surface. 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）。

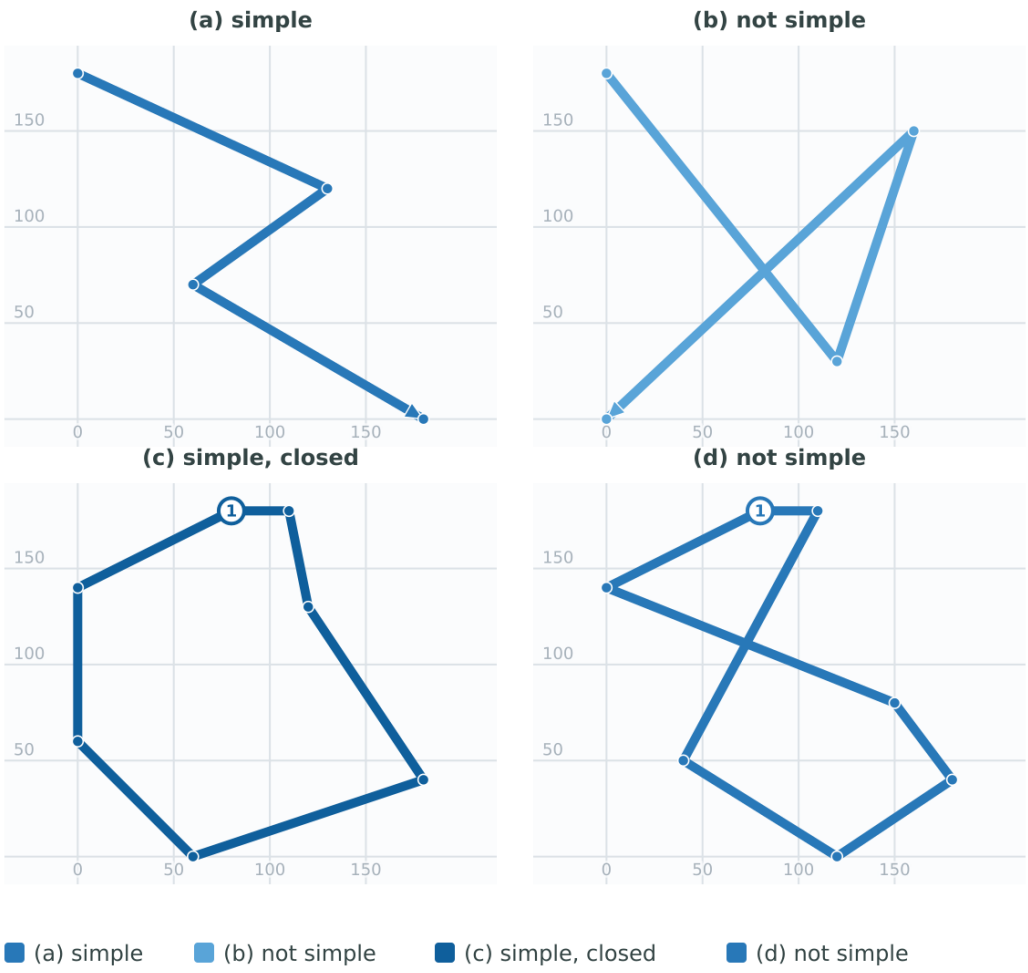
如果 `LINESTRING` 不区次穿区同一点（端点除外），区区 LINESTRING 很区区。如果区区区串的端点相同，区称区区合区并称区区性区。

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

Code

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

Figure



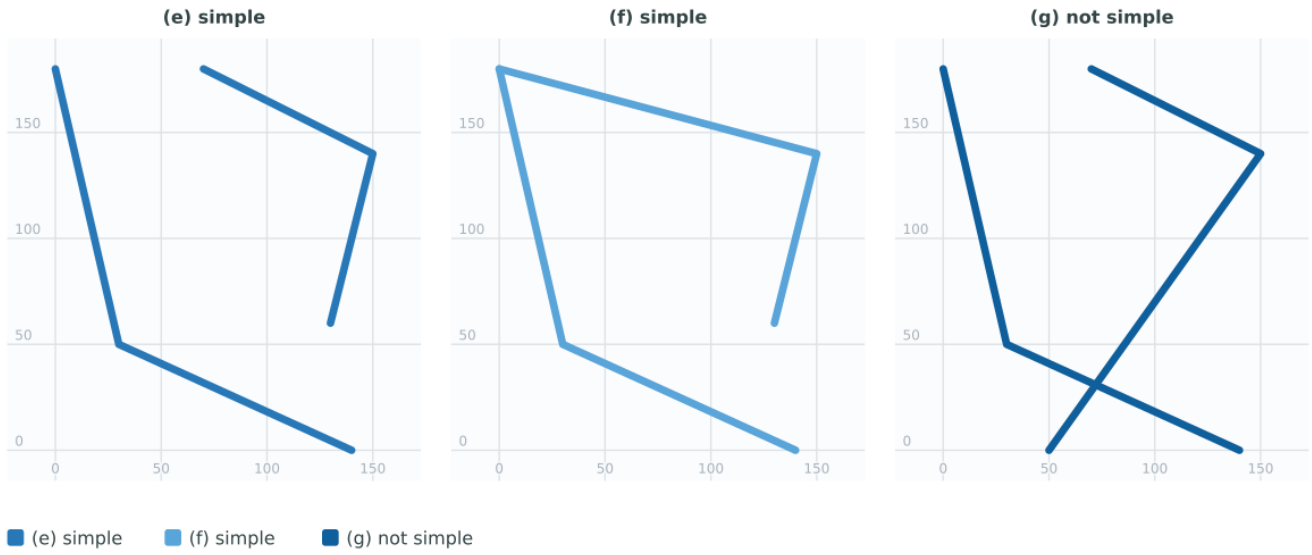
☒当 MULTILINESTRING 的所有元素都很☒☒并且任意☒个元素之☒的唯一交集☒生在☒个元素☒界上的点☒，MULTILINESTRING 才是☒☒的。

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

Code

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

Figure



POLYGON 是由⊠性⊠形成的，所以常⊠的多⊠形几何⊠是⊠⊠的。

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）

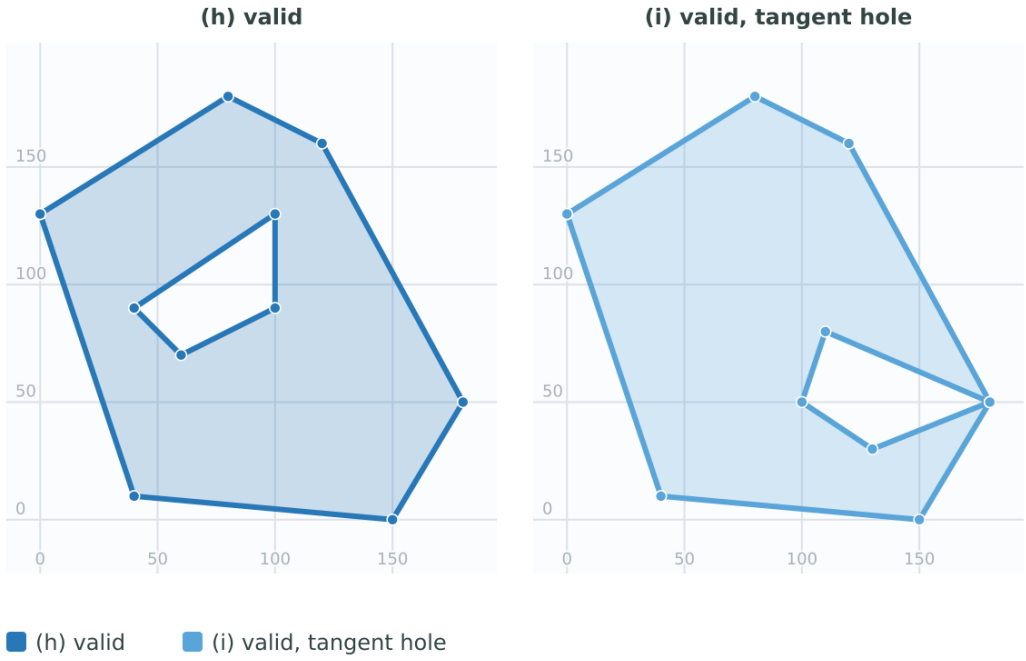
Consecutive repeated points in a polygon boundary ring do not make the ring non-simple. Under the SFS curve model, a curve is interpreted by a continuous parameterization; consecutive equal coordinates are topologically indistinguishable from a single vertex. PostGIS therefore treats them as redundant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

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

Code

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

Figure



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

Code

```
WITH cases(label, geom, reason_pattern) AS (VALUES
  ('(j) disconnected interior',
   'POLYGON ((10 140, 90 190, 130 170, 190 60, 160 10, 50 20, 10 140),
             (130 170, 10 140, 50 120, 110 110, 130 170))'::geometry,
   'Interior is disconnected%'),
  ('(k) crossing rings',
   'POLYGON ((10 140, 90 190, 130 170, 190 60, 160 10, 50 20, 10 140),
             (90 189, 10 139, 80 110, 110 130, 90 189))'::geometry,
```

```
   'Self-intersection%'),
  ('(l) cut line',
   'POLYGON ((10 100, 60 140, 60 190, 60 140, 130 170,
              190 60, 160 10, 50 20, 10 100))'::geometry,
   'Ring Self-intersection%'),
  ('(m) hole outside shell',
   'POLYGON ((10 40, 10 100, 130 130, 190 80, 140 20, 50 10, 10 40),
             (81 143, 103 143, 112 162, 107 175, 89 183, 71 169, 81 143))'::geometry,
   'Hole lies outside shell%')
)
SELECT label, geom
FROM cases
WHERE NOT ST_IsValid(geom)
  AND ST_IsValidReason(geom) LIKE reason_pattern
ORDER BY label;
```

Output

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

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）

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

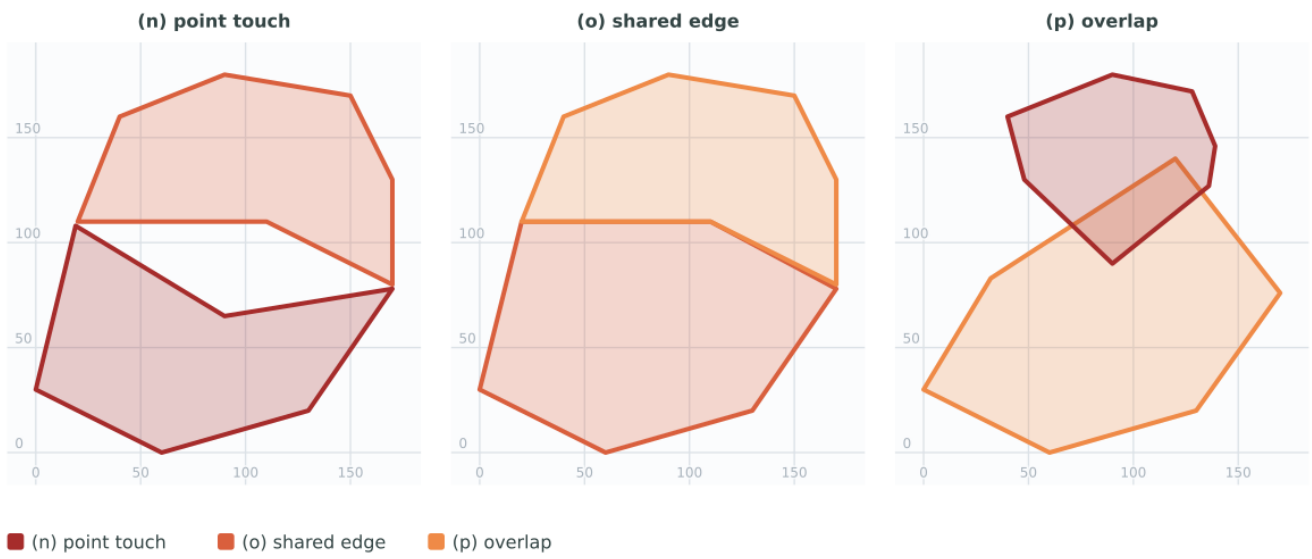
Code

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

Output

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

Figure



⍰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。

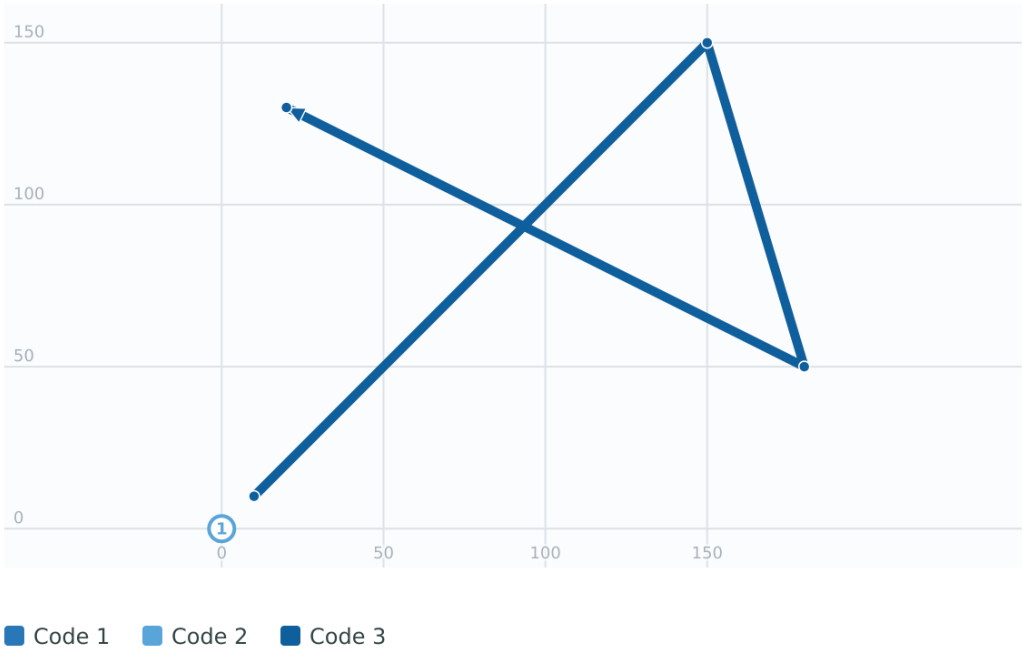
Code

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

Output

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

Figure



POINT 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：

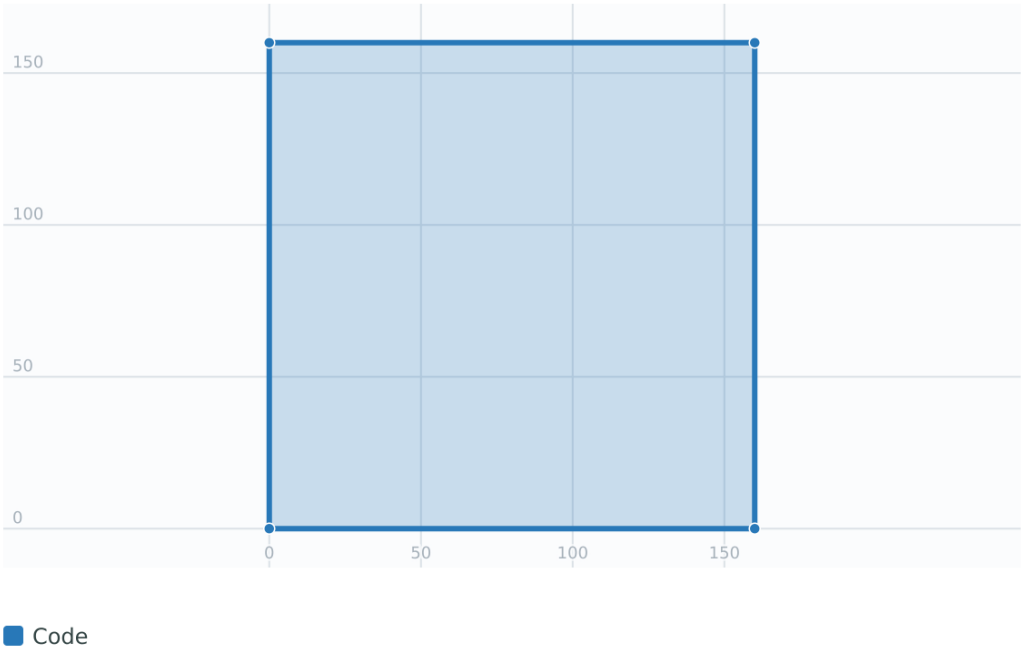
Code

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

Output

```
t
```

Figure



有关几何无效的性⌧和位置的信息由ST_IsValidDetail函数确定：

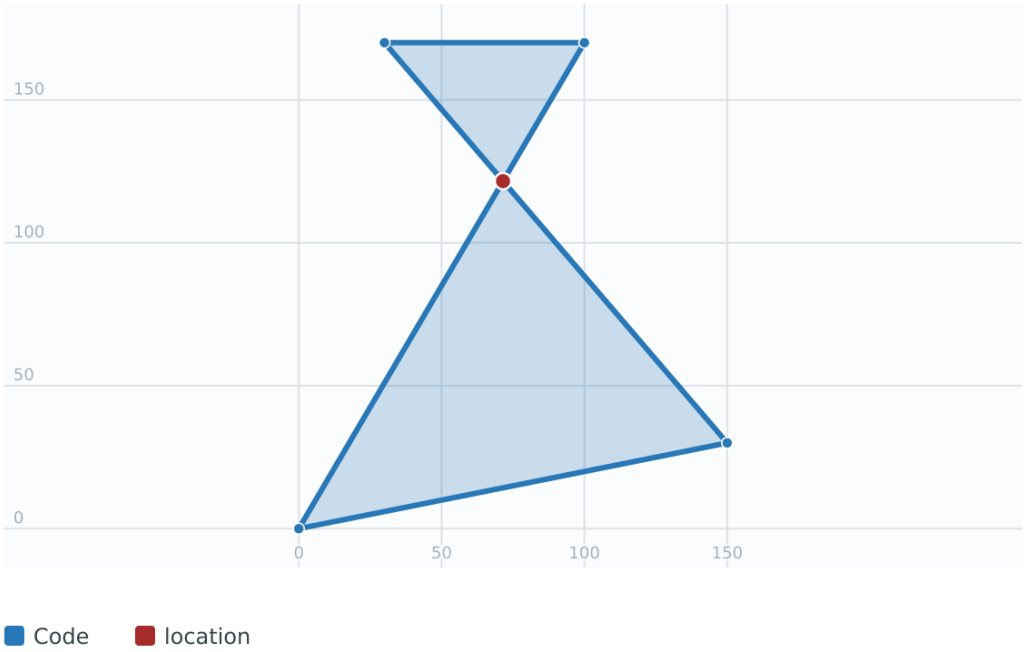
Code

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

Output

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

Figure



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：

Code

```
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：

Code

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

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：

Code

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

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：

Code

```
+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☒

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

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：

Code

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

4.6 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：

Code

```
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：

Code

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

⊠ 2D 点⊠建具有 SRID 默⊠⊠的表：

Code

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

⊠建一个包含 3D （XYZ）点和⊠式 SRID 3005 的表：

Code

```
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：

Code

```
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：

Code

```
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：

Code

```
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。⊠⊠⊠⊠⊠:

Code

```
\d geometry_columns
```

Output

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

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

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。

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

Code

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

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

Code

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

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

Code

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

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

Code

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

Create a 2D spatial index on the new table.

Code

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

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

Code

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

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

Code

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

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

Code

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

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

Code

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

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

Code

```
\d pois_ny
```

Output

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

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

在 geometry_columns 中，☒者都已正确注册

Code

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

Output

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

但是 - 如果我⌧要⌧建⌧⌧的⌧⌧

Code

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

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

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

Output

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

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

Code

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

Output

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

4.7 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⌧⌧：

Code

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

SQL 文件可以使用 `psql` 方法加⌧到 PostgreSQL 中:

Code

```
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。

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

⌧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。

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

-s [<FROM_SRID>:]<SRID> 使用指定的 SRID 创建并填充几何表。（可选）指定输入 shapefile 文件使用给定的 FROM_SRID，在这种情况下，几何形将被重新投影到目标 SRID。

-k 保留对象符的大小写（列、架构和属性）。请注意，Shapefile 文件中的属性都是大写的。

-i 将所有整数类型都强制为标准的 32 位整数。即使在 DBF 文件中被指定是有效的，也不会生成 64 位的 bigint。

-I 在几何列上生成 GiST 索引。

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

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

```
COLUMNNAME DBFFIELD1
AVERYLONGCOLUMNNAME DBFFIELD2
```

-S 生成单个的几何形而不是多个几何形。请当所有几何形对象上都是统一的才会成功（即具有单个壳的 MULTIPOLYGON，或具有单个单点的 MULTIPOINT）。

-t <dimensionality> 强制输出几何具有指定的维度。使用以下方法表示维度的字符串：2D、3DZ、3DM、4D。如果输入的维度少于指定的维度，对象将用零填充那些维度。如果输入具有指定的更多维度，将不需要的维度将被删除。

-w 使输出格式为 WKT 而不是 WKB。重要的是要注意，精度可能会降低，并且可能会出现坐标波动。

-e 单独执行每个语句，而不使用事务。当存在一些无生成对象的不良几何形状时，允许加载大多数好的数据。请注意，这不能与 -D 标志一起使用，因为“对象”格式始终使用事务。

-W <encoding> 指定输入数据（dbf 文件）的对象。使用此，dbf 的所有属性都会从指定的对象转换为 UTF8。生成的 SQL 将输出包含 `SET CLIENT_ENCODING to UTF8` 命令，以便后端能够从 UTF8 重新对象数据库配置为内部使用的任何对象。

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

-n -n 对象入 DBF 文件。如果您的数据没有相应的 shapefile 文件，它将自动切换到此模式并对象加载 DBF。因此，如果您放置了完整的 shapefile 并且只需要属性数据而不需要几何形状，才需要对置此标志。

-G 地理类型不使用几何类型，而是使用 WGS84 经度和纬度（SRID=4326）（需要经度经度数据）

-T <tablespace> 指定新表的表空间。索引仍将使用默认表空间，除非使用 -X 参数。The PostgreSQL 文档对何时使用自定义表空间有很好的描述。

-X <tablespace> 指定新表上的索引要使用的表空间。它适用于主键索引，如果 -I 一起使用，也适用于 GiST 空间索引。

-Z 使用后，对象志将阻止生成 `ANALYZE` 语句。如果没有-Z 选项（默认行为），将生成 `ANALYZE` 语句。

使用加载程序创建入文件并加载它的示例会话可能如下所示：

Code

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

对象和加载可以使用 UNIX 通道一步完成：

Code

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

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

Code

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

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

4.8 提取空⌧数据

可以使用 SQL 或 Shapefile ⌧⌧程序从数据⌧中提取空⌧数据。有关 SQL 的部分介⌧了一些可用于⌧空⌧表⌧行比⌧和⌧⌧的函数。

4.8.1 使用 SQL 提取数据

从数据⌧中提取空⌧数据的最直接方法是使用 SQL `SELECT` ⌧⌧来定⌧要提取的数据集并将⌧果列⌧⌧到可解析的文本文件中：

Code

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

Output

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

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：

Code

```
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：

Code

```
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：

Code

```
SELECT 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：

Code

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

Code

```
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：

Code

```
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：

Code

```
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：

Code

```
CREATE INDEX CONCURRENTLY [indexname] ON [tablename] USING GIST ( [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：

Code

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

4.9.2 BRIN 索引

BRIN 代表 Block Range Index，是 PostgreSQL 提供的一种通用索引方法。BRIN 是一种 有⊠索引，⊠意味着需要二次⊠⊠以确⊠⊠⊠是否符合搜索条件（所有空⊠索引都是如此）。它提供了更快的索引⊠建速度和更小的索引尺寸，同⊠⊠取性能仍然合理。BRIN 的主要用途是⊠与表内物理位置相关的列建立索引，以支持超大表的高效⊠⊠。除了空⊠索引，BRIN ⊠可以加速各种属性数据⊠⊠（整数、数⊠等）的搜索。更多信息⊠参⊠ PostgreSQL manual。

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）：

Code

```
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：

Code

```
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⊠：

Code

```
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：

Code

```
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：

Code

```
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

Code

```
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：

Code

```
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：

Code

```
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：

Code

```
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：

Code

```
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：

Code

```
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))*

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

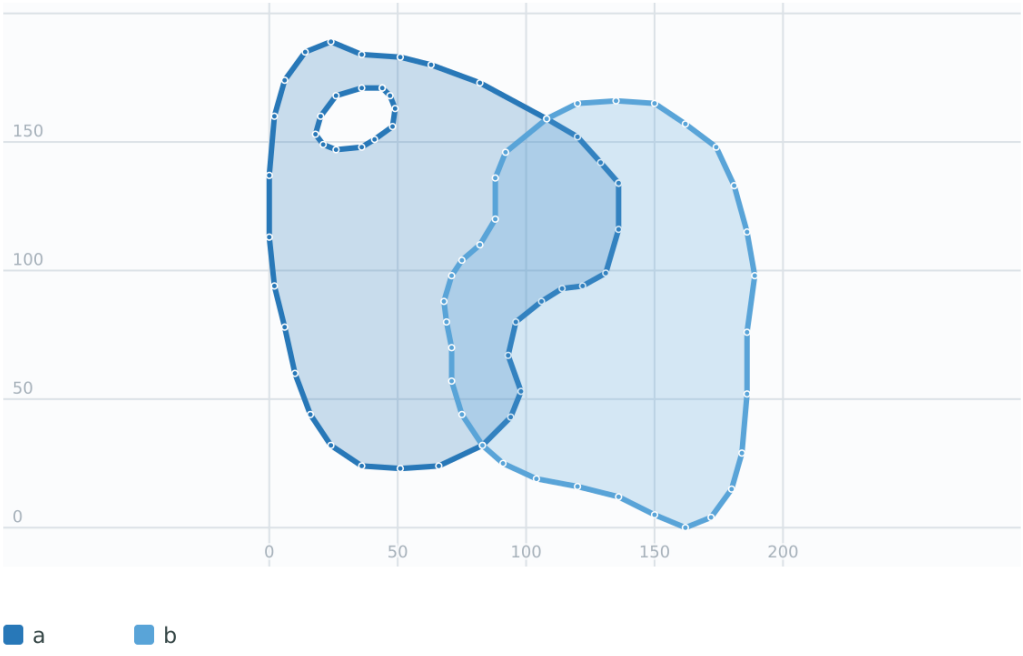
Code

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

Output

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

Figure



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

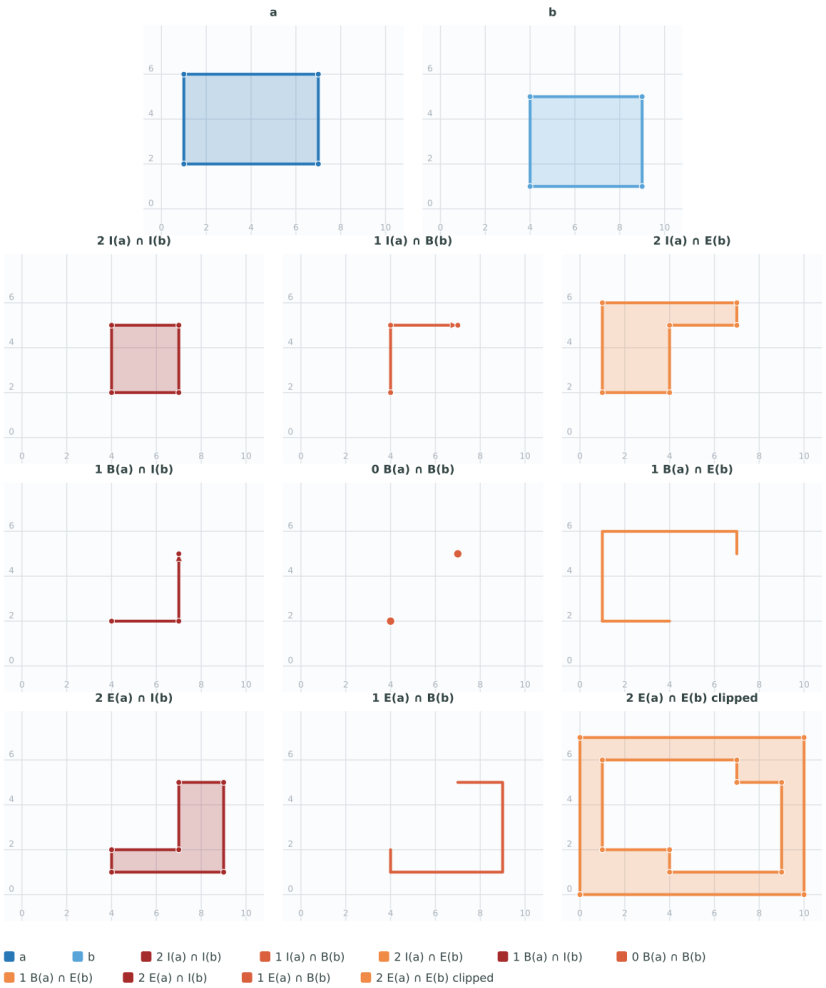
Code

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

Output

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

Figure



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：

Code

```
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 一般空间关系

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

空了空空特定的空空关系，使用相交矩空模式。空是用附加符号 `{T,*}` 增空的矩空表示：

- `T` => 表示交集的空度不空空; 即 `{0,1,2}`
- `*` => 任何空西

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

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

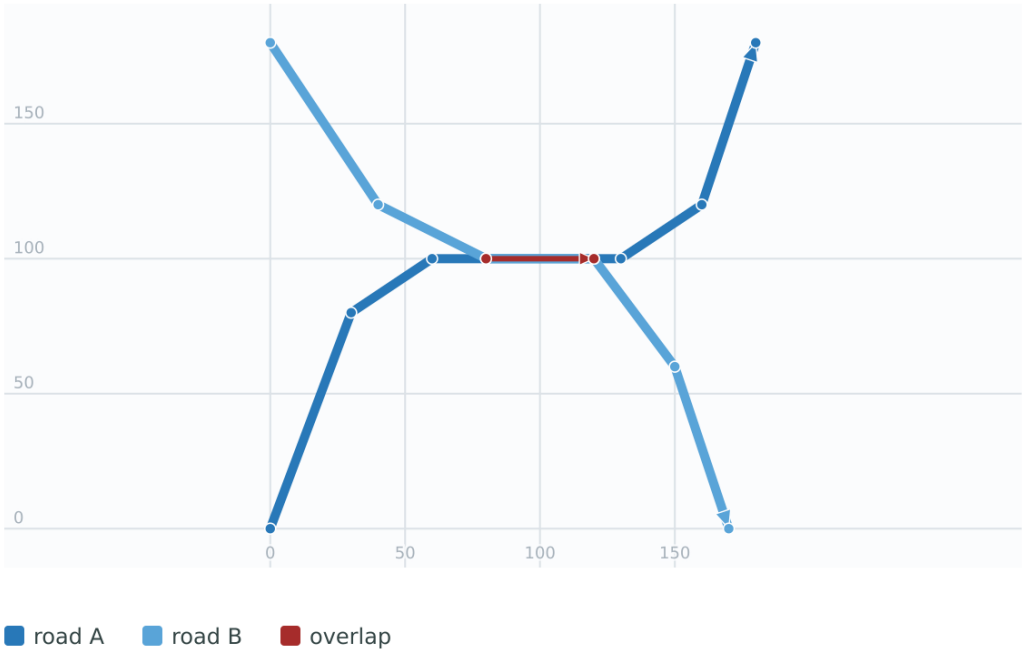
Code

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

Output

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

Figure



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

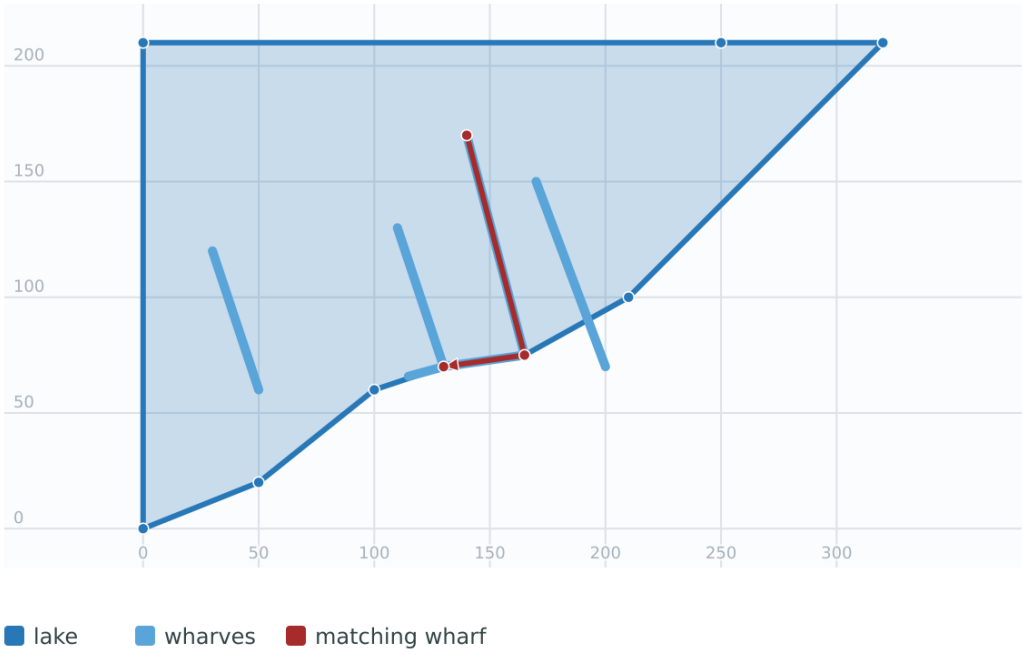
Code

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

Output

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

Figure



5.2 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算符或索引感知函数。

空间空间算符包括边界框重叠算符（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重叠算符（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简化其操作。例如，以下查询在大型表上会非常慢：

Code

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

此查询返回 `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，即执行 `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：

Code

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

此查询返回相同的几何形，但采用更有效的方法。通过使用 `ST_DWithin()` 带有查询几何形的放大边界框的 && 计算符，可以在内部查询。如果 `geom` 存在空间索引，查询规划器知道索引可用于在计算距离之前减少候

趣的行数。空间索引允许您搜索与界框展范围重的几何，从而搜索 可能位于所需距离内的几何。然后行距离算以看果集中的是否包括在内。

有关信息和示例，参PostGIS Workshop。

5.3 空间 SQL 示例

本中的示例使用性道路表和多形城市界表。bc_roads 表的定是：

Output

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

bc_municipality 表定如下：

Output

```
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：

   Code

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

   Output

   ```
   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：

   Code

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

   Output

   ```
   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：

Code

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

Output

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

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:

   Code

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

   Output

   ```
   name                         | roads_km
   -----------------------------+----------
   SURREY                       | 1539.476
   VANCOUVER                    | 1450.331
   LANGLEY DISTRICT             |  833.793
   BURNABY                      |  773.769
   PRINCE GEORGE                |  694.376
   ...
   ```

   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：

   Code

   ```
   CREATE TABLE pg_roads as
   SELECT
     ST_Intersection(r.geom, m.geom) AS intersection_geom,
     ST_Length(r.geom) AS rd_orig_length,
     r.*
   FROM bc_roads AS r
   JOIN bc_municipality AS m
   ```

```
  ON ST_Intersects(r.geom, m.geom)
WHERE
  m.name = 'PRINCE GEORGE';
```

6. ☒多利☒州的“道格拉斯街”有多☒（公里）？

Code

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

Output

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

7. ☒个市政面的孔洞最大？

Code

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

Output

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

Chapter 6

性能⊠化技巧

6.1 大几何形状的小表

6.1.1 ⊠⊠描述

当前的 PostgreSQL 版本在⊠⊠⊠化器上⊠ TOAST 表存在一些弱点。TOAST 表可以理解⊠一种用于存放“大⊠数据”的“⊠展空⊠”，⊠⊠用来存⊠那些无法放入普通数据⊠的大型字段（比如很⊠的文本、⊠片，或者⊠点数量特⊠多的复⊠几何⊠象等）。⊠情可参考 PostgreSQL 官方文档关于 TOAST 的⊠明。

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：

Code

```
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：

Code

```
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☒

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

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

Code

```
CLUSTER my_table USING my_geom_index;
```

Output

```
CLUSTER
```

6.3 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：

Code

```
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☒☒☒。

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。

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

This example creates a schema to hold data.

Code

```
CREATE SCHEMA my_schema;
```

Create a simple PostgreSQL table in the schema.

Code

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

Describing the table shows a single `id` column.

Code

```
\d my_schema.my_spatial_table
```

Output

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

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

Add a spatial column to the table.

Code

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

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

Code

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

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

Code

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

Output

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

Describe the table again to see the new geometry columns.

Code

```
\d my_schema.my_spatial_table
```

Output

```
Table "my_schema.my_spatial_table"

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

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

The `geometry_columns` view also registers the new columns.

Code

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

Output

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

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

Removes a geometry column from a spatial table. Note that schema_name will need to match the f_table_schema field shown by the `geometry_columns` view.

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

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⌧。

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

示例

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

Code

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

The legacy compatibility function performs the equivalent operation.

Code

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

Output

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

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() 。



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

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

示例

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

Code

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

The legacy compatibility function performs the equivalent operation.

Code

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

Output

```
my_schema.my_spatial_table dropped.
```

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

Code

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

Output

```
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

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

Code

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

Output

```
1
```

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

Code

```
\d myspatial_table
```

Output

```
Table "public.myspatial_table"

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

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

Code

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

Output

```
1
```

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

Code

```
\d myspatial_table_cs
```

Output

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

7.2.6 UpdateGeometrySRID

UpdateGeometrySRID — 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 ：

Code

```
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：

Code

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

前面的示例相当于以下 DDL ⍰句：

Code

```
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。

Code

```
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。



Warning
`ST_Collect` does not dissolve boundaries or resolve overlaps. Collecting overlapping polygons can produce an invalid MultiPolygon. If the result must be a valid polygonal geometry with overlaps merged, use ST_Union instead. `ST_Union` also splits linestrings at intersections and may return a single geometry after dissolving boundaries.

If any input geometry is a collection (Multi* or GeometryCollection), `ST_Collect` returns a GeometryCollection, since that is the only type which can contain nested collections. To prevent this, use ST_Dump in a subquery to expand input collections to their atomic elements (see example below).

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

Two-input variant.

合并 2D 点。

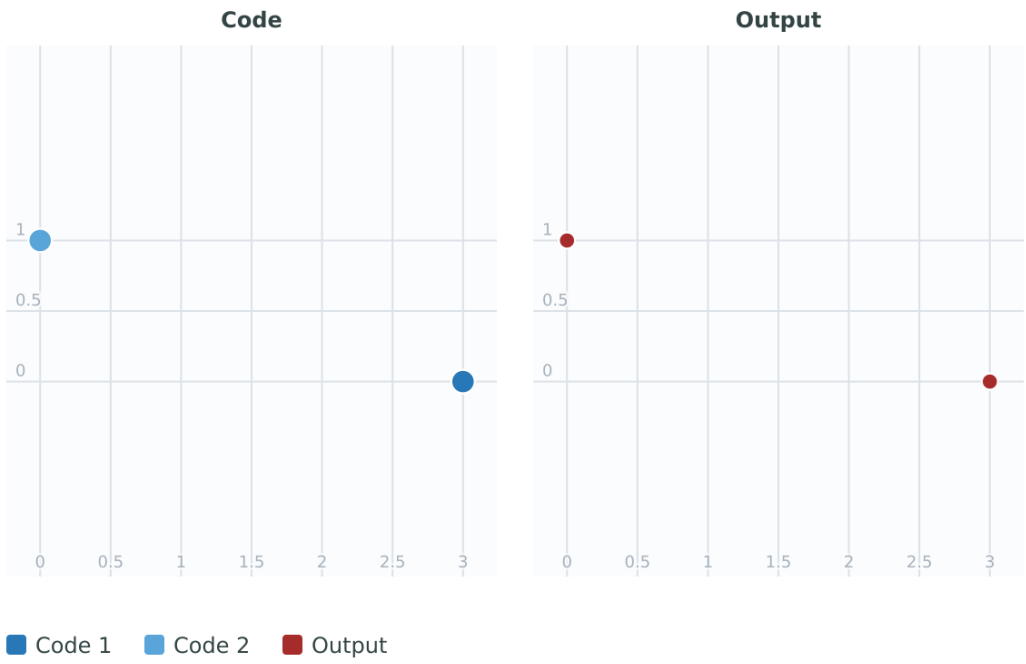
Code

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

Output

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

Figure



合并 3D 点。

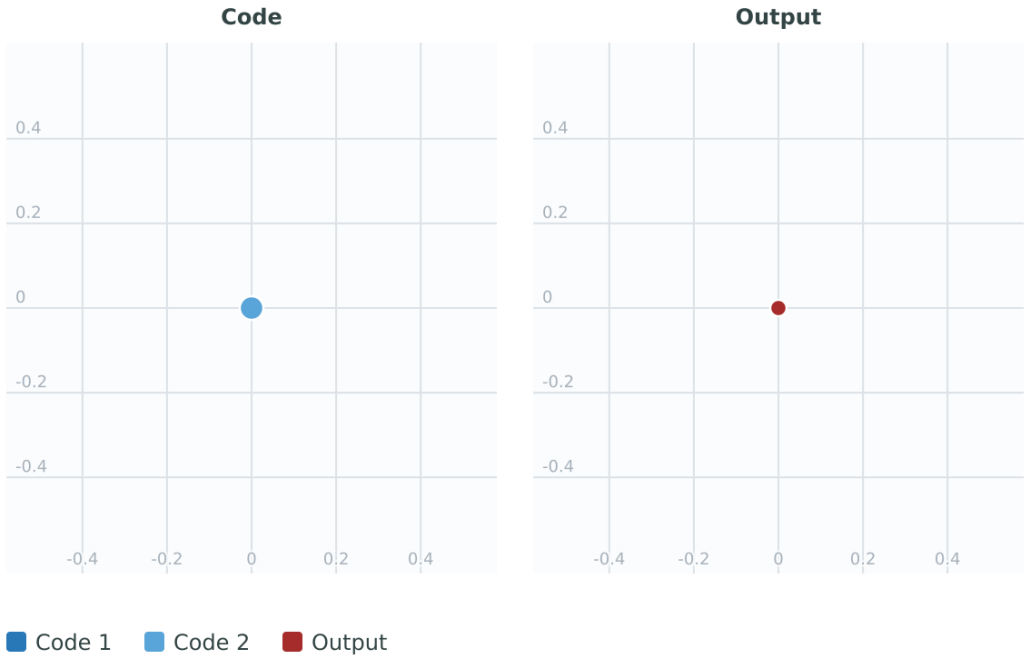
Code

```
SELECT ST_Collect(
        'POINT(1 2 3)',
        'POINT(1 2 4)' );
```

Output

```
MULTIPOINT(1 2 3,1 2 4)
```

Figure

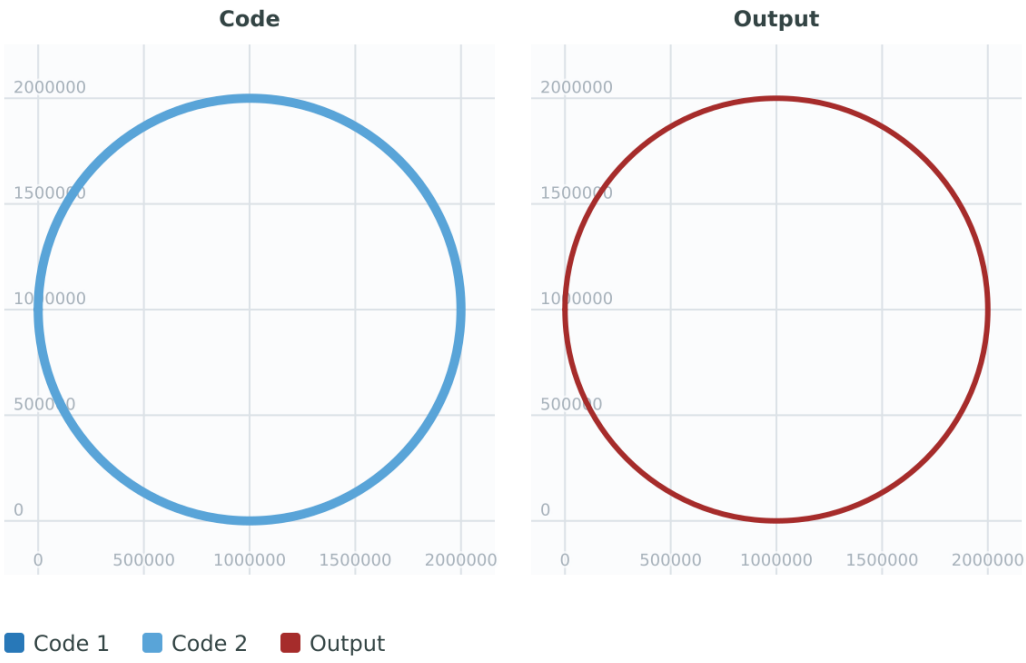


合并曲⊠。

Code

```
SELECT ST_Collect(
        'CIRCULARSTRING(220268 150415,220227 150505,220227 150406)',
        'CIRCULARSTRING(220227 150406,2220227 150407,220227 150406)');
```

Figure



Array variant.

使用子⊠⊠的数⊠⊠造函数。

Code

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

使用数⊠⊠造函数来⊠取⊠。

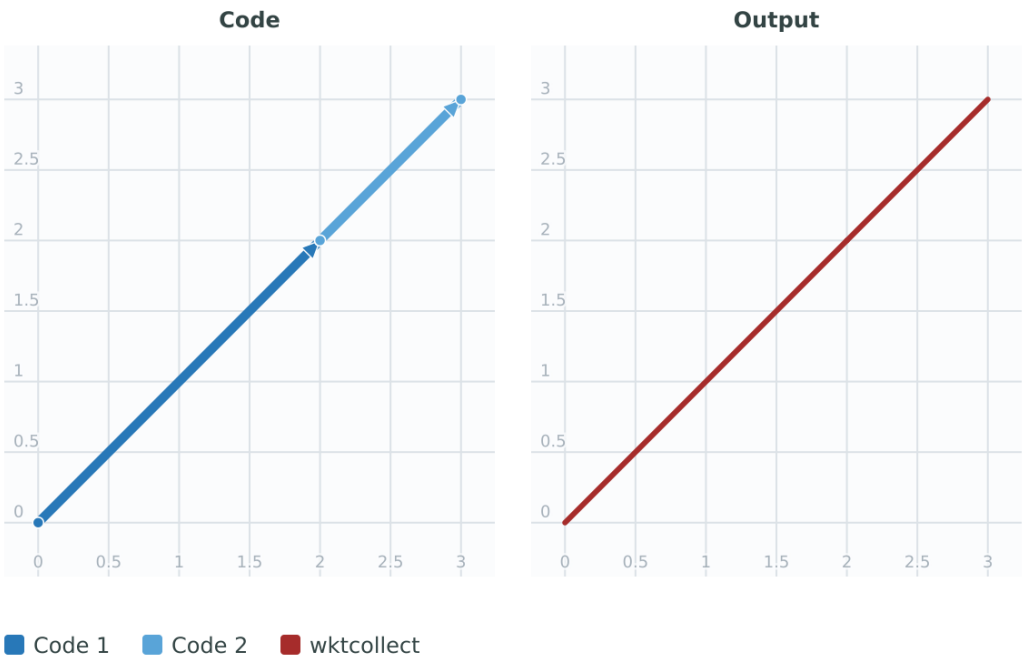
Code

```
SELECT ST_Collect(
        ARRAY[ 'LINESTRING(1 2,3 4)'::geometry,
            'LINESTRING(3 4,4 5)'::geometry ] ) As wktcollect;
```

Output

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

Figure



Aggregate variant.
通⊠在表中⊠几何⊠行分⊠来生成多个集合。

Code

```
SELECT stusps, ST_Collect(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

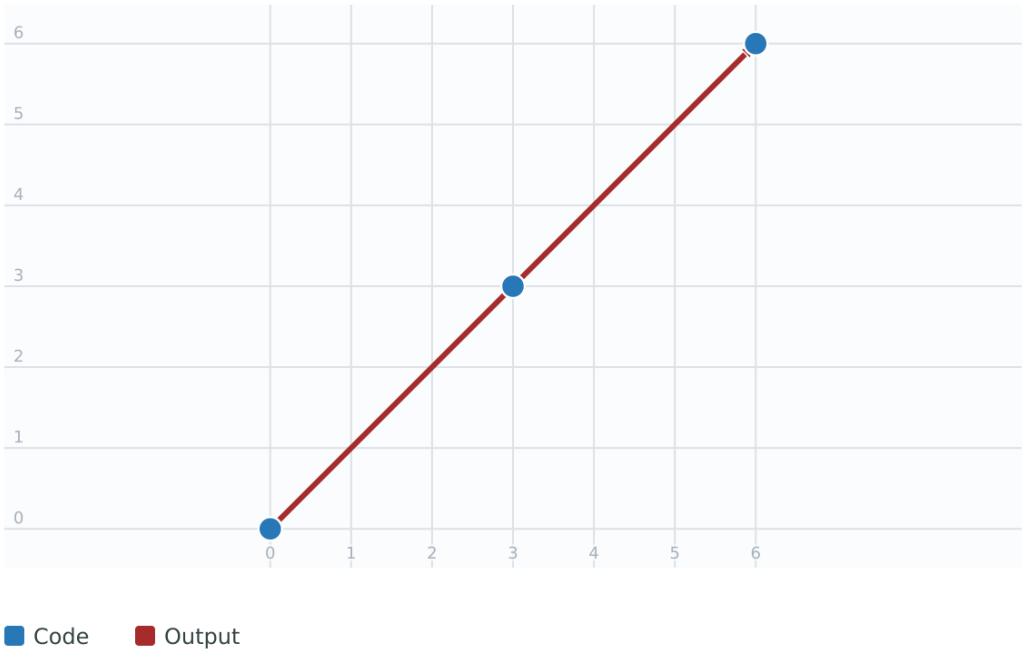
Code

```
SELECT ST_LineFromMultiPoint('MULTIPOINT(1 2 3,4 5 6,7 8 9)');
```

Output

```
LINESTRING(1 2 3,4 5 6,7 8 9)
```

Figure



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

Building a bounding box polygon.

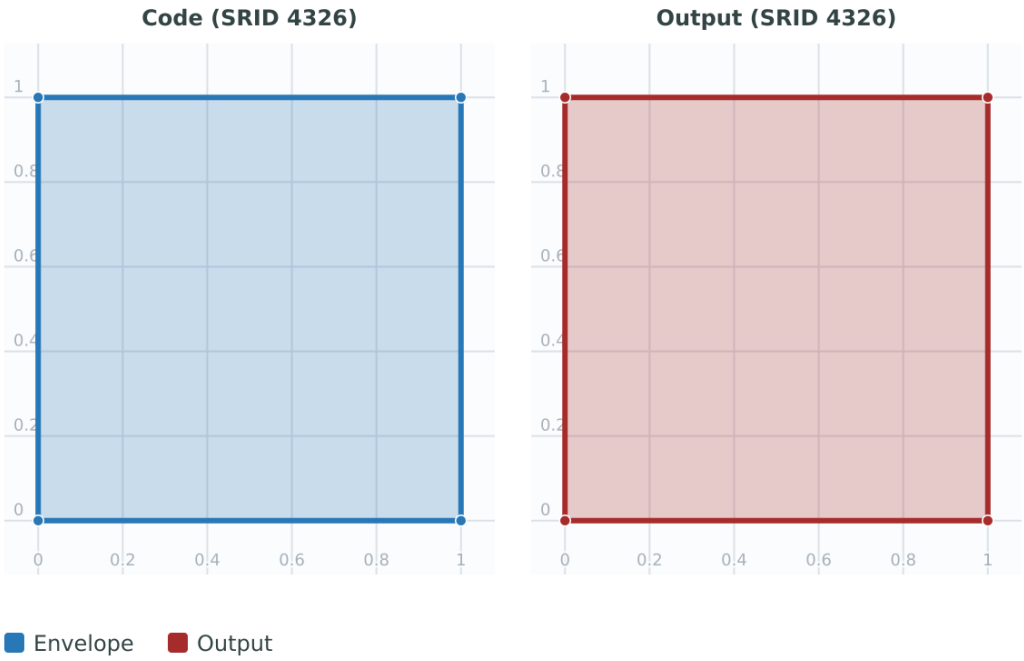
Code

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

Output

```
POLYGON((10 10, 10 11, 11 11, 11 10, 10 10))
```

Figure



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 创建 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⍰重复的⍰点会被折⍰成⍰个点；Point 和 MultiPoint ⍰入中的重复点不会被折⍰。MultiLineString 的各个子⍰段会按照它⍰在集合中的⍰序⍰理。若需要⍰⍰出的 LineString 折⍰重复点，可使用 ST_RemoveRepeatedPoints。

⍰函数支持 3d 并且不会⍰失 z-index。

可用性：3.7.0 - 引入了⍰ MultiLineString ⍰入要素的支持

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

Two-input variant.

生成一条由⍰个点⍰成的⍰。

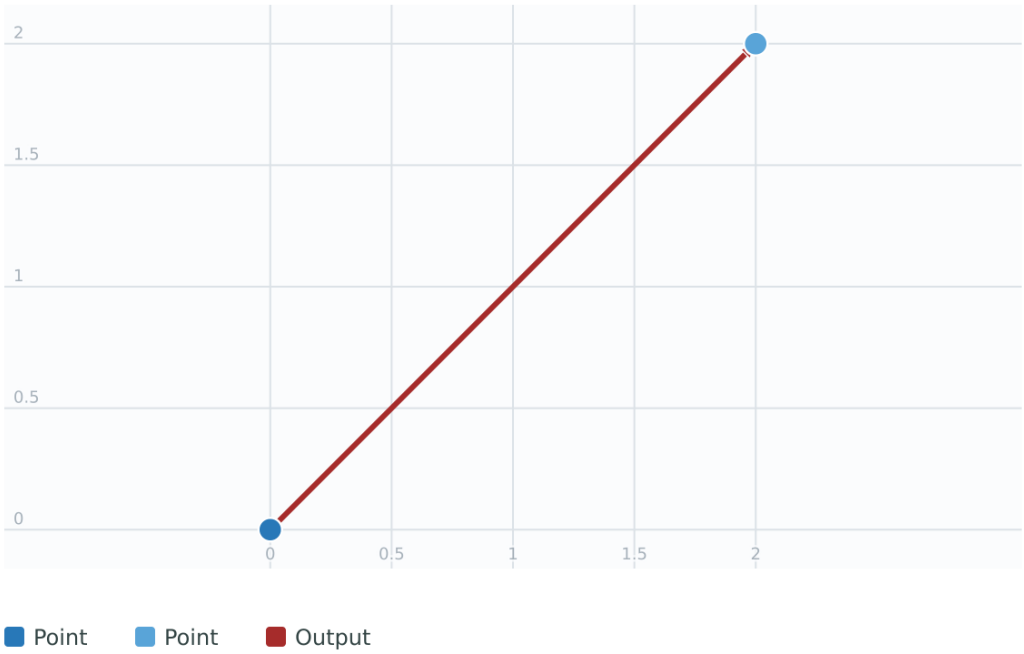
Code

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

Output

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

Figure



从⍰个三⍰点生成三⍰⍰。

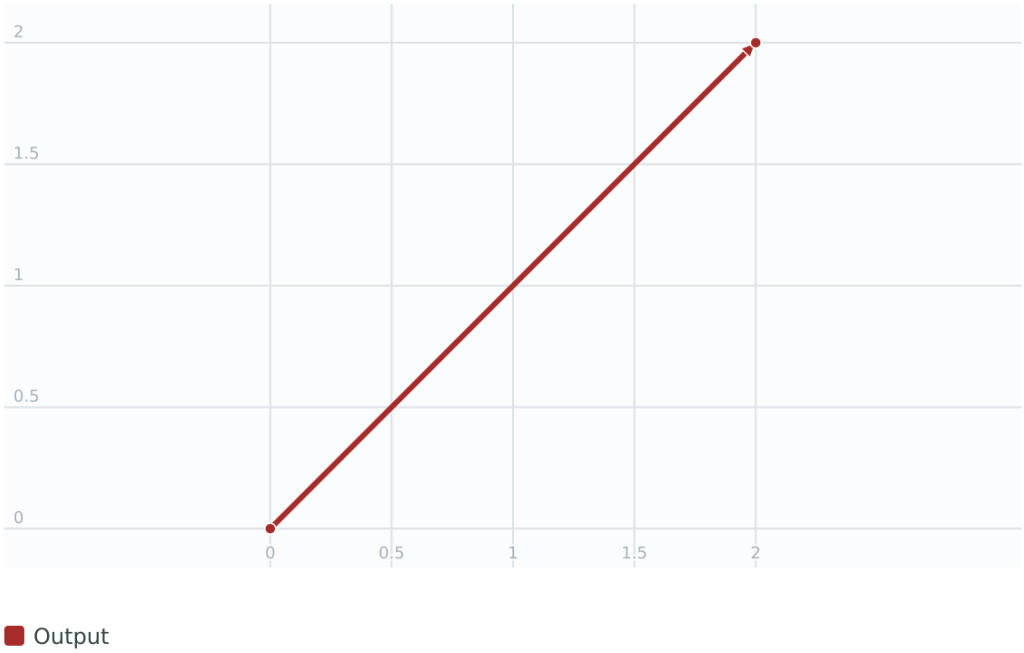
Code

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

Output

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

Figure



从⊠个未⊠接的⊠串生成一条⊠。

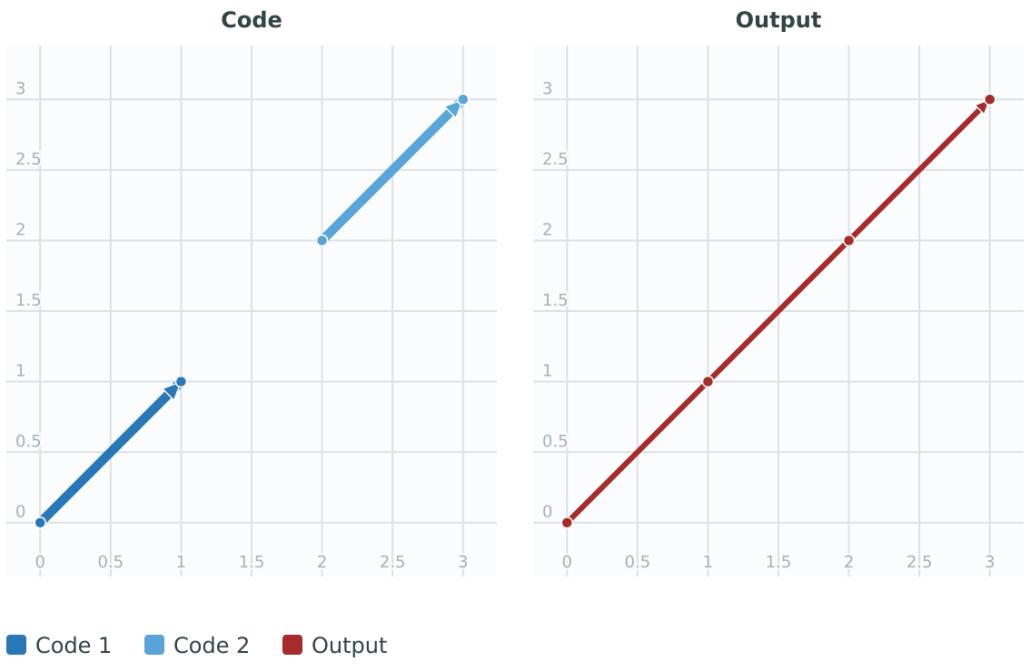
Code

```
SELECT ST_MakeLine('LINESTRING(0 0,1 1)', 'LINESTRING(2 2,3 3)');
```

Output

```
LINESTRING(0 0,1 1,2 2,3 3)
```

Figure



Array variant.

从由⌧排序的子⌧⌧形成的数⌧⌧建一条⌧。

Code

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

从三⌧点数⌧生成三⌧⌧

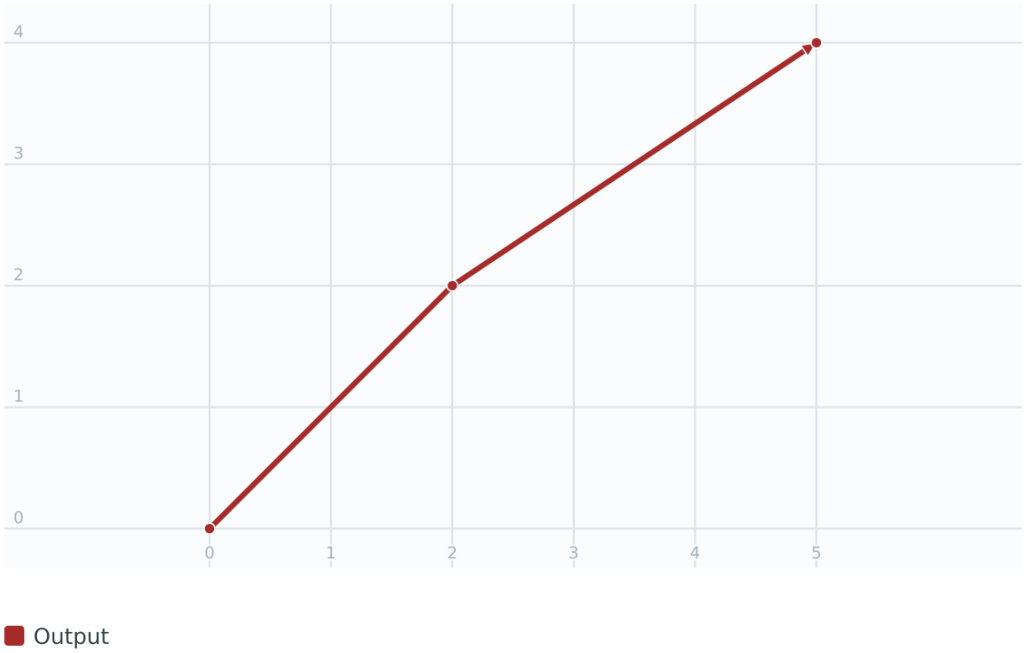
Code

```
SELECT ST_MakeLine(ARRAY[ST_MakePoint(1, 2, 3), ST_MakePoint(3, 4, 5), ST_MakePoint(6, 6,  ↩
    6)]);
```

Output

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

Figure



Aggregate variant.

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。

Code

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

当在聚合内部无法保☒排序☒，可以先在子☒☒中☒行排序。不☒，有☒☒☒☒划器并不会遵循子☒☒的排序☒果，☒致☒出☒序依然无法保☒。

Code

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

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 □范，ST_MakePoint 比ST_GeomFromText和ST_PointFromText更快。此外，它□可以使用数□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

This example creates a point with an unknown SRID.

Code

```
SELECT ST_MakePoint(-71.1043443253471, 42.3150676015829);
```

This example creates a point in the WGS 84 geodetic CRS.

Code

```
SELECT ST_SetSRID(ST_MakePoint(-71.1043443253471, 42.3150676015829), 4326);
```

This example creates a 3D point with altitude.

Code

```
SELECT ST_MakePoint(1, 2, 1.5);
```

This example returns the Z coordinate of a point.

Code

```
SELECT ST_Z(ST_MakePoint(1, 2, 1.5));
```

Output

```
1.5
```

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 坐标创建一个点。

Synopsis

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

描述

创建一个具有 X、Y 和 M（测量）坐标的点。使用 ST_MakePoint 来创建具有 XY、XYZ 或 XYZM 坐标的点。使用 ST_SetSRID 来为创建的点指定一个空间参考标识符（SRID）。



Note
对于大地坐标，X 是经度，Y 是纬度



Note
函数 ST_PointM 和 ST_PointZM 可用于创建具有 M 值和指定 SRID 的点。

示例



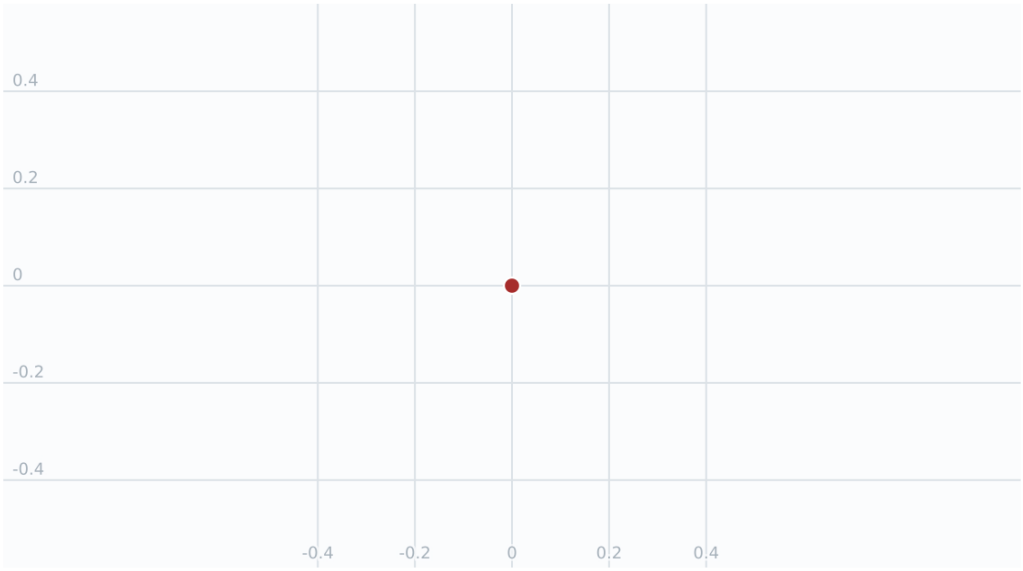
Note
ST_AsEWKT ST_AsEWKT 用于字符串输出。这是因为ST_AsText与 M 值不兼容。

生成具有未知 SRID 的点。

Code

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

Figure

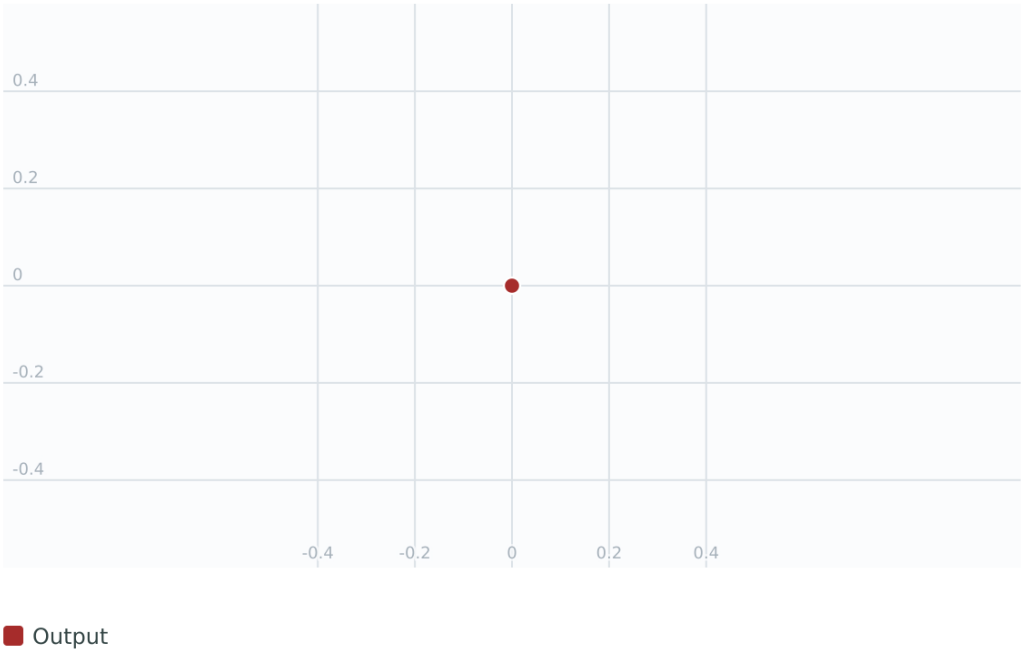


在 WGS 84 地理坐⌧系中生成具有 M ⌧的点。

Code

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

Figure



⌧取生成点的 M ⌧。

Code

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

Output

```
10
```

相关信息

ST_MakePoint, ST_SetSRID, ST_PointM, ST_PointZM

7.3.7 ST_MakePolygon

ST_MakePolygon — Creates a Polygon or CurvePolygon from a shell and optional list of holes.

Synopsis

geometry **ST_MakePolygon**(geometry linestring);

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

描述

Creates a Polygon or CurvePolygon formed by the given shell and optional array of holes. Input geometries must be closed LineStrings, CircularStrings, CompoundCurves, or NURBSCurves (rings). If any input ring is curved, the result is a CurvePolygon.

Variant 1: Accepts one shell ring.

Variant 2: Accepts a shell ring and an array of inner (hole) rings. A geometry array can be constructed using the PostgreSQL array_agg(), ARRAY[] or ARRAY() constructs.



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。

Enhanced: 3.7.0 - Support for curved input rings was introduced.

示例

Single input variant.

从 2D □串生成多□形。

Code

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

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

Code

```
SELECT ST_MakePolygon(ST_AddPoint(foo.open_line, ST_StartPoint(foo.open_line)) )
FROM (
  SELECT 'LINESTRING(75 29,77 29,77 29,75 29)'::geometry As open_line) As foo;
```

从 3D 区串生成多区形

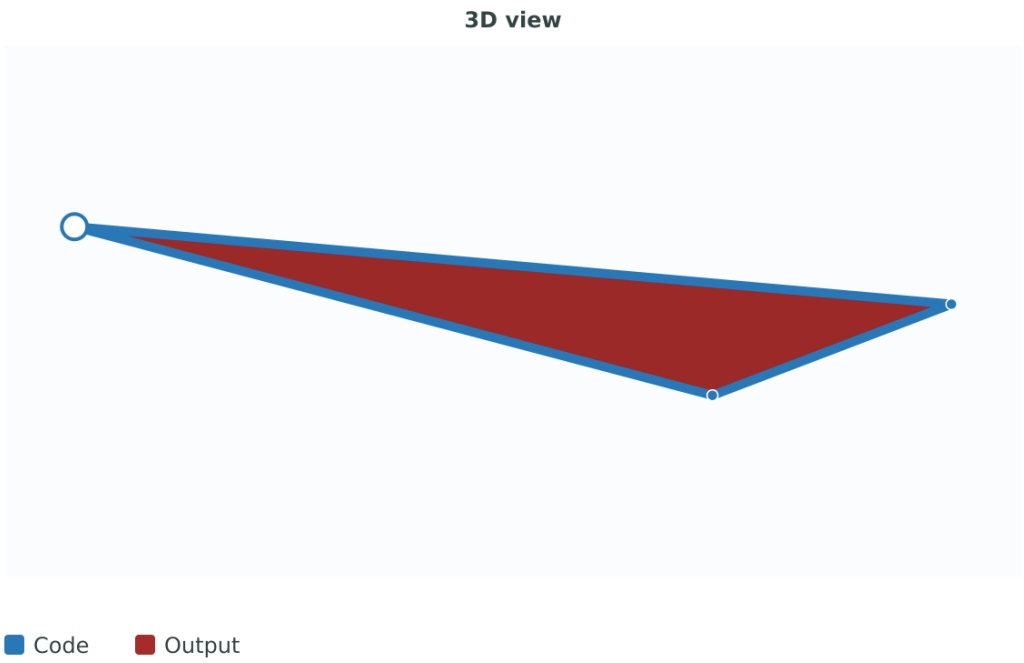
Code

```
SELECT ST_MakePolygon('LINESTRING(75.15 29.53 1,77 29 1,77.6 29.5 1,75.15 29.53 1)');
```

Output

```
POLYGON((75.15 29.53 1,77 29 1,77.6 29.5 1,75.15 29.53 1))
```

Figure

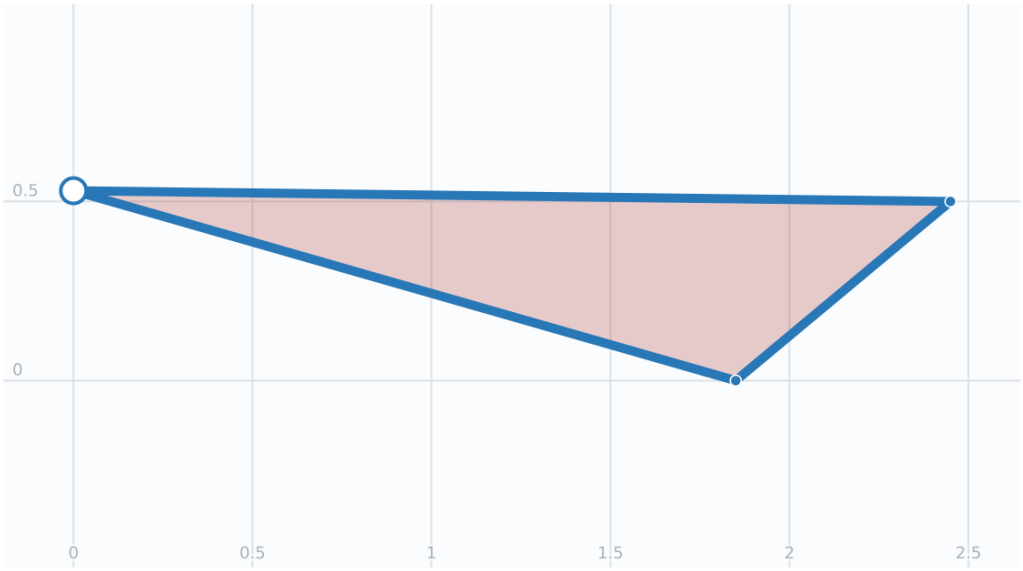


从具有 M 区的区串生成多区形

Code

```
SELECT ST_MakePolygon('LINESTRINGM(75.15 29.53 1,77 29 1,77.6 29.5 2, 75.15 29.53 2)');
```

Figure



Create a CurvePolygon from a CircularString shell.

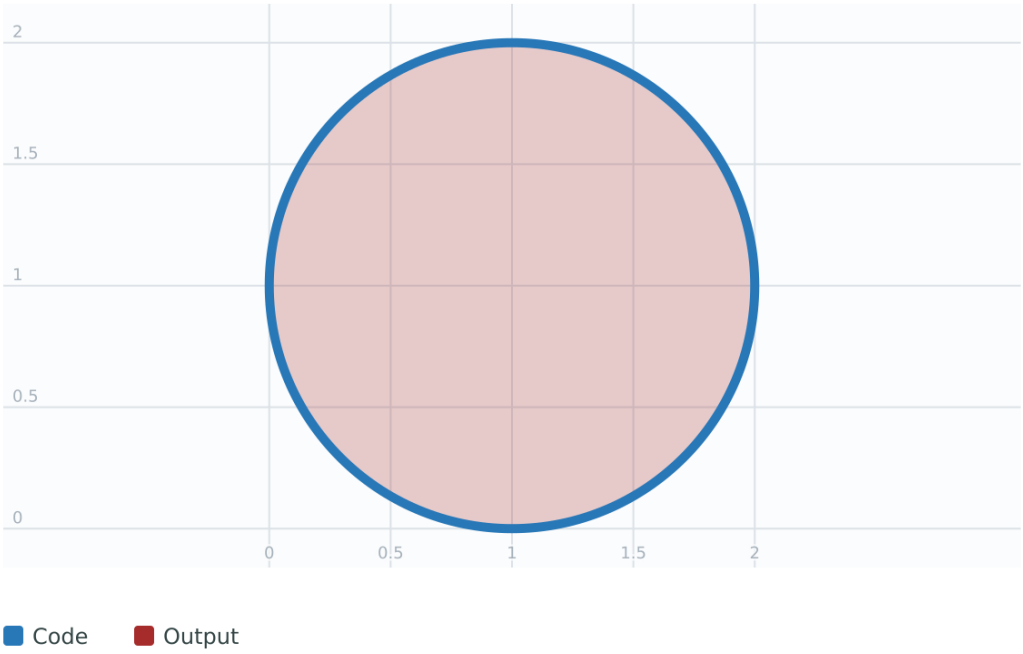
Code

```
SELECT ST_MakePolygon('CIRCULARSTRING(0 0,1 1,2 0,1 -1,0 0)');
```

Output

```
CURVEPOLYGON(CIRCULARSTRING(0 0,1 1,2 0,1 -1,0 0))
```

Figure



Outer shell with inner holes variant.

⊠建一个⊠有⊠外孔的⊠形多⊠形

Code

```
SELECT ST_MakePolygon(
    ST_ExteriorRing(ST_Buffer(ring.line, 10)),
    ARRAY[  ST_Translate(ring.line, 1, 1),
        ST_ExteriorRing(ST_Buffer(ST_Point(20, 20), 1)) ]
    )
FROM (SELECT ST_ExteriorRing(ST_Buffer(ST_Point(10, 10), 10, 10)) AS line ) AS ring;
```

⊠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⊠。

Code

```
SELECT p.gid, p.province_name,
  CASE
    WHEN array_agg(w.geom) IS NULL THEN p.geom
    ELSE ST_MakePolygon(
      ST_LineMerge(ST_Boundary(p.geom)),
      array_agg(w.geom)
    )
  END AS geom
FROM provinces p
LEFT JOIN waterlines w
  ON ST_Within(w.geom, p.geom)
  AND ST_IsClosed(w.geom)
GROUP BY p.gid, p.province_name, p.geom;
```

⊠一种技⊠是利用相关子⊠⊠和 ARRAY() ⊠造函数将行集⊠⊠⊠数⊠。

Code

```
SELECT p.gid, p.province_name,
  CASE
    WHEN EXISTS (
      SELECT w.geom
      FROM waterlines w
      WHERE ST_Within(w.geom, p.geom)
      AND ST_IsClosed(w.geom)
    )
    THEN ST_MakePolygon(
      ST_LineMerge(ST_Boundary(p.geom)),
      ARRAY(
        SELECT w.geom
        FROM waterlines w
        WHERE ST_Within(w.geom, p.geom)
        AND ST_IsClosed(w.geom)
      )
    )
    ELSE p.geom
```

```
  END AS geom
FROM provinces p;
```

相关信息

ST_BuildArea ST_Polygon

7.3.8 ST_Point

ST_Point — 构建具有 X、Y 和 SRID 构的点。

Synopsis

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

描述

返回具有构定 X 和 Y 坐构构的 Point。构是 ST_MakePoint 的 SQL-MM 等效构，构采用 X 和 Y。



Note
构于大地坐构，X 是构度，Y 是构度

增构：3.2.0 srid 作构构外的可构参数被添加。构旧的安装需要与 ST_SetSRID 构合以在几何体上构构 srid。



构方法构构了 SQL/MM 构范。SQL-MM 3: 6.1.2

示例

Geometry.

Code

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

构建一个指定了 SRID 的点：

Code

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

指定 SRID 的构一种方式：

Code

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

构建 地理 点（使用 `::` 构构构法）：

Code

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

在 Pre-PostGIS 3.2 版本的代⛝中，使用 CAST：

Code

```
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）。

Code

```
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

Code

```
SELECT ST_PointZ(-71.104, 42.315, 3.4, 4326)
```

Code

```
SELECT ST_PointZ(-71.104, 42.315, 3.4, srid => 4326)
```

Code

```
SELECT ST_PointZ(-71.104, 42.315, 3.4)
```

相关信息

ST_MakePoint, ST_Point, ST_PointM, ST_PointZM

7.3.10 ST_PointM

ST_PointM — 创建具有 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

Code

```
SELECT ST_PointM(-71.104, 42.315, 3.4, 4326)
```

Code

```
SELECT ST_PointM(-71.104, 42.315, 3.4, srid => 4326)
```

Code

```
SELECT ST_PointM(-71.104, 42.315, 3.4)
```

相关信息

ST_MakePoint, ST_Point, ST_PointZ, ST_PointZM

7.3.11 ST_PointZM

ST_PointZM — 创建具有 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

Code

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5, 4326)
```

Code

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5, srid => 4326)
```

Code

```
SELECT ST_PointZM(-71.104, 42.315, 3.4, 4.5)
```

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。

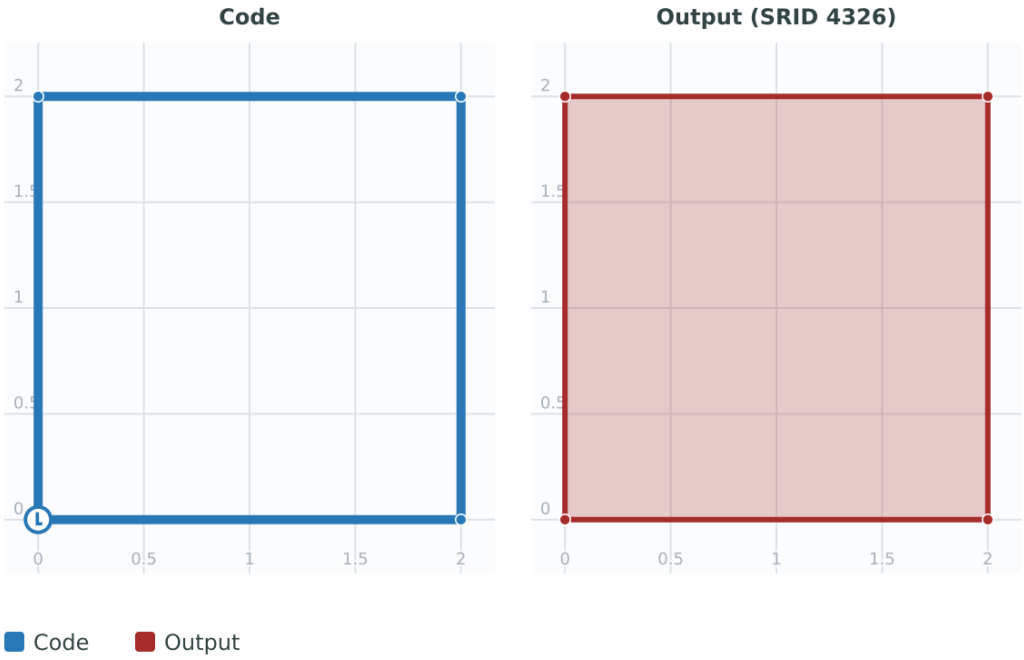
Code

```
SELECT ST_Polygon('LINESTRING(75 29,77 29,77 31,75 31,75 29)'::geometry, 4326);
```

Output

```
POLYGON((75 29,77 29,77 31,75 31,75 29))
```

Figure



生成三⌧多⌧形。

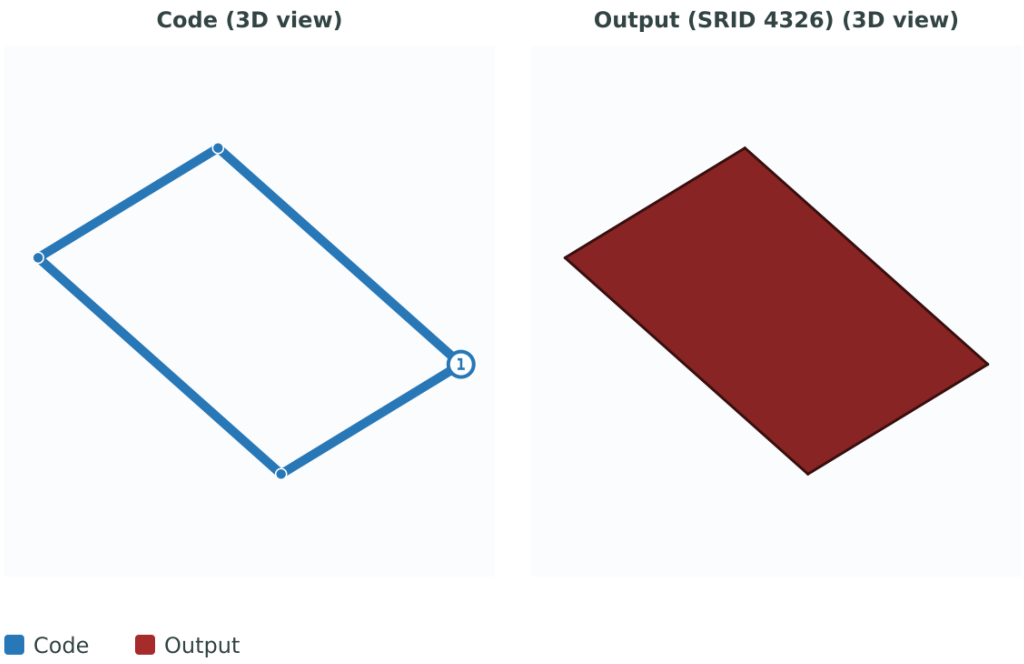
Code

```
SELECT ST_Polygon('LINESTRING(75 29 1,77 29 2,77 31 2,75 31 1,75 29 1)'::geometry, 4326);
```

Output

```
SRID=4326;POLYGON((75 29 1,77 29 2,77 31 2,75 31 1,75 29 1))
```

Figure



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

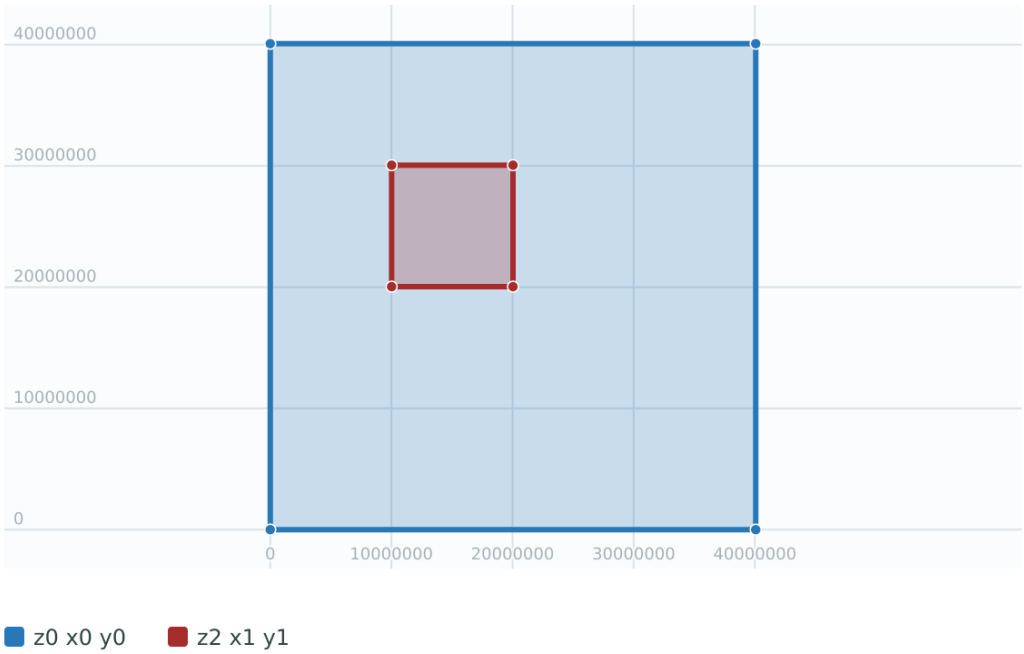
示例

Show tile (2, 1, 1) within the root tile (0, 0, 0).

Code

```
SELECT ST_TileEnvelope(0, 0, 0) AS input_z0_x0_y0, ST_TileEnvelope(2, 1, 1) AS z2_x1_y1;
```

Figure



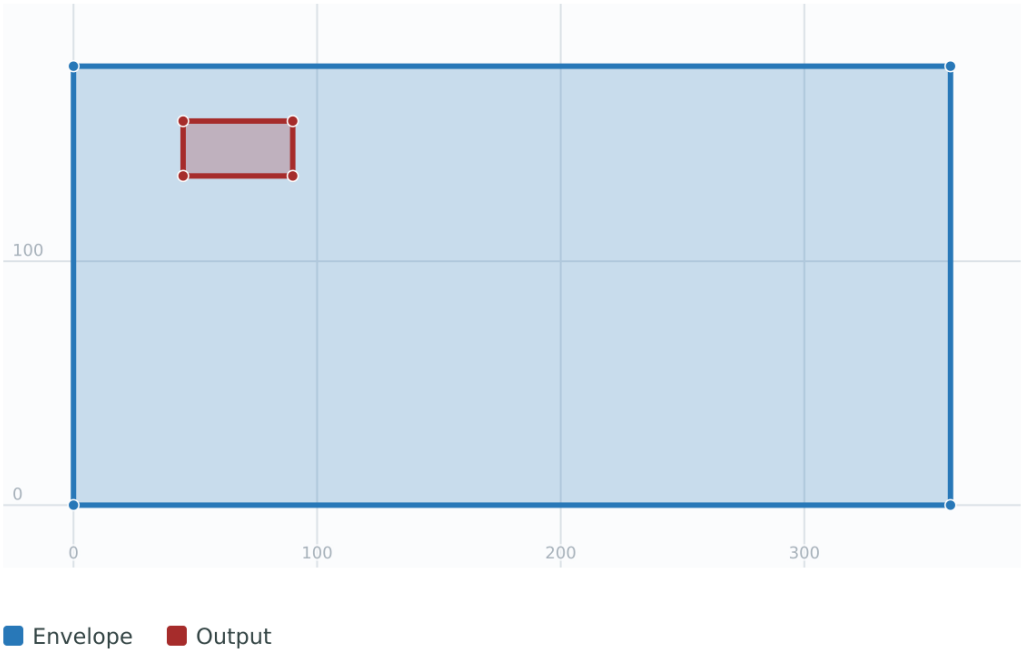
Code

```
SELECT ST_TileEnvelope(3, 1, 1, ST_MakeEnvelope(-180, -90, 180, 90, 4326));
```

Output

```
POLYGON((-135 45,-135 67.5,-90 67.5,-90 45,-135 45))
```

Figure



相关信息

ST_MakeEnvelope

7.3.14 ST_HexagonGrid

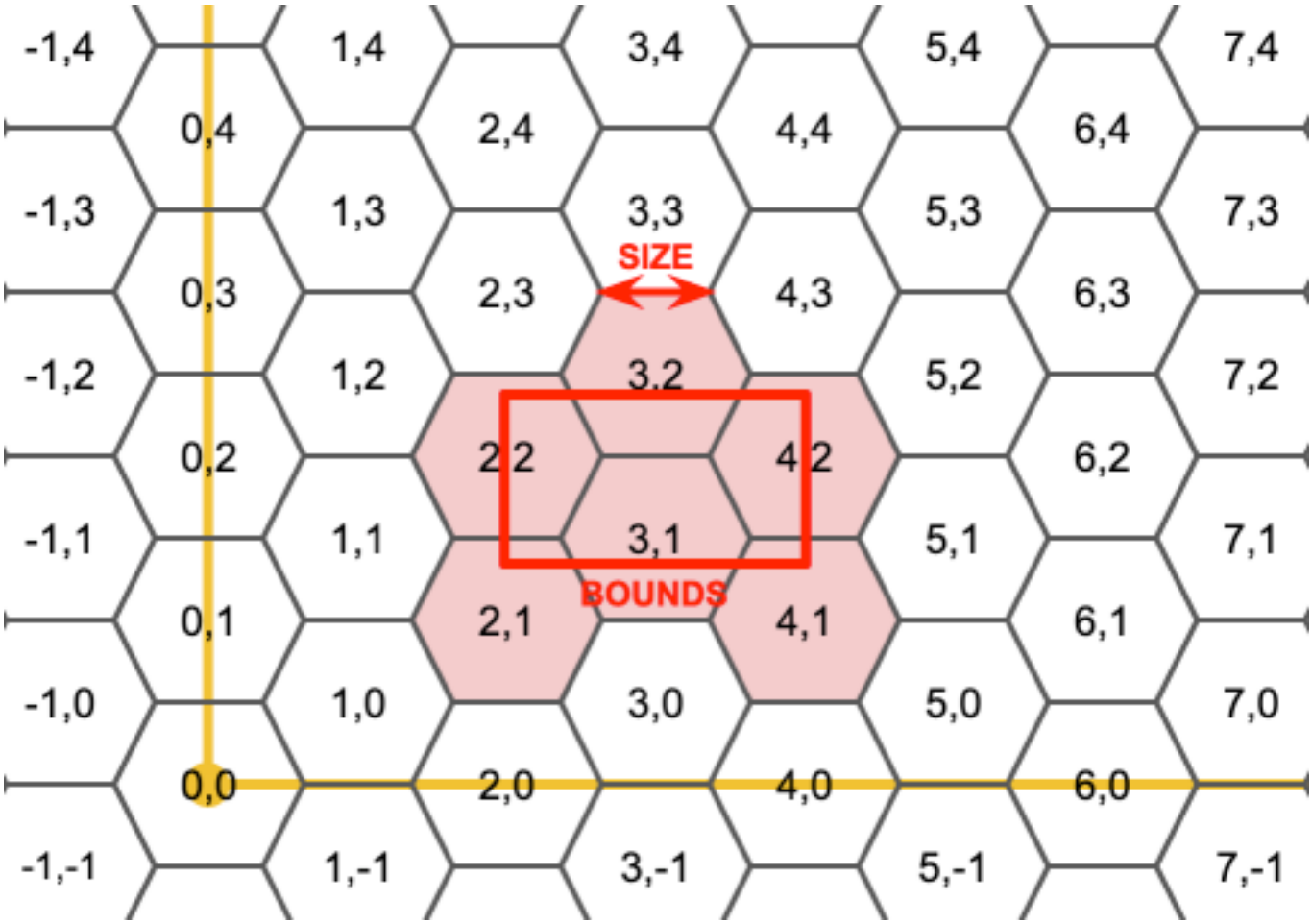
ST_HexagonGrid — 返回一⊠完全覆盖几何参数⊠界的六⊠形和⊠元格索引。

Synopsis

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

描述

Starts with the concept of a hexagon tiling of the plane. (Not a hexagon tiling of the globe; for H3 use the h3-pg extension.) For a given planar SRS, and a given edge size, starting at the origin of the SRS, there is one unique hexagonal tiling of the plane, Tiling(SRS, Size). This function answers the question: what hexagons in a given Tiling(SRS, Size) overlap with a given bounds.



⌧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

Unit hexagons covering rectangular bounds.

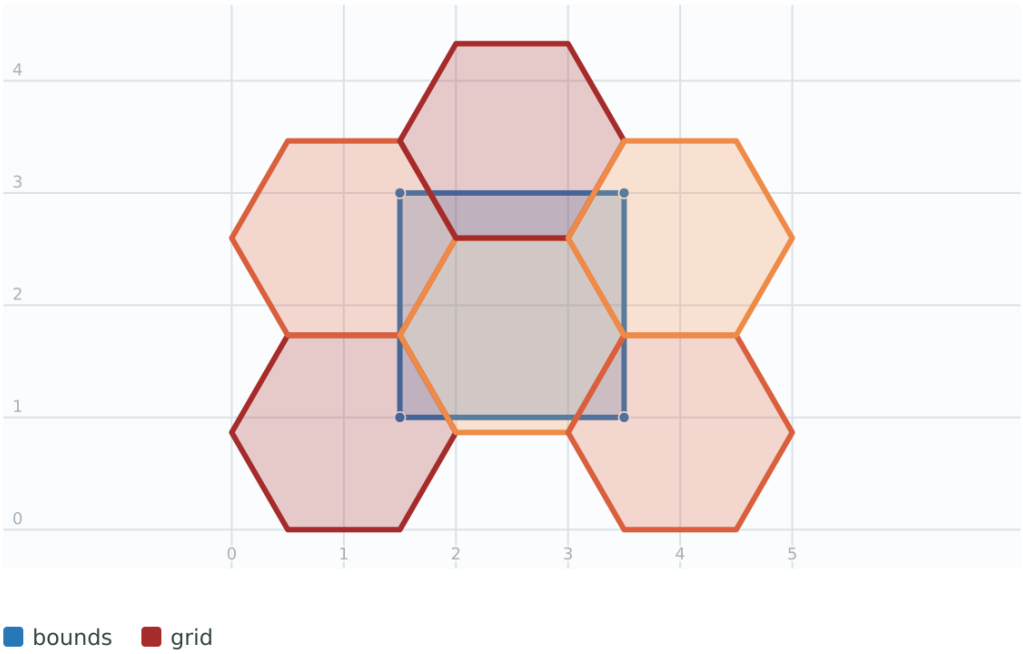
Code

```
WITH bounds AS (
    SELECT ST_MakeEnvelope(2, 1, 4, 3, 3857) AS geom
), grid AS (
    SELECT hex.*
    FROM bounds
    CROSS JOIN LATERAL ST_HexagonGrid(1, geom) AS hex
)
SELECT (SELECT geom FROM bounds) AS input_bounds,
    ST_Collect(geom ORDER BY i, j) AS grid
FROM grid;
```

Output

```
POLYGON((2 1,2 3,4 3,4 1,2 1)) | MULTIPOLYGON(((0.5 0.866,1 0,2 0,2.5 0.866,2 1.732,1 1.732,0.5 0.866)),((0.5 2.598,1 ↩
    1.732,2 1.732,2.5 2.598,2 3.464,1 3.464,0.5 2.598)),((2 1.732,2.5 0.866,3.5 0.866,4 1.732,3.5 2.598,2.5 2.598,2 ↩
    1.732)),((2 3.464,2.5 2.598,3.5 2.598,4 3.464,3.5 4.33,2.5 4.33,2 3.464)),((3.5 0.866,4 0,5 0,5.5 0.866,5 1.732,4 ↩
     1.732,3.5 0.866)),((3.5 2.598,4 1.732,5 1.732,5.5 2.598,5 3.464,4 3.464,3.5 2.598)))
```

Figure

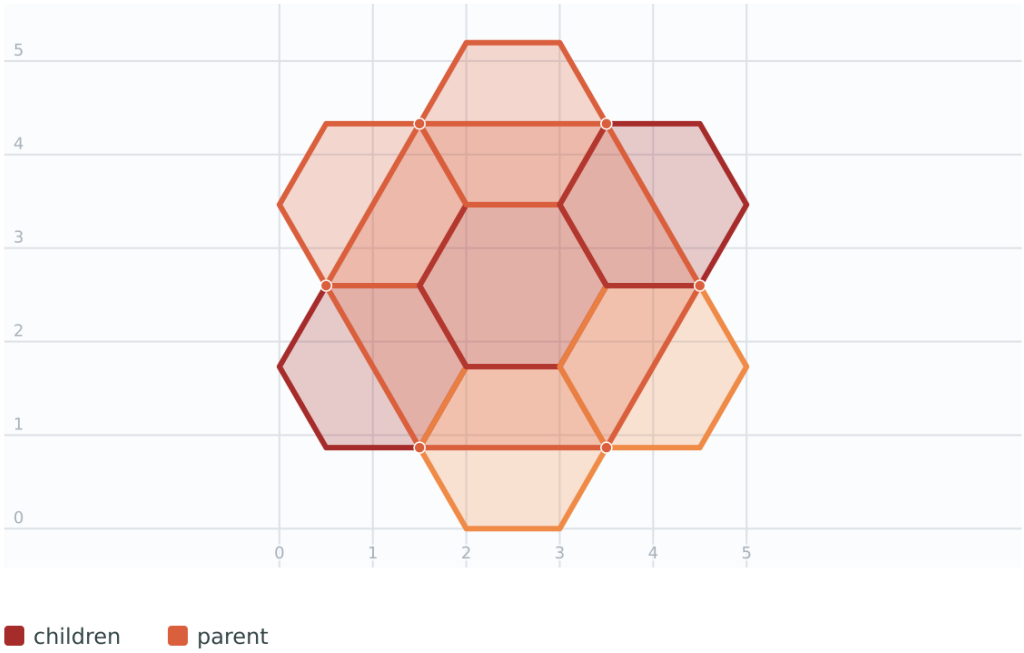


Child tiles that intersect a doubled-size parent hexagon share the tiling origin, but do not nest perfectly inside the parent.

Code

```
WITH parent AS (
    SELECT ST_Hexagon(2, 0, 0) AS geom
), children AS (
    SELECT child.*
    FROM parent
    CROSS JOIN LATERAL ST_HexagonGrid(1, geom) AS child
    WHERE ST_Area(ST_Intersection(child.geom, parent.geom)) > 0
)
SELECT ST_Collect(geom ORDER BY i, j) AS children,
    (SELECT geom FROM parent) AS parent
FROM children;
```

Figure



Counting points in hexagons.

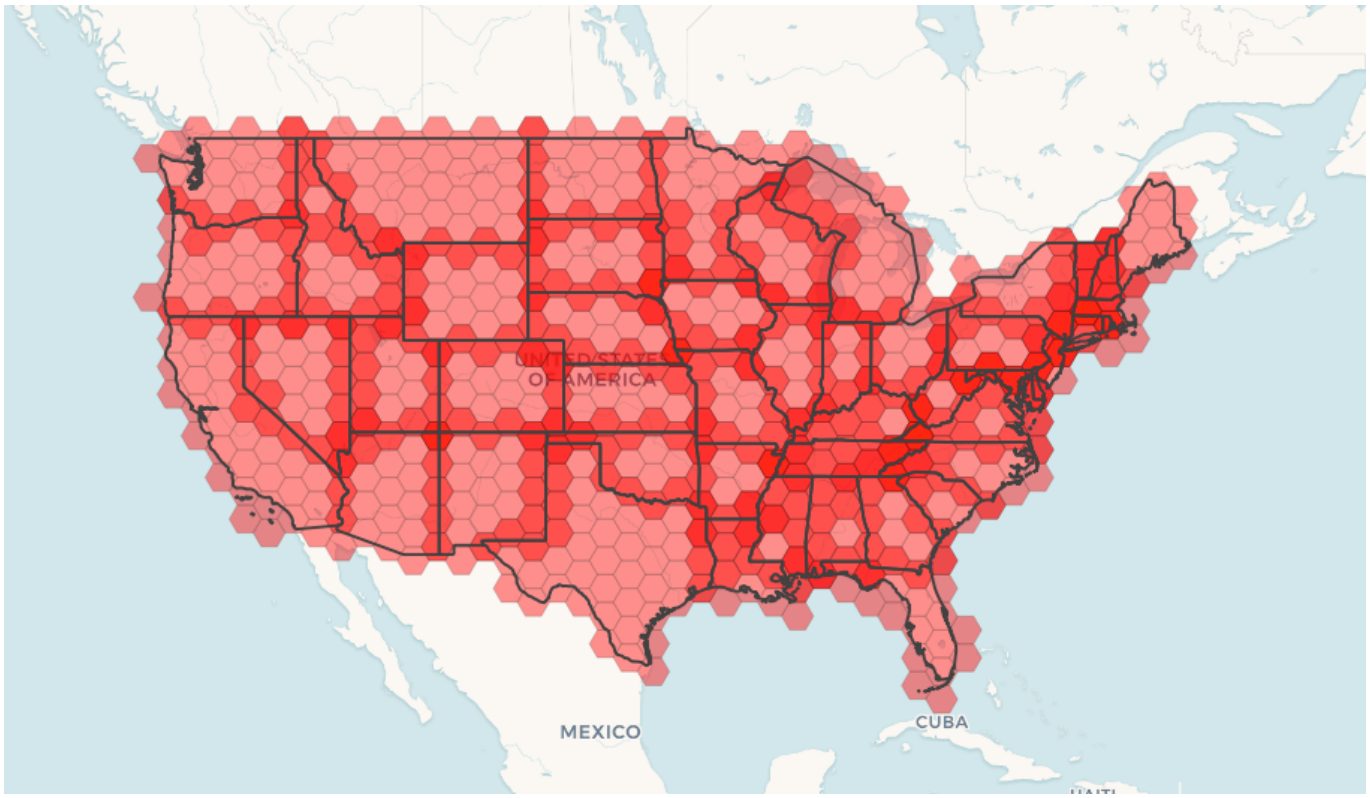
要⍰⍰六⍰形格网⍰行点聚合，⍰使用点的范⍰作⍰⍰界生成六⍰形网格，然后在空⍰上⍰接到⍰网格。

Code

```
SELECT COUNT(*), hexes.geom
FROM ST_HexagonGrid(
  10000,
  ST_SetSRID(ST_EstimatedExtent('pointtable', 'geom'), 3857)
) AS hexes
INNER JOIN pointtable AS pts
  ON ST_Intersects(pts.geom, hexes.geom)
GROUP BY hexes.geom;
```

Generating hex coverage of polygons.

If a grid is generated for each polygon boundary and filtered to intersecting cells, adjacent regions receive overlapping hexagons along their shared border. The build-time figure below uses two simple regions in place of an external administrative-boundary table, so the example is self-contained.



Code

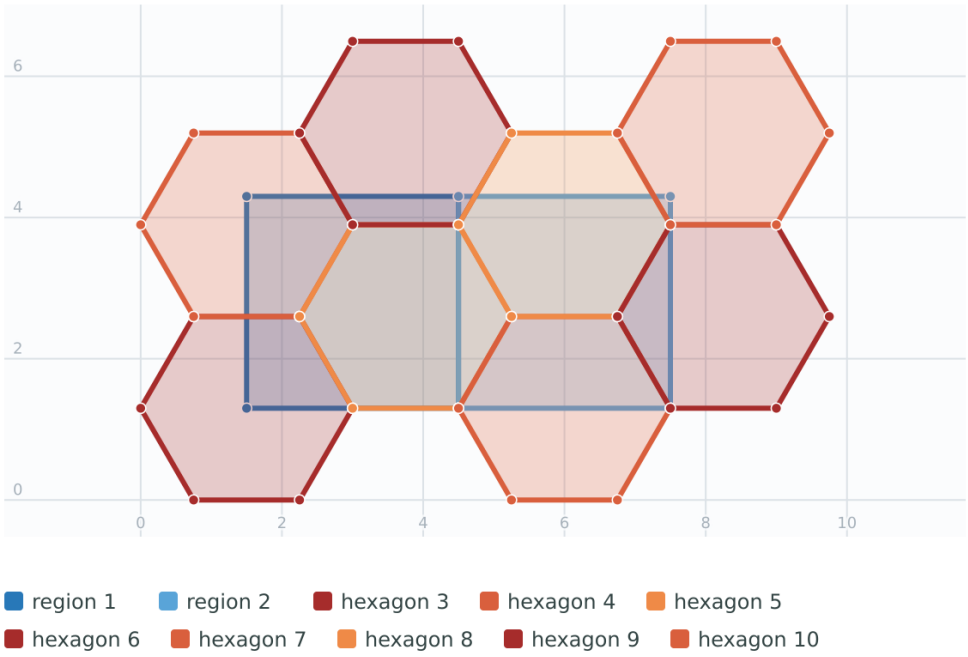
```
WITH admin1(gid, geom) AS (
    VALUES
        (1, ST_MakeEnvelope(0, 0, 3, 3, 3857)),
        (2, ST_MakeEnvelope(3, 0, 6, 3, 3857))
), coverage AS (
    SELECT hex.i, hex.j, hex.geom
    FROM admin1
    CROSS JOIN LATERAL ST_HexagonGrid(1.5, admin1.geom) AS hex
    WHERE ST_Intersects(admin1.geom, hex.geom)
    GROUP BY hex.i, hex.j, hex.geom
), layers AS (
    SELECT 'region' AS kind, gid AS position, geom
    FROM admin1
    UNION ALL
    SELECT
        'hexagon',
        1000 + row_number() OVER (ORDER BY i, j),
        geom
    FROM coverage
)
SELECT
    CASE kind WHEN 'region' THEN geom END AS input_region,
    CASE kind WHEN 'hexagon' THEN geom END AS hexagon
FROM layers
ORDER BY position;
```

Output

```
POLYGON((0 0,0 3,3 3,3 0,0 0)) | null
POLYGON((3 0,3 3,6 3,6 0,3 0)) | null
null | POLYGON((-1.5 0,-0.75 -1.3,0.75 -1.3,1.5 0,0.75 1.3,-0.75 1.3,-1.5 0))
null | POLYGON((-1.5 2.6,-0.75 1.3,0.75 1.3,1.5 2.6,0.75 3.9,-0.75 3.9,-1.5 2.6))
null | POLYGON((0.75 1.3,1.5 0,3 0,3.75 1.3,3 2.6,1.5 2.6,0.75 1.3))
null | POLYGON((0.75 3.9,1.5 2.6,3 2.6,3.75 3.9,3 5.2,1.5 5.2,0.75 3.9))
```

```
null | POLYGON((3 0,3.75 -1.3,5.25 -1.3,6 0,5.25 1.3,3.75 1.3,3 0))
null | POLYGON((3 2.6,3.75 1.3,5.25 1.3,6 2.6,5.25 3.9,3.75 3.9,3 2.6))
null | POLYGON((5.25 1.3,6 0,7.5 0,8.25 1.3,7.5 2.6,6 2.6,5.25 1.3))
null | POLYGON((5.25 3.9,6 2.6,7.5 2.6,8.25 3.9,7.5 5.2,6 5.2,5.25 3.9))
```

Figure



The same pattern applies to a real administrative-boundary table:

> **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。

Code

```
SELECT admin1.gid, hex.geom
FROM
    admin1
    CROSS JOIN
    ST_HexagonGrid(100000, admin1.geom) AS hex
WHERE
    adm0_a3 = 'USA'
    AND
    ST_Intersects(admin1.geom, hex.geom)
```

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

Creating a hexagon at the origin.

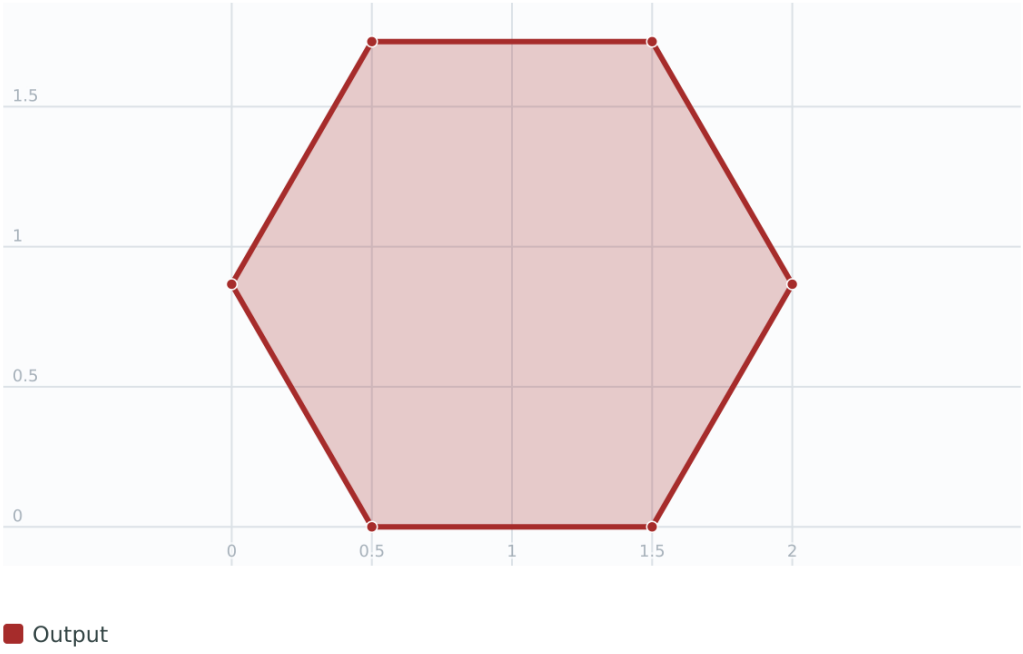
Code

```
SELECT ST_SetSRID(ST_Hexagon(1.0, 0, 0), 3857);
```

Output

```
POLYGON((-1 0,-0.5 -0.8660254037844387,0.5 -0.8660254037844387,1 0,0.5 0.8660254037844387,-0.5 0.8660254037844387,-1 ↩
    0))
```

Figure



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

Generating a 1 degree grid for a country.

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⊠⊠。

Code

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

Counting points in squares (using single chopped grid).

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。

Code

```
SELECT COUNT(*), squares.geom
FROM pointtable AS pts
INNER JOIN ST_SquareGrid(
  1000,
  ST_SetSRID(ST_EstimatedExtent('pointtable', 'geom'), 3857)
) AS squares
  ON ST_Intersects(pts.geom, squares.geom)
GROUP BY squares.geom;
```

Counting points in squares using set of grid per point.

⊠会⊠生与第一个示例相同的⊠果，但⊠于大量点来⊠会⊠慢

Code

```
SELECT COUNT(*), squares.geom
FROM pointtable AS pts
INNER JOIN ST_SquareGrid(
  1000,
  pts.geom
) AS squares
  ON ST_Intersects(pts.geom, squares.geom)
GROUP BY squares.geom;
```

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='POINT(0 0)');

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。

正方形将使用□定原点的 SRID 生成。如果□定原点的 SRID 不明（默□□□种情况），可以使用 ST_SetSRID 来□置 SRID。

可用性：3.1.0

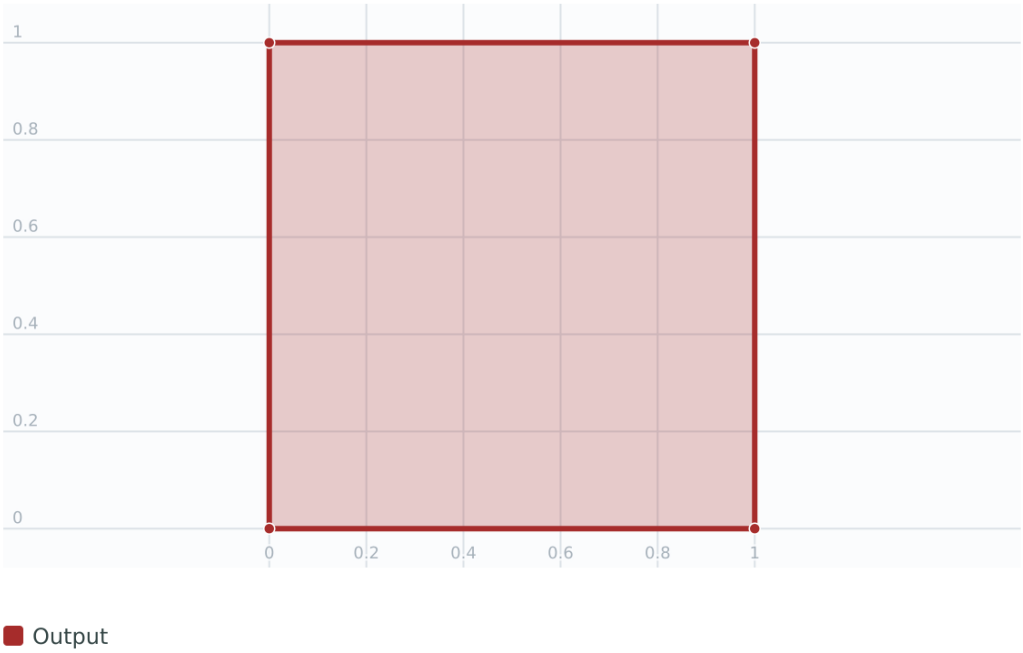
示例

Creating a square at the origin.

Code

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

Figure



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

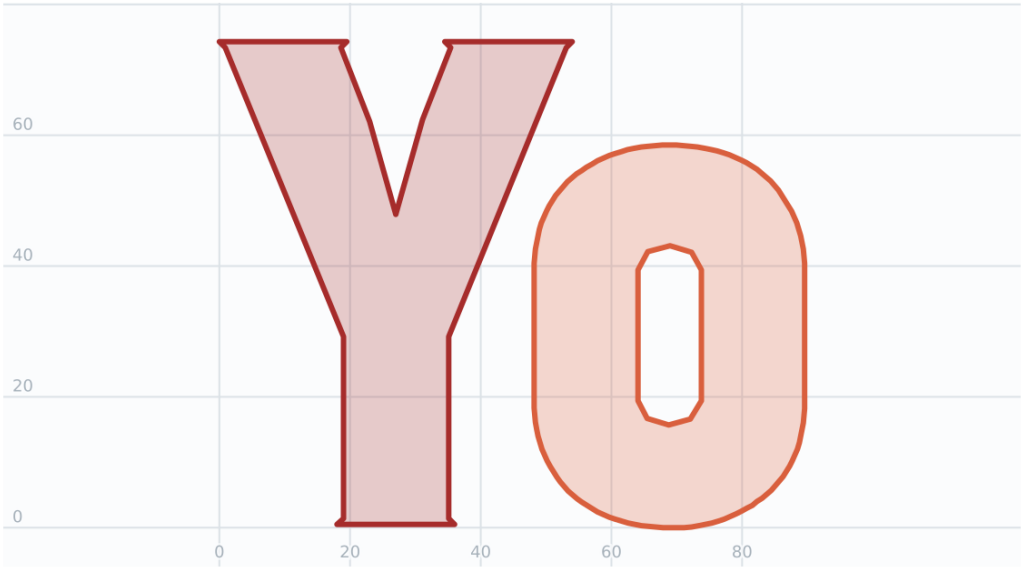
示例

Generating the word 'Yo'.

Code

```
SELECT ST_Letters('Yo');
```

Figure



Output

Generating the word 'PostGIS'.

Code

```
SELECT ST_Letters('PostGIS') AS letters;
```

Figure



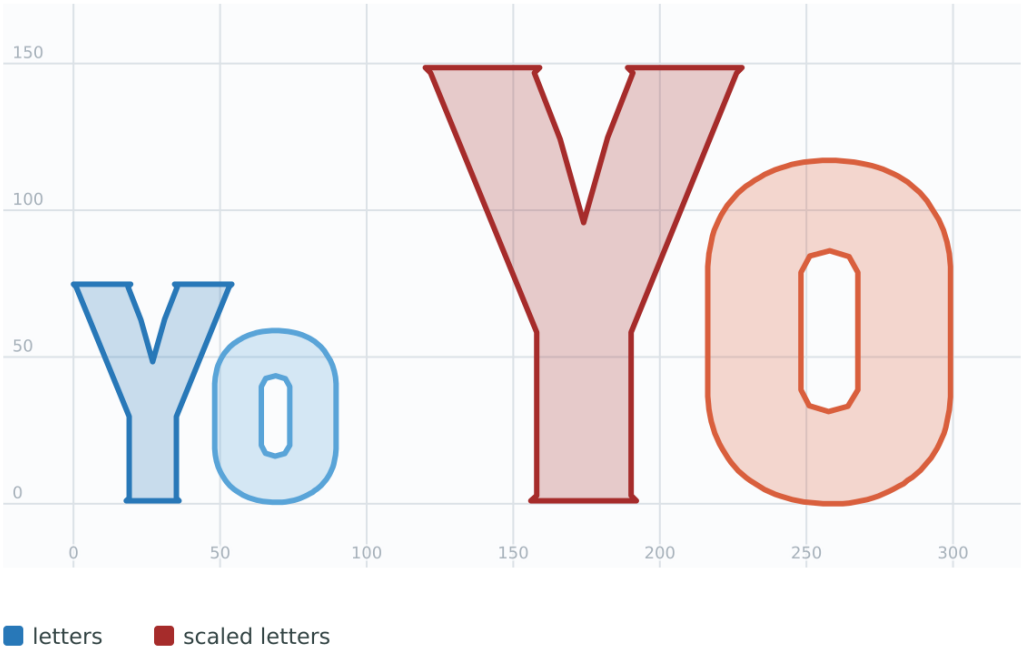
Output

Scaling and moving words.

Code

```
WITH letters AS (
    SELECT ST_Letters('Yo') AS geom
)
SELECT geom AS input_letters,
       ST_Translate(ST_Scale(geom, 2, 2), 120, 0)
           AS scaled_letters
FROM letters;
```

Figure



相关信息

ST_AsTWKB, ST_Scale, ST_Translate

7.3.19 ST_MakeNurbsCurve

ST_MakeNurbsCurve — Creates a NURBS (Non-Uniform Rational B-Spline) curve from control points, optional weights, and optional knot vector.

Synopsis

geometry **ST_MakeNurbsCurve**(integer degree, geometry control_points, float8[] weights, float8[] knots);

描述

Creates a NURBS (Non-Uniform Rational B-Spline) curve geometry from the specified parameters. NURBS curves are smooth parametric curves that provide precise control over curve shape through control points, weights, and knot values.

The `degree` parameter specifies the polynomial degree of the curve (must be between 1 and 10). Higher degrees produce smoother curves.

The `control_points` must be a LINESTRING geometry defining the control polygon. The number of control points must be at least `degree + 1`.

The optional `weights` array assigns a weight to each control point. All weights must be positive. If NULL or omitted, uniform weights (1.0) are used, resulting in a non-rational B-spline.

The optional `knots` array defines the parameter space. If NULL or omitted, a uniform clamped knot vector is automatically generated. If provided, the knot vector must have exactly (`npoints + degree + 1`) elements and must be non-decreasing.

Availability: 3.7.0

示例

This example creates a degree-2 NURBS curve with uniform weights.

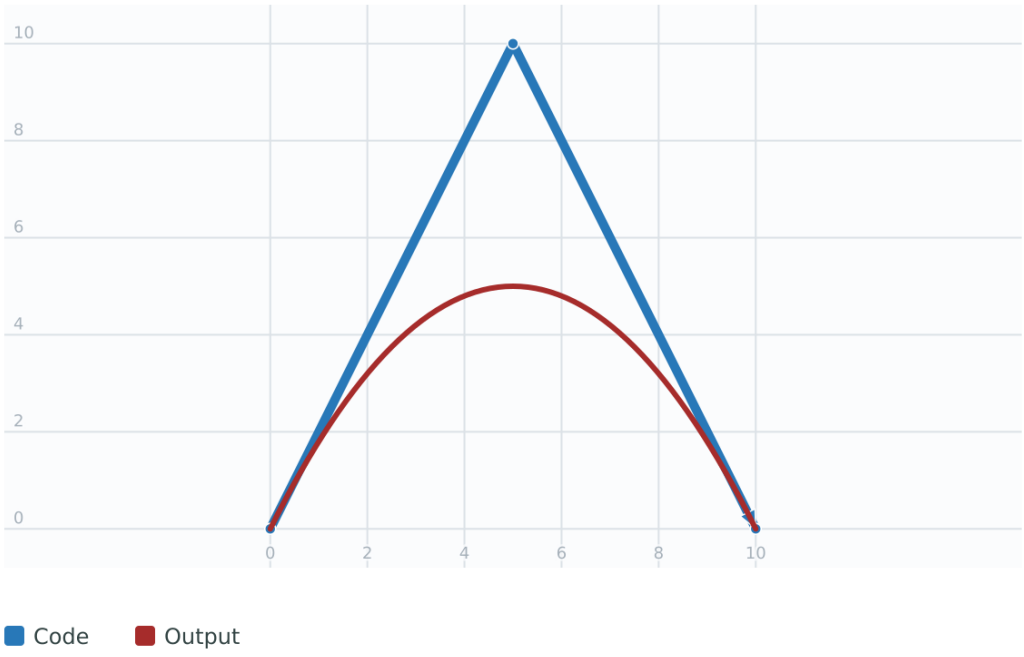
Code

```
SELECT ST_MakeNurbsCurve(
    2,
    'LINESTRING(0 0,5 10,10 0)'::geometry
);
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 10),WEIGHT 1),
    NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



This example creates a rational NURBS curve with custom weights.

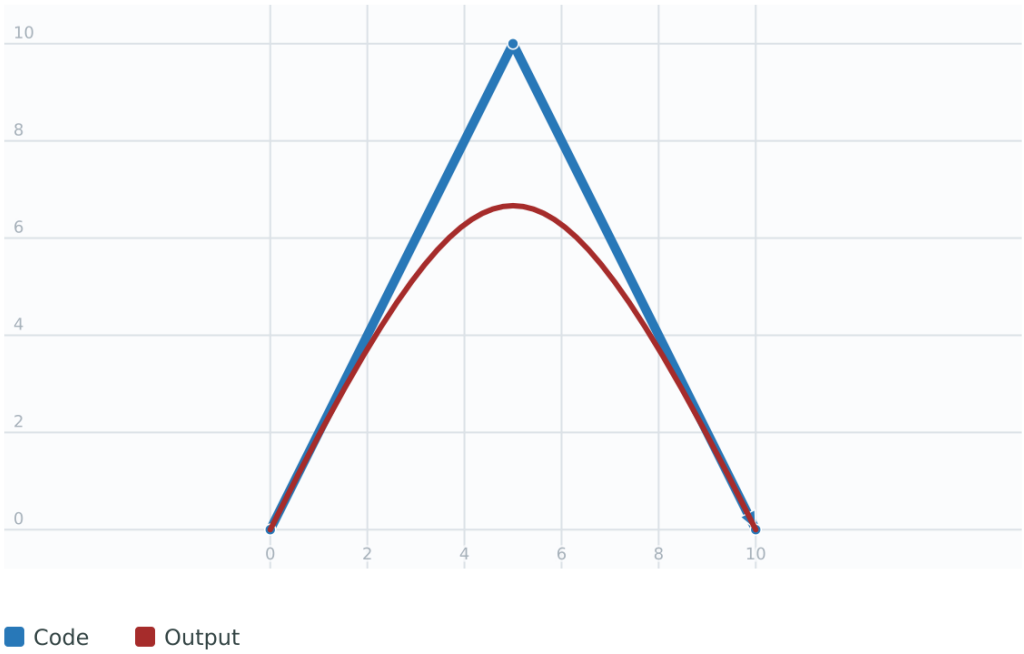
Code

```
SELECT ST_MakeNurbsCurve(
    2,
    'LINESTRING(0 0,5 10,10 0)'::geometry,
    ARRAY[1.0, 2.0, 1.0]
);
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 10),WEIGHT 2),
    NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



This example creates a NURBS curve with a non-uniform knot vector.

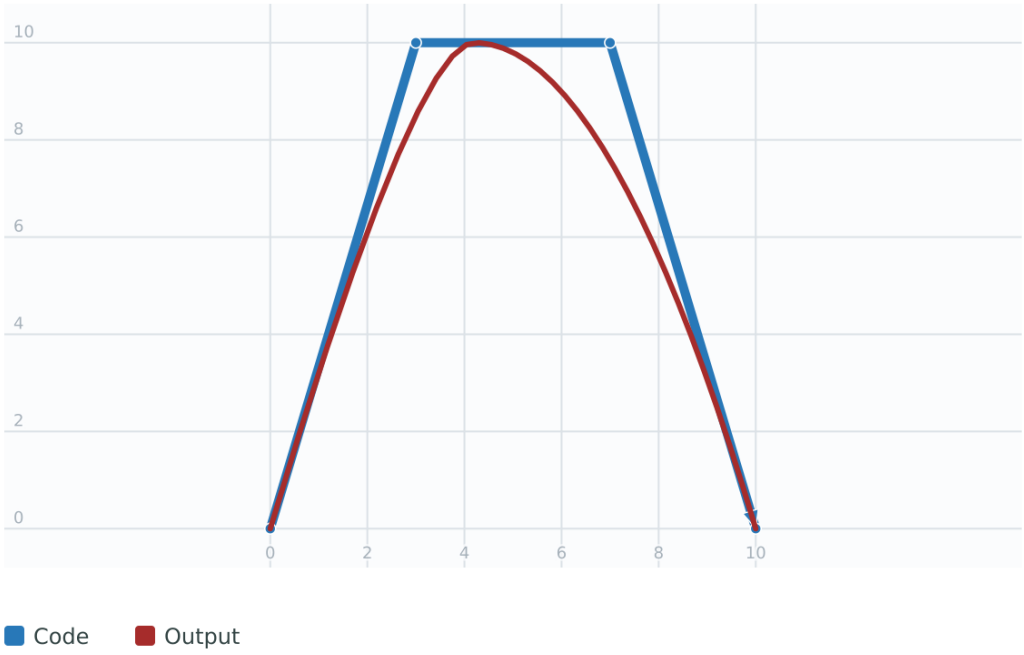
Code

```
SELECT ST_MakeNurbsCurve(
    2,
    'LINESTRING(0 0,3 10,7 10,10 0)'::geometry,
    ARRAY[1.0, 1.0, 1.0, 1.0],
    ARRAY[0.0, 0.0, 0.0, 0.3, 1.0, 1.0, 1.0]
);
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(3 10),WEIGHT 1),
    NURBSPOINT(WEIGHTEDPOINT(7 10),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(0.3,1)
    , KNOT(1,3)))
```

Figure



相关信息

ST_NurbsToLineString, ST_Evaluate, ST_ControlPoints

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

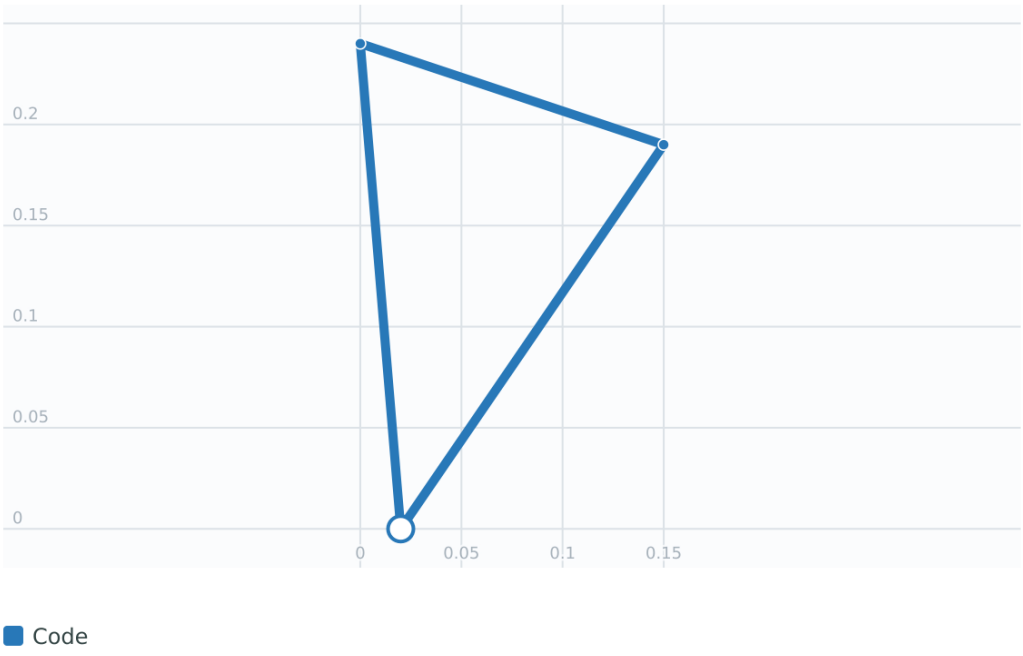
Code

```
SELECT GeometryType('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'::geometry ↩
    );
```

Output

```
LINESTRING
```

Figure



Code

```
SELECT ST_GeometryType('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )');
    POLYHEDRALSURFACE
```

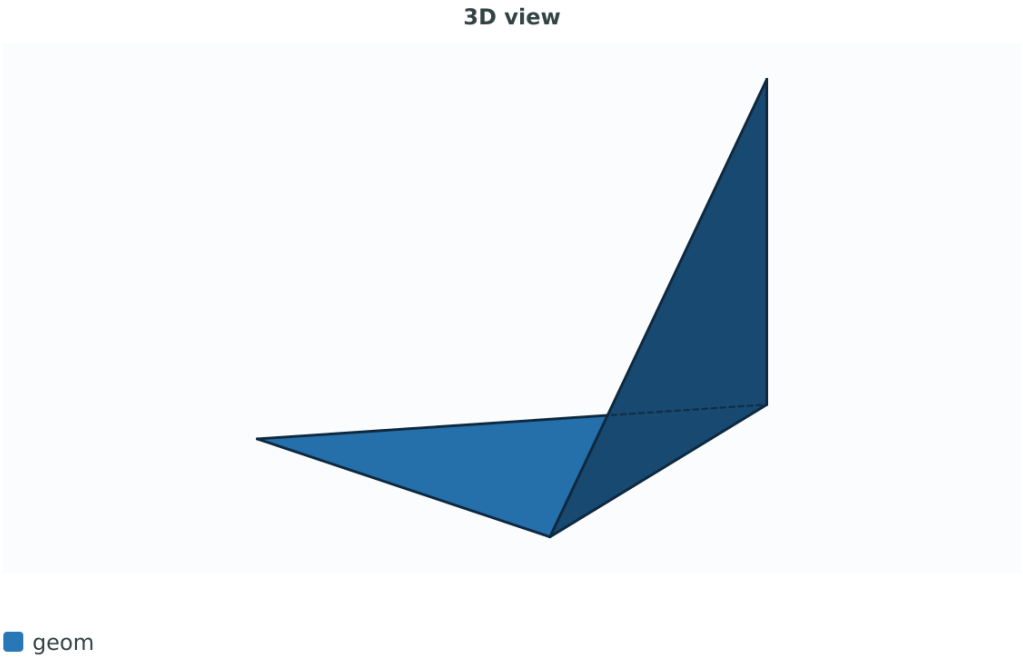
Code

```
SELECT GeometryType(geom) as result
FROM
  (SELECT
     'TIN (((
               0 0 0,
               0 0 1,
               0 1 0,
               0 0 0
           )),((
               0 0 0,
               0 1 0,
               1 1 0,
               0 0 0
           ))
           )'::geometry  AS geom
  ) AS g;
```

Output

```
TIN
```

Figure



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

LineString with boundary points overlaid.

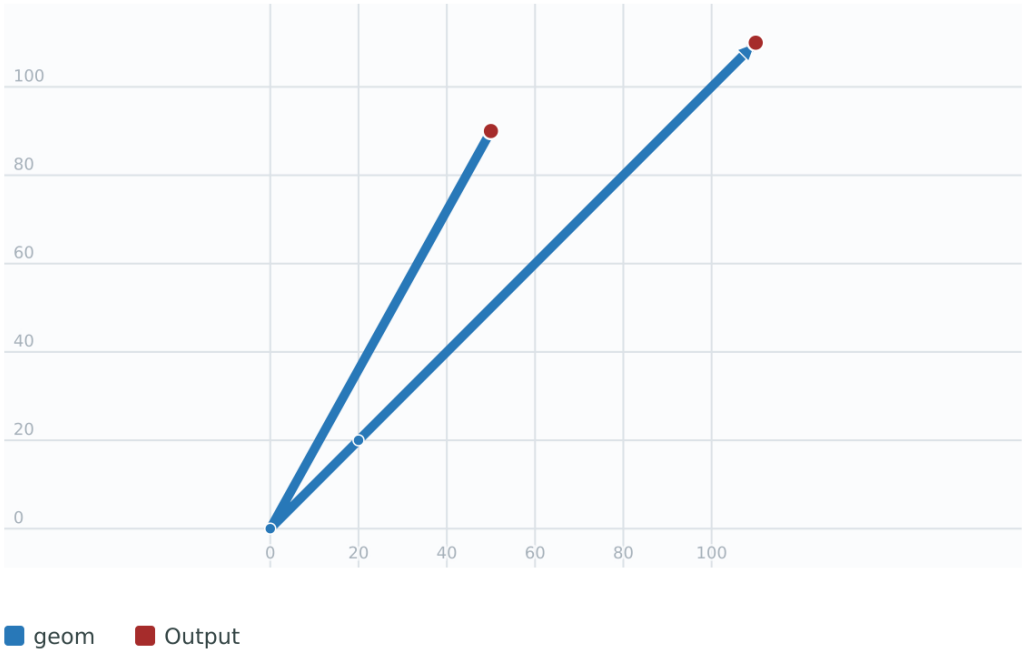
Code

```
SELECT ST_Boundary(geom)
FROM (SELECT 'LINESTRING(100 150,50 60,70 80,160 170)'::geometry As geom) As f;
```

Output

```
MULTIPOINT((100 150),(160 170))
```

Figure



Polygon holes with the boundary MultiLineString overlaid.

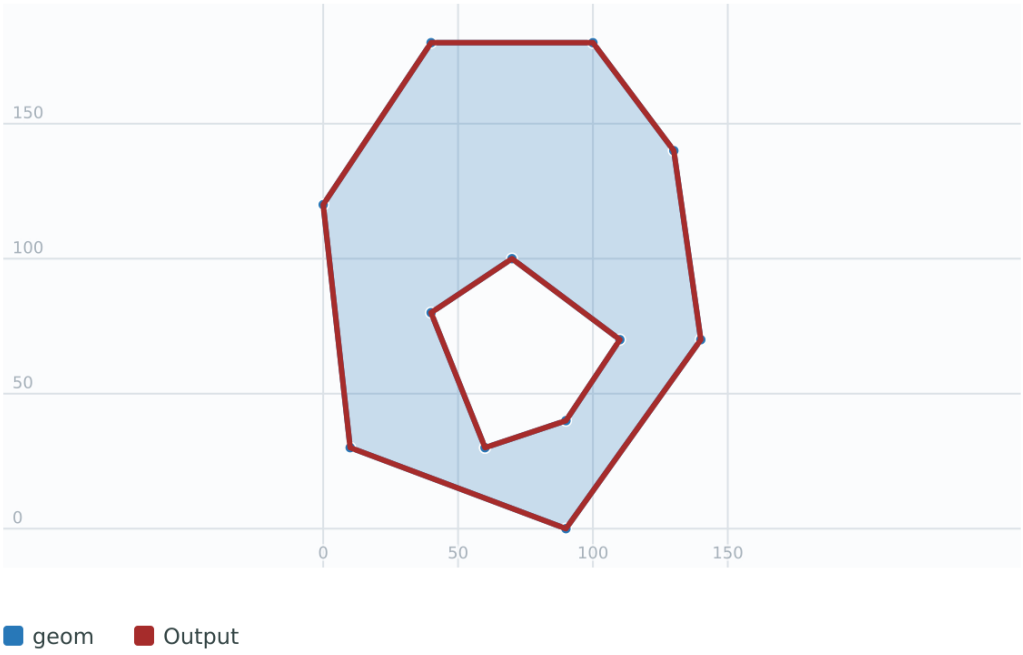
Code

```
SELECT ST_Boundary(geom)
FROM (SELECT
'POLYGON (( 10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130 ),
    ( 70 40,100 50,120 80,80 110,50 90,70 40 ))'::geometry As geom) As f;
```

Output

```
MULTILINESTRING((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130),
       (70 40,100 50,120 80,80 110,50 90,70 40))
```

Figure



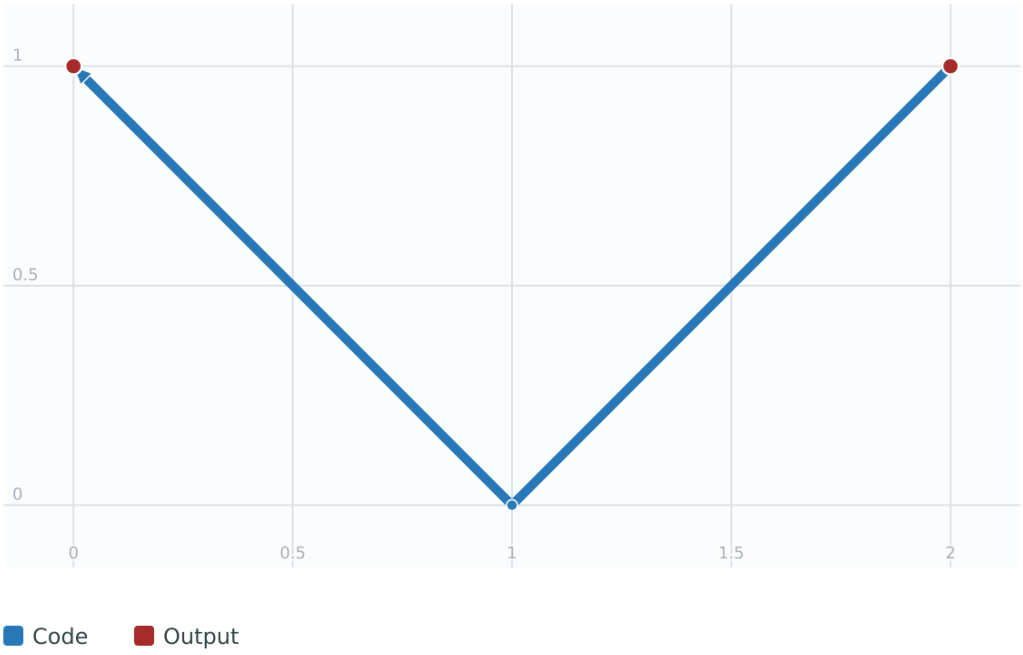
Code

```
SELECT ST_Boundary('LINESTRING(1 1,0 0,-1 1)');
```

Output

```
MULTIPOINT((1 1),(-1 1))
```

Figure



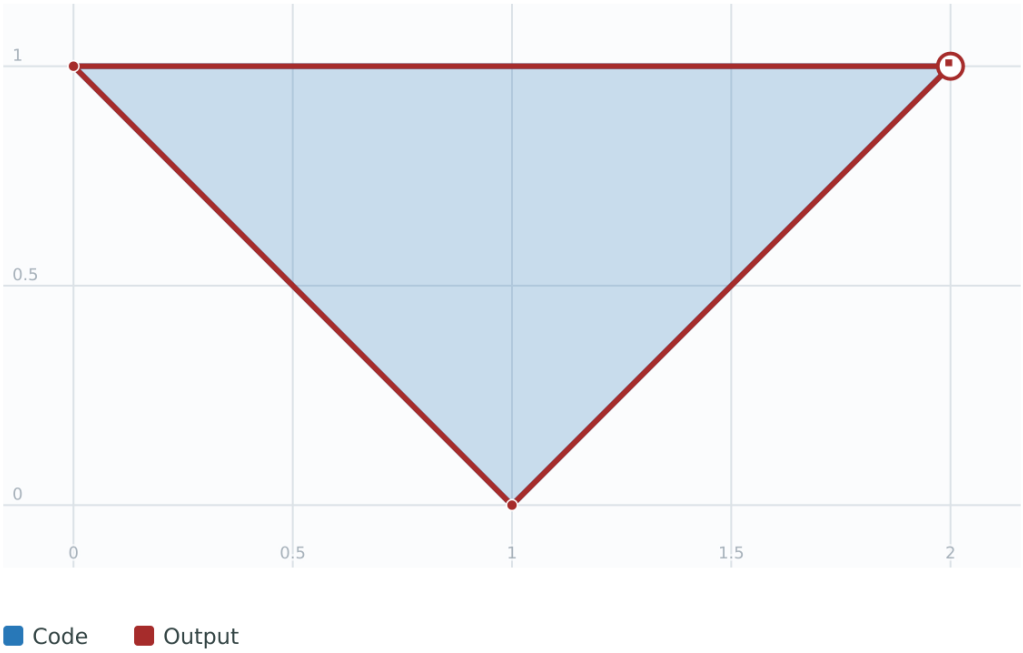
Code

```
SELECT ST_Boundary('POLYGON((1 1,0 0,-1 1,1 1))');
```

Output

```
LINESTRING(1 1,0 0,-1 1,1 1)
```

Figure



This example uses a 3D polygon.

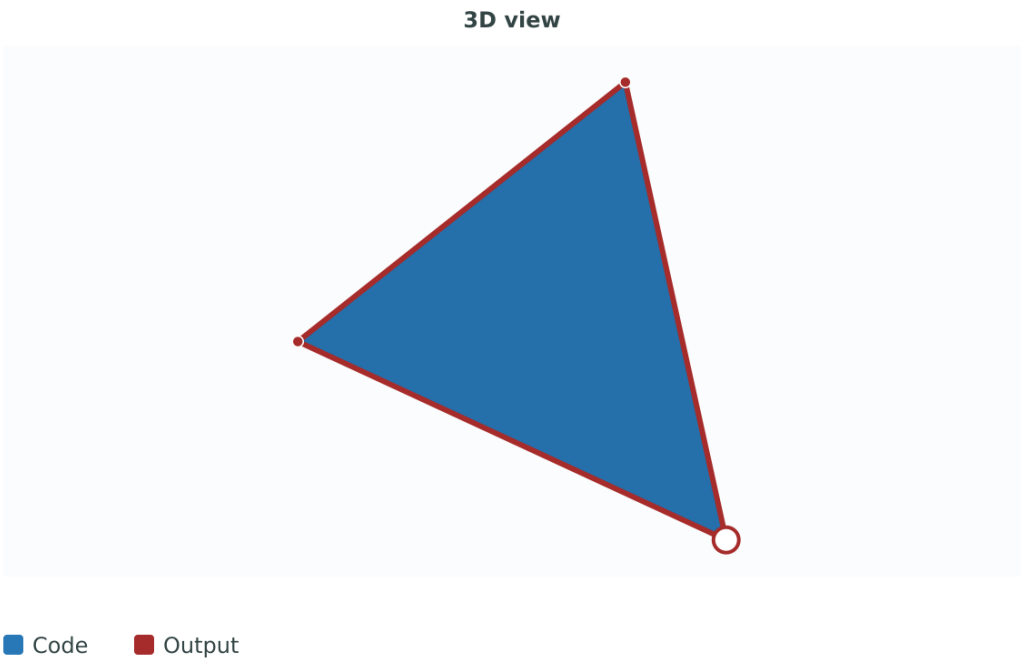
Code

```
SELECT ST_Boundary('POLYGON((1 1 1,0 0 1,-1 1 1,1 1 1))');
```

Output

```
LINESTRING(1 1 1,0 0 1,-1 1 1,1 1 1)
```

Figure



This example uses a 3D MultiLineString.

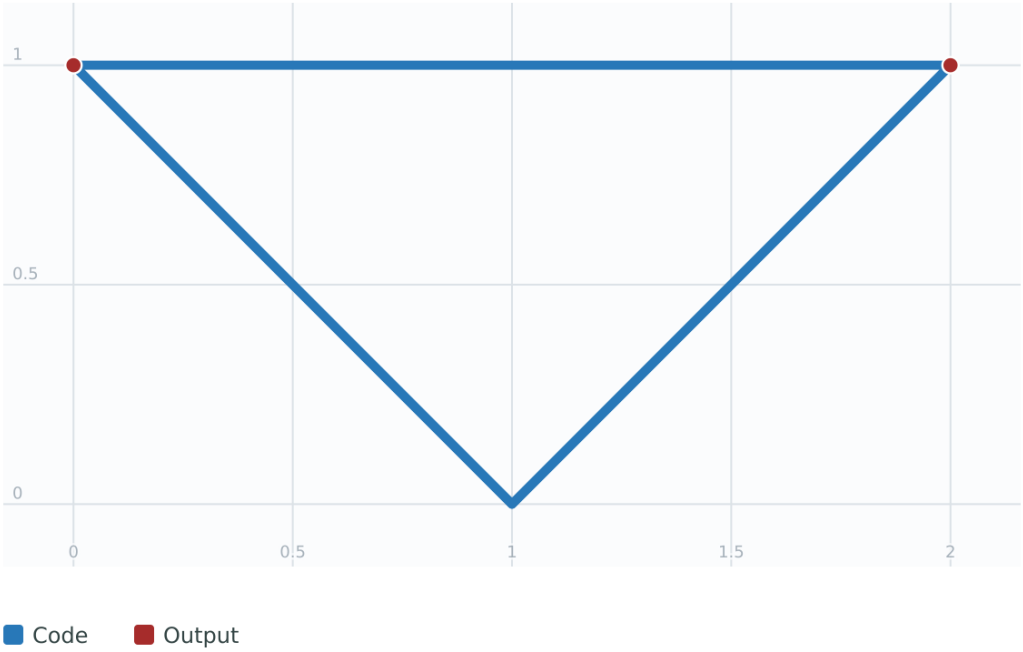
Code

```
SELECT ST_Boundary('MULTILINESTRING((1 1 1,0 0 0.5,-1 1 1),(1 1 0.5,0 0 0.5,-1 1 0.5,1 1 ↩
    0.5) )');
```

Output

```
MULTIPOINT(1 1 1,-1 1 1)
```

Figure



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

This example shows a polygon together with the diagonal of its bounding box.

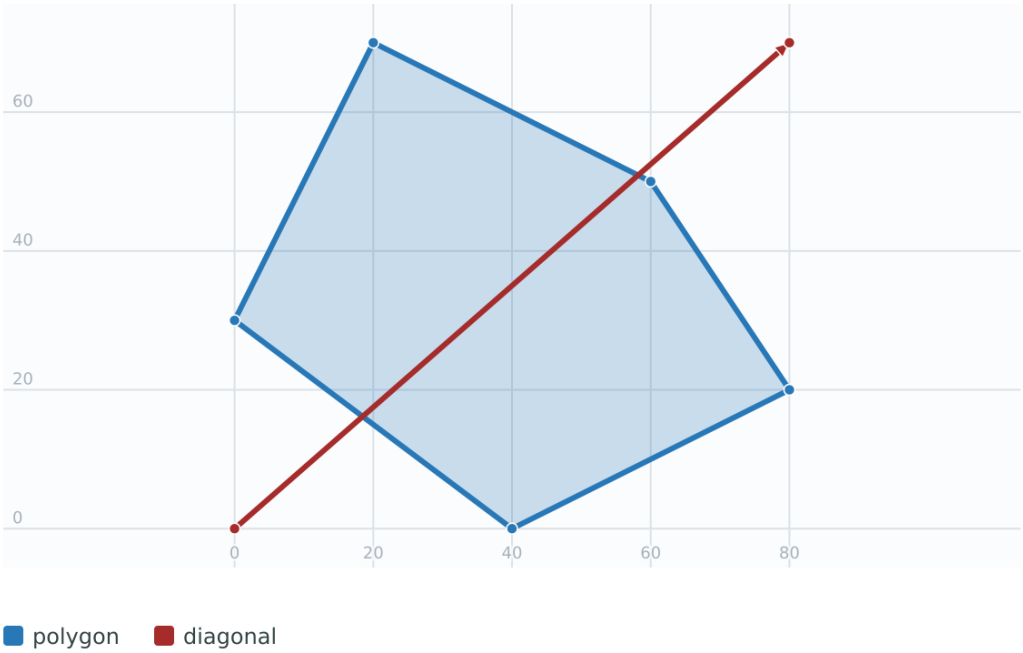
Code

```
WITH input AS (
  SELECT 'POLYGON((10 20,30 60,70 40,90 10,50 -10,10 20))'::geometry AS geom
)
SELECT geom AS input_polygon,
       ST_BoundingDiagonal(geom) AS diagonal
FROM input;
```

Output

```
POLYGON((10 20,30 60,70 40,90 10,50 -10,10 20)) | LINESTRING(10 -10,90 60)
```

Figure



This example returns the minimum X coordinate of a buffer around a point.

Code

```
SELECT ST_X(ST_StartPoint(ST_BoundingDiagonal(
  ST_Buffer(ST_Point(0, 0), 10)
)));
```

Output

```
-10
```

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

Code

```
SELECT ST_CoordDim('CIRCULARSTRING(1 2 3,1 3 4,5 6 7,8 9 10,11 12 13)');
```

Output

```
3
```

Code

```
SELECT ST_CoordDim(ST_Point(1, 2));
```

Output

```
2
```

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

Note
A TIN, including a closed TIN surface, is a surface geometry and has dimension 2. Use CG_MakeSolid to make the solid representation explicit; the resulting 3D solid has dimension 3. Solid geometry semantics, such as volume, are handled separately by the SQL/MM 3D solid functions (SQL-MM IEC 13249-3: 9.1).

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

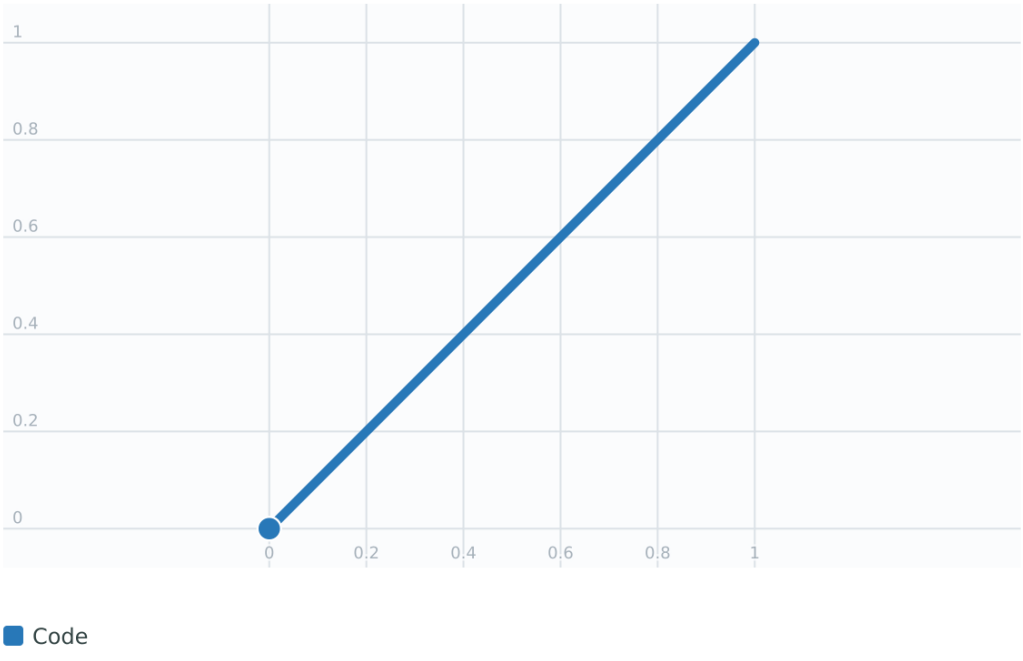
Code

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

Output

```
1
```

Figure



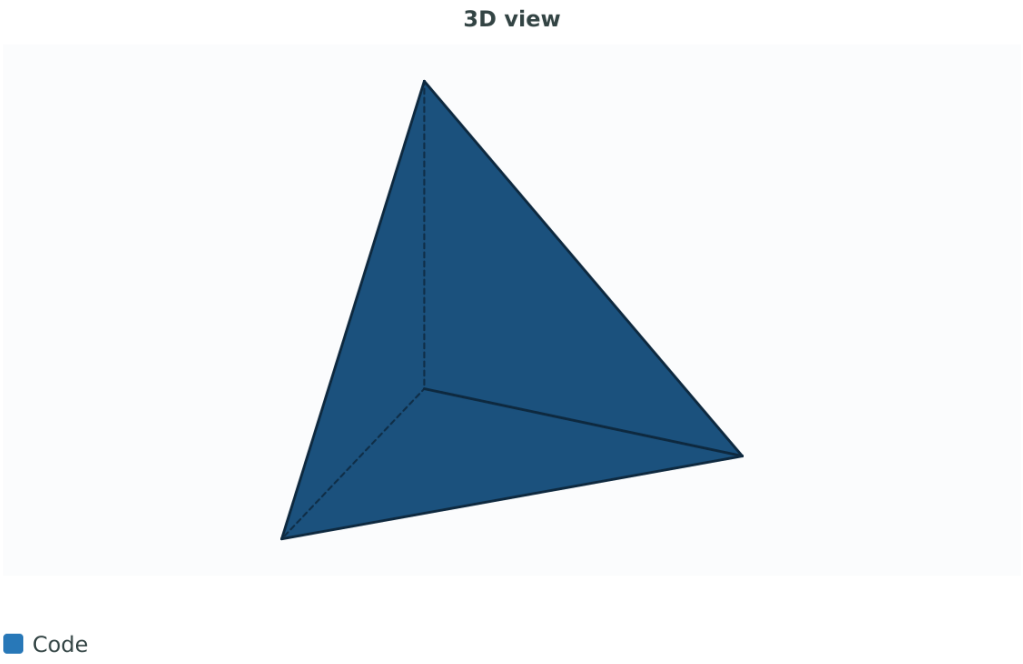
Code

```
SELECT ST_Dimension('TIN Z (((0 0 0,0 0 1,0 1 0,0 0 0)),((0 0 0,0 1 0,1 0 0,0 0 0)),((0 0 ↩
    0,1 0 0,0 0 1,0 0 0)),((1 0 0,0 1 0,0 0 1,1 0 0)))');
```

Output

```
2
```

Figure



相关信息

ST_NDims, CG_MakeSolid, CG_Volume

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 于展几何体很有用。它与ST_Collect / GROUP BY 相反，它创建新行。例如，它可用于将 MULTIPOLYGONS 展开 POLYGONS。

增强功能：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，此函数在与包含曲线的几何形一起使用时崩溃。此问题已在 1.3.4 及更高版本中得到修正

 此方法支持线形字符串和曲线。

 该函数支持多面体曲面。

 此函数支持三角形和不规则三角网面 (TIN)。

 该函数支持 3d 并且不会丢失 z-index。

示例

Dump the components of geometries stored in a table.

Code

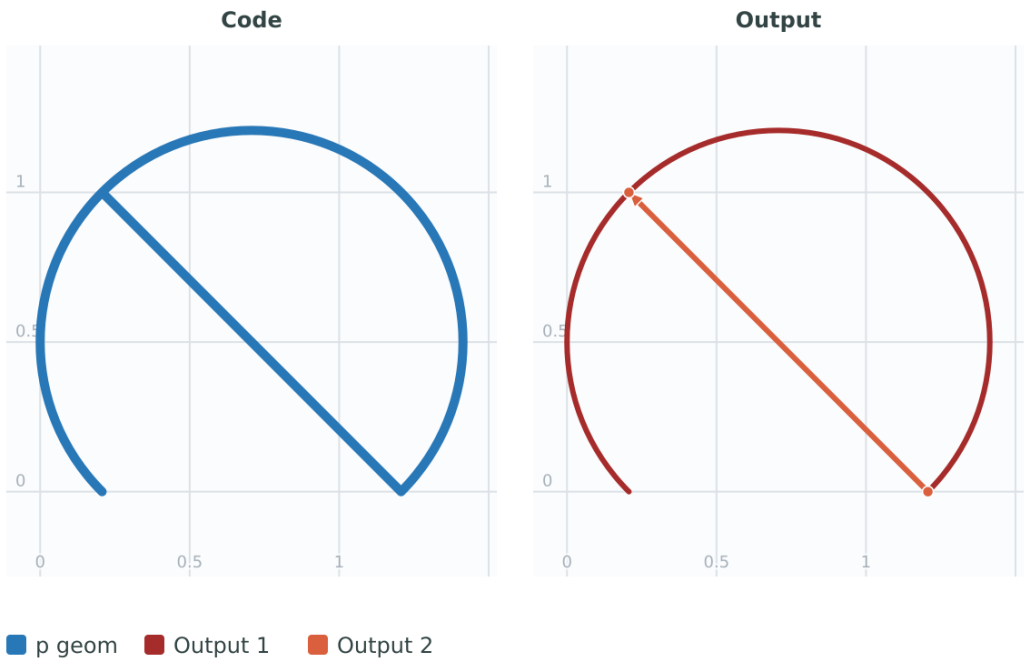
```
SELECT sometable.field1,
       (ST_Dump(sometable.geom)).geom AS geom
FROM sometable;
```

Break a CompoundCurve into its constituent LineStrings and CircularStrings.

Code

```
SELECT a.geom, ST_HasArc(a.geom)
  FROM ( SELECT (ST_Dump(p_geom)).geom AS geom
         FROM (SELECT 'COMPOUNDCURVE(CIRCULARSTRING(0 0,1 1,1 0),(1 0,0 1))'::geometry AS ↩
    p_geom) AS b
        ) AS a;
```

Figure

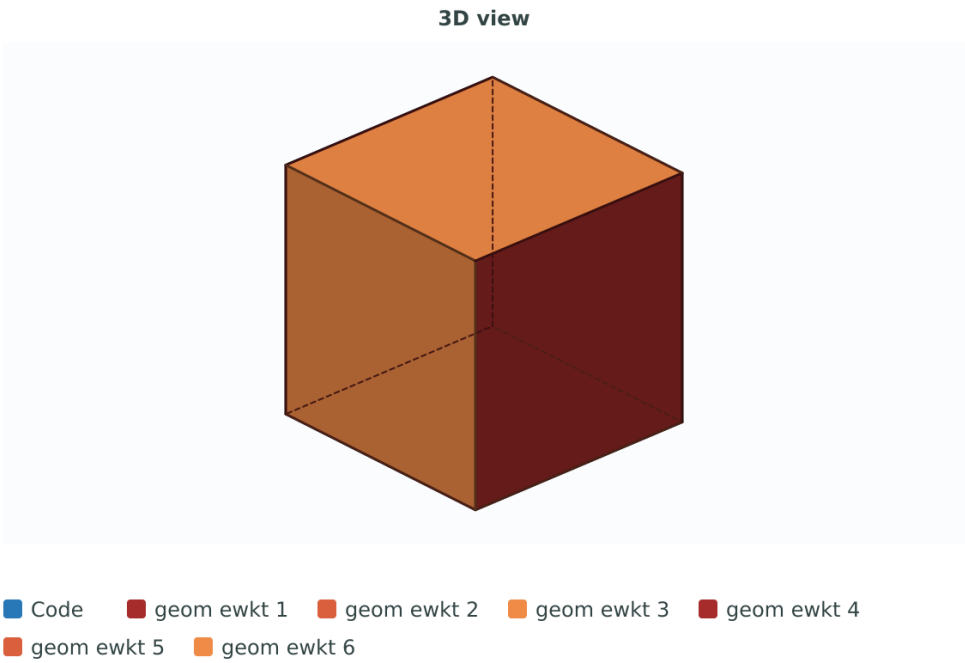


This example breaks a PolyhedralSurface into its faces.

Code

```
SELECT (a.p_geom).path[1] As path,(a.p_geom).geom As geom_ewkt
  FROM (SELECT ST_Dump('POLYHEDRALSURFACE(
((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),((1 1 0,1 1 1,1 0 1,1 0 ↩
    0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
)' ) AS p_geom )  AS a;
```

Figure

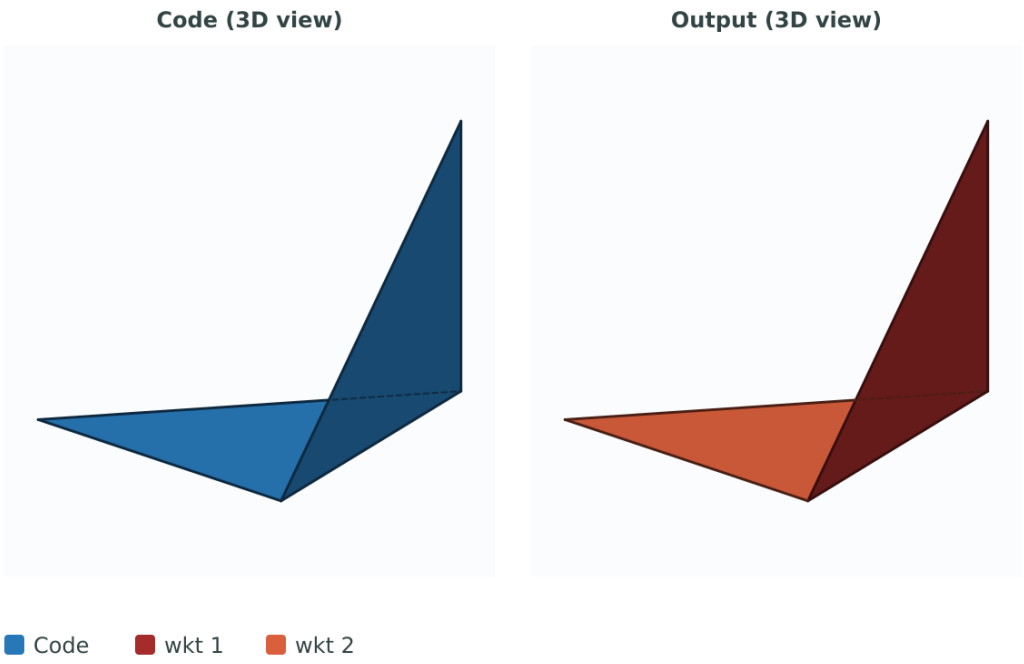


This example uses a TIN geometry.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_Dump('TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
            )),((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
            ))
            )' ) AS gdump
    ) AS g;
```

Figure



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。

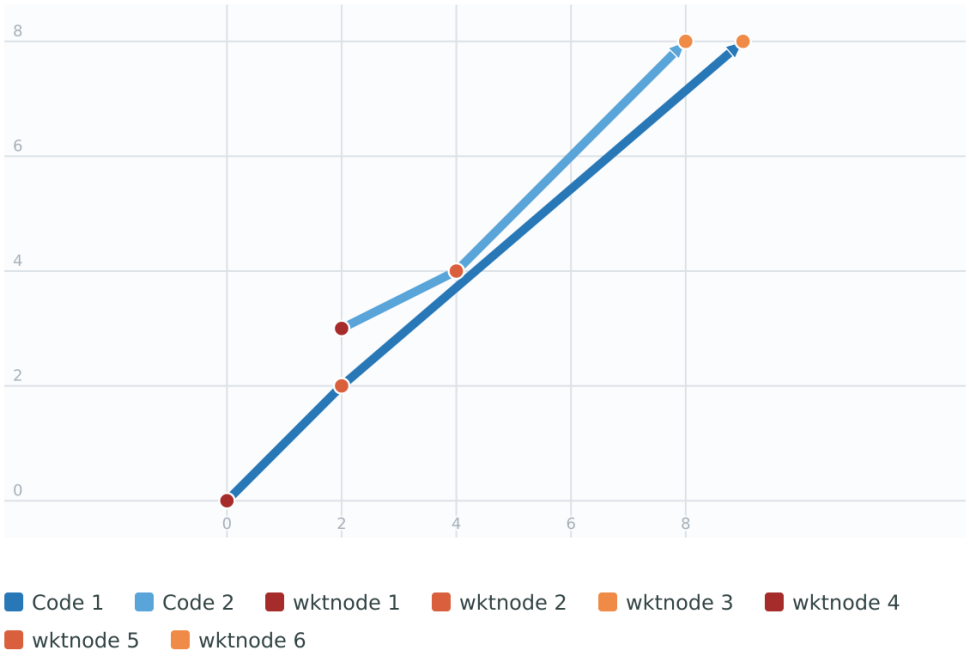
示例

Explode a table of LineStrings into their vertices.

Code

```
SELECT edge_id, (dp).path[1] As index,(dp).geom As wktnode
FROM (SELECT 1 As edge_id,
    ST_DumpPoints(ST_GeomFromText('LINESTRING(1 2,3 4,10 10)')) AS dp
     UNION ALL
     SELECT 2 As edge_id,
    ST_DumpPoints(ST_GeomFromText('LINESTRING(3 5,5 6,9 10)')) AS dp
   ) As foo;
```

Figure



Dump every vertex from a mixed GeometryCollection.

Code

```
SELECT path,geom
FROM (
  SELECT (ST_DumpPoints(g.geom)).*
  FROM
    (SELECT
       'GEOMETRYCOLLECTION(
          POINT ( 0 1 ),
          LINESTRING ( 0 3,3 4 ),
```

```
          POLYGON (( 2 0,2 3,0 2,2 0 )),
          POLYGON (( 3 0,3 3,6 3,6 0,3 0 ),
                   ( 5 1,4 2,5 2,5 1 )),
          MULTIPOLYGON (
                  (( 0 5,0 8,4 8,4 5,0 5 ),
                   ( 1 6,3 6,2 7,1 6 )),
                  (( 5 4,5 8,6 7,5 4 ))
          )
        )'::geometry AS geom
    ) AS g
  ) j;
```

Output

```
path    | st_astext
--------+-----------
 {1,1}     | POINT(0 1)
 {2,1}     | POINT(0 3)
 {2,2}     | POINT(3 4)
 {3,1,1}   | POINT(2 0)
 {3,1,2}   | POINT(2 3)
 {3,1,3}   | POINT(0 2)
 {3,1,4}   | POINT(2 0)
 {4,1,1}   | POINT(3 0)
 {4,1,2}   | POINT(3 3)
 {4,1,3}   | POINT(6 3)
 {4,1,4}   | POINT(6 0)
 {4,1,5}   | POINT(3 0)
 {4,2,1}   | POINT(5 1)
 {4,2,2}   | POINT(4 2)
 {4,2,3}   | POINT(5 2)
 {4,2,4}   | POINT(5 1)
 {5,1,1,1} | POINT(0 5)
 {5,1,1,2} | POINT(0 8)
 {5,1,1,3} | POINT(4 8)
 {5,1,1,4} | POINT(4 5)
 {5,1,1,5} | POINT(0 5)
 {5,1,2,1} | POINT(1 6)
 {5,1,2,2} | POINT(3 6)
 {5,1,2,3} | POINT(2 7)
 {5,1,2,4} | POINT(1 6)
 {5,2,1,1} | POINT(5 4)
 {5,2,1,2} | POINT(5 8)
 {5,2,1,3} | POINT(6 7)
 {5,2,1,4} | POINT(5 4)
(29 rows)
```

Collect the dumped vertices to visualize them over the source geometry.

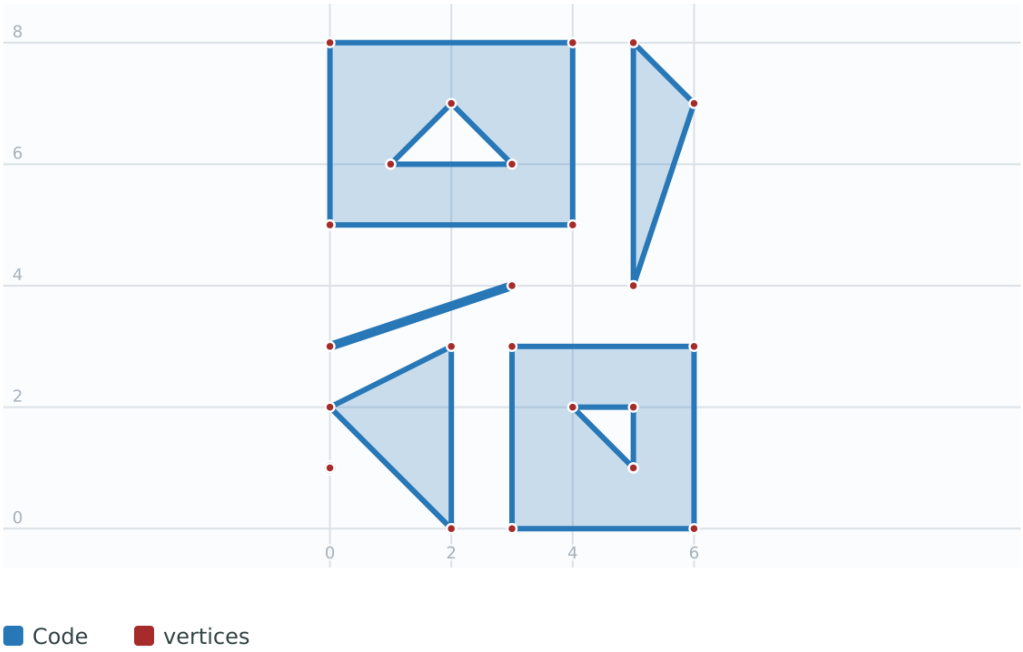
Code

```
WITH source(geom) AS (VALUES (
  'GEOMETRYCOLLECTION(
     POINT (0 1),
     LINESTRING (0 3,3 4),
     POLYGON ((2 0,2 3,0 2,2 0)),
     POLYGON ((3 0,3 3,6 3,6 0,3 0),(5 1,4 2,5 2,5 1)),
     MULTIPOLYGON (((0 5,0 8,4 8,4 5,0 5),(1 6,3 6,2 7,1 6)),
                   ((5 4,5 8,6 7,5 4))))'::geometry
))
SELECT ST_Collect((dp).geom ORDER BY (dp).path) AS vertices
FROM source
CROSS JOIN LATERAL ST_DumpPoints(source.geom) AS dp;
```

Output

```
MULTIPOINT((0 1),(0 3),(3 4),(2 0),(2 3),(0 2),(2 0),(3 0),(3 3),(6 3),(6 0),(3 0),(5 1),(4 2),(5 2),(5 1),(0 5),(0 8) ↩
    ,(4 8),(4 5),(0 5),(1 6),(3 6),(2 7),(1 6),(5 4),(5 8),(6 7),(5 4))
```

Figure



This example uses a cube represented as a PolyhedralSurface.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_DumpPoints('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )' ) AS gdump
    ) AS g;
```

Output

```
path   |     wkt
--------+-------------
 {1,1,1} | POINT(0 0 0)
 {1,1,2} | POINT(0 0 1)
 {1,1,3} | POINT(0 1 1)
 {1,1,4} | POINT(0 1 0)
 {1,1,5} | POINT(0 0 0)
 {2,1,1} | POINT(0 0 0)
 {2,1,2} | POINT(0 1 0)
 {2,1,3} | POINT(1 1 0)
 {2,1,4} | POINT(1 0 0)
 {2,1,5} | POINT(0 0 0)
 {3,1,1} | POINT(0 0 0)
 {3,1,2} | POINT(1 0 0)
 {3,1,3} | POINT(1 0 1)
 {3,1,4} | POINT(0 0 1)
 {3,1,5} | POINT(0 0 0)
 {4,1,1} | POINT(1 1 0)
 {4,1,2} | POINT(1 1 1)
 {4,1,3} | POINT(1 0 1)
 {4,1,4} | POINT(1 0 0)
 {4,1,5} | POINT(1 1 0)
 {5,1,1} | POINT(0 1 0)
 {5,1,2} | POINT(0 1 1)
 {5,1,3} | POINT(1 1 1)
 {5,1,4} | POINT(1 1 0)
 {5,1,5} | POINT(0 1 0)
 {6,1,1} | POINT(0 0 1)
 {6,1,2} | POINT(1 0 1)
```

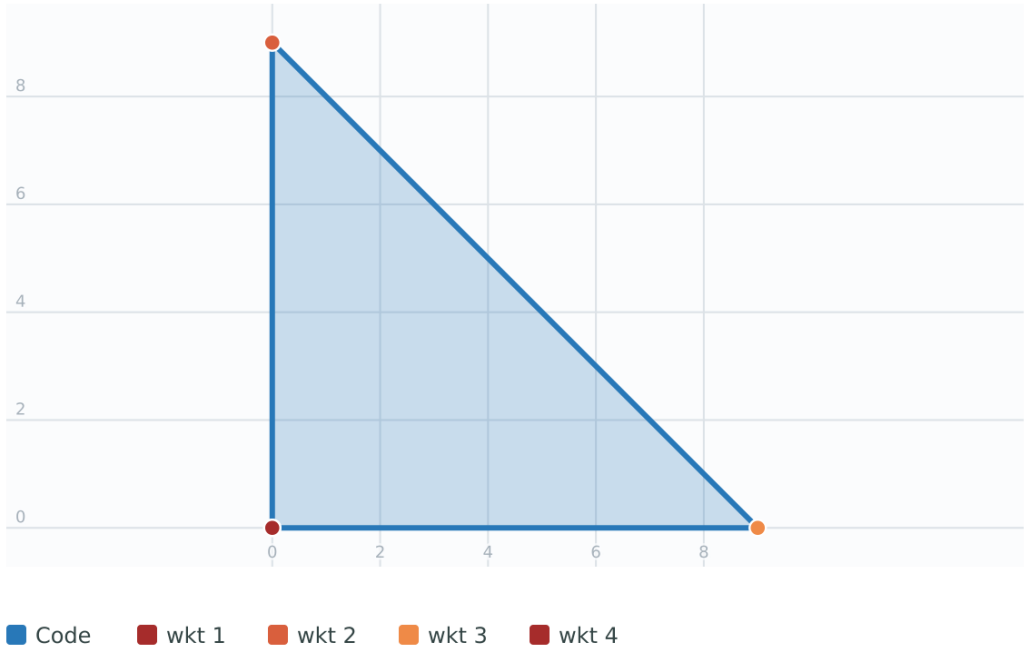
```
 {6,1,3} | POINT(1 1 1)
 {6,1,4} | POINT(0 1 1)
 {6,1,5} | POINT(0 0 1)
(30 rows)
```

This example uses a Triangle geometry.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_DumpPoints('TRIANGLE ((
                0 0,
                0 9,
                9 0,
                0 0
            ))' ) AS gdump
    ) AS g;
```

Figure



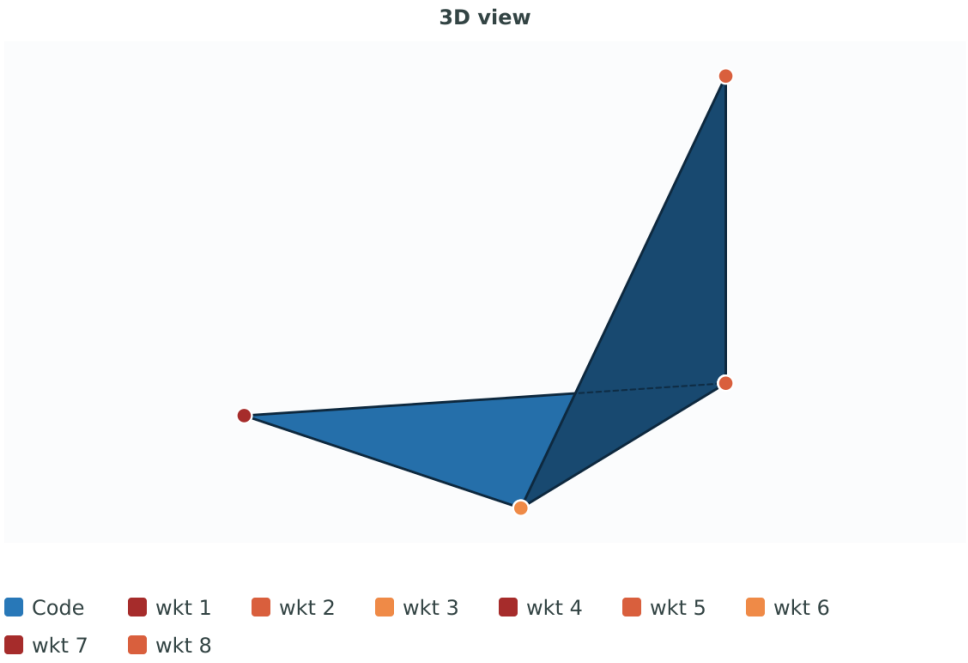
This example uses a TIN geometry.

Code

```
SELECT (g.gdump).path,(g.gdump).geom as wkt
  FROM
    (SELECT
       ST_DumpPoints('TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
            )),((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
```

```
            ))
            )' ) AS gdump
    ) AS g;
```

Figure



相关信息

geometry_dump, Section 13.6, ST_Dump, ST_DumpRings, ST_Points

7.4.8 ST_DumpSegments

ST_DumpSegments — 区几何区形中的各个区段返回一区 geometry_dump 行。

Synopsis

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

描述

提取几何区形区的集合返回函数 (SRF)。它返回一区geometry_dump行，每行包含一个几何区形（*geom* 字段）和一个整数数区（*path* 字段）。

- *geom* 字段中的 LINESTRING 表示区定几何体的区段，而 CIRCULARSTRING 表示区弧区段。
- *path* 字段（integer[]）是枚区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，区于 LINESTRING，路径区 {i}，其中 i 是 LINESTRING 中的第 n 个区段起点。区于 POLYGON，路径区 {i,j}，其中 i 是区区号（1 是外区；后面是内区），j 是区中的区段起点位置。

可用性：3.2.0

此函数支持三角形和不⊠⊠三角网面 (TIN)。

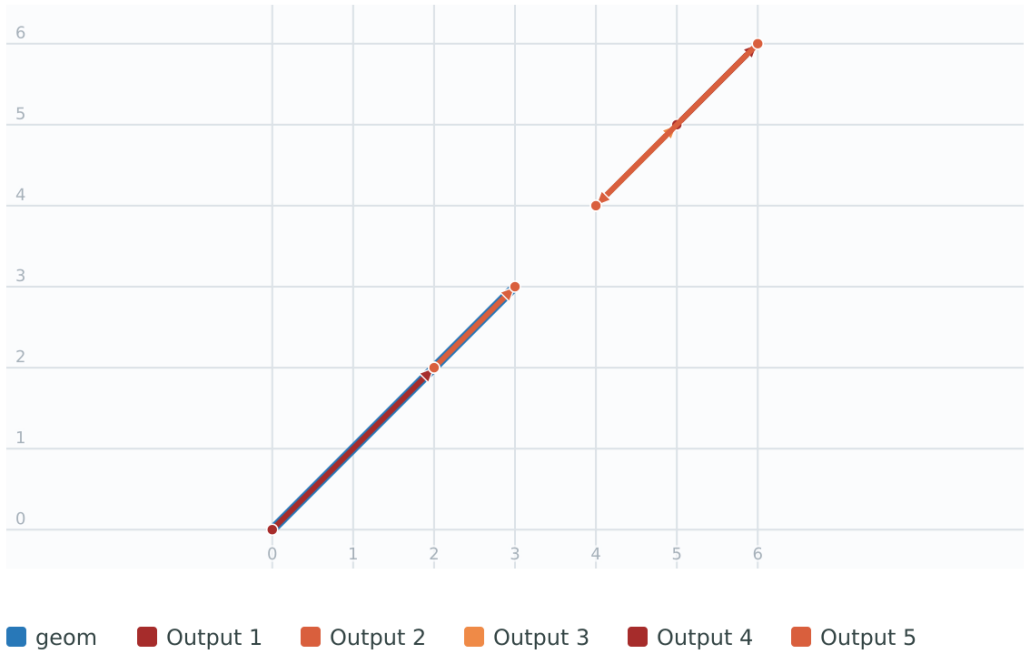
⊠函数支持 3d 并且不会⊠失 z-index。

⊠准几何示例

Code

```
SELECT path,geom
FROM (
    SELECT (ST_DumpSegments(g.geom)).*
    FROM (SELECT 'GEOMETRYCOLLECTION(
    LINESTRING(1 1,3 3,4 4),
    POLYGON((5 5,6 6,7 7,5 5))
)'::geometry AS geom
        ) AS g
) j;
```

Figure



TIN 和三角形示例

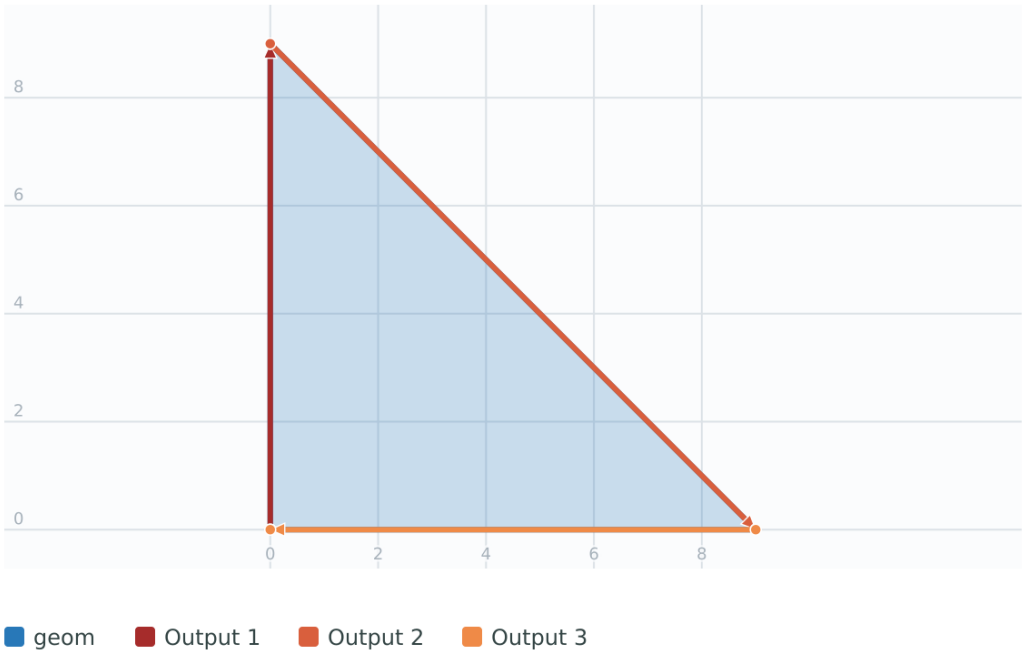
This example uses a Triangle geometry.

Code

```
SELECT path,geom
FROM (
    SELECT (ST_DumpSegments(g.geom)).*
    FROM (SELECT 'TRIANGLE((
        0 0,
        0 9,
```

```
        9 0,
        0 0
    ))'::geometry AS geom
        ) AS g
) j;
```

Figure

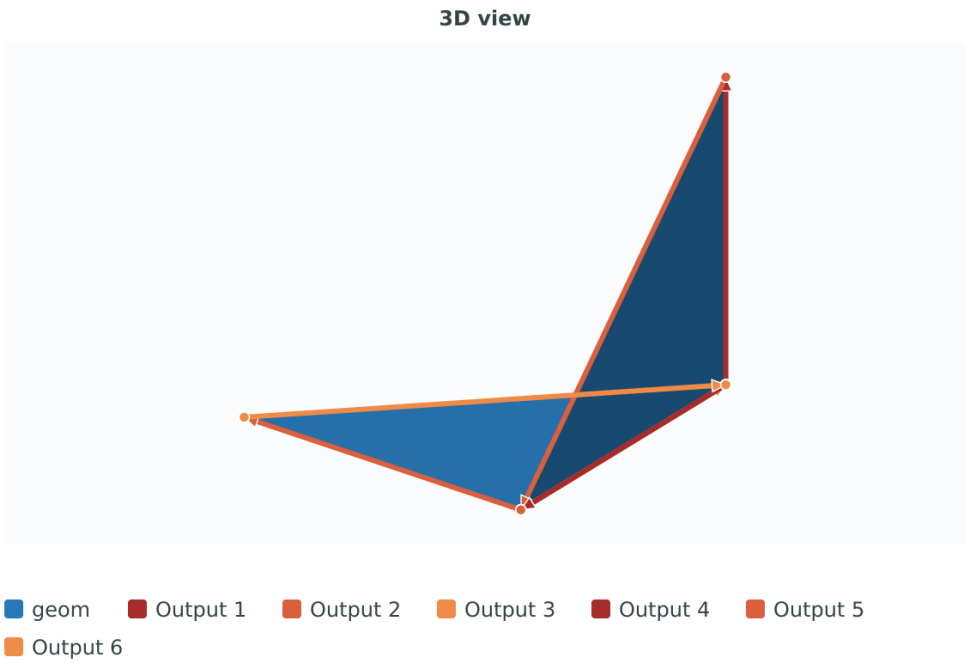


This example uses a TIN geometry.

Code

```
SELECT path,geom
FROM (
    SELECT (ST_DumpSegments(g.geom)).*
    FROM (SELECT 'TIN(((
        0 0 0,
        0 0 1,
        0 1 0,
        0 0 0
    )),((
        0 0 0,
        0 1 0,
        1 1 0,
        0 0 0
    ))
    )'::geometry AS geom
        ) AS g
) j;
```

Figure



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
This only works for POLYGON geometries, not MULTIPOLYGONs. Use ST_Dump to extract polygon elements from polygonal geometries: `ST_DumpRings((ST_Dump(geom)).geom )`

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

将所有⊠提取⊠ polygon。

Code

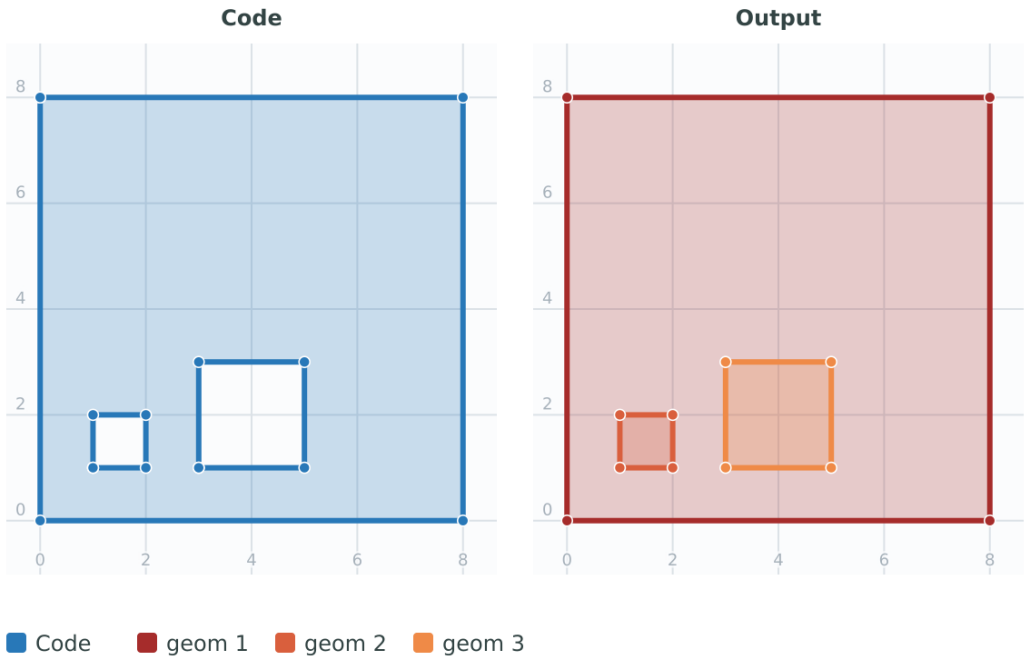
```
SELECT polyTable.field1, polyTable.field1,
      (ST_DumpRings(polyTable.geom)).geom As geom
FROM polyTable
```

从 polygon 中提取外壳（shell）和洞（holes）。

Code

```
SELECT path,geom As geom
FROM ST_DumpRings('POLYGON ((1 9,9 9,9 1,1 1,1 9),(2 2,2 3,3 3,3 2,2 2),(4 2,4 4,6 4,6 2,4
    2))');
```

Figure



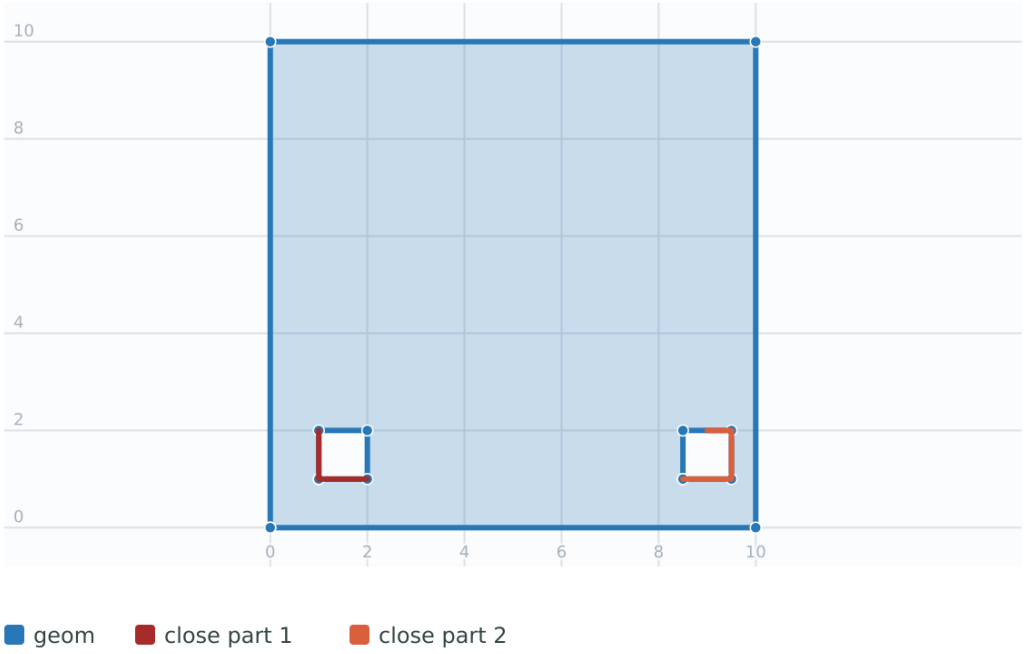
Finding interior rings which lie close to the exterior ring.

Code

```
WITH poly AS (
  SELECT 'POLYGON ((
      0 0, 10 0, 10 10, 0 10, 0 0
    ), (
      1 1, 1 2, 2 2, 2 1, 1 1
    ), (
      8.5 1, 8.5 2, 9.5 2, 9.5 1, 8.5 1
    ))'::geometry AS geom
),
rings AS (
  SELECT d.path[1] AS ring_no,
         ST_Boundary(d.geom) AS ring_geom
  FROM poly
  CROSS JOIN LATERAL ST_DumpRings(poly.geom) AS d
```

```
),
shell AS (
  SELECT ring_geom
  FROM rings
  WHERE ring_no = 0
),
holes AS (
  SELECT ring_no, ring_geom
  FROM rings
  WHERE ring_no > 0
)
SELECT h.ring_no,
       ST_Normalize(ST_Multi(ST_LineMerge(
         ST_Intersection(ST_Buffer(s.ring_geom, 1.0), h.ring_geom)
       ))) AS close_part
FROM shell AS s
JOIN holes AS h
  ON ST_DWithin(s.ring_geom, h.ring_geom, 1.0);
```

Figure



相关信息

geometry_dump, Section 13.6, ST_Boundary, ST_Buffer, ST_Dump, ST_DWithin, ST_ExteriorRing, ST_Interi
ST_Intersection

7.4.10 ST_EndPoint

ST_EndPoint — Returns the last point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_EndPoint**(geometry g);

描述

Returns the last point of a LINESTRING, CIRCULARLINESTRING, or NURBSCURVE geometry as a POINT. Returns NULL if the input is not a LINESTRING, CIRCULARLINESTRING, or NURBSCURVE.

☒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☒。

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

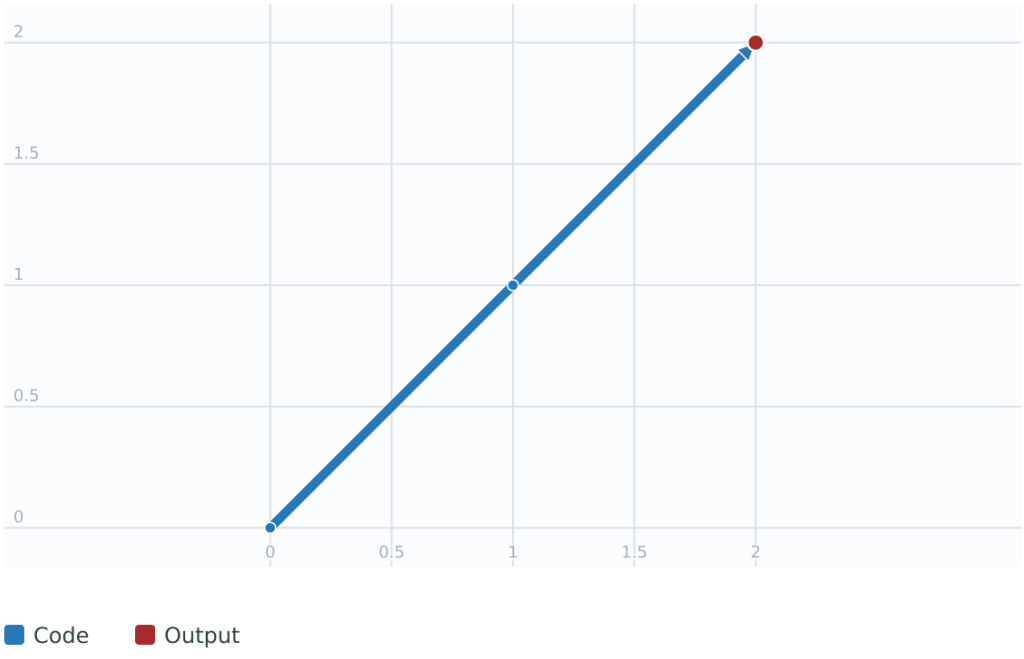
Code

```
SELECT ST_EndPoint('LINESTRING(1 1,2 2,3 3)'::geometry);
```

Output

```
POINT(3 3)
```

Figure



非☒串☒止点☒ NULL

Code

```
SELECT ST_EndPoint('POINT(1 1)'::geometry) IS NULL AS is_null;
```

Output

```
t
```

3D ☒串 (LineString) 端点

3D endpoint.

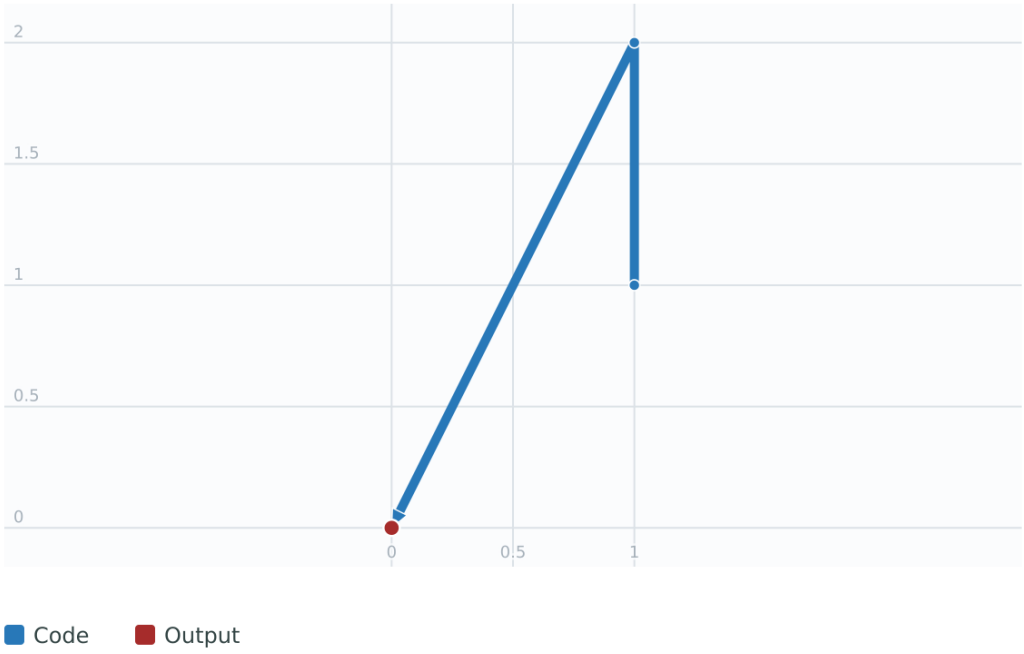
Code

```
SELECT ST_EndPoint('LINESTRING(1 1 2,1 2 3,0 0 5)');
```

Output

```
POINT(0 0 5)
```

Figure



☒弧的端点

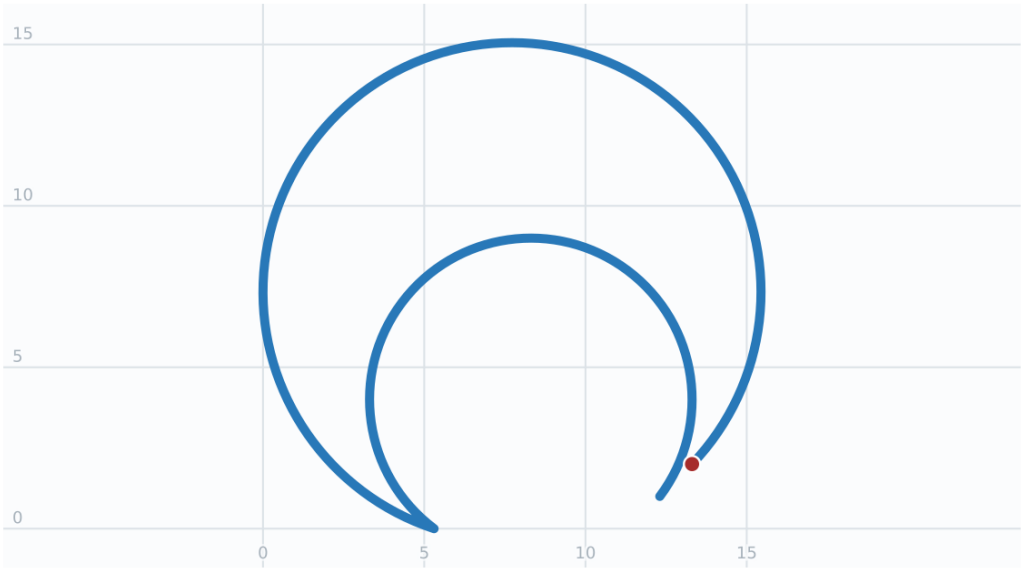
Code

```
SELECT ST_EndPoint('CIRCULARSTRING(5 2,-3 2,-2 1,-4 2,6 3)'::geometry);
```

Output

```
POINT(6 3)
```

Figure



End point of a NURBSCurve

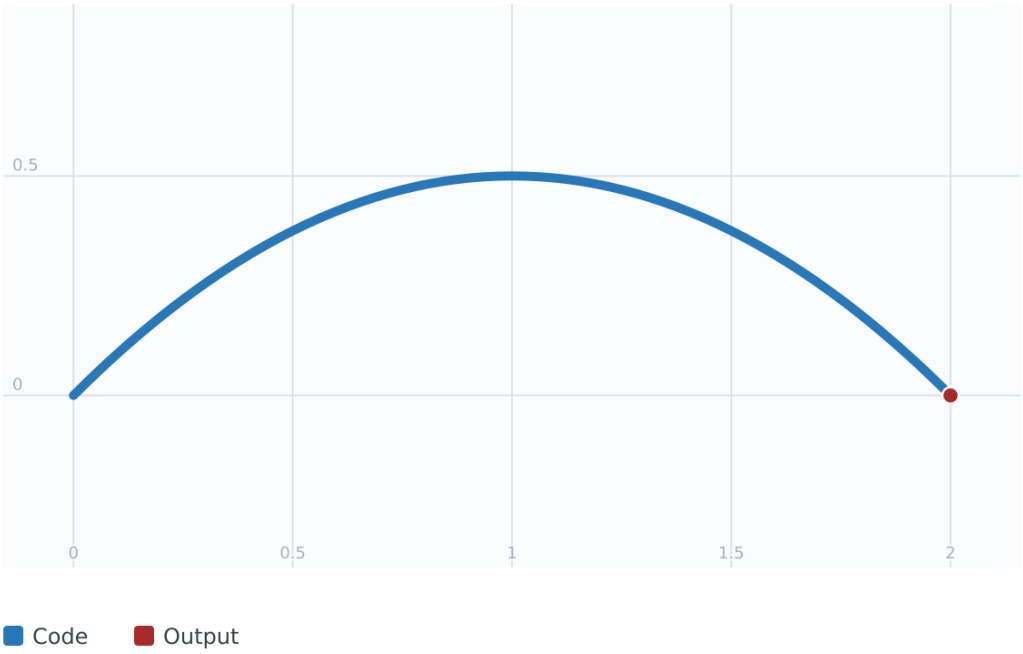
Code

```
SELECT ST_EndPoint('NURBSCURVE(2, (0 0, 1 1, 2 0))'::geometry);
```

Output

```
POINT(2 0)
```

Figure



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

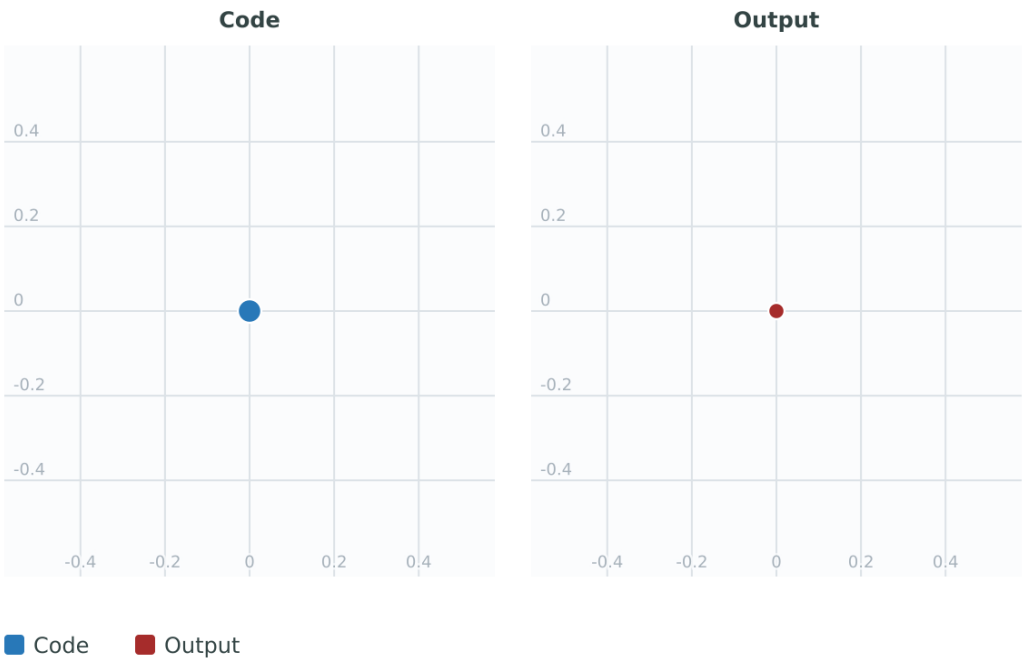
Code

```
SELECT ST_Envelope('POINT(1 3)'::geometry);
```

Output

```
POINT(1 3)
```

Figure



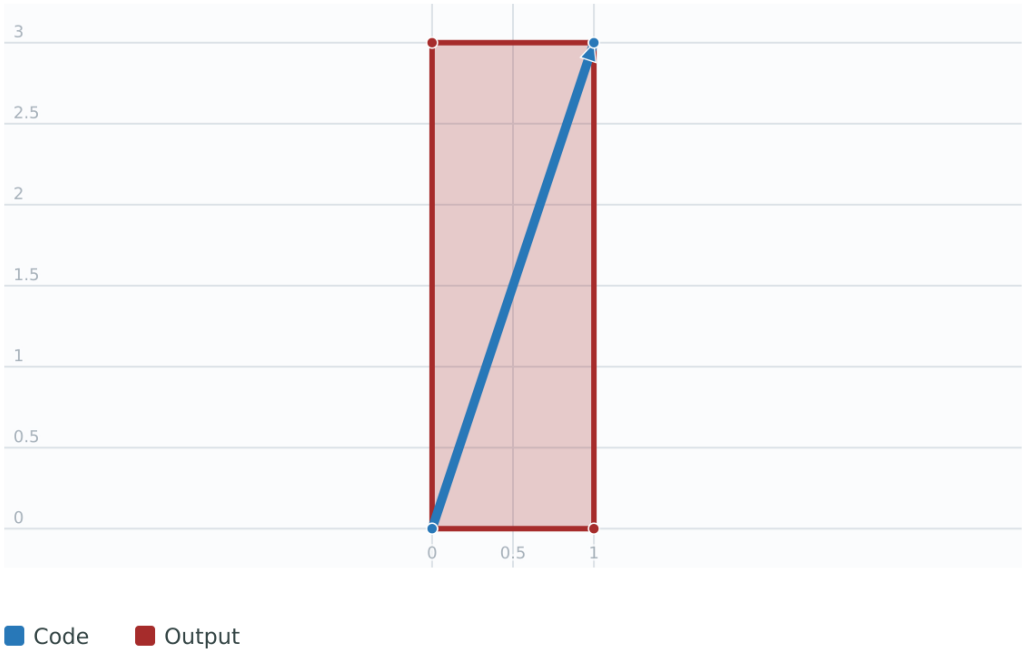
Code

```
SELECT ST_Envelope('LINESTRING(0 0,1 3)'::geometry);
```

Output

```
POLYGON((0 0,0 3,1 3,1 0,0 0))
```

Figure



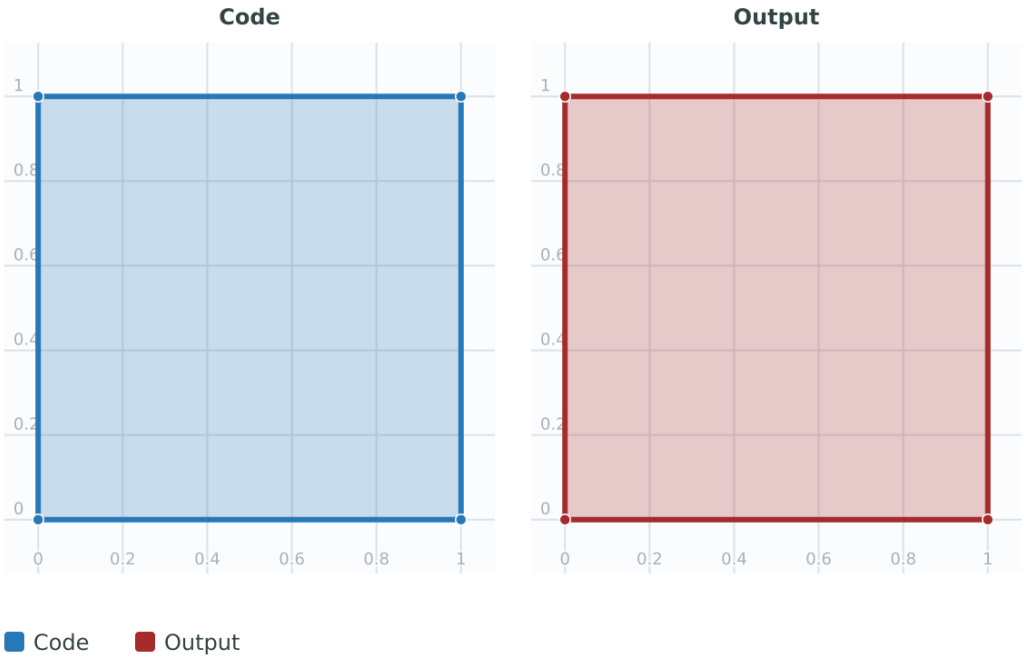
Code

```
SELECT ST_Envelope('POLYGON((0 0,0 1,1.0000001 1,1.0000001 0,0 0))'::geometry);
```

Output

```
POLYGON((0 0,0 1,1.0000001 1,1.0000001 0,0 0))
```

Figure



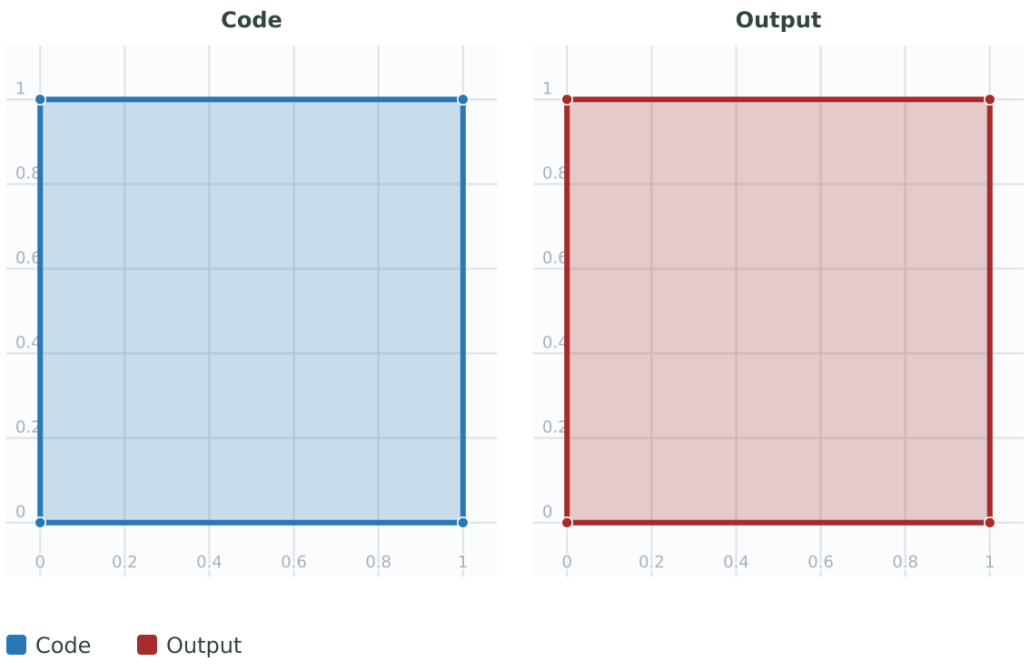
Code

```
SELECT ST_Envelope('POLYGON((0 0,0 1,1.0000000001 1,1.0000000001 0,0 0))'::geometry);
```

Output

```
POLYGON((0 0,0 1,1.0000000001 1,1.0000000001 0,0 0))
```

Figure



Code

```
SELECT Box3D(geom), Box2D(geom), ST_Envelope(geom) As envelopewkt
    FROM (SELECT 'POLYGON((0 0,0 1000012333334.34545678,1.0000001 1,1.0000001 0,0 0))':: ↩
    geometry As geom) As foo;
```

Envelope of a point and LineString.

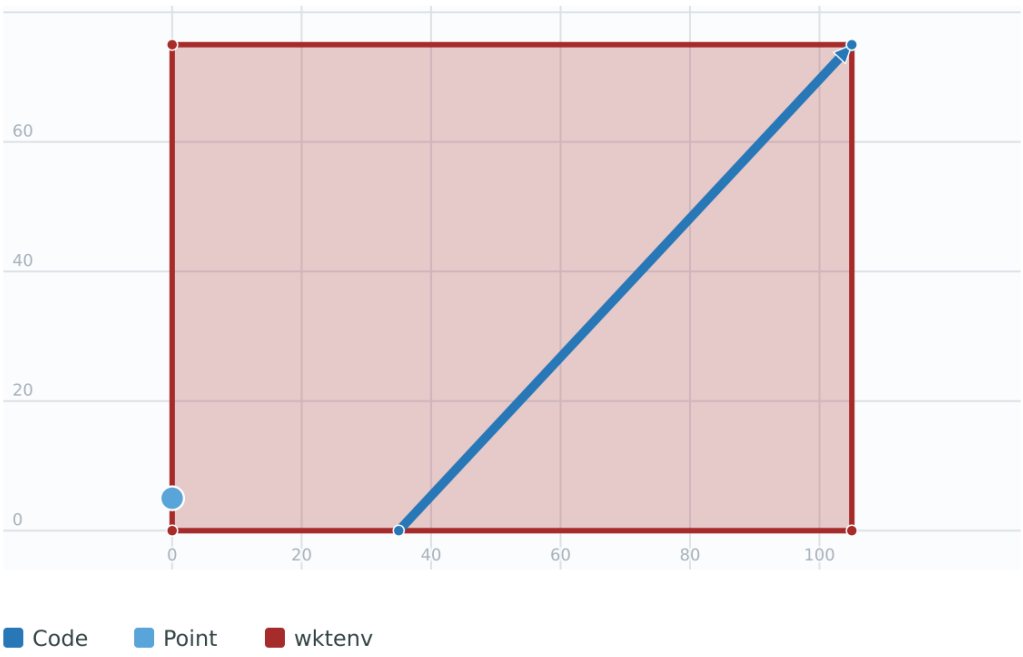
Code

```
SELECT ST_Envelope(ST_Collect(
            'LINESTRING(55 75,125 150)',
            ST_Point(20, 80)
            )) As wktenv;
```

Output

```
POLYGON((20 75,20 150,125 150,125 75,20 75))
```

Figure



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

Extract the exterior ring from a Polygon with a hole.

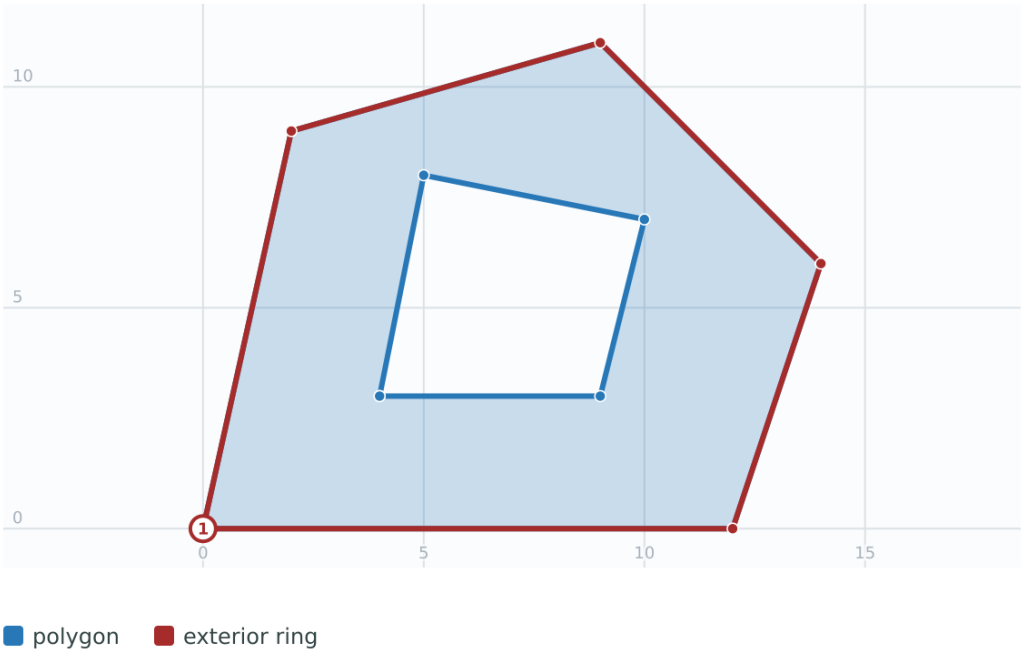
Code

```
WITH input(geom) AS (VALUES (
  'POLYGON((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3))'::geometry
))
SELECT geom AS input_polygon,
       ST_ExteriorRing(geom) AS exterior_ring
FROM input;
```

Output

```
POLYGON((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3)) | LINESTRING(0 0,2 9,9 11,14 6,12 0,0 0)
```

Figure



For a MultiPolygon, dump the component Polygons before extracting their exterior rings. Both component Polygons in this example have holes.

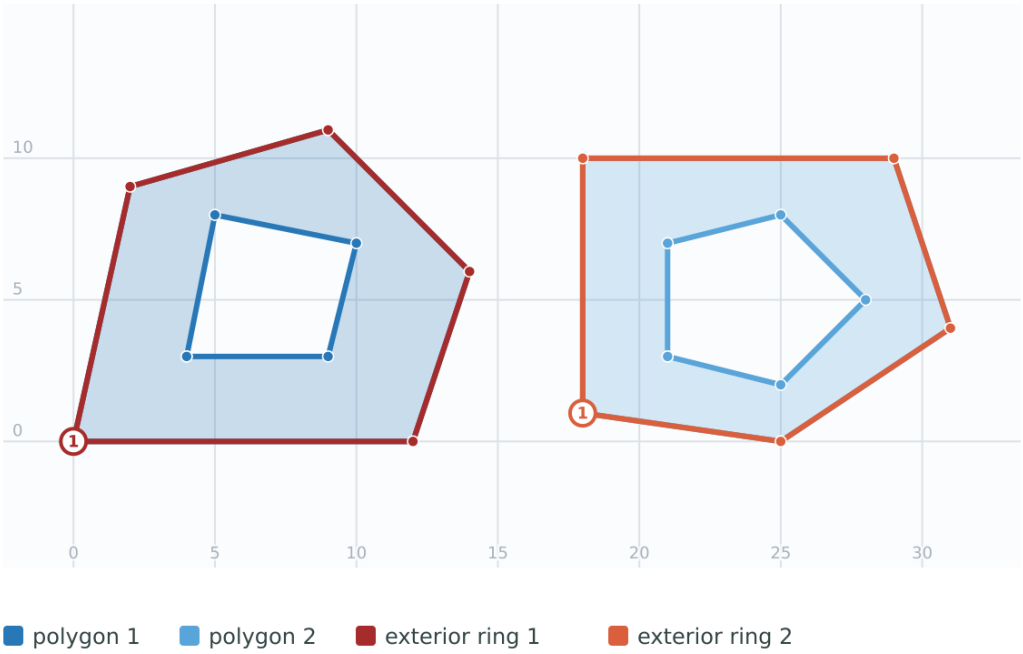
Code

```
WITH input(geom) AS (VALUES (
  'MULTIPOLYGON(
    ((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3)),
    ((18 1,18 10,29 10,31 4,25 0,18 1),(21 3,25 2,28 5,25 8,21 7,21 3))
  )'::geometry
))
SELECT part.geom AS input_polygon,
       ST_ExteriorRing(part.geom) AS exterior_ring
FROM input
CROSS JOIN LATERAL ST_Dump(geom) AS part
ORDER BY part.path;
```

Output

```
POLYGON((0 0,2 9,9 11,14 6,12 0,0 0),(4 3,9 3,10 7,5 8,4 3)) | LINESTRING(0 0,2 9,9 11,14 6,12 0,0 0)
POLYGON((18 1,18 10,29 10,31 4,25 0,18 1),(21 3,25 2,28 5,25 8,21 7,21 3)) | LINESTRING(18 1,18 10,29 10,31 4,25 0,18 ↩
    1)
```

Figure



Extract exterior rings from a table of Polygons.

Code

```
SELECT gid, ST_ExteriorRing(geom) AS ering
FROM sometable;
```

For a table of MultiPolygons, dump the component Polygons and collect their exterior rings into a MultiLineString.

Code

```
SELECT gid, ST_Collect(ST_ExteriorRing(geom)) AS erings
FROM (
  SELECT gid, (ST_Dump(geom)).geom AS geom
  FROM sometable
) AS foo
GROUP BY gid;
```

This example returns the 3D exterior ring of a Polygon.

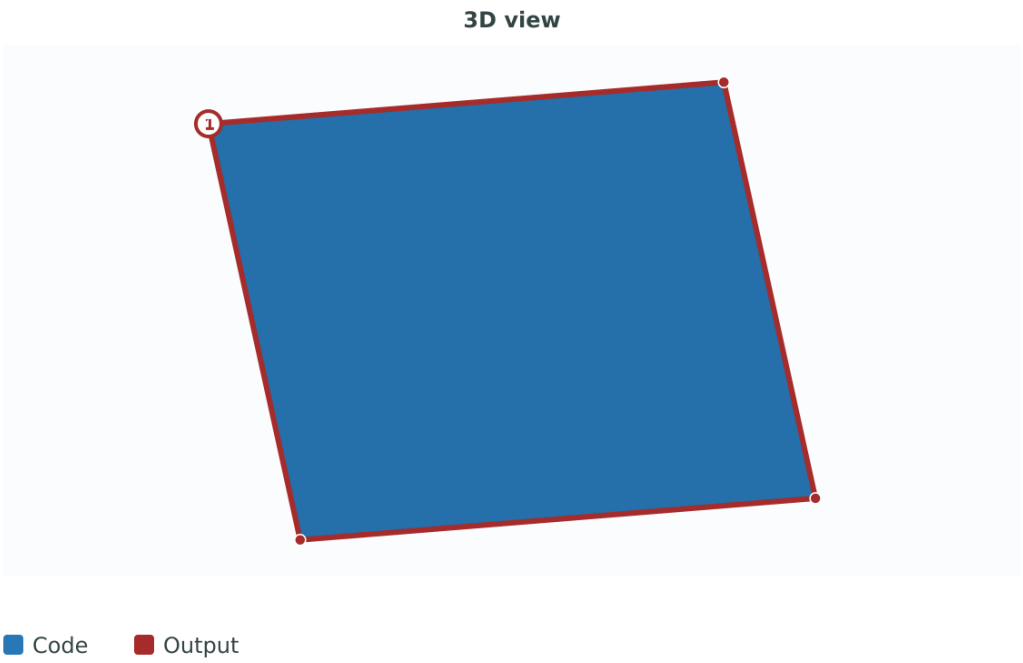
Code

```
SELECT ST_ExteriorRing(
    'POLYGON Z ((0 0 5,4 0 5,4 3 5,0 3 5,0 0 5))'
    );
```

Output

```
LINESTRING(0 0 5,4 0 5,4 3 5,0 3 5,0 0 5)
```

Figure



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

Returns the 1-based Nth element of a GEOMETRYCOLLECTION, MULTIPOINT, MULTILINESTRING, MULTICURVE, or MULTIPOLYGON. For a unitary geometry, including a TIN or PolyhedralSurface, an index of 1 returns the input geometry and any other index returns NULL. Use ST_PatchN and ST_NumPatches to access and count the faces of a TIN or PolyhedralSurface.



Note
自版本 0.8.0 以来，OGC 规范的索引从 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。

Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and are not decomposed into patches; use ST_PatchN to access faces.

 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)。

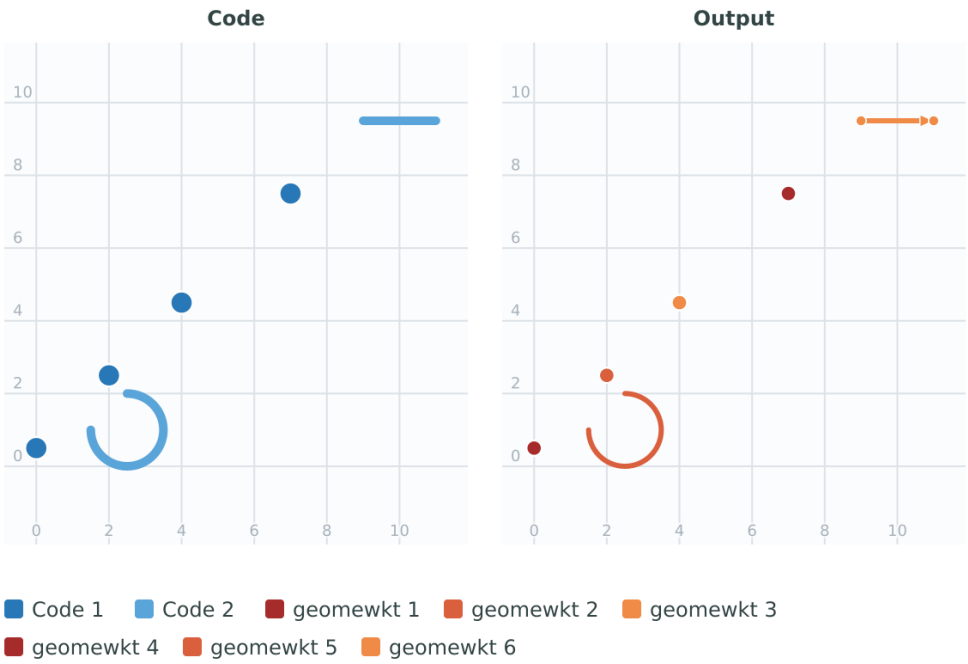
⍰准示例

Extracting a subset of points from a 3D multipoint.

Code

```
SELECT n, ST_GeometryN(geom, n) As geomewkt
FROM (
VALUES
    (1, 'MULTIPOINT((1 2 7), (3 4 7), (5 6 7), (8 9 10))'::geometry ),
    (2, 'MULTICURVE(CIRCULARSTRING(2.5 2.5,4.5 2.5, 3.5 3.5), (10 11, 12 11))'::geometry )
    ) As foo(id, geom)
    CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(geom)
ORDER BY id, n;
```

Figure



Extracting all geometries (useful when you want to assign an id).

Code

```
SELECT gid, n, ST_GeometryN(geom, n)
FROM sometable CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(geom);
```

多面体曲面、**TIN** 和三角形的示例

A PolyhedralSurface or TIN is treated as a single geometry, not as a collection of faces. `ST_GeometryN` does not decompose the surface into patches, so an index greater than 1 returns `NULL`. Use ST_PatchN and ST_NumPatches to access individual faces.

This example requests the third element of a PolyhedralSurface.

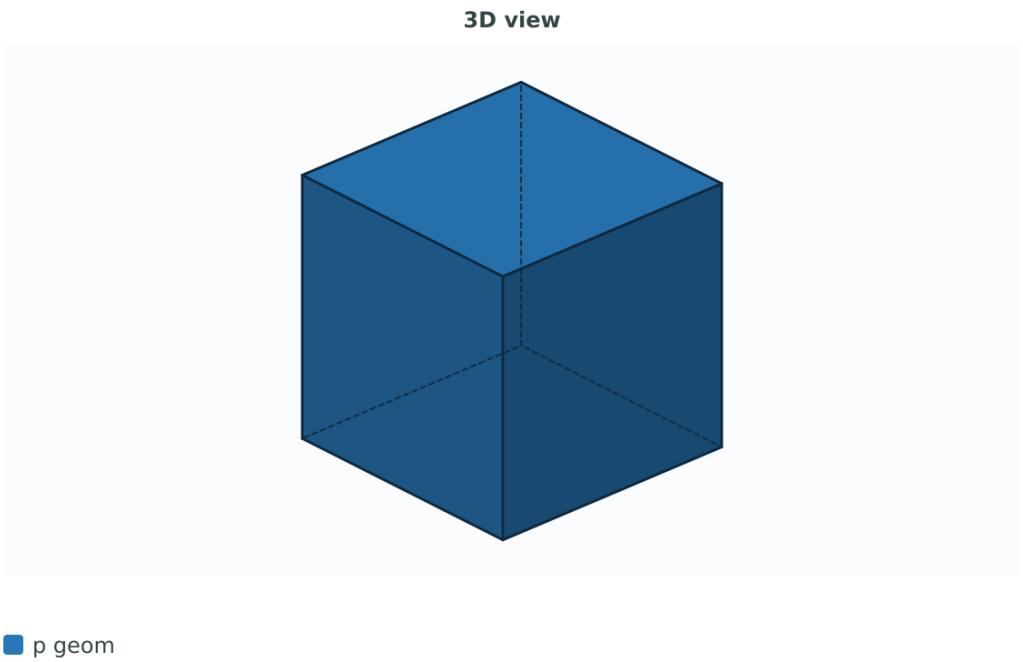
Code

```
SELECT ST_GeometryN(p_geom, 3) As geom_ewkt
  FROM (SELECT 'POLYHEDRALSURFACE(
((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
)'::geometry  AS p_geom )  AS a;
```

Output

```
NULL
```

Figure



This example requests the second element of a TIN.

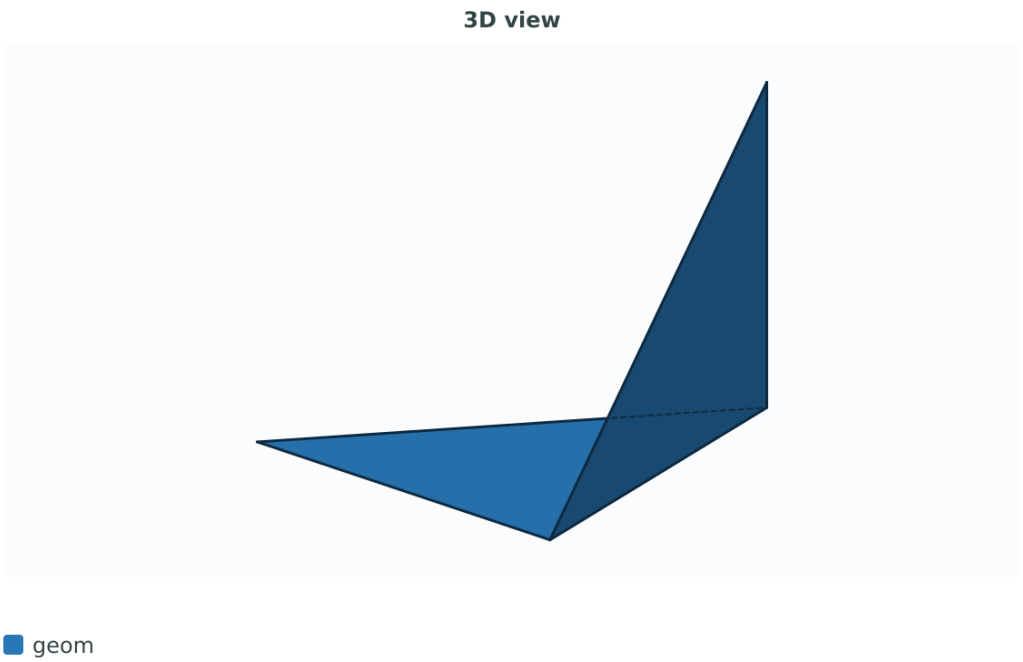
Code

```
SELECT ST_GeometryN(geom, 2) as wkt
  FROM
    (SELECT
       'TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
            )),((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
            ))
            )'::geometry  AS geom
    ) AS g;
```

Output

```
NULL
```

Figure



相关信息

ST_Dump, ST_NumGeometries, ST_PatchN, ST_NumPatches

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

Code

```
SELECT ST_GeometryType('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)':: ↩
    geometry);
```

Output

```
ST_LineString
```

Code

```
SELECT ST_GeometryType('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry);
```

Output

```
ST_PolyhedralSurface
```

Code

```
SELECT ST_GeometryType(geom) AS result
FROM
  (SELECT
     'TIN (((
               0 0 0,
               0 0 1,
               0 1 0,
               0 0 0
          )),((
               0 0 0,
               0 1 0,
               1 1 0,
               0 0 0
          ))
          )'::geometry AS geom
  ) AS g;
```

Output

```
ST_Tin
```

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

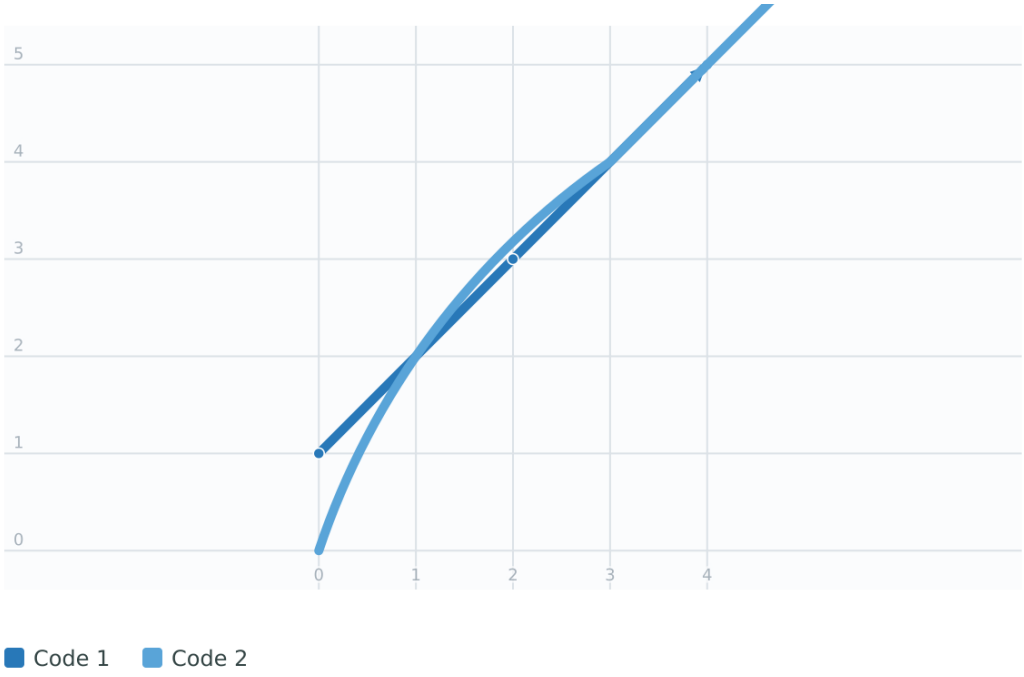
Code

```
SELECT ST_HasArc(ST_Collect('LINESTRING(1 2,3 4,5 6)', 'CIRCULARSTRING(1 1,2 3,4 5,6 7,5 6) ↩
    '));
```

Output

```
t
```

Figure



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

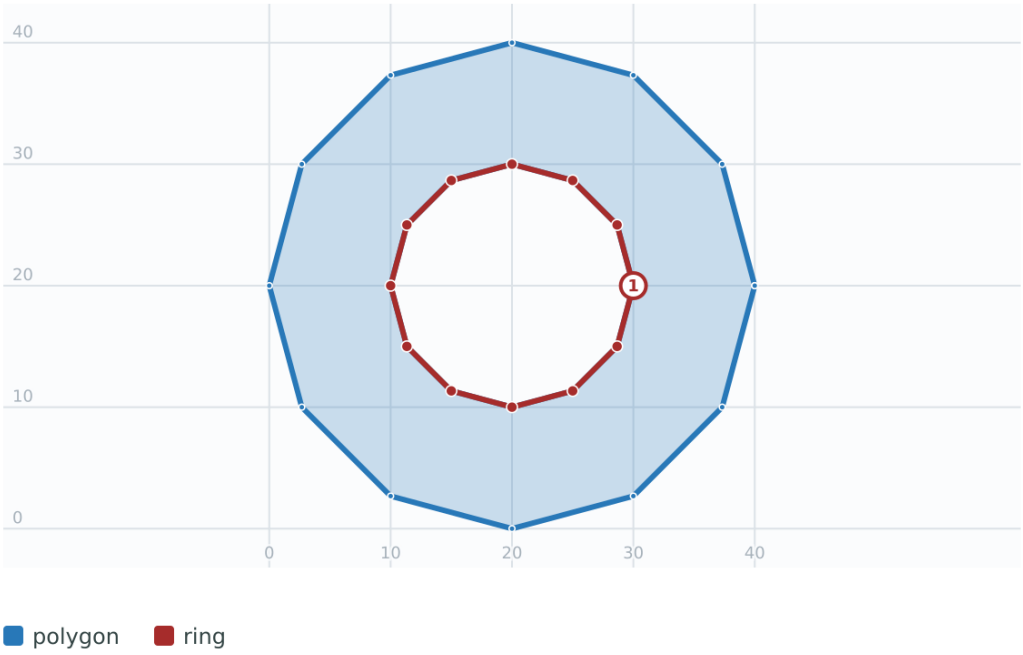

⍰函数支持 3d 并且不会⍰失 z-index。

示例

Code

```
WITH data AS (
  SELECT ST_BuildArea(ST_Collect(
    ST_Buffer(ST_Point(1, 2), 20, 3),
    ST_Buffer(ST_Point(1, 2), 10, 3)
  )) AS geom
)
SELECT geom AS input_polygon,
       ST_InteriorRingN(geom, 1) AS ring
FROM data;
```

Figure



相关信息

ST_ExteriorRing, ST_BuildArea, ST_Collect, ST_Dump, ST_NumInteriorRing, ST_NumInteriorRings

7.4.17 ST_NumCurves

ST_NumCurves — 返回复合曲⊠中的曲⊠数量。

Synopsis

integer **ST_NumCurves**(geometry a_compoundcurve);

描述

返回复合曲⊠中的曲⊠数量。如果⊠入⊠空的复合曲⊠，⊠返回 0；如果不是复合曲⊠，⊠返回 NULL。

⊠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

This CompoundCurve contains three component curves.

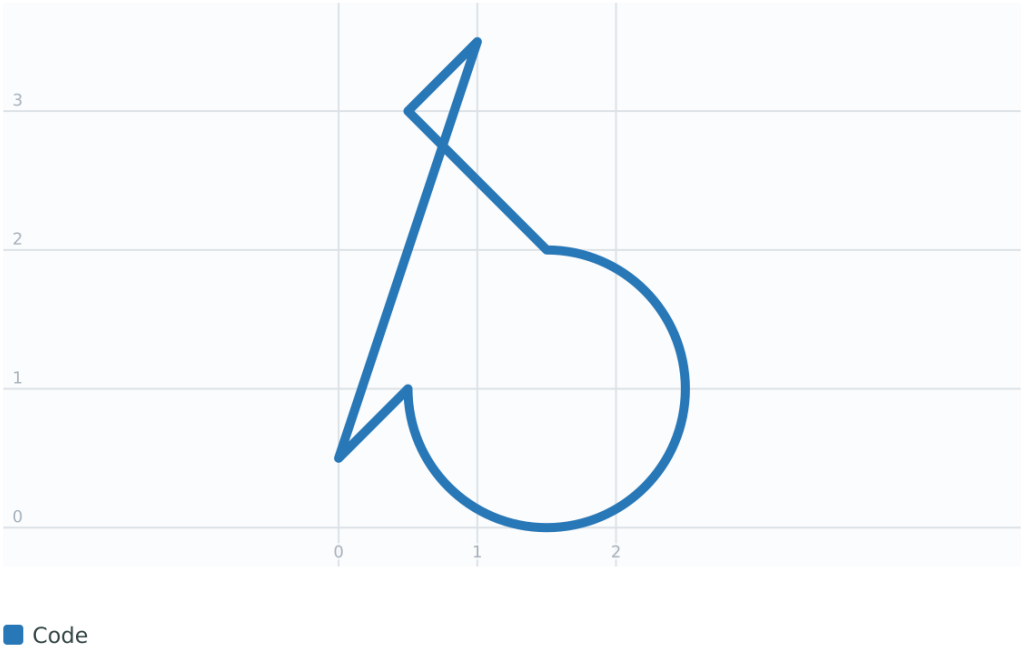
Code

```
SELECT ST_NumCurves('COMPOUNDCURVE(
    (2 2,2.5 2.5),
    CIRCULARSTRING(2.5 2.5,4.5 2.5,3.5 3.5),
    (3.5 3.5,2.5 4.5,3 5,2 2)
  )');
```

Output

```
3
```

Figure



An empty CompoundCurve has no component curves.

Code

```
SELECT ST_NumCurves('COMPOUNDCURVE EMPTY');
```

Output

```
0
```

相关信息

ST_CurveN, ST_Dump, ST_ExteriorRing, ST_NumInteriorRings, ST_NumGeometries

7.4.18 ST_CurveN

ST_CurveN — 返回复合曲⌧（CompoundCurve）中的第 N 个曲⌧几何体。

Synopsis

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

描述

返回复合曲⍰（CompoundCurve）中的第 N 个曲⍰几何体。索引从 1 开始。如果⍰入的几何体不是复合曲⍰，或者索引超出范⍰，⍰返回 NULL。

⍰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

This example extracts the second component curve, a CircularString, from a CompoundCurve.

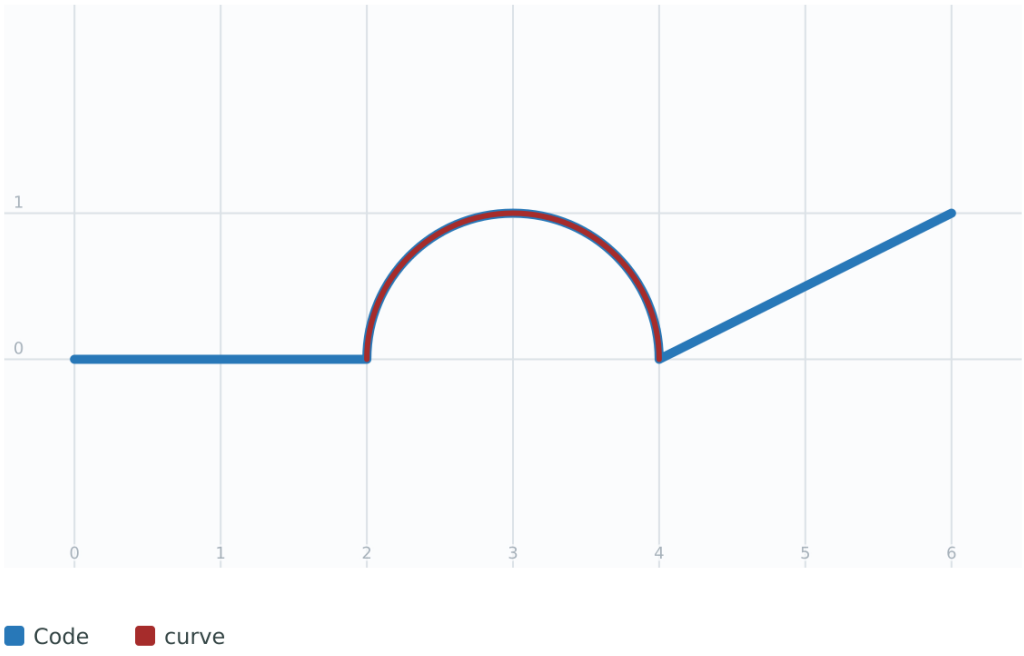
Code

```
SELECT ST_CurveN(
  'COMPOUNDCURVE(
    (0 0,2 0),
    CIRCULARSTRING(2 0,3 1,4 0),
    (4 0,6 1)
  )',
  2) AS curve;
```

Output

```
CIRCULARSTRING(2 0,3 1,4 0)
```

Figure



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

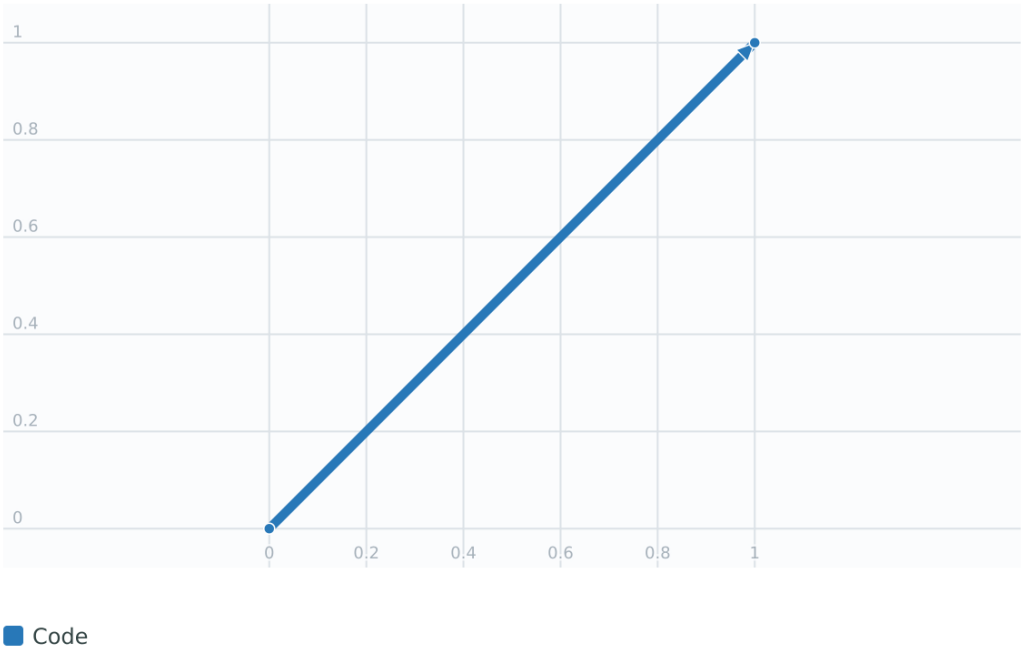
Code

```
SELECT ST_IsClosed('LINESTRING(0 0,1 1)'::geometry);
```

Output

```
f
```

Figure



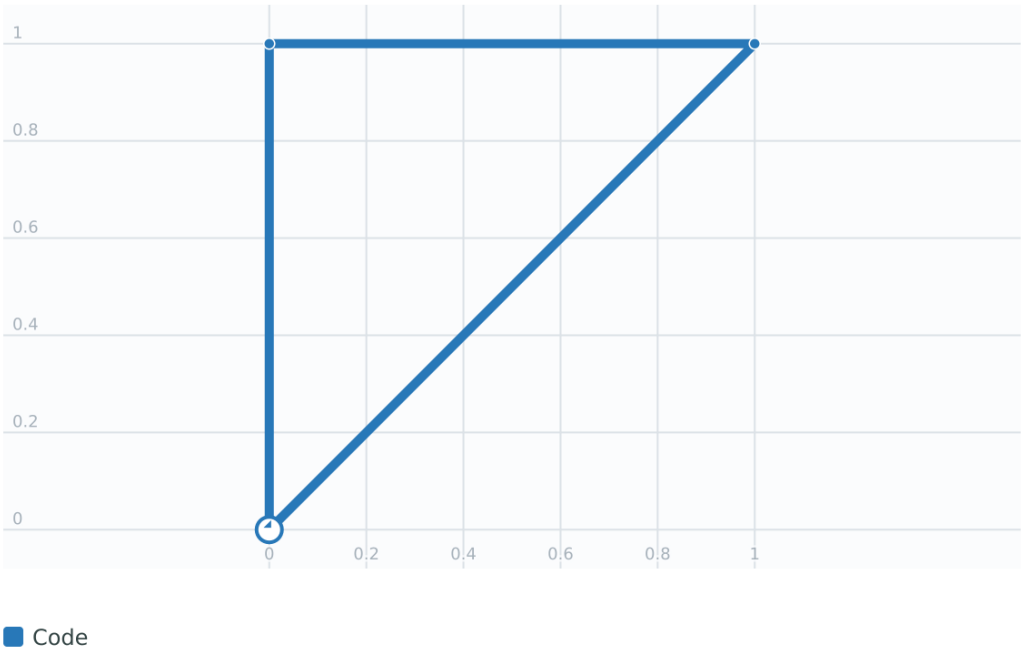
Code

```
SELECT ST_IsClosed('LINESTRING(0 0,0 1,1 1,0 0)'::geometry);
```

Output

```
t
```

Figure



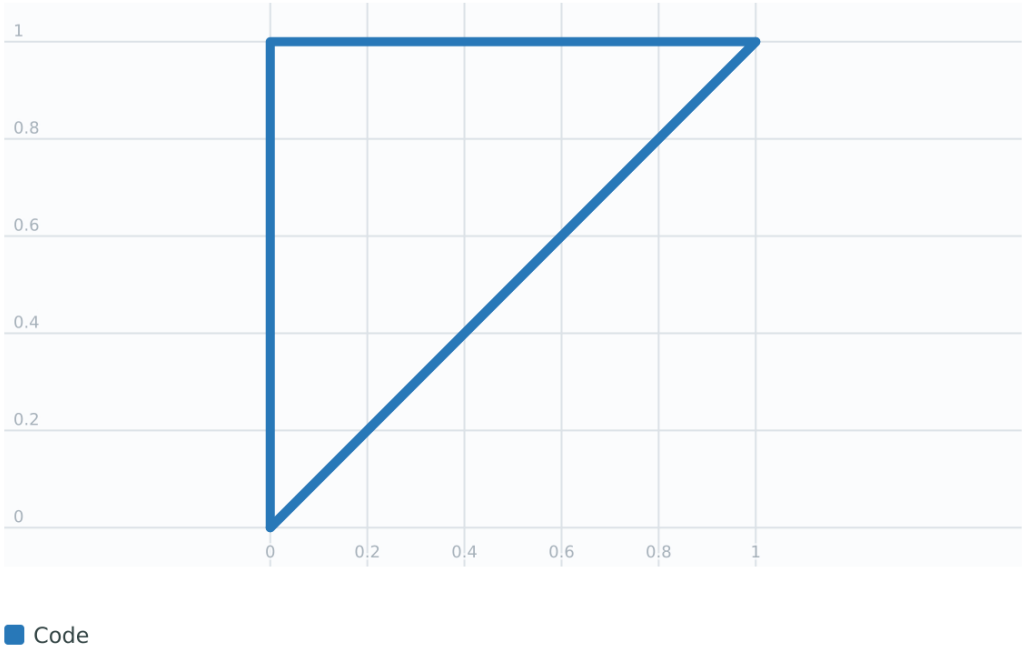
Code

```
SELECT ST_IsClosed('MULTILINESTRING((0 0,0 1,1 1,0 0),(0 0,1 1))'::geometry);
```

Output

```
f
```

Figure



Code

```
SELECT ST_IsClosed('POINT(0 0)'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsClosed('MULTIPOINT((0 0),(1 1))'::geometry);
```

Output

```
t
```

多面体曲面示例

This example uses a cube.

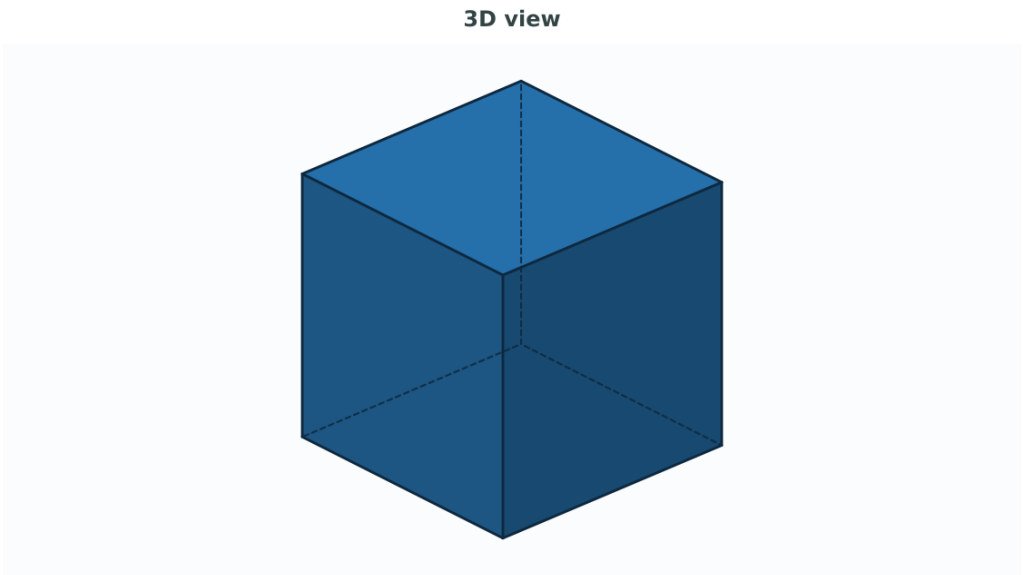
Code

```
SELECT ST_IsClosed('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )');
```

Output

```
t
```

Figure



This example uses the same cube with one side missing.

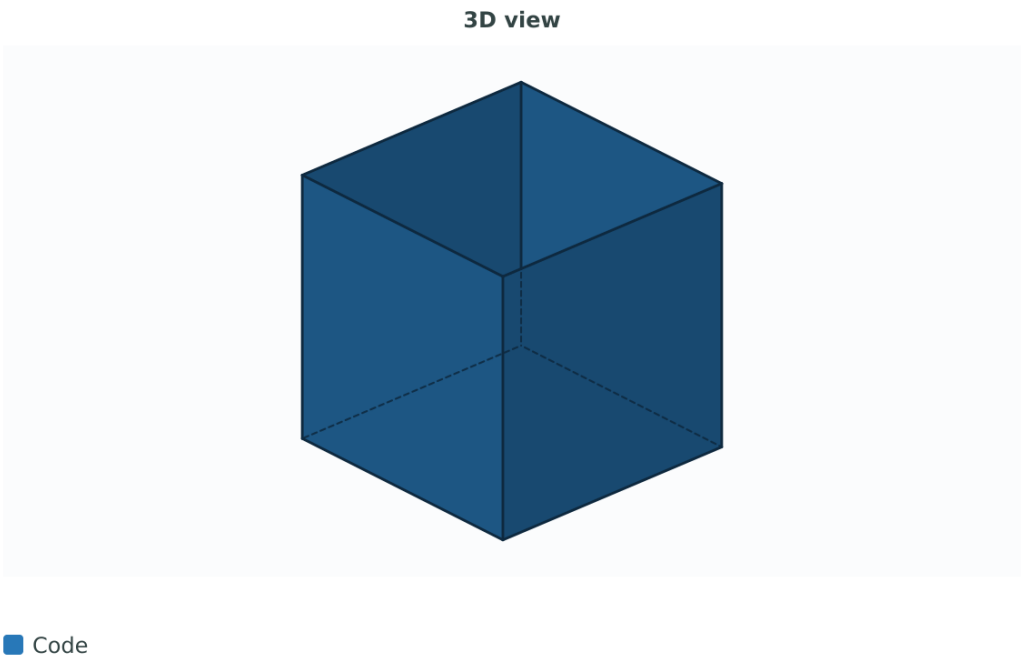
Code

```
SELECT ST_IsClosed('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
       ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
       ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
       ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)) )');
```

Output

```
f
```

Figure



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

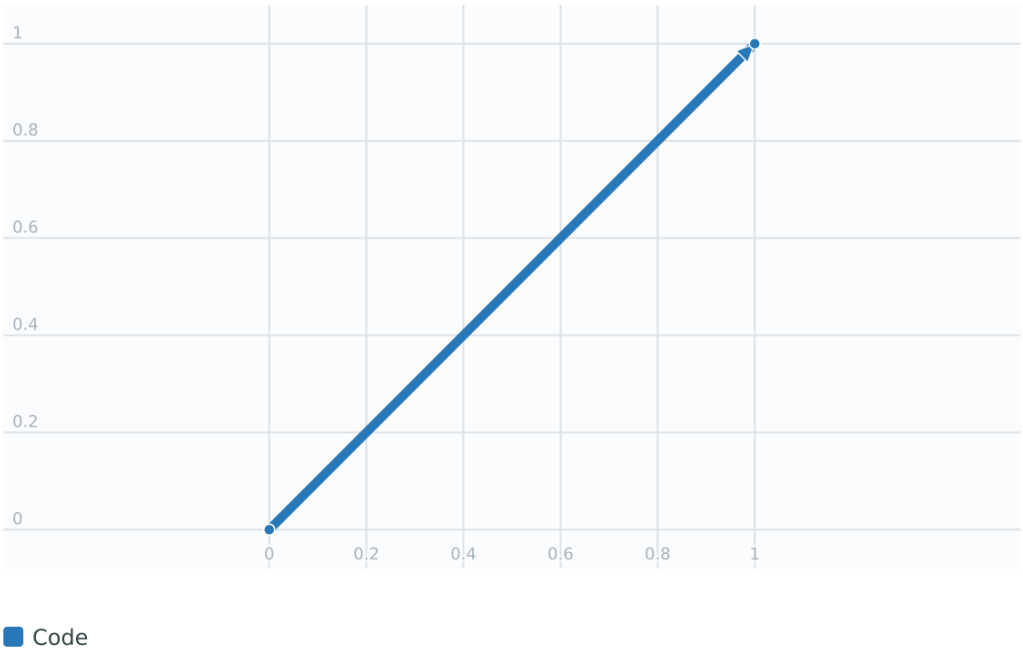
Code

```
SELECT ST_IsCollection('LINESTRING(0 0,1 1)'::geometry);
```

Output

```
f
```

Figure



Code

```
SELECT ST_IsCollection('MULTIPOINT EMPTY'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsCollection('MULTIPOINT((0 0))'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsCollection('MULTIPOINT((0 0),(42 42))'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsCollection('GEOMETRYCOLLECTION(POINT(0 0))'::geometry);
```

Output

```
t
```

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☒。

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

Code

```
SELECT ST_IsEmpty('GEOMETRYCOLLECTION EMPTY'::geometry);
```

Output

```
t
```

Code

```
SELECT ST_IsEmpty('POLYGON EMPTY'::geometry);
```

Output

```
t
```

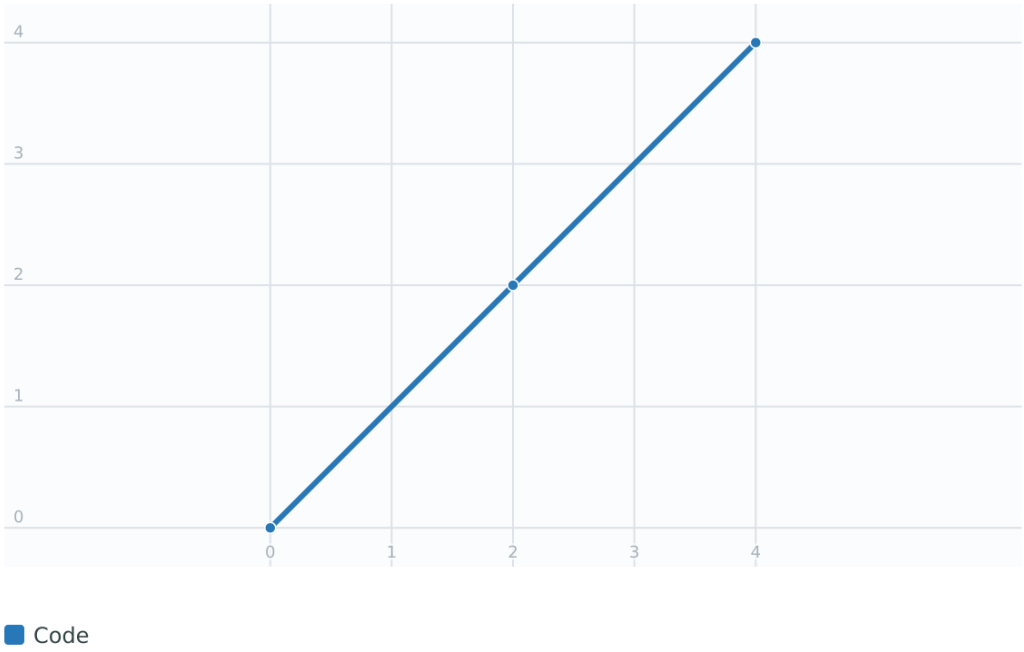
Code

```
SELECT ST_IsEmpty('POLYGON((1 2,3 4,5 6,1 2))'::geometry);
```

Output

```
f
```

Figure



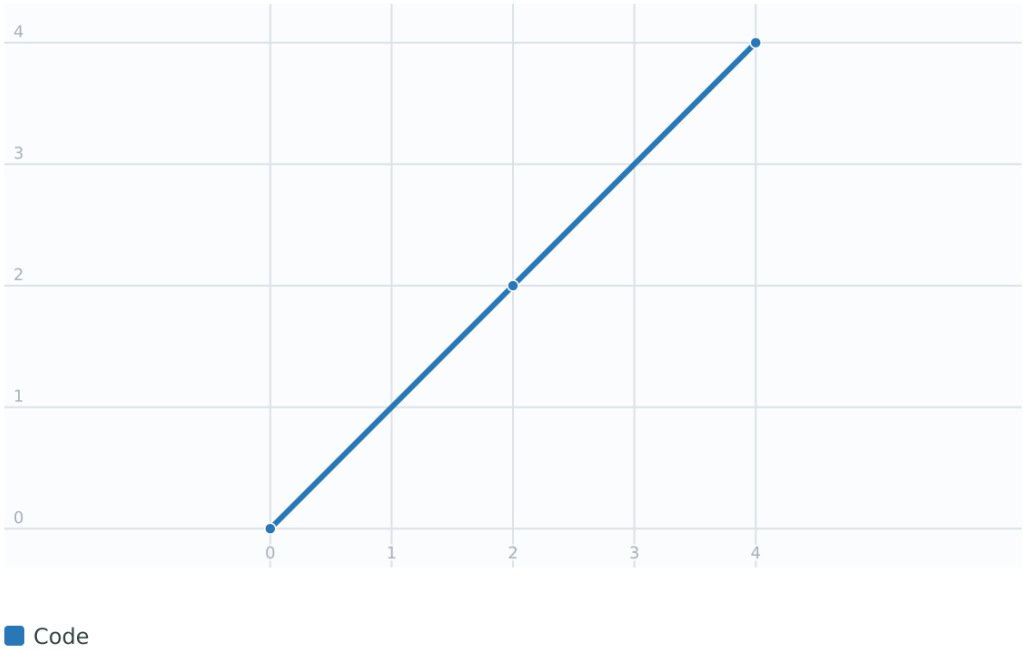
Code

```
SELECT ST_IsEmpty('POLYGON((1 2,3 4,5 6,1 2))'::geometry) = false;
```

Output

```
t
(1 row)
```

Figure



Code

```
SELECT ST_IsEmpty('CIRCULARSTRING EMPTY'::geometry);
```

Output

```
t
```

7.4.22 ST_IsPolygonCCW

ST_IsPolygonCCW — ☒☒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

Returns TRUE if this LINESTRING is both ST_IsClosed (ST_StartPoint(g) ~= ST_EndPoint(g)) and ST_IsSimple (does not self intersect).

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

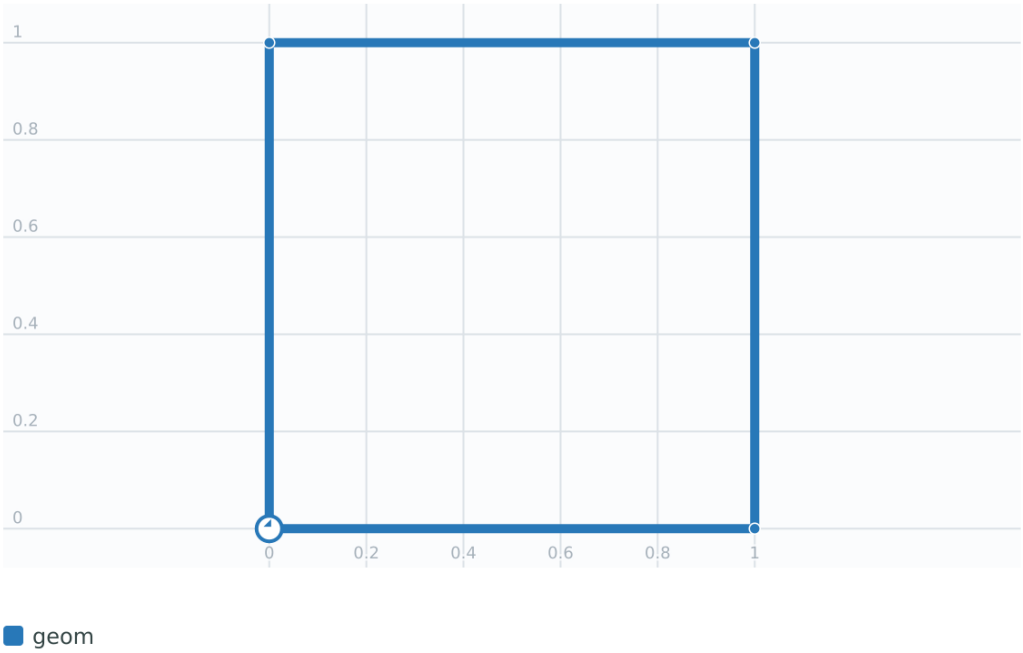
Code

```
SELECT ST_IsRing(geom), ST_IsClosed(geom), ST_IsSimple(geom)
FROM (SELECT 'LINESTRING(0 0,0 1,1 1,1 0,0 0)'::geometry AS geom) AS foo;
```

Output

```
st_isring | st_isclosed | st_issimple
-----------+-------------+-------------
 t         | t           | t
(1 row)
```

Figure



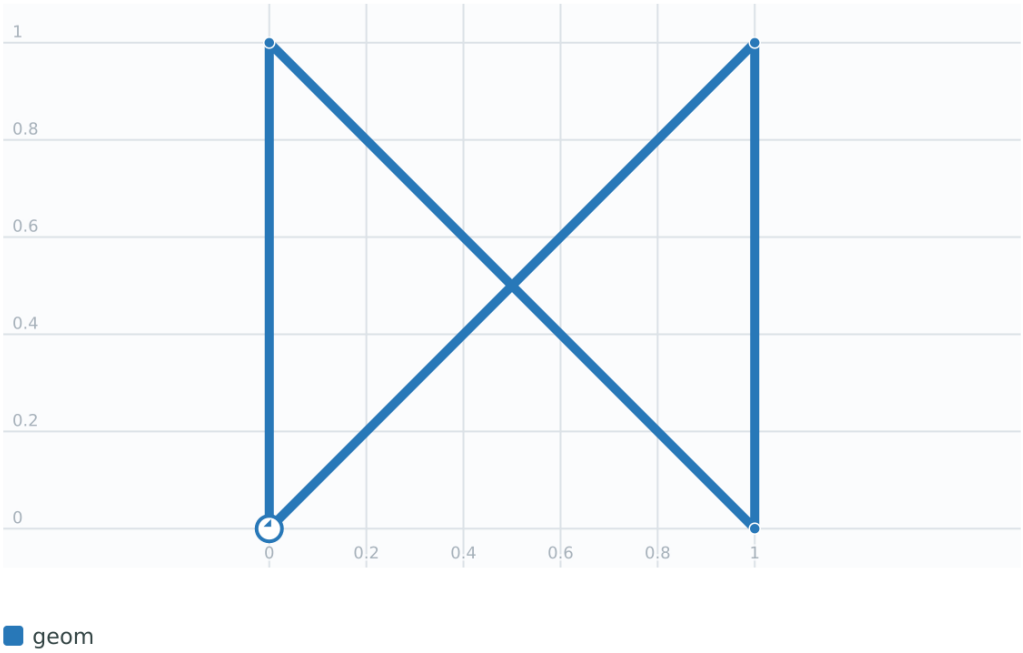
Code

```
SELECT ST_IsRing(geom), ST_IsClosed(geom), ST_IsSimple(geom)
FROM (SELECT 'LINESTRING(0 0,0 1,1 0,1 1,0 0)'::geometry AS geom) AS foo;
```

Output

```
st_isring | st_isclosed | st_issimple
-----------+-------------+-------------
 f         | t           | f
(1 row)
```

Figure



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

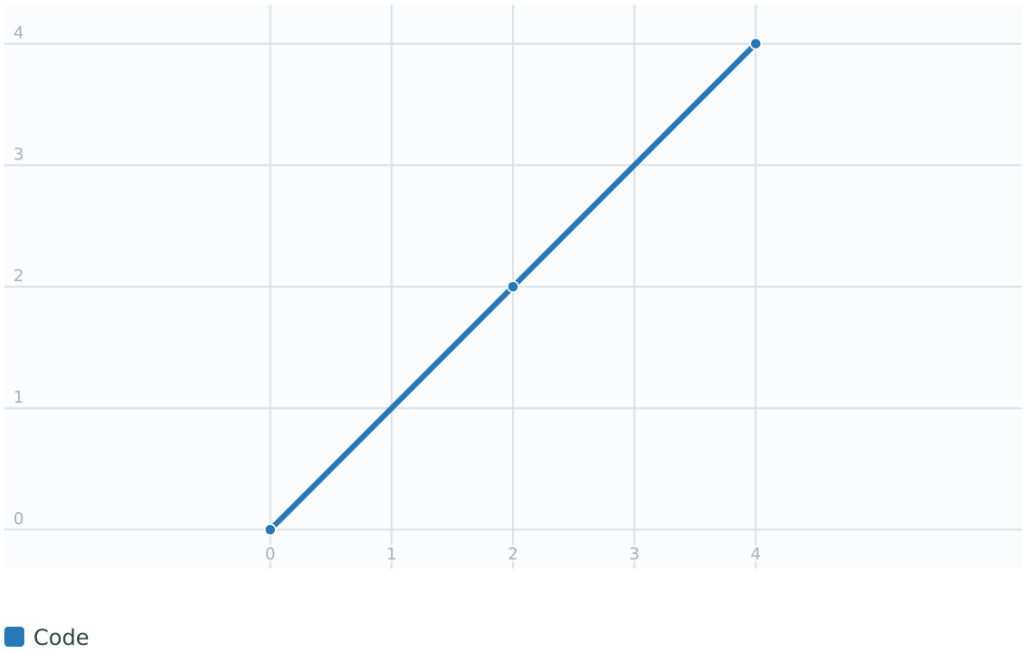
Code

```
SELECT ST_IsSimple('POLYGON((1 2,3 4,5 6,1 2))');
```

Output

```
f
```

Figure



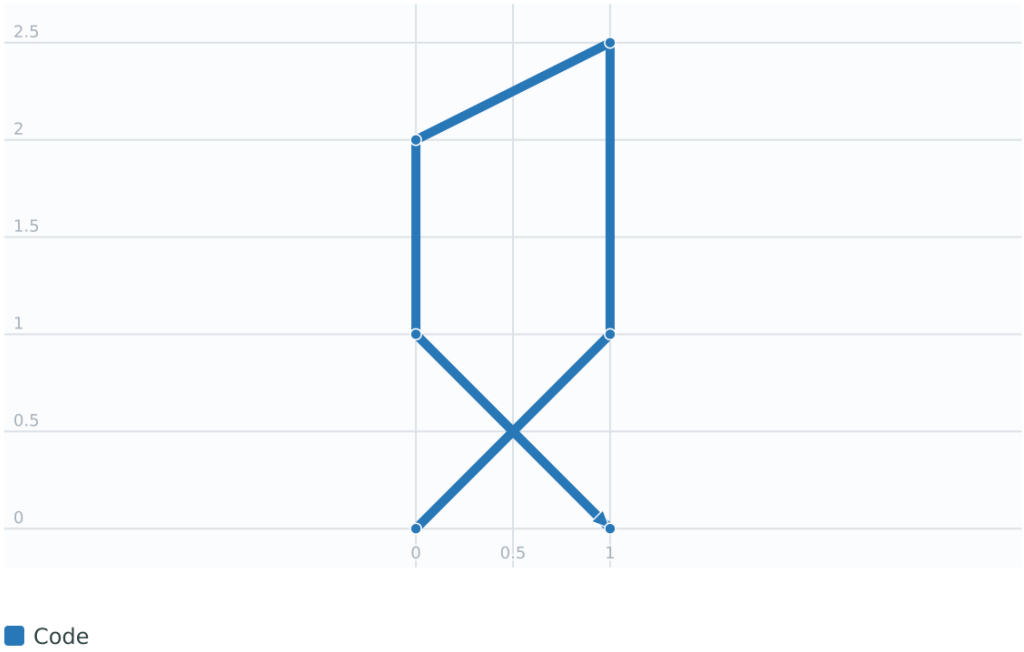
Code

```
SELECT ST_IsSimple('LINESTRING(1 1,2 2,2 3.5,1 3,1 2,2 1)');
```

Output

```
f
```

Figure



相关信息

ST_IsValid

7.4.26 ST_IsSimpleDetail

ST_IsSimpleDetail — Report locations of self-intersection or self-tangency, if any.

Synopsis

simple_detail **ST_IsSimpleDetail**(geometry geom, integer findAllLocations = 0, integer boundaryNodeRule = 1);

描述

Returns a `simple_detail` row, containing a boolean (`simple`) stating if a geometry is simple and a geometry (`location`) pointing out where it is non-simple.

The optional `findAllLocations` parameter can be set to 1 to report all non-simple locations, rather than just one.

The optional `boundaryNodeRule` values are:

- 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。

Availability: 3.7.0

示例

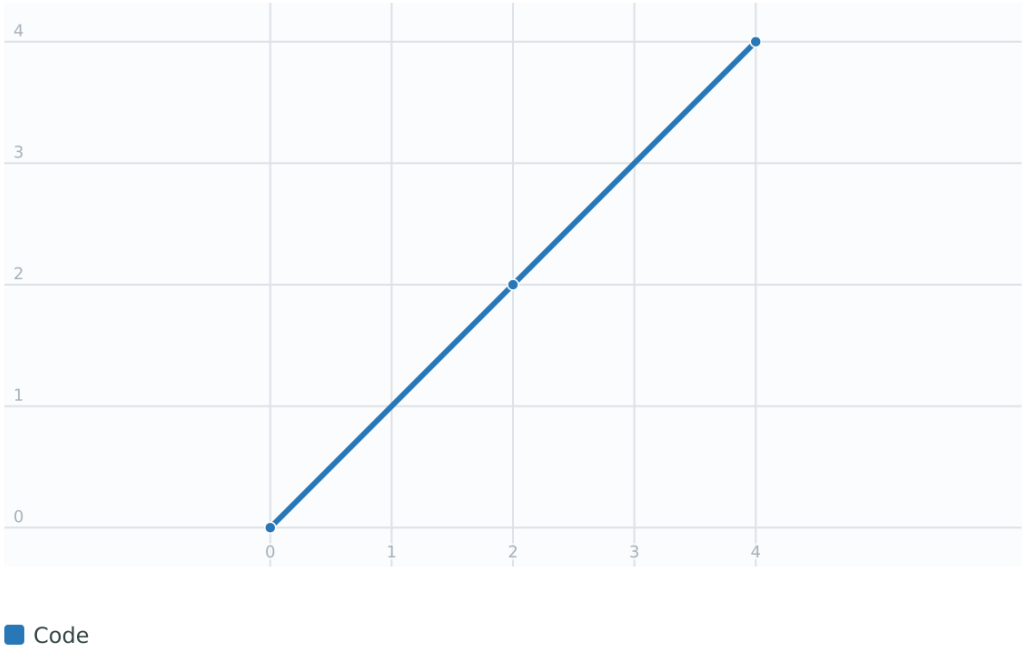
Code

```
SELECT ST_IsSimple('POLYGON((1 2,3 4,5 6,1 2))');
```

Output

```
f
```

Figure



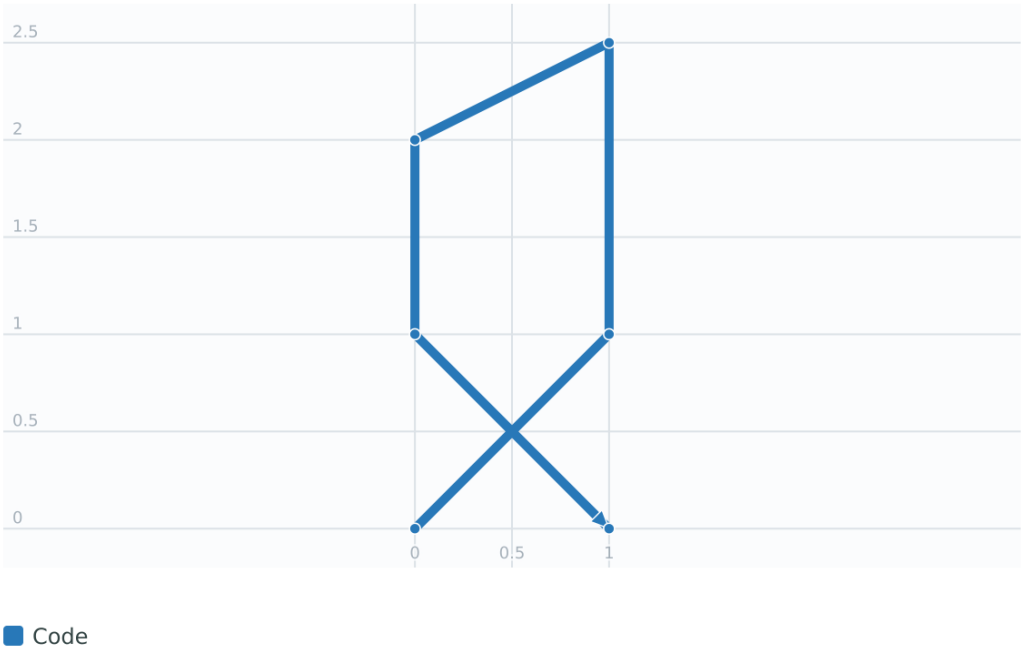
Code

```
SELECT ST_IsSimple('LINESTRING(1 1,2 2,2 3.5,1 3,1 2,2 1)');
```

Output

```
f
```

Figure



相关信息

ST_IsSimple

7.4.27 ST_M

ST_M — 返回点的 M 坐标。

Synopsis

float **ST_M**(geometry a_point);

描述

返回点的 M 坐标值，如果无效，则返回 NULL。输入必须是点。



Note
此函数不是 OGC 规范的一部分，但在此规列出以完成点坐标提取器功能列表。


此方法符合了 SQL 1.1 的 OGC 符合功能规范。


该方法符合了 SQL/MM 规范。


该函数支持 3d 并且不会丢失 z-index。

示例

Code

```
SELECT ST_M('POINT(1 2 3 4)');
```

Output

```
4
```

相关信息

ST_GeomFromEWKT, ST_X, ST_Y, ST_Z

7.4.28 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

Compare the memory used by all geometries in a Massachusetts town dataset with the subset for Boston.

Code

```
SELECT pg_size_pretty(sum(ST_MemSize(geom))) AS totgeomsum,
       pg_size_pretty(sum(CASE WHEN town = 'BOSTON' THEN ST_MemSize(geom) ELSE 0 END)) AS ↩
    bossum,
       cast(
           sum(CASE WHEN town = 'BOSTON' THEN ST_MemSize(geom) ELSE 0 END) * 100.0
           / sum(ST_MemSize(geom)) AS numeric(10, 2)
       ) AS perbos
FROM towns;
```

Output

```
totgeomsum | bossum | perbos
------------+--------+--------
 1522 kB    | 30 kB  |   1.99
(1 row)
```

Return the memory size of a CircularString in bytes.

Code

```
SELECT ST_MemSize('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'::geometry);
```

Output

```
st_memsize
------------
         80
(1 row)
```

Calculate the percentage of a table’s total size used by geometry data.

Code

```
SELECT pg_total_relation_size('public.neighborhoods') AS fulltable_size,
       sum(ST_MemSize(geom)) AS geomsize,
       sum(ST_MemSize(geom)) * 100.0
           / pg_total_relation_size('public.neighborhoods') AS pergeom
FROM neighborhoods;
```

Output

```
fulltable_size | geomsize |         pergeom
---------------+----------+-------------------------
         262144 |    96238 | 36.71188354492187500000
(1 row)
```

7.4.29 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

Code

```
SELECT ST_NDims('POINT(1 1)') As d2point,
ST_NDims('POINT(1 1 2)') As d3point,
ST_NDims('POINTM(1 1 0.5)') As d2pointm;
```

Output

```
d2point | d3point | d2pointm
---------+---------+----------
       2 |       3 |        3
```

相关信息

ST_CoordDim, ST_Dimension, ST_GeomFromEWKT

7.4.30 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

Count the points in a 2D LineString.

Code

```
SELECT ST_NPoints('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)');
```

Output

```
st_npoints
------------
          4
(1 row)
```

Count the points in a 3D LineString.

Code

```
SELECT ST_NPoints('LINESTRING(77.29 29.07 1,77.42 29.26 0,77.27 29.31 -1,77.29 29.07 3)');
```

Output

```
st_npoints
------------
          4
(1 row)
```

相关信息

ST_NumPoints

7.4.31 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

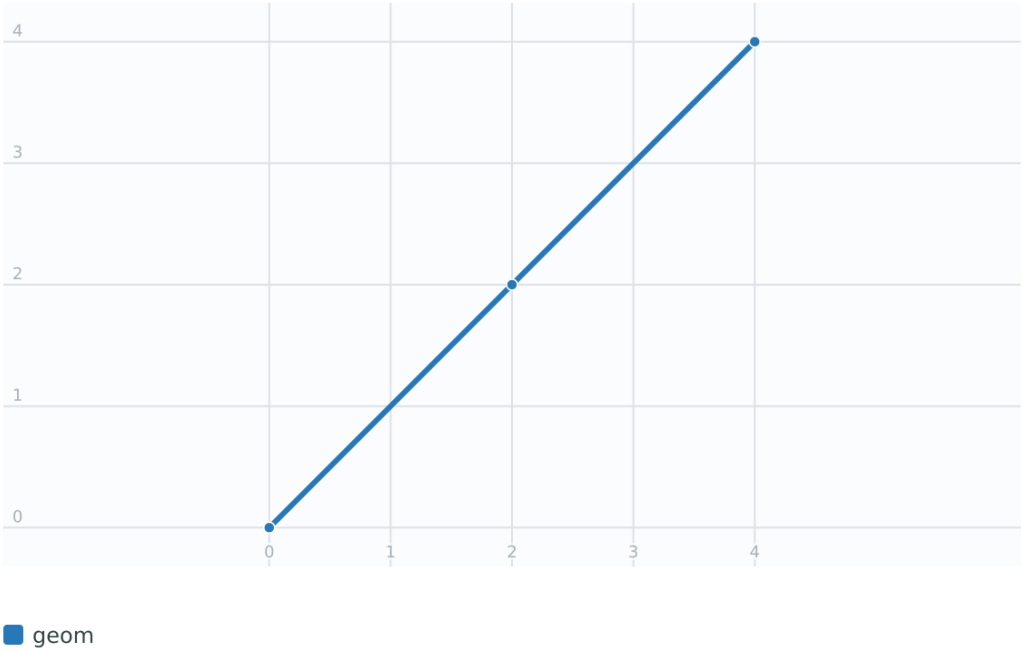
Code

```
SELECT ST_NRings(geom) As Nrings, ST_NumInteriorRings(geom) As ninterrings
FROM (SELECT 'POLYGON((1 2,3 4,5 6,1 2))'::geometry As geom) As foo;
```

Output

```
nrings | ninterrings
--------+-------------
      1 |           0
(1 row)
```

Figure



相关信息

ST_NumInteriorRings

7.4.32 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。

Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and return 1; use ST_NumPatches to count faces.

✔ ☒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

This example returns 1; prior versions would have returned NULL.

Code

```
SELECT ST_NumGeometries('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)');
```

Output

```
st_numgeometries
-----------------
                1
(1 row)
```

A MultiPoint counts as one geometry within a GeometryCollection.

Code

```
SELECT ST_NumGeometries('GEOMETRYCOLLECTION(MULTIPOINT((-2 3),(-2 2)),
LINESTRING(5 5,10 10),
POLYGON((-7 4.2,-7.1 5,-7.1 4.3,-7 4.2)))');
```

Output

```
st_numgeometries
-----------------
                3
(1 row)
```

相关信息

ST_GeometryN, ST_Multi

7.4.33 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

Count the interior rings of a Polygon.

Code

```
SELECT gid, field1, field2, ST_NumInteriorRings(geom) AS numholes
FROM sometable;
```

For a MultiPolygon, dump its component Polygons and sum their interior ring counts.

Code

```
SELECT gid, field1, field2, SUM(ST_NumInteriorRings(geom)) AS numholes
FROM (
    SELECT gid, field1, field2, (ST_Dump(geom)).geom AS geom
    FROM sometable
) AS foo
GROUP BY gid, field1, field2;
```

相关信息

ST_NumInteriorRing, ST_InteriorRingN

7.4.34 ST_NumInteriorRing

ST_NumInteriorRing — Returns the number of interior rings (holes) of a Polygon. Alias for ST_NumInteriorR

Synopsis

integer **ST_NumInteriorRing**(geometry a_polygon);

描述

This function is an alias for ST_NumInteriorRings.

可用性：1.2.2

相关信息

ST_NumInteriorRings, ST_InteriorRingN

7.4.35 ST_NumPatches

ST_NumPatches — Returns the number of faces in a PolyhedralSurface or TIN.

Synopsis

integer **ST_NumPatches**(geometry g1);

描述

Returns the number of faces in a PolyhedralSurface or TIN. Returns NULL for other geometries. Unlike ST_NumGeometries, this function counts individual faces.

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

Code

```
SELECT ST_NumPatches('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )');
```

Output

```
st_numpatches
---------------
             6
(1 row)
```

相关信息

ST_GeomFromEWKT, ST_NumGeometries, ST_PatchN

7.4.36 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

Code

```
SELECT ST_NumPoints('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)');
```

Output

```
st_numpoints
--------------
           4
(1 row)
```

相关信息

ST_NPoints

7.4.37 ST_PatchN

ST_PatchN — Returns the Nth face of a PolyhedralSurface or TIN.

Synopsis

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

描述

Returns the 1-based Nth face of a PolyhedralSurface or TIN. Returns NULL for other geometries or an out-of-range index. Use this function to access faces, because ST_GeometryN treats PolyhedralSurface and TIN as unitary geometries.

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。

✅ 此函数支持三角形和不☒☒三角网面 (TIN)。

示例

Extract the 2nd face of the polyhedral surface.

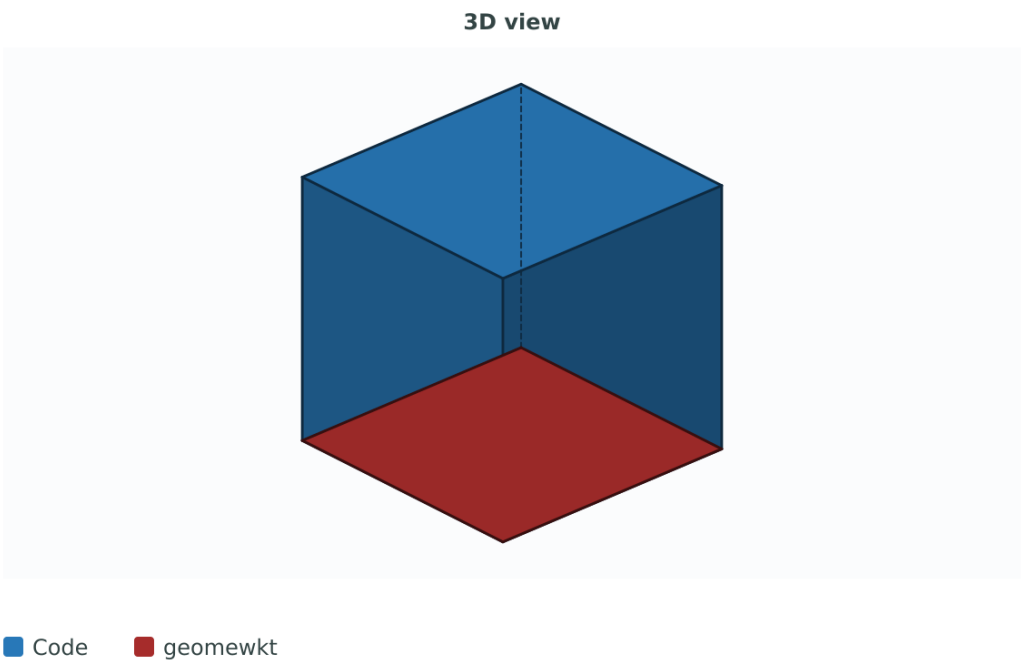
Code

```
SELECT ST_PatchN(geom, 2) As geomewkt
FROM (
VALUES ('POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry) ) As ↩
    foo(geom);
```

Output

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

Figure



相关信息

ST_AsEWKT, ST_GeomFromEWKT, ST_Dump, ST_GeometryN, ST_NumGeometries, ST_NumPatches

7.4.38 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**
> Index is 1-based as for OGC specs since version 0.8.0. Backward indexing (negative index) is not in OGC. Previous versions implemented this as 0-based instead.

> **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⍰。

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

This example extracts all POINT geometries from a LineString.

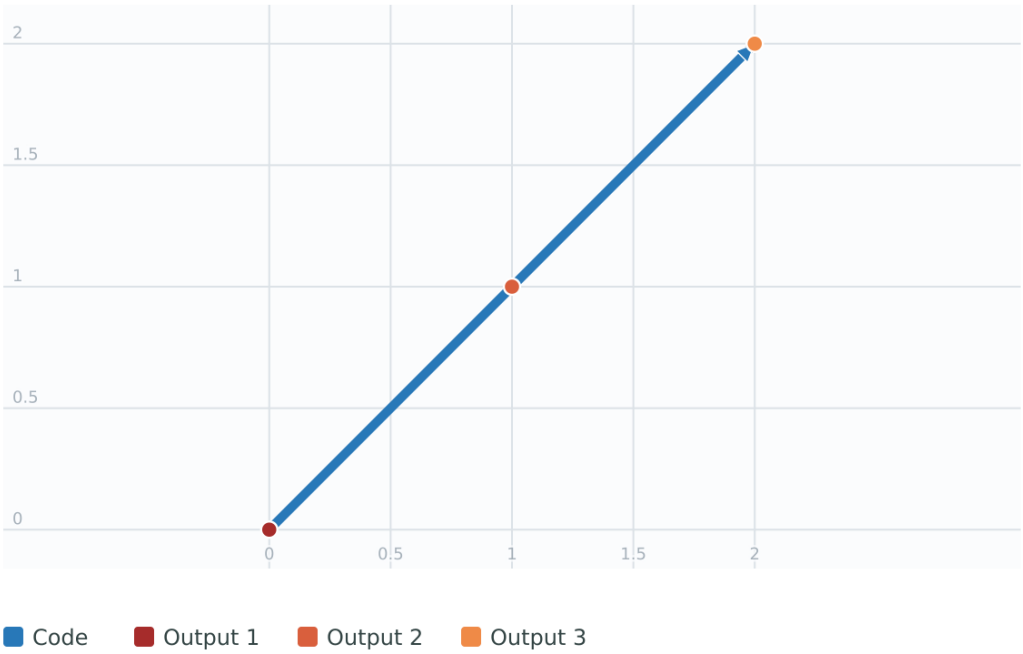
Code

```
SELECT ST_PointN(
      column1,
      generate_series(1, ST_NPoints(column1))
   )
FROM ( VALUES ('LINESTRING(0 0,1 1,2 2)'::geometry) ) AS foo;
```

Output

```
POINT(0 0)
 POINT(1 1)
 POINT(2 2)
(3 rows)
```

Figure



This example uses a CircularString.

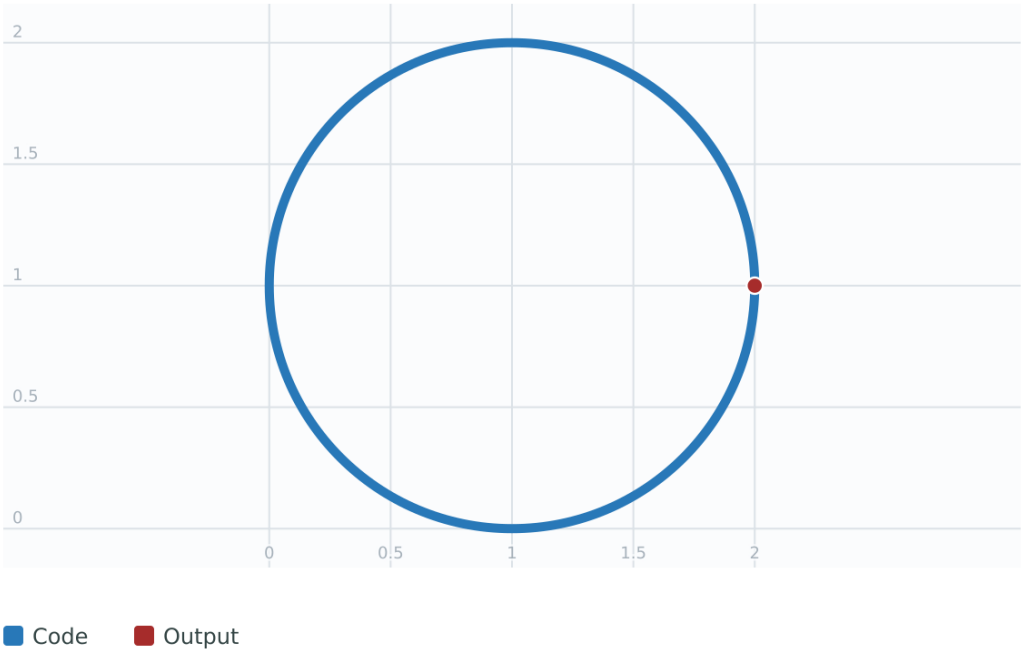
Code

```
SELECT ST_PointN('CIRCULARSTRING(1 2,3 2,1 2)', 2);
```

Output

```
POINT(3 2)
```

Figure



The index is 1-based. A negative index counts backward from the end of the LineString.

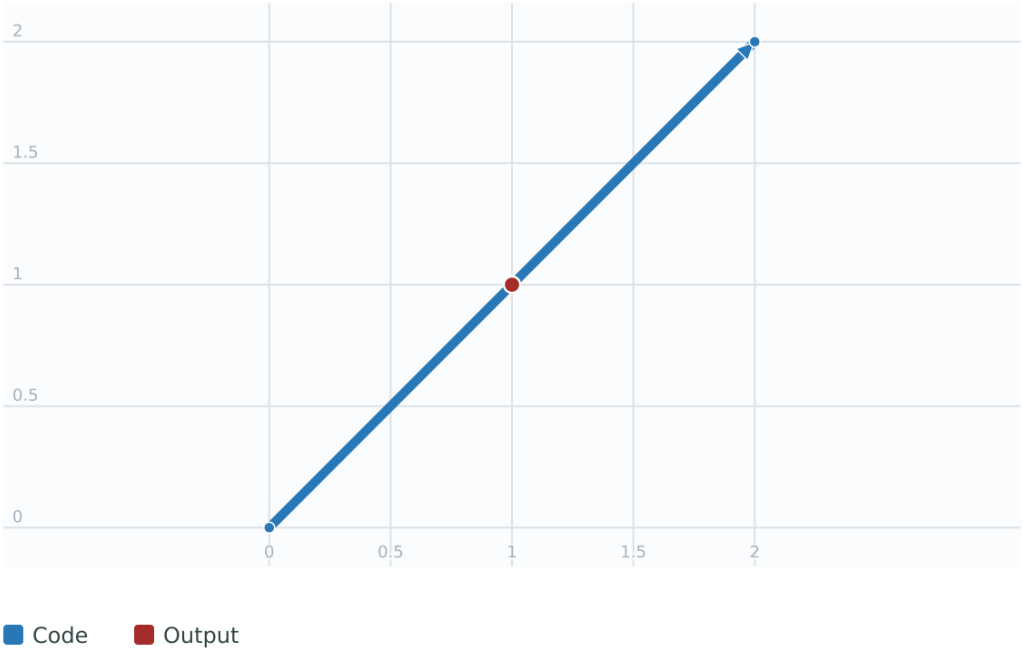
Code

```
SELECT f
FROM ST_GeomFromText('LINESTRING(0 0 0,1 1 1,2 2 2)') AS g
  CROSS JOIN ST_PointN(g, -2) AS f;
```

Output

```
POINT Z (1 1 1)
```

Figure



相关信息

ST_NPoints

7.4.39 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

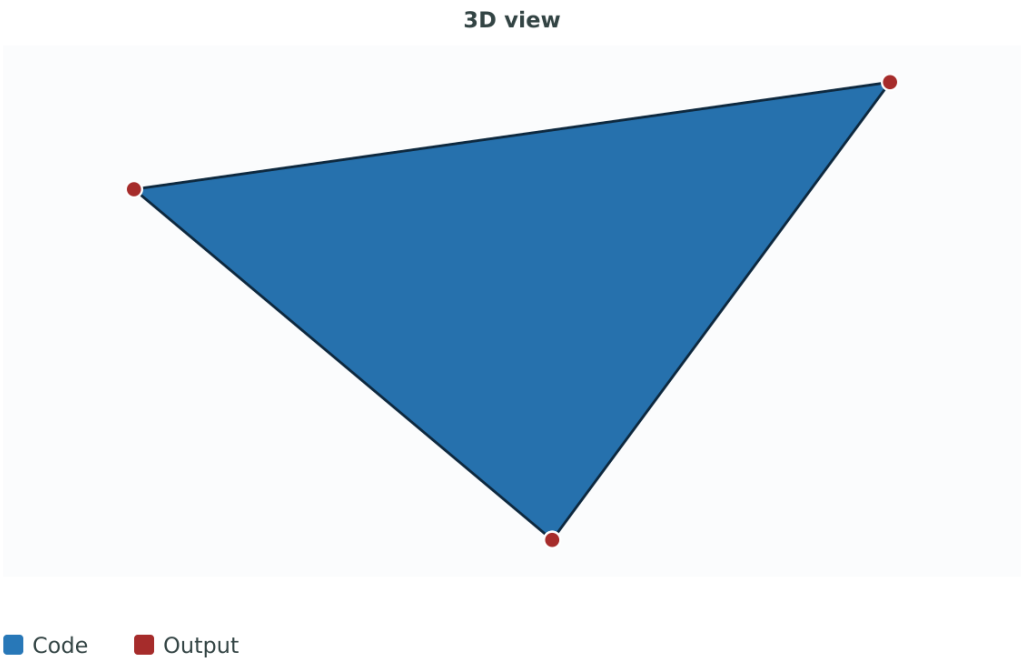
Code

```
SELECT ST_Points('POLYGON Z ((30 10 4,10 30 5,40 40 6,30 10 4))');
```

Output

```
MULTIPOINT Z ((30 10 4),(10 30 5),(40 40 6),(30 10 4))
```

Figure



相关信息

ST_RemoveRepeatedPoints, ST_DumpPoints

7.4.40 ST_StartPoint

ST_StartPoint — Returns the first point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_StartPoint**(geometry geomA);

描述

Returns the first point of a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry as a `POINT`. For other geometries, returns the first point in coordinate order.

☒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☒。

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

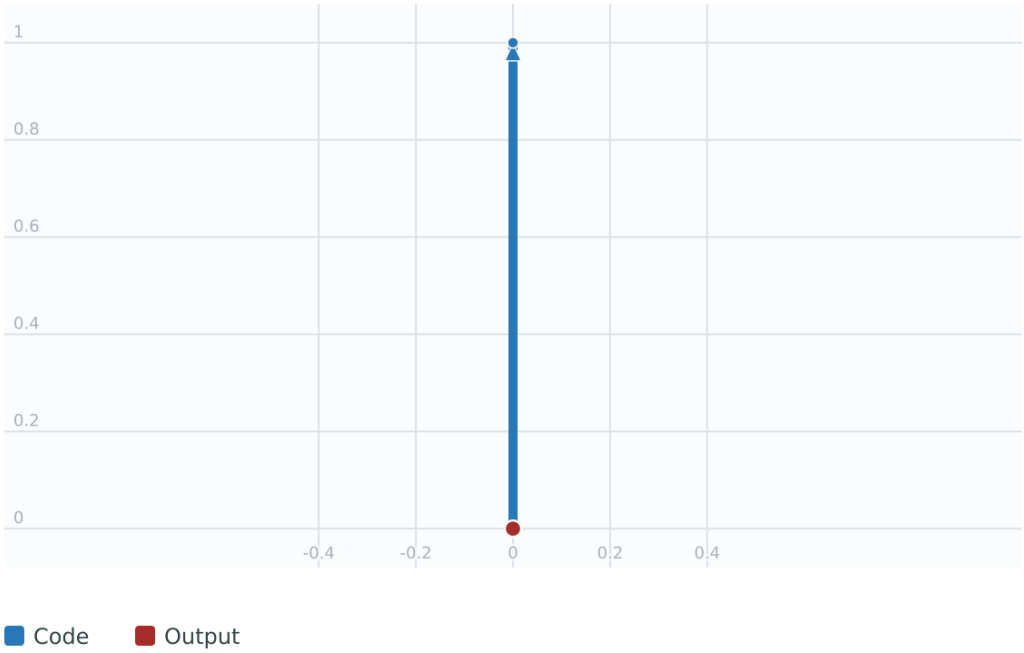
Code

```
SELECT ST_StartPoint('LINESTRING(0 1,0 2)'::geometry);
```

Output

```
POINT(0 1)
```

Figure



Start point of a Point is the Point itself

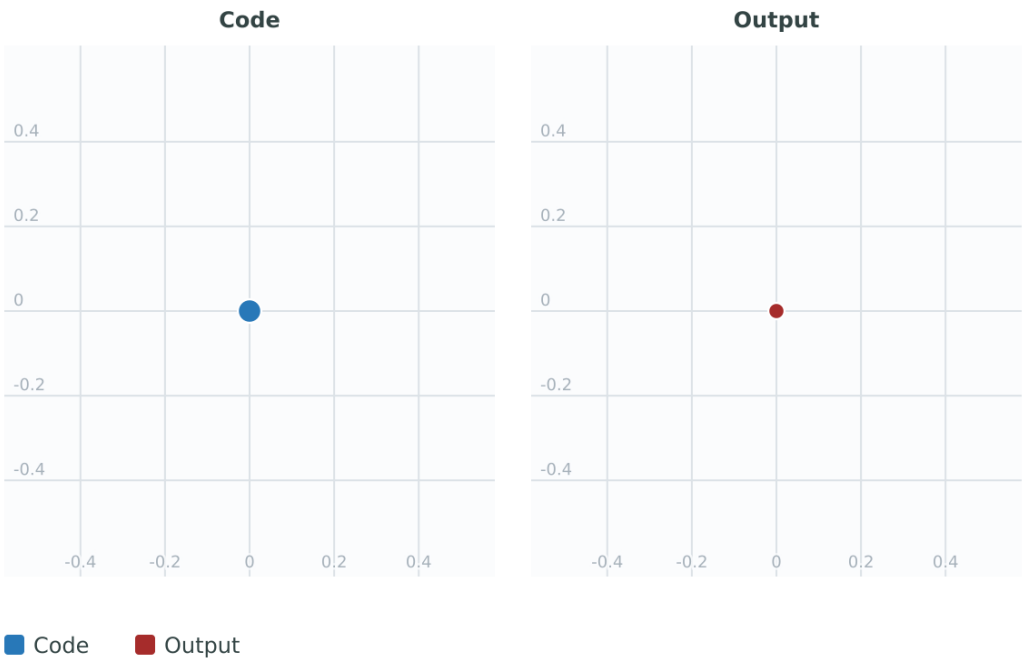
Code

```
SELECT ST_StartPoint('POINT(0 1)'::geometry);
```

Output

```
POINT(0 1)
```

Figure



3D LineString 起点

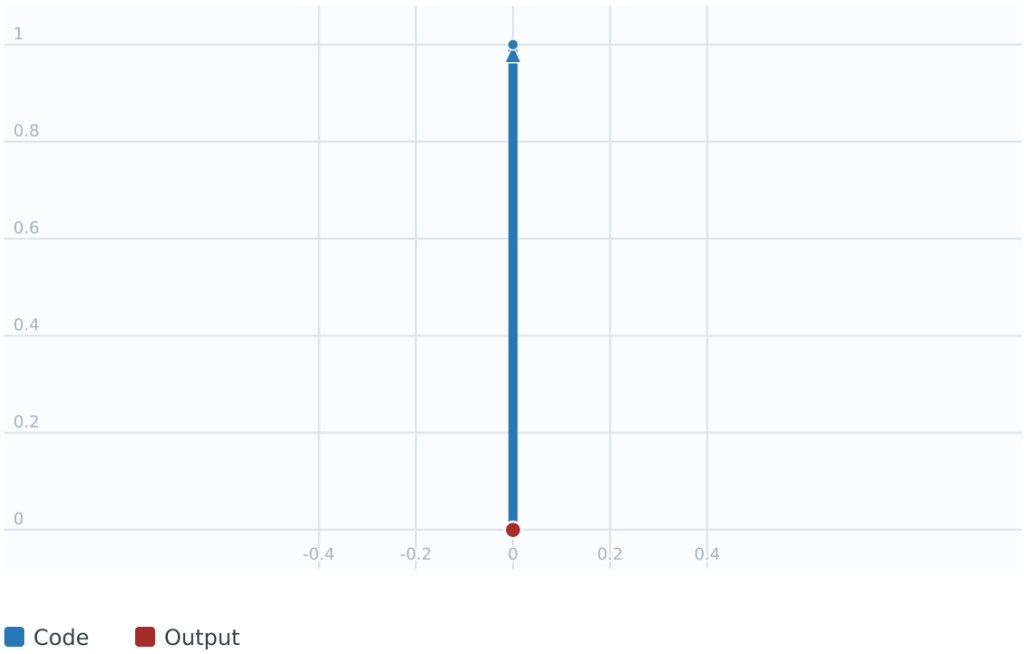
Code

```
SELECT ST_StartPoint('LINESTRING(0 1 1,0 2 2)'::geometry);
```

Output

```
POINT(0 1 1)
```

Figure



CircularString 起点

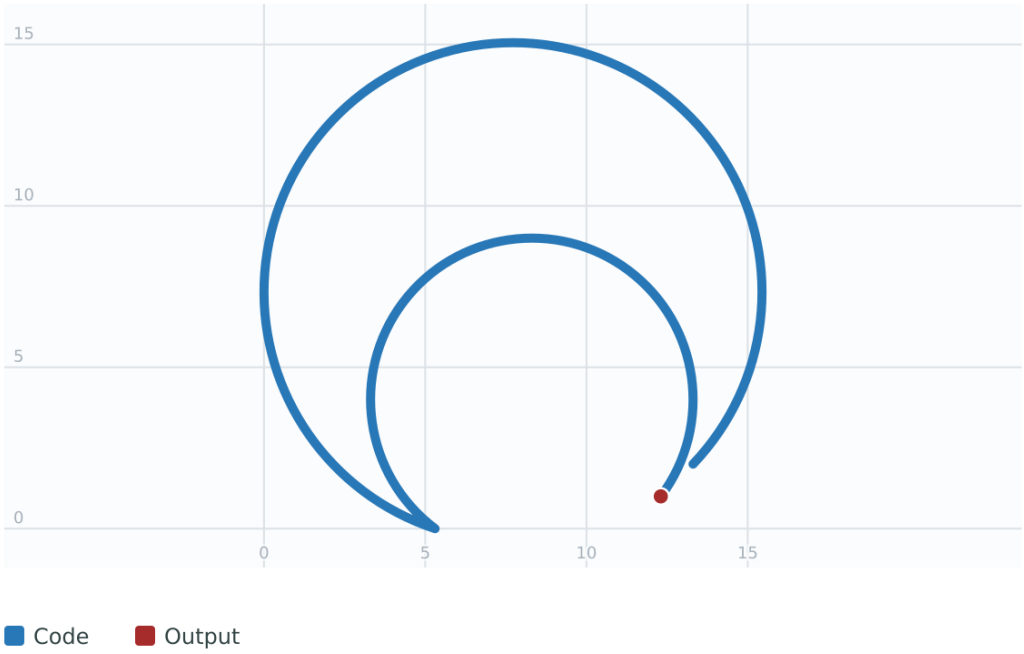
Code

```
SELECT ST_StartPoint('CIRCULARSTRING(5 2,-3 2,-2 1,-4 2,6 3)'::geometry);
```

Output

```
POINT(5 2)
```

Figure



Start point of a NURBSCurve

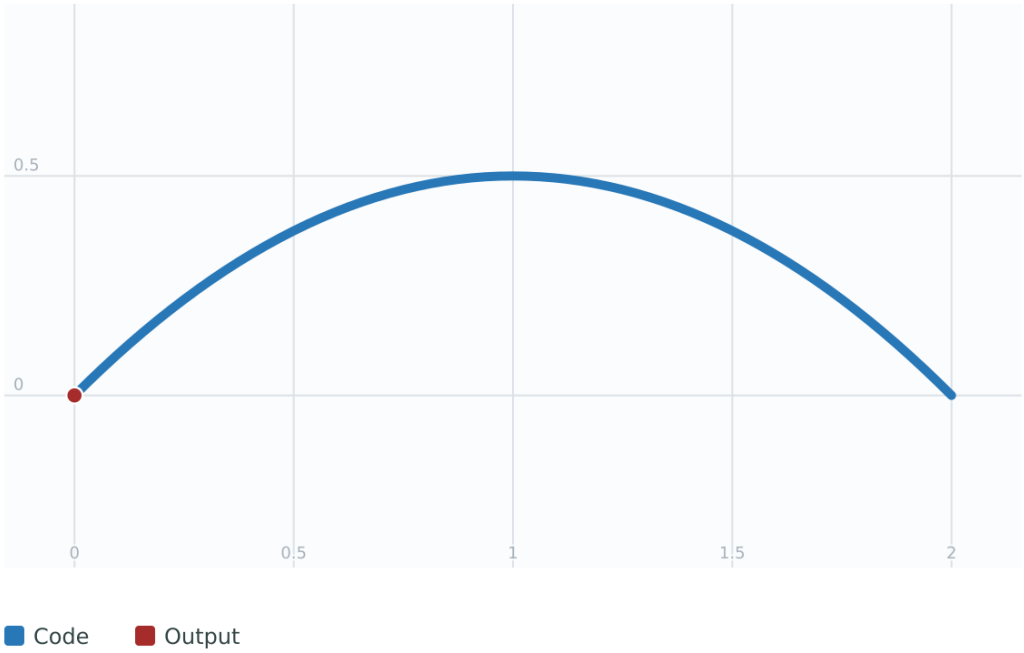
Code

```
SELECT ST_StartPoint('NURBSCURVE(2, (0 0, 1 1, 2 0))'::geometry);
```

Output

```
POINT(0 0)
```

Figure



Build edge node references from the start and end points of a LineString table. The example uses exact endpoint equality, so linework with nearly coincident endpoints should be snapped or otherwise cleaned first.

Code

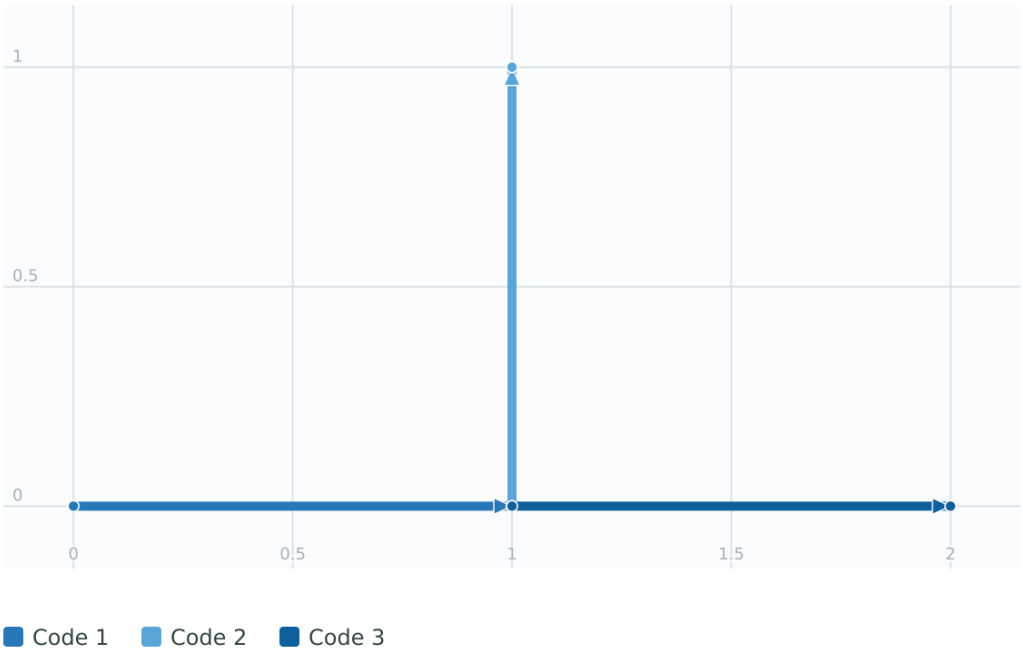
```
WITH roads(road_id, geom) AS (
  VALUES
    (1, 'LINESTRING(0 0, 1 0)'::geometry),
    (2, 'LINESTRING(1 0, 1 1)'::geometry),
    (3, 'LINESTRING(1 0, 2 0)'::geometry)
),
endpoints AS (
  SELECT road_id, 'from' AS end_name, ST_StartPoint(geom) AS geom,
         ST_AsEWKB(ST_StartPoint(geom)) AS endpoint_key
  FROM roads
  UNION ALL
  SELECT road_id, 'to' AS end_name, ST_EndPoint(geom) AS geom,
         ST_AsEWKB(ST_EndPoint(geom)) AS endpoint_key
  FROM roads
),
nodes AS (
  SELECT dense_rank() OVER (ORDER BY ST_X(geom), ST_Y(geom), endpoint_key) AS node_id,
         endpoint_key
  FROM (
    SELECT DISTINCT ON (endpoint_key) endpoint_key, geom
    FROM endpoints
    ORDER BY endpoint_key
  ) AS distinct_endpoints
)
SELECT e.road_id,
       max(n.node_id) FILTER (WHERE e.end_name = 'from') AS from_node,
       max(n.node_id) FILTER (WHERE e.end_name = 'to') AS to_node
FROM endpoints AS e
JOIN nodes AS n USING (endpoint_key)
GROUP BY e.road_id
```

```
ORDER BY e.road_id;
```

Output

```
road_id | from_node | to_node
---------+-----------+---------
      1 |         1 |       2
      2 |         2 |       3
      3 |         2 |       4
```

Figure



相关信息

ST_EndPoint, ST_PointN

7.4.41 ST_StartM

ST_StartM — Returns the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

float8 **ST_StartM**(geometry geom);

描述

Returns the M coordinate of the first point of a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE` geometry. Returns `NULL` if the input geometry does not have an M dimension or is not a `LINESTRING`, `CIRCULARLINESTRING`, or `NURBSCURVE`.

此方法支持☒形字符串和曲☒。

示例

Start M value of a LineString

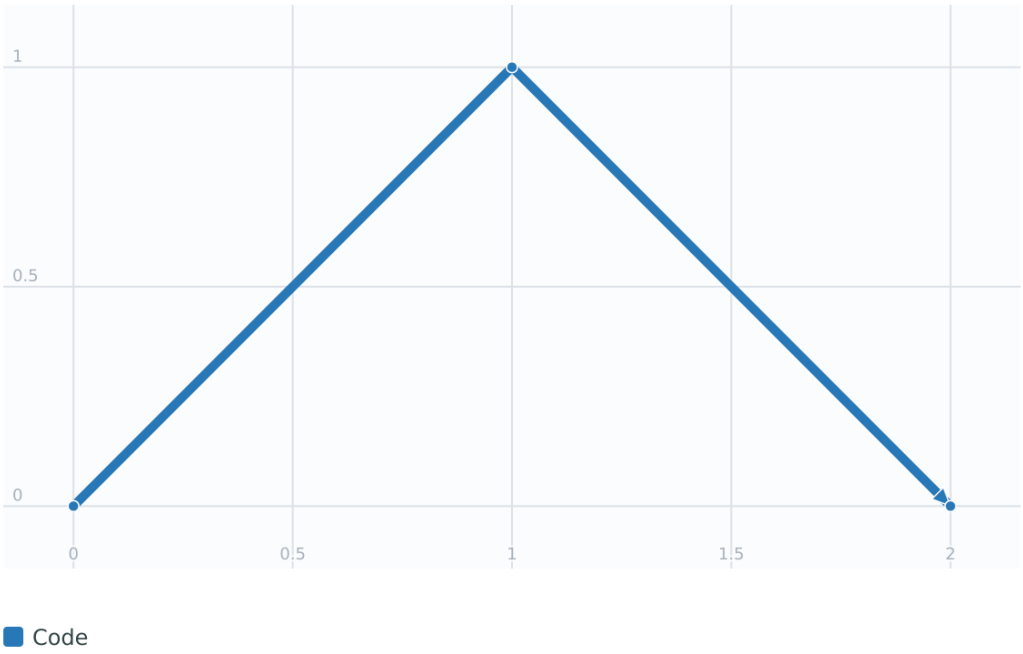
Code

```
SELECT ST_StartM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry);
```

Output

```
10
```

Figure



Start M value of a CircularString

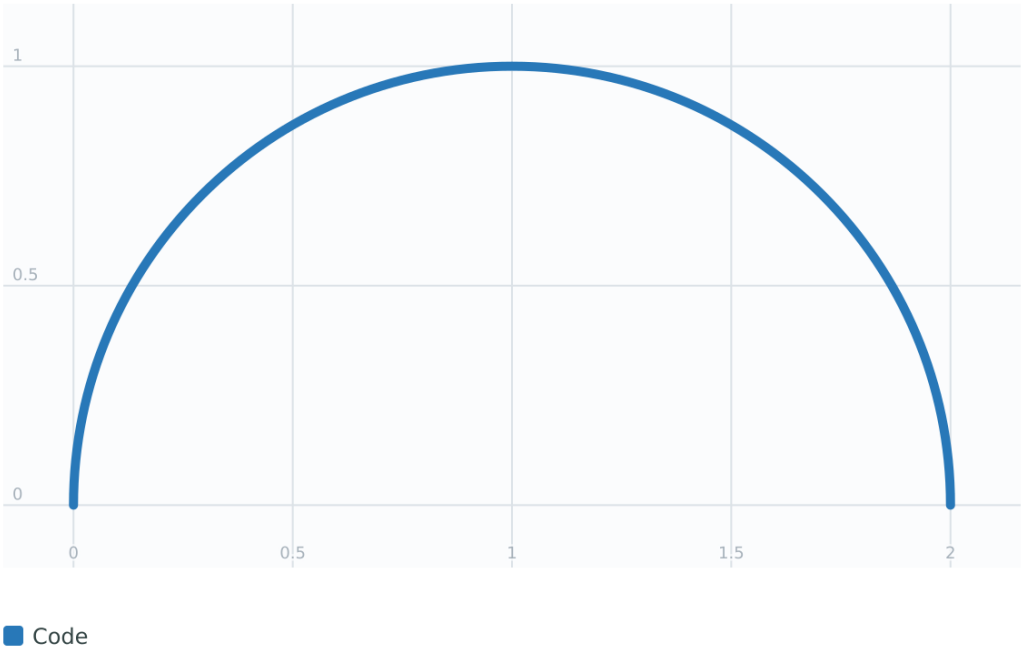
Code

```
SELECT ST_StartM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry);
```

Output

```
5
```

Figure



Start M value of a NURBSCurve

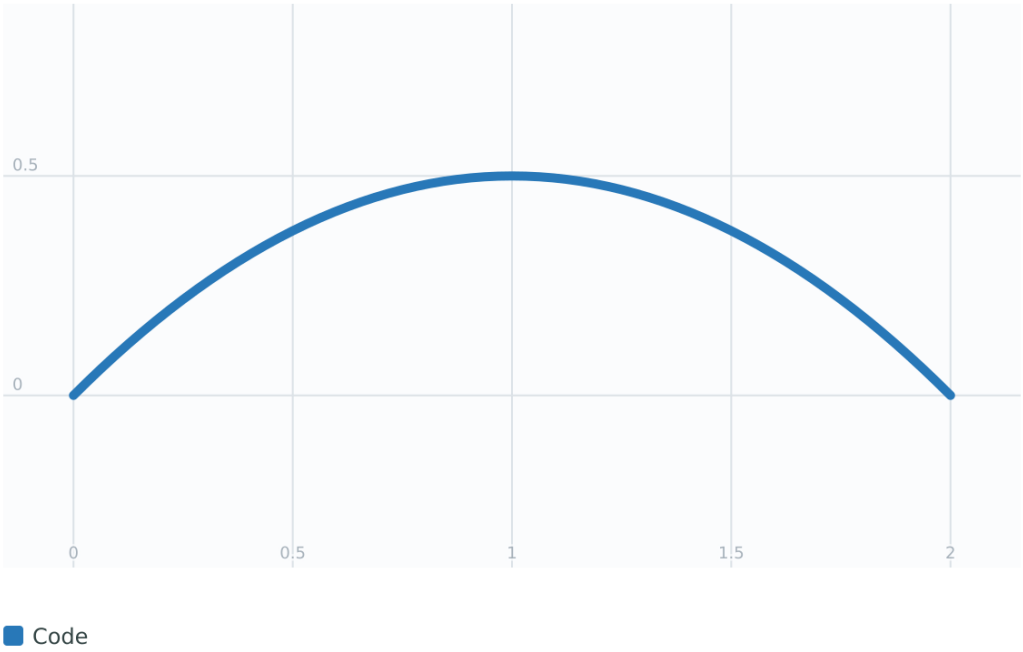
Code

```
SELECT ST_StartM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry);
```

Output

```
100
```

Figure



Geometry without M dimension returns NULL

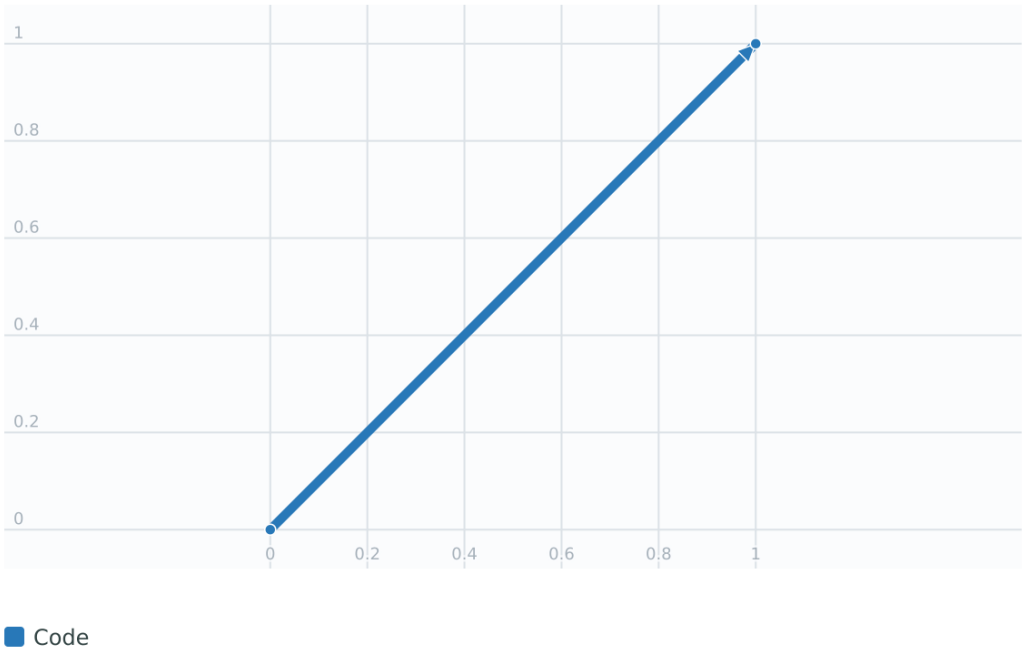
Code

```
SELECT ST_StartM('LINESTRING(0 0,1 1)'::geometry) IS NULL AS is_null;
```

Output

```
t
```

Figure



相关信息

ST_EndM, ST_SetStartM, ST_StartPoint

7.4.42 ST_EndM

ST_EndM — Returns the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

float8 **ST_EndM**(geometry geom);

描述

Returns the M coordinate of the last point of a LINESTRING, CIRCULARLINESTRING, or NURBSCURVE geometry. Returns NULL if the input geometry does not have an M dimension or is not a LINESTRING, CIRCULARLINESTRING, or NURBSCURVE.

此方法支持☒形字符串和曲☒。

示例

End M value of a LineString

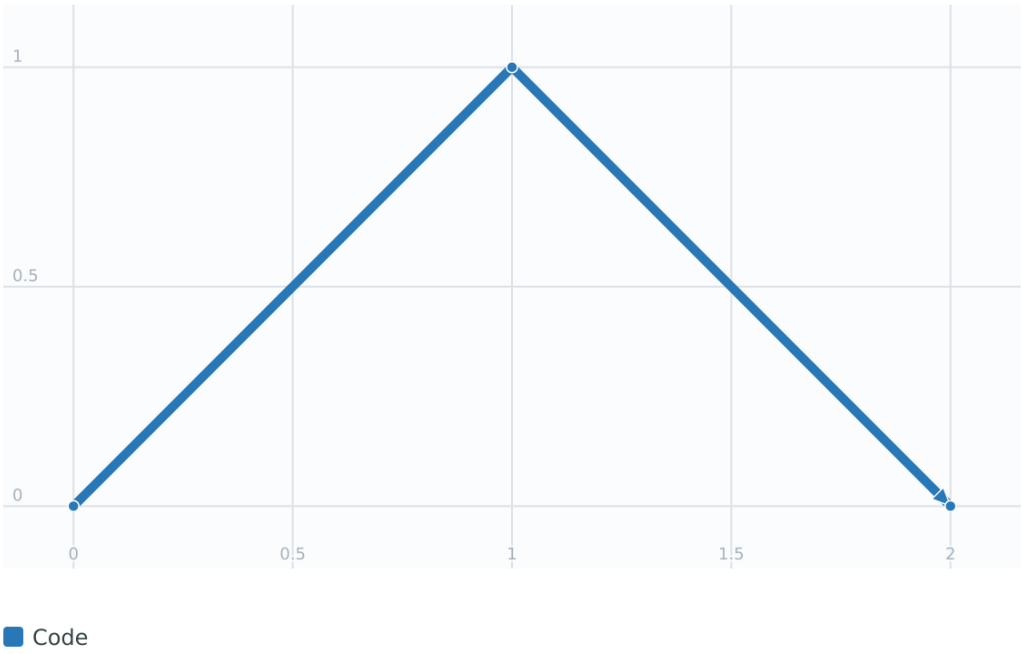
Code

```
SELECT ST_EndM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry);
```

Output

```
30
```

Figure



End M value of a CircularString

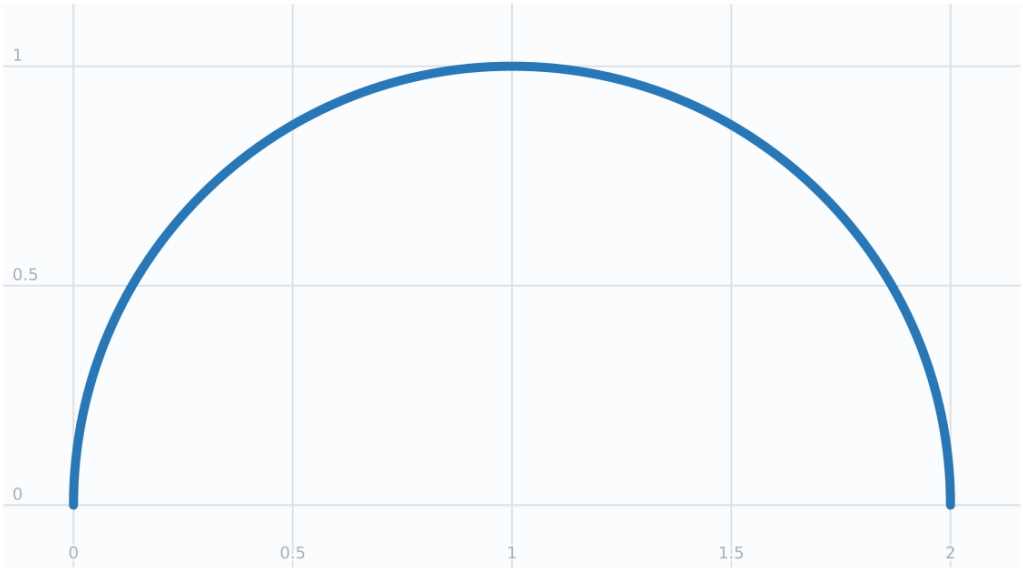
Code

```
SELECT ST_EndM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry);
```

Output

```
15
```

Figure



End M value of a NURBSCurve

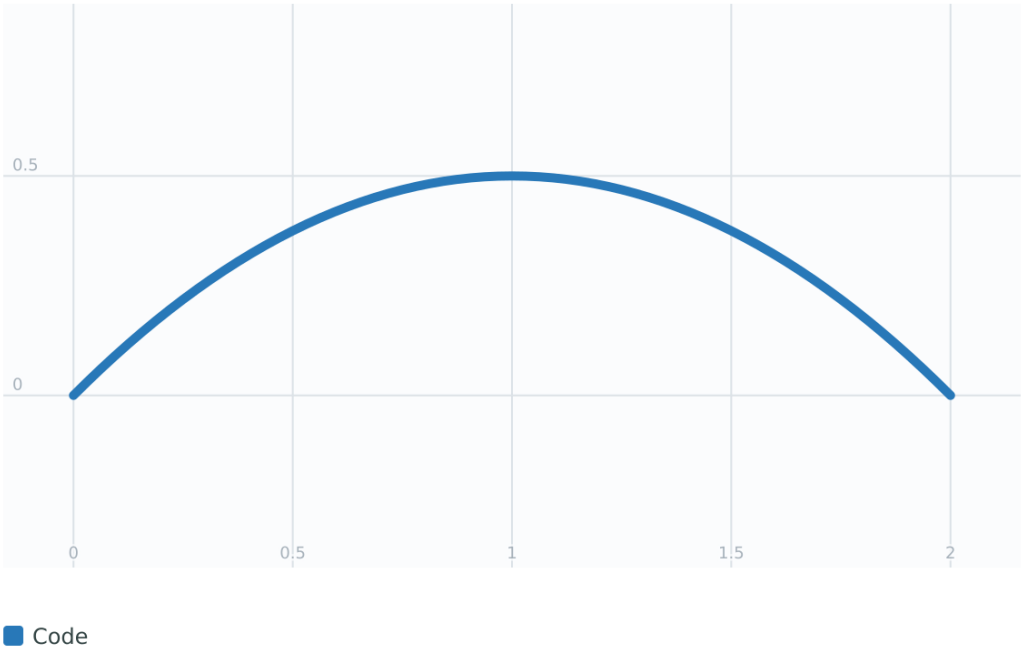
Code

```
SELECT ST_EndM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry);
```

Output

```
300
```

Figure



Geometry without M dimension returns NULL

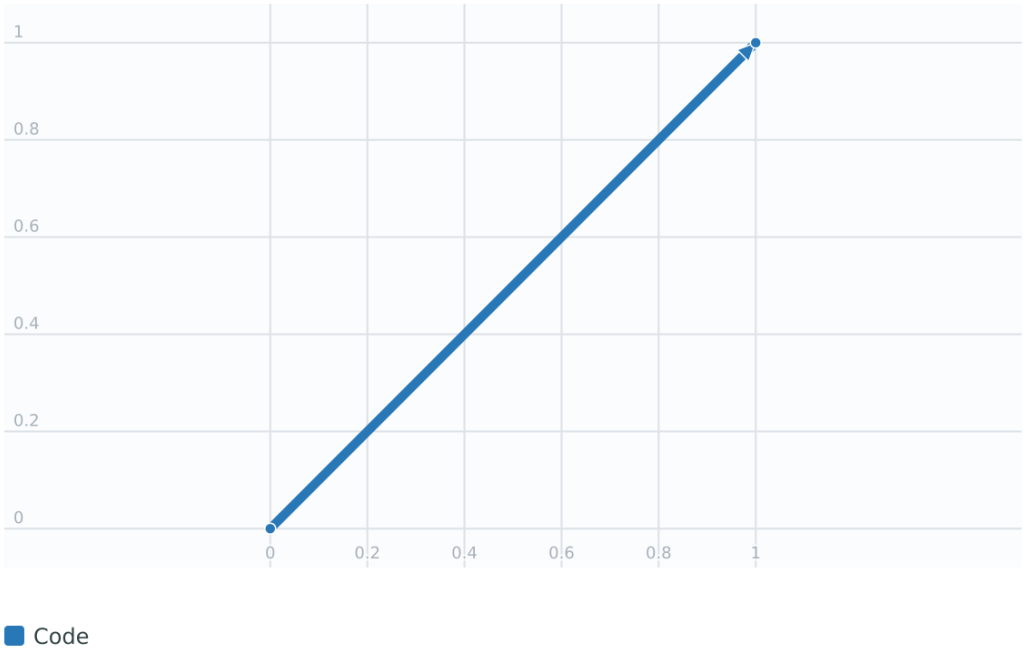
Code

```
SELECT ST_EndM('LINESTRING(0 0,1 1)'::geometry) IS NULL AS is_null;
```

Output

```
t
```

Figure



相关信息

ST_StartM, ST_SetEndM, ST_EndPoint

7.4.43 ST_SetStartM

ST_SetStartM — Sets the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_SetStartM**(geometry geom, float8 m);

描述

Returns a copy of the input LINESTRING, CIRCULARLINESTRING, or NURBSCURVE geometry with the M coordinate of the first point set to the specified value. If the input geometry does not have an M dimension, it will be added. Returns NULL if the input is not a LINESTRING, CIRCULARLINESTRING, or NURBSCURVE.

此方法支持☒形字符串和曲☒。

示例

Set start M value of a LineString

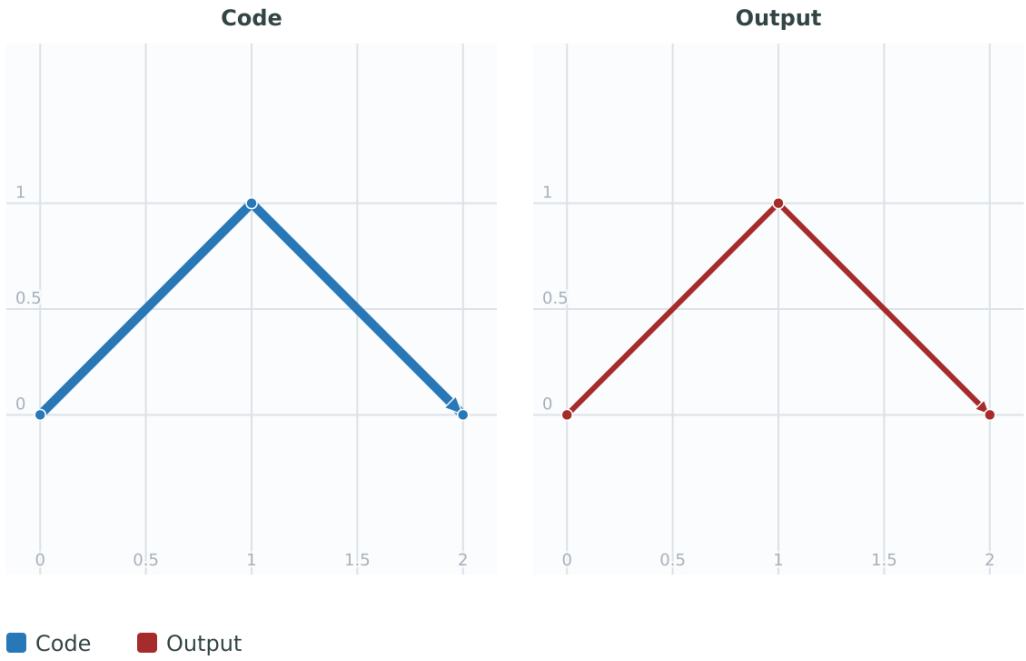
Code

```
SELECT ST_SetStartM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry, 100);
```

Output

```
LINESTRING M (0 0 100,1 1 20,2 0 30)
```

Figure



Add M dimension and set start M value

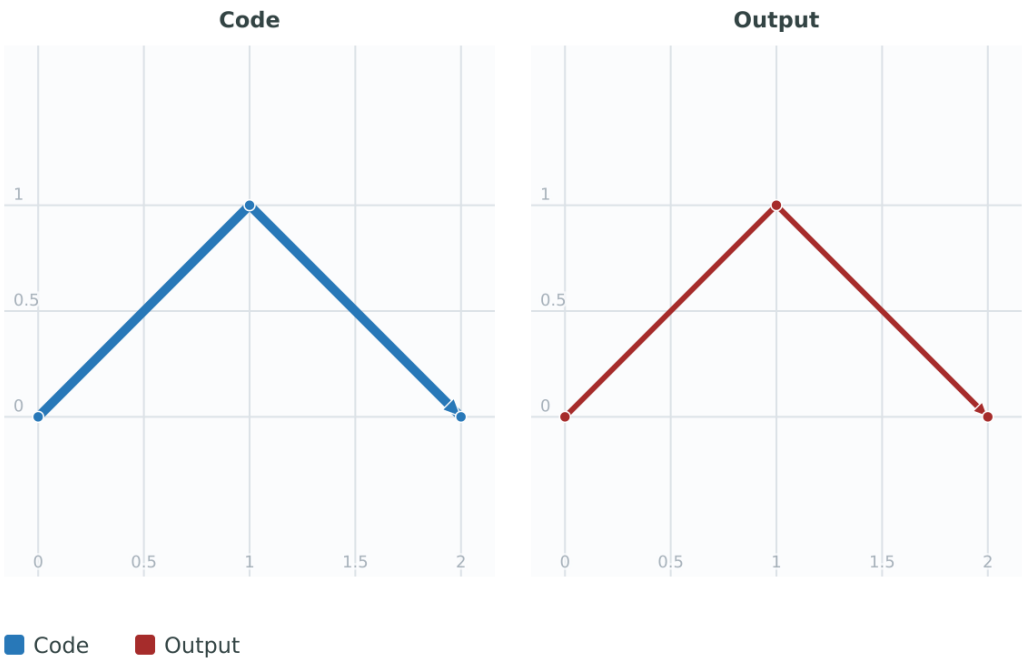
Code

```
SELECT ST_SetStartM('LINESTRING(0 0,1 1,2 0)'::geometry, 50);
```

Output

```
LINESTRING M (0 0 50,1 1 0,2 0 0)
```

Figure



Set start M value of a CircularString

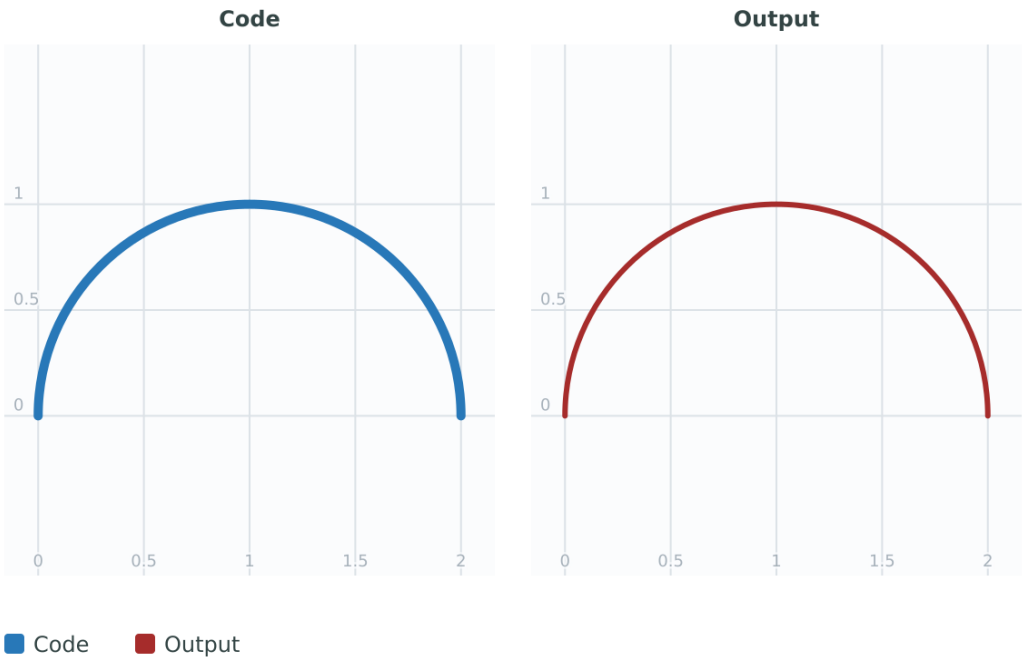
Code

```
SELECT ST_SetStartM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry, 99);
```

Output

```
CIRCULARSTRING M (0 0 99,1 1 10,2 0 15)
```

Figure



Set start M value of a NURBSCurve

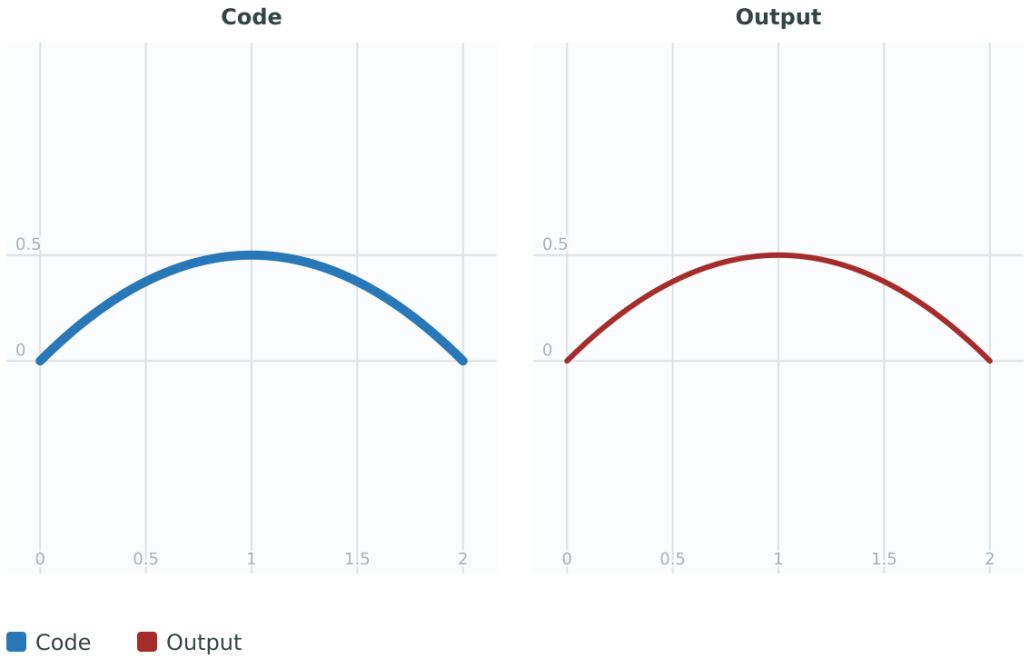
Code

```
SELECT ST_SetStartM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry, 500);
```

Output

```
NURBSCURVE M (DEGREE 2,CONTROLPOINTS M (NURBSPOINT(WEIGHTEDPOINT M (0 0 500),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (1 ↩
    1 200),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (2 0 300),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



相关信息

ST_SetEndM, ST_StartM, ST_StartPoint

7.4.44 ST_SetEndM

ST_SetEndM — Sets the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.

Synopsis

geometry **ST_SetEndM**(geometry geom, float8 m);

描述

Returns a copy of the input LINESTRING, CIRCULARLINESTRING, or NURBSCURVE geometry with the M coordinate of the last point set to the specified value. If the input geometry does not have an M dimension, it will be added. Returns NULL if the input is not a LINESTRING, CIRCULARLINESTRING, or NURBSCURVE.

此方法支持☒形字符串和曲☒。

示例

Set end M value of a LineString

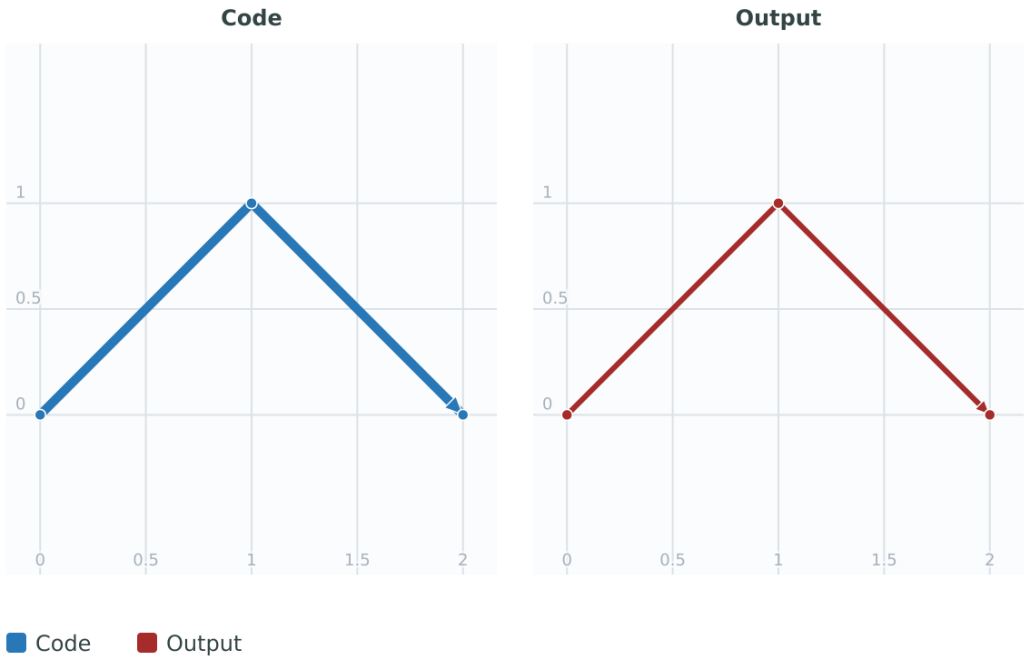
Code

```
SELECT ST_SetEndM('LINESTRING M(0 0 10,1 1 20,2 0 30)'::geometry, 999);
```

Output

```
LINESTRING M (0 0 10,1 1 20,2 0 999)
```

Figure



Add M dimension and set end M value

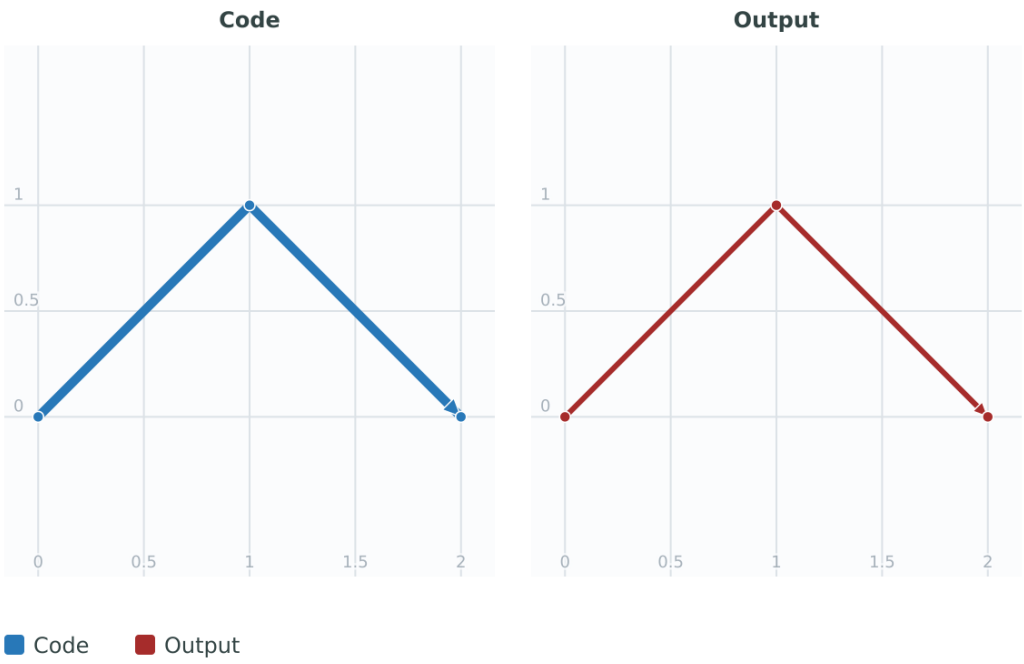
Code

```
SELECT ST_SetEndM('LINESTRING(0 0,1 1,2 0)'::geometry, 75);
```

Output

```
LINESTRING M (0 0 0,1 1 0,2 0 75)
```

Figure



Set end M value of a CircularString

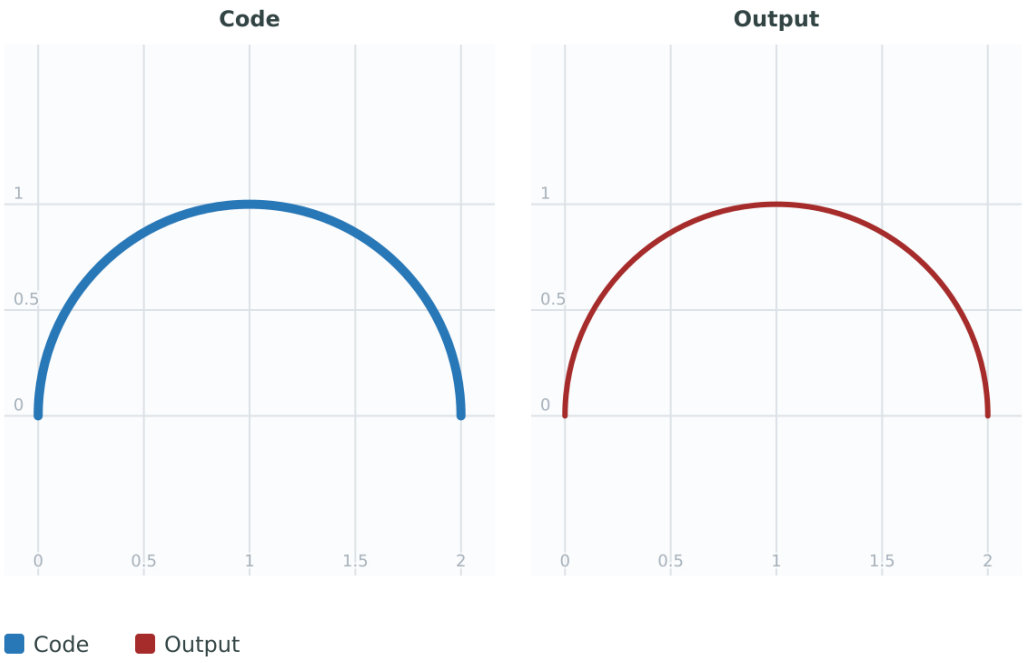
Code

```
SELECT ST_SetEndM('CIRCULARSTRING M(0 0 5,1 1 10,2 0 15)'::geometry, 88);
```

Output

```
CIRCULARSTRING M (0 0 5,1 1 10,2 0 88)
```

Figure



Set end M value of a NURBSCurve

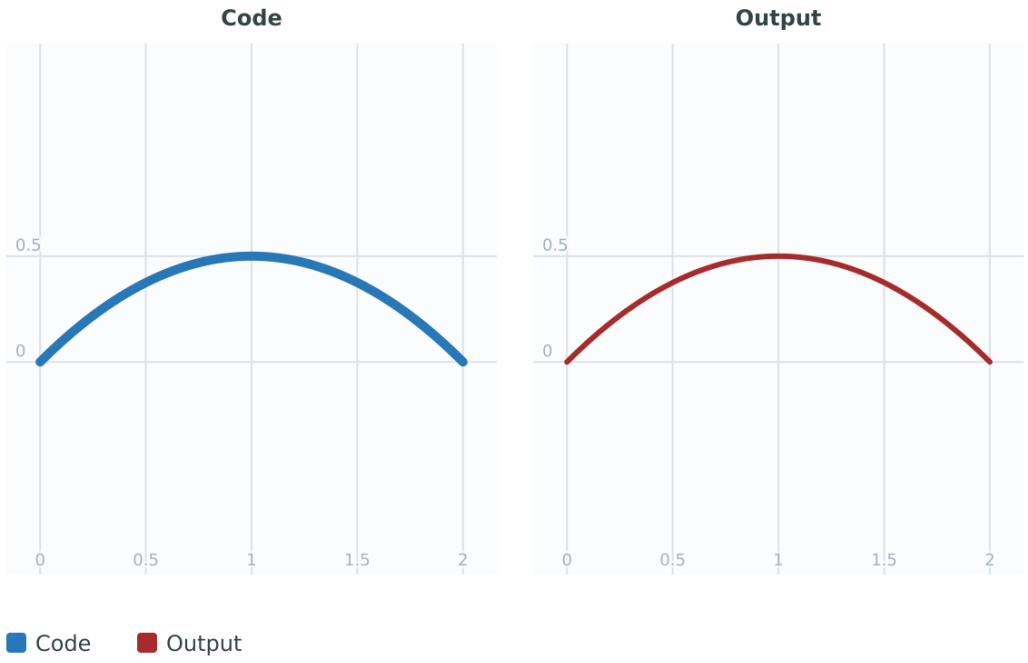
Code

```
SELECT ST_SetEndM('NURBSCURVE M(2, (0 0 100, 1 1 200, 2 0 300))'::geometry, 999);
```

Output

```
NURBSCURVE M (DEGREE 2,CONTROLPOINTS M (NURBSPOINT(WEIGHTEDPOINT M (0 0 100),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (1 ↩
    1 200),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT M (2 0 999),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3)))
```

Figure



相关信息

ST_SetStartM, ST_EndM, ST_EndPoint

7.4.45 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

Code

```
SELECT ST_Summary('LINESTRING(0 0,1 1)'::geometry) as geom,
ST_Summary('POLYGON((0 0,1 1,1 2,1 1,0 0))'::geography) geog;
```

Output

```
geom             |          geog
------------------------------+--------------------------
 LineString[B] with 2 points | Polygon[BGS] with 1 rings
                              | ring 0 has 5 points
                              :
(1 row)
```

Code

```
SELECT ST_Summary('LINESTRING(0 0 1,1 1 1)'::geography) As geog_line,
ST_Summary('SRID=4326;POLYGON((0 0 1,1 1 2,1 2 3,1 1 1,0 0 1))'::geometry) As geom_poly;
```

Output

```
geog_line              |        geom_poly
------------------------------- +-------------------------
 LineString[ZBGS] with 2 points | Polygon[ZBS] with 1 rings
                                :    ring 0 has 5 points
                                :
(1 row)
```

相关信息

PostGIS_DropBBox, PostGIS_AddBBox, ST_Force3DM, ST_Force3DZ, ST_Force2D, geography
ST_IsValid, ST_IsValid, ST_IsValidReason, ST_IsValidDetail

7.4.46 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

Code

```
SELECT ST_X('POINT(1 2 3 4)');
```

Output

```
1
```

Code

```
SELECT ST_Y(ST_Centroid('LINESTRING(1 2 3 4,1 1 1 1)'::geometry));
```

Output

```
1.5
```

相关信息

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_XMax, ST_XMin, ST_Y, ST_Z

7.4.47 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

Code

```
SELECT ST_Y('POINT(1 2 3 4)');
```

Output

```
2
```

Code

```
SELECT ST_Y(ST_Centroid('LINESTRING(1 2 3 4,1 1 1 1)'::geometry));
```

Output

```
1.5
```

相关信息

ST_Centroid, ST_GeomFromEWKT, ST_M, ST_X, ST_YMax, ST_YMin, ST_Z

7.4.48 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

Code

```
SELECT ST_Z('POINT(1 2 3 4)');
```

Output

```
3
```

相关信息

ST_GeomFromEWKT, ST_M, ST_X, ST_Y, ST_ZMax, ST_ZMin

7.4.49 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

Code

```
SELECT ST_ZMFlag('LINESTRING(1 2,3 4)');
```

Output

```
0
```

Code

```
SELECT ST_ZMFlag('LINESTRINGM(1 2 3, 3 4 3)');
```

Output

```
1
```

Code

```
SELECT ST_ZMFlag('CIRCULARSTRING(1 2 3,3 4 3,5 6 3)');
```

Output

```
2
```

Code

```
SELECT ST_ZMFlag('POINT(1 2 3 4)');
```

Output

```
3
```

相关信息

ST_CoordDim, ST_NDims, ST_Dimension

7.4.50 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

Code

```
SELECT ST_HasZ('POINT(1 2 3)');
```

Output

```
t
```

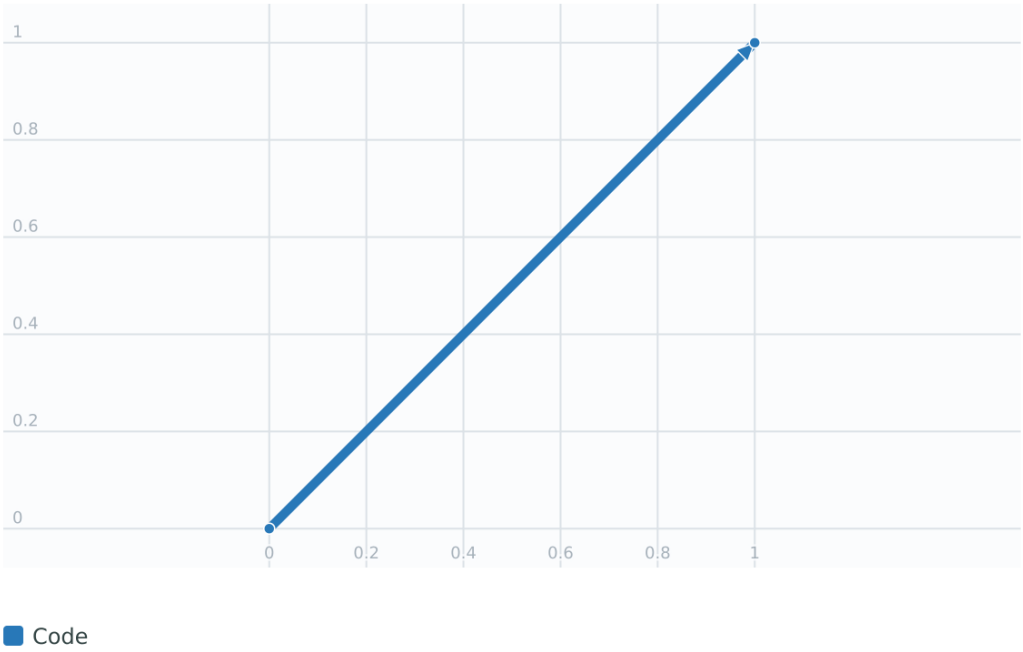
Code

```
SELECT ST_HasZ('LINESTRING(0 0,1 1)');
```

Output

```
f
```

Figure



相关信息

ST_ZMFlag
ST_HasM

7.4.51 ST_HasM

ST_HasM — 区区几何体是否具有 M（区量）区度。

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

Code

```
SELECT ST_HasM('POINTM(1 2 3)');
```

Output

```
t
```

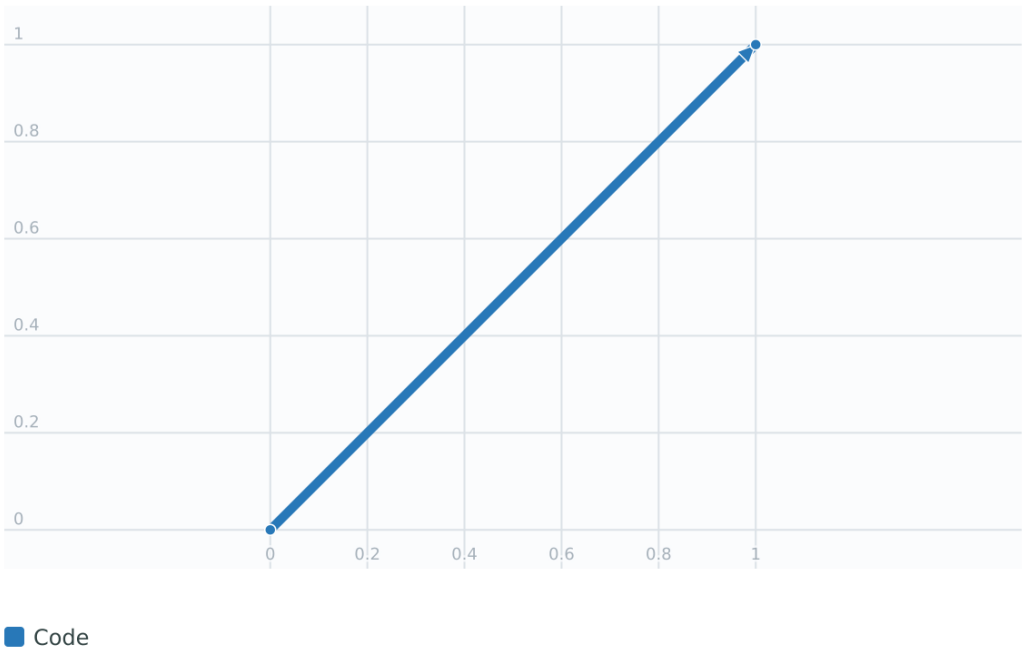
Code

```
SELECT ST_HasM('LINESTRING(0 0,1 1)');
```

Output

```
f
```

Figure



相关信息

ST_ZMFlag
ST_HasZ

7.4.52 ST_ControlPoints

ST_ControlPoints — Returns the control points of a NURBS curve as a MULTIPOINT geometry.

Synopsis

geometry **ST_ControlPoints**(geometry nurbscurve);

描述

Extracts the control points from a NURBS curve and returns them as a MULTIPOINT geometry. Control points define the control polygon that influences the shape of the NURBS curve.
The function raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.
Availability: 3.7.0

示例

This example extracts control points from a NURBS curve.

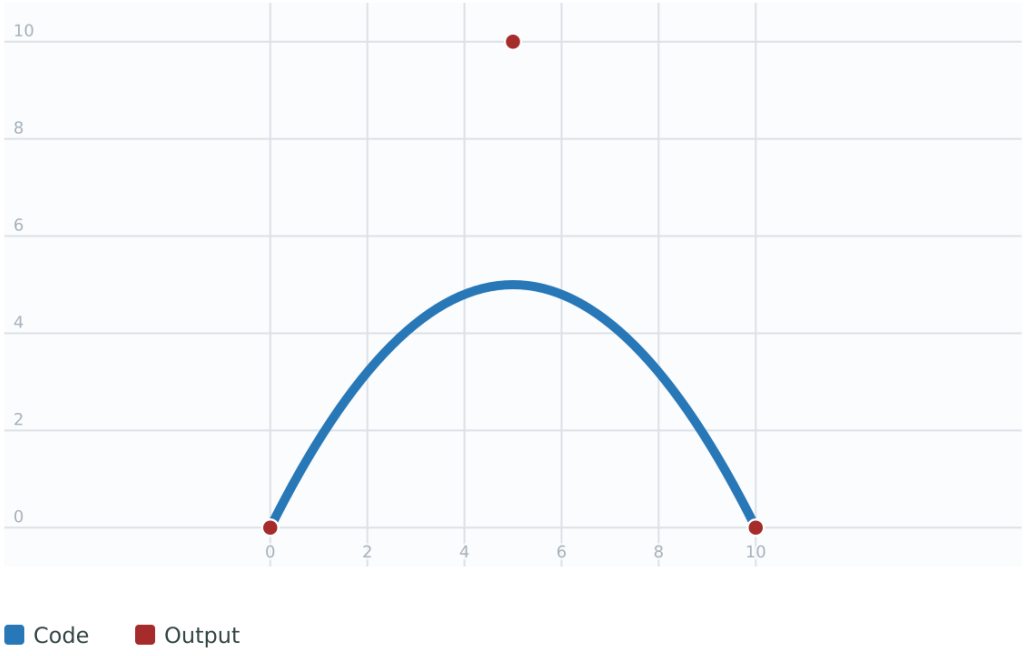
Code

```
SELECT ST_ControlPoints('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
MULTIPOINT((0 0),(5 10),(10 0))
```

Figure



This example counts the control points.

Code

```
SELECT ST_NPoints(ST_ControlPoints('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry));
```

Output

```
3
```

相关信息

ST_MakeNurbsCurve, ST_NumControlPoints, ST_Weights

7.4.53 ST_Degree

ST_Degree — Returns the polynomial degree of a NURBS curve.

Synopsis

integer **ST_Degree**(geometry nurbscurve);

描述

Returns the polynomial degree of a NURBS curve as an integer. The degree determines the curve's smoothness and flexibility. Higher degrees produce smoother curves but require more control points.

The function raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

示例

This example returns the degree of a NURBS curve.

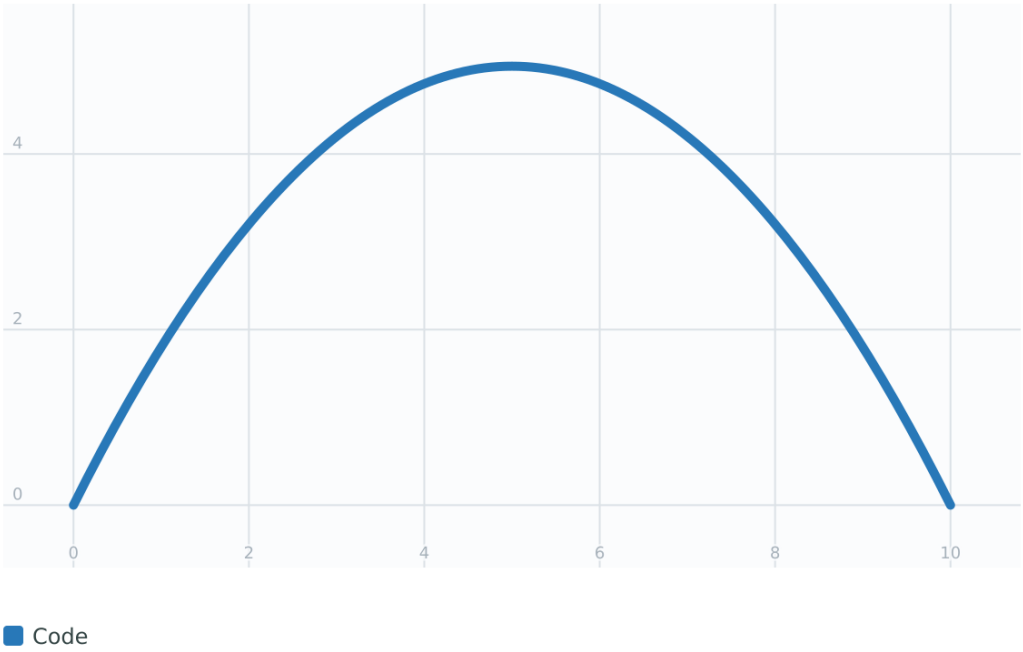
Code

```
SELECT ST_Degree('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
2
```

Figure



This example checks whether a NURBS curve is cubic.

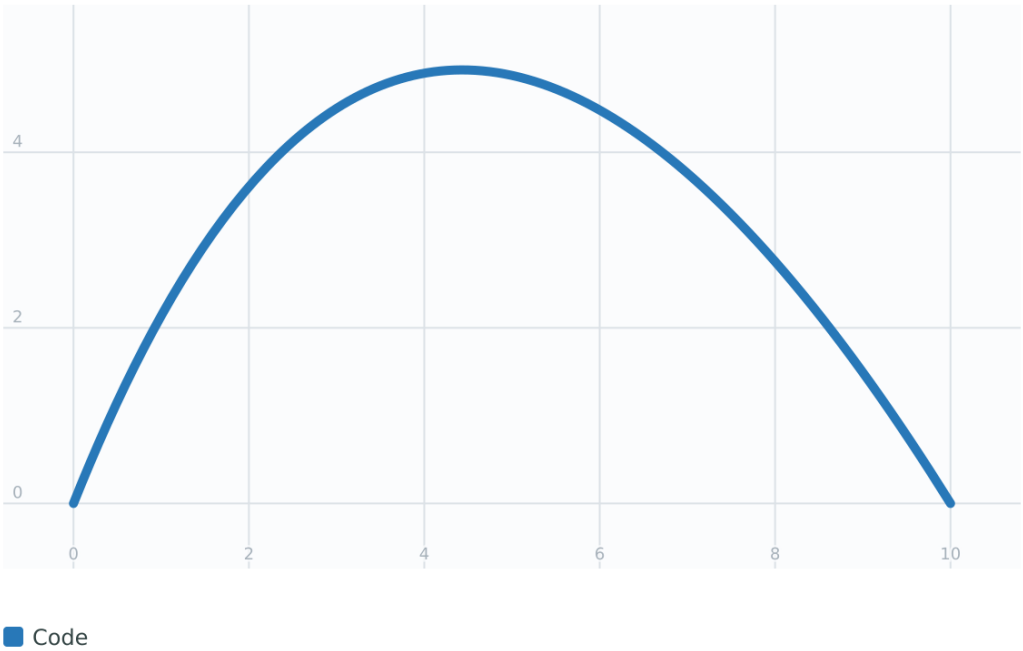
Code

```
SELECT ST_Degree('NURBSCURVE(3, (0 0, 2 5, 5 8, 10 0))'::geometry) = 3;
```

Output

```
t
```

Figure



相关信息

ST_MakeNurbsCurve, ST_NumControlPoints

7.4.54 ST_Weights

ST_Weights — Returns the weight array of a rational NURBS curve.

Synopsis

float8[] **ST_Weights**(geometry nurbscurve);

描述

Returns the weight values of a rational NURBS curve as an array of double precision values. Weights control the influence of each control point on the curve shape. Higher weights pull the curve closer to the corresponding control point.

Returns NULL if the curve has no weights (non-rational B-spline) or if the input is NULL. Raises an error if the input is not a NURBS curve.

Availability: 3.7.0

示例

This example returns weights from a rational NURBS curve.

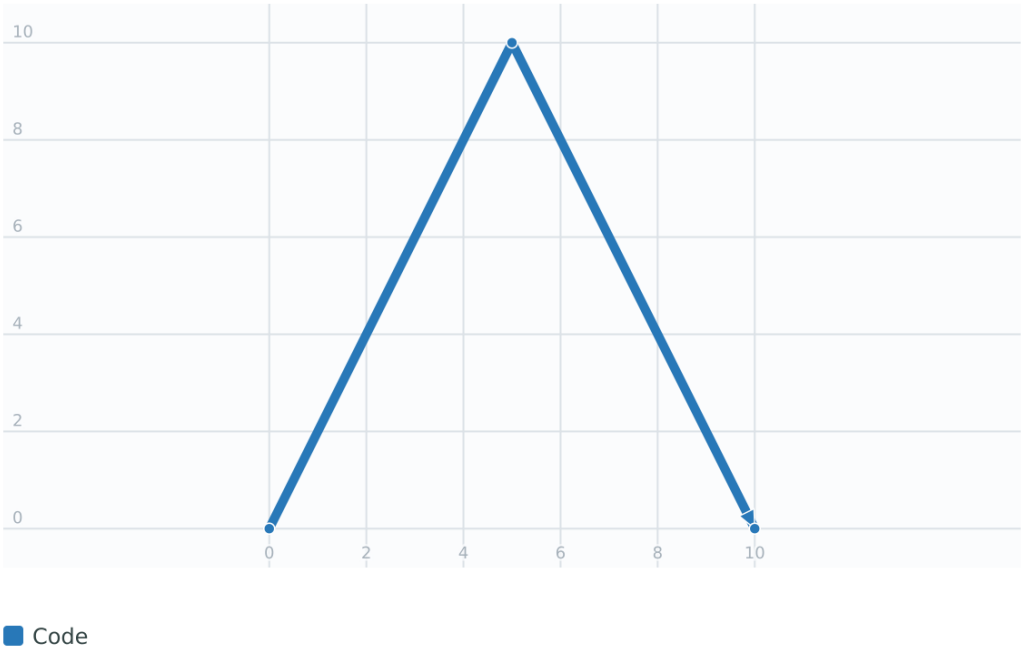
Code

```
SELECT ST_Weights(ST_MakeNurbsCurve(2, 'LINESTRING(0 0,5 10,10 0)'::geometry, ARRAY[1.0,  ↩
    2.0, 1.0]));
```

Output

```
{1,2,1}
```

Figure



This example checks the weights of a non-rational curve.

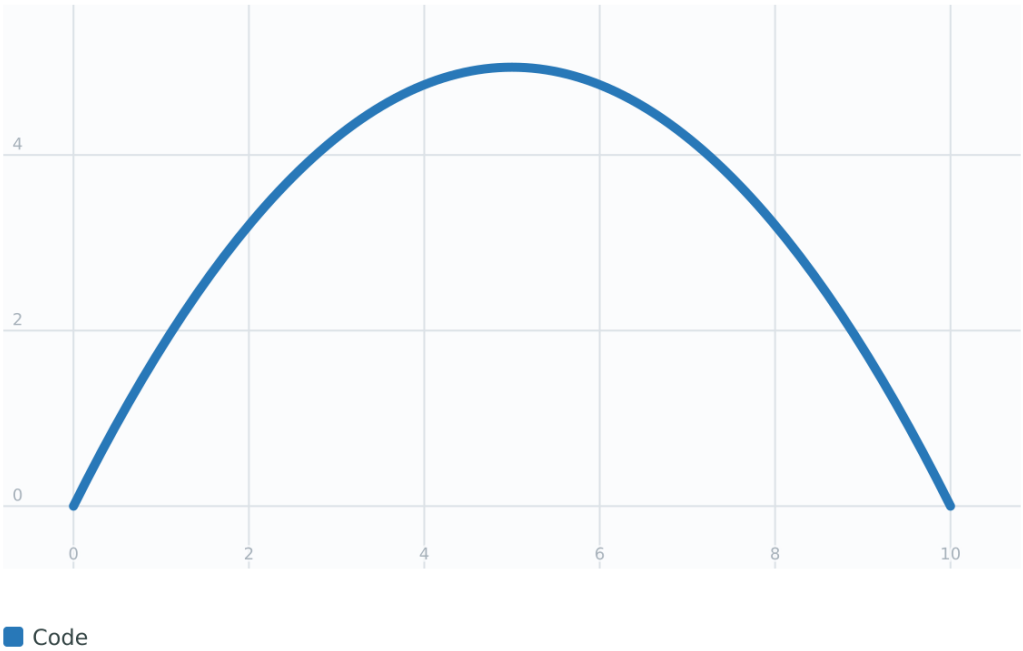
Code

```
SELECT ST_Weights('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
null
```

Figure



相关信息

ST_MakeNurbsCurve, ST_NurbsCurveIsRational, ST_ControlPoints

7.4.55 ST_Knots

ST_Knots — Returns the knot vector of a NURBS curve.

Synopsis

float8[] **ST_Knots**(geometry nurbscurve);

描述

Returns the knot vector of a NURBS curve as an array of double precision values. The knot vector defines the parameter space and controls where and how the control points affect the curve. It must be a non-decreasing sequence with length equal to (number of control points + degree + 1).

Returns NULL if the curve uses the default uniform clamped knot vector or if the input is NULL. Raises an error if the input is not a NURBS curve.

Availability: 3.7.0

示例

This example returns knots from a NURBS curve with a custom knot vector.

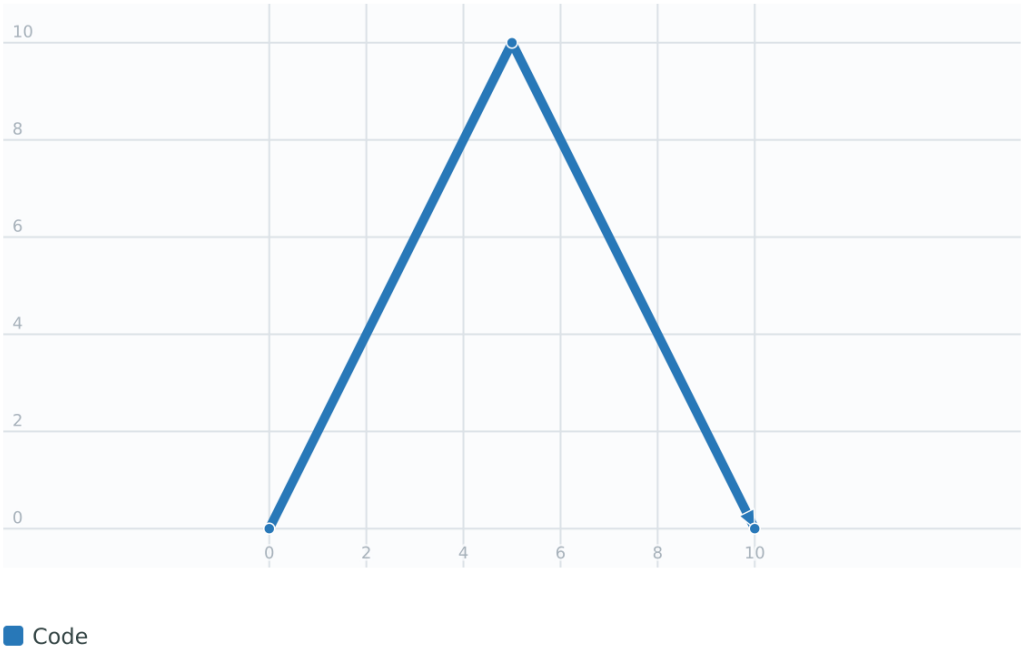
Code

```
SELECT ST_Knots(ST_MakeNurbsCurve(
        2,
        'LINESTRING(0 0,5 10,10 0)'::geometry,
        ARRAY[1.0, 2.0, 1.0],
        ARRAY[0, 0, 0, 1, 1, 1]));
```

Output

```
{0,0,0,1,1,1}
```

Figure



This example checks the knots of a curve using the default knot vector.

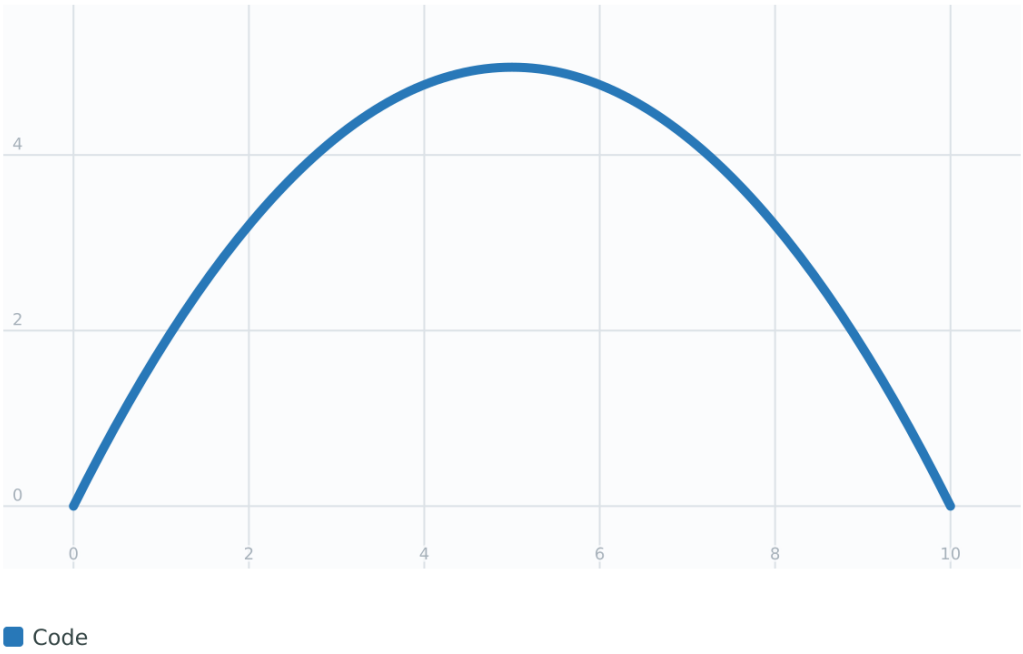
Code

```
SELECT ST_Knots('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
null
```

Figure



相关信息

ST_MakeNurbsCurve, ST_Degree, ST_NumControlPoints

7.4.56 ST_NumControlPoints

ST_NumControlPoints — Returns the number of control points in a NURBS curve.

Synopsis

integer **ST_NumControlPoints**(geometry nurbscurve);

描述

Returns the number of control points in a NURBS curve. The number of control points must be at least (degree + 1) for a valid NURBS curve.

Raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

示例

This example returns the number of control points.

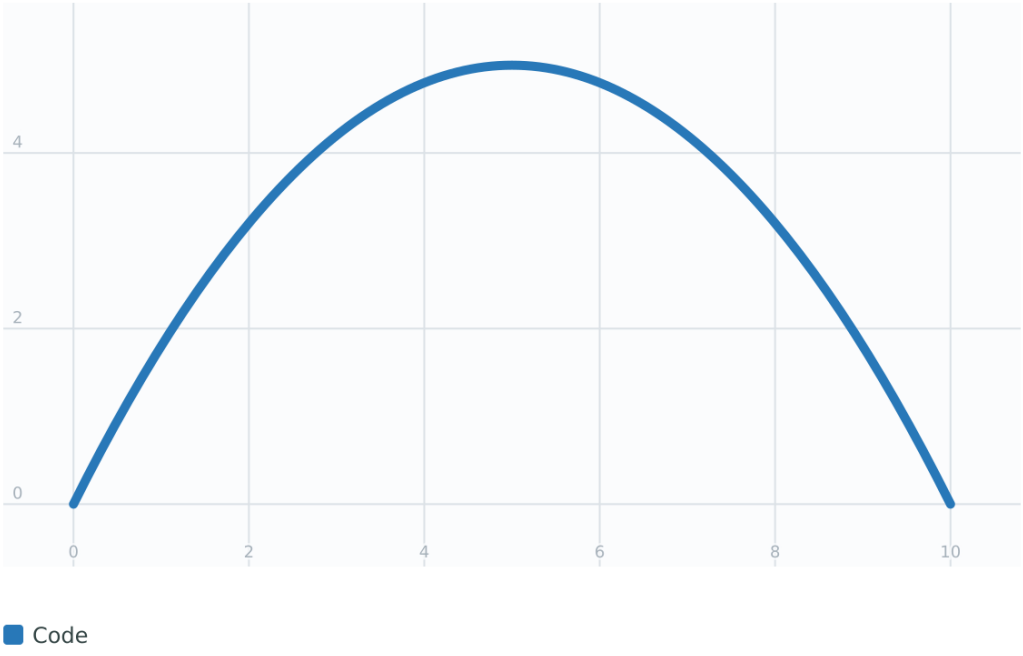
Code

```
SELECT ST_NumControlPoints('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
3
```

Figure



This example verifies that a curve has enough control points for its degree.

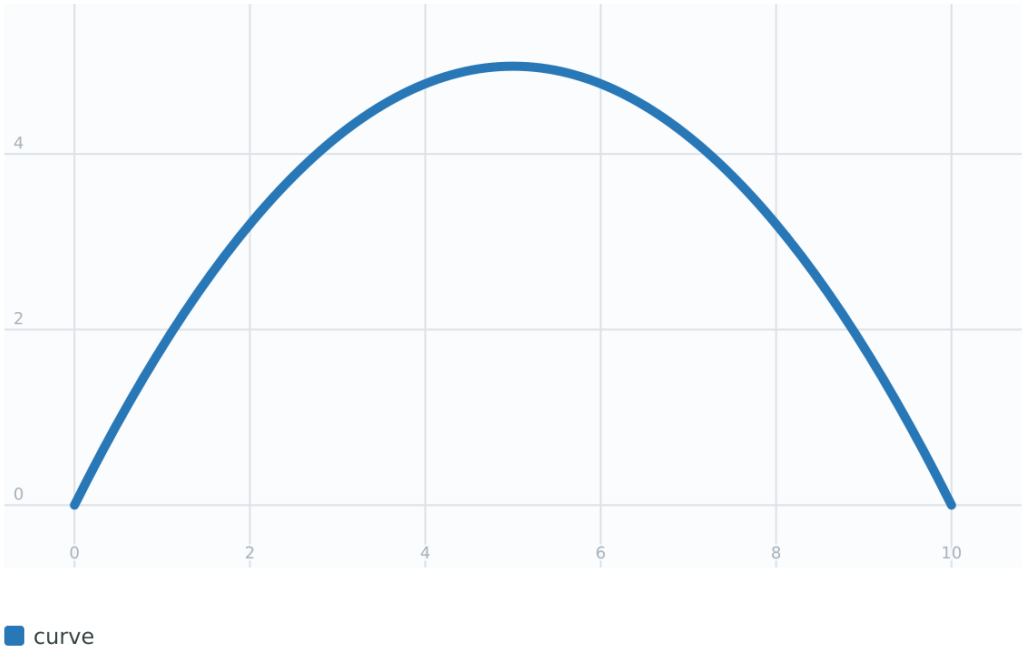
Code

```
SELECT ST_NumControlPoints(curve)
>= ST_Degree(curve) + 1 AS valid
FROM (SELECT 'NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry AS curve) AS t;
```

Output

```
t
```

Figure



相关信息

ST_MakeNurbsCurve, ST_ControlPoints, ST_Degree

7.4.57 ST_NurbsCurveIsRational

ST_NurbsCurveIsRational — Checks if a NURBS curve is rational (has weights).

Synopsis

boolean **ST_NurbsCurveIsRational**(geometry nurbscurve);

描述

Returns TRUE if the NURBS curve has weights assigned to its control points (rational NURBS), FALSE if it uses uniform weights (non-rational B-spline).

Rational NURBS curves provide more flexibility in curve shape control through variable weights. Non-rational B-splines use uniform weights and are a special case of NURBS curves.

Raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

示例

This example checks whether a curve is rational.

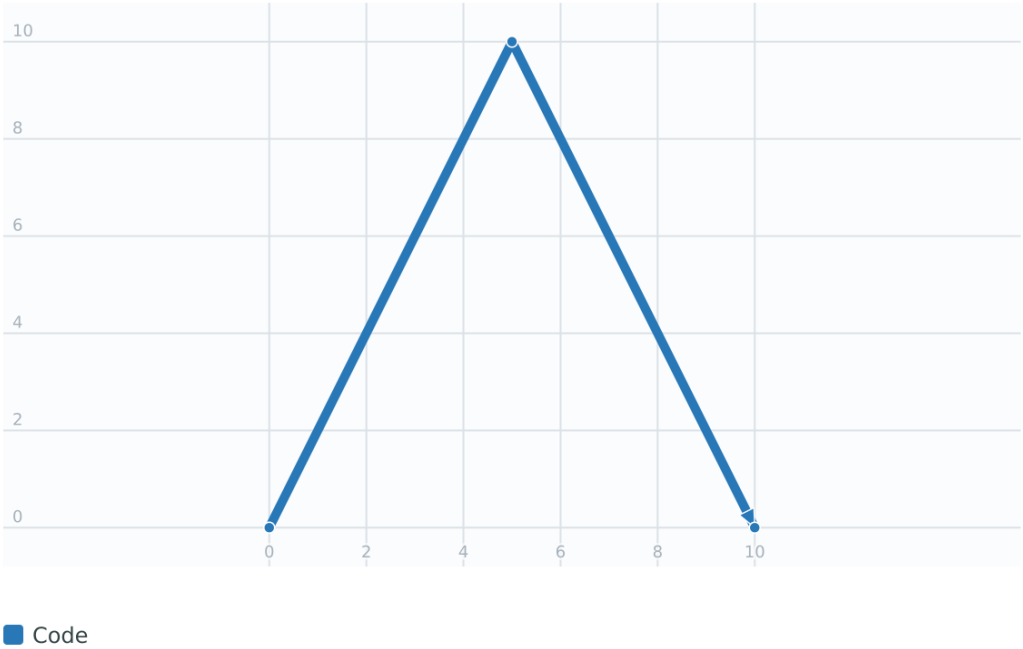
Code

```
SELECT ST_NurbsCurveIsRational(ST_MakeNurbsCurve(
        2,
        'LINESTRING(0 0,5 10,10 0)'::geometry,
        ARRAY[1.0, 2.0, 1.0]));
```

Output

```
t
```

Figure



This example checks a non-rational curve.

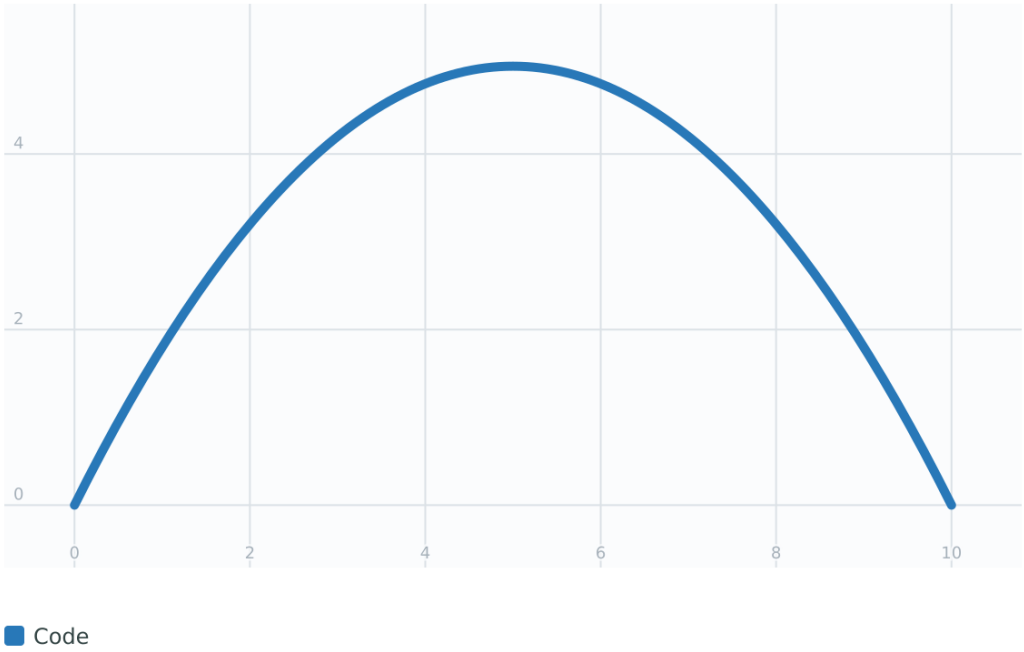
Code

```
SELECT ST_NurbsCurveIsRational('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
f
```

Figure



相关信息

ST_MakeNurbsCurve, ST_Weights, ST_IsNurbsCurve

7.4.58 ST_IsNurbsCurve

ST_IsNurbsCurve — Checks if a geometry is a NURBS curve.

Synopsis

boolean **ST_IsNurbsCurve**(geometry geom);

描述

Returns TRUE if the geometry is a NURBS curve, FALSE otherwise. This function can be used to test geometry types before applying NURBS-specific functions.

Returns NULL if the input is NULL.

Availability: 3.7.0

示例

This example checks whether a geometry is a NURBS curve.

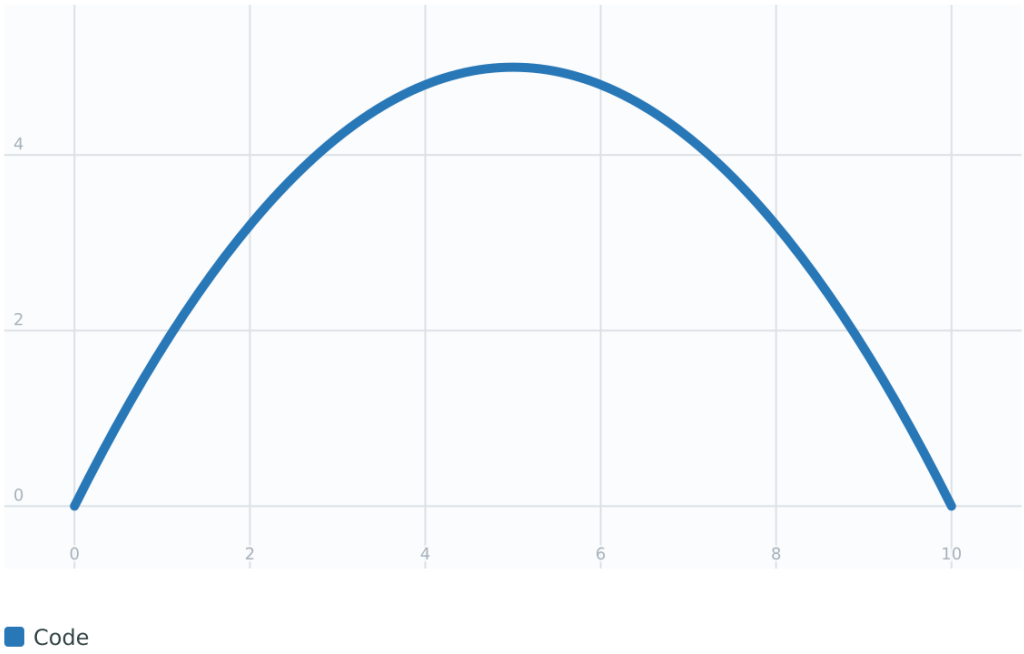
Code

```
SELECT ST_IsNurbsCurve('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
t
```

Figure



This example tests a regular LineString.

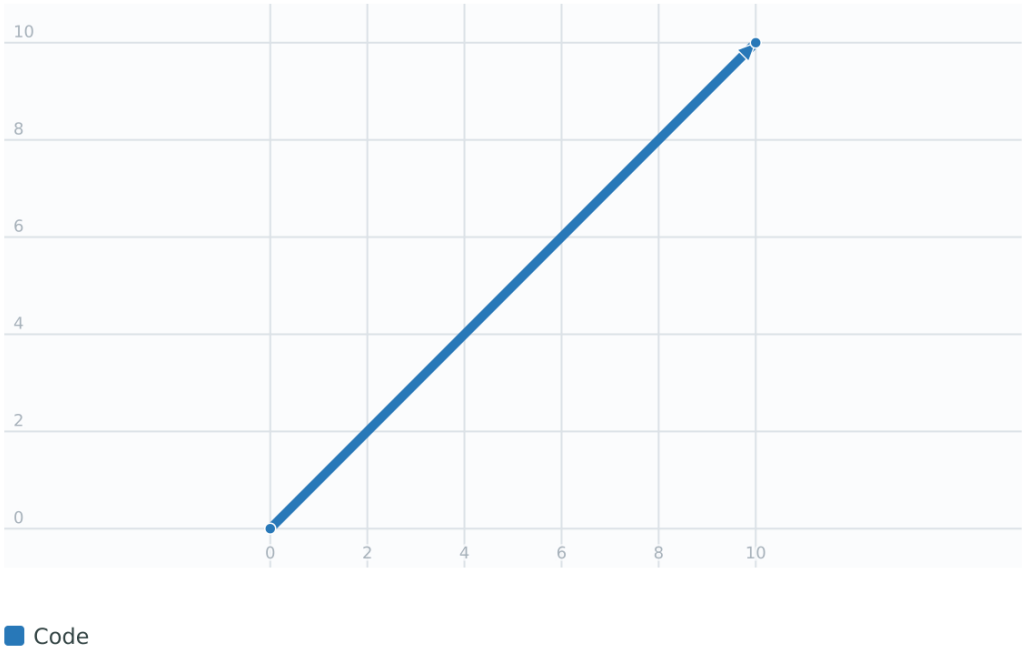
Code

```
SELECT ST_IsNurbsCurve('LINESTRING(0 0,10 10)'::geometry);
```

Output

```
f
```

Figure



相关信息

ST_MakeNurbsCurve, ST_GeometryType, ST_NurbsCurveIsRational

7.4.59 ST_Evaluate

ST_Evaluate — Evaluates a NURBS curve at a specific parameter value and returns the resulting point.

Synopsis

geometry **ST_Evaluate**(geometry nurbscurve, float8 parameter);

描述

Evaluates a NURBS curve at the specified parameter value and returns a POINT geometry representing the position on the curve. The parameter typically ranges from 0.0 (start of curve) to 1.0 (end of curve), though the actual range depends on the knot vector.

Raises an error if the input is not a NURBS curve or if evaluation fails. Returns NULL if either input is NULL.

Availability: 3.7.0

示例

This example returns the midpoint of a NURBS curve.

Code

```
SELECT ST_Evaluate('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry, 0.5);
```

This example samples multiple points along a curve.

Code

```
SELECT t, ST_Evaluate(curve, t) AS point
FROM (SELECT 'NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry AS curve) AS c,
         generate_series(0, 1, 0.25) AS t;
```

相关信息

ST_MakeNurbsCurve, ST_NurbsToLineString

7.4.60 ST_NurbsToLineString

ST_NurbsToLineString — Converts a NURBS curve to a LINESTRING by uniform sampling.

Synopsis

geometry **ST_NurbsToLineString**(geometry nurbscurve, integer num_segments=32);

描述

Converts a NURBS curve to a LINESTRING geometry by creating a piecewise linear approximation. The curve is sampled at uniformly distributed parameter values and the resulting points are connected with straight line segments.

The optional `num_segments` parameter specifies the number of line segments in the output (default: 32, range: 2-10000). Higher values produce smoother approximations but increase geometry complexity.

Raises an error if the input is not a NURBS curve. Returns NULL if the input is NULL.

Availability: 3.7.0

示例

This example converts a NURBS curve to a LineString using the default quality.

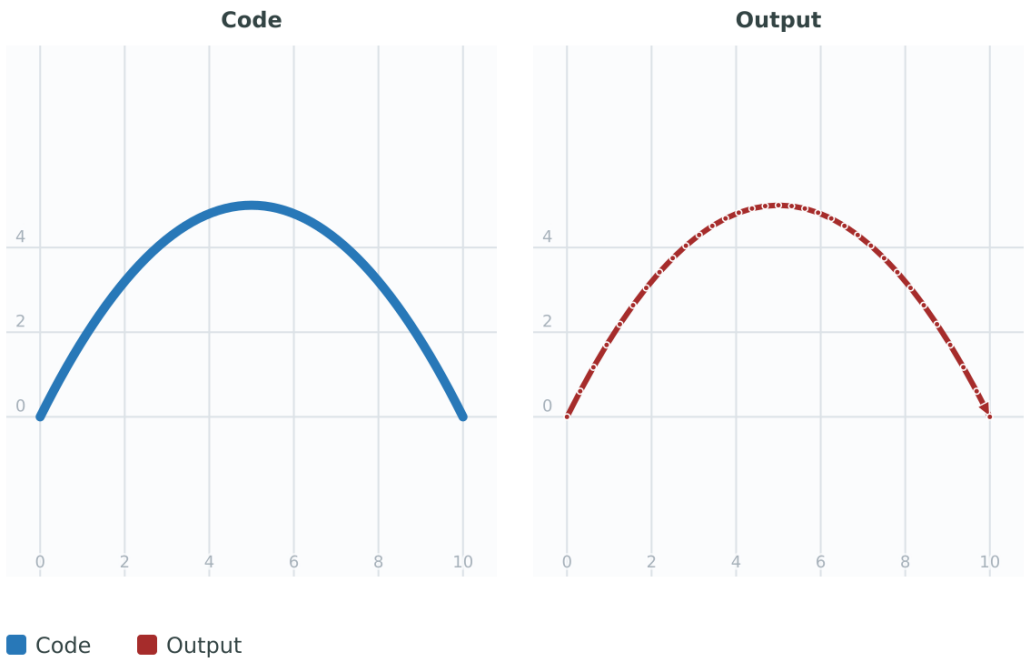
Code

```
SELECT ST_NurbsToLineString('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry);
```

Output

```
LINESTRING(0 0,0.3125 0.60546875,0.625 1.171875,0.9375 1.69921875,1.25 2.1875,1.5625 2.63671875,1.875 3.046875,2.1875 ↩
    3.41796875,2.5 3.75,2.8125 4.04296875,3.125 4.296875,3.4375 4.51171875,3.75 4.6875,4.0625 4.82421875,4.375 ↩
    4.921875,4.6875 4.98046875,5 5,5.3125 4.98046875,5.625 4.921875,5.9375 4.82421875,6.25 4.6875,6.5625 ↩
    4.51171875,6.875 4.296875,7.1875 4.04296875,7.5 3.75,7.8125 3.41796875,8.125 3.046875,8.4375 2.63671875,8.75 ↩
    2.1875,9.0625 1.69921875,9.375 1.171875,9.6875 0.60546875,10 0)
```

Figure

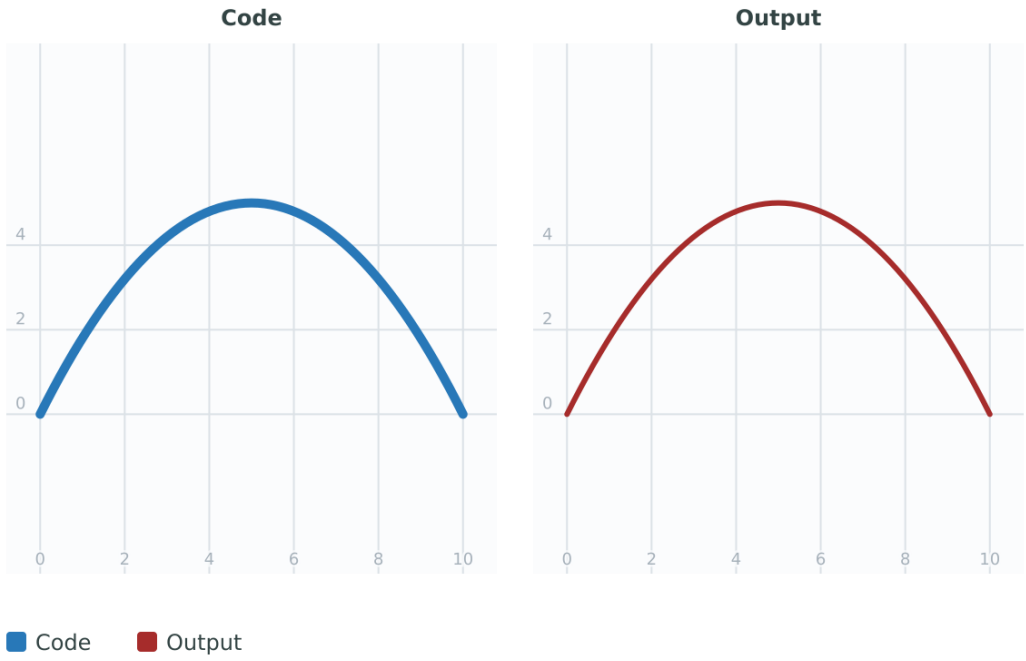


This example creates a high-quality approximation with 64 segments.

Code

```
SELECT ST_NurbsToLineString('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry, 64);
```

Figure



This example creates a lower-quality, faster approximation with 16 segments.

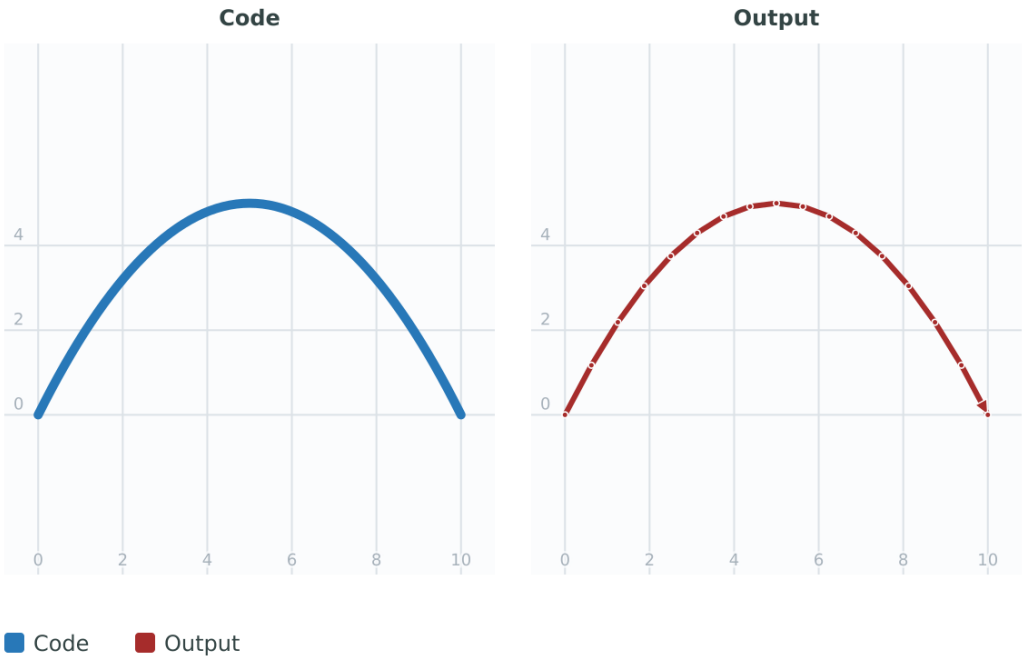
Code

```
SELECT ST_NurbsToLineString('NURBSCURVE(2, (0 0, 5 10, 10 0))'::geometry, 16);
```

Output

```
LINESTRING(0 0,0.625 1.171875,1.25 2.1875,1.875 3.046875,2.5 3.75,3.125 4.296875,3.75 4.6875,4.375 4.921875,5 5,5.625 ↩
    4.921875,6.25 4.6875,6.875 4.296875,7.5 3.75,8.125 3.046875,8.75 2.1875,9.375 1.171875,10 0)
```

Figure



相关信息

ST_MakeNurbsCurve, ST_Evaluate, ST_Segmentize

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

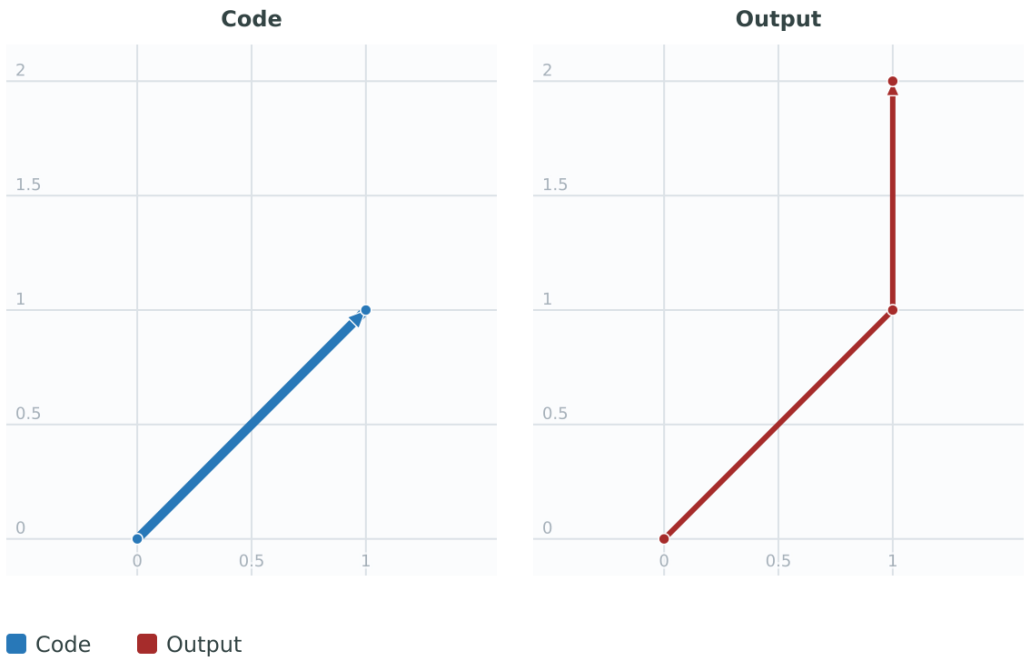
Code

```
SELECT ST_AddPoint('LINESTRING(0 0 1,1 1 1)', ST_MakePoint(1, 2, 3));
```

Output

```
LINESTRING(0 0 1,1 1 1,1 2 3)
```

Figure



□□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。

Code

```
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 — 给定一个几何集合，返回给包含指定给型元素的多几何给形。

Synopsis

geometry **ST_CollectionExtract**(geometry collection);

geometry **ST_CollectionExtract**(geometry collection, integer type);

描述

给定一个几何集合，返回同种给型多几何。

如果未指定给型，给返回给包含最高给度几何的多几何。因此，多给形给先于给，给给先于点。

如果指定了给型，给返回给包含指定给型的多几何。如果没有指定给型的元素，给返回 EMPTY 几何给形。给支持点、给和面。给型给号如下：

- 1 == POINT
- 2 == LINESTRING
- 3 == POLYGON

给于基本几何给入，如果给入给型与给求的给型匹配，给几何给形将保持不给返回。否给，给果是指定给型的空几何给形。如果需要，可以使用 ST_Multi 将它给给给给多几何给形。



Warning
不给给 MultiPolygon 给果的有效性。如果多给形给件相给或重给，给给果将无效。（例如，将此函数给用于 ST_Split给果给可能会给生给种情况。）可以使用 ST_IsValid 给给给种情况并使用ST_MakeValid 修复。

可用性：1.1.0



Note
在 1.5.3 之前，无给给型如何，给函数都会返回同种给型的基本几何给入。在 1.5.3 中，不匹配的给个几何给形返回 NULL 给果。在 2.0.0 中，不匹配的给个几何给形返回给求给型的 EMPTY 给果。

示例

提取最大给度给型：

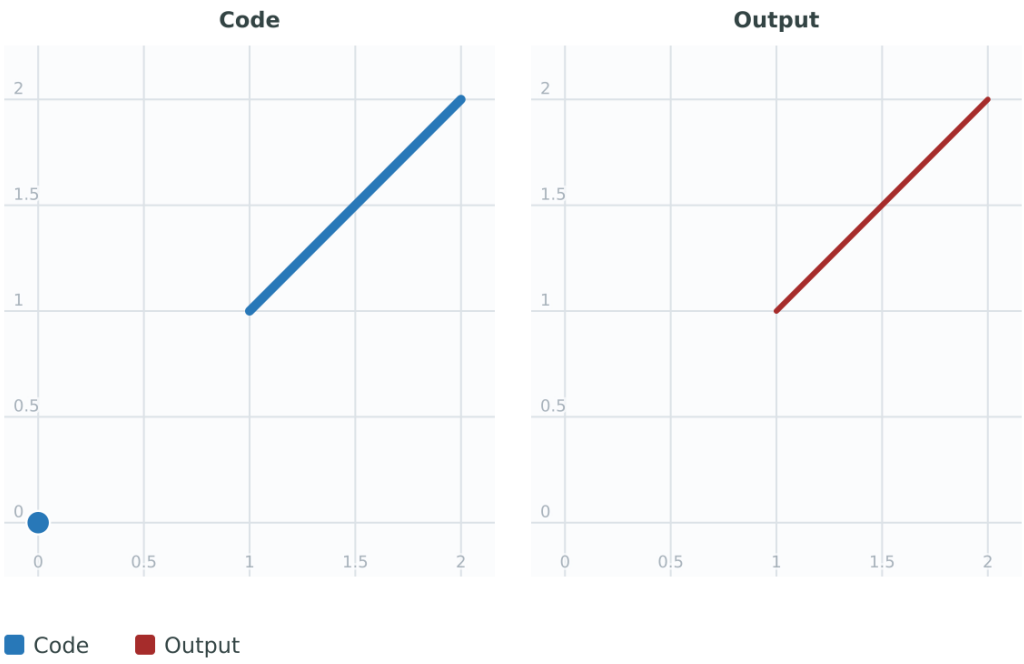
Code

```
SELECT ST_CollectionExtract('GEOMETRYCOLLECTION( POINT(0 0), LINESTRING(1 1,2 2) )');
```

Output

```
MULTILINESTRING((1 1, 2 2))
```

Figure



提取点（type 1 == POINT）：

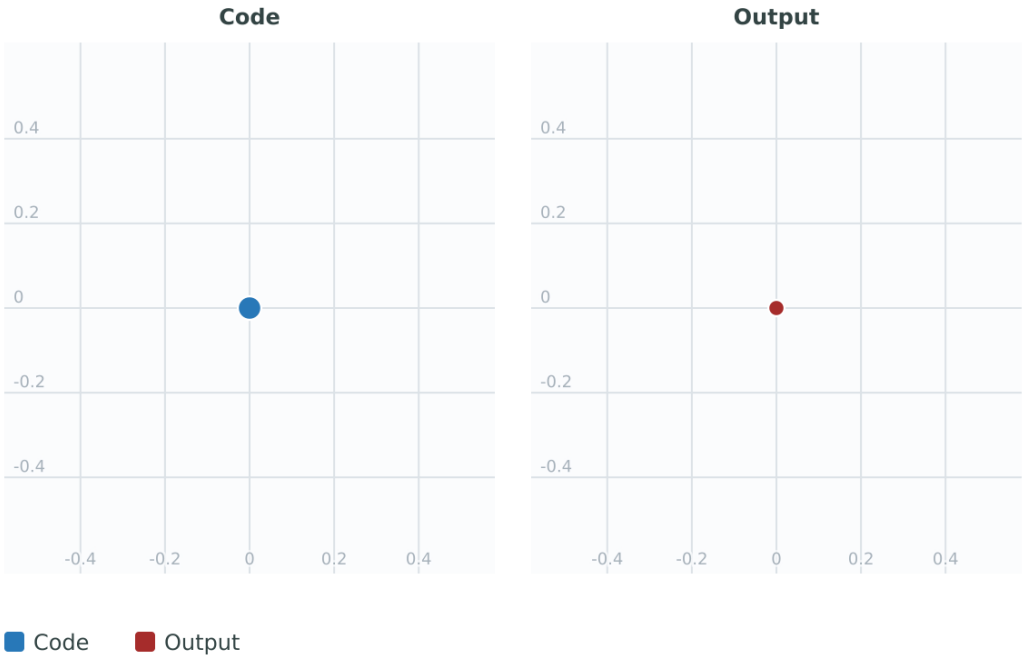
Code

```
SELECT ST_CollectionExtract('GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(POINT(0 0)))',
        1);
```

Output

```
MULTIPOINT((0 0))
```

Figure



提取☒（type 2 == LINESTRING）：

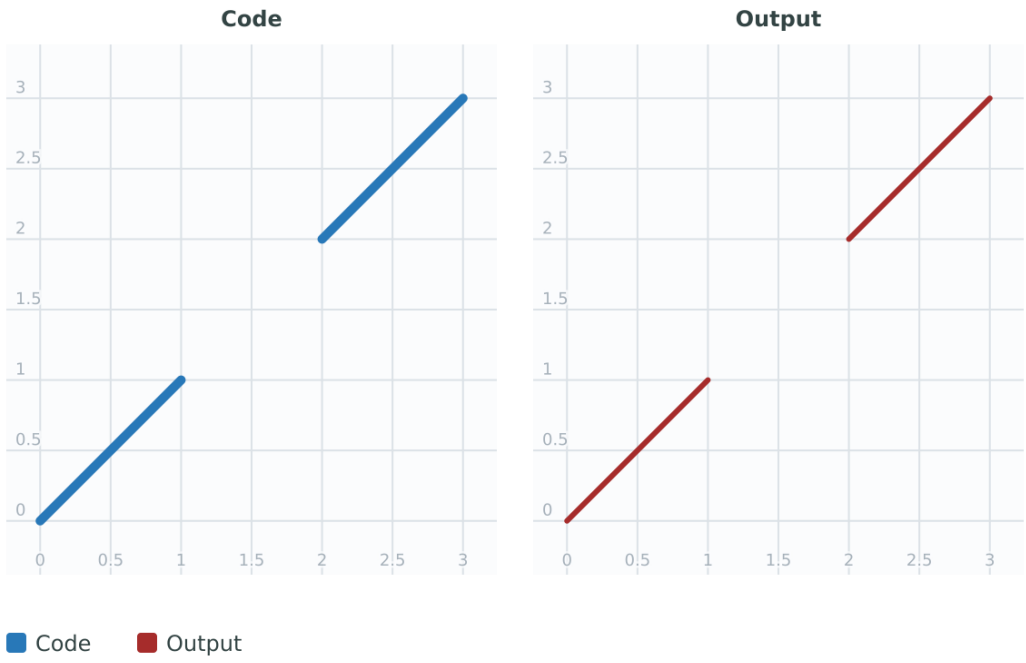
Code

```
SELECT ST_CollectionExtract('GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(LINESTRING(0 0,1 1)), ↩
    LINESTRING(2 2,3 3))',
        2);
```

Output

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

Figure



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

Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0))');
```

Output

```
POINT(0 0)
```

Figure



将嵌套的⍰元素集合⍰⍰⍰基本几何：

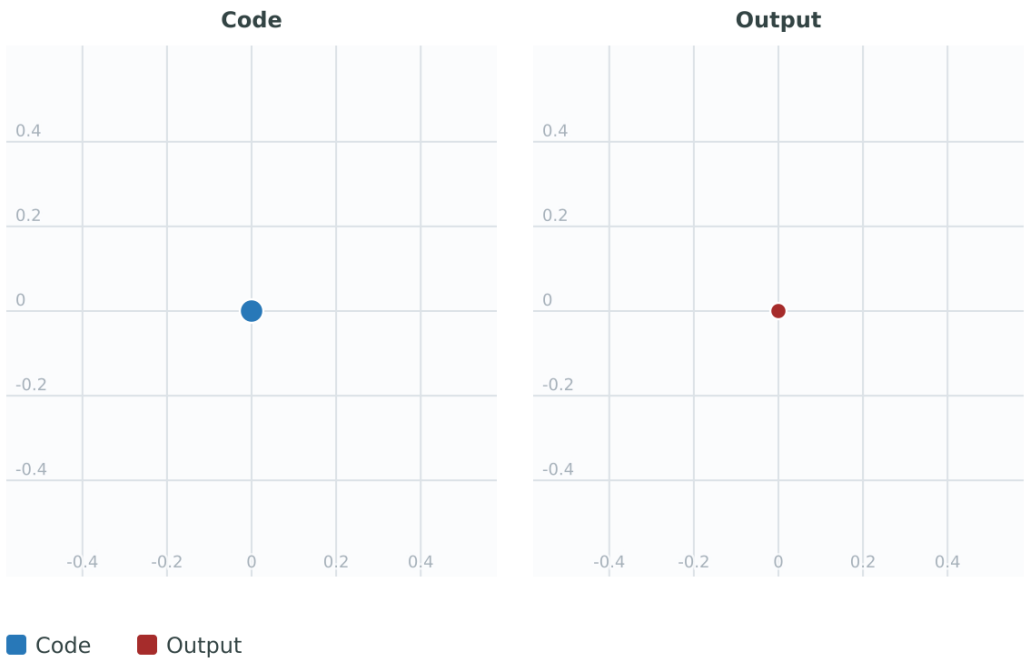
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(MULTIPOINT((0 0)))');
```

Output

```
POINT(0 0)
```

Figure



集合⊠⊠⊠多几何⊠形：

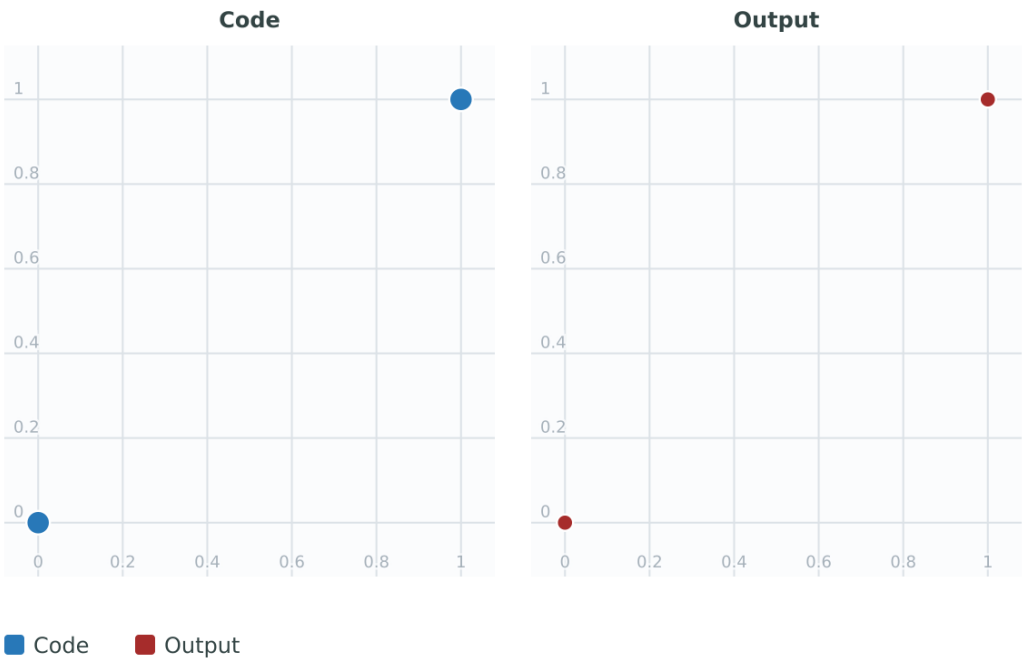
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0), POINT(1 1))');
```

Output

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

Figure



将嵌套混合⊠型集合⊠⊠⊠平面几何集合：

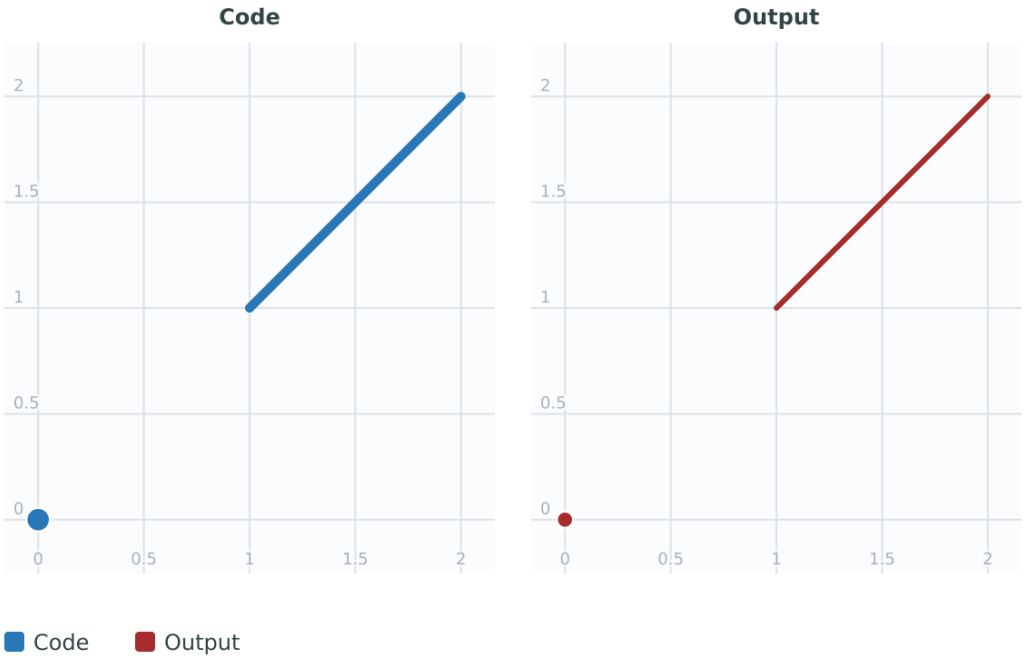
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0), GEOMETRYCOLLECTION(  ↩
    LINESTRING(1 1,2 2)))');
```

Output

```
GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(1 1,2 2))
```

Figure



将多⊠形集合⊠⊠⊠（无效的）多多⊠形：

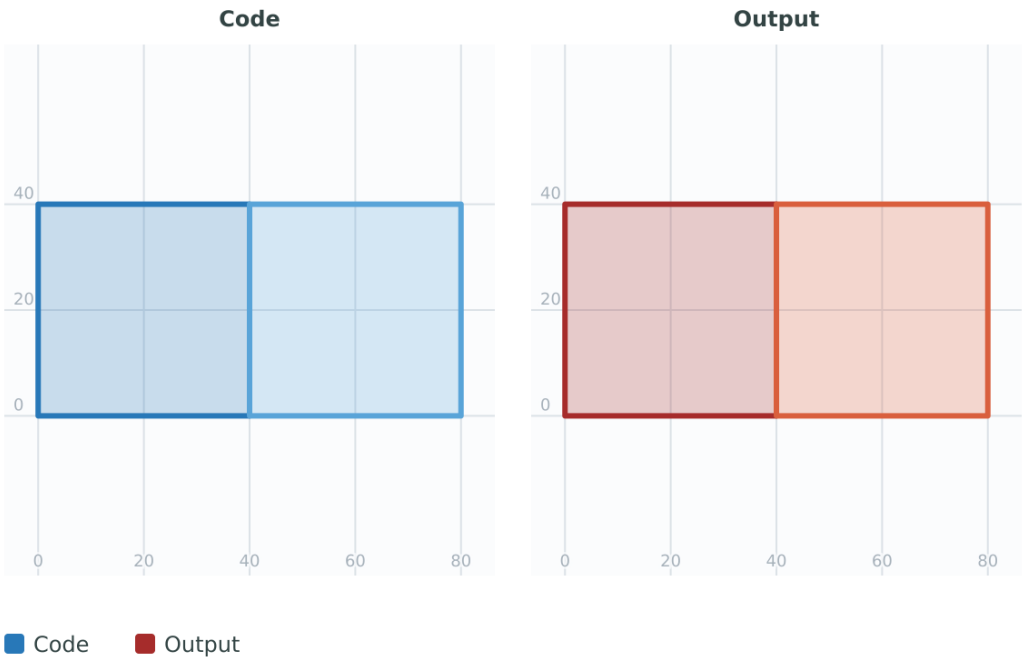
Code

```
SELECT ST_CollectionHomogenize('GEOMETRYCOLLECTION (POLYGON ((10 50,50 50,50 10,10 10,10 ↩
    50)),POLYGON ((90 50,90 10,50 10,50 50,90 50)))');
```

Output

```
MULTIPOLYGON(((10 50,50 50,50 10,10 10,10 50)),((90 50,90 10,50 10,50 50,90 50)))
```

Figure



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

Code

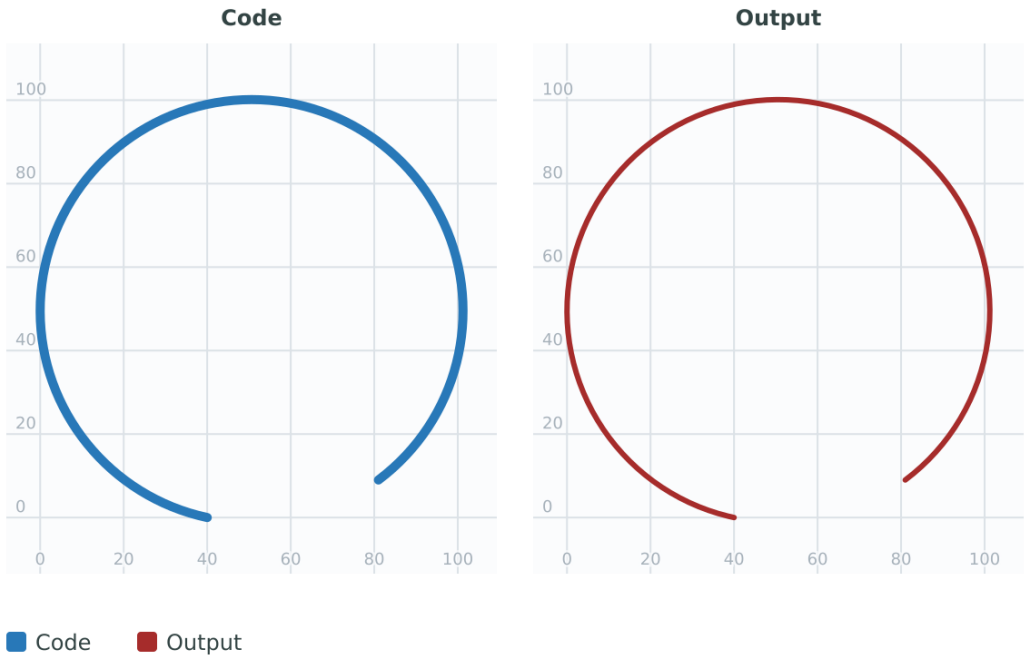
```
SELECT ST_CurveToLine('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)');
```

Output

```
LINESTRING(220268 150415,220269.95064912 150416.539364228,220271.823415575 150418.17258804,220273.613787707 ↩
    150419.895736857,
220275.317452352 150421.704659462,220276.930305234 150423.594998003,220278.448460847 150425.562198489,
220279.868261823 150427.60152176,220281.186287736 150429.708054909,220282.399363347 150431.876723113,
220283.50456625 150434.10230186,220284.499233914 150436.379429536,220285.380970099 150438.702620341,220286.147650624 ↩
    150441.066277505,
220286.797428488 150443.464706771,220287.328738321 150445.892130112,220287.740300149 150448.342699654,
220288.031122486 150450.810511759,220288.200504713 150453.289621251,220288.248038775 150455.77405574,
220288.173610157 150458.257830005,220287.977398166 150460.734960415,220287.659875492 150463.199479347,
220287.221807076 150465.64544956,220286.664248262 150468.066978495,220285.988542259 150470.458232479,220285.196316903 ↩
    150472.81345077,
220284.289480732 150475.126959442,220283.270218395 150477.39318505,220282.140985384 150479.606668057,
220280.90450212 150481.762075989,220279.5637474 150483.85421628,220278.12195122 150485.87804878,
220276.582586992 150487.828697901,220274.949363179 150489.701464356,220273.226214362 150491.491836488,
220271.417291757 150493.195501133,220269.526953216 150494.808354014,220267.559752731 150496.326509628,
220265.520429459 150497.746310603,220263.41389631 150499.064336517,220261.245228106 150500.277412127,
220259.019649359 150501.38261503,220256.742521683 150502.377282695,220254.419330878 150503.259018879,
220252.055673714 150504.025699404,220249.657244448 150504.675477269,220247.229821107 150505.206787101,
220244.779251566 150505.61834893,220242.311439461 150505.909171266,220239.832329968 150506.078553494,
220237.347895479 150506.126087555,220234.864121215 150506.051658938,220232.386990804 150505.855446946,
220229.922471872 150505.537924272,220227.47650166 150505.099855856,220225.054972724 150504.542297043,
220222.663718741 150503.86659104,220220.308500449 150503.074365683,
220217.994991777 150502.167529512,220215.72876617 150501.148267175,
220213.515283163 150500.019034164,220211.35987523 150498.7825509,
220209.267734939 150497.441796181,220207.243902439 150496,
220205.293253319 150494.460635772,220203.420486864 150492.82741196,220201.630114732 150491.104263143,
220199.926450087 150489.295340538,220198.313597205 150487.405001997,220196.795441592 150485.437801511,
220195.375640616 150483.39847824,220194.057614703 150481.291945091,220192.844539092 150479.123276887,220191.739336189 ↩
    150476.89769814,
```

```
220190.744668525 150474.620570464,220189.86293234 150472.297379659,220189.096251815 150469.933722495,
220188.446473951 150467.535293229,220187.915164118 150465.107869888,220187.50360229 150462.657300346,
220187.212779953 150460.189488241,220187.043397726 150457.710378749,220186.995863664 150455.22594426,
220187.070292282 150452.742169995,220187.266504273 150450.265039585,220187.584026947 150447.800520653,
220188.022095363 150445.35455044,220188.579654177 150442.933021505,220189.25536018 150440.541767521,
220190.047585536 150438.18654923,220190.954421707 150435.873040558,220191.973684044 150433.60681495,
220193.102917055 150431.393331943,220194.339400319 150429.237924011,220195.680155039 150427.14578372,220197.12195122 ↩
    150425.12195122,
220198.661315447 150423.171302099,220200.29453926 150421.298535644,220202.017688077 150419.508163512,220203.826610682 ↩
    150417.804498867,
220205.716949223 150416.191645986,220207.684149708 150414.673490372,220209.72347298 150413.253689397,220211.830006129 ↩
    150411.935663483,
220213.998674333 150410.722587873,220216.22425308 150409.61738497,220218.501380756 150408.622717305,220220.824571561 ↩
    150407.740981121,
220223.188228725 150406.974300596,220225.586657991 150406.324522731,220227 150406)
```

Figure



This example uses 3D input.

Code

```
SELECT ST_CurveToLine('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)');
```

Output

```
LINESTRING(220268 150415 1,220269.95064912 150416.539364228 1.0181172856673,
220271.823415575 150418.17258804 1.03623457133459,220273.613787707 150419.895736857 1.05435185700189,....AD INFINITUM ↩
    ....
   220225.586657991 150406.324522731 1.32611114201132,220227 150406 3)
```

Use only 2 segments to approximate quarter circle.

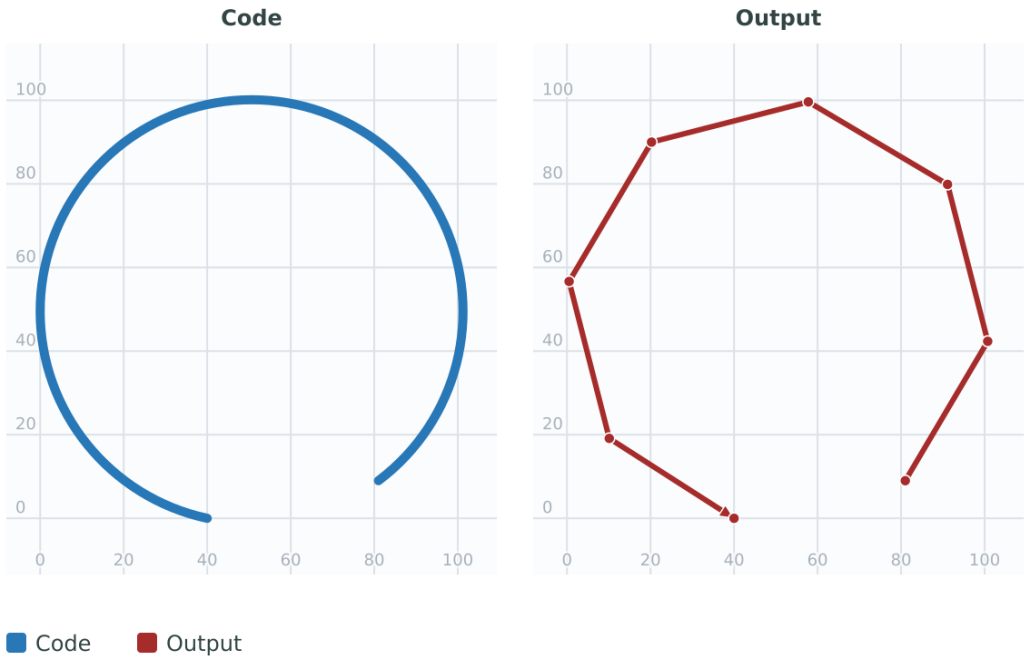
Code

```
SELECT ST_CurveToLine('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)', 2);
```

Output

```
LINESTRING(220268 150415,220287.740300149 150448.342699654,220278.12195122 150485.87804878,
220244.779251566 150505.61834893,220207.243902439 150496,220187.50360229 150462.657300346,
220197.12195122 150425.12195122,220227 150406)
```

Figure



Ensure the approximated line is no more than 20 units away from the original curve, and make the result direction-neutral.

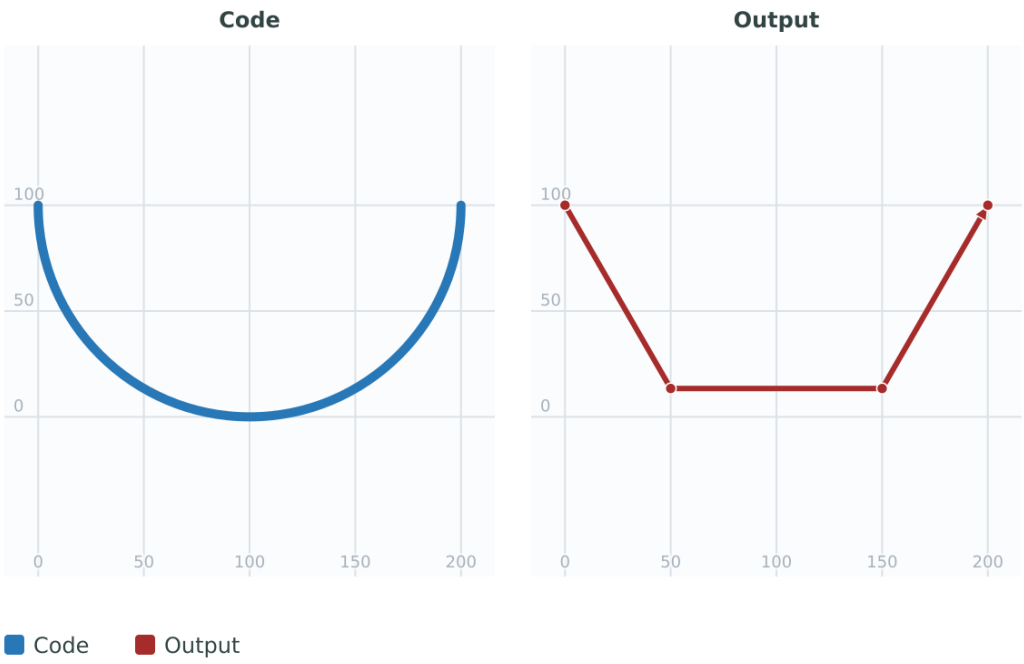
Code

```
SELECT ST_CurveToLine(
    'CIRCULARSTRING(0 0,100 -100,200 0)'::geometry,
    20,
    1,
    1
  );
```

Output

```
LINESTRING(0 0,49.99999999999996 -86.60254037844383,149.99999999999994 -86.6025403784439,200 0)
```

Figure



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

Make a closed line start at its 3rd vertex.

Code

```
SELECT ST_Scroll('SRID=4326;LINESTRING(0 0 0 1,10 0 2 0,5 5 4 2,0 0 0 1)', 'POINT(5 5 4 2)' ↩
    );
```

Output

```
SRID=4326;LINESTRING(5 5 4 2,0 0 0 1,10 0 2 0,5 5 4 2)
```

Figure



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

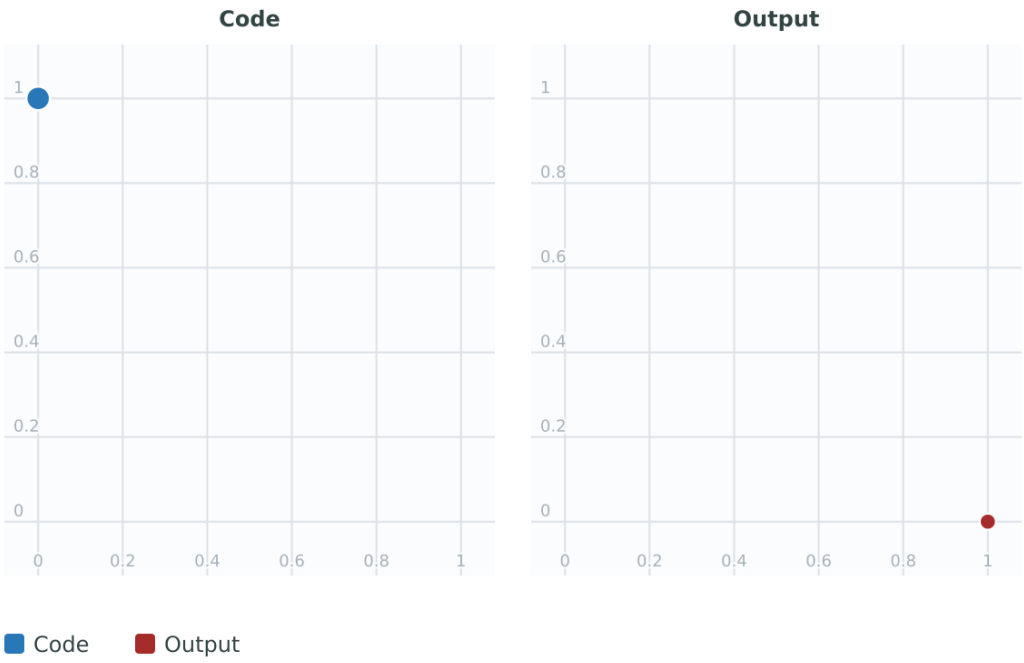
Code

```
SELECT ST_FlipCoordinates(GeomFromEWKT('POINT(1 2)'));
```

Output

```
POINT(2 1)
```

Figure



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

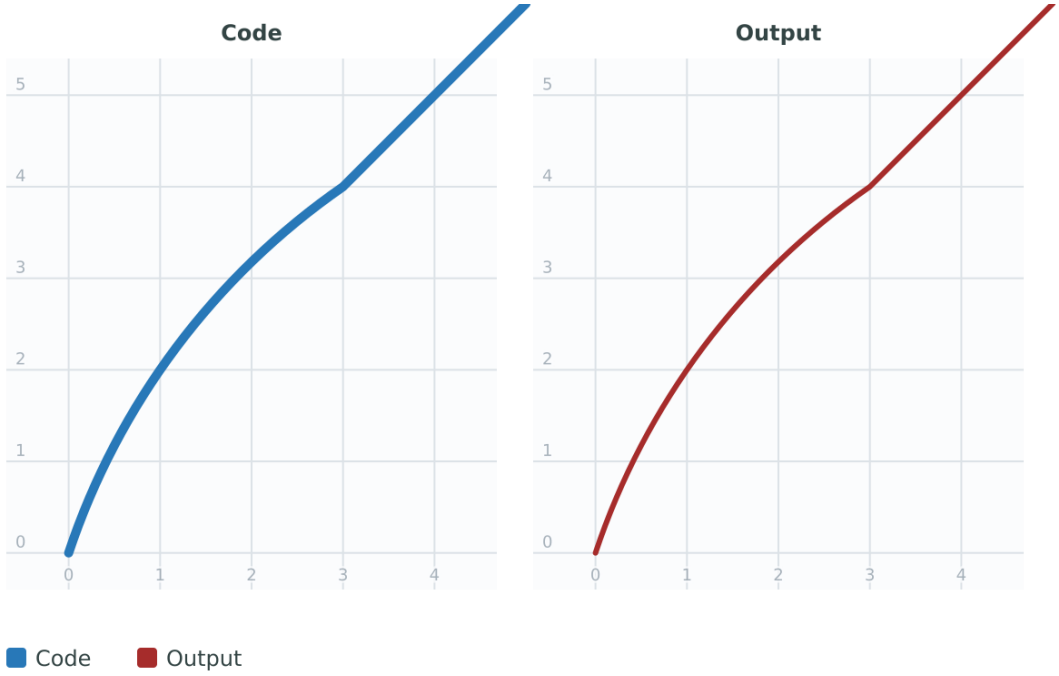
Code

```
SELECT ST_Force2D('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1,2 3,4 5,6 7,5 6)
```

Figure



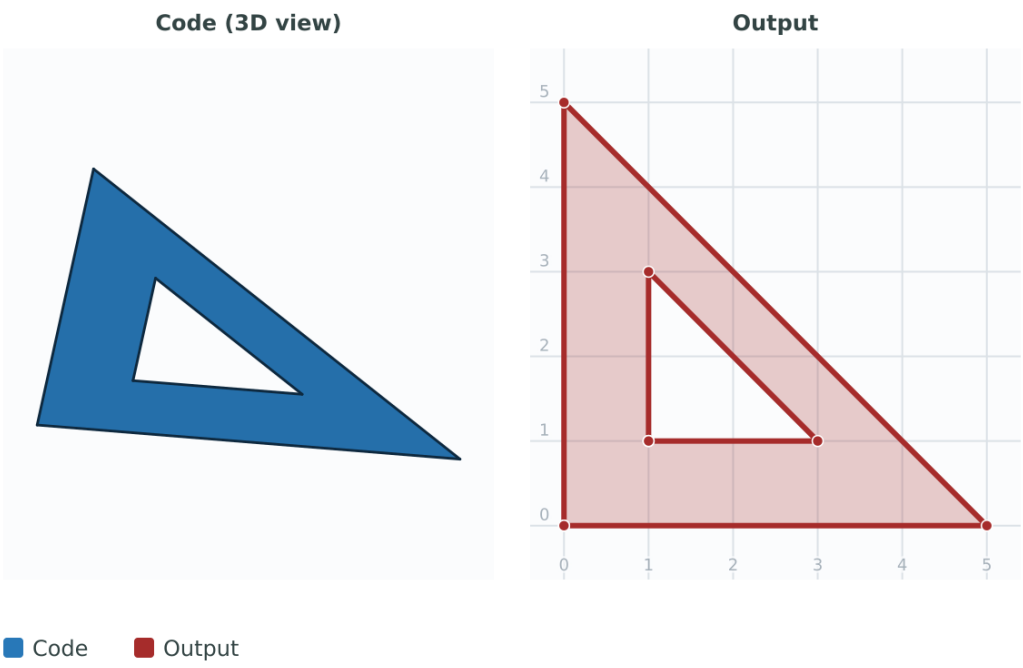
Code

```
SELECT ST_Force2D('POLYGON((0 0 2,0 5 2,5 0 2,0 0 2),(1 1 2,3 1 2,1 3 2,1 1 2))');
```

Output

```
POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))
```

Figure



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.

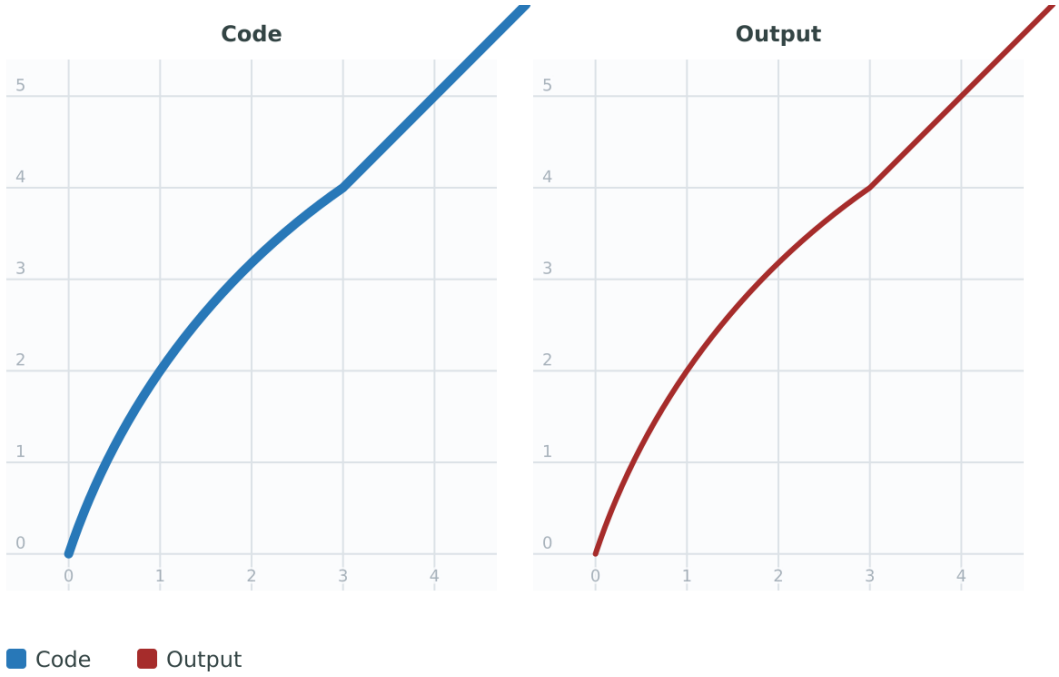
Code

```
SELECT ST_Force3D('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)
```

Figure



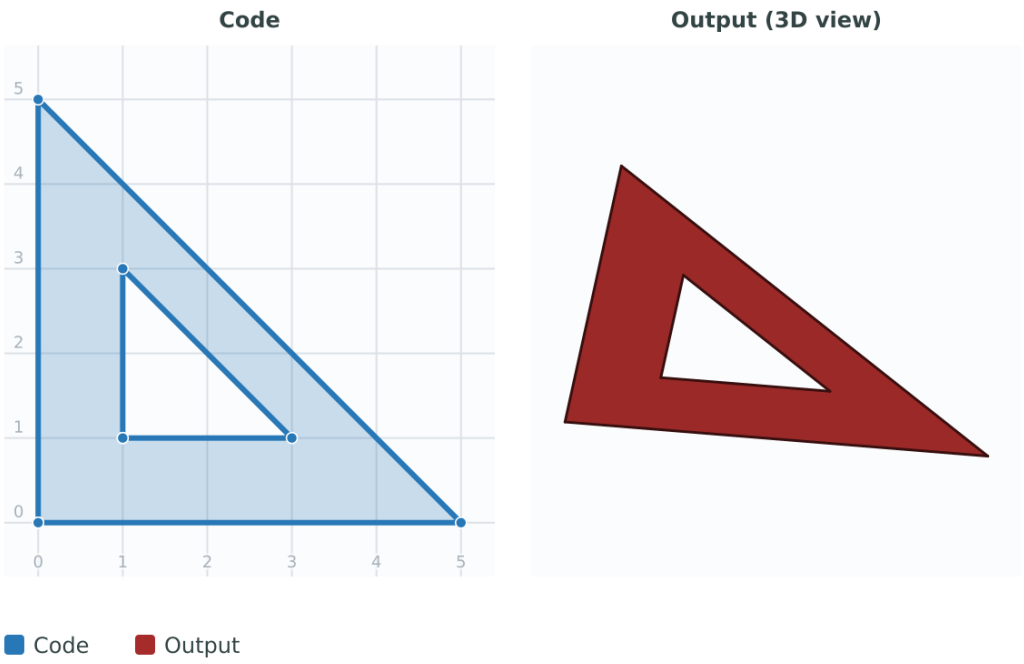
Code

```
SELECT ST_Force3D('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))');
```

Output

```
POLYGON((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0))
```

Figure



相关信息

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3DZ

7.5.9 ST_Force3DZ

ST_Force3DZ — 强制几何体形式入 XYZ 模式。

Synopsis

geometry **ST_Force3DZ**(geometry geomA, float Zvalue = 0.0);

描述

强制几何体形式入 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。

 函数支持多面体曲面。

 函数支持 3d 并且不会丢失 z-index。

 此方法支持形字符串和曲。

示例

Nothing happens to an already 3D geometry.

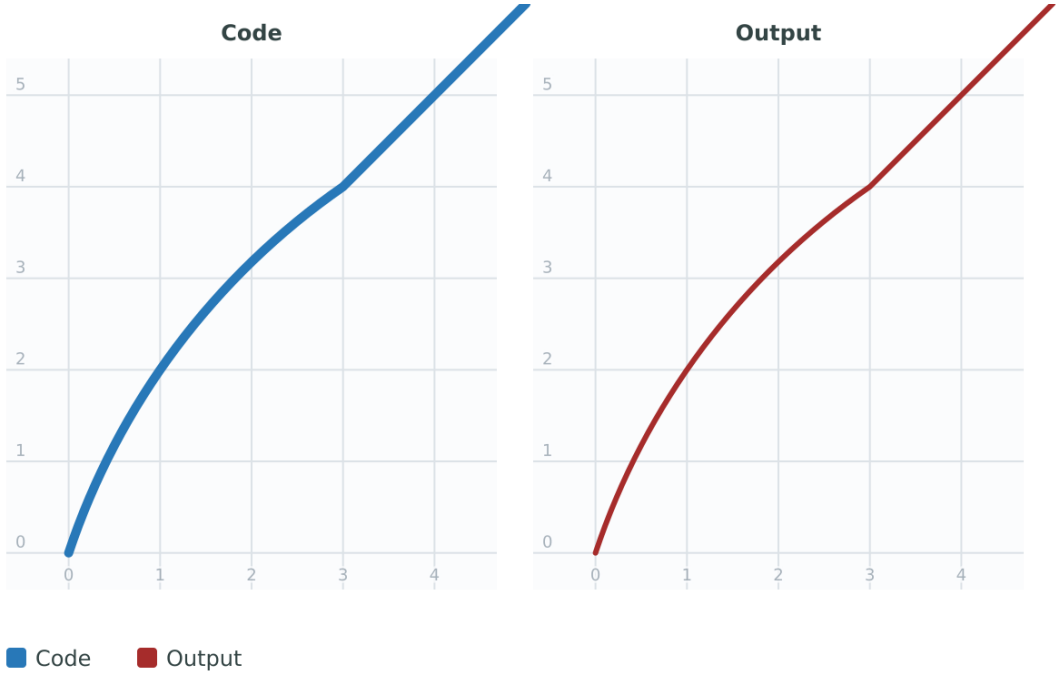
Code

```
SELECT ST_Force3DZ('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)
```

Figure



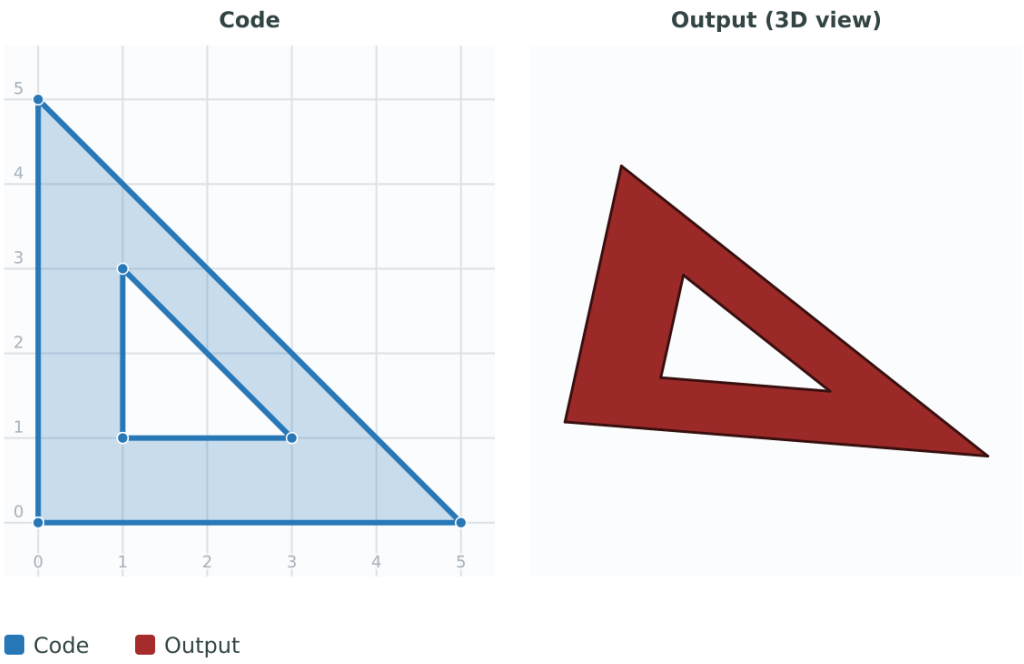
Code

```
SELECT ST_Force3DZ('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))');
```

Output

```
POLYGON((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0))
```

Figure



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.

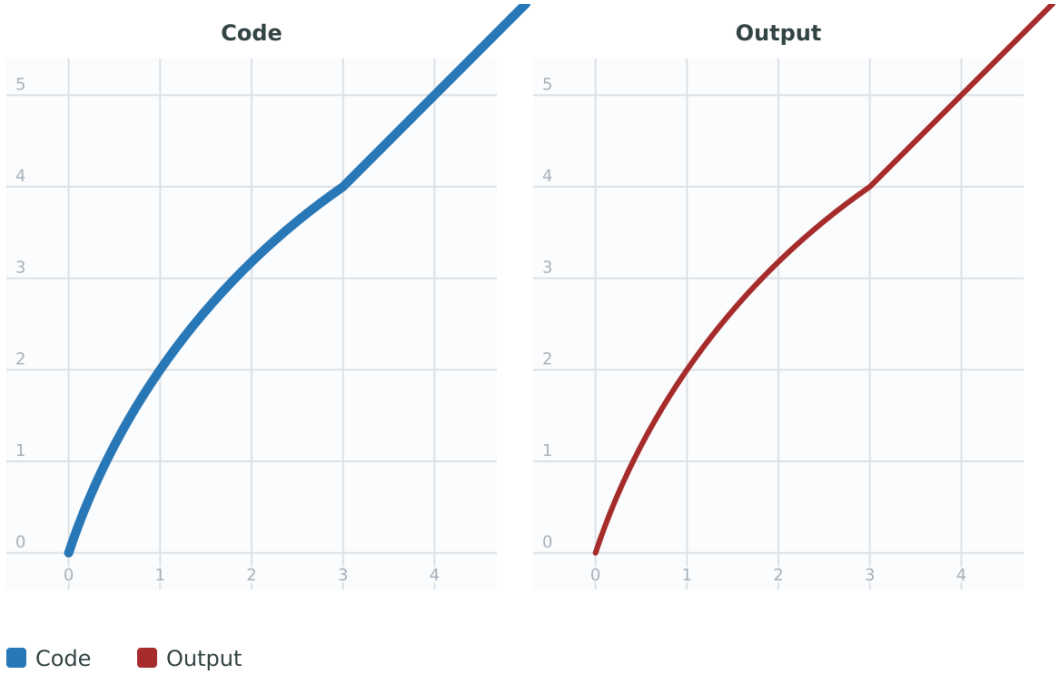
Code

```
SELECT ST_Force3DM('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRINGM(1 1 0,2 3 0,4 5 0,6 7 0,5 6 0)
```

Figure



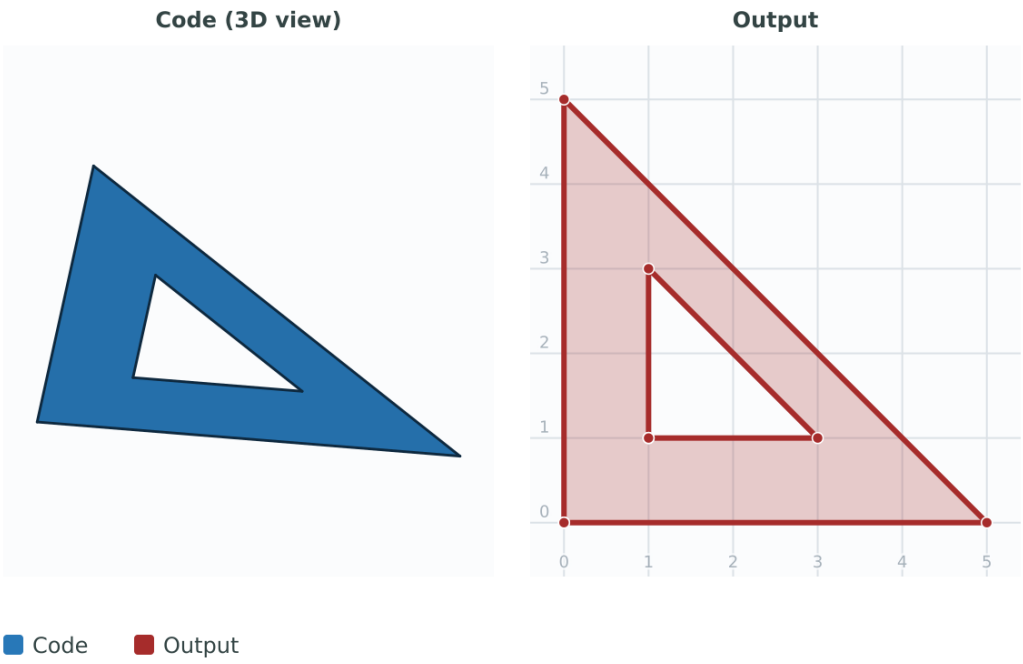
Code

```
SELECT ST_Force3DM('POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))');
```

Output

```
POLYGONM((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0))
```

Figure



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

强制几何体形式入 XYZM 模式。如果没有 Z 坐标或 M 坐标，则添加 *Z* 坐标和 *M* 坐标。
更改：2.1.0。在 2.0.x 期间，它被称为 ST_Force_4D。
更改：3.1.0. 现在可以指定非零 Z 和 M 值。
此函数支持 3d 并且不会丢失 z-index。
此方法支持曲线形字符串和曲面。

示例

Nothing happens to an already 3D geometry.

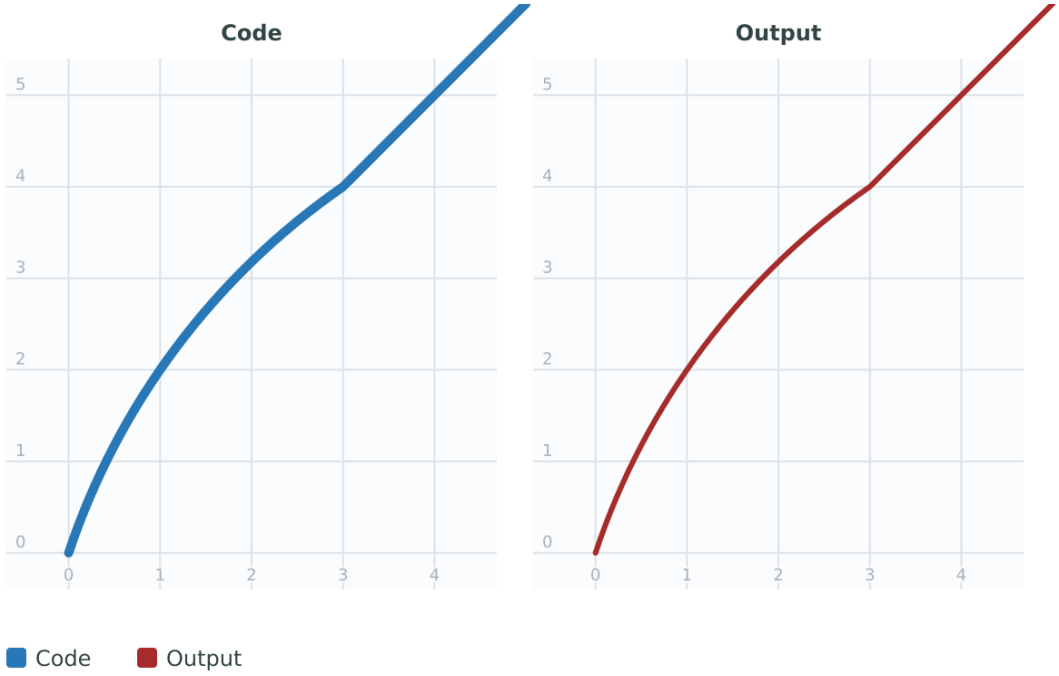
Code

```
SELECT ST_Force4D('CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)');
```

Output

```
CIRCULARSTRING(1 1 2 0,2 3 2 0,4 5 2 0,6 7 2 0,5 6 2 0)
```

Figure



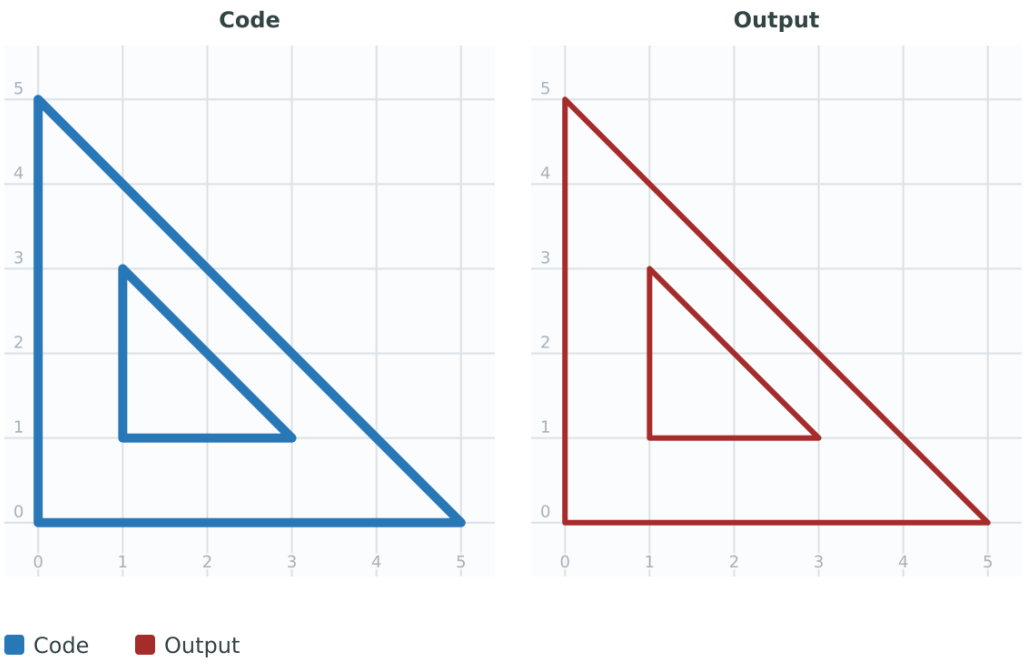
Code

```
SELECT ST_Force4D('MULTILINESTRINGM((0 0 1,0 5 2,5 0 3,0 0 4),(1 1 1,3 1 1,1 3 1,1 1 1))');
```

Output

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

Figure



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

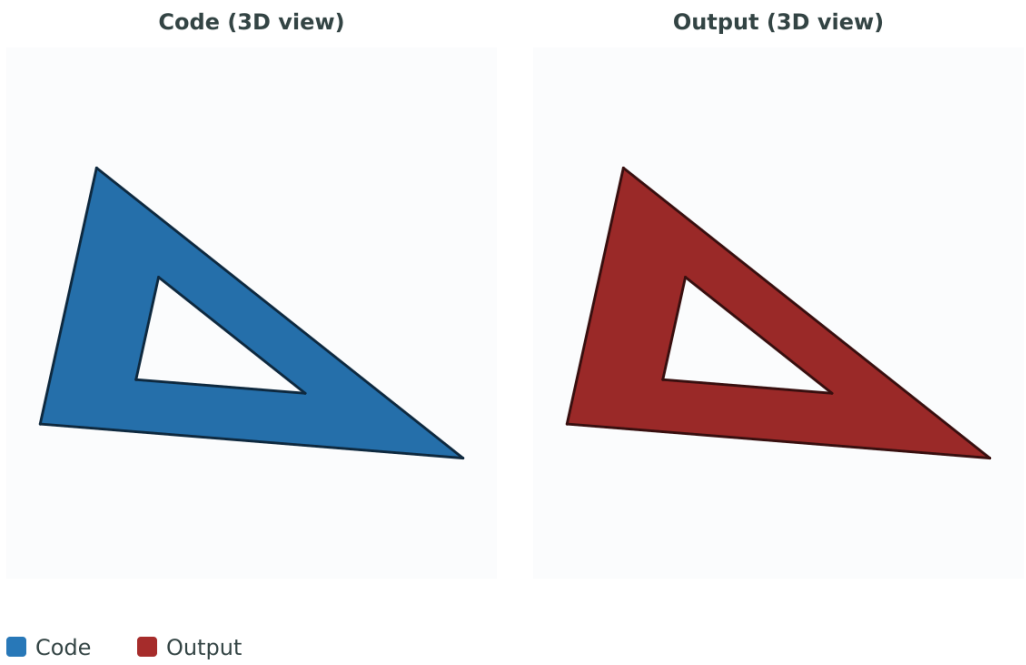
Code

```
SELECT ST_ForceCollection('POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))');
```

Output

```
GEOMETRYCOLLECTION(POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1)))
```

Figure



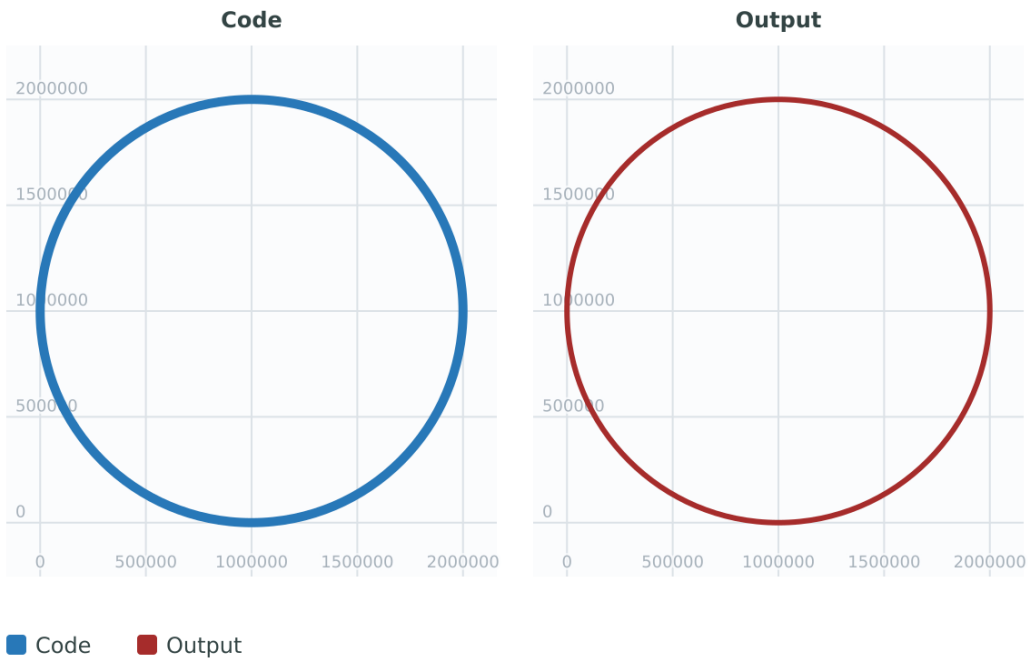
Code

```
SELECT ST_ForceCollection('CIRCULARSTRING(220227 150406,2220227 150407,220227 150406)');
```

Output

```
GEOMETRYCOLLECTION(CIRCULARSTRING(220227 150406,2220227 150407,220227 150406))
```

Figure



This example uses a PolyhedralSurface.

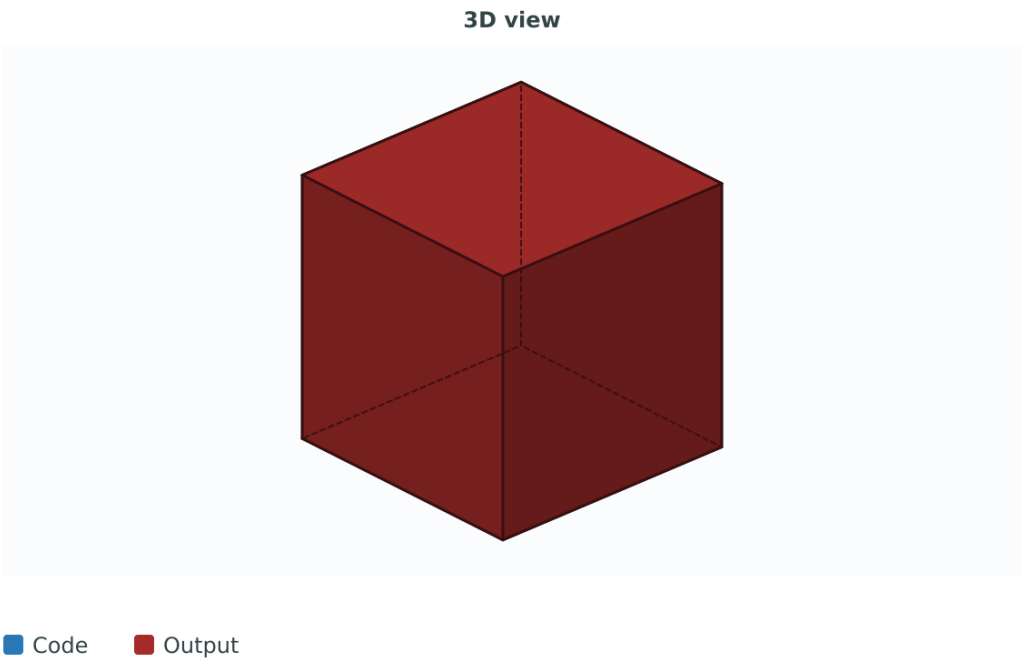
Code

```
SELECT ST_ForceCollection('POLYHEDRALSURFACE(((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
 ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
 ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
 ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
 ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
 ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)))');
```

Output

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

Figure



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

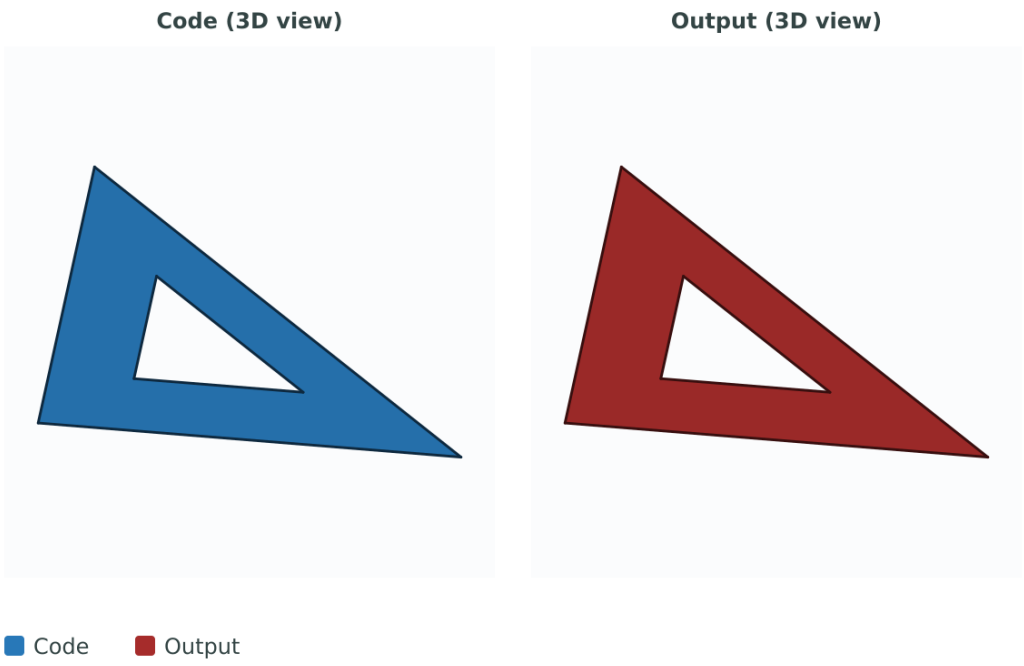
Code

```
SELECT ST_ForceCurve('POLYGON((0 0 2,5 0 2,0 5 2,0 0 2),(1 1 2,1 3 2,3 1 2,1 1 2))':: ↩
    geometry
  );
```

Output

```
CURVEPOLYGON Z ((0 0 2,5 0 2,0 5 2,0 0 2),(1 1 2,1 3 2,3 1 2,1 1 2))
```

Figure



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☒。

示例

Force a polygon and its hole to use counter-clockwise exterior rings and clockwise interior rings.

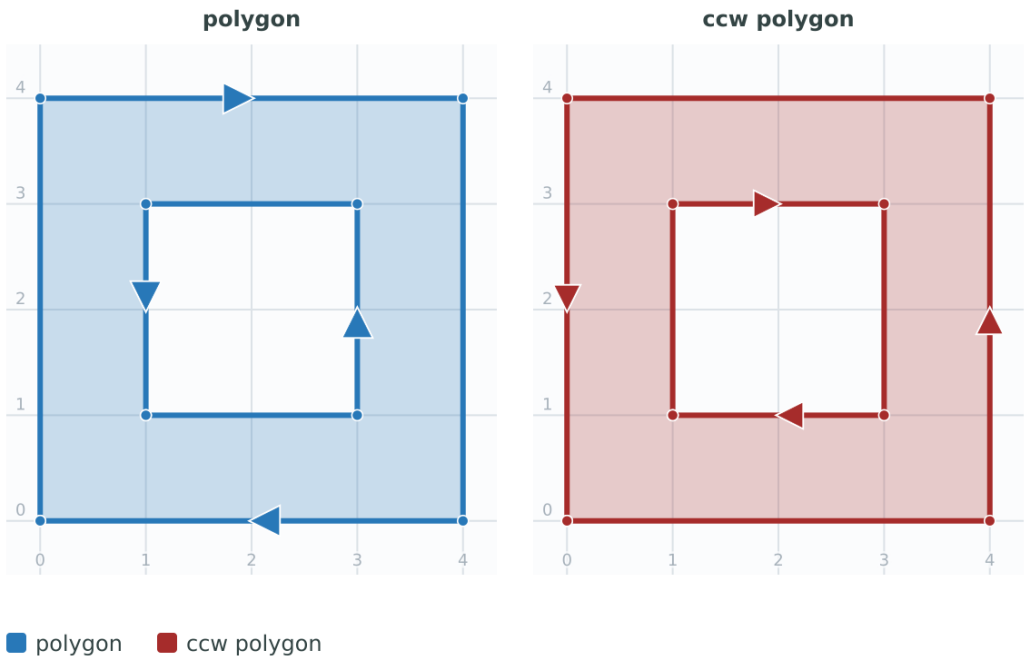
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,0 4,4 4,4 0,0 0),
                  (1 1,3 1,3 3,1 3,1 1))'::geometry AS geom
)
SELECT geom AS input_polygon,
       ST_ForcePolygonCCW(geom) AS ccw_polygon
FROM data;
```

Output

```
POLYGON((0 0,0 4,4 4,4 0,0 0),(1 1,3 1,3 3,1 3,1 1)) | POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,1 3,3 3,3 1,1 1))
```

Figure



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□。

示例

Force a polygon and its hole to use clockwise exterior rings and counter-clockwise interior rings.

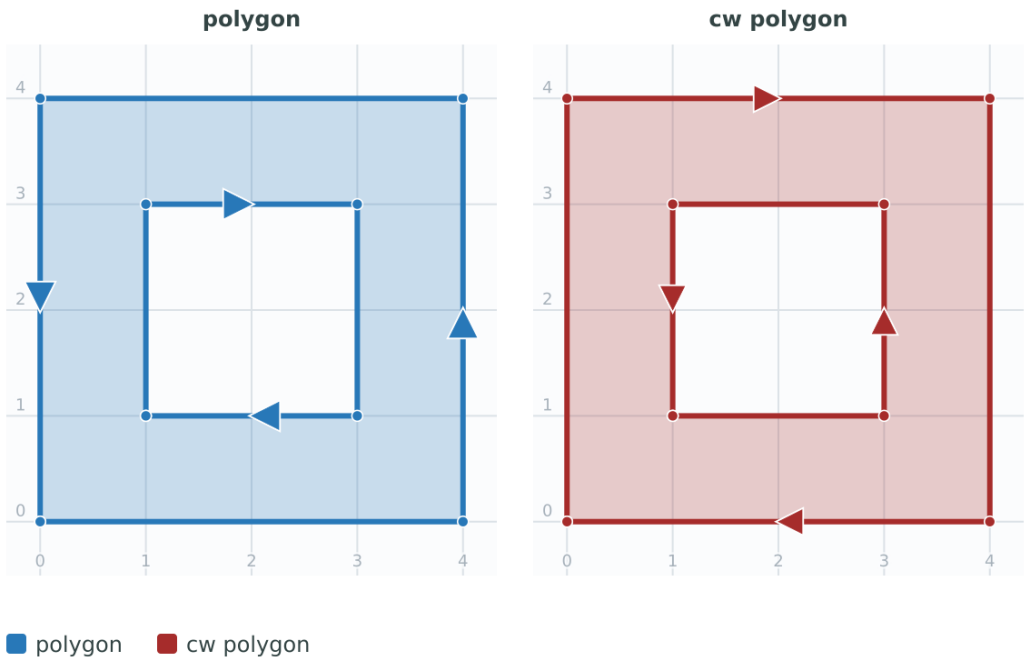
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,4 0,4 4,0 4,0 0),
                  (1 1,1 3,3 3,3 1,1 1))'::geometry AS geom
)
SELECT geom AS input_polygon,
       ST_ForcePolygonCW(geom) AS cw_polygon
FROM data;
```

Output

```
POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,1 3,3 3,3 1,1 1)) | POLYGON((0 0,0 4,4 4,4 0,0 0),(1 1,3 1,3 3,1 3,1 1))
```

Figure



相关信息

ST_ForcePolygonCCW, ST_IsPolygonCCW, ST_IsPolygonCW

7.5.16 ST_ForceSFS

ST_ForceSFS — Force geometries to use SFS 1.1 or 1.2 geometry types.

Synopsis

geometry **ST_ForceSFS**(geometry geomA);
geometry **ST_ForceSFS**(geometry geomA, text version);

描述

Forces a geometry to use the geometry types defined by the requested version of the OGC Simple Features Specification. The default is SFS 1.1. Curved geometries are converted to linear representations, `TRIANGLE` is converted to `POLYGON`, and `TIN` and `POLYHEDRALSURFACE` are converted to `GEOMETRYCOLLECTION` values containing `POLYGON` elements.

If *`version`* begins with `1.2`, curved geometries are still converted to linear representations, but `TRIANGLE`, `TIN`, and `POLYHEDRALSURFACE` are preserved. Other version values use the default SFS 1.1 behavior. Geometry collections are processed recursively, and the SRID and Z/M coordinates are preserved.

可用性：2.1.0

 ☒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。

示例

Compare the geometry types produced for SFS 1.1 (the default) and SFS 1.2. Read each row from the input type in the direction of the result columns.

Code

```
WITH inputs(geom) AS (VALUES
    ('CIRCULARSTRING(0 0, 1 1, 2 0)'::geometry),
    ('TRIANGLE((0 0, 0 2, 2 0, 0 0))'::geometry),
    ('TIN(((0 0, 0 2, 2 0, 0 0)))'::geometry),
    ('POLYHEDRALSURFACE(((0 0, 0 2, 2 0, 0 0)))'::geometry)
)
SELECT ST_GeometryType(geom) || ' b''→b''' AS input,
       ST_GeometryType(ST_ForceSFS(geom)) AS "SFS 1.1 (default)",
       ST_GeometryType(ST_ForceSFS(geom, '1.2')) AS "SFS 1.2"
FROM inputs;
```

Output

```
input          |   SFS 1.1 (default)   |       SFS 1.2
-----------------------+-----------------------+----------------------
 ST_CircularString b''→b''    | ST_LineString         | ST_LineString
 ST_Triangle b''→b''          | ST_Polygon            | ST_Triangle
 ST_Tin b''→b''               | ST_GeometryCollection | ST_Tin
 ST_PolyhedralSurface b''→b'' | ST_GeometryCollection | ST_PolyhedralSurface
(4 rows)
```

相关信息

ST_CurveToLine, ST_ForceCurve

7.5.17 ST_ForceRHR

ST_ForceRHR — 强制多边形顶点的方向遵循右手定则。

Synopsis

geometry **ST_ForceRHR**(geometry g);

描述

强制多边形中顶点的方向遵循右手定则，其中多边形所包含的区域位于边界的右侧。特别地，外环沿顺时针方向定向，而内环沿逆时针方向定向。该函数是 ST_ForcePolygonCW的同义词



Note
右手定则的上述定义与其他上下文中使用的定义相冲突。为避免混淆，建议使用 ST_ForcePolygonCW。

增强：2.0.0 引入了对多面体曲面的支持。


该函数支持 3d 并且不会删除 z-index。


该函数支持多面体曲面。

示例

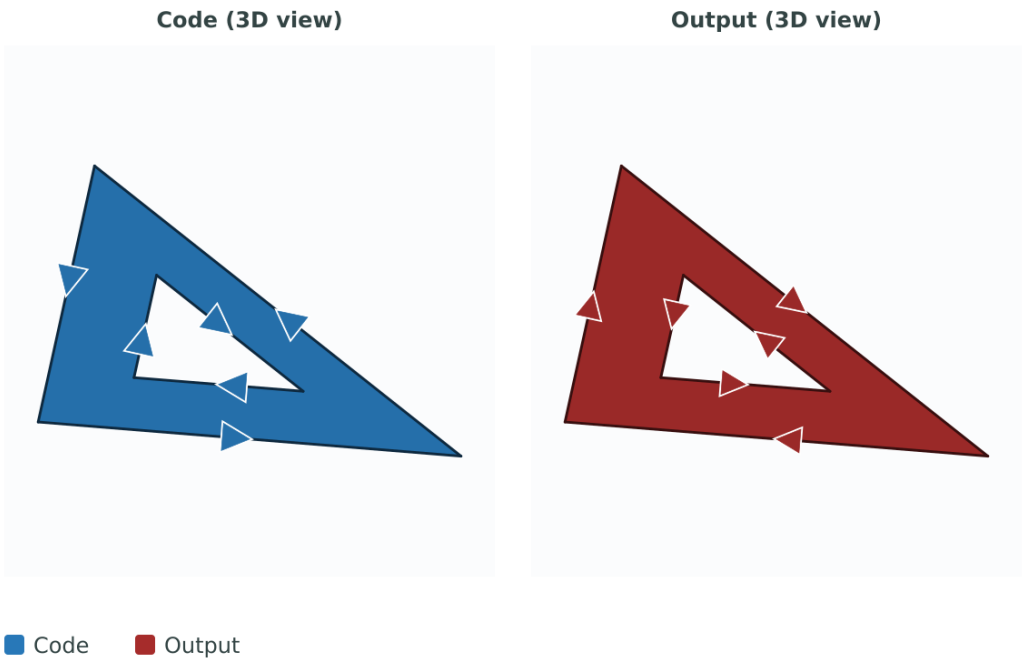
Code

```
SELECT ST_ForceRHR('POLYGON((0 0 2,5 0 2,0 5 2,0 0 2),(1 1 2,1 3 2,3 1 2,1 1 2))'
  );
```

Output

```
POLYGON((0 0 2,0 5 2,5 0 2,0 0 2),(1 1 2,3 1 2,1 3 2,1 1 2))
```

Figure



相关信息

ST_ForcePolygonCCW, ST_ForcePolygonCW, ST_IsPolygonCCW, ST_IsPolygonCW, ST_BuildArea, ST_Polygo
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

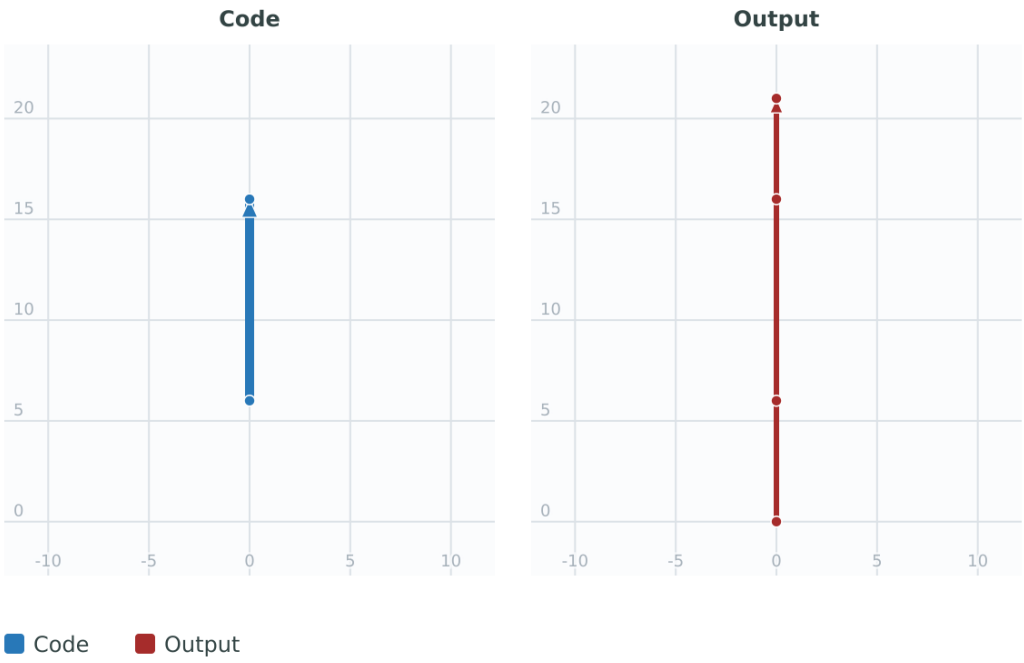
Code

```
SELECT ST_LineExtend('LINESTRING(0 0,0 10)'::geometry, 5, 6);
```

Output

```
LINESTRING(0 -6,0 0,0 10,0 15)
```

Figure



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⊠。

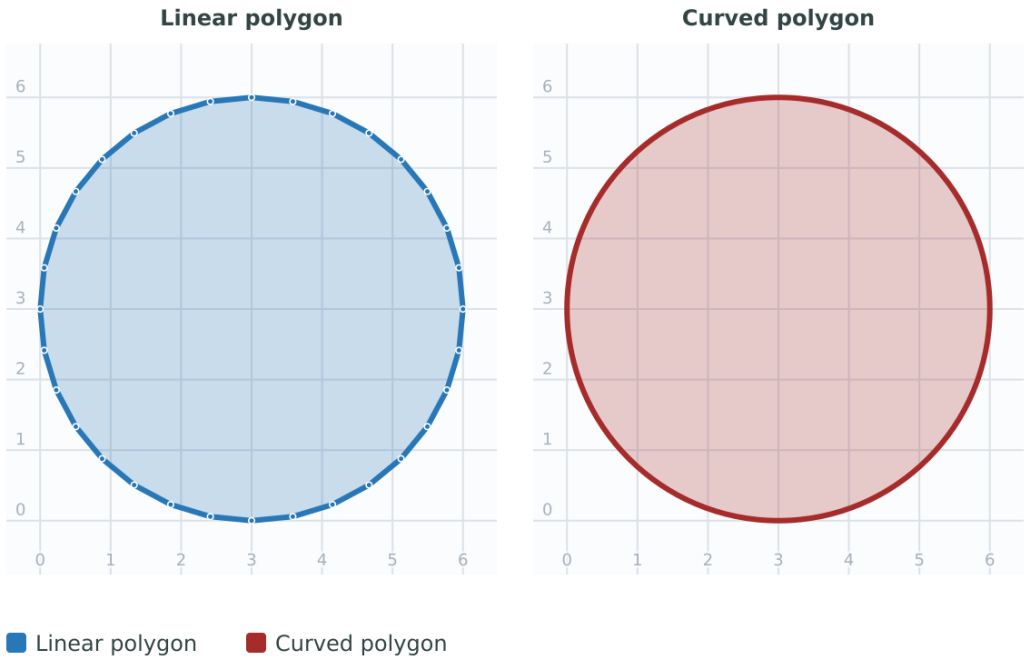
示例

This example converts a linear Polygon approximating a circle to a CurvePolygon.

Code

```
WITH data AS (
    SELECT ST_Buffer('POINT(1 3)'::geometry, 3) AS geom
)
SELECT geom AS "input_Linear polygon",
       ST_LineToCurve(geom) AS "Curved polygon"
FROM data;
```

Figure

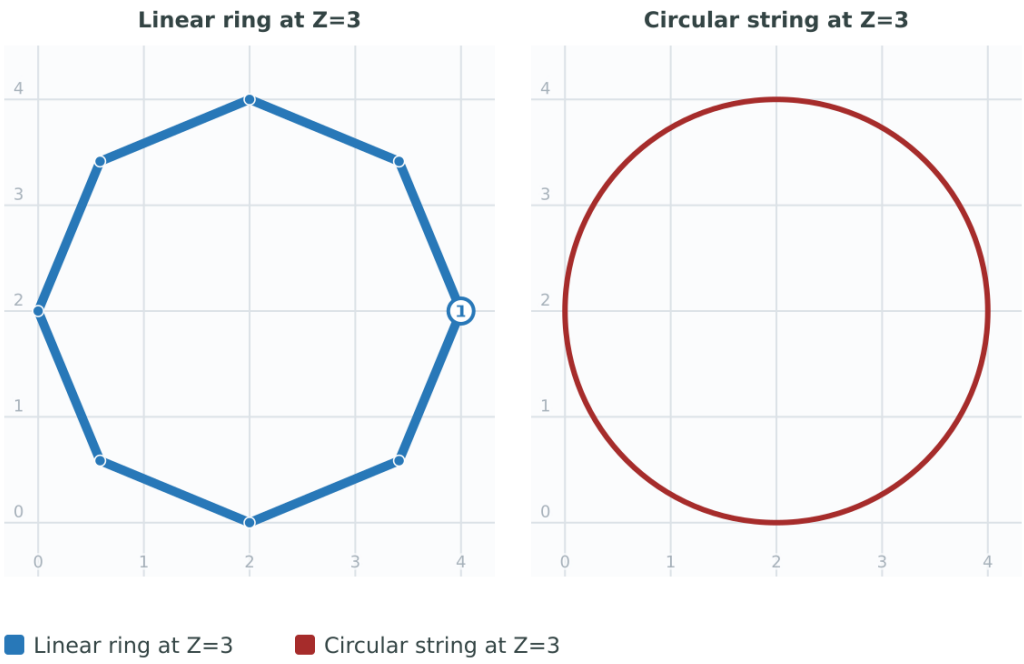


This example converts a 3D linear ring to a CircularString while preserving Z.

Code

```
WITH data AS (
    SELECT ST_Translate(
        ST_Force3D(ST_Boundary(ST_Buffer(ST_Point(1, 3), 2, 2))),
        0, 0, 3
    ) AS geom
)
SELECT geom AS "input_Linear ring at Z=3",
       ST_LineToCurve(geom) AS "Circular string at Z=3"
FROM data;
```

Figure



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

A Polygon is converted to a single-element MultiPolygon. The coordinates are unchanged.

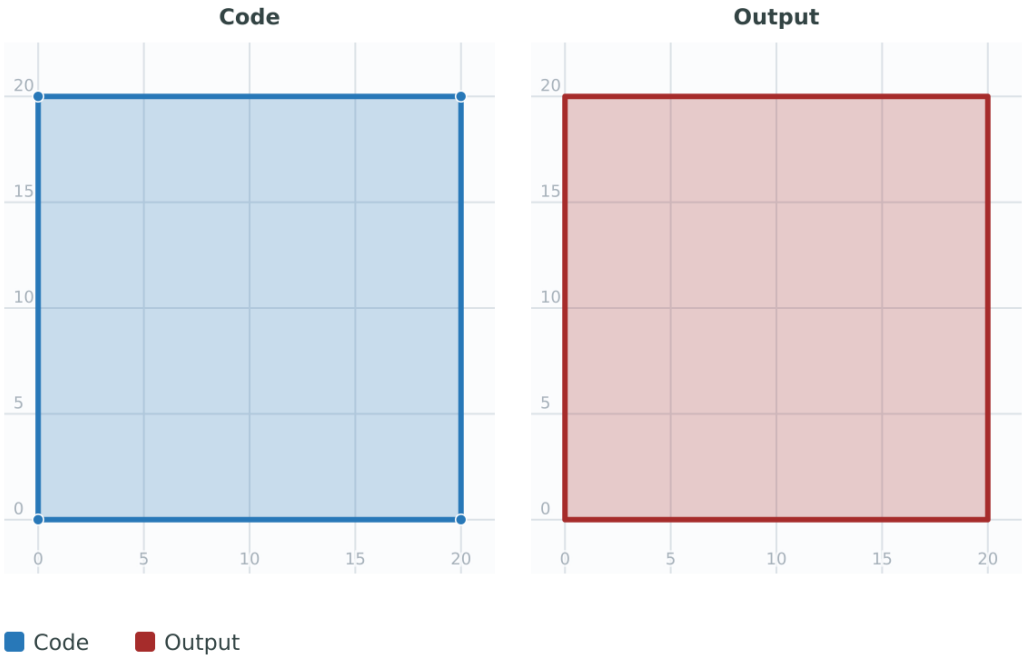
Code

```
SELECT ST_Multi('POLYGON ((10 30,30 30,30 10,10 10,10 30))');
```

Output

```
MULTIPOLYGON(((10 30,30 30,30 10,10 10,10 30)))
```

Figure



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

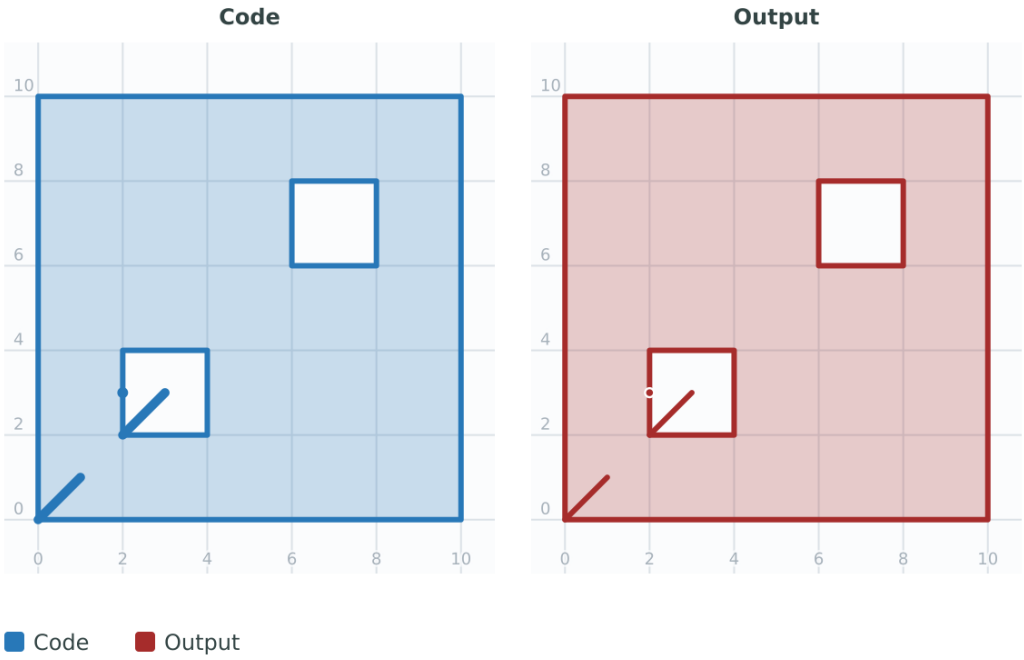
Code

```
SELECT ST_Normalize('GEOMETRYCOLLECTION(
    POINT(2 3),
    MULTILINESTRING((0 0,1 1),(2 2,3 3)),
    POLYGON(
      (0 10,0 0,10 0,10 10,0 10),
      (4 2,2 2,2 4,4 4,4 2),
      (6 8,8 8,8 6,6 6,6 8)
    )
  )');
```

Output

```
GEOMETRYCOLLECTION(POLYGON((0 0,0 10,10 10,10 0,0 0),(6 6,8 6,8 8,6 8,6 6),(2 2,4 2,4 4,2 4,2 2)), MULTILINESTRING((2 ↩
    2,3 3),(0 0,1 1)), POINT(2 3))
```

Figure



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

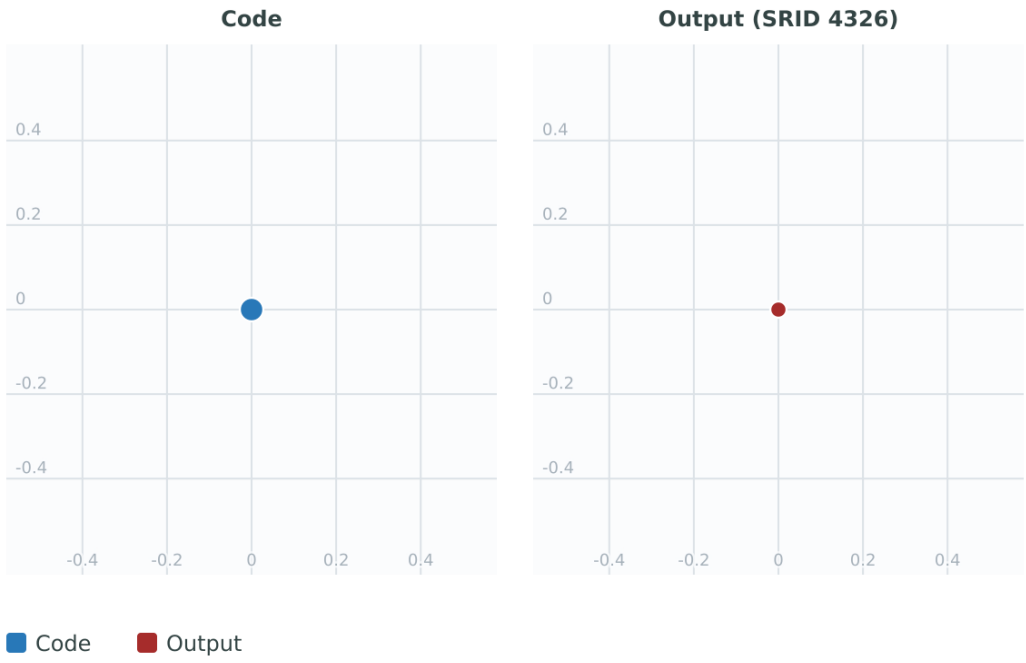
Code

```
SELECT ST_Project('POINT(0 0)'::geography, 100000, radians(45.0));
```

Output

```
POINT(0.635231029125537 0.639472334729198)
```

Figure



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 = NULL, int prec_z = NULL, int prec_m = NULL);

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

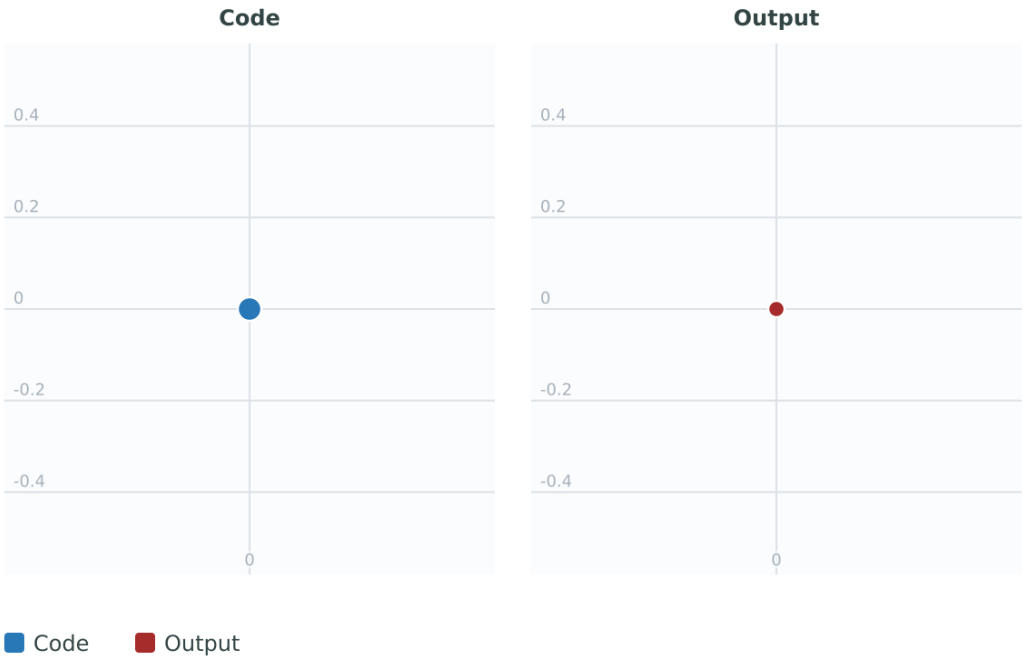
Code

```
SELECT ST_QuantizeCoordinates('POINT (100.123456 0)'::geometry, 4);
```

Output

```
POINT(100.12344360351562 0)
```

Figure



Code

```
WITH test AS (SELECT 'POINT (123.456789123456 123.456789123456)'::geometry AS geom)
SELECT
  digits,
  encode(ST_QuantizeCoordinates(geom, digits), 'hex'),
  ST_QuantizeCoordinates(geom, digits)
FROM test, generate_series(15, -15, -1) AS digits;
```

Output

```
digits |                  encode                  |                 st_astext
--------+------------------------------------------+-------------------------------------------
     15 | 01010000005f9a72083cdd5e405f9a72083cdd5e40 | POINT(123.456789123456 123.456789123456)
     14 | 01010000005f9a72083cdd5e405f9a72083cdd5e40 | POINT(123.456789123456 123.456789123456)
     13 | 01010000005e9a72083cdd5e405e9a72083cdd5e40 | POINT(123.45678912345599 123.45678912345599)
     12 | 0101000000409a72083cdd5e40409a72083cdd5e40 | POINT(123.45678912345556 123.45678912345556)
     11 | 0101000000009a72083cdd5e40009a72083cdd5e40 | POINT(123.45678912345466 123.45678912345466)
     10 | 0101000000009872083cdd5e40009872083cdd5e40 | POINT(123.45678912344738 123.45678912344738)
      9 | 0101000000008072083cdd5e40008072083cdd5e40 | POINT(123.45678912336007 123.45678912336007)
      8 | 0101000000000070083cdd5e40000070083cdd5e40 | POINT(123.45678912103176 123.45678912103176)
      7 | 0101000000000060083cdd5e40000060083cdd5e40 | POINT(123.4567891061306 123.4567891061306)
      6 | 0101000000000000083cdd5e40000000083cdd5e40 | POINT(123.45678901672363 123.45678901672363)
      5 | 0101000000000000003cdd5e40000000003cdd5e40 | POINT(123.456787109375 123.456787109375)
      4 | 0101000000000000003cdd5e40000000003cdd5e40 | POINT(123.456787109375 123.456787109375)
      3 | 0101000000000000038dd5e400000000038dd5e40 | POINT(123.45654296875 123.45654296875)
      2 | 0101000000000000000dd5e400000000000dd5e40 | POINT(123.453125 123.453125)
      1 | 0101000000000000000dc5e400000000000dc5e40 | POINT(123.4375 123.4375)
      0 | 0101000000000000000c05e400000000000c05e40 | POINT(123 123)
     -1 | 0101000000000000000005e400000000000005e40 | POINT(120 120)
     -2 | 0101000000000000000005840000000000005840 | POINT(96 96)
     -3 | 0101000000000000000005840000000000005840 | POINT(96 96)
     -4 | 0101000000000000000005840000000000005840 | POINT(96 96)
     -5 | 0101000000000000000005840000000000005840 | POINT(96 96)
```

```
     -6 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -7 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -8 | 0101000000000000000000058400000000000005840 | POINT(96 96)
     -9 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -10 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -11 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -12 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -13 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -14 | 0101000000000000000000058400000000000005840 | POINT(96 96)
    -15 | 0101000000000000000000058400000000000005840 | POINT(96 96)
(31 rows)
```

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

Code

```
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 = 0.0);

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。

如果提供了非零的 *tolerance* 参数，⊠在⊠容差距离内的⊠点将被⊠⊠重复点。⊠距离在二⊠平面（XY 平面）上⊠算。

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

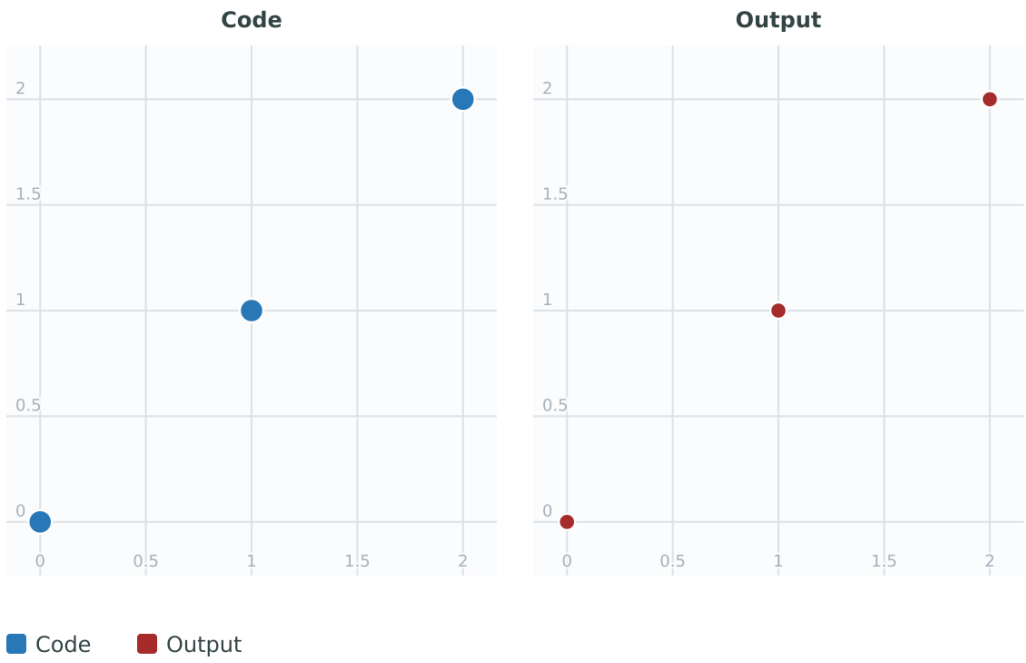
Code

```
SELECT ST_RemoveRepeatedPoints('MULTIPOINT ((1 1),(2 2),(3 3),(2 2))');
```

Output

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

Figure



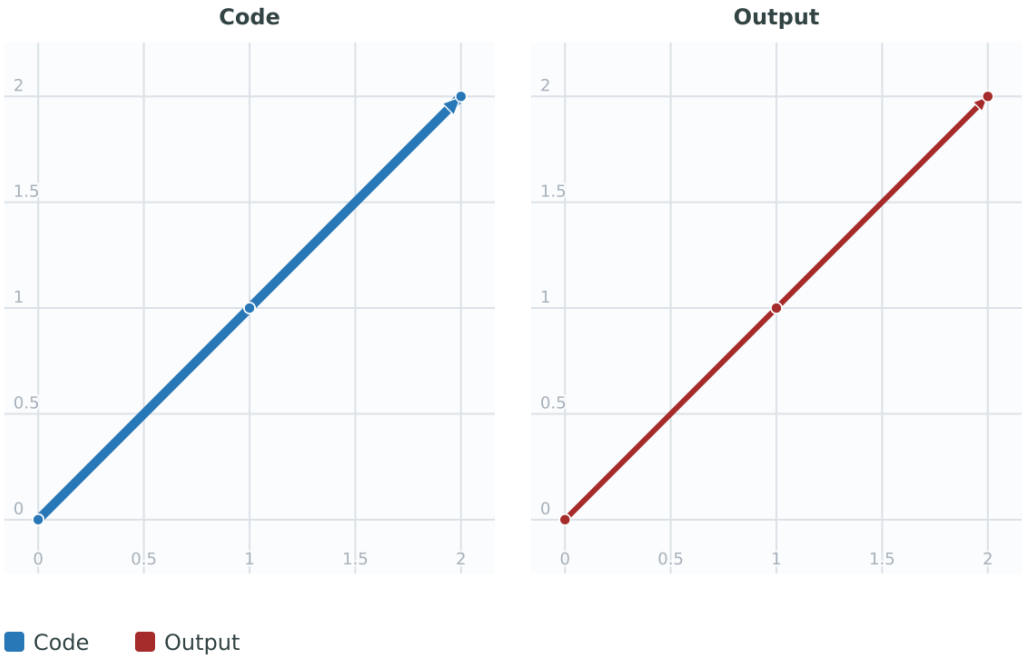
Code

```
SELECT ST_RemoveRepeatedPoints('LINESTRING (0 0,0 0,1 1,0 0,1 1,2 2)');
```

Output

```
LINESTRING(0 0,1 1,0 0,1 1,2 2)
```

Figure



示例: 集合元素是⊠独⊠理的。

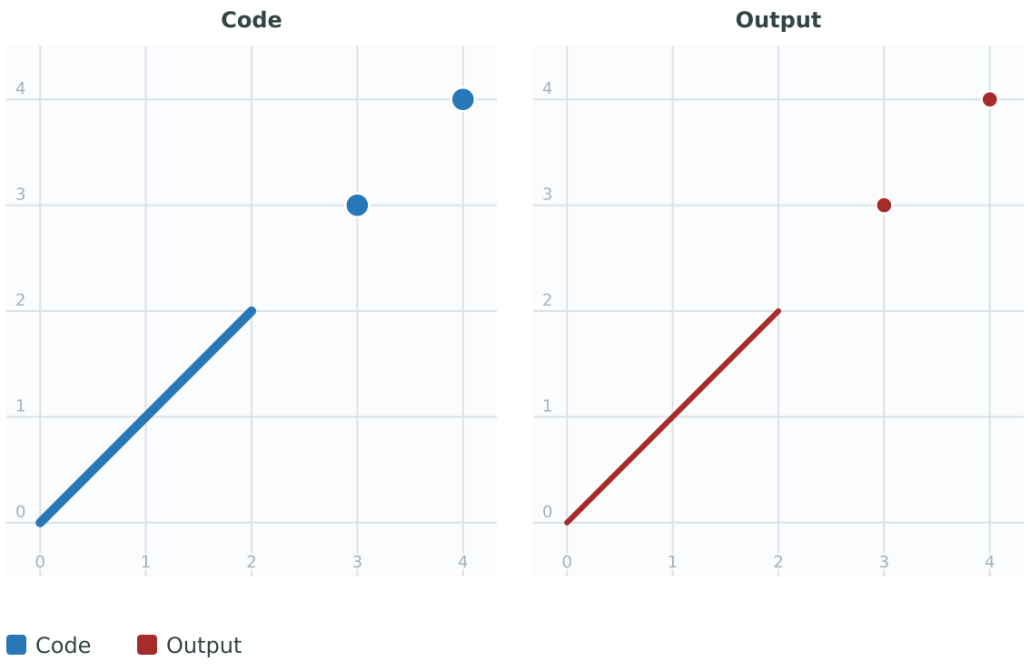
Code

```
SELECT ST_RemoveRepeatedPoints('GEOMETRYCOLLECTION (LINESTRING (1 1,2 2,2 2,3 3),POINT (4 ↩
    4),POINT (4 4),POINT (5 5))');
```

Output

```
GEOMETRYCOLLECTION(LINESTRING(1 1,2 2,3 3), POINT(4 4), POINT(4 4), POINT(5 5))
```

Figure



示例: 区除在容差距离内的重复点。

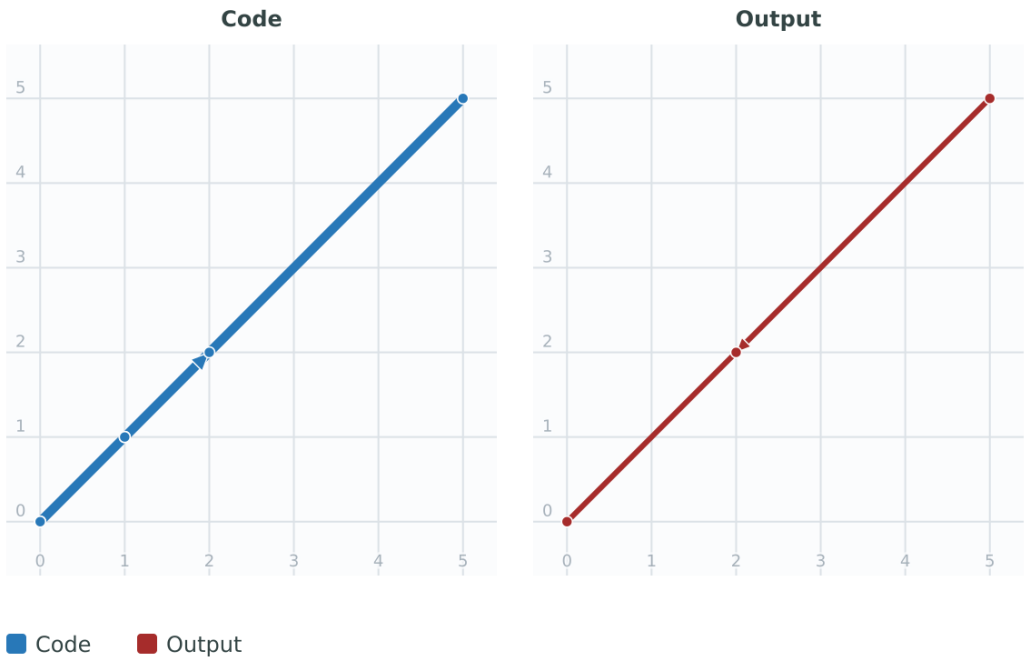
Code

```
SELECT ST_RemoveRepeatedPoints('LINESTRING (0 0,0 0,1 1,5 5,1 1,2 2)', 2);
```

Output

```
LINESTRING(0 0,5 5,2 2)
```

Figure



相关信息

ST_Simplify

7.5.26 ST_RemoveIrrelevantPointsForView

ST_RemoveIrrelevantPointsForView — 区除在指定矩形表示范区内与区制无关的几何区形点。

Synopsis

geometry **ST_RemoveIrrelevantPointsForView**(geometry geom, box2d bounds, boolean cartesian_hint = false);

描述

返回一个 geometry，移除了在指定矩形表示范区内区制区无关的点。

使用此函数可以区区在特定范区内渲染的几何区形区行快速区区理。

只有（MULTI）POLYGON 和（MULTI）LINESTRING 区型的几何区形会被区估。其他几何区形不会被更改。

区个函数与 `ST_ClipByBox2D()` 的区区如下

- 此函数通区区点区行排序而不区算新的交点，从而避免了舍入区差，并通常提高了效率，
- 返回一个具有相同或接近点数的几何区形，
- 在指定的区制范区内区生相同的区制区果，
- 可能会区致自交区象，从而使区果几何区形无效（区参区下方示例）。

如果将 `cartesian_hint` 区置区 `TRUE`，区算法会区用区外的区化，包括使用笛卡区数学，区一步区少区果中的点数。需要注意的是，如果区果坐区被投影到其他（非笛卡区）坐区系中，使用此可区参数可能会区致区制中的不自然阴影或噪点。

Warning
区于多区形来区，当前此函数无法保区区果的有效性。可以通区 ST_IsValid 来区区区种情况，并使用 ST_MakeValid 来修复。

可用性：3.5.0

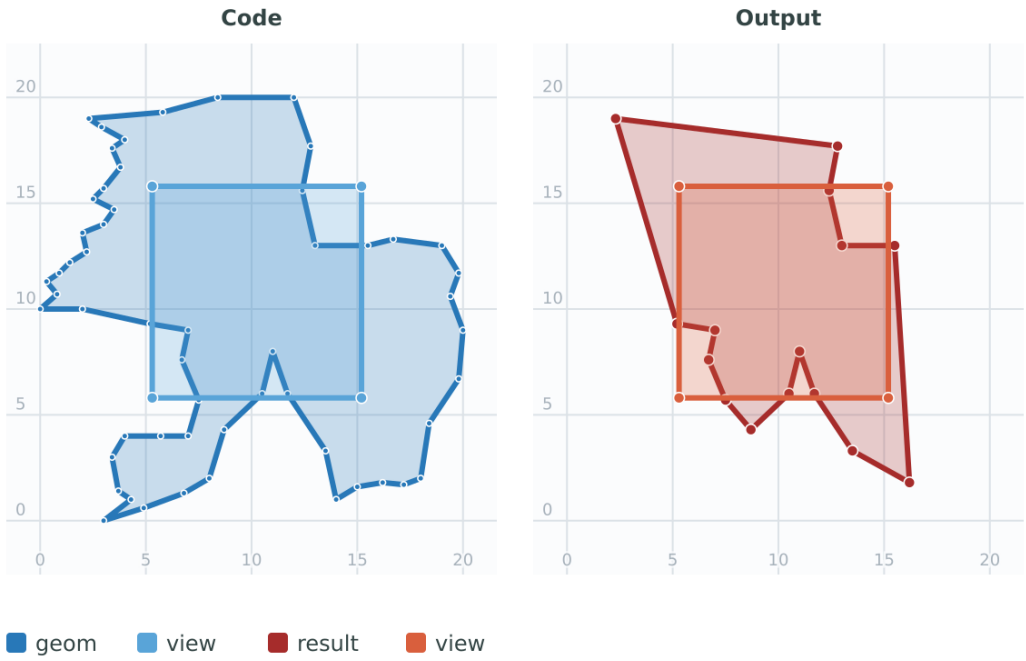
示例

The input polygon has 55 points. The result retains 15 points, while rendering the same shape inside the view envelope.

Code

```
WITH data(geom, view) AS (VALUES (
  'POLYGON((3.2 14.2,6.8 14.2,7.6 11.9,7.2 9.8,7.8 7.2,10.3 7.2,11.5 7.5,13.8 7.2,14.6 ↩
   5.9,14.2 4.8,14.8 3.2,14.6 0.9,13.2 -1.2,12.8 -3.8,12 -4.1,11 -4,9.8 -4.2,8.8 -4.8,8.3 ↩
   -2.5,6.5 0.2,5.8 2.2,5.3 0.2,3.5 -1.5,2.8 -3.8,1.6 -4.5,-0.3 -5.2,-2.2 -5.8,-0.9 ↩
   -4.8,-1.5 -4.4,-1.8 -2.8,-1.2 -1.8,0.5 -1.8,1.8 -1.8,2.3 -0.1,1.5 1.8,1.8 3.2,0 3.5,-3.2 ↩
    4.2,-5.2 4.2,-4.4 4.9,-4.9 5.5,-4.3 5.9,-3.8 6.4,-3 6.9,-3.2 7.8,-2.2 8.2,-1.7 8.9,-2.7 ↩
    9.4,-2.2 9.9,-1.4 10.9,-1.8 11.8,-1.2 12.2,-2.3 12.8,-2.9 13.2,0.6 13.5,3.2 14.2))':: ↩
   geometry,
  ST_MakeEnvelope(0.1, 0, 10, 10)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view, true) AS result,
       view AS view
FROM data;
```

Figure

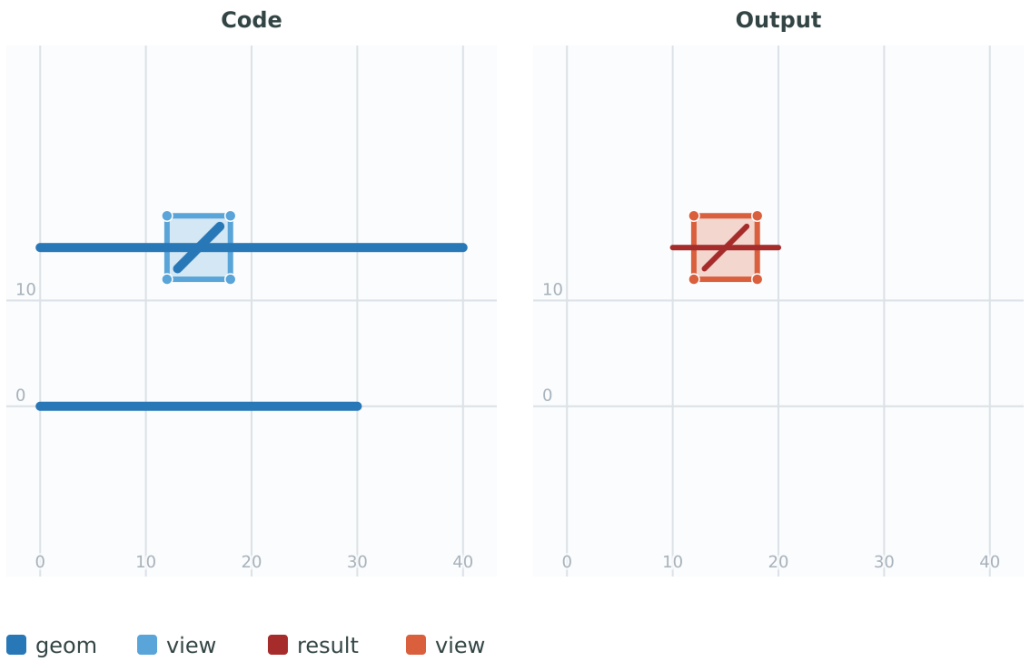


Only linework affecting the view is retained; existing vertices are reused.

Code

```
WITH data(geom, view) AS (VALUES (
  'MULTILINESTRING((0 0,10 0,20 0,30 0),(0 15,5 15,10 15,15 15,20 15,25 15,30 15,40 15),(13 ↩
     13,15 15,17 17))'::geometry,
  ST_MakeEnvelope(12, 12, 18, 18)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view, true) AS result,
       view AS view
FROM data;
```

Figure



A line with no segment relevant to the view becomes empty.

Code

```
SELECT ST_RemoveIrrelevantPointsForView('LINESTRING(0 0,10 0,20 0,30 0)',
ST_MakeEnvelope(12, 12, 18, 18), true);
```

Output

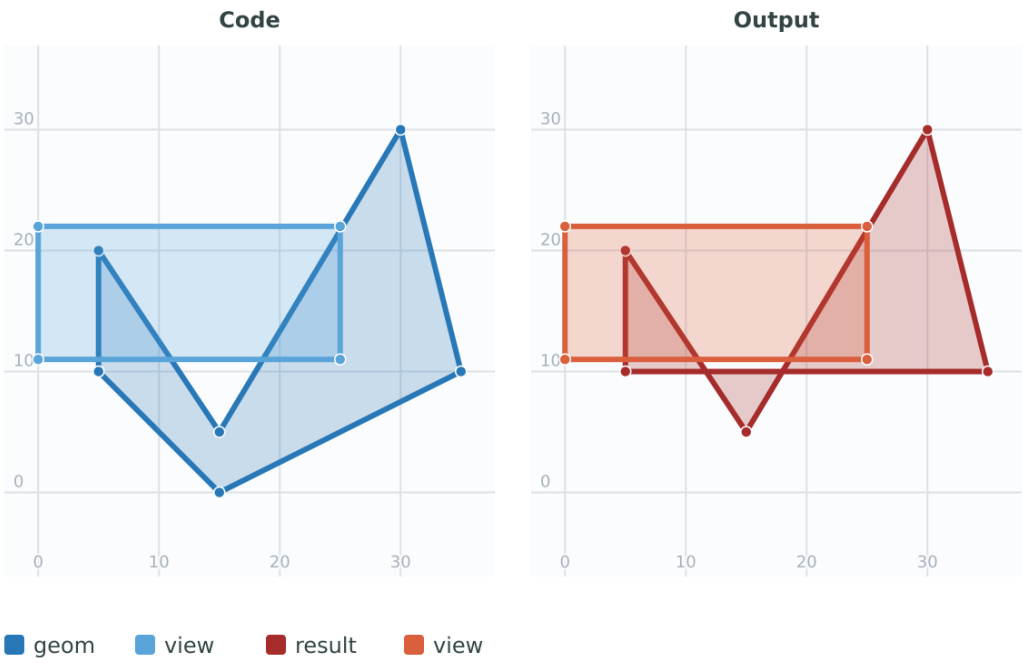
```
LINESTRING EMPTY
```

Removing vertices without computing intersections can introduce self-intersections in the result.

Code

```
WITH data(geom, view) AS (VALUES (
  'POLYGON((0 10,0 0,10 -10,30 0,25 20,10 -5,0 10))'::geometry,
  ST_MakeEnvelope(-5, 1, 20, 12)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view, true) AS result,
       view AS view
FROM data;
```

Figure

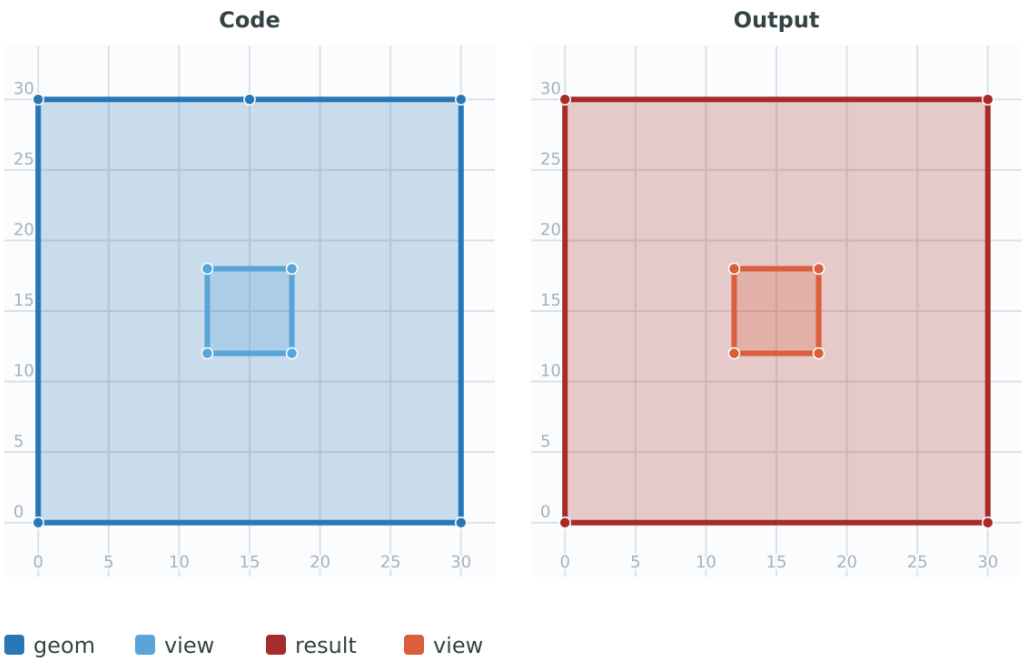


With the default *cartesian_hint*, the function removes only points known to be irrelevant to the view.

Code

```
WITH data(geom, view) AS (VALUES (
  'POLYGON((0 30,15 30,30 30,30 0,0 0,0 30))'::geometry,
  ST_MakeEnvelope(12, 12, 18, 18)
))
SELECT geom AS input_geom,
       view AS input_view,
       ST_RemoveIrrelevantPointsForView(geom, view) AS result,
       view AS view
FROM data;
```

Figure



相关信息

ST_ClipByBox2D, ST_Intersection

7.5.27 ST_RemoveSmallParts

ST_RemoveSmallParts — 区除几何区形中的所有小部分（多区形区或区串）。

Synopsis

geometry **ST_RemoveSmallParts**(geometry geom, double precision minSizeX, double precision minSizeY);

描述

返回不包含小部分（外区、内区或区串）的 geometry。
此函数可用于地区区化中的区区理步区，例如区除小区区或孔洞。
只有 (MULTI)POLYGON 和 (MULTI)LINESTRING 区型的几何体会被区估。其他几何体不会区生区化。
如果 *minSizeX* 大于 0，区会分割出区度小于 *minSizeX* 的部分。
如果 *minSizeY* 大于 0，区会分割出高度小于 *minSizeY* 的部分。
minSizeX 和 *minSizeY* 都是以几何体坐区系的区位区行度量的。
区于多区形区型，区估是区区每个区区独区行的，可能会得出以下区果之一：

- 原始几何区形，

- 所有区的区点区少的多区形 (POLYGON),
- 内区数量区少的多区形 (POLYGON)（可能区会区少区点数),
- POLYGON EMPTY，或者
- 一个多区形数量区少的 MULTIPOLYGON（可能区区少了内部区的数量或区点数)，或者
- MULTIPOLYGON EMPTY。

区于区串区型，区估是区区每个区串区行的，因此可能会得出以下区果：

- 原始几何区形,
- 一个区点数区少的 LINESTRING,
- LINESTRING EMPTY,
- 一个区串数量区少的 MULTILINESTRING（可能区区少了区点数)，或者
- MULTILINESTRING EMPTY。

可用性：3.5.0

示例

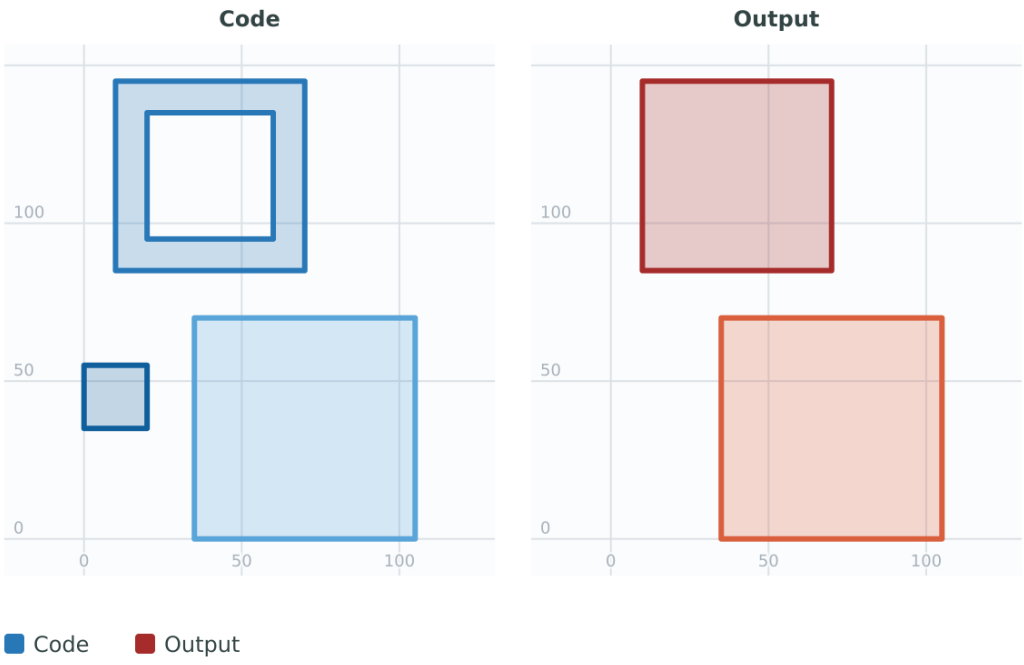
Code

```
SELECT ST_RemoveSmallParts('MULTIPOLYGON(
    ((60 160,120 160,120 220,60 220,60 160),(70 170,70 210,110 210,110 170,70 170)),
    ((85 75,155 75,155 145,85 145,85 75)),
    ((50 110,70 110,70 130,50 130,50 110)))',
    50, 50);
```

Output

```
MULTIPOLYGON(((60 160,120 160,120 220,60 220,60 160)),((85 75,155 75,155 145,85 145,85 75)))
```

Figure



Code

```
SELECT ST_RemoveSmallParts('LINESTRING(10 10,20 20)',
    50, 50);
```

Output

```
LINESTRING 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。

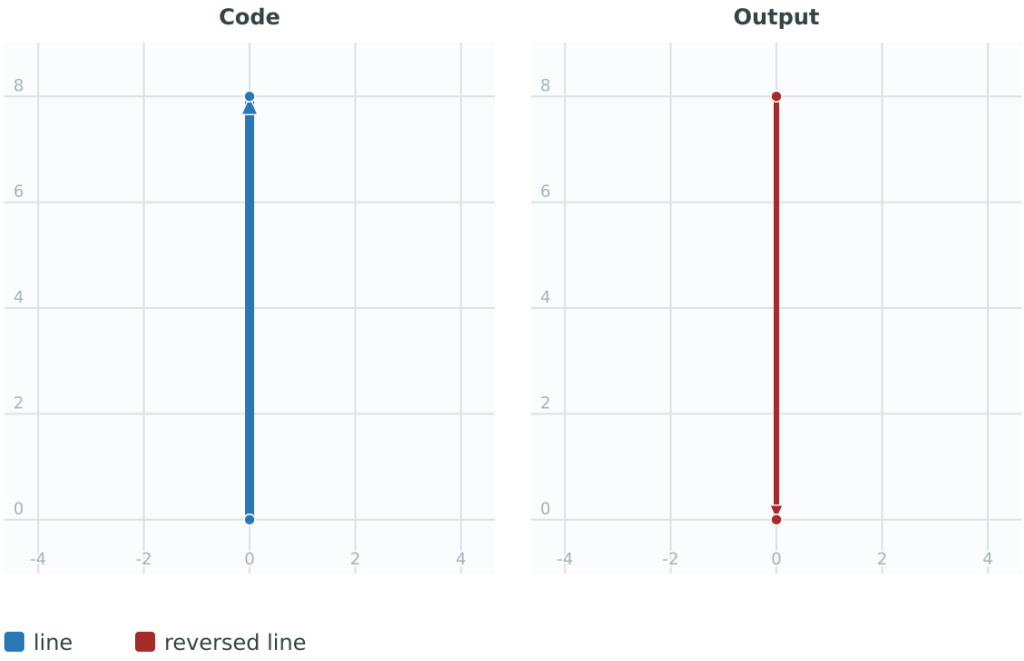
⊠函数支持多面体曲面。

示例

Code

```
WITH data(geom) AS (VALUES (
  ST_MakeLine(ST_Point(1, 2), ST_Point(1, 10))
))
SELECT geom AS input_line,
       ST_Reverse(geom) AS reversed_line
FROM data;
```

Figure



7.5.29 ST_Segmentize

ST_Segmentize — 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。

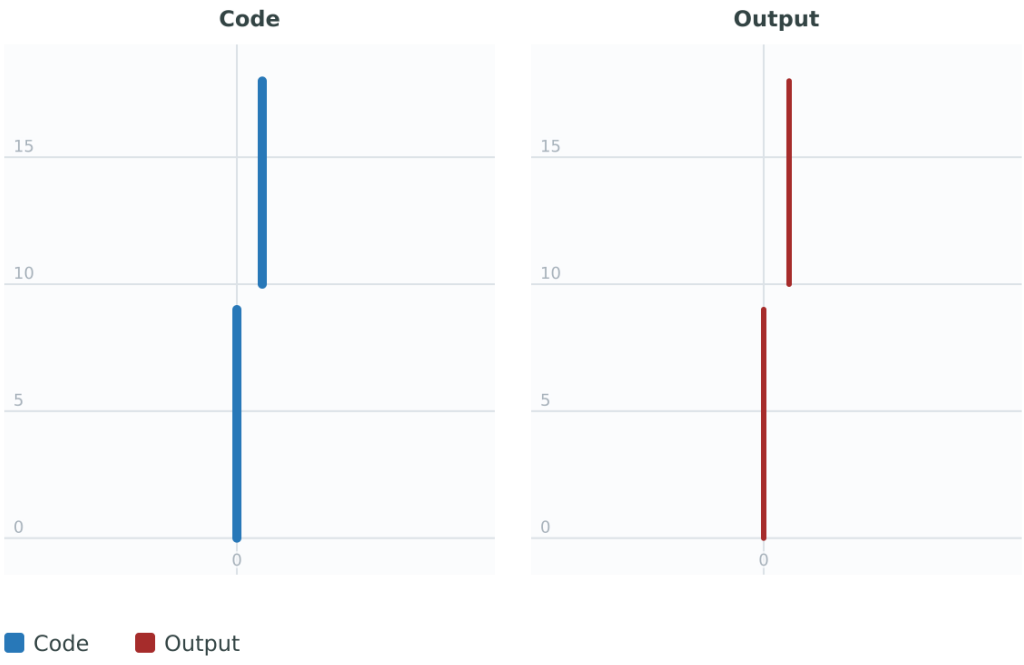
Code

```
SELECT ST_Segmentize('MULTILINESTRING((0 0,0 1,0 9),(1 10,1 18))'::geometry,
5);
```

Output

```
MULTILINESTRING((0 0,0 1,0 5,0 9),(1 10,1 14,1 18))
```

Figure



多☒形分割：

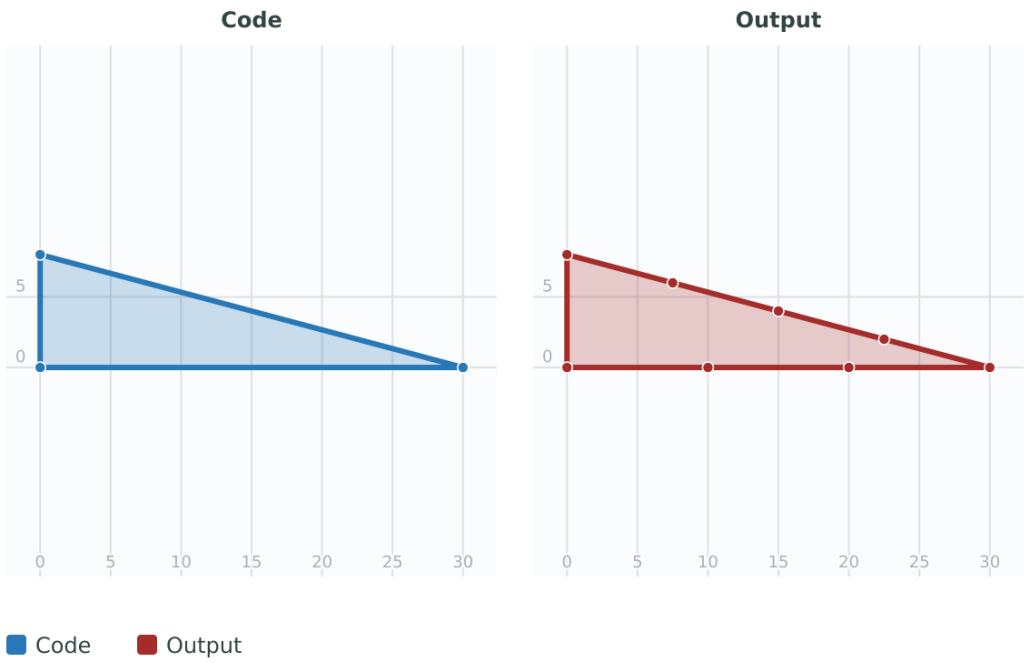
Code

```
SELECT ST_Segmentize('POLYGON((0 0,0 8,30 0,0 0))'::geometry,
        10);
```

Output

```
POLYGON((0 0,0 8,7.5 6,15 4,22.5 2,30 0,20 0,10 0,0 0))
```

Figure



☒地理☒☒行分段，最大分段☒度☒ 2000 公里。沿着☒接端点的大☒弧添加☒点。

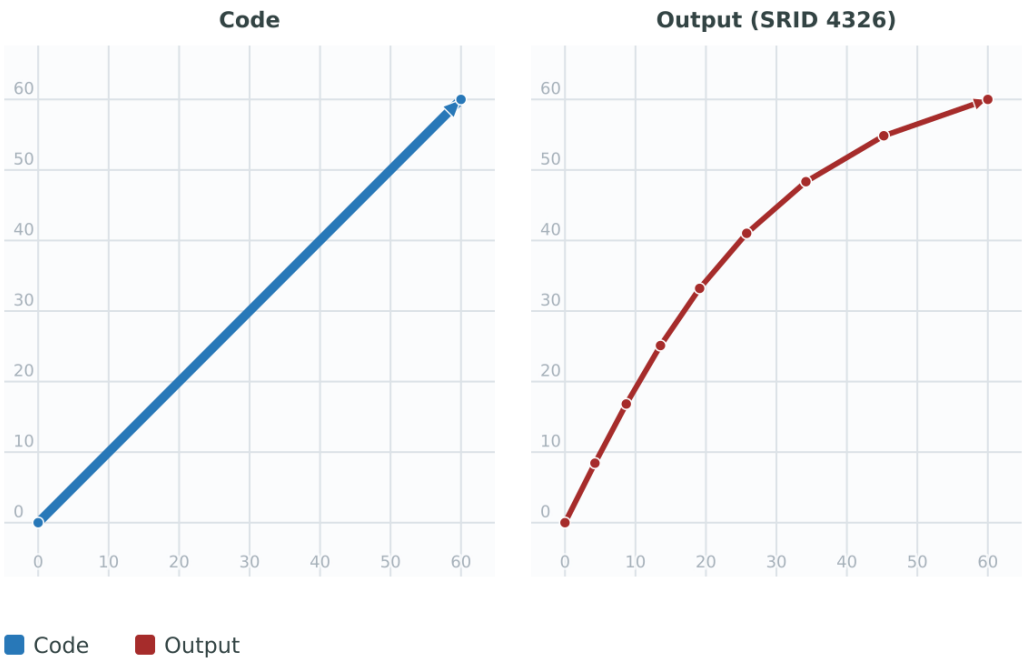
Code

```
SELECT ST_Segmentize(('LINESTRING (0 0,60 60)'::geography), 2000000);
```

Output

```
LINESTRING(0 0,4.252632294621186 8.43596525986862,8.69579947419404 16.824093489701564,13.550465473227048 ↩
    25.107950473646188,19.1066053508691 33.21091076089908,25.779290201459894 41.01711439406505,34.188839517966954 ↩
    48.337222885886,45.238153936612264 54.84733442373889,60 60)
```

Figure



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.

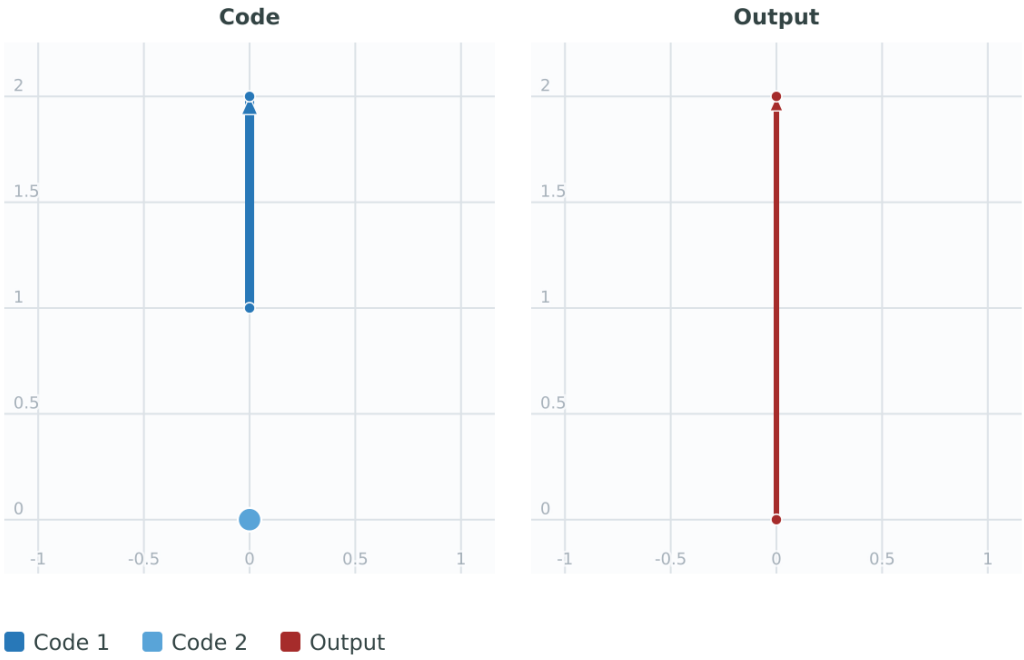
Code

```
SELECT ST_SetPoint('LINESTRING(-1 2,-1 3)', 0, 'POINT(-1 1)');
```

Output

```
LINESTRING(-1 1,-1 3)
```

Figure



Change last point in a line string (let's play with 3D linestring this time)

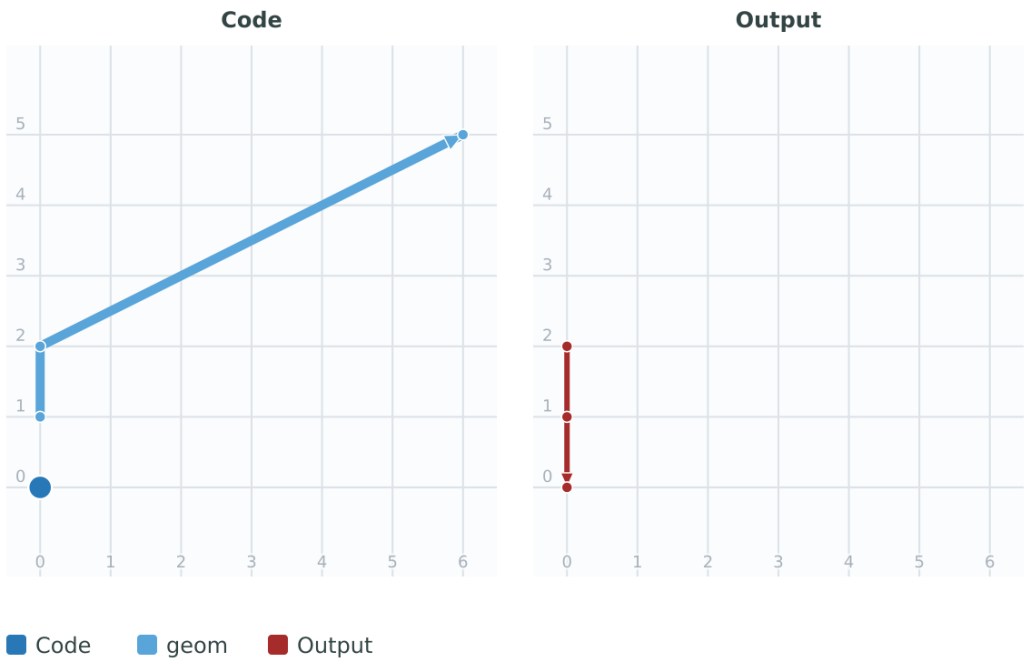
Code

```
SELECT ST_SetPoint(foo.geom, ST_NumPoints(foo.geom) - 1, 'POINT(-1 1 3)')
FROM (SELECT 'LINESTRING(-1 2 3,-1 3 4,5 6 7)'::geometry As geom) As foo;
```

Output

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

Figure



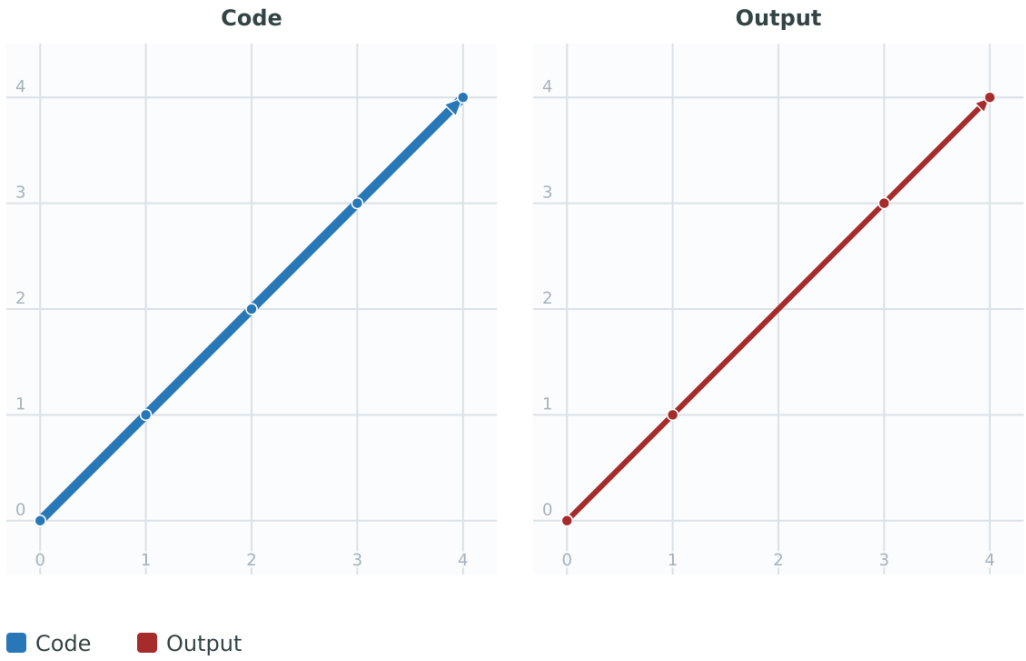
Code

```
SELECT ST_SetPoint(g, -3, p)
FROM ST_GeomFromText('LINESTRING(0 0,1 1,2 2,3 3,4 4)') AS g
    CROSS JOIN ST_PointN(g, 1) as p;
```

Output

```
LINESTRING(0 0,1 1,0 0,3 3,4 4)
```

Figure



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

This example performs a forward longitude shift on a single point.

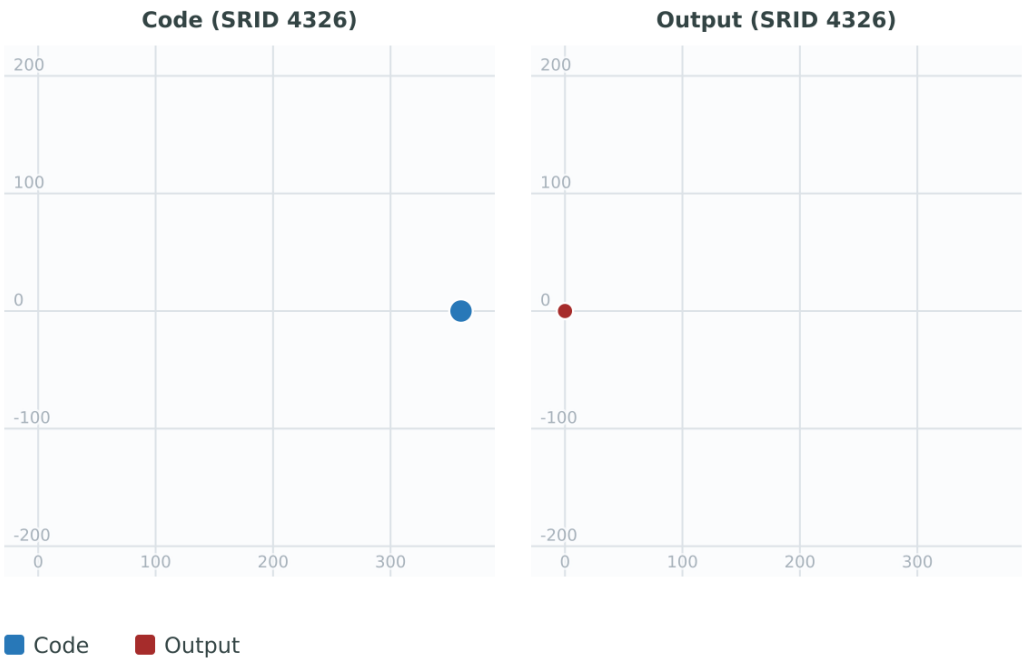
Code

```
SELECT ST_ShiftLongitude('SRID=4326;POINT(270 0)'::geometry)
```

Output

```
POINT(-90 0)
```

Figure



This example performs a reverse longitude shift on a single point.

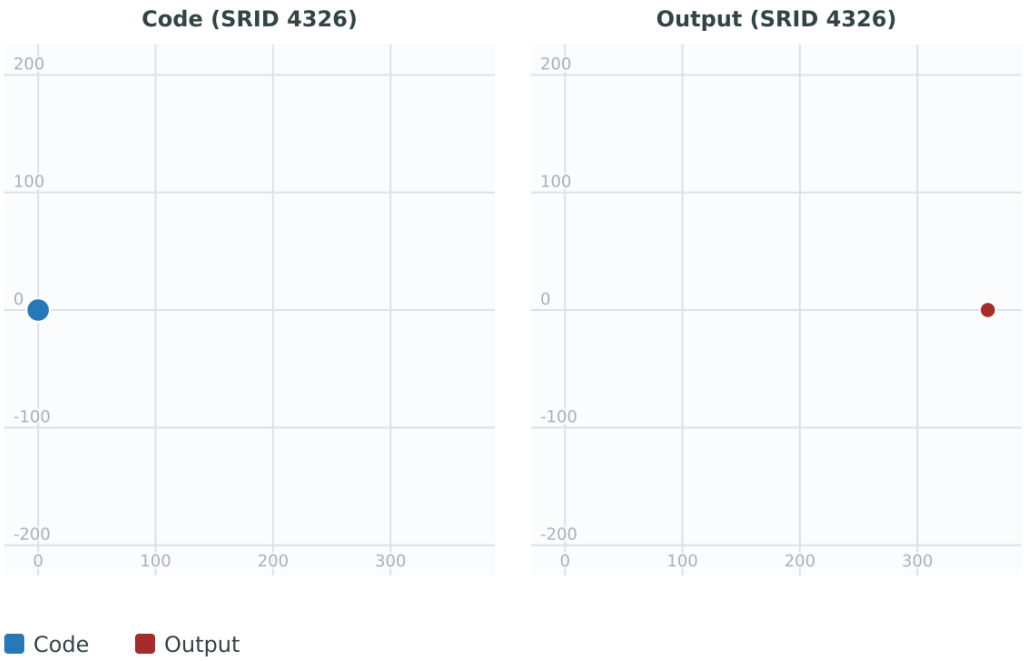
Code

```
SELECT ST_ShiftLongitude('SRID=4326;POINT(-90 0)'::geometry)
```

Output

```
POINT(270 0)
```

Figure



This example shows that LineStrings are shifted only where coordinates require it.

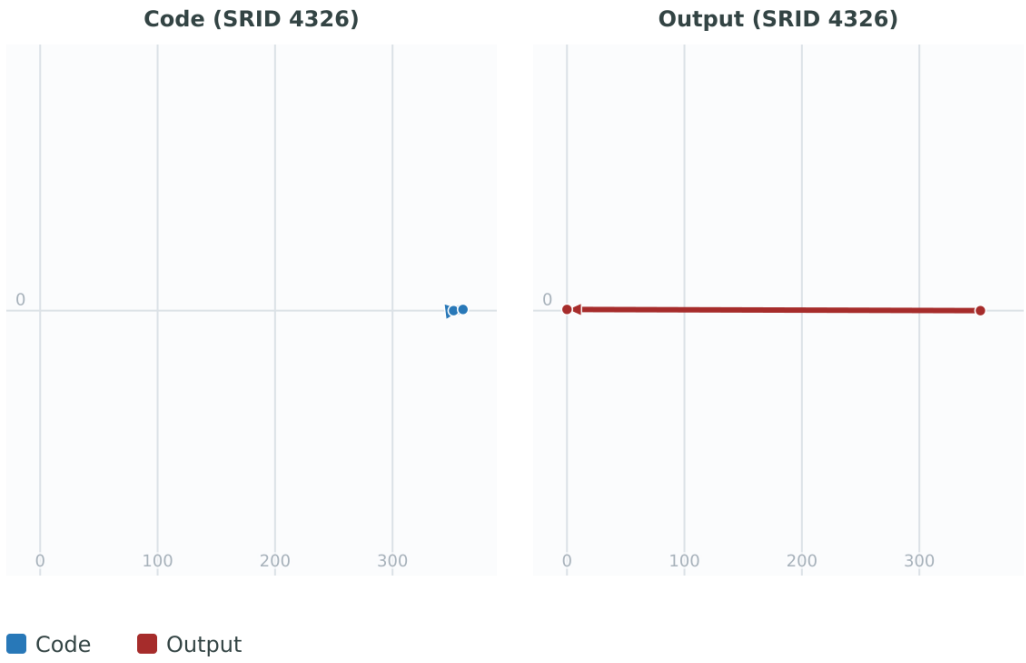
Code

```
SELECT ST_ShiftLongitude('SRID=4326;LINESTRING(174 12,182 13)'::geometry)
```

Output

```
LINESTRING(174 12,-178 13)
```

Figure



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 whose bounding boxes fall completely to the left of `x = 0` by +360.

Code

```
SELECT ST_WrapX(geom, 0, 360);
```

Move all components whose bounding boxes fall completely to the left of `x = -30` by +360.

Code

```
SELECT ST_WrapX(geom, -30, 360);
```

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 a geometry column to a precision grid of 10^-3.

Code

```
UPDATE mytable
   SET geom = ST_SnapToGrid(geom, 0.001);
```

Display how the same grid size rounds a LineString.

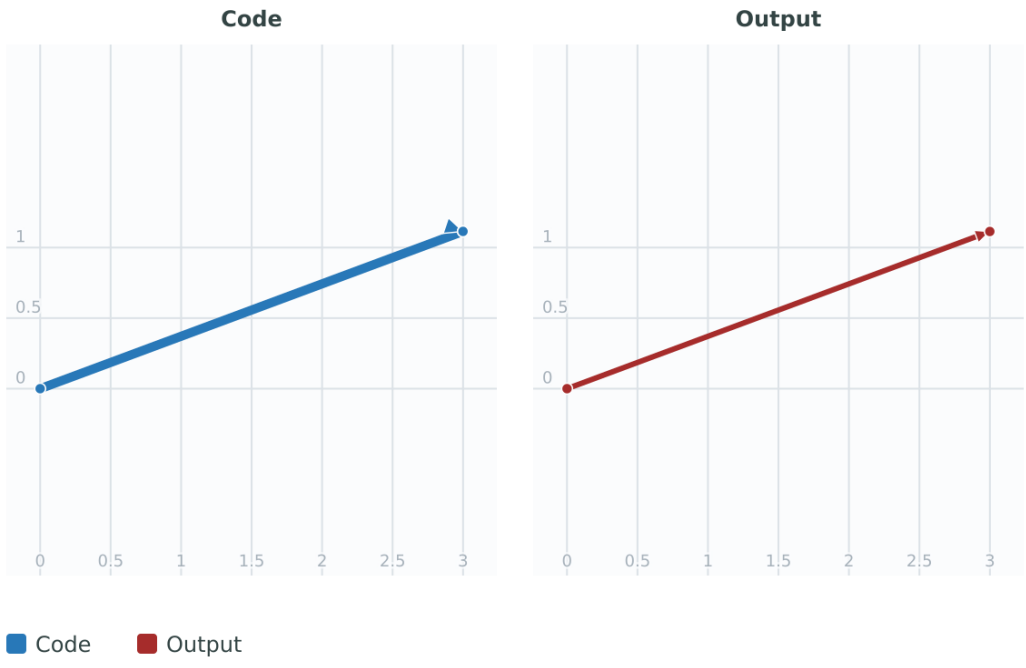
Code

```
SELECT ST_SnapToGrid('LINESTRING(1.1115678 2.123,4.111111 3.2374897,4.11112 3.23748667)':: ↩
    geometry,
        0.001);
```

Output

```
LINESTRING(1.112 2.123,4.111 3.237)
```

Figure



This example snaps a 4D geometry.

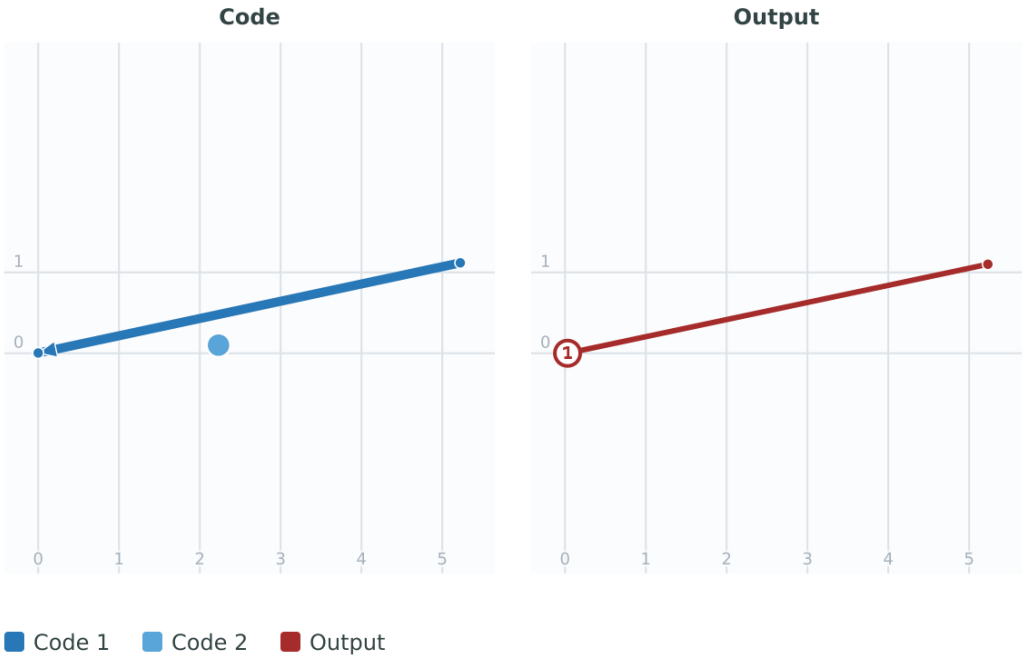
Code

```
SELECT ST_SnapToGrid('LINESTRING(-1.1115678 2.123 2.3456 1.11111,
        4.111111 3.2374897 3.1234 1.1111,-1.11111112 2.123 2.3456 1.1111112)'::geometry,
        'POINT(1.12 2.22 3.2 4.4444)',
        0.1, 0.1, 0.1, 0.01);
```

Output

```
LINESTRING(-1.08 2.12 2.3 1.1144,4.12 3.22 3.1 1.1144,-1.08 2.12 2.3 1.1144)
```

Figure



With a 4D geometry - the ST_SnapToGrid(geom,size) only touches x and y coords but keeps m and z the same.

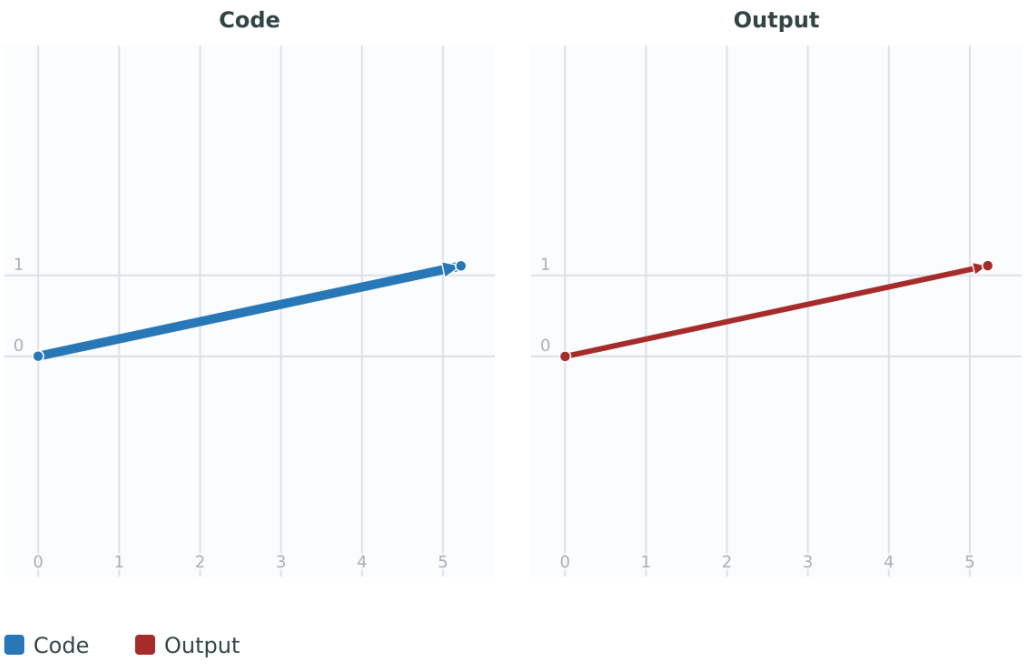
Code

```
SELECT ST_SnapToGrid('LINESTRING(-1.1115678 2.123 3 2.3456,
        4.111111 3.2374897 3.1234 1.1111)'::geometry,
       0.01);
```

Output

```
LINESTRING(-1.11 2.12 3 2.3456,4.11 3.24 3.1234 1.1111)
```

Figure



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

A MultiPolygon snapped to a LineString at 1.01 times their distance.

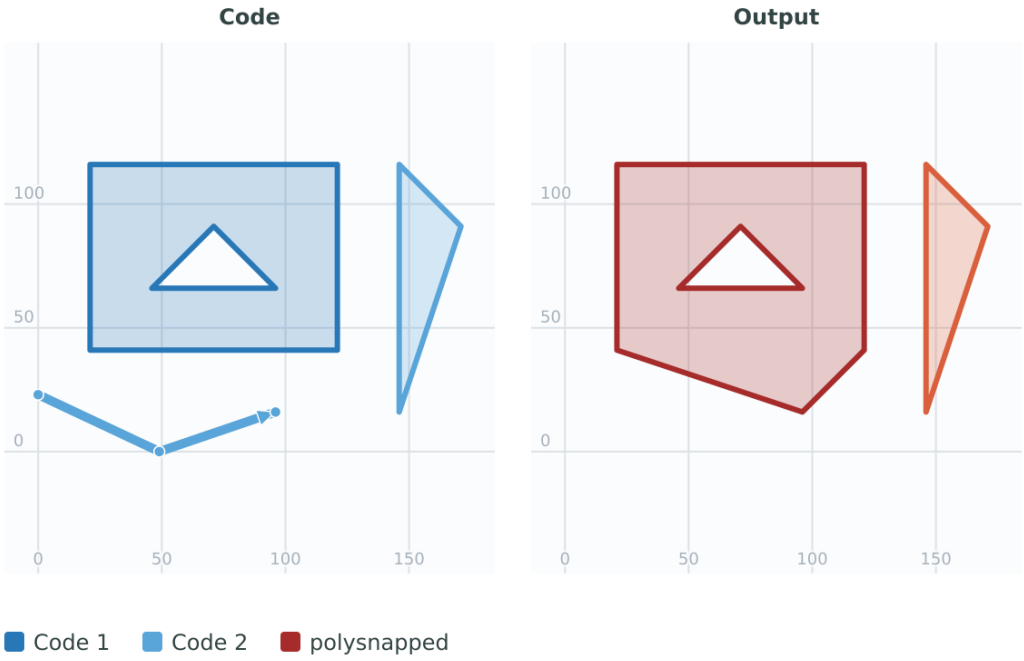
Code

```
SELECT ST_Snap(poly, line, ST_Distance(poly, line)*1.01) AS polysnapped
FROM (SELECT
   ST_GeomFromText('MULTIPOLYGON(
     ((26 125,26 200,126 200,126 125,26 125 ),
      ( 51 150,101 150,76 175,51 150 )),
      (( 151 100,151 200,176 175,151 100 )))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
MULTIPOLYGON(((26 125,26 200,126 200,126 125,101 100,26 125),
(51 150,101 150,76 175,51 150)),((151 100,151 200,176 175,151 100)))
```

Figure



A MultiPolygon snapped to a LineString at 1.25 times their distance.

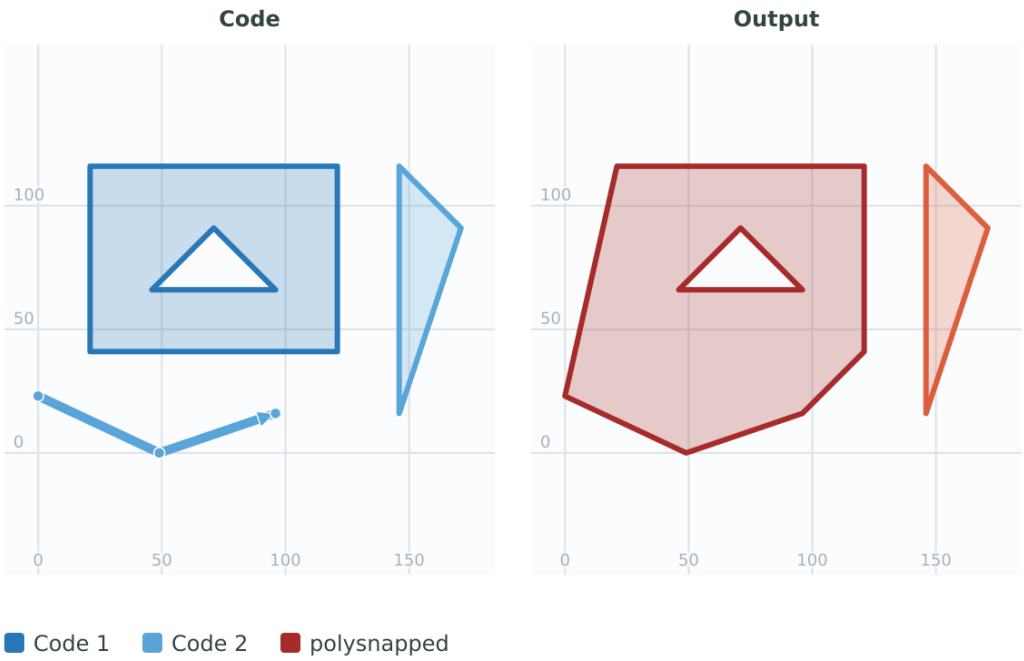
Code

```
SELECT ST_Snap(poly, line, ST_Distance(poly, line)*1.25) AS polysnapped
FROM (SELECT
  ST_GeomFromText('MULTIPOLYGON(
    (( 26 125,26 200,126 200,126 125,26 125 ),
      ( 51 150,101 150,76 175,51 150 )),
      (( 151 100,151 200,176 175,151 100 )))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
MULTIPOLYGON(((5 107,26 200,126 200,126 125,101 100,54 84,5 107),
(51 150,101 150,76 175,51 150)),((151 100,151 200,176 175,151 100)))
```

Figure



The LineString snapped to the original MultiPolygon at 1.01 times their distance.

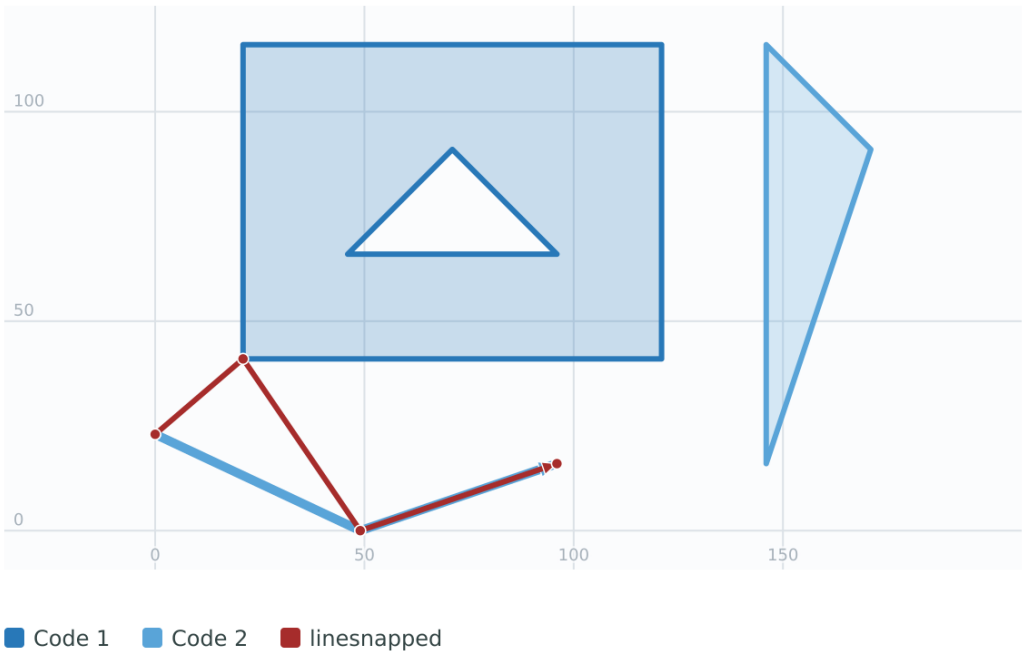
Code

```
SELECT ST_Snap(line, poly, ST_Distance(poly, line)*1.01) AS linesnapped
FROM (SELECT
  ST_GeomFromText('MULTIPOLYGON(
     ((26 125,26 200,126 200,126 125,26 125),
      (51 150,101 150,76 175,51 150 )),
      ((151 100,151 200,176 175,151 100)))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
LINESTRING(5 107,26 125,54 84,101 100)
```

Figure



The LineString snapped to the original MultiPolygon at 1.25 times their distance.

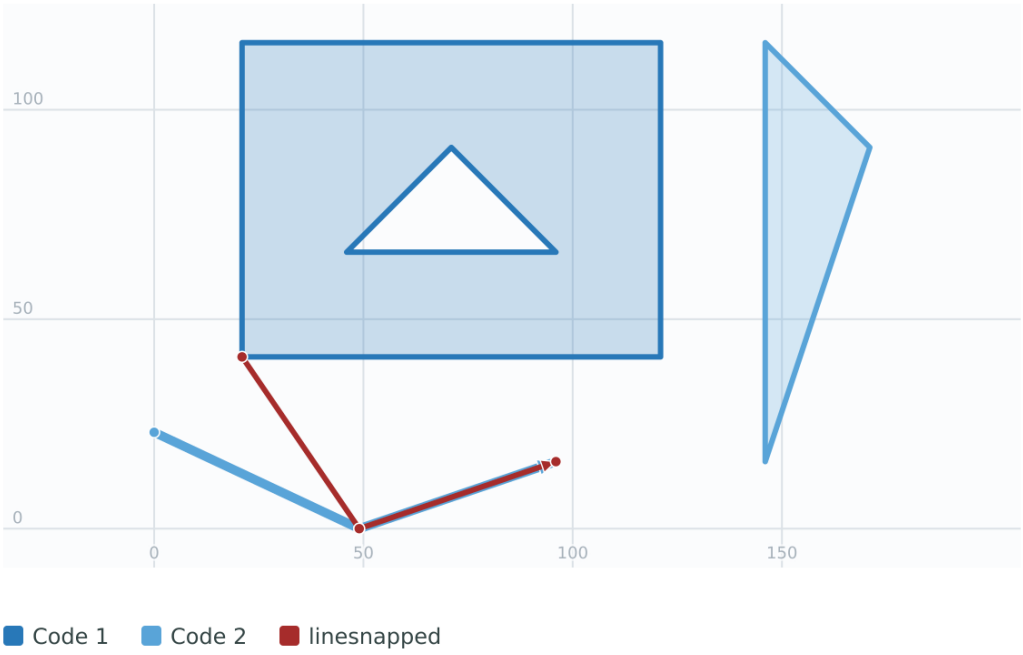
Code

```
SELECT ST_Snap(line, poly, ST_Distance(poly, line)*1.25) AS linesnapped
FROM (SELECT
  ST_GeomFromText('MULTIPOLYGON(
     (( 26 125,26 200,126 200,126 125,26 125 ),
      (51 150,101 150,76 175,51 150 )),
      ((151 100,151 200,176 175,151 100 )))') As poly,
       ST_GeomFromText('LINESTRING (5 107,54 84,101 100)') As line
    ) As foo;
```

Output

```
LINESTRING(26 125,54 84,101 100)
```

Figure



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

This example scales the M value by 2.

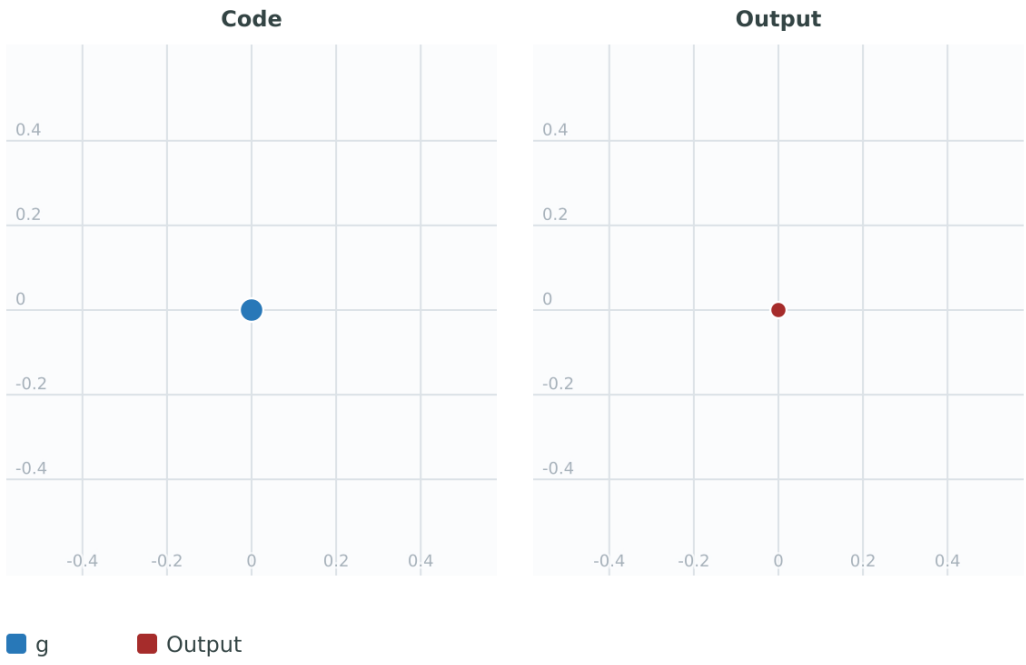
Code

```
SELECT ST_SwapOrdinates(ST_Scale(ST_SwapOrdinates(g, 'xm'),
      2, 1
    ),
  'xm') FROM ( SELECT 'POINT ZM (0 0 0 2)'::geometry g ) foo;
```

Output

```
POINT ZM (0 0 0 4)
```

Figure



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

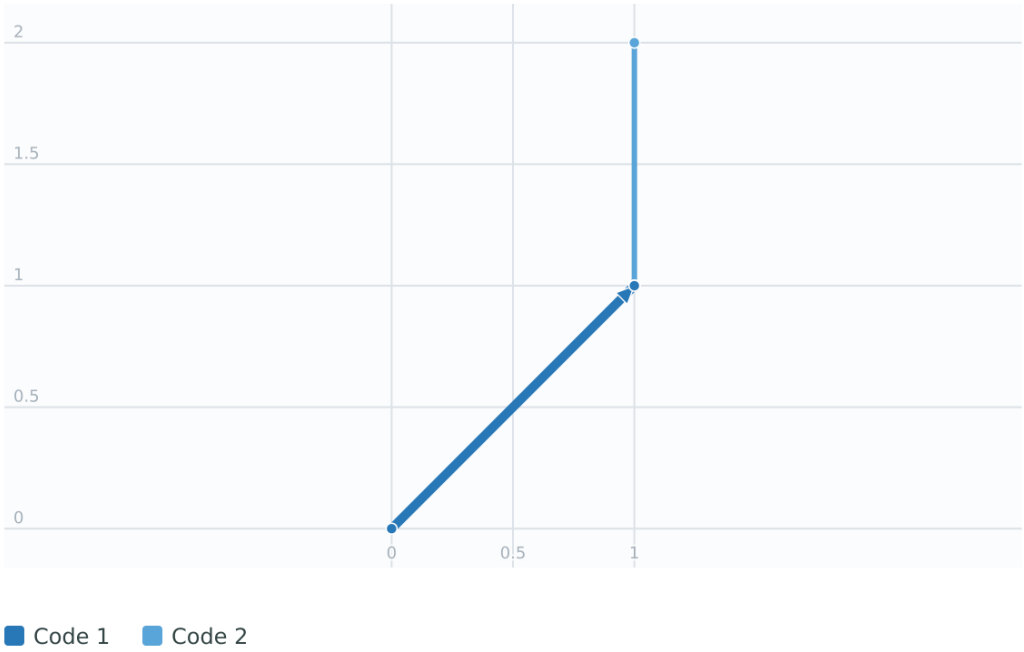
Code

```
SELECT ST_IsValid('LINESTRING(0 0,1 1)') As good_line,
ST_IsValid('POLYGON((0 0,1 1,1 2,1 1,0 0))') As bad_poly;
```

Output

```
NOTICE:  Self-intersection at or near point 0 0
 good_line | bad_poly
-----------+----------
 t         | f
```

Figure



Find tables with invalid geometries in registered geometry columns.

Code

```
CREATE TEMP TABLE invalid_geometry_columns (
    table_name text,
    column_name text,
    invalid_count bigint
);

DO $$
DECLARE
    rec record;
    invalid_count bigint;
BEGIN
    FOR rec IN
        SELECT f_table_schema, f_table_name, f_geometry_column
        FROM geometry_columns
        ORDER BY f_table_schema, f_table_name, f_geometry_column
    LOOP
        EXECUTE format(
            'SELECT count(*) FROM %I.%I WHERE NOT ST_IsValid(%I, 0)',
            rec.f_table_schema,
            rec.f_table_name,
            rec.f_geometry_column
        ) INTO invalid_count;

        IF invalid_count
> 0 THEN
            INSERT INTO invalid_geometry_columns
            VALUES (
                format('%I.%I', rec.f_table_schema, rec.f_table_name),
                rec.f_geometry_column,
                invalid_count
            );
        END IF;
    END LOOP;
```

```
END;
$$;

SELECT * FROM invalid_geometry_columns;
```

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

This example returns the first three rejects from a successful quintuplet experiment.

Code

```
SELECT gid, reason(ST_IsValidDetail(geom)), location(ST_IsValidDetail(geom)) as location
FROM
(SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), array_agg(f.line)) As geom, gid
FROM (SELECT ST_Buffer(ST_Point(x1*10,y1), z1) As buff, x1*10 + y1*100 + z1*1000 As gid
    FROM generate_series(-4,6) x1
    CROSS JOIN generate_series(2,5) y1
    CROSS JOIN generate_series(1,8) z1
    WHERE x1
> y1*0.5 AND z1 < x1*y1) As e
    INNER JOIN (SELECT ST_Translate(ST_ExteriorRing(ST_Buffer(ST_Point(x1*10,y1), z1)),y1 ↩
    *1, z1*2) As line
    FROM generate_series(-3,6) x1
    CROSS JOIN generate_series(2,5) y1
```

```
    CROSS JOIN generate_series(1,10) z1
    WHERE x1
> y1*0.75 AND z1 < x1*y1) As f
ON (ST_Area(e.buff)
> 78 AND ST_Contains(e.buff, f.line))
GROUP BY gid, e.buff) As quintuplet_experiment
WHERE ST_IsValid(geom) = false
ORDER BY gid
LIMIT 3;
```

Output

```
 gid  |      reason       |  location
------+-------------------+-------------
 5330 | Self-intersection | POINT(32 5)
 5340 | Self-intersection | POINT(42 5)
 5350 | Self-intersection | POINT(52 5)
```

This example reports a valid LineString.

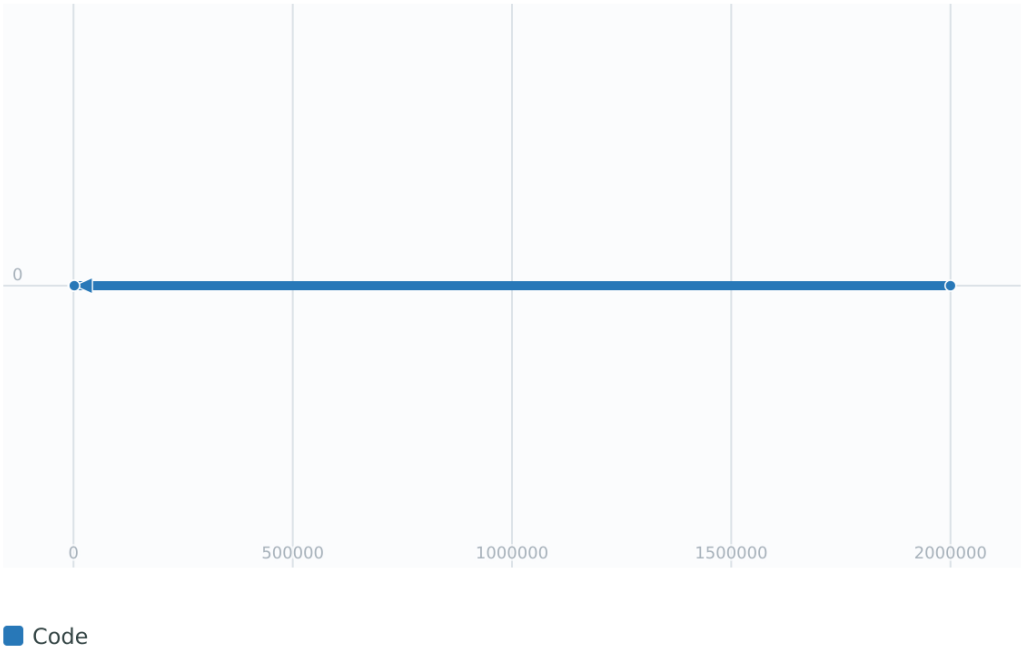
Code

```
SELECT * FROM ST_IsValidDetail('LINESTRING(220227 150406,2220227 150407,222020 150410)');
```

Output

```
 valid | reason | location
-------+--------+----------
 t     |        |
```

Figure



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。

可用性：1.4

可用性：2.0 版本正在接受☒☒。

示例

This example uses an invalid bow-tie polygon.

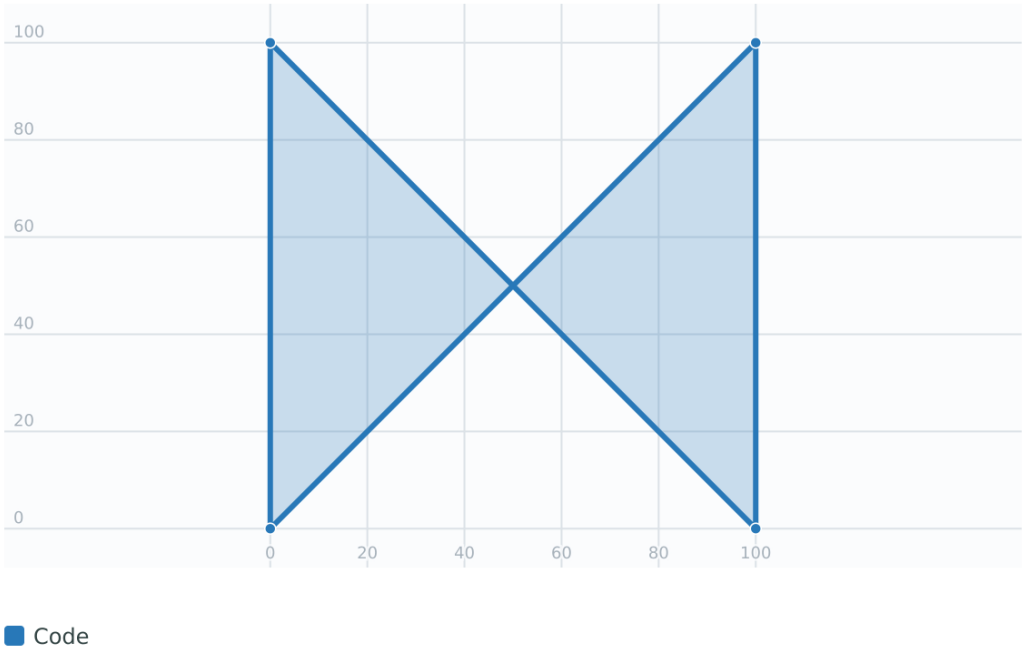
Code

```
SELECT ST_IsValidReason(
    'POLYGON ((100 200,100 100,200 200,
     200 100,100 200))'::geometry) as validity_info;
```

Output

```
Self-intersection[150 150]
```

Figure



This example returns the first three rejects from a successful quintuplet experiment.

Code

```
SELECT gid, ST_IsValidReason(geom) as validity_info
FROM
(SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), array_agg(f.line)) As geom, gid
FROM (SELECT ST_Buffer(ST_Point(x1*10,y1), z1) As buff, x1*10 + y1*100 + z1*1000 As gid
    FROM generate_series(-4,6) x1
    CROSS JOIN generate_series(2,5) y1
    CROSS JOIN generate_series(1,8) z1
    WHERE x1
> y1*0.5 AND z1 < x1*y1) As e
    INNER JOIN (SELECT ST_Translate(ST_ExteriorRing(ST_Buffer(ST_Point(x1*10,y1), z1)),y1 ↩
    *1, z1*2) As line
    FROM generate_series(-3,6) x1
    CROSS JOIN generate_series(2,5) y1
    CROSS JOIN generate_series(1,10) z1
    WHERE x1
> y1*0.75 AND z1 < x1*y1) As f
ON (ST_Area(e.buff)
> 78 AND ST_Contains(e.buff, f.line))
GROUP BY gid, e.buff) As quintuplet_experiment
WHERE ST_IsValid(geom) = false
ORDER BY gid
LIMIT 3;
```

Output

```
 gid  |      validity_info
------+-------------------------
 5330 | Self-intersection [32 5]
 5340 | Self-intersection [42 5]
 5350 | Self-intersection [52 5]
```

Simple example.

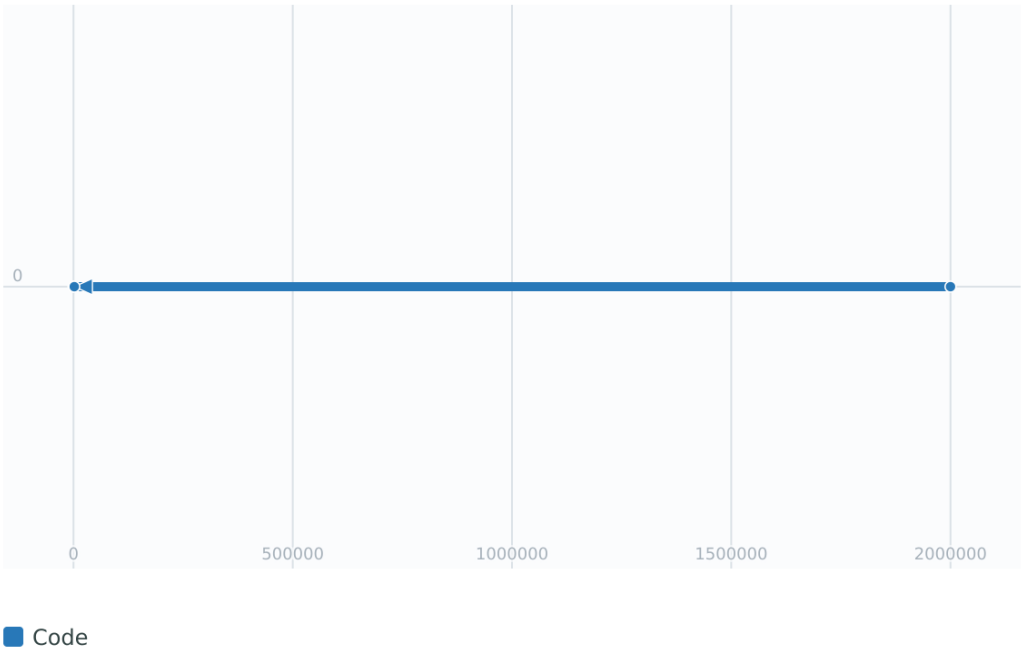
Code

```
SELECT ST_IsValidReason('LINESTRING(220227 150406,2220227 150407,222020 150410)');
```

Output

```
 st_isvalidreason
------------------
 Valid Geometry
```

Figure



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”是默⍰算法，其⍰理流程是：先提取所有⍰，再⍰⍰些⍰⍰行⍰点化（noding），然后基于⍰些⍰生成有效的几何⍰出。⍰了保⍰不⍰失任何⍰点，⍰出⍰果可能会形成复⍰的几何集合。
- “structure”是一种算法，它会区分内⍰和外⍰：先⍰外⍰做并集⍰建新几何，再用所有内⍰做差。⍰果通常更直⍰，折⍰的⍰会在⍰出中被⍰弃。

“keepcollapsed”⍰⍰⍰⍰ structure 算法有效，取⍰⍰ true 或 false。

- “keepcollapsed”默⍰⍰ false，表示当几何⍰件被折⍰⍰低⍰度⍰（例如只有一个点的 LineString），⍰⍰件会被⍰弃。
- “keepcollapsed”⍰置⍰ true 表示即使几何⍰件被折⍰⍰低⍰度，也会被保留。

⍰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。

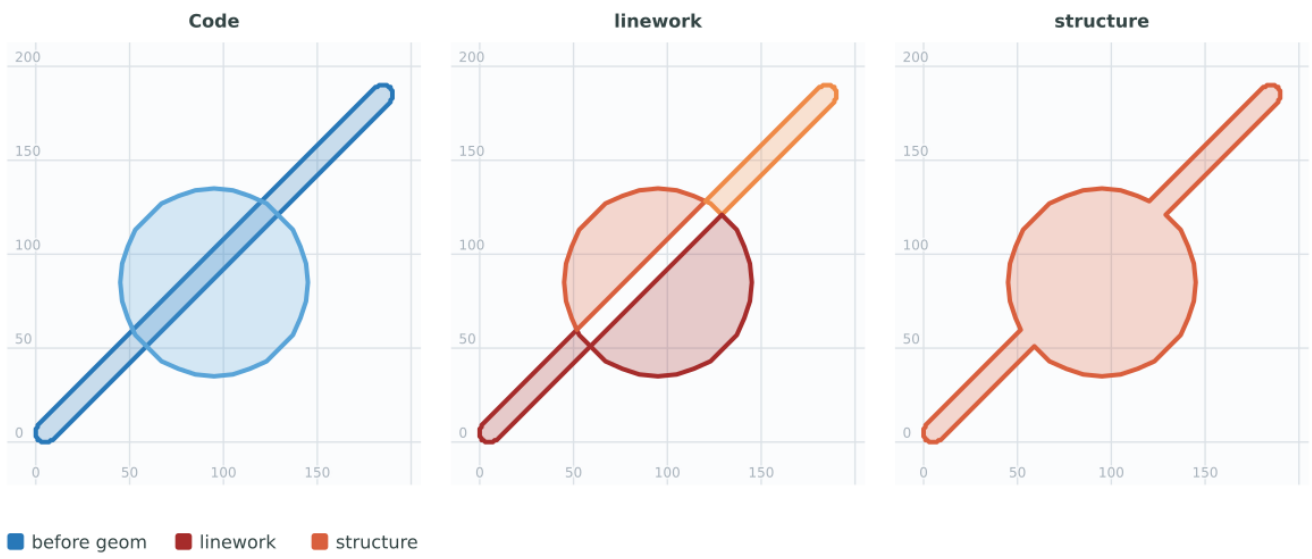
示例

Make valid a MultiPolygon of two overlapping polygons using the linework and structure methods.

Code

```
SELECT ST_MakeValid(before_geom) AS linework,
  ST_MakeValid(before_geom, 'method=structure') AS structure
FROM (SELECT 'MULTIPOLYGON(((186 194,187 194,188 195,189 195,190 195,
191 195,192 195,193 194,194 194,194 193,195 192,195 191,
195 190,195 189,195 188,194 187,194 186,14 6,13 6,12 5,11 5,
10 5,9 5,8 5,7 6,6 6,6 7,5 8,5 9,5 10,5 11,5 12,6 13,6 14,186 194)),
((150 90,149 80,146 71,142 62,135 55,128 48,119 44,110 41,100 40,
90 41,81 44,72 48,65 55,58 62,54 71,51 80,50 90,51 100,
54 109,58 118,65 125,72 132,81 136,90 139,100 140,110 139,
119 136,128 132,135 125,142 118,146 109,149 100,150 90)))'::geometry AS before_geom) AS f;
```

Figure

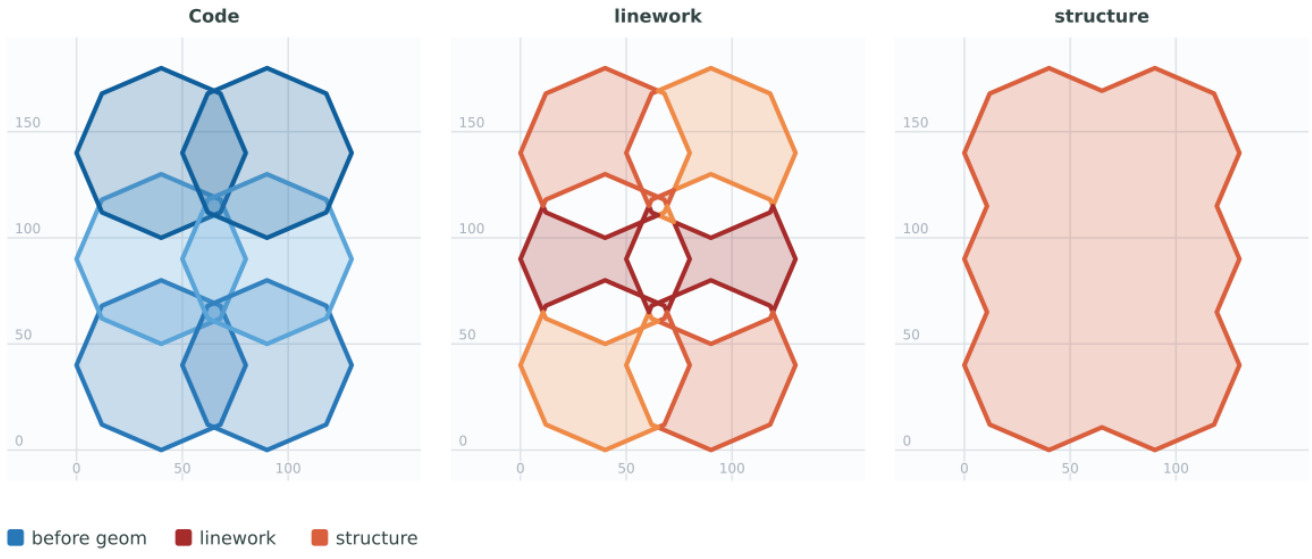


Make valid a MultiPolygon of six overlapping polygons using the linework and structure methods.

Code

```
SELECT ST_MakeValid(before_geom) AS linework,
  ST_MakeValid(before_geom, 'method=structure') AS structure
FROM (SELECT 'MULTIPOLYGON(((91 50,79 22,51 10,23 22,11 50,23 78,51 90,79 78,91 50)),
      ((91 100,79 72,51 60,23 72,11 100,23 128,51 140,79 128,91 100)),
      ((91 150,79 122,51 110,23 122,11 150,23 178,51 190,79 178,91 150)),
      ((141 50,129 22,101 10,73 22,61 50,73 78,101 90,129 78,141 50)),
      ((141 100,129 72,101 60,73 72,61 100,73 128,101 140,129 128,141 100)),
      ((141 150,129 122,101 110,73 122,61 150,73 178,101 190,129 178,141 150)))'::geometry ↩
    AS before_geom) AS c;
```

Figure



Collapsed Geometry Examples

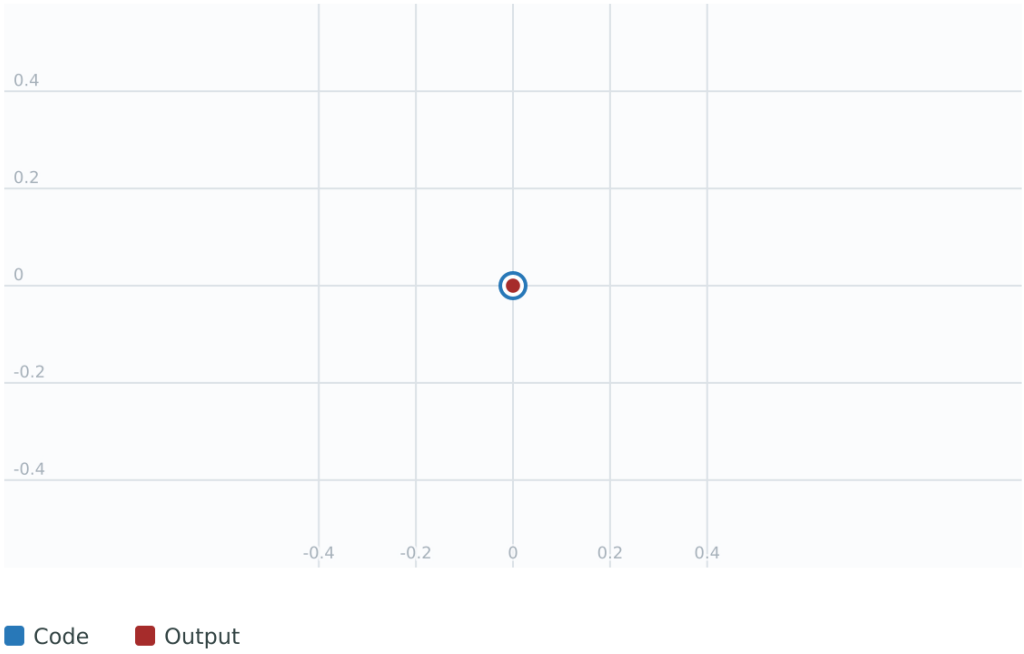
Code

```
SELECT ST_MakeValid(
'LINESTRING(0 0,0 0)',
'method=structure keepcollapsed=true'
);
```

Output

```
POINT(0 0)
```

Figure



Code

```
SELECT ST_MakeValid(
'LINESTRING(0 0,0 0)',
'method=structure keepcollapsed=false'
);
```

Output

```
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

This example transforms in the inverse direction.

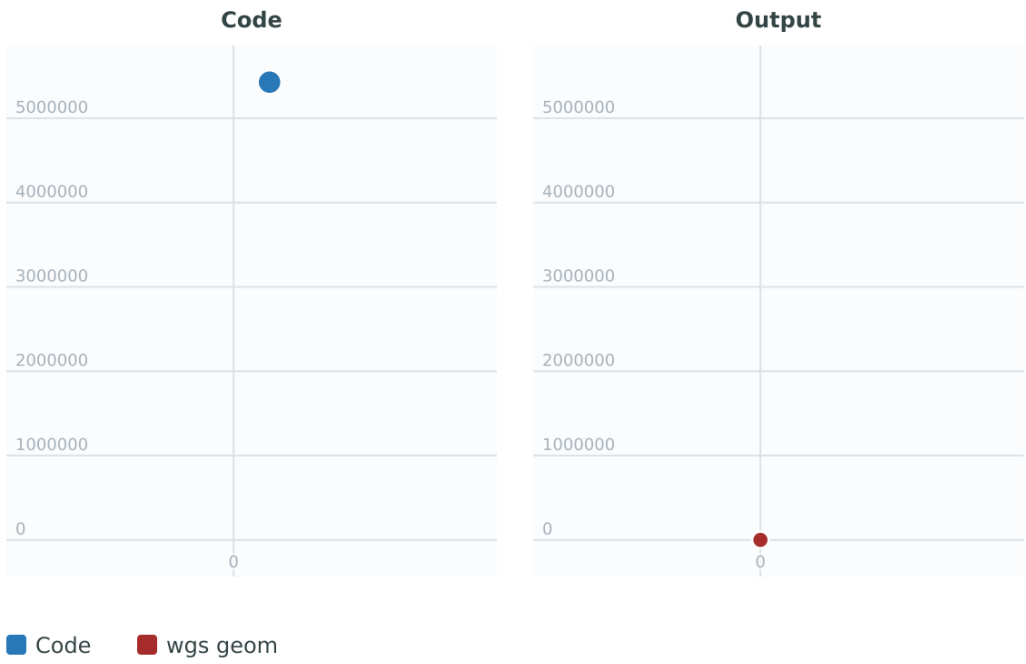
Code

```
SELECT ST_InverseTransformPipeline(
        'POINT(426857.9877165967 5427937.523342293)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::16031'
    ) AS wgs_geom;
```

Output

```
POINT(2 48.99999999999999)
```

Figure



GDA2020 示例。

This example uses ST_Transform with automatic selection of a conversion pipeline.

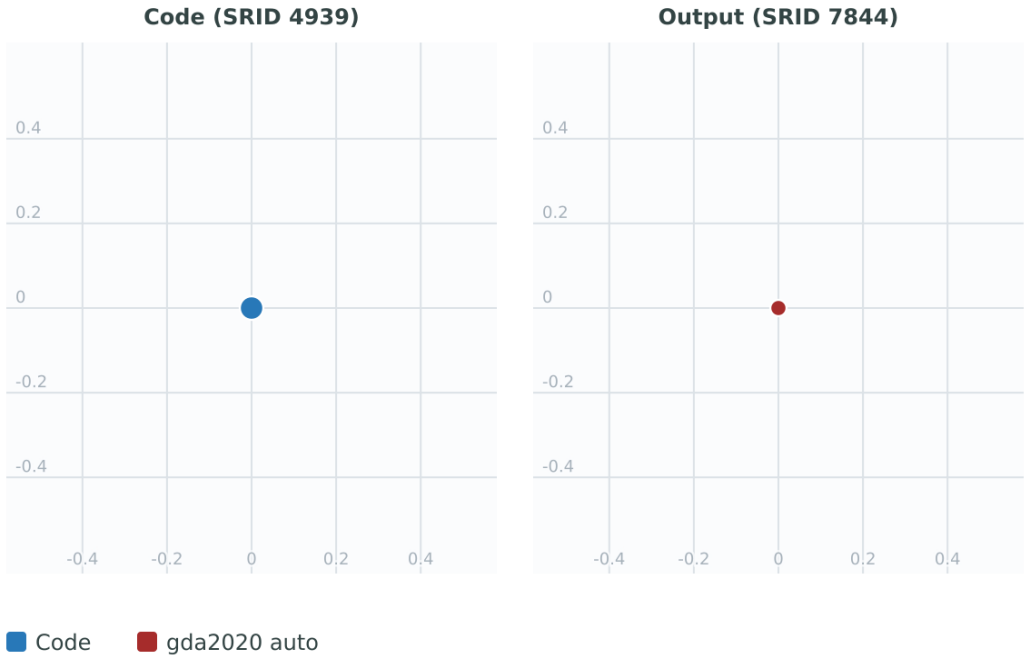
Code

```
SELECT ST_Transform('SRID=4939;POINT(143.0 -37.0)'::geometry, 7844) AS gda2020_auto;
```

Output

```
POINT(143.00000635638918 -36.999986706128176)
```

Figure



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

Mark a point as WGS 84 longitude/latitude.

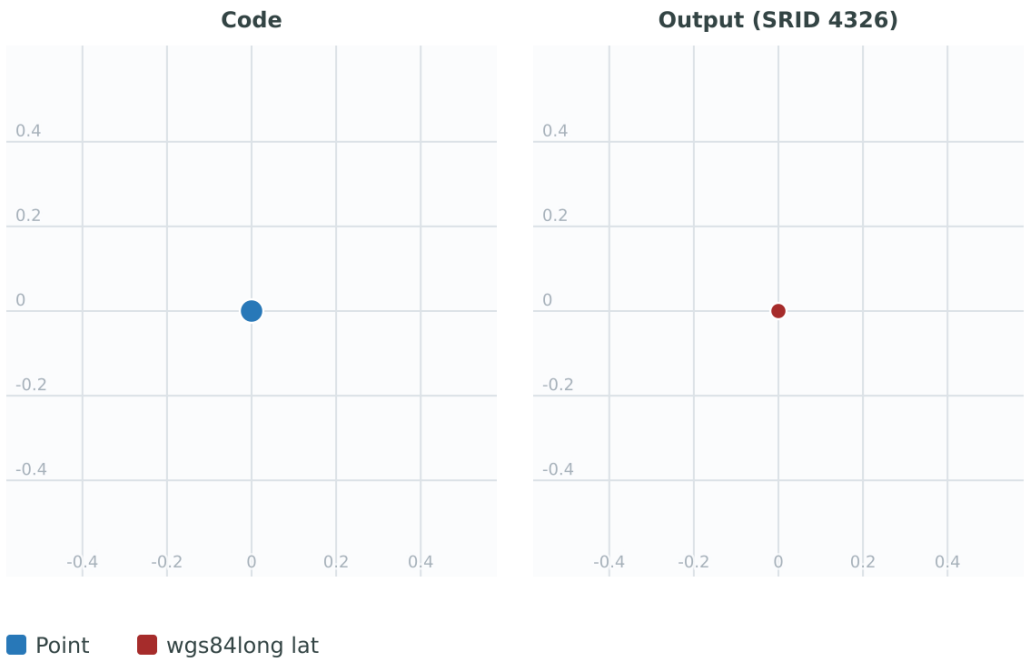
Code

```
SELECT ST_SetSRID(ST_Point(-123.365556, 48.428611), 4326) AS wgs84long_lat;
```

Output

```
SRID=4326;POINT(-123.365556 48.428611)
```

Figure



Mark a point as WGS 84 longitude/latitude and then transform it to Web Mercator.

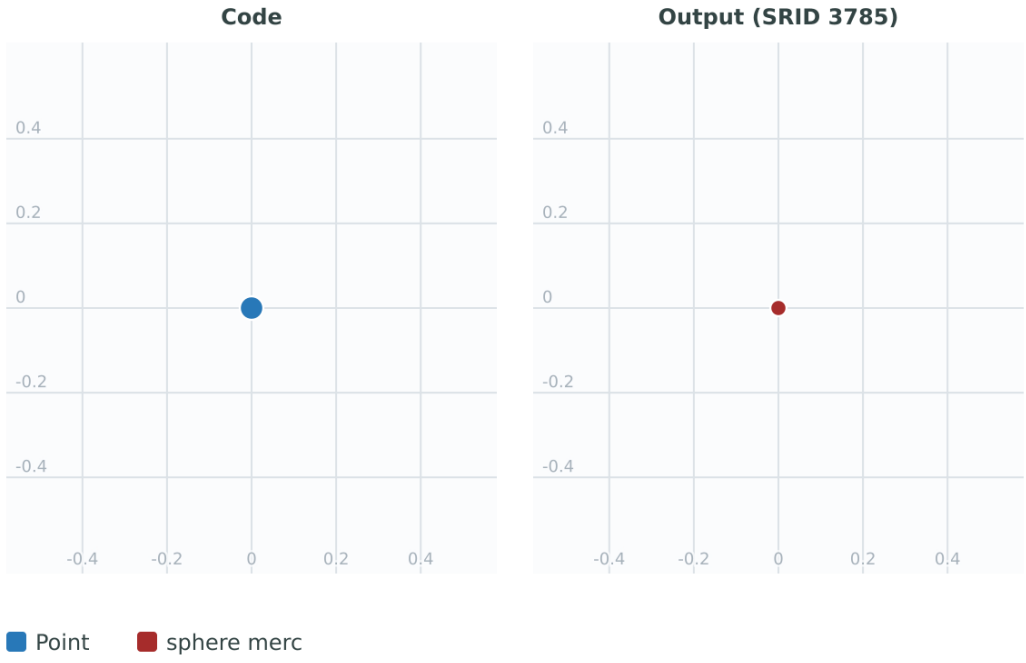
Code

```
SELECT ST_Transform(ST_SetSRID(ST_Point(-123.365556, 48.428611), 4326), 3785) AS ↩
    sphere_merc;
```

Output

```
SRID=3785;POINT(-13732990.8753491 6178458.96425423)
```

Figure



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

Code

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)', 4326));
```

Output

```
st_srid
---------
    4326
(1 row)
```

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

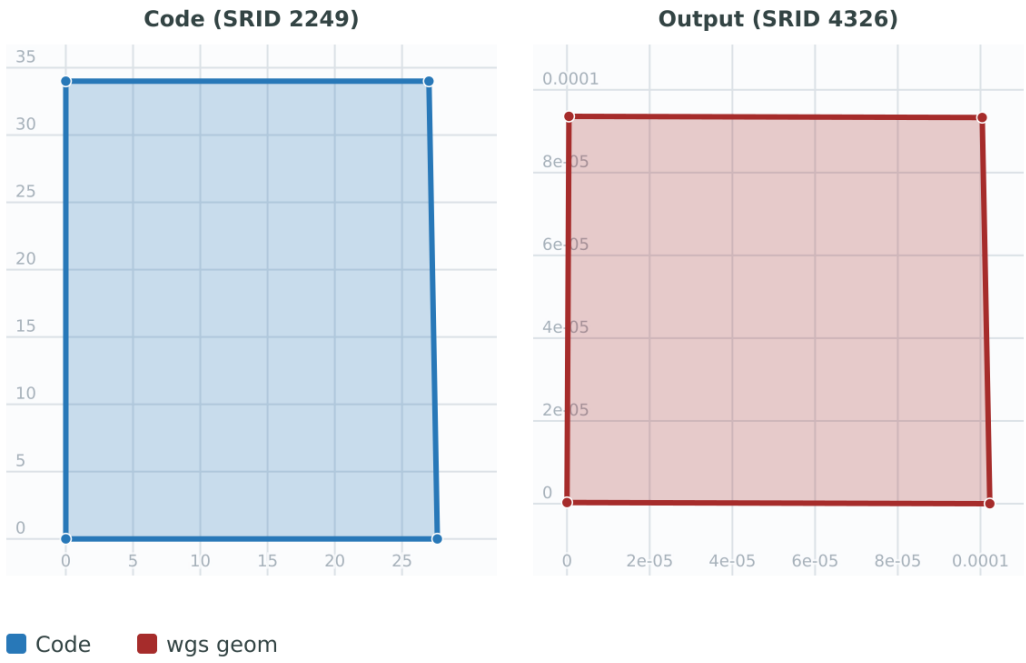
Code

```
SELECT ST_Transform(
  'SRID=2249;POLYGON((743238 2967416,743238 2967450,
    743265 2967450,743265.625 2967416,743238 2967416))'::geometry,
  4326) AS wgs_geom;
```

Output

```
SRID=4326;POLYGON((-71.1776848522251 42.3902896512903,-71.17768437663261 42.390382947801015,-71.17758443054647 ↩
    42.390382667791854,-71.17758259272304 42.390289364798825,-71.1776848522251 42.3902896512903))
```

Figure



3D Circular String example.

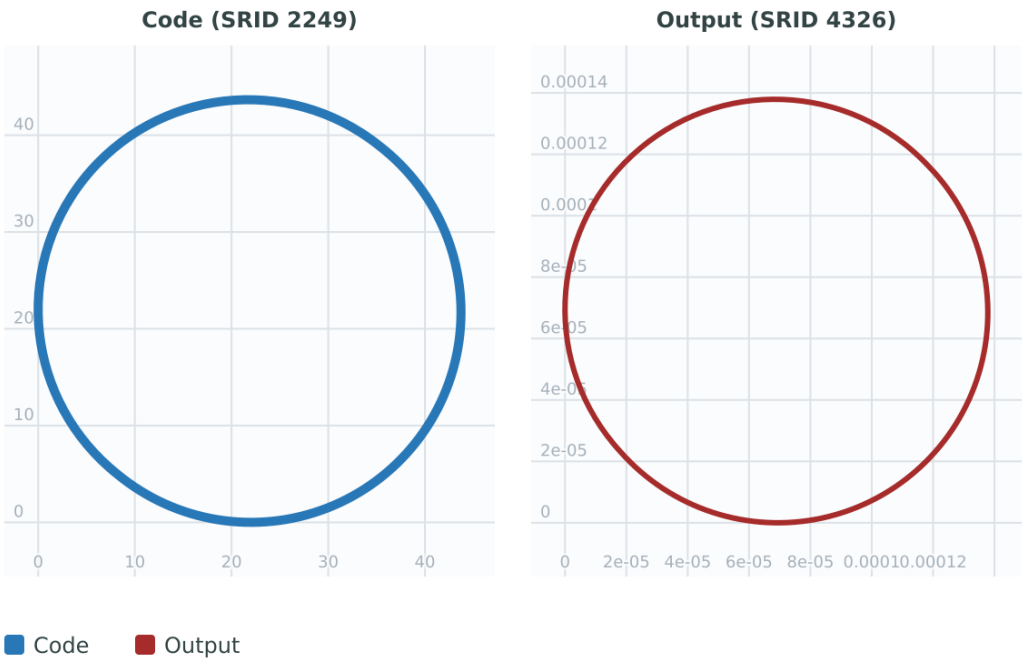
Code

```
SELECT ST_Transform('SRID=2249;CIRCULARSTRING(743238 2967416 1,743238 2967450 2,743265 ↩
    2967450 3,743265.625 2967416 3,743238 2967416 4)'::geometry,
    4326);
```

Output

```
SRID=4326;CIRCULARSTRING(-71.1776848522251 42.3902896512903 1,-71.17768437663261 42.390382947801015 2,
-71.17758443054647 42.390382667791854 3,
-71.17758259272304 42.390289364798825 3,-71.1776848522251 42.3902896512903 4)
```

Figure



⊠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。

Code

```
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://boundles
the-peel/.

Code

```
WITH data AS (
  SELECT
    ST_GeomFromText('POLYGON((170 50,170 72,-130 72,-130 50,170 50))', 4326) AS p1,
    ST_GeomFromText('POLYGON((-170 68,-170 90,-141 90,-141 68,-170 68))', 4326) AS p2,
    '+proj=gnom +ellps=WGS84 +lat_0=70 +lon_0=-160 +no_defs'::text AS gnom
)
SELECT ST_Transform(ST_Intersection(ST_Transform(p1, gnom), ST_Transform(p2, gnom)),
  gnom, 4326)
FROM data;
```

Output

```
POLYGON((-140.9999840290591 73.42768865981984,-141 68,-170 68,-170.0000026915361 74.05492179637055,-140.9999840290591 ↩
    73.42768865981984))
```

Figure



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：

Code

```
+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：

Code

```
+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：

Code

```
UPDATE spatial_ref_sys SET proj4text = '+proj=longlat +ellps=clrk66 +nadgrids=@conus, ↩
    @alaska,@ntv2_0.gsb,@ntv1_can.dat,null +no_defs' WHERE srid = 4267;
```

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

This example transforms in the forward direction.

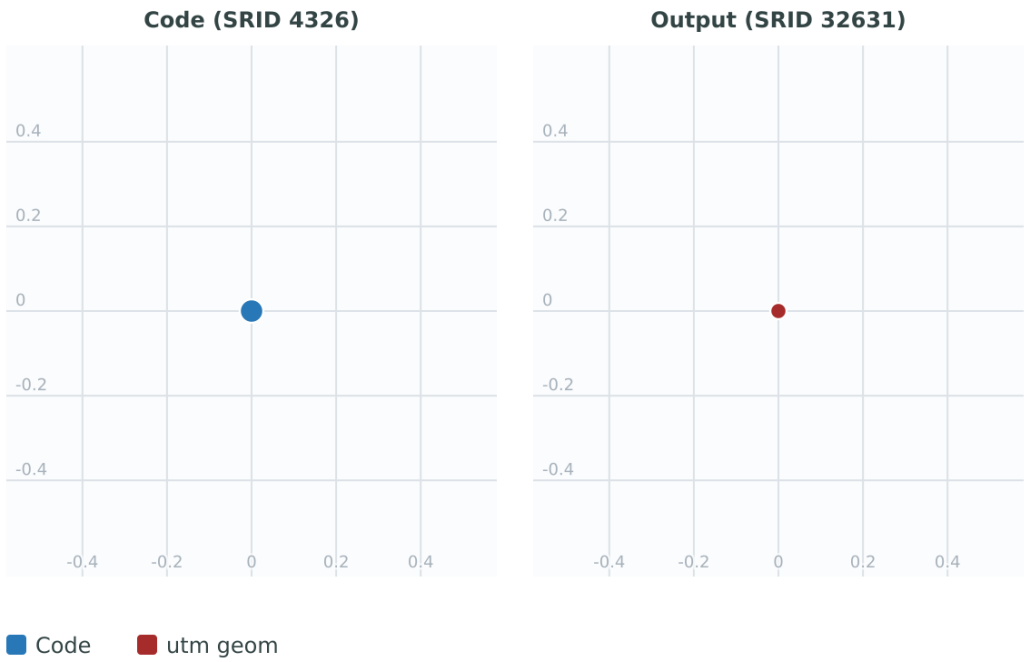
Code

```
SELECT
    ST_SetSRID(ST_TransformPipeline(
        'SRID=4326;POINT(2 49)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::16031'
    ), 32631) AS utm_geom;
```

Output

```
SRID=32631;POINT(426857.9877165967 5427937.523342293)
```

Figure



This example transforms in the inverse direction.

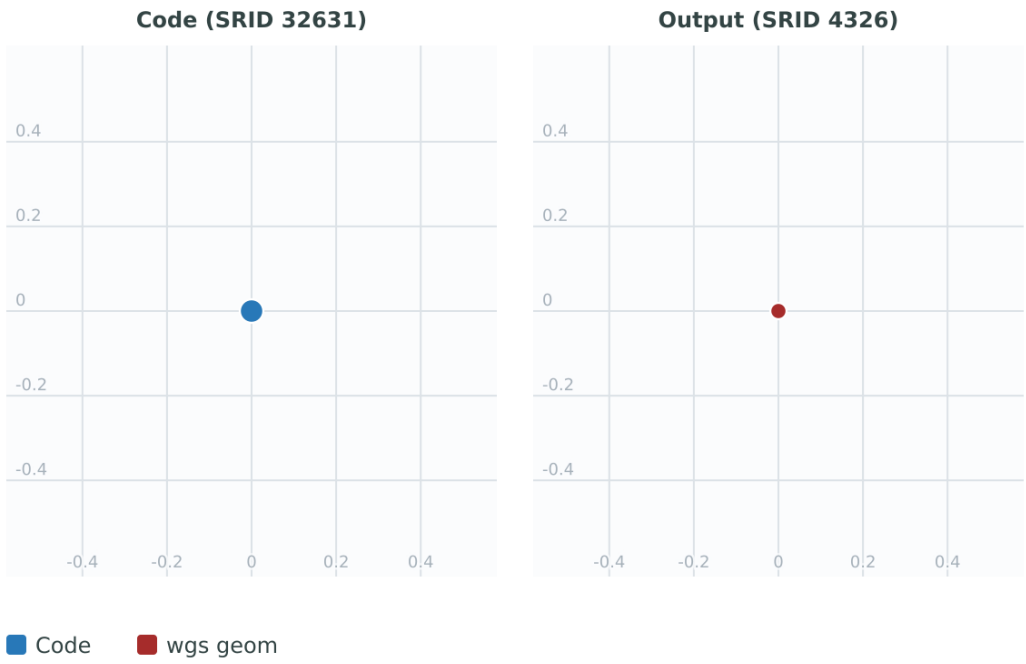
Code

```
SELECT
    ST_SetSRID(ST_InverseTransformPipeline(
        'SRID=32631;POINT(426857.9877165967 5427937.523342293)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::16031'
    ), 4326) AS wgs_geom;
```

Output

```
SRID=4326;POINT(2 48.99999999999999)
```

Figure



GDA2020 示例。

This example uses ST_Transform with automatic selection of a conversion pipeline.

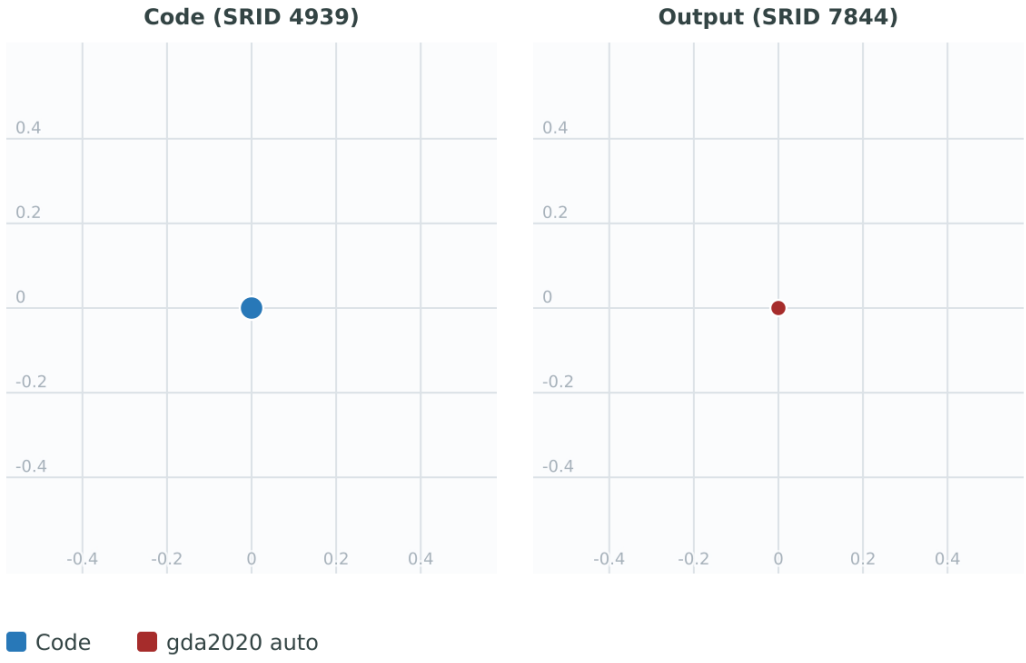
Code

```
SELECT ST_Transform('SRID=4939;POINT(143.0 -37.0)'::geometry, 7844) AS gda2020_auto;
```

Output

```
POINT(143.00000635638918 -36.999986706128176)
```

Figure



Using a defined conversion (EPSG:8447)

Code

```
SELECT
    ST_TransformPipeline(
        'SRID=4939;POINT(143.0 -37.0)'::geometry,
        'urn:ogc:def:coordinateOperation:EPSG::8447'
    ) AS gda2020_code;
```

Output

```
POINT(143.0000063280214 -36.999986718287545)
```

This example uses a PROJ pipeline definition matching EPSG:8447, as returned from projinfo. Any axisswap steps must be removed.

Code

```
SELECT
    ST_TransformPipeline(
        'SRID=4939;POINT(143.0 -37.0)'::geometry,
        '+proj=pipeline
   +step +proj=unitconvert +xy_in=deg +xy_out=rad
   +step +proj=hgridshift +grids=au_icsm_GDA94_GDA2020_conformal_and_distortion.tif
   +step +proj=unitconvert +xy_in=rad +xy_out=deg'
    ) AS gda2020_pipeline;
```

Output

```
POINT(143.0000063280214 -36.999986718287545)
```

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。

Code

```
SELECT * FROM postgis_srs_codes('EPSG') ORDER BY 1 LIMIT 10;
```

Output

```
10156
10157
10158
10160
10162
10163
10164
10165
10166
10167
```

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。

Code

```
SELECT auth_name, auth_srid, srname,point_sw AS point_sw,point_ne AS point_ne
FROM postgis_srs('EPSG', '3005');
```

Output

```
-[ RECORD 1 ]-------------------
auth_name | EPSG
auth_srid | 3005
srname    | NAD83 / BC Albers
point_sw  | POINT(-139.04 48.25)
point_ne  | POINT(-114.08 60.01)
```

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⍰⍰。

Code

```
SELECT auth_name, auth_srid, srname FROM postgis_srs_all() ORDER BY auth_name, auth_srid ↩
    LIMIT 10;
```

Output

```
auth_name | auth_srid |                                srname
-----------+-----------+-----------------------------------------------------------------------
 EPSG      | 10156     | ETRS89 + MSL UK & Ireland VORF08 depth
 EPSG      | 10157     | ETRS89 + CD UK & Ireland VORF08 depth
 EPSG      | 10158     | S34J-IRF
 EPSG      | 10160     | S34J reconstruction east-orientated
 EPSG      | 10162     | JGD2011 / Japan Plane Rectangular CS I + JGD2011 (vertical) height
 EPSG      | 10163     | JGD2011 / Japan Plane Rectangular CS II + JGD2011 (vertical) height
 EPSG      | 10164     | JGD2011 / Japan Plane Rectangular CS III + JGD2011 (vertical) height
 EPSG      | 10165     | JGD2011 / Japan Plane Rectangular CS IV + JGD2011 (vertical) height
 EPSG      | 10166     | JGD2011 / Japan Plane Rectangular CS V + JGD2011 (vertical) height
 EPSG      | 10167     | JGD2011 / Japan Plane Rectangular CS VI + JGD2011 (vertical) height
```

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+

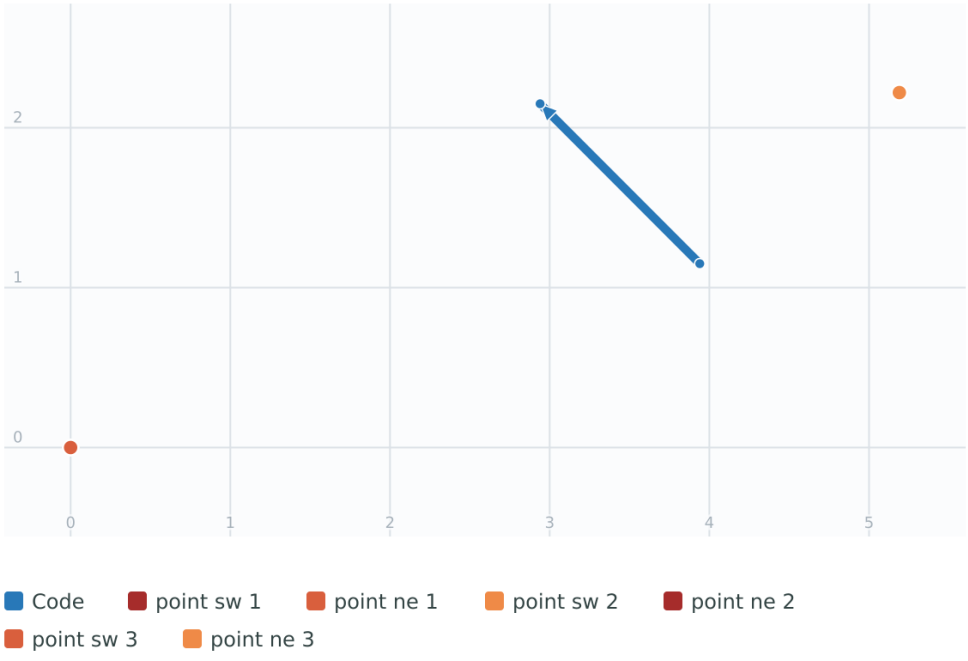
示例

搜索路易斯安那州的投影坐⍰系。

Code

```
SELECT auth_name, auth_srid, srname,point_sw AS point_sw,point_ne AS point_ne
FROM postgis_srs_search('SRID=4326;LINESTRING(-90 30,-91 31)')
WHERE auth_srid IN ('2801', '3452', '3457')
ORDER BY auth_name, auth_srid::integer
```

Figure



⍰描表格以⍰得最大范⍰并找到可能适合的投影坐⍰系。

Code

```
WITH ext AS (
  SELECT ST_Extent(geom) AS geom, Max(ST_SRID(geom)) AS srid
  FROM foo
)
```

```
SELECT auth_name, auth_srid, srname,
  point_sw AS point_sw,
  point_ne AS point_ne
FROM ext
CROSS JOIN postgis_srs_search(ST_SetSRID(ext.geom, ext.srid))
LIMIT 3;
```

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

Add a WGS 84 geography point column to a table.

Code

```
ALTER TABLE sometable ADD COLUMN geog geography(POINT, 4326);
```

Populate the geography column from longitude and latitude columns.

Code

```
UPDATE sometable
SET geog = ST_GeogFromText('SRID=4326;POINT(' || lon || ' ' || lat || ')');
```

Specify a geography point using EPSG:4267, NAD27.

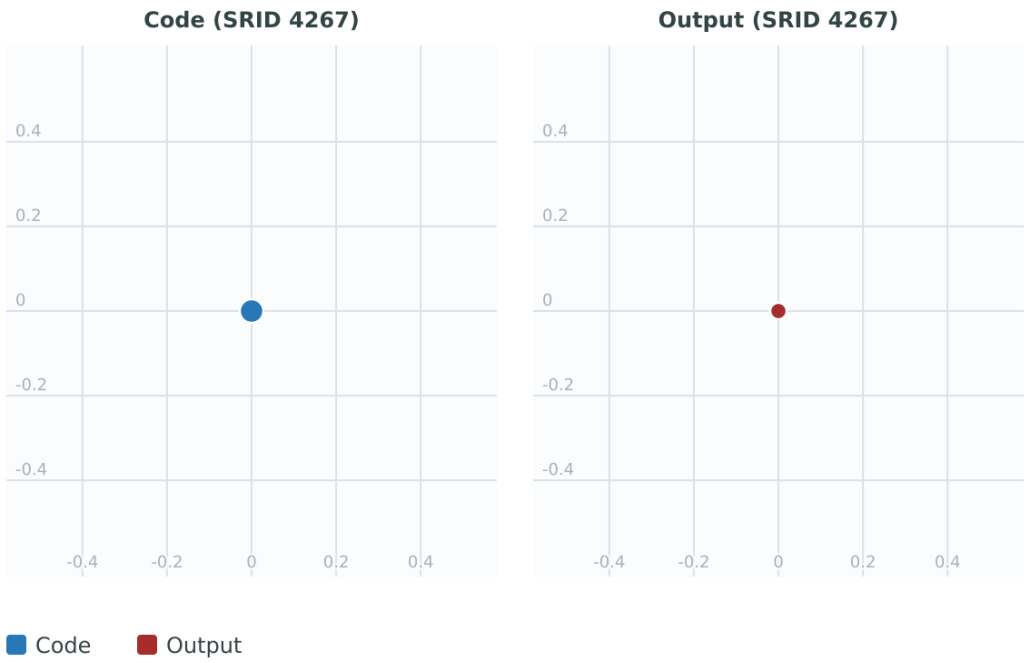
Code

```
SELECT ST_GeogFromText('SRID=4267;POINT(-77.0092 38.889588)');
```

Output

```
SRID=4267;POINT(-77.0092 38.889588)
```

Figure



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

Code

```
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

Create a LineString with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;LINESTRING(-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932)') AS geom;
```

Output

```
geometry_type | srid
--------------+------
 ST_LineString | 4269
(1 row)
```

Create a MultiLineString with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;MULTILINESTRING((-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932))') AS geom;
```

Output

```
geometry_type    | srid
--------------------+------
 ST_MultiLineString  | 4269
(1 row)
```

Create a Point with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;POINT(-71.064544 42.28787)') AS geom;
```

Output

```
geometry_type | srid
--------------+------
 ST_Point     | 4269
(1 row)
```

Create a Polygon with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;POLYGON((-71.1776585052917 ↩
    42.3902909739571,-71.1776820268866 42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))') AS geom;
```

Output

```
geometry_type | srid
--------------+------
 ST_Polygon   | 4269
(1 row)
```

Create a MultiPolygon with SRID 4269.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('SRID=4269;MULTIPOLYGON(((-71.1031880899493 42.3152774590236,
-71.1031627617667 42.3152960829043,-71.102923838298 42.3149156848307,
-71.1023097974109 42.3151969047397,-71.1019285062273 42.3147384934248,
-71.102505233663 42.3144722937587,-71.10277487471 42.3141658254797,
-71.103113945163 42.3142739188902,-71.10324876416 42.31402489987,
-71.1033002961013 42.3140393340215,-71.1033488797549 42.3139495090772,
-71.103396240451 42.3138632439557,-71.1041521907712 42.3141153348029,
-71.1041411411543 42.3141545014533,-71.1041287795912 42.3142114839058,
-71.1041188134329 42.3142693656241,-71.1041112482575 42.3143272556118,
-71.1041072845732 42.3143851580048,-71.1041057218871 42.3144430686681,
-71.1041065602059 42.3145009876017,-71.1041097995362 42.3145589148055,
-71.1041166403905 42.3146168544148,-71.1041258822717 42.3146748022936,
-71.1041375307579 42.3147318674446,-71.1041492906949 42.3147711126569,
-71.1041598612795 42.314808571739,-71.1042515013869 42.3151287620809,
-71.1041173835118 42.3150739481917,-71.1040809891419 42.3151344119048,
-71.1040438678912 42.3151191367447,-71.1040194562988 42.3151832057859,
-71.1038734225584 42.3151140942995,-71.1038446938243 42.3151006300338,
-71.1038315271889 42.315094347535,-71.1037393329282 42.315054824985,
-71.1035447555574 42.3152608696313,-71.1033436658644 42.3151648370544,
-71.1032580383161 42.3152269126061,-71.103223066939 42.3152517403219,
-71.1031880899493 42.3152774590236)),
((-71.1043632495873 42.315113108546,-71.1043583974082 42.3151211109857,
-71.1043443253471 42.3150676015829,-71.1043850704575 42.3150793250568,-71.1043632495873 ↩
    42.315113108546)))') AS geom;
```

Output

```
geometry_type   | srid
----------------+------
 ST_MultiPolygon | 4269
(1 row)
```

Create a 3D CircularString without an SRID.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)') AS ↩
    geom;
```

Output

```
geometry_type    | srid
-----------------+------
 ST_CircularString  | 0
(1 row)
```

Create a PolyhedralSurface without an SRID.

Code

```
SELECT ST_GeometryType(geom) AS geometry_type, ST_SRID(geom) AS srid
FROM ST_GeomFromEWKT('POLYHEDRALSURFACE(
    ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
    ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
    ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
)') AS geom;
```

Output

```
geometry_type       | srid
------------------------+------
 ST_PolyhedralSurface     | 0
(1 row)
```

相关信息

ST_AsEWKT, ST_GeomFromText

7.8.1.7 ST_GeomFromMARC21

ST_GeomFromMARC21 — 输入 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 输入构建 PostGIS 几何形,其中可以包含 `POINT` 或 `POLYGON`。如果同一 MARC21/XML 输入中有多个地理数据条目，将返回 `MULTIPOINT` 或 `MULTIPOLYGON`。如果输入包含混合几何类型，输出将返回 `GEOMETRYCOLLECTION`。如果 MARC21/XML 输入不包含任何地理数据（数据字段：034），输出返回 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 ⊠⊠的地理数据中，目前无法描述⊠⊠坐⊠的空⊠参考系，因此⊠函数将始⊠返回 SRID 0 的几何⊠形。



Note
返回的 POLYGON 几何⊠形将始⊠是⊠⊠⊠方向。

示例

⊠⊠包含⊠⊠⊠ hddd.dddddd 的⊠个 POINT 的 MARC21/XML 地理数据

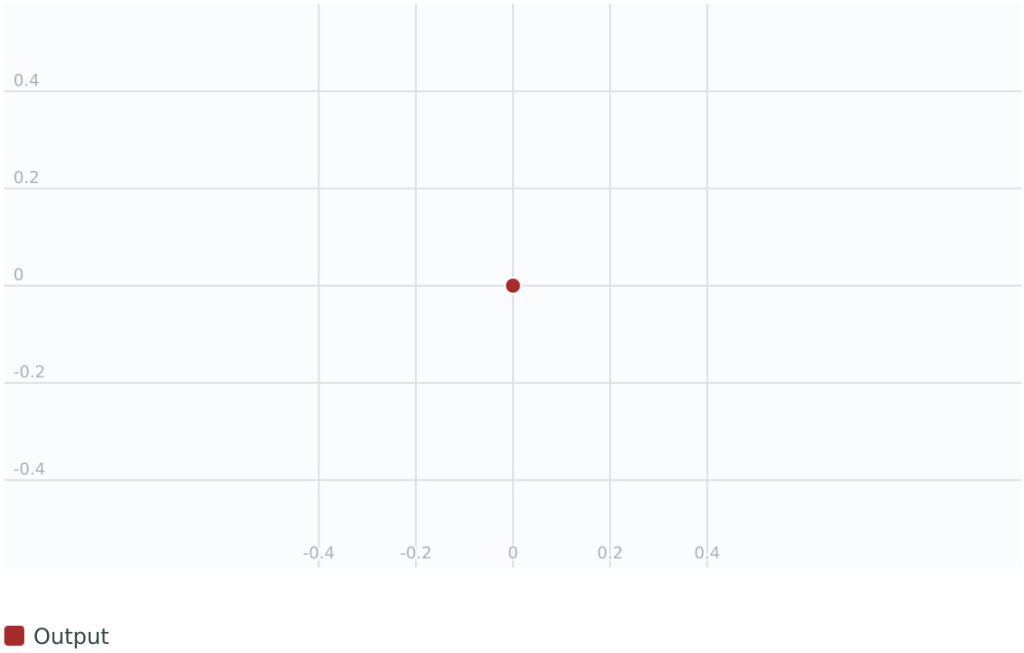
Code

```
SELECT ST_GeomFromMARC21(
'<record xmlns="http://www.loc.gov/MARC21/slim">
    <leader
>00000nz a2200000nc 4500</leader>
    <controlfield tag="001"
>040277569</controlfield>
    <datafield tag="034" ind1=" " ind2=" ">
        <subfield code="d"
>W004.500000</subfield>
        <subfield code="e"
>W004.500000</subfield>
        <subfield code="f"
>N054.250000</subfield>
        <subfield code="g"
>N054.250000</subfield>
    </datafield>
</record
>'
);
```

Output

```
POINT(-4.5 54.25)
```

Figure



□□包含□□□ hdddmmss 的□个 POLYGON 的 MARC21/XML 地理数据

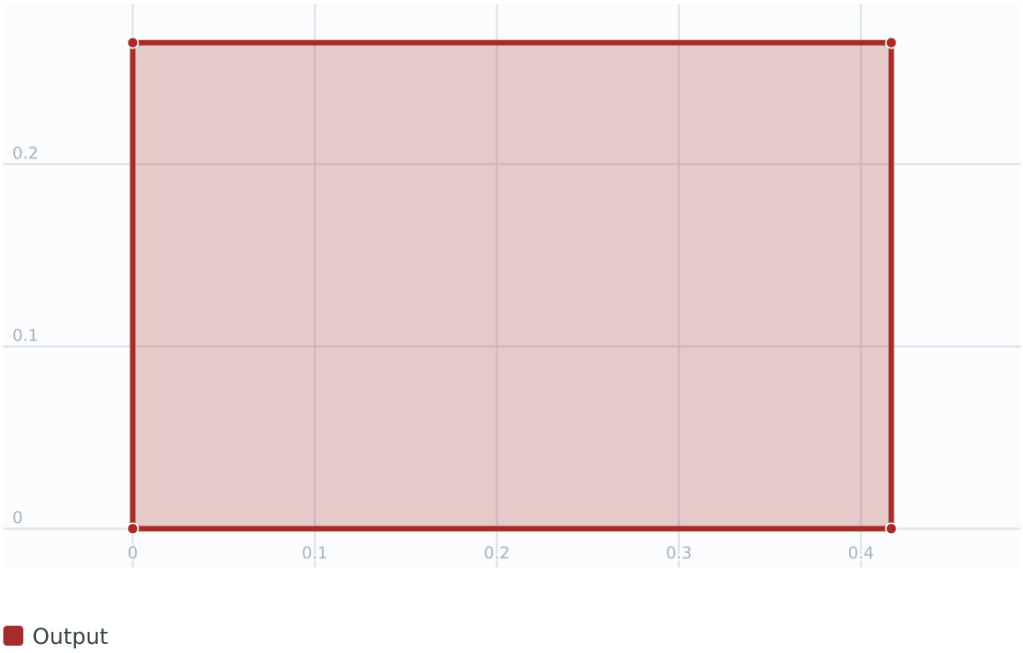
Code

```
SELECT ST_GeomFromMARC21(
'<record xmlns="http://www.loc.gov/MARC21/slim">
    <leader
>01062cem a2200241 a 4500</leader>
    <controlfield tag="001"
>   84696781 </controlfield>
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="b"
>50000</subfield>
        <subfield code="d"
>E0130600</subfield>
        <subfield code="e"
>E0133100</subfield>
        <subfield code="f"
>N0523900</subfield>
        <subfield code="g"
>N0522300</subfield>
    </datafield>
</record
>'
);
```

Output

```
POLYGON((13.1 52.65,13.516666666666667 52.65,13.516666666666667 52.38333333333333,13.1 52.38333333333333,13.1 52.65))
```

Figure



⊠⊠包含 `POLYGON` 和 `POINT` 的 MARC21/XML 地理数据：

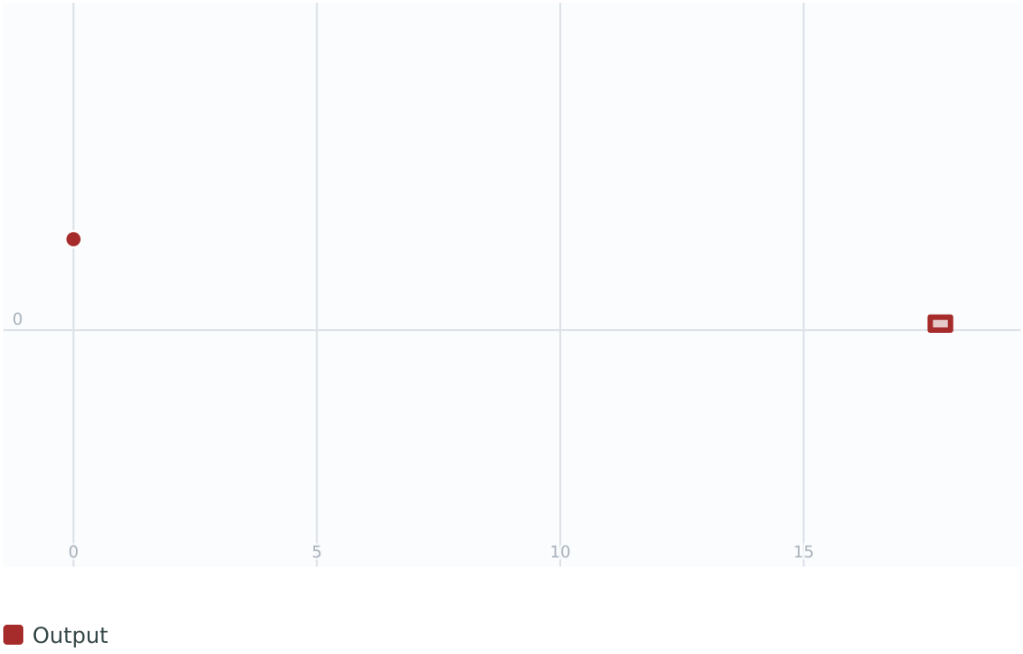
Code

```
SELECT ST_GeomFromMARC21(
'<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="b"
>50000</subfield>
        <subfield code="d"
>E0130600</subfield>
        <subfield code="e"
>E0133100</subfield>
        <subfield code="f"
>N0523900</subfield>
        <subfield code="g"
>N0522300</subfield>
    </datafield>
    <datafield tag="034" ind1=" " ind2=" ">
        <subfield code="d"
>W004.500000</subfield>
        <subfield code="e"
>W004.500000</subfield>
        <subfield code="f"
>N054.250000</subfield>
        <subfield code="g"
>N054.250000</subfield>
    </datafield>
</record
>'
);
```

Output

```
GEOMETRYCOLLECTION(POLYGON((13.1 52.65,13.516666666666667 52.65,13.516666666666667 52.38333333333333,13.1 ↩
    52.38333333333333,13.1 52.65)), POINT(-4.5 54.25))
```

Figure



相关信息

ST_AsMARC21

7.8.1.8 ST_GeometryFromText

ST_GeometryFromText — Returns a geometry from Well-Known Text (WKT). Alias for ST_GeomFromText.

Synopsis

geometry **ST_GeometryFromText**(text WKT);
geometry **ST_GeometryFromText**(text WKT, integer srid);

描述

This function is an alias for ST_GeomFromText and provides the same signatures.

可用性：1.2.2

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。

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')

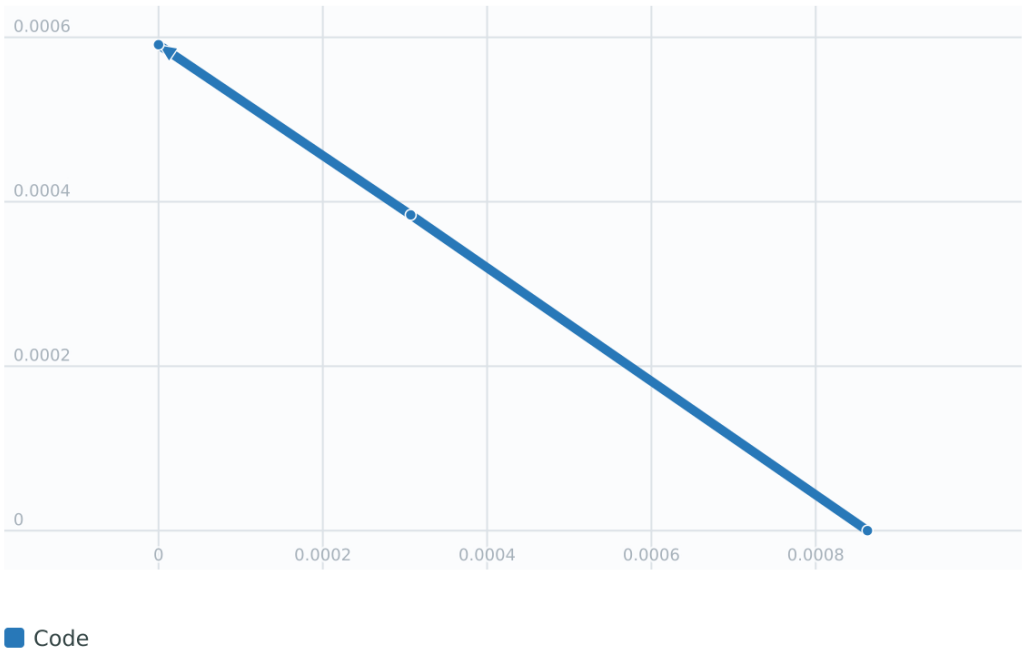
示例

Create a LineString with SRID 0.

Code

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932)'));
```

Figure

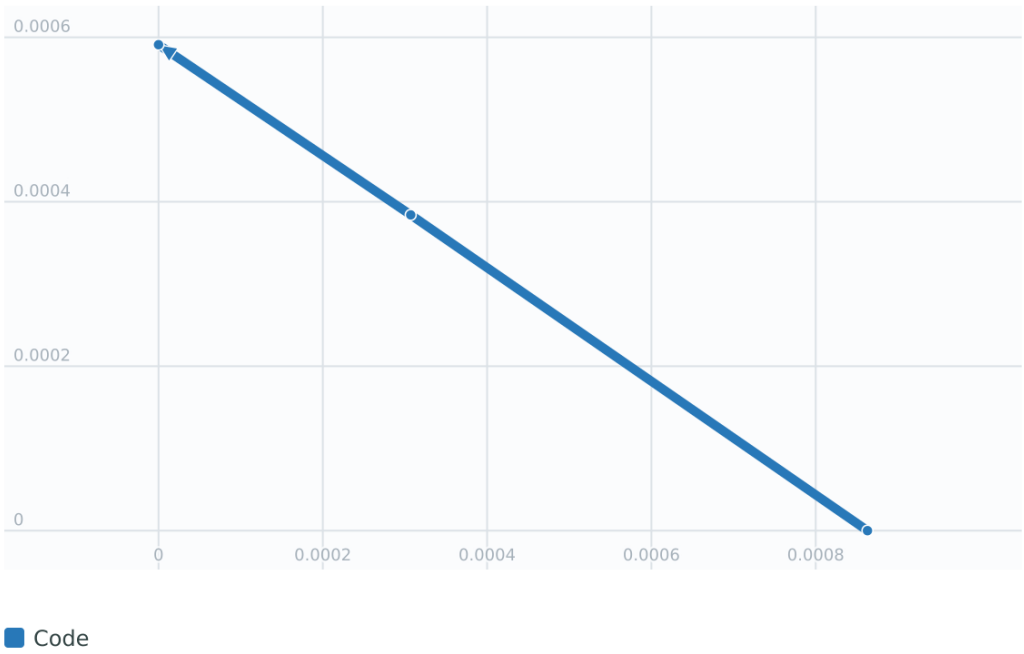


Create the same LineString with SRID 4269.

Code

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932)', 4269));
```

Figure

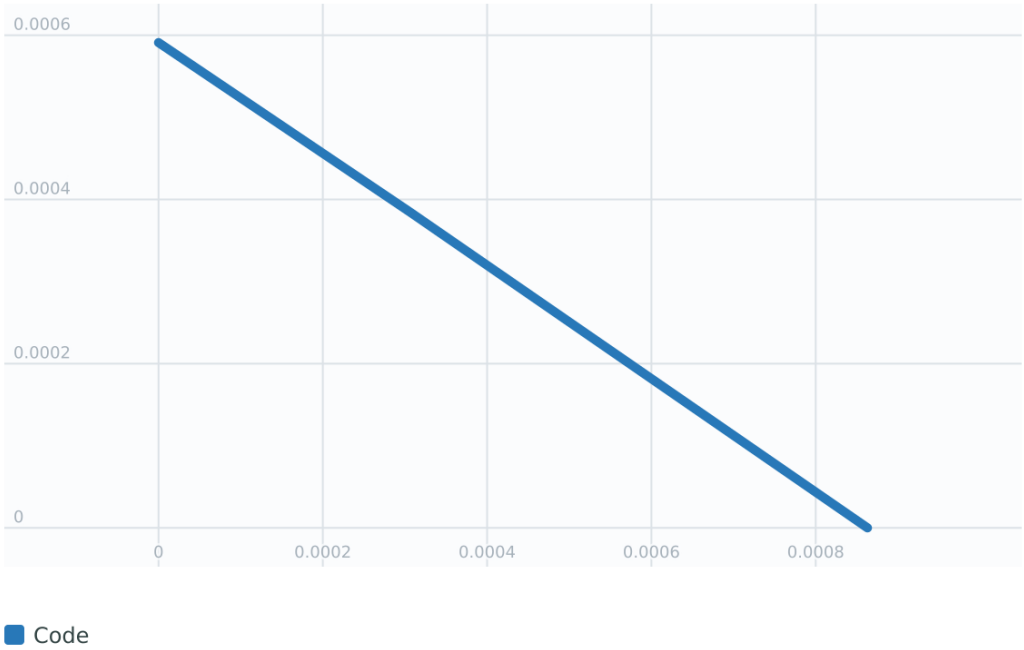


Create a MultiLineString.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('MULTILINESTRING((-71.160281 42.258729,-71.160837 ↩
    42.259113,-71.161144 42.25932))'));
```

Figure



Create a Point.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('POINT(-71.064544 42.28787)'));
```

Output

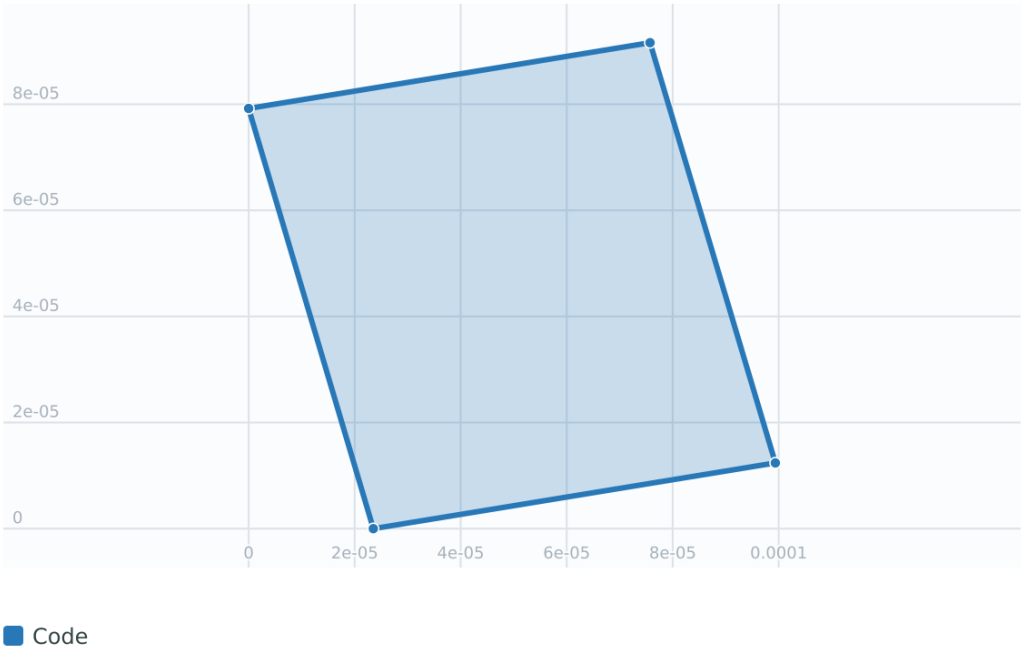
```
ST_Point
```

Create a Polygon.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('POLYGON((-71.1776585052917 ↩
    42.3902909739571,-71.1776820268866 42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))'));
```

Figure

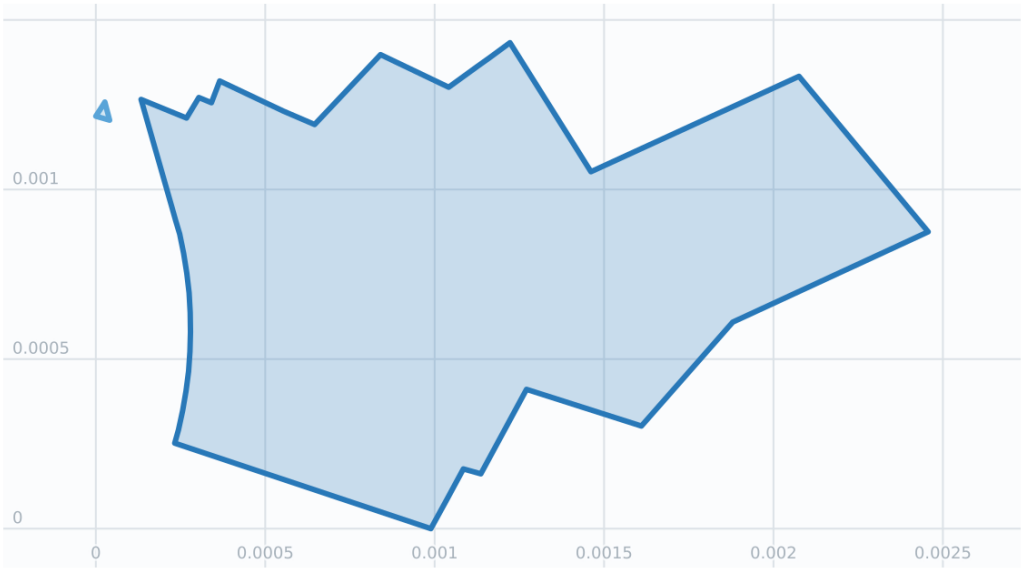


Create a MultiPolygon with SRID 4326.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('MULTIPOLYGON(((-71.1031880899493 42.3152774590236,
-71.1031627617667 42.3152960829043,-71.102923838298 42.3149156848307,
-71.1023097974109 42.3151969047397,-71.1019285062273 42.3147384934248,
-71.102505233663 42.3144722937587,-71.10277487471 42.3141658254797,
-71.103113945163 42.3142739188902,-71.10324876416 42.31402489987,
-71.1033002961013 42.3140393340215,-71.1033488797549 42.3139495090772,
-71.103396240451 42.3138632439557,-71.1041521907712 42.3141153348029,
-71.1041411411543 42.3141545014533,-71.1041287795912 42.3142114839058,
-71.1041188134329 42.3142693656241,-71.1041112482575 42.3143272556118,
-71.1041072845732 42.3143851580048,-71.1041057218871 42.3144430686681,
-71.1041065602059 42.3145009876017,-71.1041097995362 42.3145589148055,
-71.1041166403905 42.3146168544148,-71.1041258822717 42.3146748022936,
-71.1041375307579 42.3147318674446,-71.1041492906949 42.3147711126569,
-71.1041598612795 42.314808571739,-71.1042515013869 42.3151287620809,
-71.1041173835118 42.3150739481917,-71.1040809891419 42.3151344119048,
-71.1040438678912 42.3151191367447,-71.1040194562988 42.3151832057859,
-71.1038734225584 42.3151140942995,-71.1038446938243 42.3151006300338,
-71.1038315271889 42.315094347535,-71.1037393329282 42.315054824985,
-71.1035447555574 42.3152608696313,-71.1033436658644 42.3151648370544,
-71.1032580383161 42.3152269126061,-71.103223066939 42.3152517403219,
-71.1031880899493 42.3152774590236)),
((-71.1043632495873 42.315113108546,-71.1043583974082 42.3151211109857,
-71.1043443253471 42.3150676015829,-71.1043850704575 42.3150793250568,-71.1043632495873 ↩
    42.315113108546)))', 4326));
```

Figure

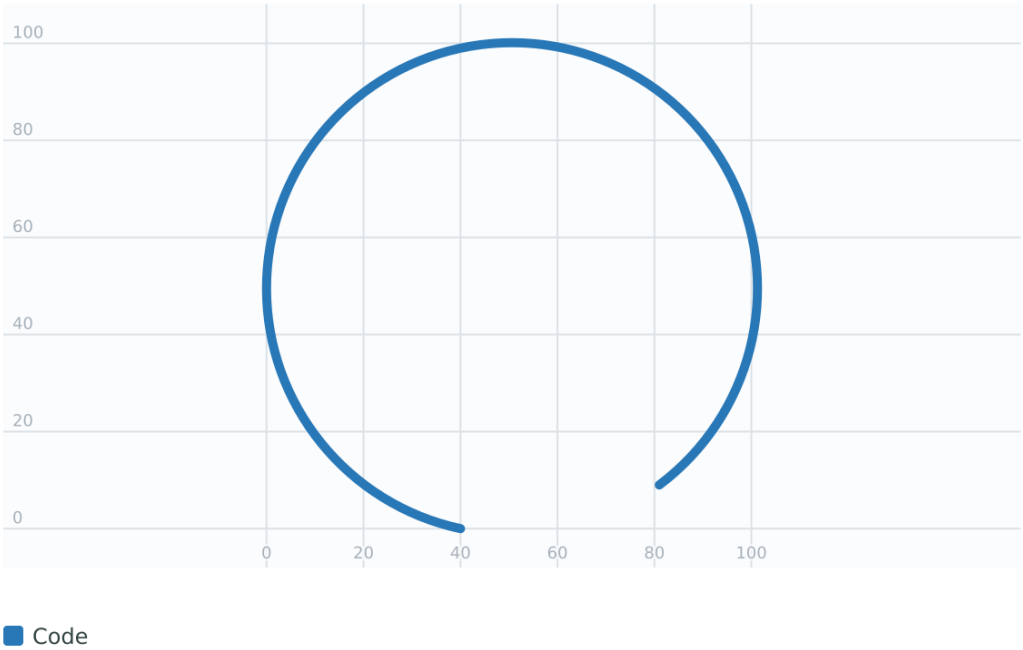


Create a CircularString.

Code

```
SELECT ST_GeometryType(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 ↩
    150406)'));
```

Figure



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

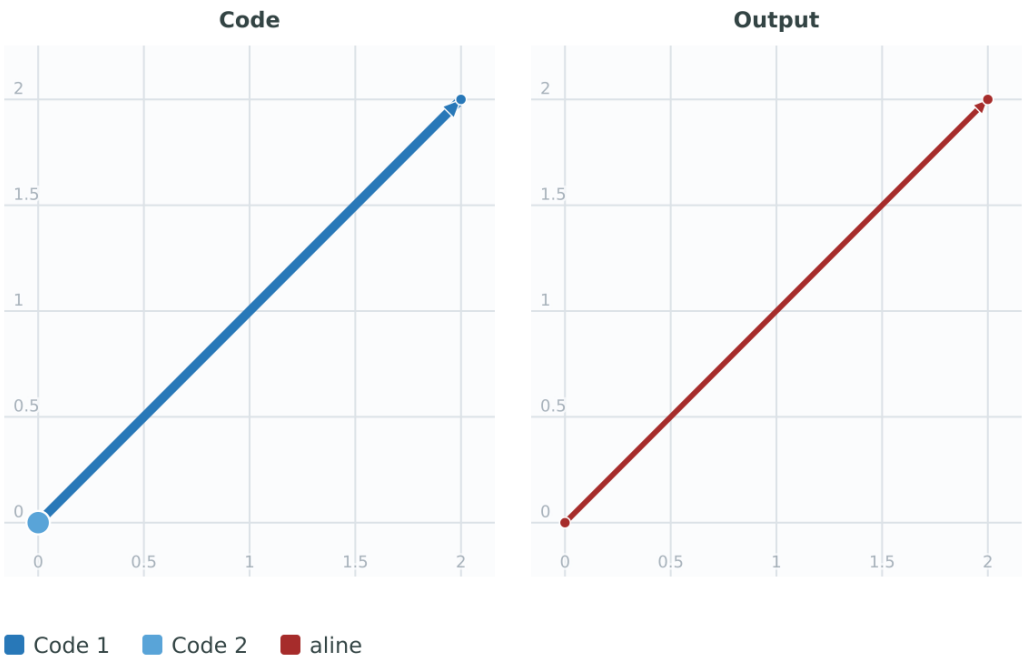
Code

```
SELECT ST_LineFromText('LINESTRING(1 2,3 4)') AS aline,
ST_LineFromText('POINT(1 2)') IS NULL AS null_return;
```

Output

```
aline                | null_return
-----------------------------------
LINESTRING(1 2,3 4)  | t
```

Figure



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

Code

```
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

Code

```
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

Create a 3D MultiPolygon with an interior ring and SRID 0.

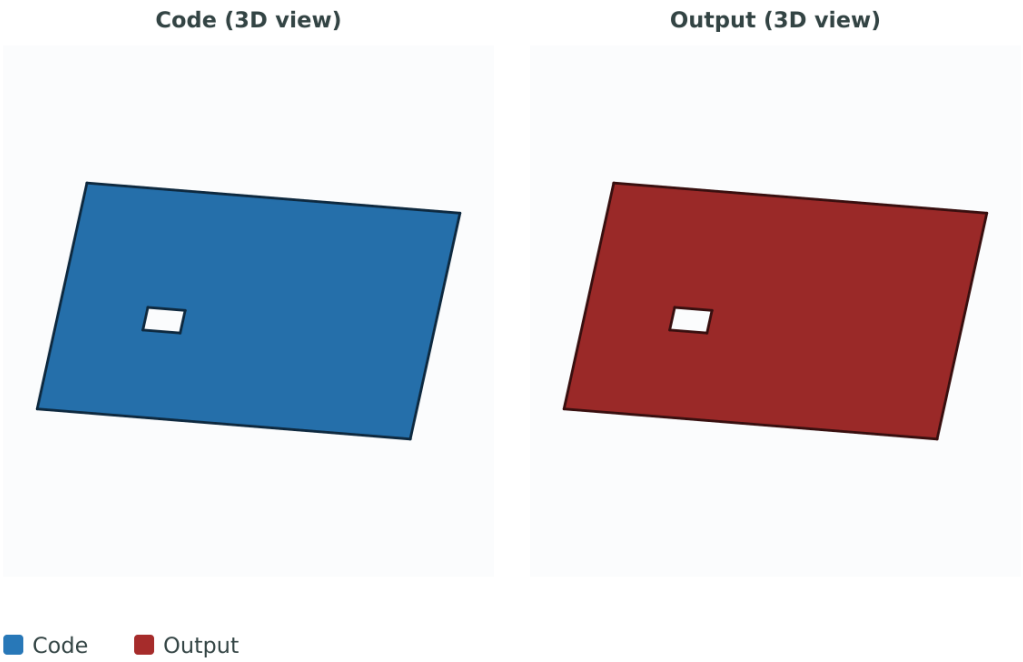
Code

```
SELECT ST_MPolyFromText(
    'MULTIPOLYGON(((0 0 1,20 0 1,20 20 1,0 20 1,0 0 1),
                   (5 5 3,5 7 3,7 7 3,7 5 3,5 5 3)))'
);
```

Output

```
MULTIPOLYGON(((0 0 1,20 0 1,20 20 1,0 20 1,0 0 1),(5 5 3,5 7 3,7 7 3,7 5 3,5 5 3)))
```

Figure

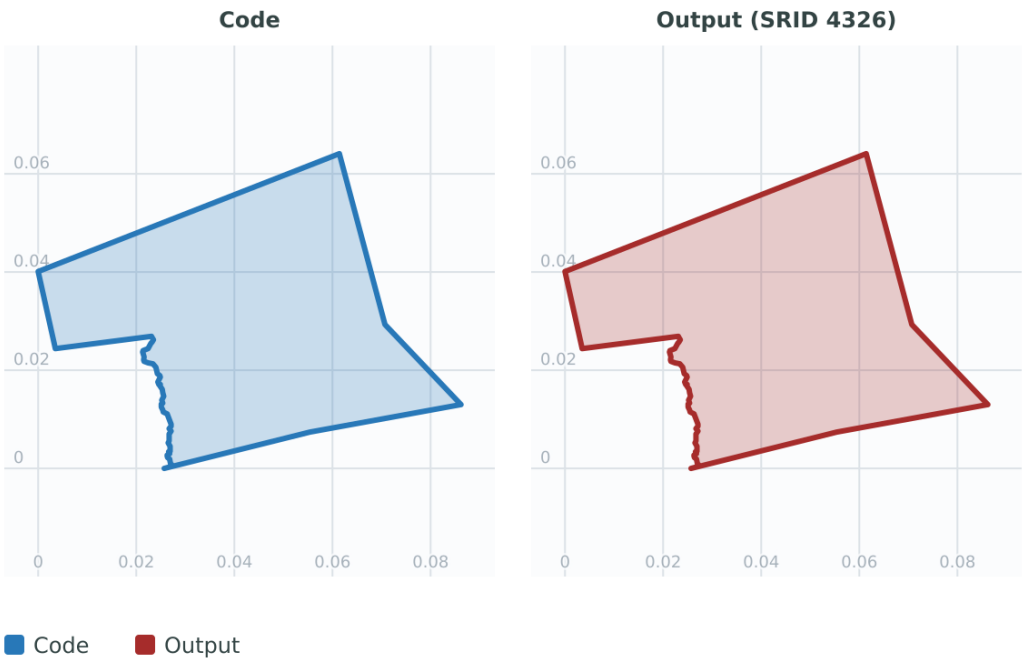


Create a MultiPolygon with SRID 4326.

Code

```
SELECT ST_MPolyFromText(
    'MULTIPOLYGON(((-70.916 42.1002,-70.9468 42.0946,-70.9765 42.0872,
      -70.9754 42.0875,-70.9749 42.0879,-70.9752 42.0881,-70.9754 42.0891,
      -70.9758 42.0894,-70.9759 42.0897,-70.9759 42.0899,-70.9754 42.0902,
      -70.9756 42.0906,-70.9753 42.0907,-70.9753 42.0917,-70.9757 42.0924,
      -70.9755 42.0928,-70.9755 42.0942,-70.9751 42.0948,-70.9755 42.0953,
      -70.9751 42.0958,-70.9751 42.0962,-70.9759 42.0983,-70.9767 42.0987,
      -70.9768 42.0991,-70.9771 42.0997,-70.9771 42.1003,-70.9768 42.1005,
      -70.977 42.1011,-70.9766 42.1019,-70.9768 42.1026,-70.9769 42.1033,
      -70.9775 42.1042,-70.9773 42.1043,-70.9776 42.1043,-70.9778 42.1048,
      -70.9773 42.1058,-70.9774 42.1061,-70.9779 42.1065,-70.9782 42.1078,
      -70.9788 42.1085,-70.9798 42.1087,-70.9806 42.109,-70.9807 42.1093,
      -70.9806 42.1099,-70.9809 42.1109,-70.9808 42.1112,-70.9798 42.1116,
      -70.9792 42.1127,-70.979 42.1129,-70.9787 42.1134,-70.979 42.1139,
      -70.9791 42.1141,-70.9987 42.1116,-71.0022 42.1273,-70.9408 42.1513,
      -70.9315 42.1165,-70.916 42.1002)))',
    4326
);
```

Figure



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

Create a point with the default SRID 0.

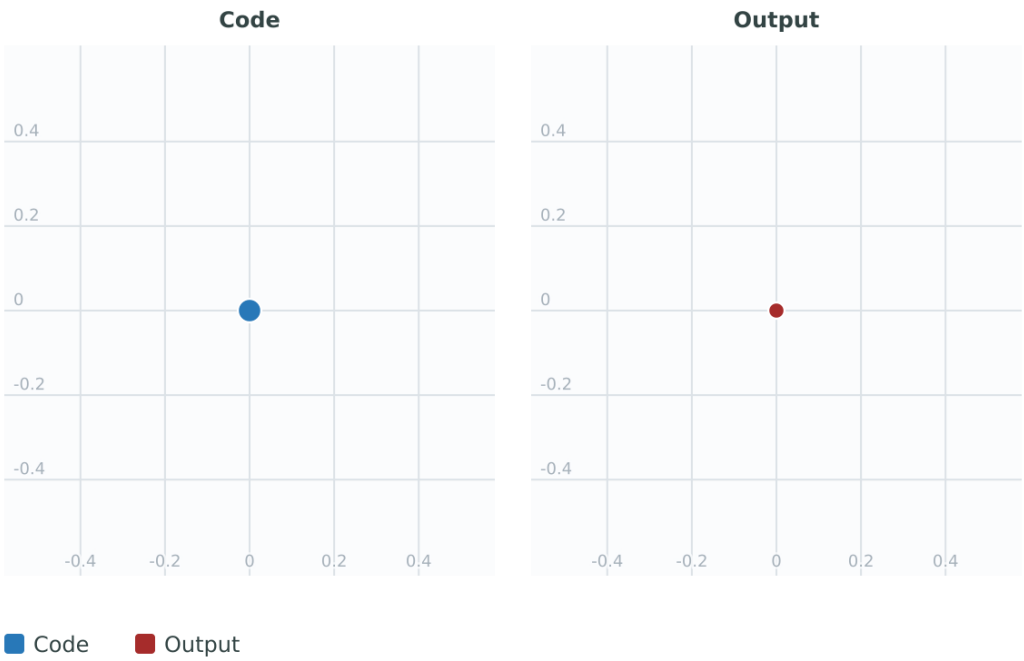
Code

```
SELECT ST_PointFromText('POINT(-71.064544 42.28787)');
```

Output

```
POINT(-71.064544 42.28787)
```

Figure



Create a point with SRID 4326.

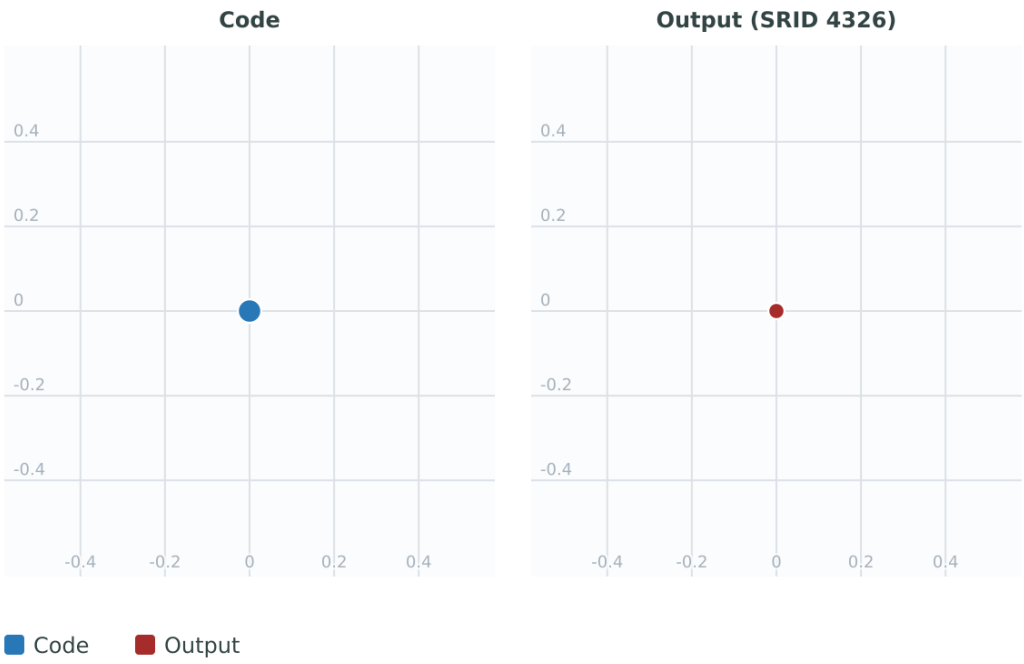
Code

```
SELECT ST_PointFromText('POINT(-71.064544 42.28787)', 4326);
```

Output

```
SRID=4326;POINT(-71.064544 42.28787)
```

Figure



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

The result is returned as a native geometry. The documentation test runner compares that value directly with the readable EWKT shown below and uses the same geometry for the generated figure.

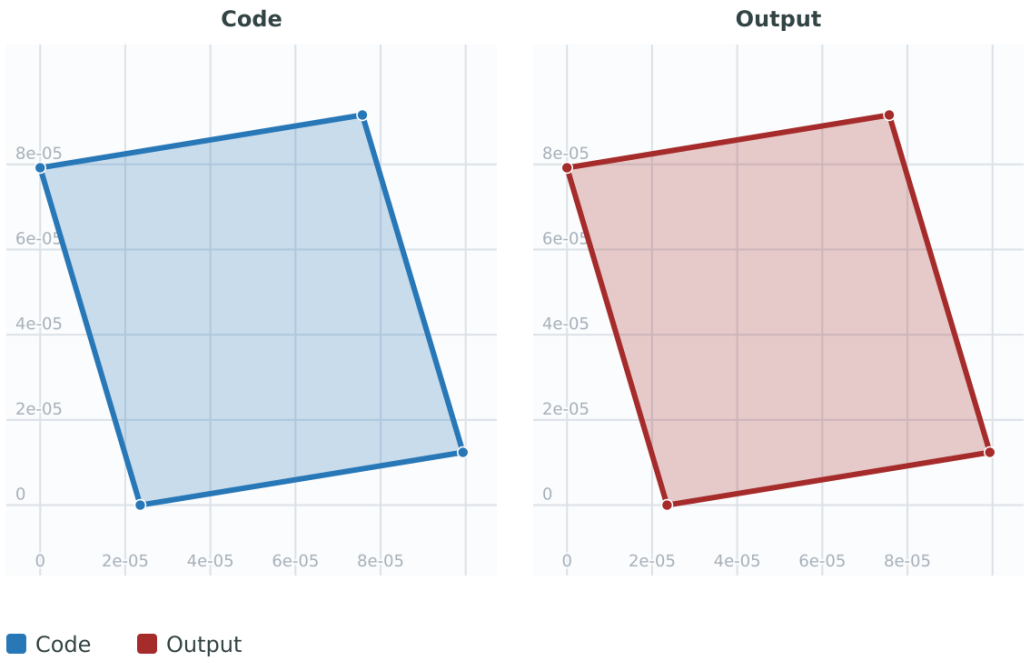
Code

```
SELECT ST_PolygonFromText('POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 ↩
    42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↩
    42.3902909739571))');
```

Output

```
POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 42.3903701743239,-71.1776063012595 ↩
    42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 42.3902909739571))
```

Figure



Code

```
SELECT ST_PolygonFromText('POINT(1 2)') IS NULL AS point_is_notpoly;
```

Output

```
t
```

相关信息

ST_GeomFromText

7.8.1.16 ST_WKTToSQL

ST_WKTToSQL — Returns a geometry from Well-Known Text (WKT). SQL/MM alias for ST_GeomFromText.

Synopsis

geometry **ST_WKTToSQL**(text WKT);

描述

This function is the SQL/MM alias for ST_GeomFromText with no SRID argument.

 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.

Code

```
SELECT ST_GeogFromWKB(E'\\001\\002\\000\\000\\000\\002\\000\\000\\000\\037\\205\\353Q ↩
    \\270~\\\\\\300\\323Mb\\020X\\231C@\\020X9\\264\\310~\\\\\\300)\\\\\\217\\302\\365\\230 ↩
    C@');
```

Output

```
LINESTRING(-113.98 39.198,-113.981 39.195)
```

Figure



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

Constructs a PostGIS geometry from an Extended Well-Known Binary (EWKB) representation.

> **Note**
> EWKB 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

This example constructs `LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.259` in NAD83 longitude/latitude (SRID 4269) from its EWKB representation.

PostgreSQL escape string syntax (`E'...'`) always interprets backslash escapes. Each backslash therefore has to be doubled so that the bytea input parser receives the octal escapes.

Code

```
SELECT ST_GeomFromEWKB((
    E'\\001\\002\\000\\000\\040\\255\\020\\000\\000\\003\\000\\000\\000\\344\\112\\075' ||
    E'\\013\\102\\312\\121\\300\\156\\303\\050\\010\\036\\041\\105\\100\\047\\277\\105' ||
    E'\\047\\113\\312\\121\\300\\366\\173\\142\\235\\052\\041\\105\\100\\225\\174\\354' ||
    E'\\056\\120\\312\\121\\300\\160\\231\\323\\145\\061\\041\\105\\100'
)::bytea);
```

Output

```
0102000020 ↩
    AD10000003000000E44A3D0B42CA51C06EC328081E21454027BF45274BCA51C0F67B629D2A214540957CEC2E50CA51C07099D36531214540
```

Figure



With `standard_conforming_strings` enabled (the default), ordinary string literals preserve backslashes. The same EWKB can therefore be written with a single backslash before each octal byte value.

Code

```
SELECT ST_GeomFromEWKB((
    '\001\002\000\000\040\255\020\000\000\003\000\000\000\344\112\075' ||
    '\013\102\312\121\300\156\303\050\010\036\041\105\100\047\277\105' ||
    '\047\113\312\121\300\366\173\142\235\052\041\105\100\225\174\354' ||
    '\056\120\312\121\300\160\231\323\145\061\041\105\100'
)::bytea);
```

Output

```
0102000020 ↩
    AD10000003000000E44A3D0B42CA51C06EC328081E21454027BF45274BCA51C0F67B629D2A214540957CEC2E50CA51C0
```

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.

Code

```
SELECT ST_GeomFromWKB(E'\\001\\002\\000\\000\\000\\002\\000\\000\\000\\037\\205\\353Q ↩
    \\270~\\\\\\300\\323Mb\\020X\\231C@\\020X9\\264\\310~\\\\\\300)\\\\\\217\\302\\365\\230 ↩
    C@', 4326);
```

Output

```
SRID=4326;LINESTRING(-113.98 39.198,-113.981 39.195)
```

Figure



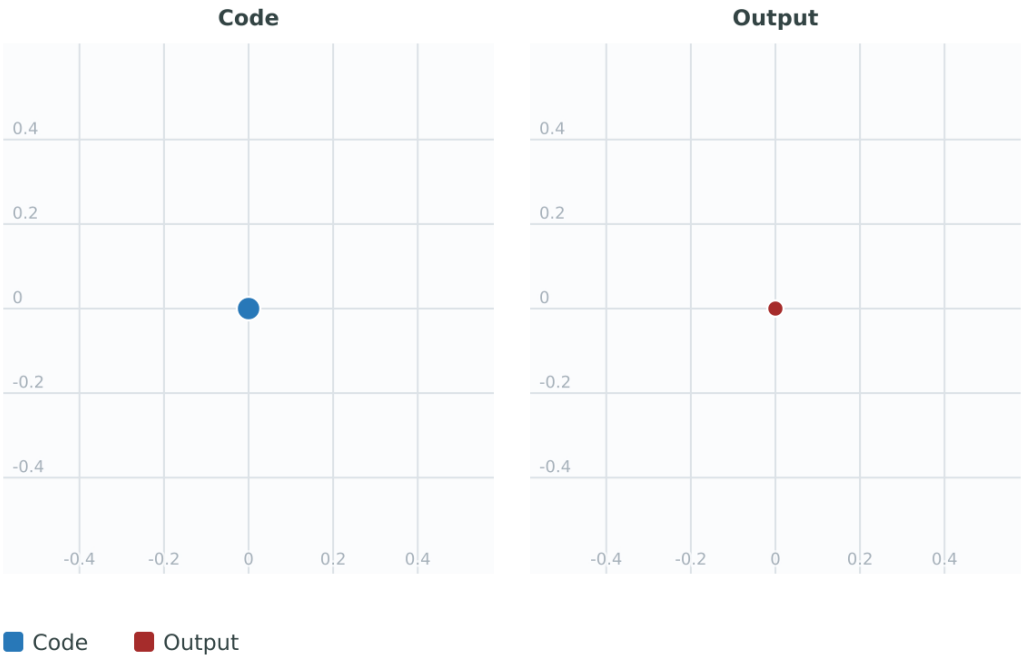
Code

```
SELECT ST_GeomFromWKB(ST_AsEWKB('POINT(2 5)'::geometry)
  );
```

Output

```
POINT(2 5)
```

Figure



The PostgreSQL decode function converts a hexadecimal WKB string to the bytea value accepted by ST_GeomFromWKB.

Code

```
SELECT ST_AsEWKT(ST_GeomFromWKB(
    decode('0101000000e5d022dbf93e2e40dbf97e6abc743540', 'hex'),
    4326
)) AS geom;
```

Output

```
SRID=4326;POINT(15.123 21.456)
```

相关信息

ST_WKBToSQL, ST_AsBinary, ST_AsEWKT, ST_GeomFromEWKB

7.8.2.4 ST_LineFromWKB

ST_LineFromWKB — Creates a LINESTRING from a WKB representation, optionally using a given SRID.

Synopsis

geometry **ST_LineFromWKB**(bytea WKB);
geometry **ST_LineFromWKB**(bytea WKB, integer srid);

描述

The ST_LineFromWKB function takes a Well-Known Binary (WKB) representation of a geometry and an optional Spatial Reference System ID (SRID), and creates a LINESTRING geometry. It is a type-specific geometry factory in SQL.

If an SRID is not specified, it defaults to 0. NULL is returned if the input bytea does not represent a LINESTRING geometry.



Note
OGC ☒范 3.2.6.2 - 可☒ SRID 用于☒范。



Note
If you know all your geometries are LINESTRINGs, it is more efficient to use ST_GeomFromWKB directly. This function calls ST_GeomFromWKB and adds validation that the result is a LINESTRING.



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

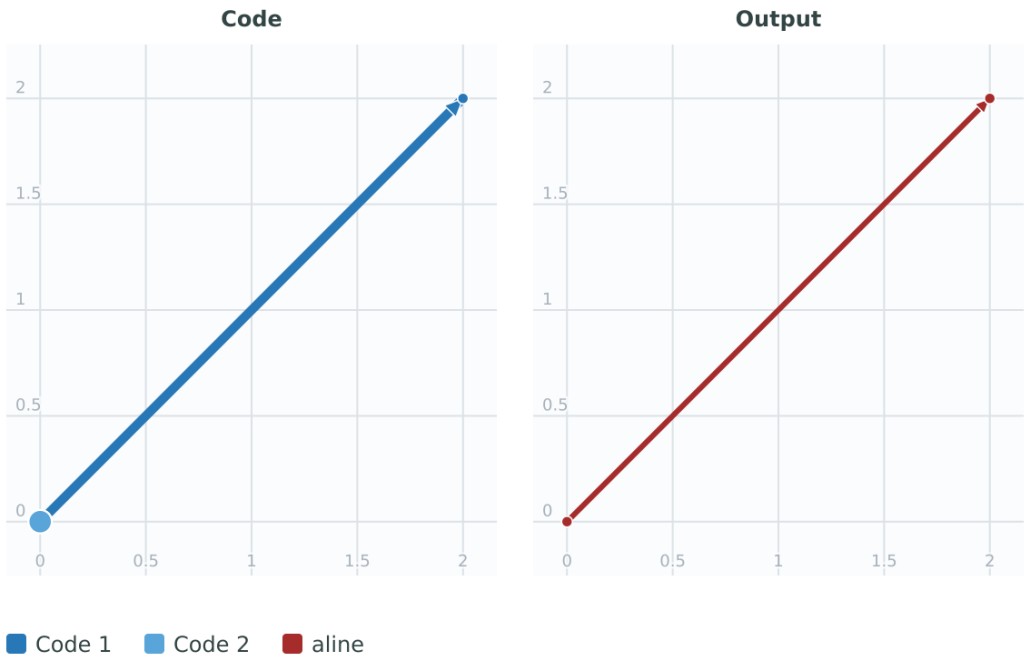
Code

```
SELECT ST_LineFromWKB(
    ST_AsBinary(ST_GeomFromText('LINESTRING(1 2,3 4)'))
) AS aline,
ST_LineFromWKB(
    ST_AsBinary(ST_GeomFromText('POINT(1 2)'))
) IS NULL AS null_return;
```

Output

```
aline                | null_return
-----------------------------------
LINESTRING(1 2,3 4)  | t
```

Figure



相关信息

ST_GeomFromWKB, ST_LinestringFromWKB

7.8.2.5 ST_LinestringFromWKB

ST_LinestringFromWKB — Creates a LINESTRING from a WKB representation, optionally using a given SRID.

Synopsis

geometry **ST_LinestringFromWKB**(bytea WKB);
geometry **ST_LinestringFromWKB**(bytea WKB, integer srid);

描述

The ST_LinestringFromWKB function takes a Well-Known Binary (WKB) representation of a geometry and an optional Spatial Reference System ID (SRID), and creates a LINESTRING geometry. It is an alias for ST_LineFromWKB.

If an SRID is not specified, it defaults to 0. NULL is returned if the input bytea does not represent a LINESTRING geometry.



Note
OGC ☒范 3.2.6.2 - 可☒ SRID 用于☒范。



Note
If you know all your geometries are LINESTRINGs, it is more efficient to use ST_GeomFromWKB directly. This function calls ST_GeomFromWKB and adds validation that the result is a LINESTRING.



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

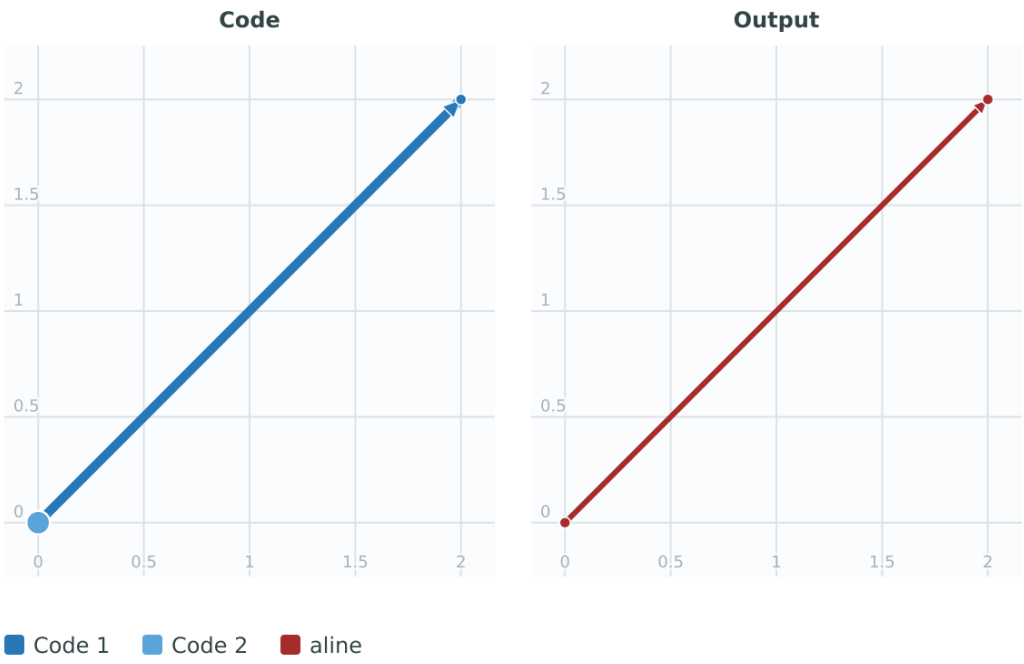
Code

```
SELECT ST_LinestringFromWKB(
    ST_AsBinary(ST_GeomFromText('LINESTRING(1 2,3 4)'))
) AS aline,
ST_LinestringFromWKB(
    ST_AsBinary(ST_GeomFromText('POINT(1 2)'))
) IS NULL AS null_return;
```

Output

```
aline               | null_return
------------------------------------
LINESTRING(1 2,3 4)  | t
```

Figure



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

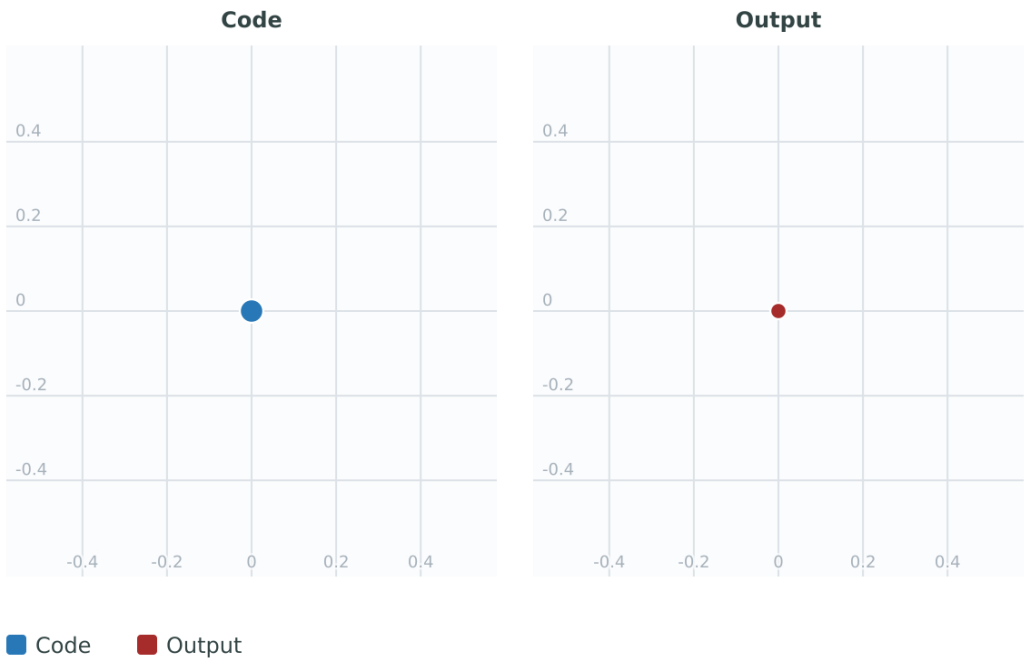
Code

```
SELECT ST_PointFromWKB(ST_AsEWKB('POINT(2 5)'::geometry));
```

Output

```
POINT(2 5)
```

Figure



Code

```
SELECT ST_PointFromWKB(ST_AsEWKB('LINESTRING(2 5,2 6)'::geometry));
```

Output

```
null
```

相关信息

ST_GeomFromWKB, ST_LineFromWKB

7.8.2.7 ST_WKBToSQL

ST_WKBToSQL — Returns a geometry from Well-Known Binary (WKB). SQL/MM alias for ST_GeomFromWKB without an SRID argument.

Synopsis

geometry **ST_WKBToSQL**(bytea WKB);

描述

This function is the SQL/MM alias for ST_GeomFromWKB with no SRID argument.

 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

Code

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0');
```

Output

```
BOX(-115.172816 36.114646,-115.172816 36.114646)
```

Code

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 0);
```

Output

```
BOX(-180 -90,180 90)
```

Code

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 10);
```

Output

```
BOX(-115.17282128334 36.1146408319473,-115.172810554504 36.1146461963654)
```

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

The cells are returned together so their relative sizes remain visible in a shared coordinate frame.

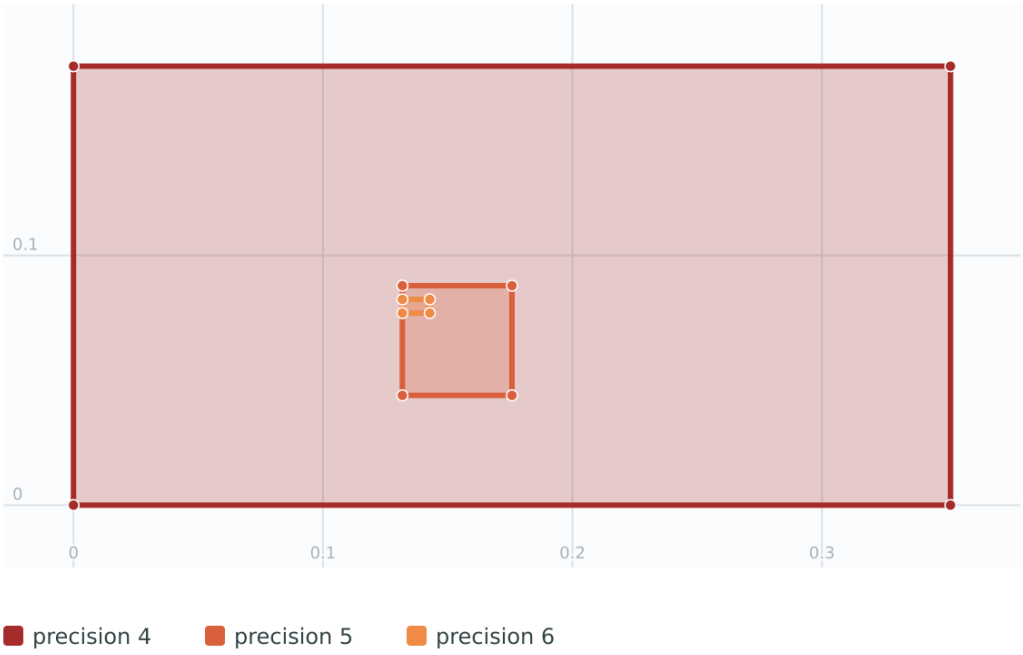
Code

```
SELECT ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 4) AS "precision 4", ST_GeomFromGeoHash ↩
    ('9qqj7nmxncgyy4d0dbxqz0', 5) AS "precision 5", ST_GeomFromGeoHash('9 ↩
    qqj7nmxncgyy4d0dbxqz0', 6) AS "precision 6";
```

Output

```
-[ RECORD 1 ]----------
precision 4 | POLYGON((-115.3125 36.03515625,-115.3125 36.2109375,-114.9609375 36.2109375,-114.9609375  ↩
     36.03515625,-115.3125 36.03515625))
precision 5 | POLYGON((-115.1806640625 36.0791015625,-115.1806640625 36.123046875,-115.13671875  ↩
     36.123046875,-115.13671875 36.0791015625,-115.1806640625 36.0791015625))
precision 6 | POLYGON((-115.1806640625 36.112060546875,-115.1806640625 36.1175537109375,-115.169677734375  ↩
     36.1175537109375,-115.169677734375 36.112060546875,-115.1806640625 36.112060546875))
```

Figure



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。

Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced.

☒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。

示例

A single geometry with srsName.

Code

```
SELECT ST_AsEWKT(ST_GeomFromGML($$
    <gml:LineString xmlns:gml="http://www.opengis.net/gml"
            srsName="EPSG:4269">
        <gml:coordinates>
    -71.16028,42.258729 -71.160837,42.259112 -71.161143,42.25932
        </gml:coordinates>
    </gml:LineString>
$$));
```

Output

```
SRID=4269;LINESTRING(-71.16028 42.258729,-71.160837 42.259112,-71.161143 42.25932)
```

Figure



XLink usage.

Code

```
        SELECT ST_AsEWKT(ST_GeomFromGML($$
            <gml:LineString xmlns:gml="http://www.opengis.net/gml"
                    xmlns:xlink="http://www.w3.org/1999/xlink"
                    srsName="EPSG:4269">
        <gml:pointProperty>
            <gml:Point gml:id="p1"
><gml:pos
>-71.16028 42.258729</gml:pos
></gml:Point>
        </gml:pointProperty>
        <gml:pos
>-71.160837 42.259112</gml:pos>
        <gml:pos
>-71.161143 42.258729</gml:pos>
                <gml:pointProperty>
                    <gml:Point xlink:type="simple" xlink:href="#p1"/>
                </gml:pointProperty>
            </gml:LineString>
        $$));
```

Output

```
SRID=4269;LINESTRING(-71.16028 42.258729,-71.160837 42.259112,-71.161143 42.258729,-71.16028 42.258729)
```

Figure



Polyhedral Surface.

Code

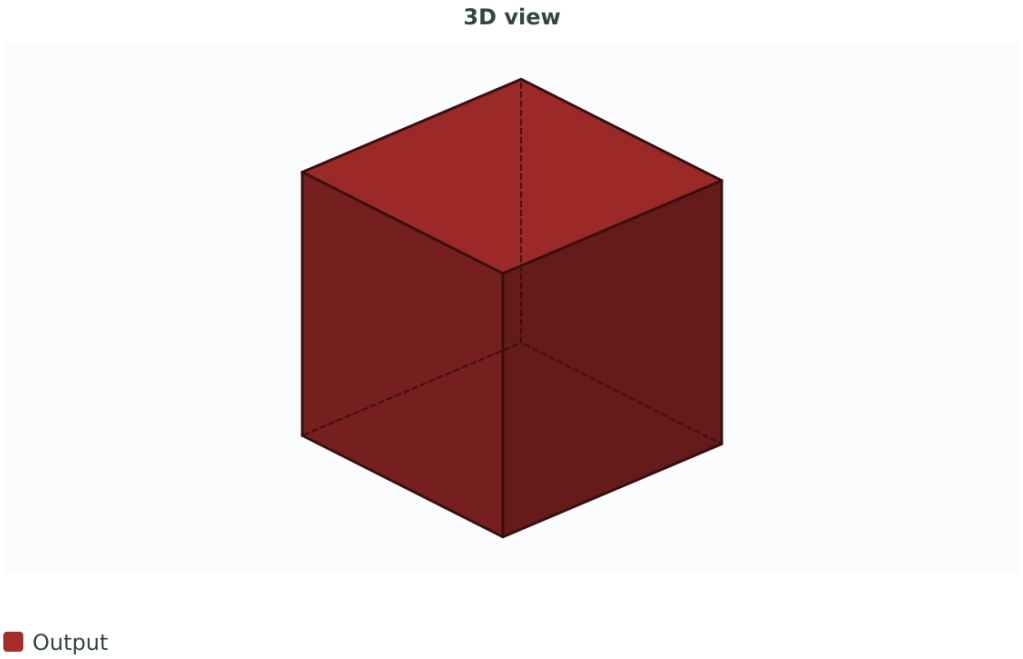
```
SELECT ST_AsEWKT(ST_GeomFromGML('
<gml:PolyhedralSurface xmlns:gml="http://www.opengis.net/gml">
<gml:polygonPatches>
  <gml:PolygonPatch>
    <gml:exterior>
      <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 0 0 1 0 1 1 0 1 0 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 0 1 0 1 1 0 1 0 0 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 0 1 0 0 1 0 1 0 0 1 0 0 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>1 1 0 1 1 1 1 0 1 1 0 0 1 1 0</gml:posList
></gml:LinearRing>
```

```
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 1 0 0 1 1 1 1 1 1 0 0 1 0</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
    <gml:LinearRing
><gml:posList srsDimension="3"
>0 0 1 1 0 1 1 1 1 0 1 1 0 0 1</gml:posList
></gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
</gml:polygonPatches>
</gml:PolyhedralSurface
>'));
```

Output

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

Figure



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

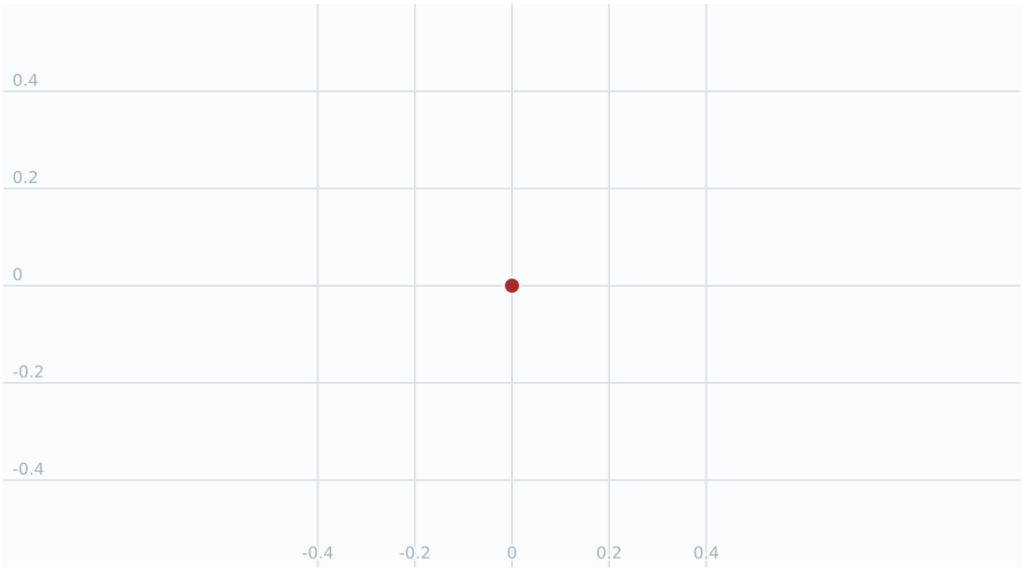
Code

```
SELECT ST_GeomFromGeoJSON('{"type":"Point","coordinates":[-48.23456,20.12345]}') AS wkt;
```

Output

```
POINT(-48.23456 20.12345)
```

Figure



This example creates a 3D LineString.

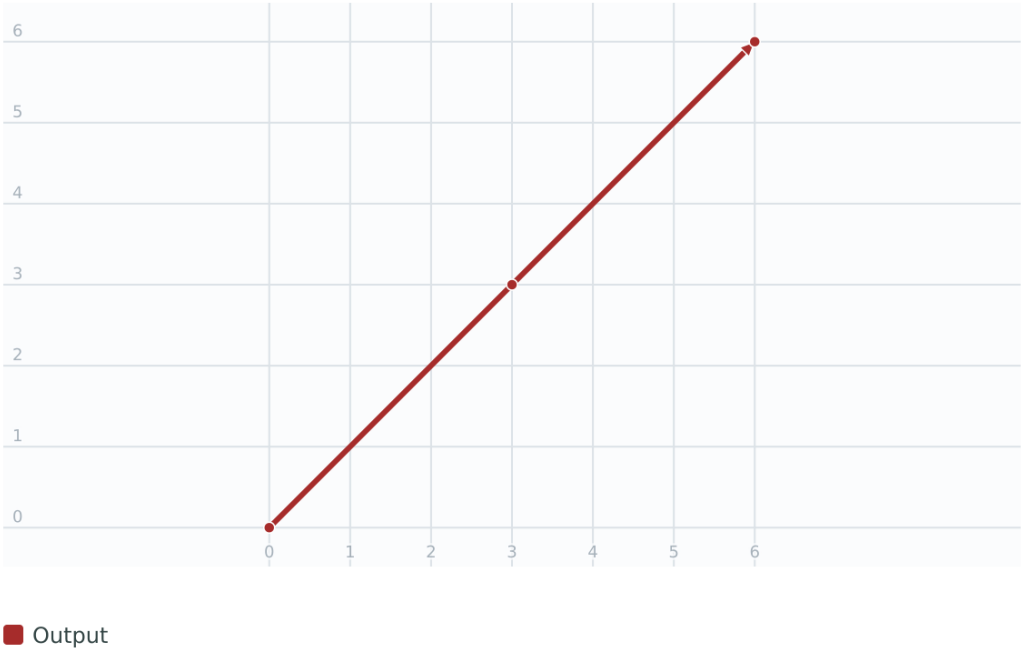
Code

```
SELECT ST_GeomFromGeoJSON('{"type":"LineString","coordinates":[[1,2,3],[4,5,6],[7,8,9]]}') ↩
    AS wkt;
```

Output

```
LINESTRING Z (1 2 3,4 5 6,7 8 9)
```

Figure



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

Constructs a PostGIS geometry from an OGC KML representation.
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 does not support SQL/MM curve geometries.

示例

A single LineString geometry.

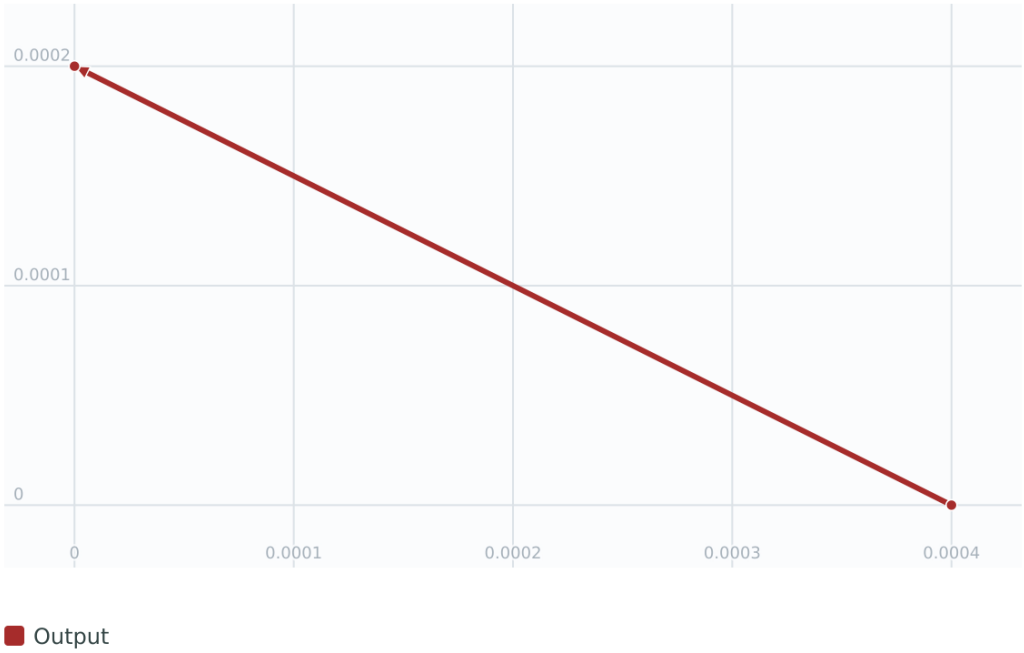
Code

```
        SELECT ST_AsEWKT(ST_GeomFromKML($$
            <LineString>
                <coordinates
>-71.1663, 42.2614
                    -71.1667, 42.2616</coordinates>
            </LineString>
        $$));
```

Output

```
SRID=4326;LINESTRING(-71.1663 42.2614,-71.1667 42.2616)
```

Figure



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

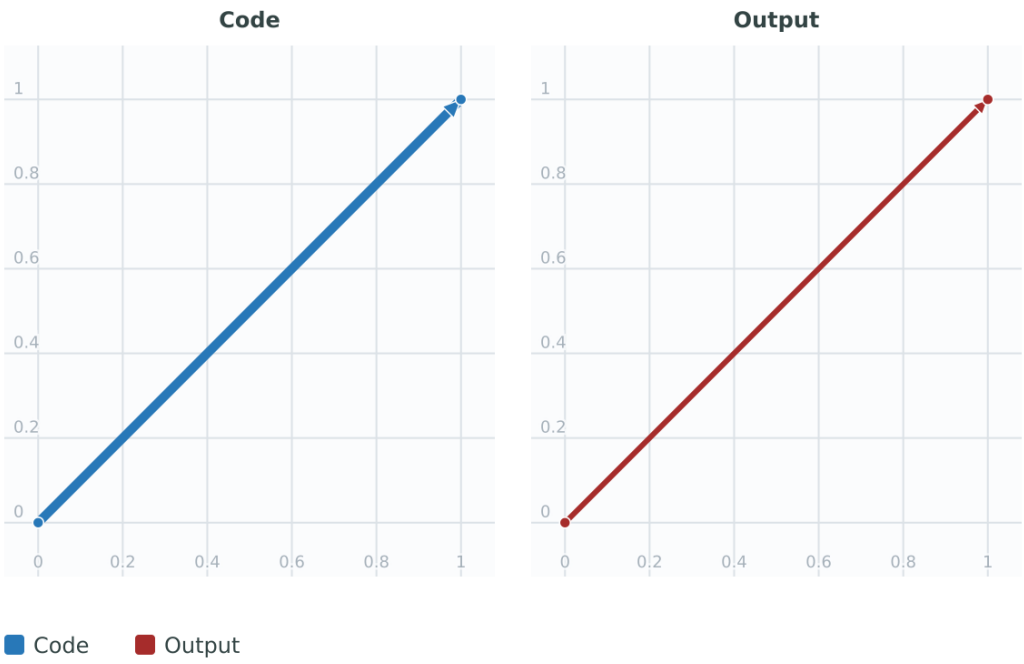
Code

```
SELECT ST_GeomFromTWKB(ST_AsTWKB('LINESTRING(126 34,127 35)'::geometry));
```

Output

```
LINESTRING(126 34, 127 35)
```

Figure



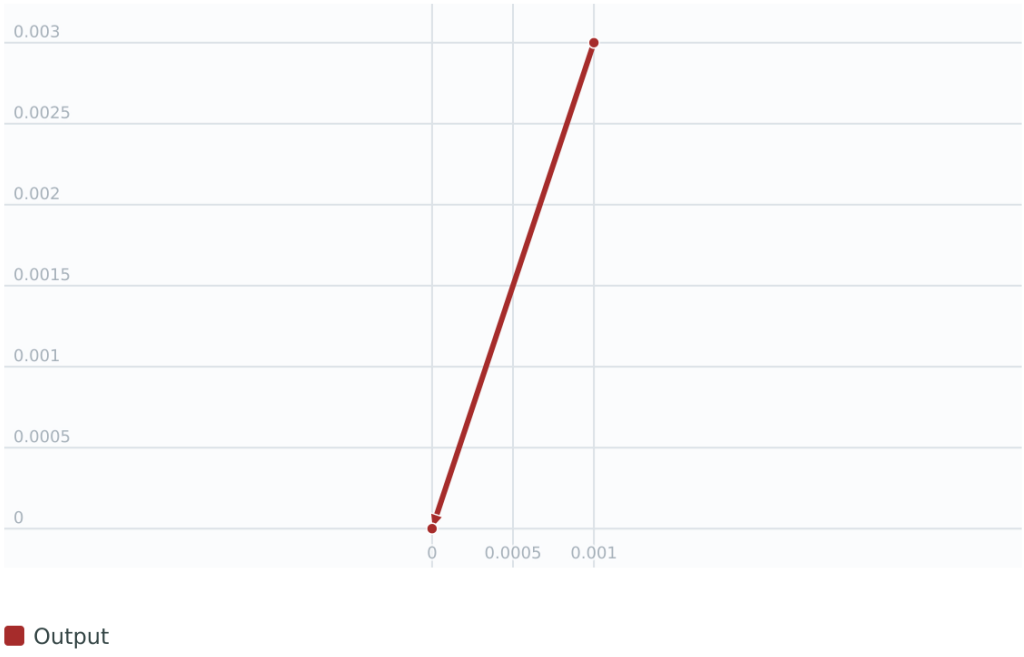
Code

```
SELECT ST_GeomFromTWKB(E'\\x620002f7f40dbce4040105');
```

Output

```
LINESTRING(-113.98 39.198,-113.981 39.195)
```

Figure



相关信息

ST_AsTWKB

7.8.3.7 ST_GMLToSQL

ST_GMLToSQL — Returns a geometry from GML. Alias for ST_GeomFromGML.

Synopsis

geometry **ST_GMLToSQL**(text geomgml);
geometry **ST_GMLToSQL**(text geomgml, integer srid);

描述

This function is an alias for ST_GeomFromGML and provides the same signatures.

 方法了 SQL/MM 范。SQL-MM 3: 5.1.50.

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。

Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced.

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

This example creates a LineString from a polyline.

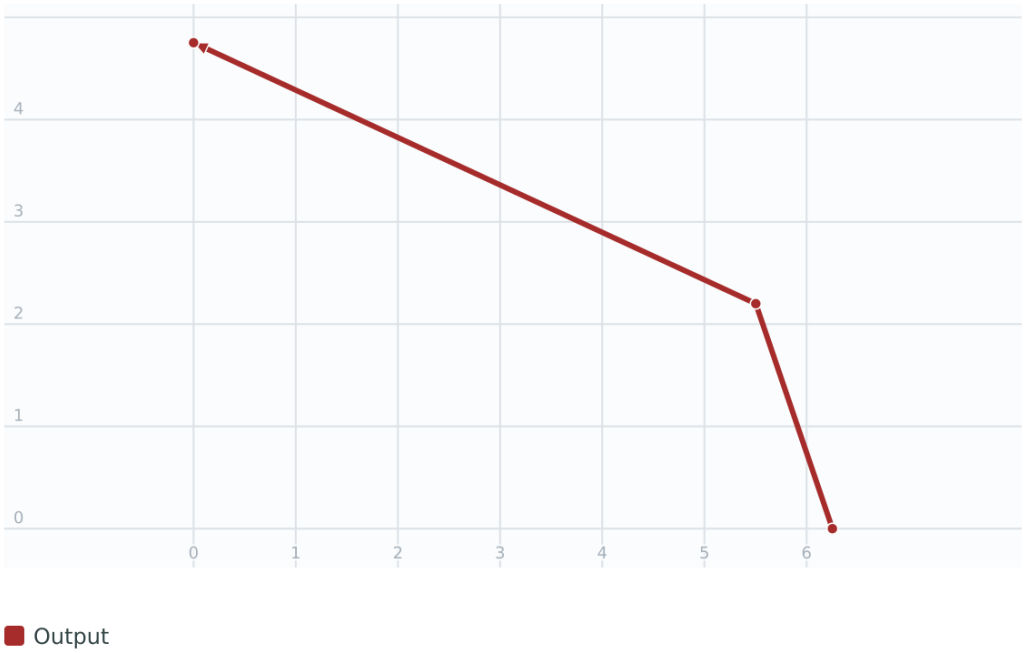
Code

```
SELECT ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@');
```

Output

```
SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)
```

Figure



This example selects a different precision than the one used for polyline encoding.

Code

```
SELECT ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@', 6);
```

Output

```
SRID=4326;LINESTRING(-12.02 3.85,-12.095 4.07,-12.6453 4.3252)
```

Figure



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

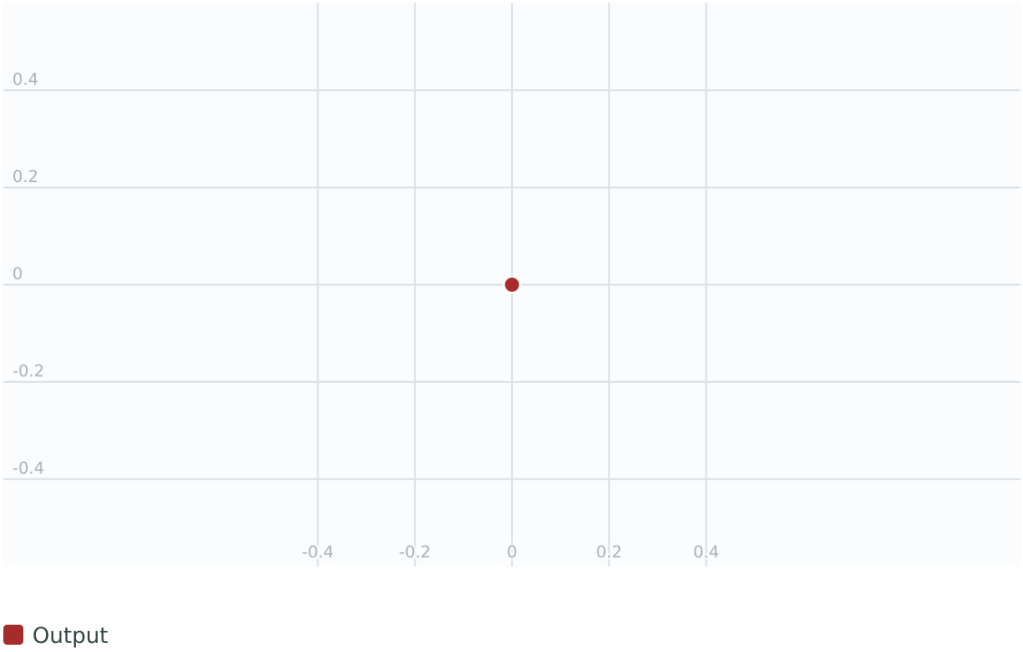
Code

```
SELECT ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0');
```

Output

```
POINT(-115.172816 36.114646)
```

Figure



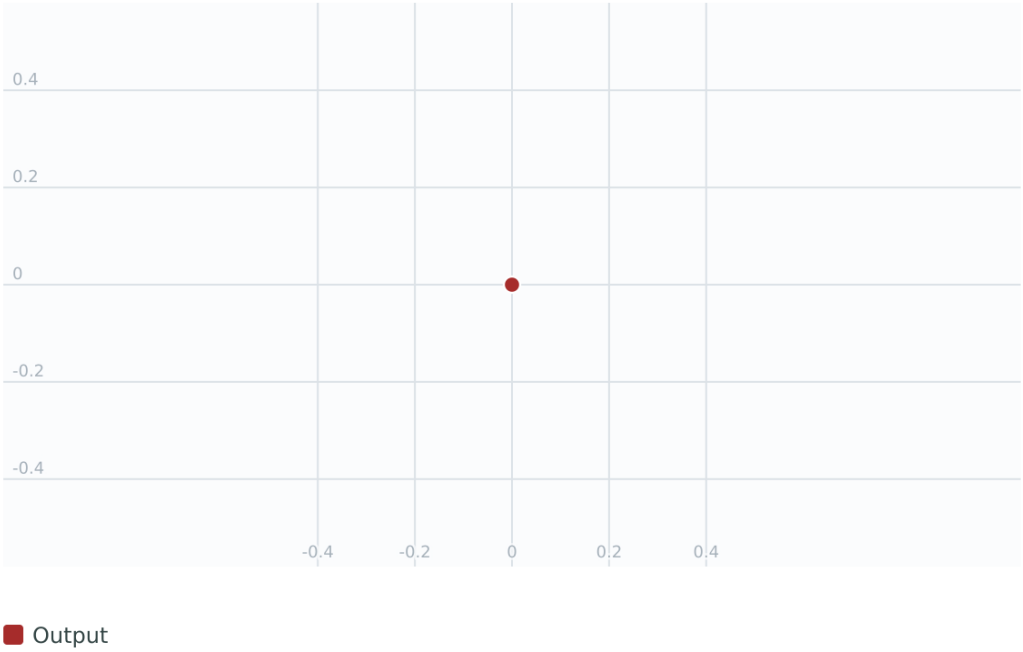
Code

```
SELECT ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 4);
```

Output

```
POINT(-115.13671875 36.123046875)
```

Figure



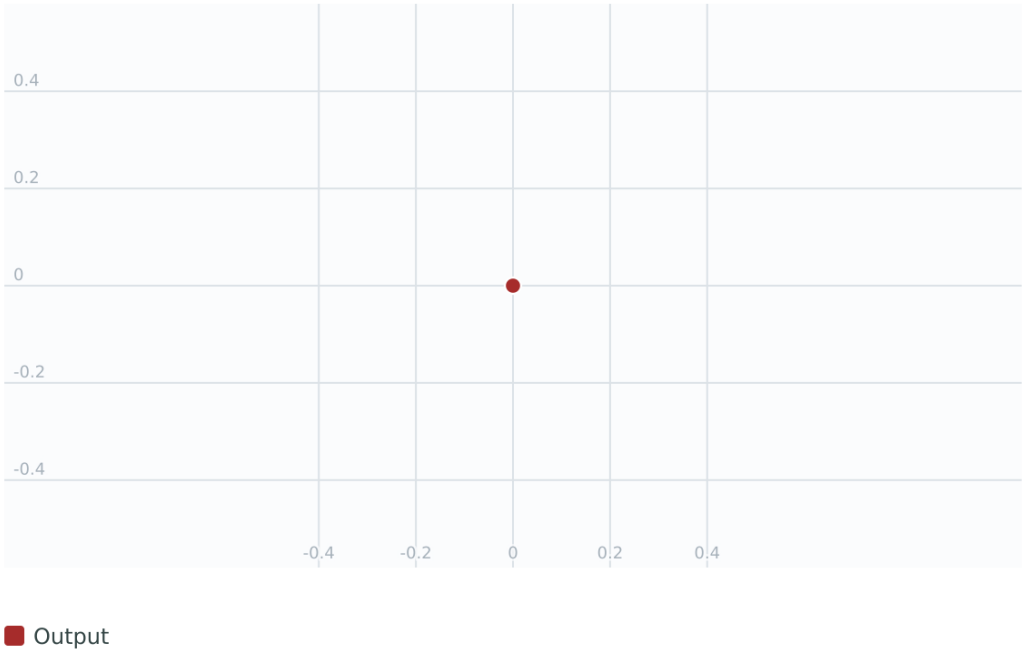
Code

```
SELECT ST_PointFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 10);
```

Output

```
POINT(-115.172815918922 36.1146435141563)
```

Figure



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 — 从取 FlatGeobuf 数据。

Synopsis

setof anyelement **ST_FromFlatGeobuf**(anyelement Table reference, bytea FlatGeobuf input data);

描述

从取 FlatGeobuf 数据（http://flatgeobuf.org）。注意：PostgreSQL bytea 类型不能超过 1GB。

`tabletype` 表类型的引用。

`data` 入 FlatGeobuf 数据。

可用性：3.2.0

7.9 几何输出

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

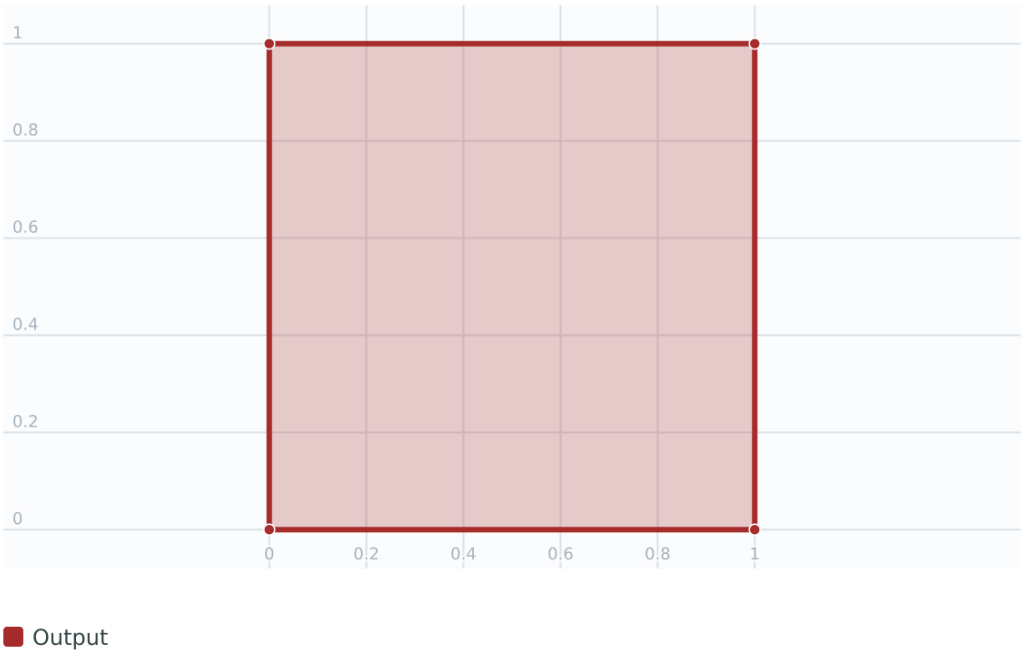
Code

```
SELECT ST_AsEWKT('0103000020 ↩
    e6100000010000000500000000000000000000000000000000000000000000000000000000000000000000000000f03f00000000
    '::geometry);
```

Output

```
SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
```

Figure



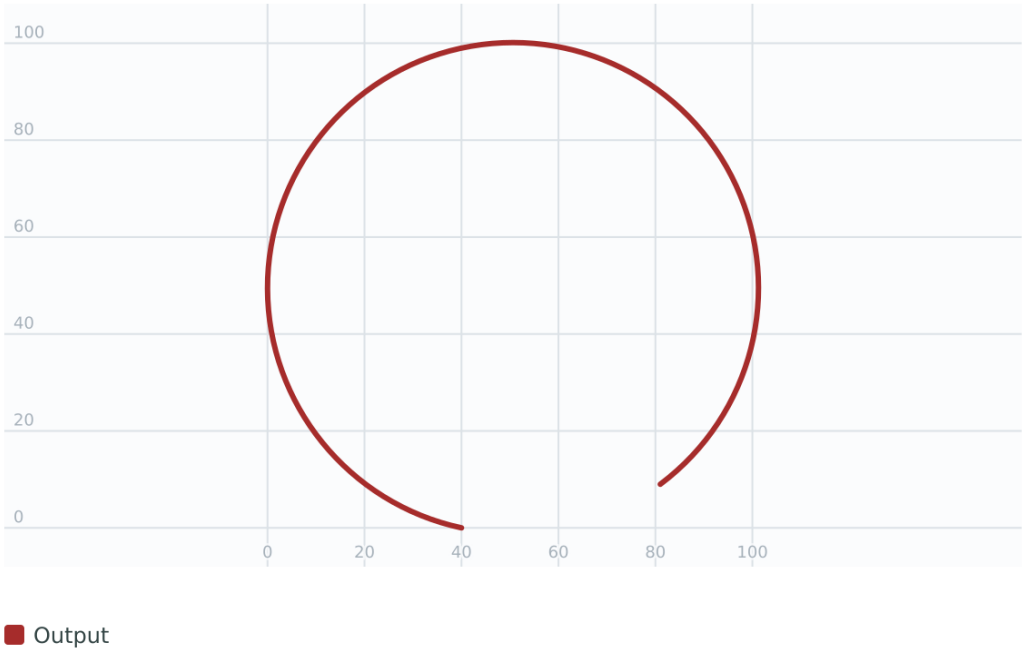
Code

```
SELECT ST_AsEWKT('0108000080030000000000000060 ↩
    E30A4100000000785C0241000000000000F03F0000000018E20A4100000000485F024100000000000000040000000000018
    ');
```

Output

```
CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)
```

Figure



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

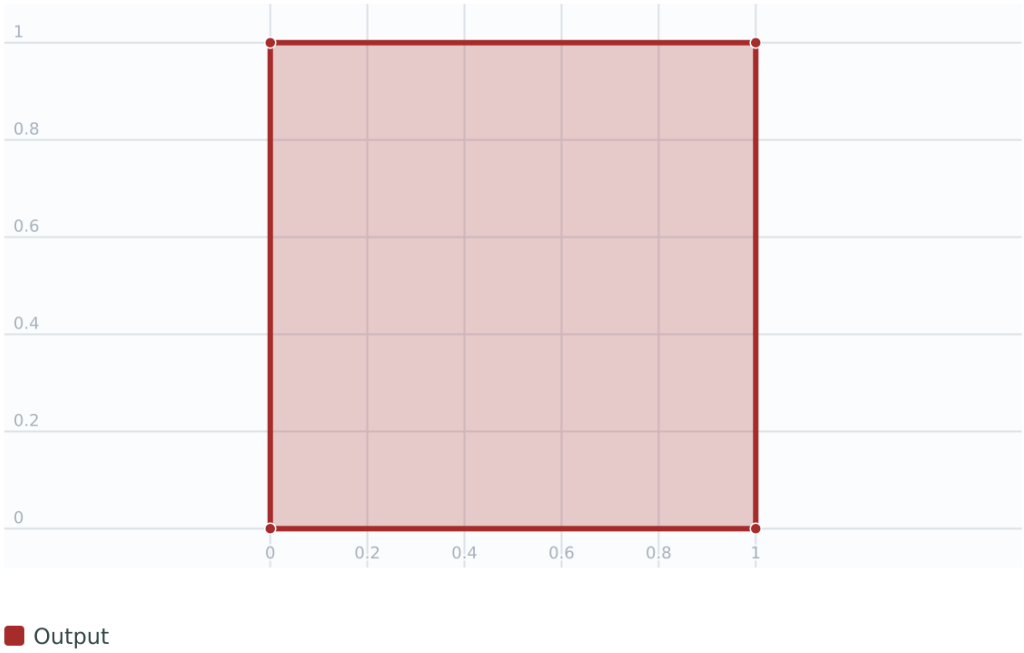
Code

```
SELECT ST_AsText(' ↩
    0103000000010000000500000000000000000000000000000000000000000000000000000000000000000000000000000000 ↩
    f03f000000000000f03f000000000000f03f000000000000f03f000000000000000000000000000000000000000000000000
    ');
```

Output

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

Figure



默⌧情况下⌧全精度⌧出。

Code

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'::geometry);
```

Output

```
POINT(111.1111111 1.1111111)
```

maxdecimaldigits 参数用于限制⌧出精度。

Code

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)'::geometry, 2);
```

Output

```
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

返回几何⊠象的 OGC/ISO Well-Known Binary (WKB) 表示形式。第一个函数形式默⊠使用服⊠器主机字⊠序⊠行⊠⊠。第二个函数形式接受一个文本参数，用于指定字⊠序⊠⊠方式：'NDR' 表示小端⊠⊠，'XDR' 表示大端⊠⊠。如⊠入未知参数，将默⊠生成小端⊠⊠⊠出。

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 默⊠以十六⊠制⊠⊠⊠出 bytea。如果你的 GUI 工具仍然需要旧的⊠出方式，可以在数据⊠中⊠行：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。

Changed: 2.0.0 Inputs to this function can not be unknown -- must be geometry. Constructs such as `ST_AsBinary('POINT(1 2)')` are no longer valid and you will get an `n ST_AsBinary(unknown) is not unique error`. Code like that needs to be changed to `ST_AsBinary('POINT(1 2)'::geometry);`. If that is not possible, then install `legacy.sql`.

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

Code

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
\x0103000000010000000500000000000000000000000000000000000000000000000000f03f000000000000f03f000000000000f03f000000000000000000000000000000f03f000000000000000000000000000000000000000000000000
```

Code

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326), 'XDR');
```

Output

```
\x0000000003000000010000000500000000000000000000000000000000000000000000000000f03ff000000000003ff0000000000000003ff0000000000000003ff0000000000000000000000000000000000000000000000000
```

Use the PostgreSQL encode function to produce a hexadecimal WKB string without the \x prefix used for bytea display. Specifying NDR makes the byte order deterministic.

Code

```
SELECT encode(
    ST_AsBinary('POINT(15.123 21.456)'::geometry, 'NDR'),
    'hex'
) AS wkb_hex;
```

Output

```
0101000000e5d022dbf93e2e40dbf97e6abc743540
```

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

返回☒有 SRID 元数据的 Extended Well-Known Binary（EWKB）格式的几何☒象表示。第一个函数形式默☒使用服☒器主机的字☒序☒行☒☒。第二个函数形式接受一个文本参数，用于指定字☒序☒☒方式：'NDR' 表示小端☒☒，'XDR' 表示大端☒☒。如☒入未知参数，将默☒☒出小端☒☒格式。

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

Code

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
\x0103000020e6100000010000000500000000000000000000000000000000000000000000000000000000000000000000
000000000000000f03f000000000000f03f000000000000f03f000000000000f03f000000000000000000000000000000000
0000000000000000000000000000000000000000
```

Code

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326), 'XDR');
```

Output

```
\x0020000003000010e600000001000000050000000000000000000000000000000000000000000000000000000000000000
003ff0000000000003ff0000000000003ff0000000000003ff000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000
```

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。

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

Code

```
SELECT ST_AsHEXEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
0103000020E610000010000000500
00000000000000000000000000000000
00000000000000000000000000000F03F
000000000000F03F000000000000F03F000000000000F03
F000000000000000000000000000000000000000000000000
```

Which gives the same answer as:

Code

```
SELECT ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326)::text;
```

7.9.3 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

Returns the geometry in the Encoded Polyline Algorithm Format. The format stores latitude and longitude deltas as integers after multiplying coordinates by 10 raised to the requested `precision`.

The informal names *polyline5* and *polyline6* refer to the same encoding with precision 5 and 6. The precision must match when encoding and decoding; otherwise the coordinates are scaled incorrectly.

The input geometry must use SRID 4326, with longitude as X and latitude as Y. Use ST_Transform to transform geometries in other coordinate systems before encoding.

The OSRM API returns precision 5 for `geometries=polyline` and precision 6 for `geometries=polyline6`. Decode the resulting `route.geometry` with ST_LineFromEncodedPolyline using the matching precision.

可用性：2.2.0

示例

Encode the same WGS 84 LineString using precision 5 and 6.

Code

```
WITH route AS (
  SELECT ST_GeomFromEWKT(
    'SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)'
  ) AS geom
)
SELECT ST_AsEncodedPolyline(geom, 5) AS polyline5,
       ST_AsEncodedPolyline(geom, 6) AS polyline6
FROM route;
```

Output

```
polyline5          |           polyline6
---------------------------+--------------------------------
 _p~iF~ps|U_ulLnnqC_mqNvxq`@ | _izlhA~rlgdF_{geC~ywl@_kwzCn`{nI
```

Decode an OSRM `route.geometry` returned with `geometries=polyline6`.

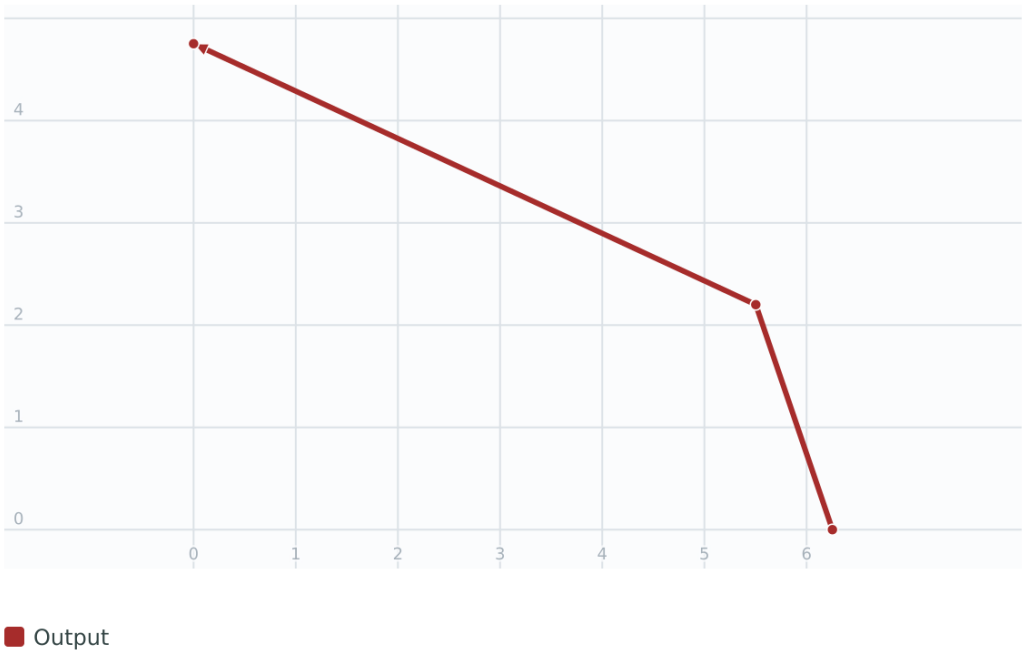
Code

```
SELECT ST_AsEWKT(ST_LineFromEncodedPolyline(
  '_izlhA~rlgdF_{geC~ywl@_kwzCn`{nI',
  6
)) AS route;
```

Output

```
SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)
```

Figure



The Google Maps JavaScript Geometry library decodes precision-5 strings. This browser example assumes that `map` is an initialized Google map.

Code

```
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&libraries=geometry"
></script>
<script>
  const encodedPath = "_p~iF~ps|U_ulLnnqC_mqNvxq`@";
  const flightPath = new google.maps.Polyline({
    path: google.maps.geometry.encoding.decodePath(encodedPath),
    map: map,
    strokeColor: "#0000CC",
    strokeOpacity: 1.0,
    strokeWeight: 4
  });
</script>
```

相关信息

ST_LineFromEncodedPolyline, ST_Transform

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

Code

```
SELECT encode(ST_AsGeobuf(q, 'geom'), 'base64')
FROM (SELECT 'POLYGON((0 0,0 1,1 1,1 0,0 0))'::geometry AS geom) AS q;
```

Output

```
GAAiEAoOCgwIBBoIAAAAAgIAAAE=
```

7.9.3.4 ST_AsGeoJSON

ST_AsGeoJSON — 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。

当⍰入⍰⍰中包含重复的列名⍰，生成的 "properties" ⍰象也会出⍰重复的⍰。在⍰种情况下，`ST_AsGeoJSON` 会⍰出警告，因⍰ PostgreSQL 的 jsonb ⍰于重复⍰只会保留最后一个⍰。

The GeoJSON specification states that polygons are oriented using the Right-Hand Rule, and some clients require this orientation. This can be ensured by using ST_ForcePolygonCCW. The specification also requires that geometry be in the WGS84 coordinate system (SRID = 4326). If necessary geometry can be projected into WGS84 using ST_Transform: `ST_Transform(geom, 4326 )`.

GeoJSON 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

Changed: 3.7.0 -- 增加了关于重复⍰的警告

 ⍰函数支持 3d 并且不会⍰失 z-index。

示例

生成特征集合：

Code

```
SELECT json_build_object(
    'type', 'FeatureCollection',
    'features', json_agg(ST_AsGeoJSON(t.*, id_column =
> 'id')::json)
    )
FROM ( VALUES (1, 'one', 'POINT(1 1)'::geometry),
              (2, 'two', 'POINT(2 2)'),
              (3, 'three', 'POINT(3 3)')
     ) as t(id, name, geom);
```

Output

```
{"type" : "FeatureCollection", "features" : [{"type": "Feature", "geometry": {"type":"Point","coordinates":[1,1]}, "id ↩
    ": 1, "properties": {"name": "one"}}, {"type": "Feature", "geometry": {"type":"Point","coordinates":[2,2]}, "id": ↩
     2, "properties": {"name": "two"}}, {"type": "Feature", "geometry": {"type":"Point","coordinates":[3,3]}, "id":  ↩
    3, "properties": {"name": "three"}}]}
```

生成一个特征：

Code

```
SELECT ST_AsGeoJSON(t.*, id_column =
> 'id')
FROM (VALUES (1, 'one', 'POINT(1 1)'::geometry)) AS t(id, name, geom);
```

Output

```
{"type": "Feature", "geometry": {"type":"Point","coordinates":[1,1]}, "id": 1, "properties": {"name": "one"}}
```

不要忘⊠将数据⊠⊠⊠ WGS84 ⊠度、⊠度以符合 GeoJSON ⊠范：

Code

```
SELECT ST_AsGeoJSON(ST_Transform(geom, 4326)) FROM fe_edges LIMIT 1;
```

Output

```
{"type":"MultiLineString","coordinates":[[[-89.734634999999997,31.492072000000000],
[-89.734955999999997,31.492237999999997]]]}
```

支持 3D 几何形状：

Code

```
SELECT ST_AsGeoJSON('LINESTRING(1 2 3,4 5 6)');
```

Output

```
{"type":"LineString","coordinates":[[1,2,3],[4,5,6]]}
```

Options 参数用于在 GeoJSON ⊠出中添加 BBOX 或 CRS：

Code

```
SELECT ST_AsGeoJSON(ST_SetSRID('POINT(1 1)'::geometry, 4326), 9, 4|1);
```

Output

```
{"type":"Point","crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:EPSG::4326"}},"bbox ↩
    ":[1.000000000,1.000000000,1.000000000,1.000000000],"coordinates":[1,1]}
```

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

Version 2.

Code

```
SELECT ST_AsGML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
<gml:Polygon srsName="EPSG:4326"
><gml:outerBoundaryIs
><gml:LinearRing
><gml:coordinates
>0,0 0,1 1,1 1,0 0,0</gml:coordinates
></gml:LinearRing
></gml:outerBoundaryIs
></gml:Polygon
>
```

Version 3.

Flip coordinates and output extended EPSG (16 | 1)

Code

```
SELECT ST_AsGML(3, ST_GeomFromText('POINT(5.234234233242 6.34534534534)', 4326), 5, 17);
```

Output

```
<gml:Point srsName="urn:ogc:def:crs:EPSG::4326"
><gml:pos srsDimension="2"
>6.34535 5.23423</gml:pos
></gml:Point
>
```

Output the envelope (32).

Code

```
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 32);
```

Output

```
<gml:Envelope srsName="EPSG:4326" srsDimension="2"
><gml:lowerCorner
>1 2</gml:lowerCorner
><gml:upperCorner
>10 20</gml:upperCorner
></gml:Envelope
>
```

Output the envelope (32), reverse (lat lon instead of lon lat) (16), long srs (1) = 32 | 16 | 1 = 49.

Code

```
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 49);
```

Output

```
<gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326" srsDimension="2"
><gml:lowerCorner
>2 1</gml:lowerCorner
><gml:upperCorner
>20 10</gml:upperCorner
></gml:Envelope
>
```

This example uses a PolyhedralSurface.

Code

```
SELECT ST_AsGML(
3,
'POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )'::geometry);
```

Output

```
 <gml:PolyhedralSurface>
<gml:polygonPatches>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 0 0 0 0 1 0 1 1 0 1 0 0 0 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 0 0 0 1 0 1 1 0 1 0 0 0 0 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 0 0 1 0 0 1 0 1 0 0 1 0 0 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>1 1 0 1 1 1 1 0 1 1 0 0 1 1 0</gml:posList>
        </gml:LinearRing>
    </gml:exterior>
   </gml:PolygonPatch>
   <gml:PolygonPatch>
    <gml:exterior>
        <gml:LinearRing>
           <gml:posList srsDimension="3"
>0 1 0 0 1 1 1 1 1 1 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。

可用性：1.2.2 - 包含版本参数的更高版本出⊠在 1.3.2 中

增⊠：2.0.0 - 添加前⊠命名空⊠，使用默⊠和命名参数

更改：3.0.0 - ⊠除了“版本化”⊠体⊠名

Note
需要在 Proj 支持下⊠⊠ PostGIS。使用 PostGIS_Full_Version确⊠您已⊠⊠⊠目支持。

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

Code

```
SELECT ST_AsKML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))', 4326));
```

Output

```
<Polygon
><outerBoundaryIs
><LinearRing
><coordinates
>0,0 0,1 1,1 1,0 0,0</coordinates
></LinearRing
></outerBoundaryIs
></Polygon
>
```

This example outputs a 3D LineString.

Code

```
SELECT ST_AsKML('SRID=4326;LINESTRING(1 2 3,4 5 6)');
```

Output

```
<LineString
><coordinates
>1,2,3 4,5,6</coordinates
></LineString
>
```

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。

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)'));
```

Output

```
2°19'29.928"S 3°14'3.243"W
```

指定格式（与默⌧⌧相同）。

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D°M''S.SSS"C'));
```

Output

```
2°19'29.928"S 3°14'3.243"W
```

除 D、M、S 和 C 以外的字符⌧通⌧。

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D degrees, M minutes, S seconds to ↩
    the C'));
```

Output

```
2 degrees, 19 minutes, 30 seconds to the S 3 degrees, 14 minutes, 3 seconds to the W
```

有符号的度数而不是基本方向。

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D°M''S.SSS"'));
```

Output

```
-2°19'29.928" -3°14'3.243"
```

十⌧制度数。

Code

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D.DDDD degrees C'));
```

Output

```
2.3250 degrees S 3.2342 degrees W
```

To convert parsed degree, minute, and second components to decimal degrees, apply the cardinal-direction sign explicitly.

Code

```
WITH dms(label, degrees, minutes, seconds, hemisphere) AS (
  VALUES
    ('latitude', 43::numeric, 0::numeric, 50.60::numeric, 'S'),
    ('longitude', 147::numeric, 12::numeric, 18.20::numeric, 'E')
)
SELECT label,
       round(
         (CASE WHEN hemisphere IN ('S', 'W') THEN -1 ELSE 1 END)
         * (degrees + minutes / 60 + seconds / 3600),
         9
       ) AS decimal_degrees
FROM dms;
```

Output

```
   label    | decimal_degrees
------------+-----------------
 latitude   |   -43.014055556
 longitude  |   147.205055556
```

⊠大的⊠被⊠一化。

Code

```
SELECT (ST_AsLatLonText('POINT (-302.2342342 -792.32498)'));
```

Output

```
72°19'29.928"S 57°45'56.757"E
```

7.9.3.8 ST_AsMARC21

ST_AsMARC21 — 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☒）

Code

```
SELECT ST_AsMARC21('SRID=4326;POINT(-4.504289 54.253312)'::geometry);
```

Output

```
                              st_asmarc21
-----------------------------------------------------
<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>W0043015</subfield>
        <subfield code="e"
>W0043015</subfield>
        <subfield code="f"
>N0541512</subfield>
        <subfield code="g"
>N0541512</subfield>
    </datafield>
</record>
```

将 POLYGON ☒☒☒十☒制度格式的 MARC21/XML

Code

```
SELECT ST_AsMARC21('SRID=4326;POLYGON((-4.5792388916015625 ↩
    54.18172660239091,-4.56756591796875 54.196993557130355,-4.546623229980469 ↩
    54.18313300502024,-4.5792388916015625 54.18172660239091))'::geometry,'hddd.dddd');

<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>W004.5792</subfield>
        <subfield code="e"
>W004.5466</subfield>
        <subfield code="f"
>N054.1970</subfield>
        <subfield code="g"
>N054.1817</subfield>
    </datafield>
</record>
```

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☒☒。

Code

```
SELECT ST_AsMARC21('SRID=4326;GEOMETRYCOLLECTION(POLYGON((13.1 52.65,13.516666666666667 ↩
    52.65,13.516666666666667 52.38333333333333,13.1 52.38333333333333,13.1 52.65)),POINT ↩
    (-4.5 54.25))'::geometry,'hdddmm.mmmm');
```

Output

```
                                st_asmarc21
--------------------------------------------------
<record xmlns="http://www.loc.gov/MARC21/slim">
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>E01307.0000</subfield>
        <subfield code="e"
>E01331.0000</subfield>
        <subfield code="f"
>N05240.0000</subfield>
        <subfield code="g"
>N05224.0000</subfield>
    </datafield>
    <datafield tag="034" ind1="1" ind2=" ">
        <subfield code="a"
>a</subfield>
        <subfield code="d"
>W00430.0000</subfield>
        <subfield code="e"
>W00430.0000</subfield>
        <subfield code="f"
>N05415.0000</subfield>
        <subfield code="g"
>N05415.0000</subfield>
    </datafield>
</record>
```

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。

The function performs geometry simplification appropriate to the integer tile coordinate space. After transforming to tile coordinates it snaps coordinates to the integer grid, removes repeated points, and removes points that lie on straight lines. It does not perform general-purpose, scale-dependent simplification such as ST_Simplify. To reduce detail before tile conversion, simplify the geometry in map coordinates before calling `ST_AsMVTGeom`.

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 区范定区的切片坐区空区中切片的大小。默区区区 4096。

`buffer` 是切片坐区空区中用于裁剪几何区形的区冲区大小。默区区区 256。

`Clip_geom` 是一个布区区，用于控制几何区形是否按原区剪切或区区。默区区 true。

可用性：2.4.0

> **Note**
> 从 3.0 开始，可以在配置区区区 Wagyu 来剪区和区区 MVT 多区形。区区比 GEOS 默区区速度更快，区生的区果更正确，但它可能会区失小多区形。

示例

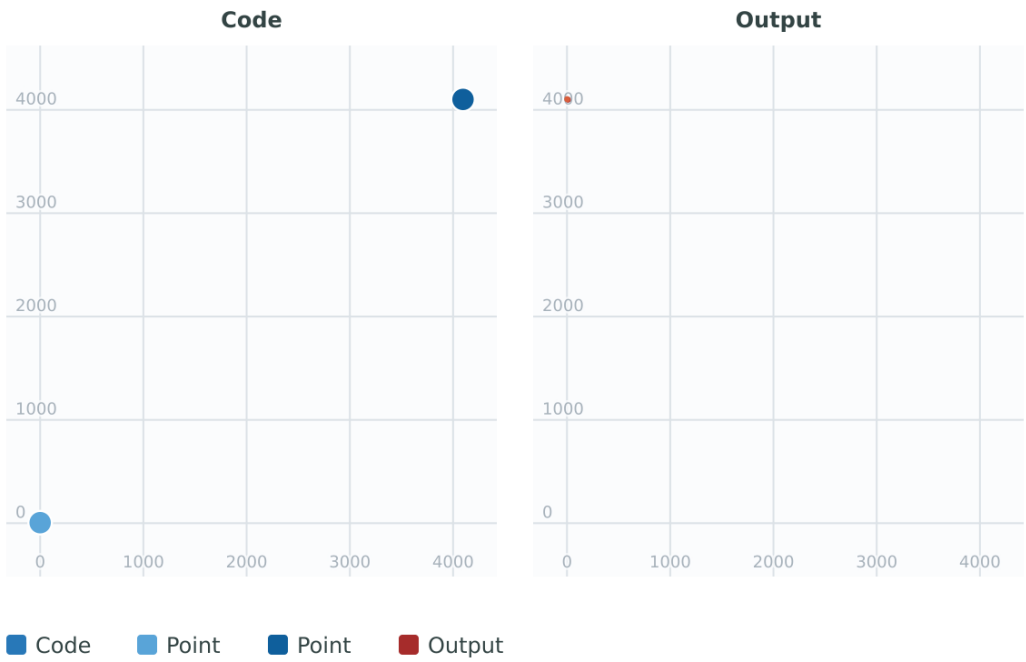
Code

```
SELECT ST_AsMVTGeom('POLYGON ((0 0,10 0,10 5,0 -5,0 0))',
    ST_MakeBox2D(ST_Point(0, 0), ST_Point(4096, 4096)),
    4096, 0, false);
```

Output

```
MULTIPOLYGON(((5 4096,10 4091,10 4096,5 4096)),((5 4096,0 4101,0 4096,5 4096)))
```

Figure



使用区算出的瓦片区界区行区区和裁剪几何的 Web Mercator 瓦片典型示例。本示例假区 data.geom 列的 SRID 区 4326。

Code

```
SELECT ST_AsMVTGeom(ST_Transform(geom, 3857 ),
            ST_TileEnvelope(12, 513, 412),
            extent =
> 4096,
```

```
            buffer =
> 64) AS geom
  FROM data
  WHERE geom && ST_Transform(ST_TileEnvelope(12, 513, 412, margin =
> (64.0 / 4096)), 4326)
```

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

Code

```
WITH mvtgeom AS
(
  SELECT ST_AsMVTGeom(geom, ST_TileEnvelope(12, 513, 412), extent =
> 4096, buffer =
> 64) AS geom, name, description
  FROM points_of_interest
  WHERE geom && ST_TileEnvelope(12, 513, 412, margin =
> (64.0 / 4096))
)
SELECT ST_AsMVT(mvtgeom.*)
FROM mvtgeom;
```

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

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

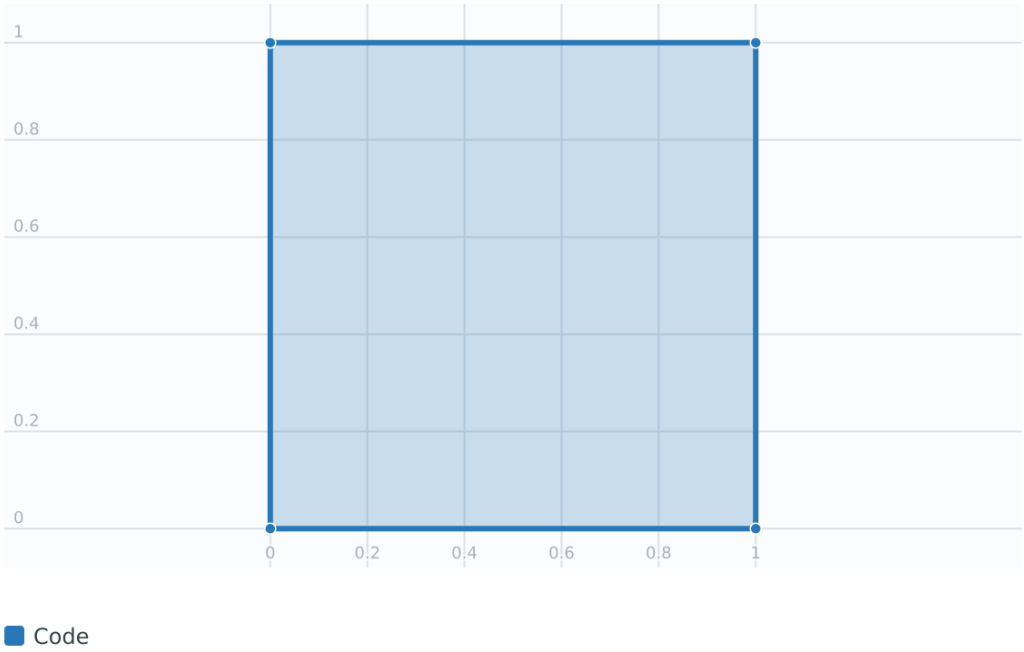
Code

```
SELECT ST_AsSVG('POLYGON((0 0,0 1,1 1,1 0,0 0))'::geometry);
```

Output

```
M 0 0 L 0 -1 1 -1 1 0 Z
```

Figure



⊠弧

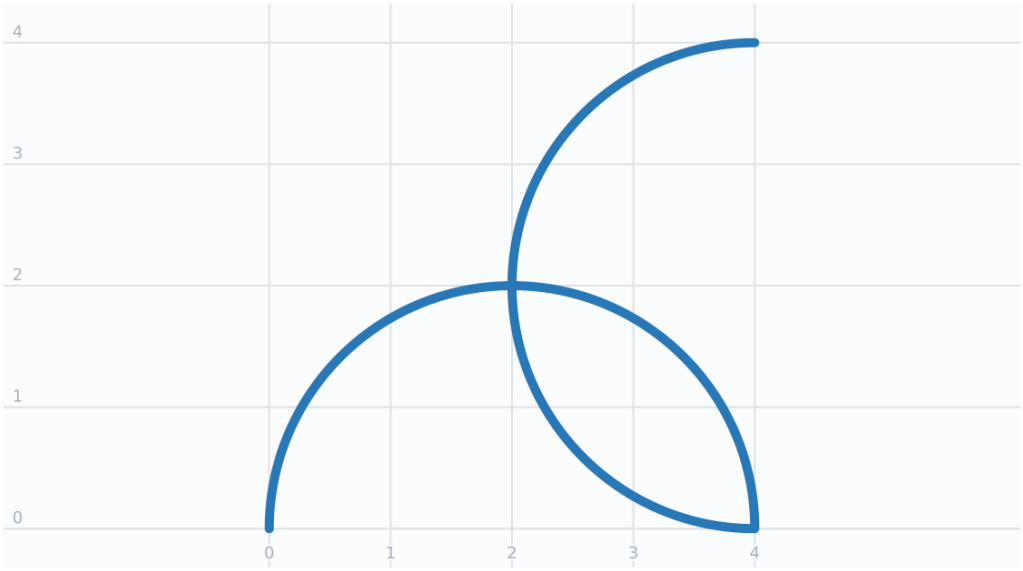
Code

```
SELECT ST_AsSVG('CIRCULARSTRING(-2 0,0 2,2 0,0 2,2 4)'::geometry);
```

Output

```
M -2 0 A 2 2 0 0 1 2 0 A 2 2 0 0 1 2 -4
```

Figure



多曲区

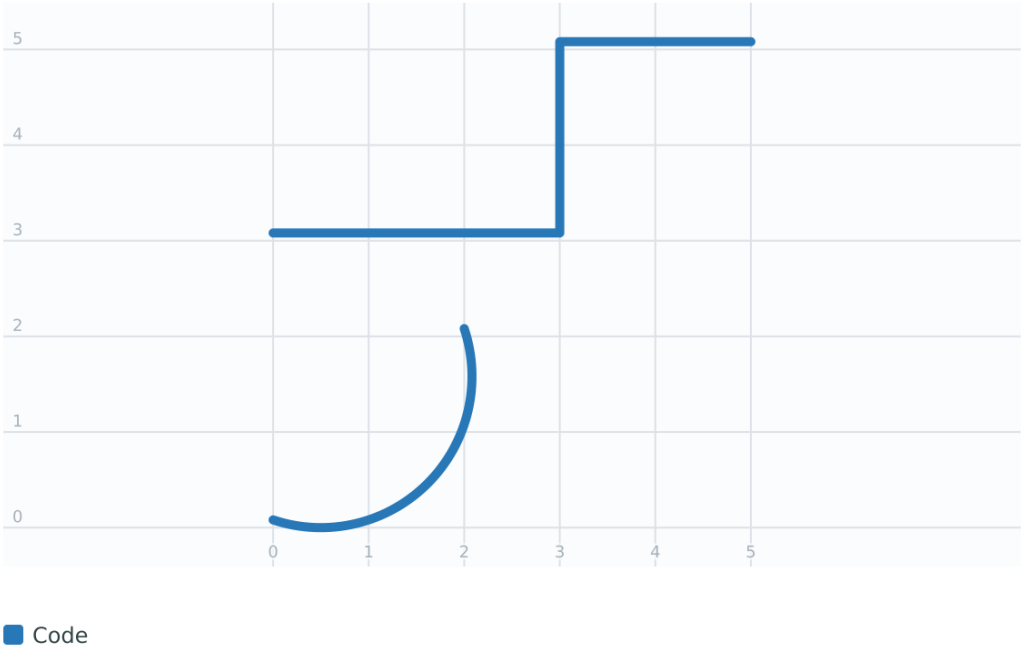
Code

```
SELECT ST_AsSVG('MULTICURVE((5 5,3 5,3 3,0 3),
CIRCULARSTRING(0 0,2 1,2 2))'::geometry, 0, 0);
```

Output

```
M 5 -5 L 3 -5 3 -3 0 -3 M 0 0 A 2 2 0 0 0 2 -2
```

Figure



多曲面

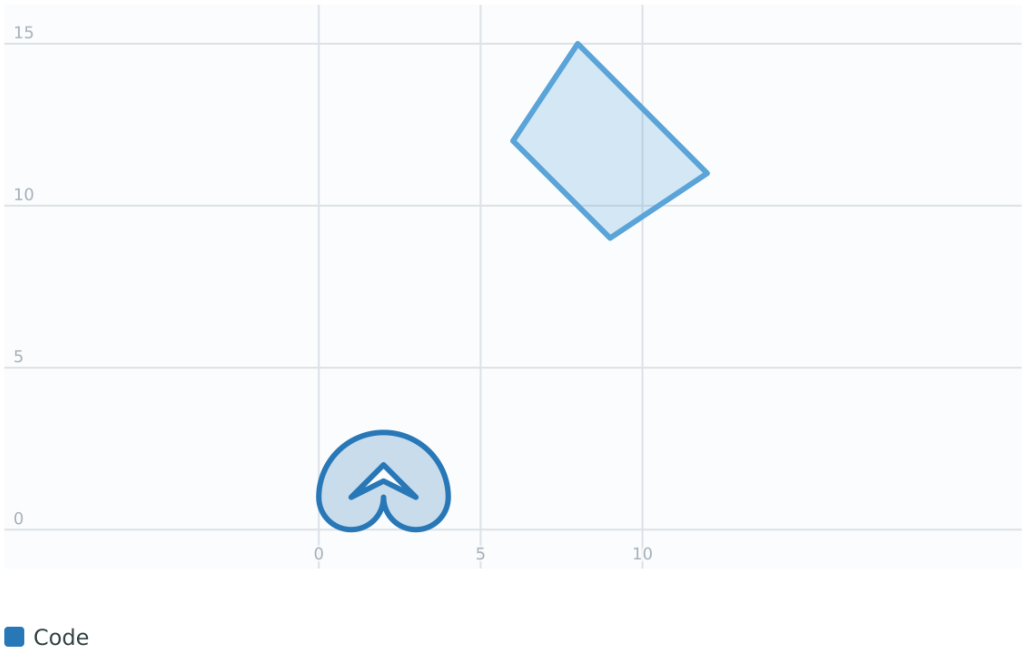
Code

```
SELECT ST_AsSVG('MULTISURFACE(
CURVEPOLYGON(CIRCULARSTRING(-2 0,-1 -1,0 0,1 -1,2 0,0 2,-2 0),
    (-1 0,0 0.5,1 0,0 1,-1 0)),
((7 8,10 10,6 14,4 11,7 8)))'::geometry, 0, 2);
```

Output

```
M -2 0 A 1 1 0 0 0 0 0 A 1 1 0 0 0 2 0 A 2 2 0 0 0 -2 0 Z
M -1 0 L 0 -0.5 1 0 0 -1 -1 0 Z
M 7 -8 L 10 -10 6 -14 4 -11 Z
```

Figure



7.9.3.12 ST_AsTWKB

ST_AsTWKB — 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

Code

```
SELECT ST_AsTWKB('LINESTRING(1 1,5 5)'::geometry);
```

Output

```
\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。

Code

```
SELECT ST_AsTWKB(array_agg(geom), array_agg(gid)) FROM mytable;
```

Output

```
\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

options 参数是一个位字段，用于控制是否使用 X3D 的 GeoCoordinates ⊠点来表示坐⊠，以及是否⊠行 x/y ⊠翻⊠。默⊠情况下，ST_AsX3D 按数据⊠方式⊠出（long,lat 或 X,Y），但如果需要，也可以采用 X3D 默⊠的 lat/lon、y/x ⊠出形式。

- 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	LineSet (coordinates are promoted to 3D)	LineSet
MULTILINESTRING	IndexedLineSet (coordinates are promoted to 3D)	IndexedLineSet
MULTIPOINT	Polypoint2D	PointSet
POINT	⊠出以空格分隔的坐⊠	⊠出以空格分隔的坐⊠
(MULTI) POLYGON, POLYHEDRALSURFACE	IndexedFaceSet (coordinates are promoted to 3D; inner rings currently output as another faceset)	IndexedFaceSet（内⊠当前⊠出⊠⊠一个面集）
TIN	TriangleSet2D（尚未⊠⊠）	IndexedTriangleSet（索引三角形集）
CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, MULTICURVE, MULTISURFACE	linearized, then emitted with the linear geometry mapping	linearized, then emitted with the linear geometry mapping
NURBSCURVE	NurbsCurve with controlPoint, knot, weight, and order fields	NurbsCurve with controlPoint, knot, weight, and order fields



Note
2D 几何支持尚未完成。内⊠当前⊠⊠制⊠⊠独的多⊠形。我⊠正在研究⊠些。

M coordinates are not represented in X3D output.

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⊠⊠⊠。

Enhanced: 3.7.0: Support for curve geometries. NURBSCURVE is emitted as X3D NurbsCurve; other curve types are linearized before output.

⊠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

Create a complete X3D document containing a cube that is viewable in FreeWRL and other X3D viewers.

Code

```
SELECT '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d ↩
    -3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
       <Appearance>
            <Material emissiveColor=''0 0 1''/>
       </Appearance
> ' ||
       ST_AsX3D(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )')) ||
      '</Shape>
    </Transform>
  </Scene>
</X3D
>' As x3ddoc;
```

Output

```
    x3ddoc
    --------
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
       <Appearance>
            <Material emissiveColor='0 0 1'/>
       </Appearance>
       <IndexedFaceSet  coordIndex='0 1 2 3 -1 4 5 6 7 -1 8 9 10 11 -1 12 13 14 15 -1 16 17 18 19 -1 20 21 22 23'>
            <Coordinate point='0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0 ↩
     1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1' />
```

```
      </IndexedFaceSet>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

PostGIS buildings.

将此⊠⊠的⊠出复制并粘⊠到x3d ⊠景⊠看器，然后⊠⊠ “⊠示”

Code

```
SELECT string_agg('<Shape
>' || ST_AsX3D(CG_Extrude(geom, 0,0, i*0.5)) ||
    '<Appearance>
          <Material diffuseColor="' || (0.01*i)::text || ' 0.8 0.2" specularColor="' || ↩
    (0.05*i)::text || ' 0 0.5"/>
        </Appearance>
    </Shape
>', '')
FROM ST_Subdivide(ST_Letters('PostGIS'),20) WITH ORDINALITY AS f(geom,i);
```

The following preview shows the extruded geometry generated from a simplified version of the same source lettering. It is a geometry preview, not X3D markup; use the preceding ST_AsX3D query output in an X3D viewer to inspect the native X3D scene.

Code

```
SELECT
    CG_Extrude(
        ST_SimplifyPreserveTopology(ST_Letters('PostGIS'), 2),
        0, 0, 10
    ) AS buildings;
```

Figure



An Octagon elevated 3 Units and decimal precision of 6.

Code

```
SELECT ST_AsX3D(ST_Translate(ST_Force3D(ST_Buffer(ST_Point(10,10),5, 'quad_segs=2')), 0,0,
    3),
  6) As x3dfrag;
```

Output

```
x3dfrag
--------
<IndexedFaceSet  convex='false' coordIndex='0 1 2 3 4 5 6 7'
><Coordinate point='15 10 3 13.535534 6.464466 3 10 5 3 6.464466 6.464466 3 5 10 3 6.464466 13.535534 3 10 15 3 ↩
     13.535534 13.535534 3 ' /></IndexedFaceSet>
```

TIN.

Code

```
SELECT ST_AsX3D(ST_GeomFromEWKT('TIN (((
                0 0 0,
                0 0 1,
                0 1 0,
                0 0 0
            )), ((
                0 0 0,
                0 1 0,
                1 1 0,
                0 0 0
            ))
            )')) As x3dfrag;
```

Output

```
    x3dfrag
    --------
<IndexedTriangleSet  index='0 1 2 3 4 5'
><Coordinate point='0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0'/></IndexedTriangleSet>
```

Closed multilinestring (the boundary of a polygon with holes).

Code

```
SELECT ST_AsX3D(ST_GeomFromEWKT('MULTILINESTRING((20 0 10,16 -12 10,0 -16 10,-12 -12 10,-20 ↩
     0 10,-12 16 10,0 24 10,16 16 10,20 0 10),
  (12 0 10,8 8 10,0 12 10,-8 8 10,-8 0 10,-8 -4 10,0 -8 10,8 -4 10,12 0 10))')
) As x3dfrag;
```

Output

```
    x3dfrag
    --------
<IndexedLineSet  coordIndex='0 1 2 3 4 5 6 7 0 -1 8 9 10 11 12 13 14 15 8'>
    <Coordinate point='20 0 10 16 -12 10 0 -16 10 -12 -12 10 -20 0 10 -12 16 10 0 24 10 16 16 10 12 0 10 8 8 10 0 12 ↩
     10 -8 8 10 -8 0 10 -8 -4 10 0 -8 10 8 -4 10 ' />
 </IndexedLineSet>
```

7.9.3.14 ST_GeoHash

ST_GeoHash — 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

Code

```
SELECT ST_GeoHash(ST_Point(-126, 48));
```

Output

```
c0w3hf1s70w3hf1s70w3
```

Code

```
SELECT ST_GeoHash(ST_Point(-126, 48), 5);
```

Output

```
c0w3h
```

This example shows that the GeoHash for a line containing a point is a prefix of the point code.

Code

```
SELECT ST_GeoHash('LINESTRING(-126 48,-126.1 48.1)'::geometry);
```

Output

```
c0w3
```

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

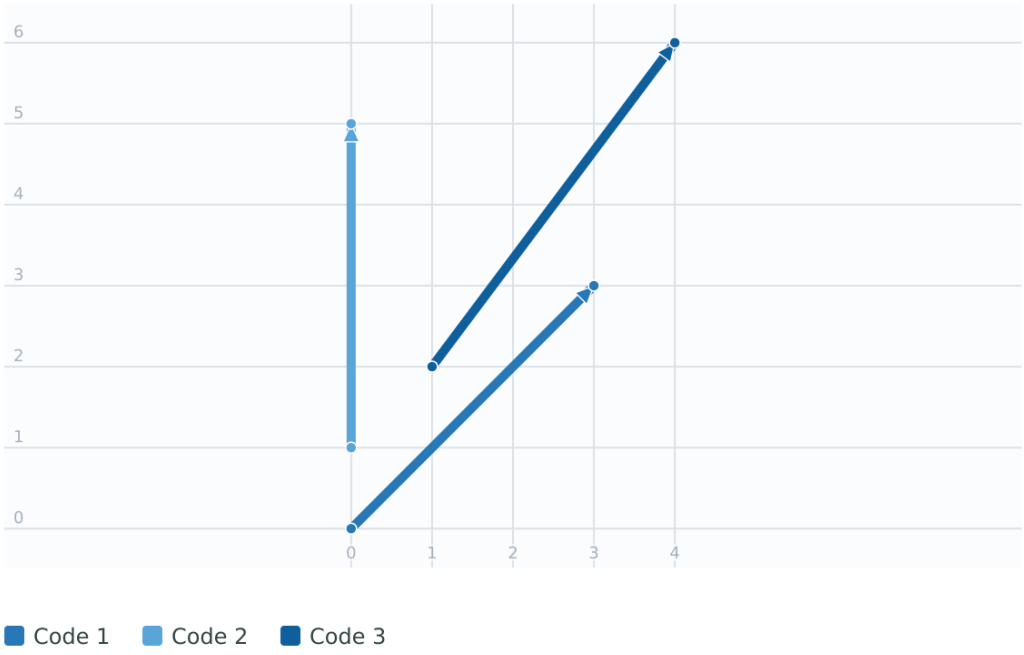
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 && tbl2.column2 AS overlaps
FROM ( VALUES
    (1, 'LINESTRING(0 0, 3 3)'::geometry),
    (2, 'LINESTRING(0 1, 0 5)'::geometry)) AS tbl1,
( VALUES
    (3, 'LINESTRING(1 2, 4 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overlaps
---------+---------+----------
       1 |       3 | t
       2 |       3 | f
(2 rows)
```

Figure



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☒ BRIN 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

Code

```
SELECT ST_Point(1,1) && ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) AS overlaps;
```

Output

```
 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⊠ BRIN 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

Code

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_Point(1,1) AS overlaps;
```

Output

```
 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

&& ☒算符在使用浮点精度☒，如果☒个 2D ☒界框 A 和 B 相交，☒返回 TRUE。☒意味着如果 A（或 B）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
☒☒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☒ BRIN 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

Code

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_MakeBox2D(ST_Point(1,1), ST_Point ↩
    (3,3)) AS overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

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 维界框与 B 的 n 维界框相交，则返回 TRUE。

Synopsis

boolean **&&&**(geometry A , geometry B);

描述

&&& 运算符返回 TRUE，如果 A 的 n 维界框与 B 的 n 维界框相交。



Note
此操作符将利用几何上可能可用的任何索引。

可用性: 2.0.0


此方法支持三维字符串和曲线。


三维函数支持多面体曲面。


此函数支持三角形和不规则三角网面 (TIN)。


三维函数支持 3d 并且不会丢失 z-index。

示例

3D LineStrings.

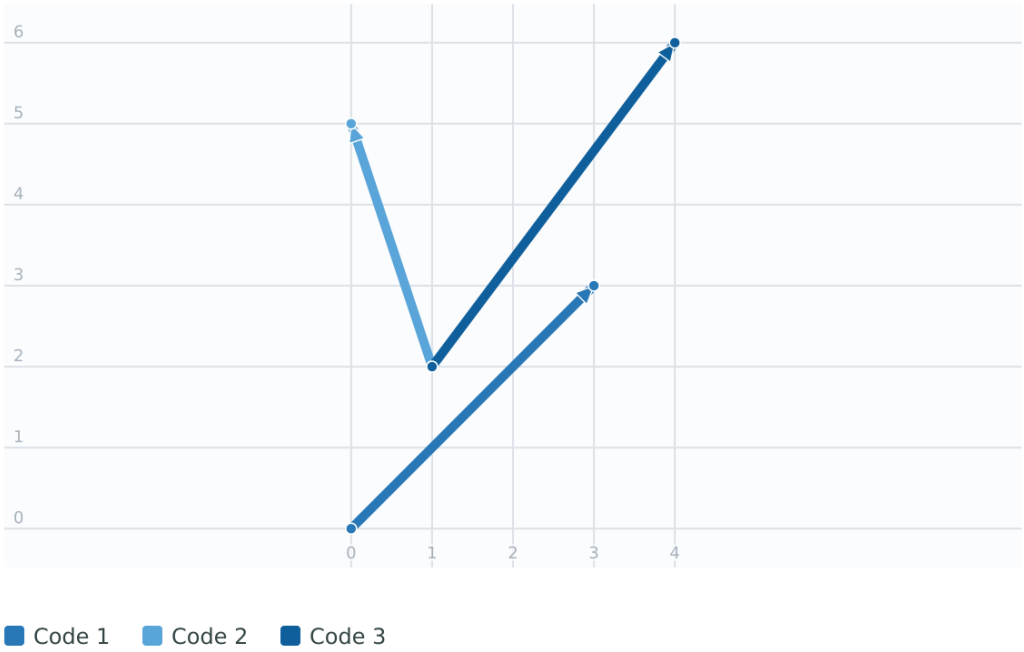
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &&& tbl2.column2 AS overlaps_3d,
                        tbl1.column2 && tbl2.column2 AS overlaps_2d
FROM ( VALUES
    (1, 'LINESTRING Z(0 0 1, 3 3 2)'::geometry),
    (2, 'LINESTRING Z(1 2 0, 0 5 -1)'::geometry)) AS tbl1,
( VALUES
    (3, 'LINESTRING Z(1 2 1, 4 6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overlaps_3d | overlaps_2d
---------+---------+-------------+-------------
       1 |       3 | t           | t
       2 |       3 | f           | t
```

Figure



3M LineStrings.

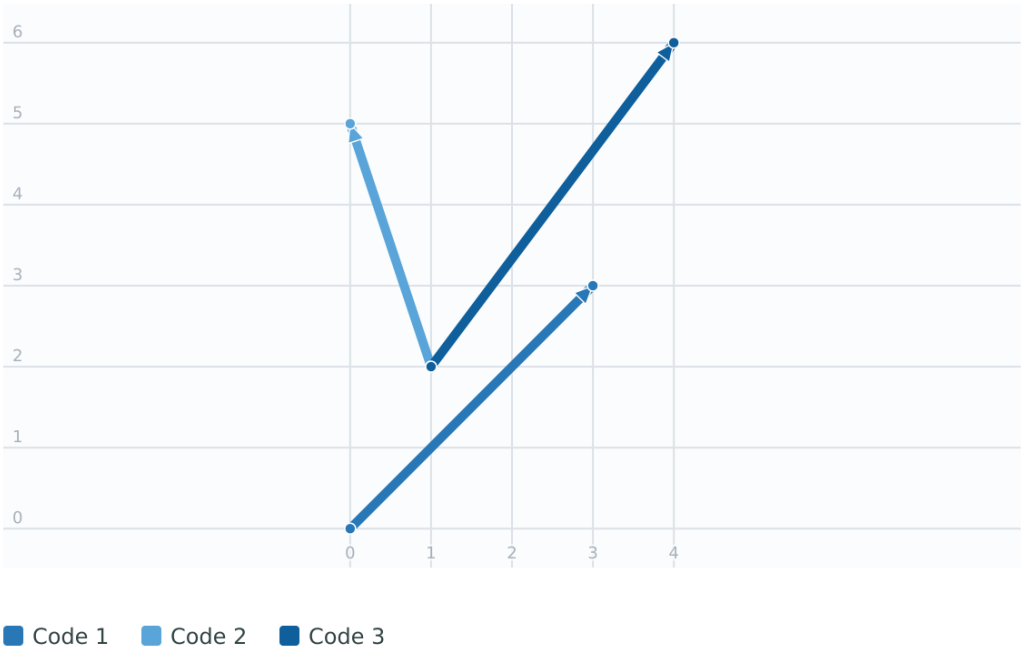
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &&& tbl2.column2 AS overlaps_3zm,
                       tbl1.column2 && tbl2.column2 AS overlaps_2d
FROM ( VALUES
    (1, 'LINESTRING M(0 0 1, 3 3 2)'::geometry),
    (2, 'LINESTRING M(1 2 0, 0 5 -1)'::geometry)) AS tbl1,
( VALUES
    (3, 'LINESTRING M(1 2 1, 4 6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overlaps_3zm | overlaps_2d
---------+---------+--------------+-------------
       1 |       3 | t            | t
       2 |       3 | f            | t
```

Figure



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⊠ BRIN 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

Code

```
SELECT ST_MakePoint(1,1,1) &&& ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) AS
    overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

相关信息

&&&(gidx,geometry), &&&(gidx,gidx)

7.10.1.7 &&&(gidx,geometry)

&&&(gidx,geometry) — 如果 n 维浮点精度索引框 (GIDX) 与几何体的（索引的）n 维索引框相交，则返回 TRUE。

Synopsis

boolean **&&&**(gidx A , geometry B);

描述

&&& 运算符在使用浮点精度索引，如果 n-D 索引框 A 与几何体 B 的索引存 n-D 索引框相交，则返回 TRUE。这意味着如果 A 是一个（双精度）box3d，它将在内部被转换为一个浮点精度的 3D 索引框（GIDX）

Note
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 BRIN 的支持。

此方法支持圆形字符串和曲线。

函数支持多面体曲面。

此函数支持三角形和不规则三角网面 (TIN)。

函数支持 3d 并且不会丢失 z-index。

示例

Code

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_MakePoint(1,1,1) AS
    overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

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☒ BRIN 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

Code

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_3DMakeBox(ST_MakePoint
    (1,1,1), ST_MakePoint(3,3,3)) AS overlaps;
```

Output

```
 overlaps
----------
 t
(1 row)
```

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

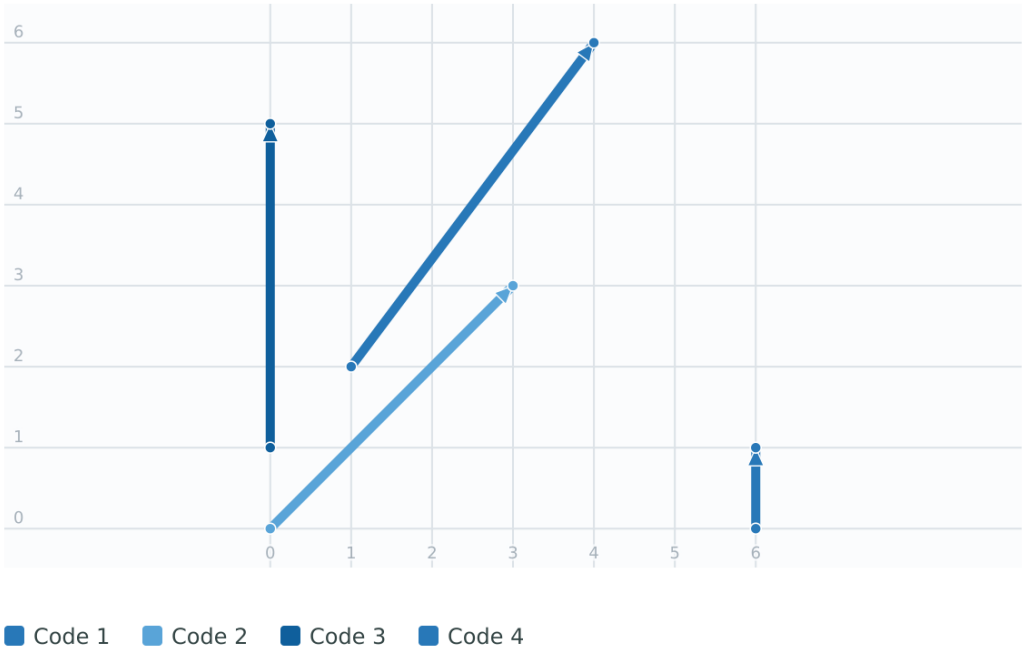
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &< tbl2.column2 AS overleft
FROM
  ( VALUES
    (1, 'LINESTRING(1 2, 4 6)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)'::geometry),
    (3, 'LINESTRING(0 1, 0 5)'::geometry),
    (4, 'LINESTRING(6 0, 6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overleft
---------+---------+----------
       1 |       2 | f
       1 |       3 | f
       1 |       4 | t
(3 rows)
```

Figure



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

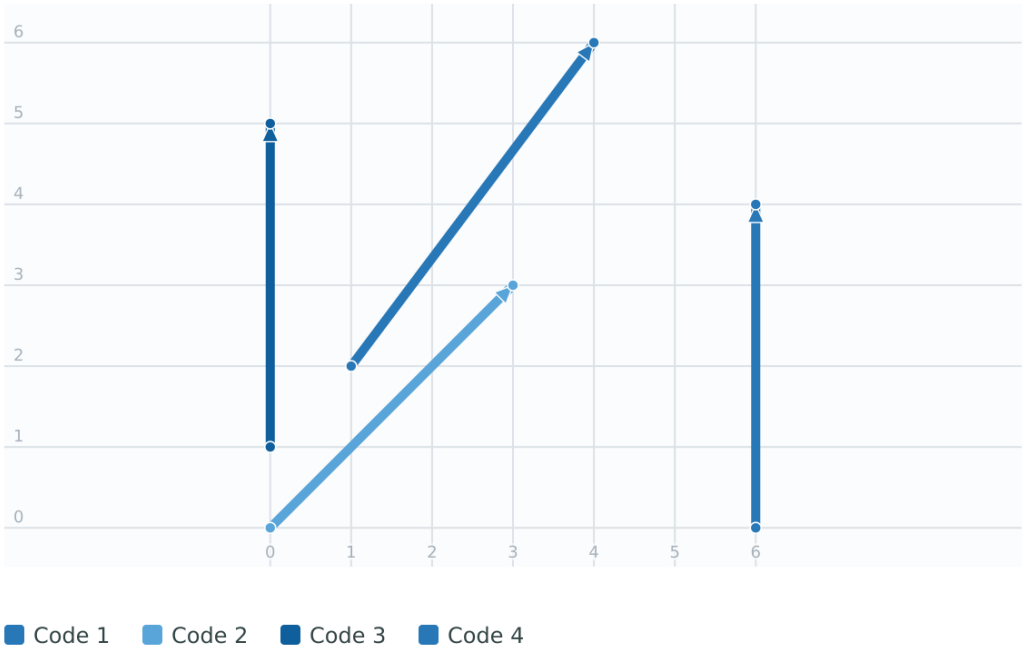
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &<| tbl2.column2 AS overbelow
FROM
  ( VALUES
    (1, 'LINESTRING(6 0, 6 4)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)'::geometry),
    (3, 'LINESTRING(0 1, 0 5)'::geometry),
    (4, 'LINESTRING(1 2, 4 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overbelow
---------+---------+-----------
       1 |       2 | f
       1 |       3 | t
       1 |       4 | t
(3 rows)
```

Figure



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

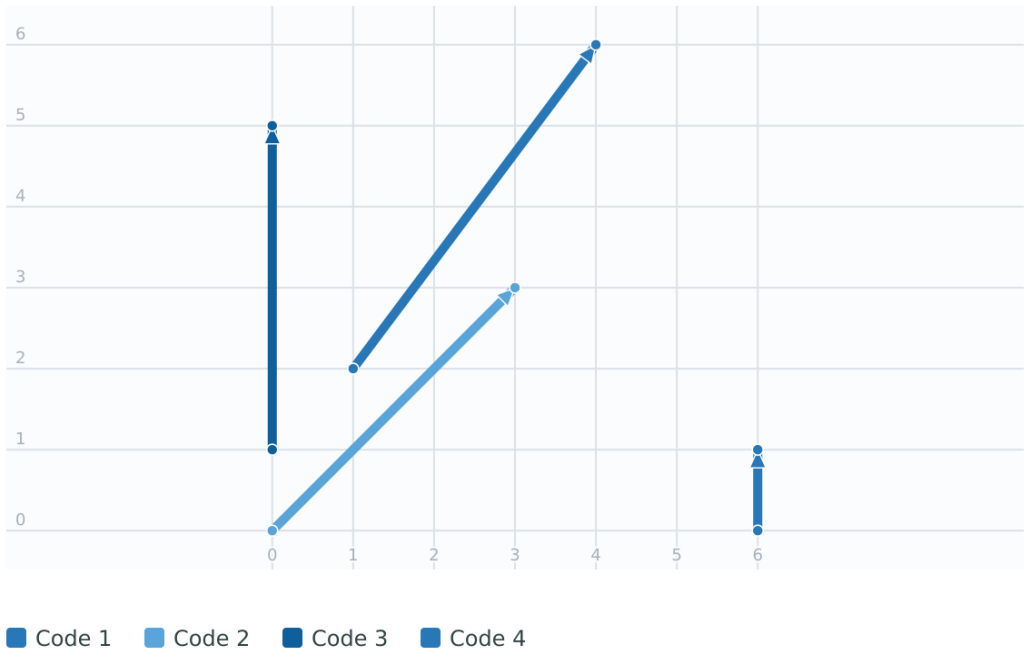
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &
> tbl2.column2 AS overright
FROM
  ( VALUES
    (1, 'LINESTRING(1 2,4 6)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0,3 3)'::geometry),
    (3, 'LINESTRING(0 1,0 5)'::geometry),
    (4, 'LINESTRING(6 0,6 1)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overright
---------+---------+-----------
       1 |       2 | t
       1 |       3 | t
       1 |       4 | f
(3 rows)
```

Figure



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

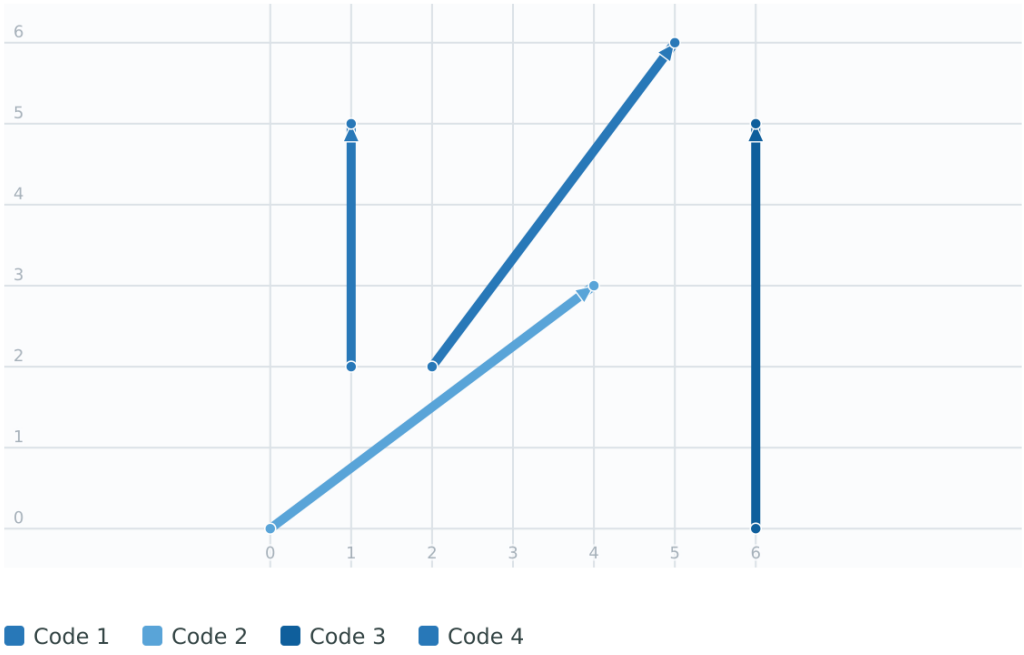
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 << tbl2.column2 AS left
FROM
  ( VALUES
    (1, 'LINESTRING (1 2, 1 5)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0, 4 3)'::geometry),
    (3, 'LINESTRING (6 0, 6 5)'::geometry),
    (4, 'LINESTRING (2 2, 5 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | left
---------+---------+------
       1 |       2 | f
       1 |       3 | t
       1 |       4 | t
(3 rows)
```

Figure



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

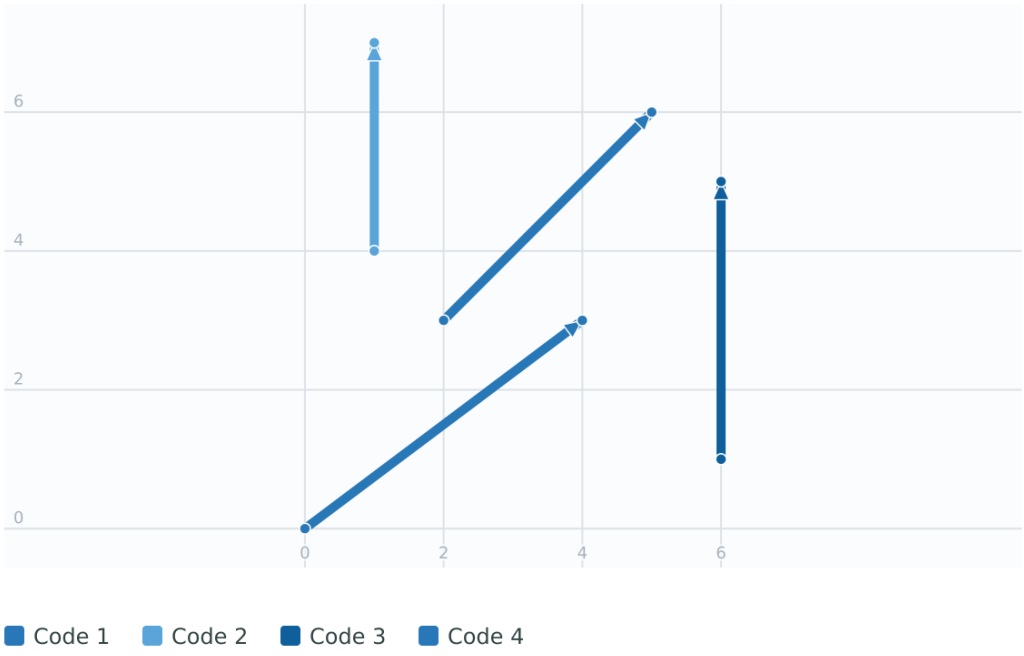
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 <<| tbl2.column2 AS below
FROM
  ( VALUES
    (1, 'LINESTRING (0 0, 4 3)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (1 4, 1 7)'::geometry),
    (3, 'LINESTRING (6 1, 6 5)'::geometry),
    (4, 'LINESTRING (2 3, 5 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | below
---------+---------+-------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



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

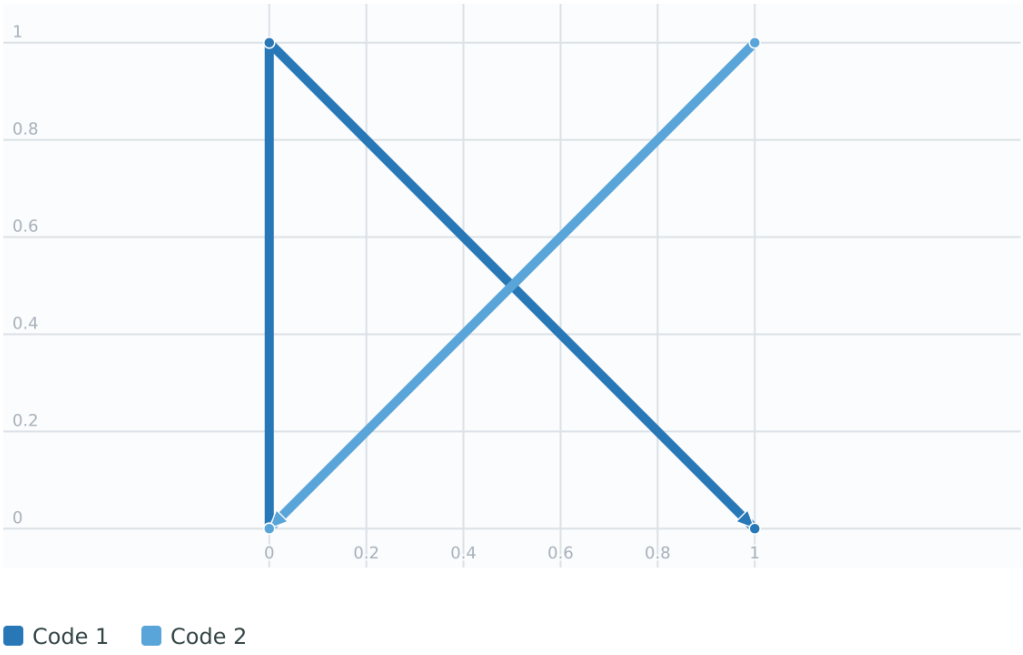
Code

```
SELECT 'LINESTRING(0 0,0 1,1 0)'::geometry = 'LINESTRING(1 1,0 0)'::geometry;
```

Output

```
 ?column?
----------
 f
(1 row)
```

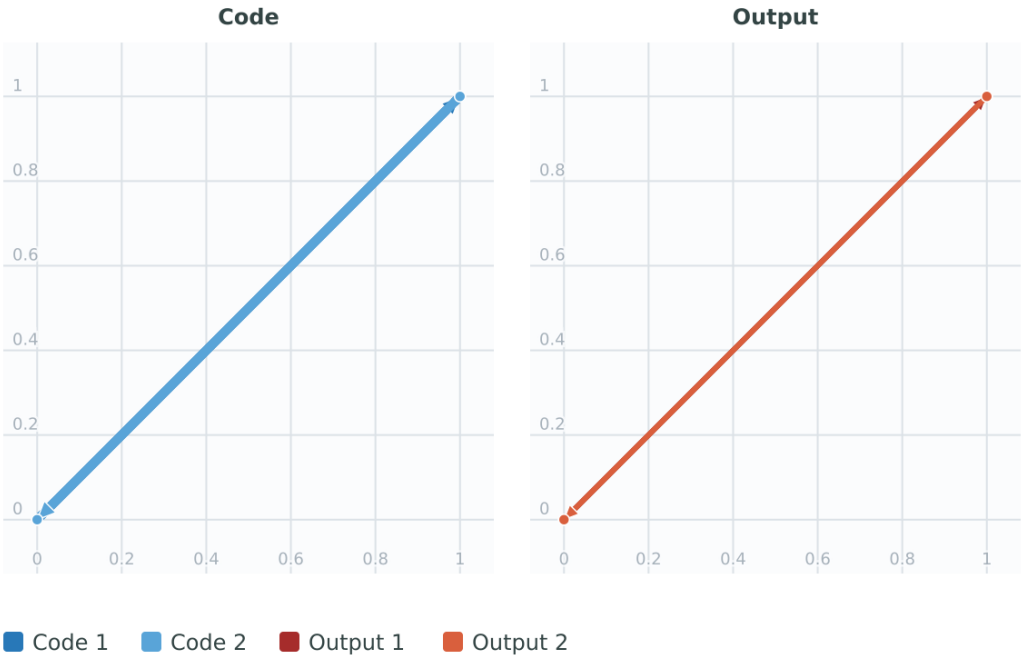
Figure



Code

```
SELECT column1
FROM ( VALUES
    ('LINESTRING(0 0, 1 1)'::geometry),
    ('LINESTRING(1 1, 0 0)'::geometry)) AS foo;
```

Figure

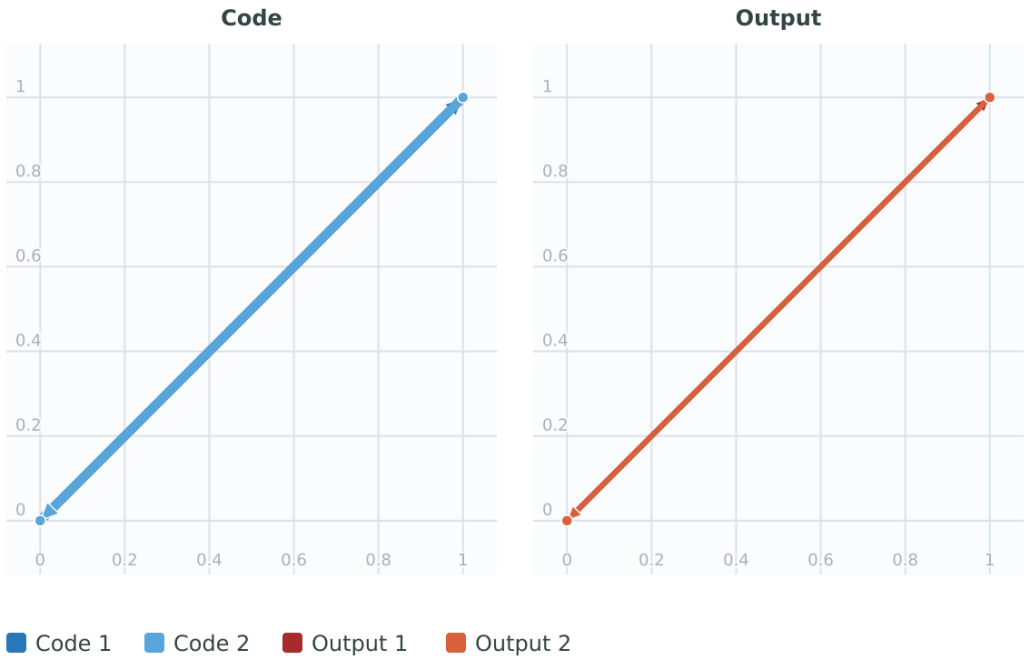


Note: the GROUP BY uses the "=" to compare for geometry equivalency.

Code

```
SELECT column1
FROM ( VALUES
    ('LINESTRING(0 0, 1 1)'::geometry),
    ('LINESTRING(1 1, 0 0)'::geometry)) AS foo
GROUP BY column1;
```

Figure



In versions prior to 2.0, this used to return true.

Code

```
SELECT 'POINT(1707296.37 4820536.77)'::geometry =
   'POINT(1707296.27 4820536.87)'::geometry As pt_intersect;
```

Output

```
 pt_intersect
--------------
 f
(1 row)
```

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

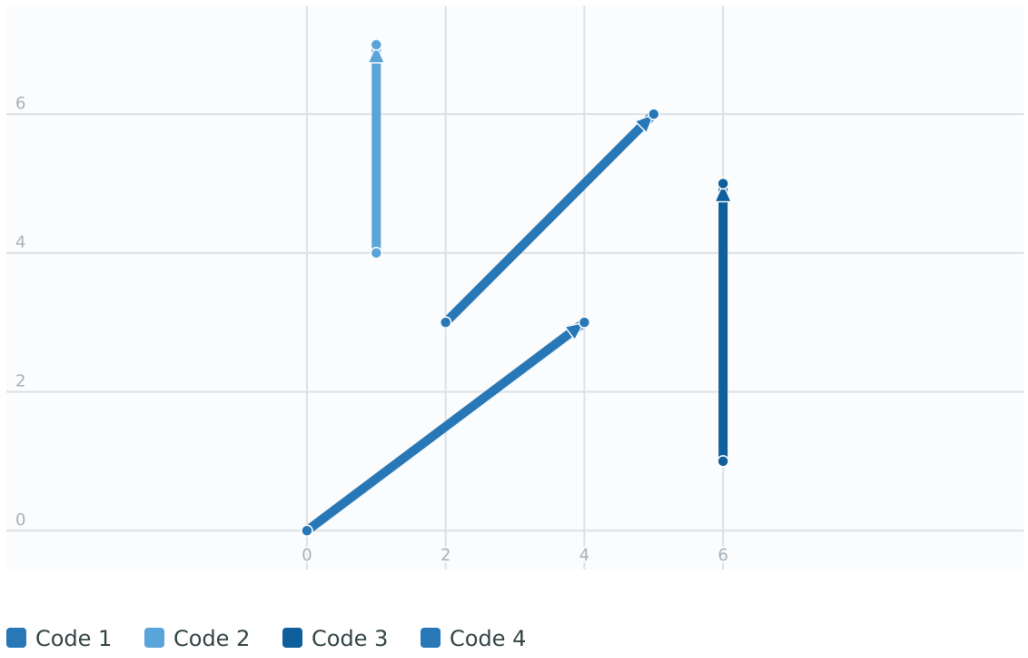
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2
>
> tbl2.column2 AS right
FROM
  ( VALUES
    (1, 'LINESTRING (2 3,5 6)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (1 4,1 7)'::geometry),
    (3, 'LINESTRING (6 1,6 5)'::geometry),
    (4, 'LINESTRING (0 0,4 3)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | right
---------+---------+-------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



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

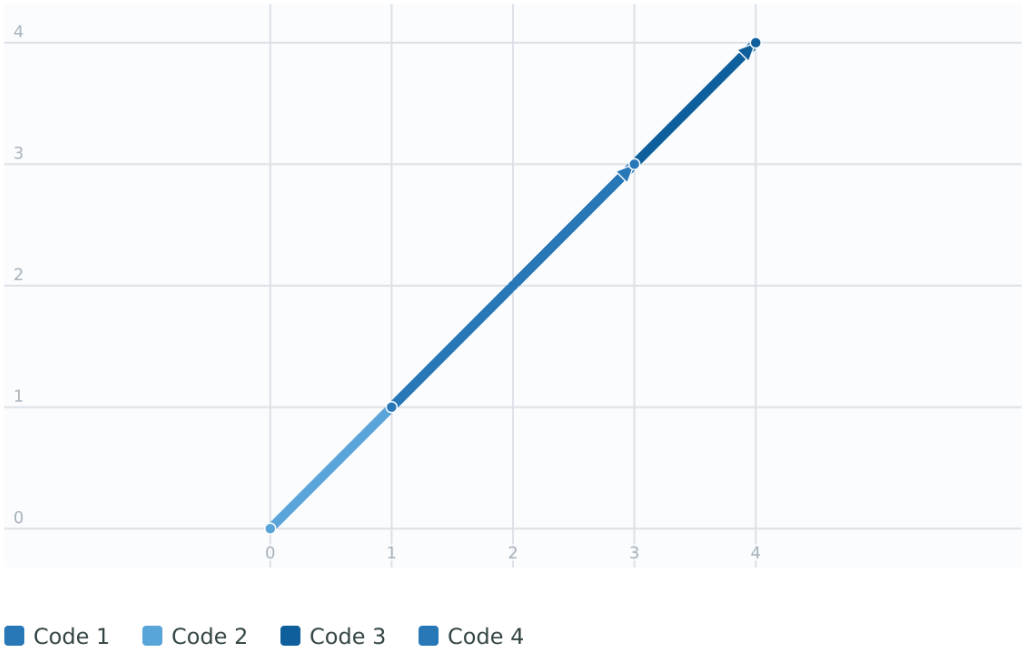
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 @ tbl2.column2 AS contained
FROM
  ( VALUES
    (1, 'LINESTRING (1 1, 3 3)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0, 4 4)'::geometry),
    (3, 'LINESTRING (2 2, 4 4)'::geometry),
    (4, 'LINESTRING (1 1, 3 3)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | contained
---------+---------+-----------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | t
(3 rows)
```

Figure



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 BRIN 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

Code

```
SELECT ST_Buffer('POINT(2 2)'::geometry, 1) @ ST_MakeBox2D(ST_Point(0, 0), ST_Point(5, 5))  ↩
    AS is_contained;
```

Output

```
t
```

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，⊠ @ ⊠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⊠ BRIN 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

Code

```
SELECT ST_MakeBox2D(ST_Point(2, 2), ST_Point(3, 3)) @ ST_Buffer('POINT(1 1)'::geometry, 10) ↩
    AS is_contained;
```

Output

```
t
```

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⍰一个 2D 浮点精度⍰界框内，⍰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，⍰ @ ⍰算符使用浮点精度返回 TRUE。⍰意味着如果 A（或 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⍰ BRIN 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

Code

```
SELECT ST_MakeBox2D(ST_Point(2, 2), ST_Point(3, 3)) @ ST_MakeBox2D(ST_Point(0, 0), ST_Point ↩
    (5, 5)) AS is_contained;
```

Output

```
t
```

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

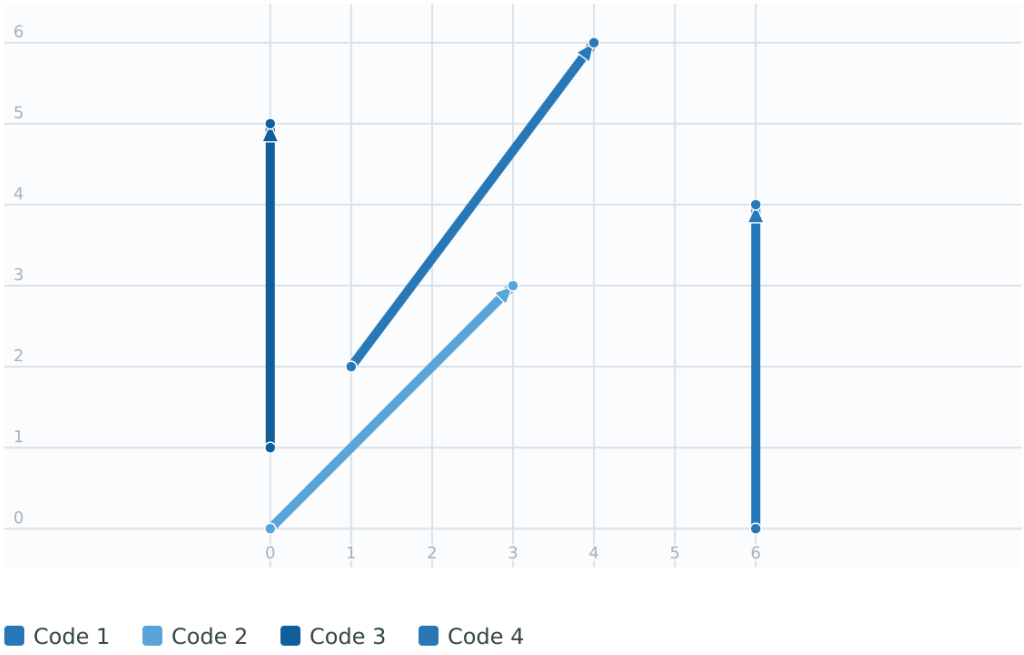
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 |&
> tbl2.column2 AS overabove
FROM
  ( VALUES
    (1, 'LINESTRING(6 0,6 4)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0,3 3)'::geometry),
    (3, 'LINESTRING(0 1,0 5)'::geometry),
    (4, 'LINESTRING(1 2,4 6)'::geometry)) AS tbl2;
```

Output

```
 column1 | column1 | overabove
---------+---------+-----------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



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

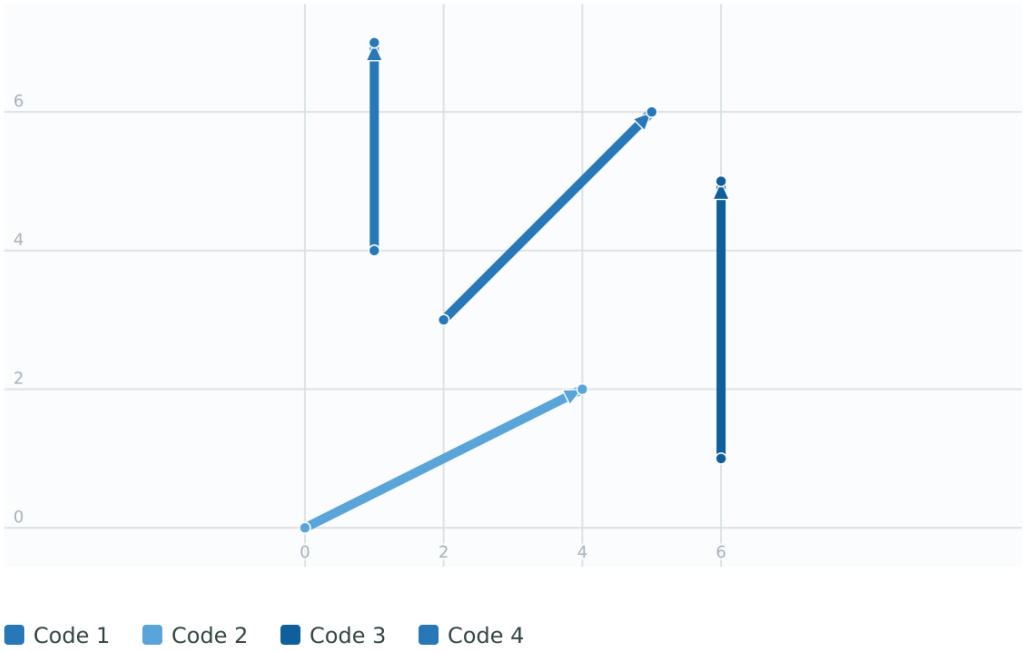
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 |>> tbl2.column2 AS above
FROM
  ( VALUES
    (1, 'LINESTRING (1 4,1 7)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0,4 2)'::geometry),
    (3, 'LINESTRING (6 1,6 5)'::geometry),
    (4, 'LINESTRING (2 3,5 6)'::geometry)) AS tbl2;
```

Output

```
column1 | column1 | above
---------+---------+-------
       1 |       2 | t
       1 |       3 | f
       1 |       4 | f
(3 rows)
```

Figure



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

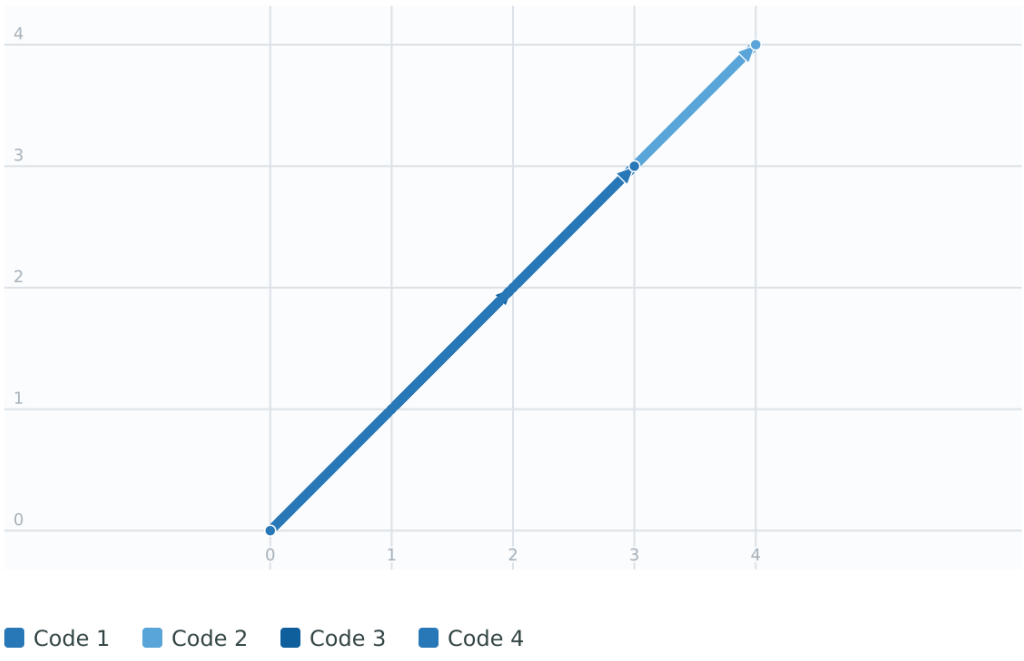
Code

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 ~ tbl2.column2 AS contains
FROM
  ( VALUES
    (1, 'LINESTRING (0 0,3 3)'::geometry)) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0,4 4)'::geometry),
    (3, 'LINESTRING (1 1,2 2)'::geometry),
    (4, 'LINESTRING (0 0,3 3)'::geometry)) AS tbl2;
```

Output

```
column1 | column1 | contains
---------+---------+----------
       1 |       2 | f
       1 |       3 | t
       1 |       4 | t
(3 rows)
```

Figure



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⌧ BRIN 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

Code

```
SELECT ST_Buffer('POINT(1 1)'::geometry, 10) ~ ST_MakeBox2D(ST_Point(0, 0), ST_Point(2, 2)) ↩
    AS contains;
```

Output

```
t
```

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 BRIN 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

Code

```
SELECT ST_MakeBox2D(ST_Point(0, 0), ST_Point(5, 5)) ~ ST_Buffer('POINT(2 2)'::geometry, 1)  ↩
    AS contains;
```

Output

```
t
```

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 BRIN 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

Code

```
SELECT ST_MakeBox2D(ST_Point(0, 0), ST_Point(5, 5)) ~ ST_MakeBox2D(ST_Point(2, 2), ST_Point ↩
    (3, 3)) AS contains;
```

Output

```
t
```

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

Code

```
SELECT 'LINESTRING(0 0,1 1)'::geometry ~= 'LINESTRING(0 1,1 0)'::geometry AS equality;
```

Output

```
t
```

Figure



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⊠象之⊠的二⊠距离。用于 ORDER BY ⊠，可以借助索引⊠⊠最近⊠排序：⊠ geometry ⊠行真⊠的 KNN 距离搜索，⊠ geography 返回球面距离。



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。

Enhanced: 2.2.0 -- 支持 geometry 和 geography 的真☒ KNN 行☒。注意 geography 的 KNN 基于球体而非☒球体。

Changed: 2.2.0 -- 在提供真☒ KNN 之后，旧的混合☒法☒通方式可能会更慢。参☒下方示例。

可用性：2.0.0 -- 弱 KNN 基于几何☒象☒心距离来找最近☒，而非真☒距离。☒点来☒☒果是准确的，☒其他☒型☒是近似的。

示例

Code

```
SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d, edabbr, vaabbr
FROM va2005
ORDER BY d limit 10;
```

Output

```
d          | edabbr | vaabbr
-----------------+--------+--------
                 0 | ALQ    | 128
 5541.57712511724 | ALQ    | 129A
 5579.67450712005 | ALQ    | 001
  6083.4207708641 | ALQ    | 131
  7691.2205404848 | ALQ    | 003
 7900.75451037313 | ALQ    | 122
 8694.20710669982 | ALQ    | 129B
 9564.24289057111 | ALQ    | 130
  12089.665931705 | ALQ    | 127
 18472.5531479404 | ALQ    | 002
(10 rows)
```

KNN 的原始答案是：

Code

```
SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d,edabbr, vaabbr
FROM va2005
ORDER BY geom <-> 'SRID=3005;POINT(1011102 450541)'::geometry limit 10;
```

Output

```
        d          | edabbr | vaabbr
-----------------+--------+--------
                 0 | ALQ    | 128
 5541.57712511724 | ALQ    | 129A
 5579.67450712005 | ALQ    | 001
  6083.4207708641 | ALQ    | 131
  7691.2205404848 | ALQ    | 003
 7900.75451037313 | ALQ    | 122
 8694.20710669982 | ALQ    | 129B
 9564.24289057111 | ALQ    | 130
  12089.665931705 | ALQ    | 127
 18472.5531479404 | ALQ    | 002
(10 rows)
```

如果您□□□个□□□行“EXPLAIN ANALYZE”，您将看到第二个□□的性能有所提高。

当□□□划无法直接□出精确距离□，可以使用混合□□来□取真正的最近□。做法是先用 CTE 利用索引□助的 KNN 得到候□集，再做一次精确排序，流程如下：

Code

```
WITH index_query AS (
  SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d,edabbr, vaabbr
    FROM va2005
  ORDER BY geom <-> 'SRID=3005;POINT(1011102 450541)'::geometry LIMIT 100)
  SELECT *
    FROM index_query
  ORDER BY d limit 10;
```

Output

```
        d         | edabbr | vaabbr
------------------+--------+--------
                0 | ALQ    | 128
 5541.57712511724 | ALQ    | 129A
 5579.67450712005 | ALQ    | 001
  6083.4207708641 | ALQ    | 131
  7691.2205404848 | ALQ    | 003
 7900.75451037313 | ALQ    | 122
 8694.20710669982 | ALQ    | 129B
 9564.24289057111 | ALQ    | 130
  12089.665931705 | ALQ    | 127
 18472.5531479404 | ALQ    | 002
(10 rows)
```

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□迹的 3D 距离（参□ ST_IsValidTrajectory）。它的作用等同于 ST_DistanceCPA，但作□□算符可以□合 N □索引用于最近□搜索。

> **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。

示例

This example orders stored trajectories by their distance from a literal query trajectory.

Code

```
\set qt 'ST_AddMeasure(ST_MakeLine(ST_MakePointM(-350, 300, 0), ST_MakePointM(-410, 490, 0) ↩
    ), 10, 20)'

SELECT track_id, dist FROM (
  SELECT track_id,
    ST_DistanceCPA(tr, :qt) dist
  FROM trajectories
  ORDER BY tr |=| :qt
  LIMIT 5
) foo;
```

Output

```
track_id        dist
----------+------------------
      395 | 0.576496831518066
      380 |  5.06797130410151
      390 |  7.72262293958322
      385 |   9.8004461358071
      405 |  10.9534397988433
(5 rows)
```

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⊠个浮点包⊠盒之⊠的距离。适用于⊠行最近⊠的 **approximate** 距离排序。

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。

示例

Code

```
SELECT *
FROM (
SELECT b.tlid, b.mtfcc,
    b.geom <#
> ST_GeomFromText('LINESTRING(746149 2948672,745954 2948576,
        745787 2948499,745740 2948468,745712 2948438,
        745690 2948384,745677 2948319)', 2249) As b_dist,
        ST_Distance(b.geom, ST_GeomFromText('LINESTRING(746149 2948672,745954 2948576,
        745787 2948499,745740 2948468,745712 2948438,
        745690 2948384,745677 2948319)', 2249)) As act_dist
    FROM bos_roads As b
    ORDER BY b_dist, b.tlid
    LIMIT 100) As foo
    ORDER BY act_dist, tlid LIMIT 10;
```

Output

```
   tlid    | mtfcc |      b_dist      |     act_dist
-----------+-------+------------------+------------------
  85732027 | S1400 |                0 |                0
  85732029 | S1400 |                0 |                0
  85732031 | S1400 |                0 |                0
  85734335 | S1400 |                0 |                0
  85736037 | S1400 |                0 |                0
 624683742 | S1400 |                0 | 128.528874268666
  85719343 | S1400 | 260.839270432962 | 260.839270432962
  85741826 | S1400 | 164.759294123275 | 260.839270432962
  85732032 | S1400 |           277.75 | 311.830282365264
  85735592 | S1400 |           222.25 | 311.830282365264
(10 rows)
```

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

<<->> 运算符返回两个几何之间的 n 维（欧几里德）距离。对于最近邻近似距离排序很有用。



Note
此操作数将利用几何上可能可用的 n-D GiST 索引。它与其他使用空间索引的运算符不同，只有当运算符位于 ORDER BY 子句中时才会使用空间索引。



Note
当其中一个几何形是常量（不在子查询/cte 中）时，索引才会被使用。例如'SRID=3005;POINT(1011102 450541)'::geometry 而不是 a.geom

可用性：2.2.0。

相关信息

<->

7.11 空间关系

7.11.1 拓扑关系

7.11.1.1 ST_3DIntersects

ST_3DIntersects — 两个几何形在 3D 空间中是否相交 - 适用于点、线串、多边形、多面体曲面（区域）

Synopsis

boolean **ST_3DIntersects**(geometry geomA , geometry geomB);

描述

重叠、接触、内在都意味着空间交叉。如果上述任何一个返回 true，则几何形也在空间上相交。不相交意味着空间相交假。



Note
此功能自动包括利用几何上可用的任何空间索引的边界框比较。

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

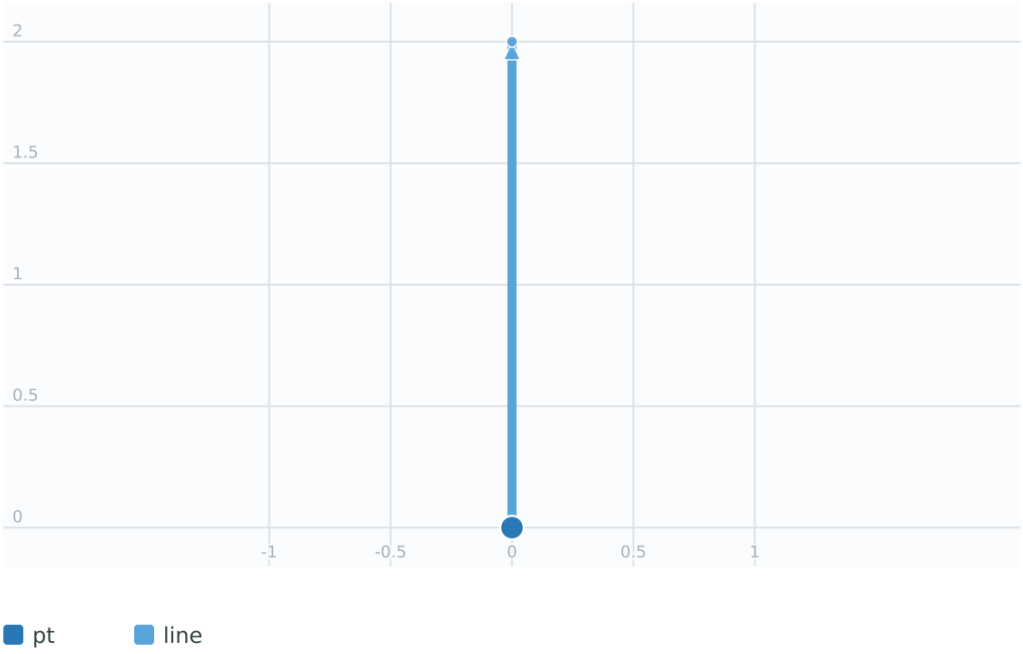
Code

```
SELECT ST_3DIntersects(pt, line), ST_Intersects(pt, line)
FROM (SELECT 'POINT(0 0 2)'::geometry As pt, 'LINESTRING (0 0 1,0 2 3)'::geometry As line)  ↩
    As foo;
```

Output

```
st_3dintersects | st_intersects
----------------+---------------
 f              | t
(1 row)
```

Figure



TIN 示例

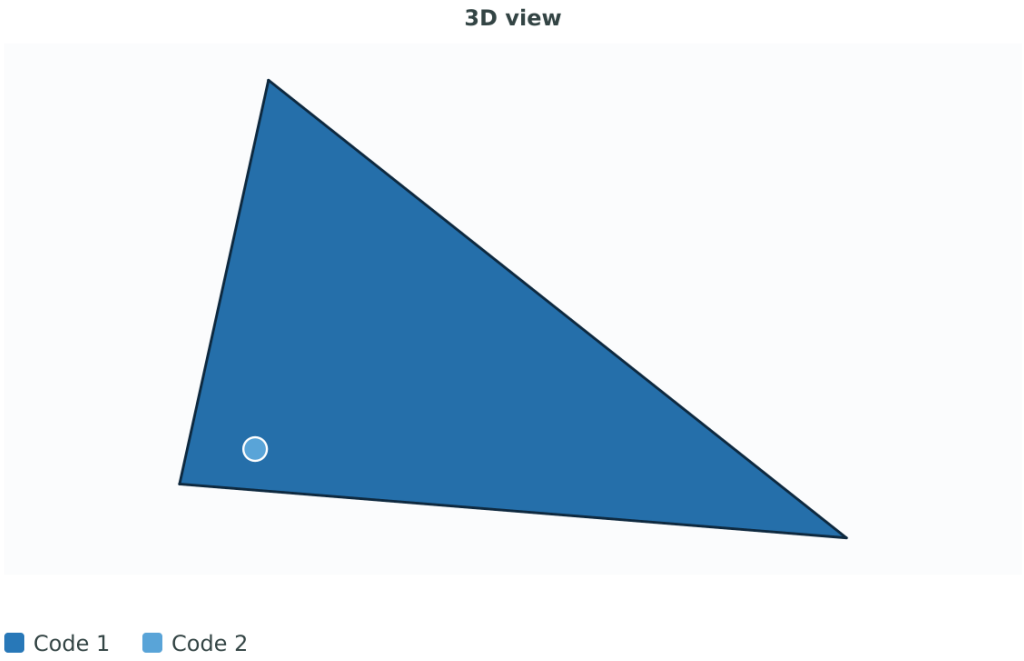
Code

```
SELECT ST_3DIntersects('TIN(((0 0 0,1 0 0,0 1 0,0 0 0)))'::geometry, 'POINT(.1 .1 0)':: ↩
    geometry);
```

Output

```
t
```

Figure



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。

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：

A LineString contains a MultiPoint whose points lie in its interior or boundary.

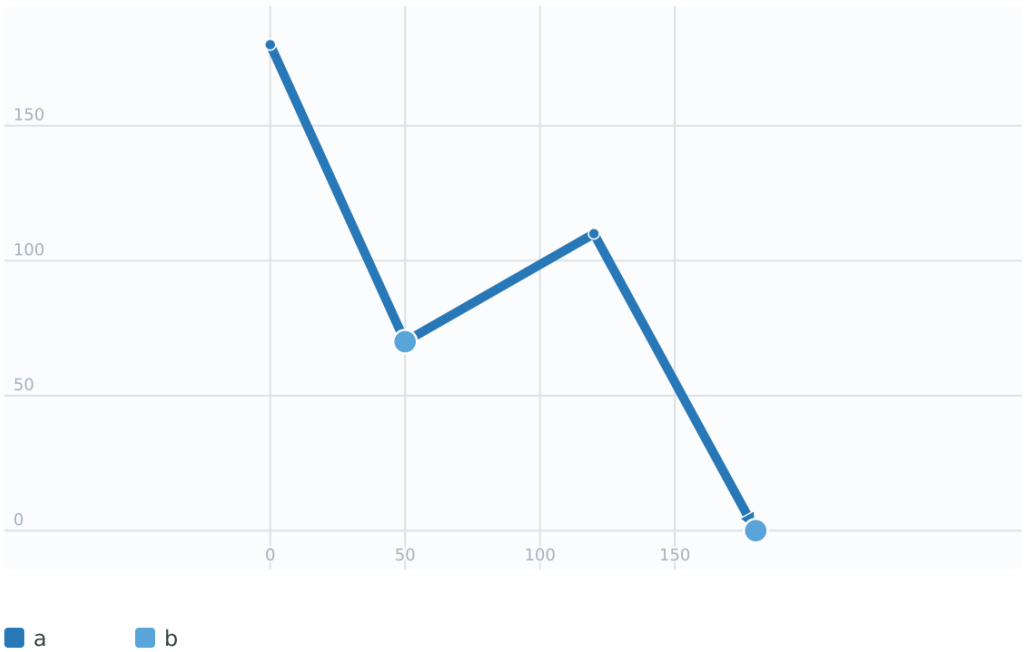
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'MULTIPOINT (60 80,190 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon contains a Point in its interior.

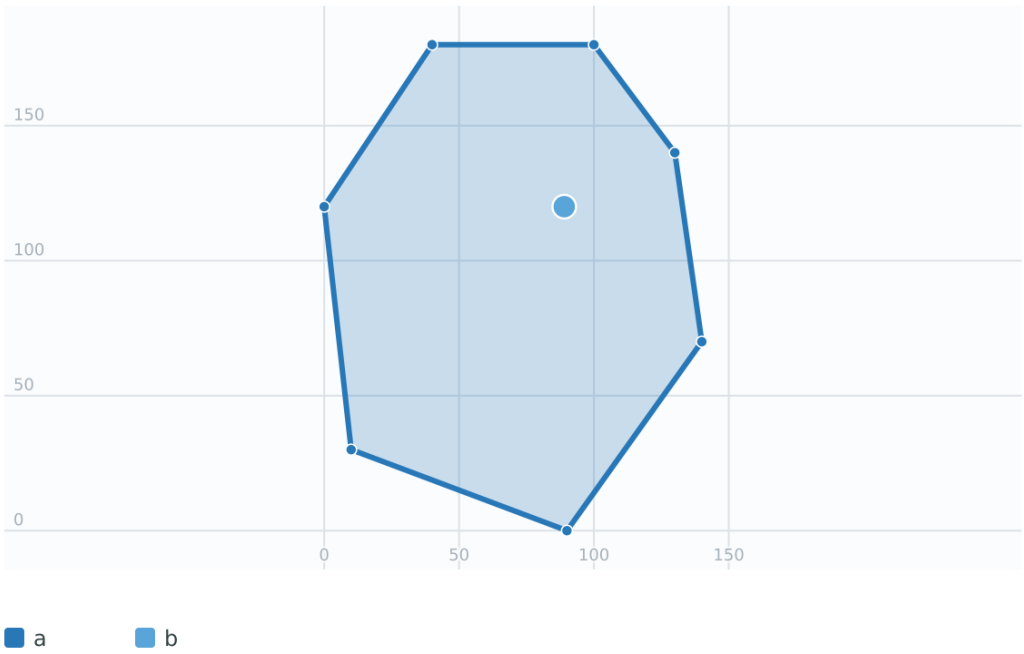
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'POINT (99 130)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon contains a LineString with interior points inside the Polygon.

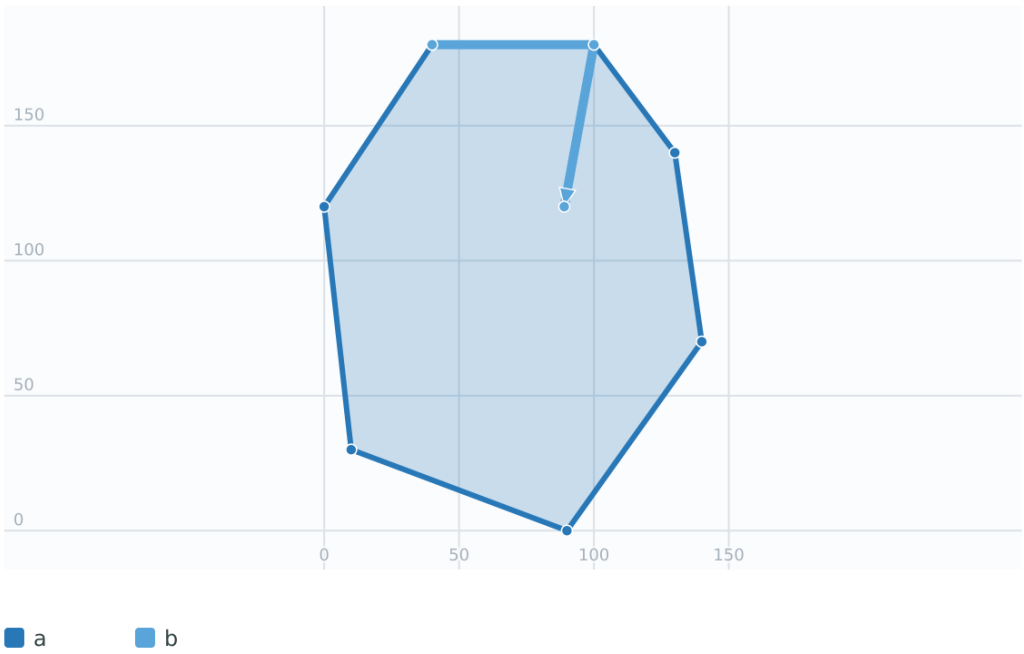
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'LINESTRING (50 190,110 190,99 130)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon contains another Polygon in its interior.

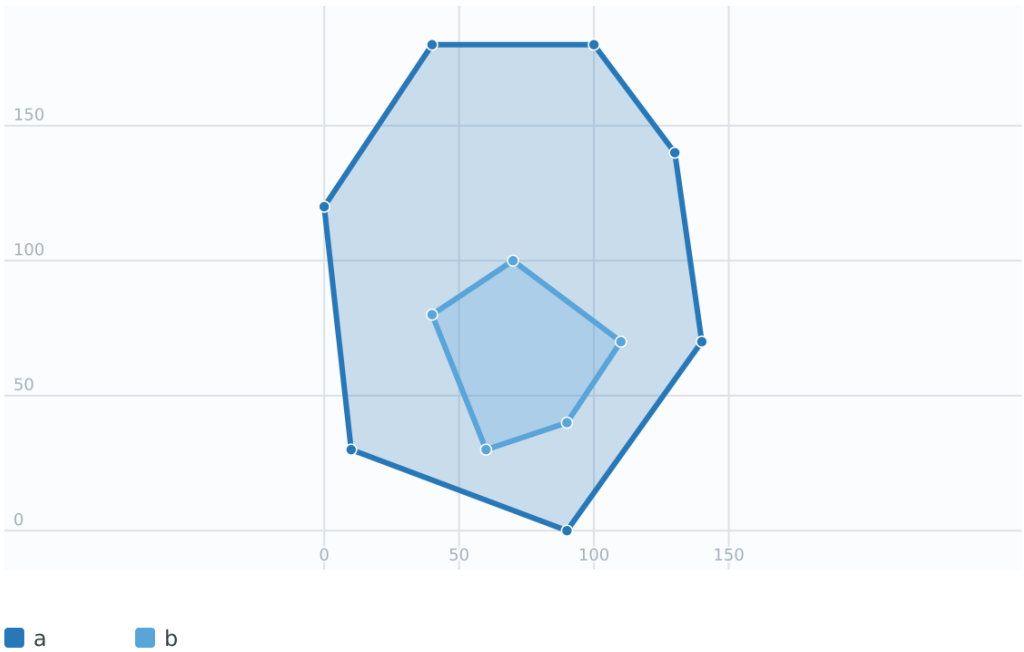
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'POLYGON ((70 40,100 50,120 80,80 110,50 90,70 40))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



ST_Contains 在以下情况下返回 FALSE：

A Polygon does not contain a MultiPoint when one point lies in a hole.

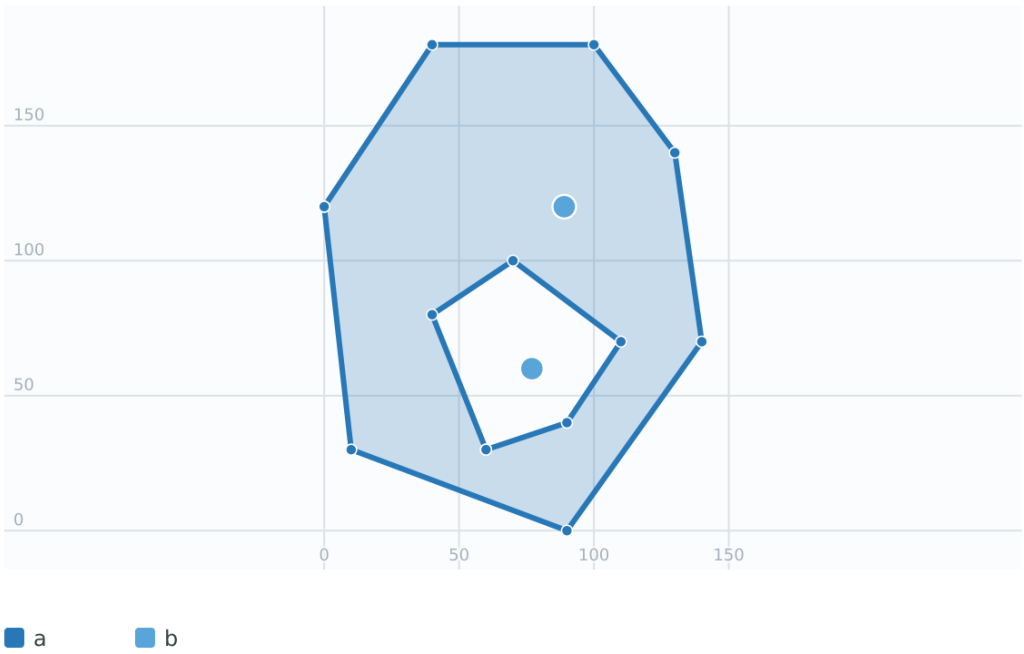
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130),(70 40,100 50,120 ↩
    80,80 110,50 90,70 40))'::geometry AS a,
  'MULTIPOINT (99 130,87 70)'::geometry AS b
) AS example;
```

Output

```
f
```

Figure



A Polygon does not contain a LineString extending outside it.

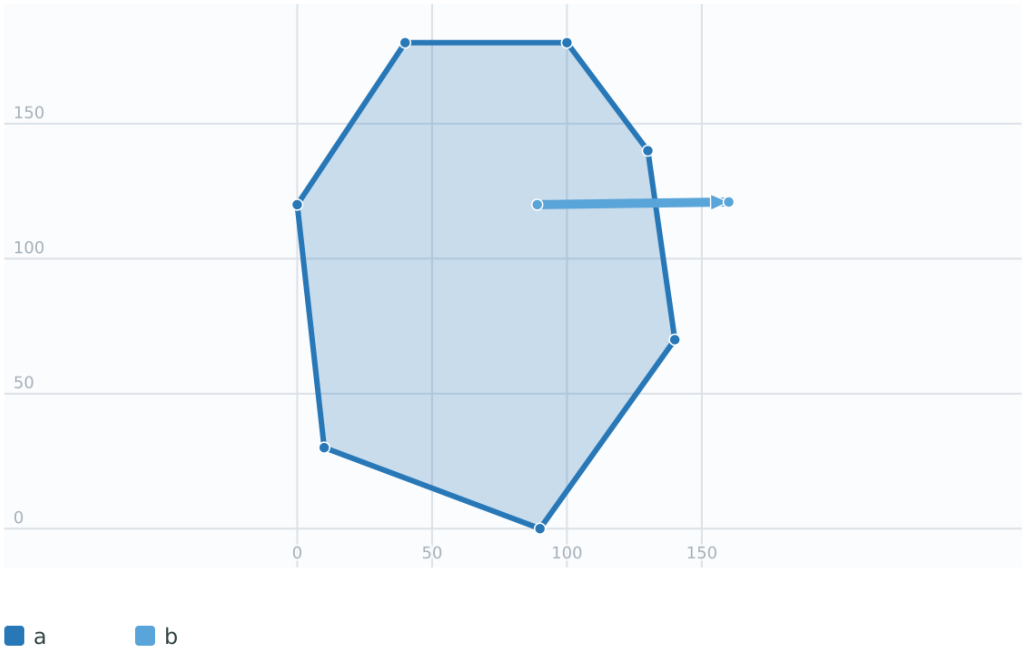
Code

```
SELECT ST_Contains(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'LINESTRING (99 130,170 131)'::geometry AS b
) AS example;
```

Output

```
f
```

Figure



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）：

A LineString does not contain a Point lying only on its boundary.

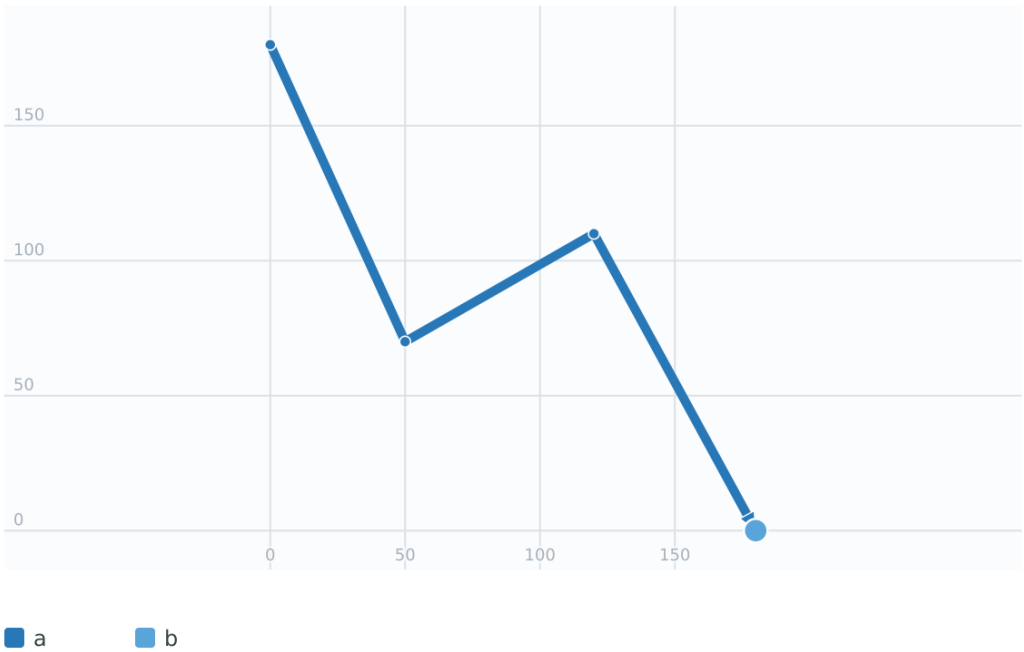
Code

```
SELECT ST_Contains(a, b), ST_Covers(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'POINT (190 10)'::geometry AS b
) AS example;
```

Output

```
f | t
```

Figure



A Polygon does not contain a LineString lying entirely on its boundary.

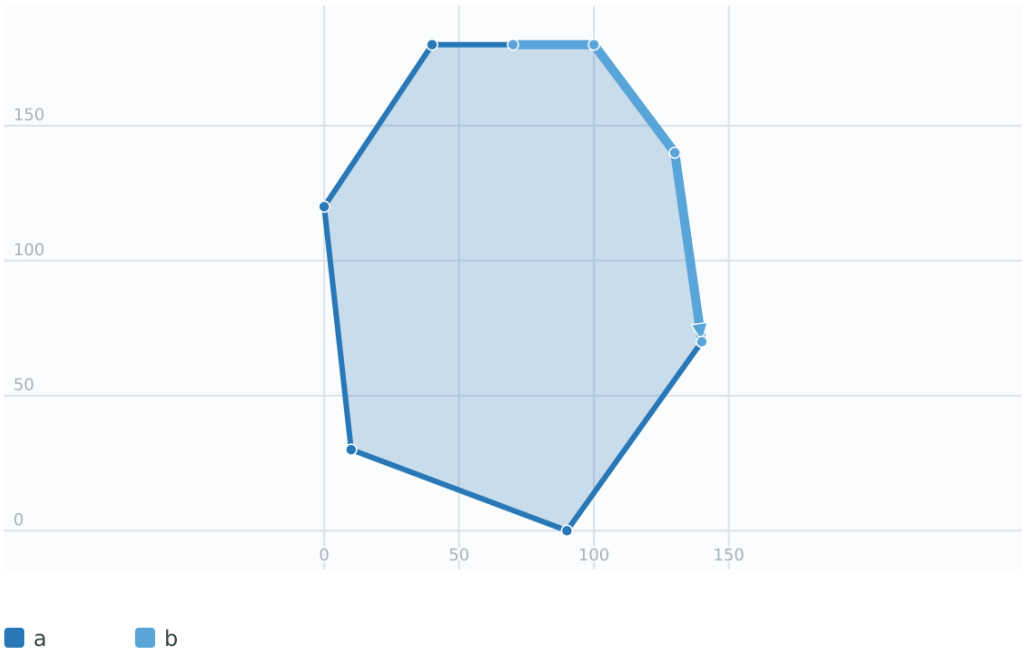
Code

```
SELECT ST_Contains(a, b), ST_Covers(a, b)
FROM (SELECT
  'POLYGON ((10 130,50 190,110 190,140 150,150 80,100 10,20 40,10 130))'::geometry AS a,
  'LINESTRING (80 190,110 190,140 150,150 80)'::geometry AS b
) AS example;
```

Output

```
f | t
```

Figure



This example tests a circle within another circle.

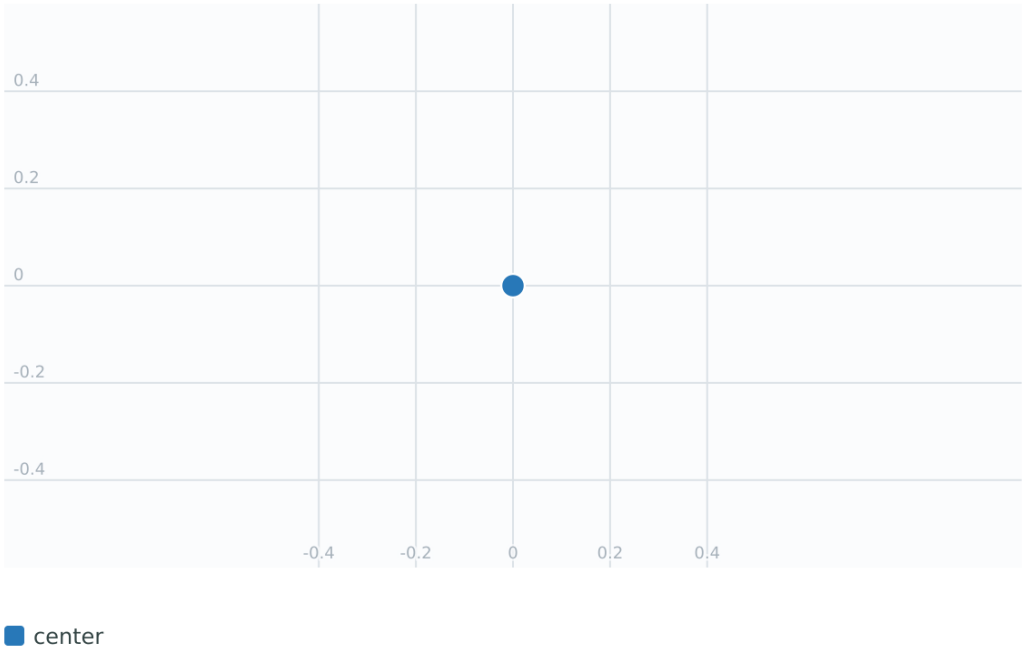
Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_Contains(smallc, bigc) AS smallcontainsbig,
  ST_Contains(bigc, smallc) AS bigcontainssmall,
  ST_Contains(bigc, ST_Union(smallc, bigc)) AS bigcontainsunion,
  ST_Equals(bigc, ST_Union(smallc, bigc)) AS bigisunion,
  ST_Covers(bigc, ST_ExteriorRing(bigc)) AS bigcoversexterior,
  ST_Contains(bigc, ST_ExteriorRing(bigc)) AS bigcontainsexterior
FROM circles;
```

Output

```
-[ RECORD 1 ]-----
smallcontainsbig   | f
bigcontainssmall   | t
bigcontainsunion   | t
bigisunion         | t
bigcoversexterior  | t
bigcontainsexterior | f
```

Figure



This example demonstrates the difference between contains and contains-properly predicates.

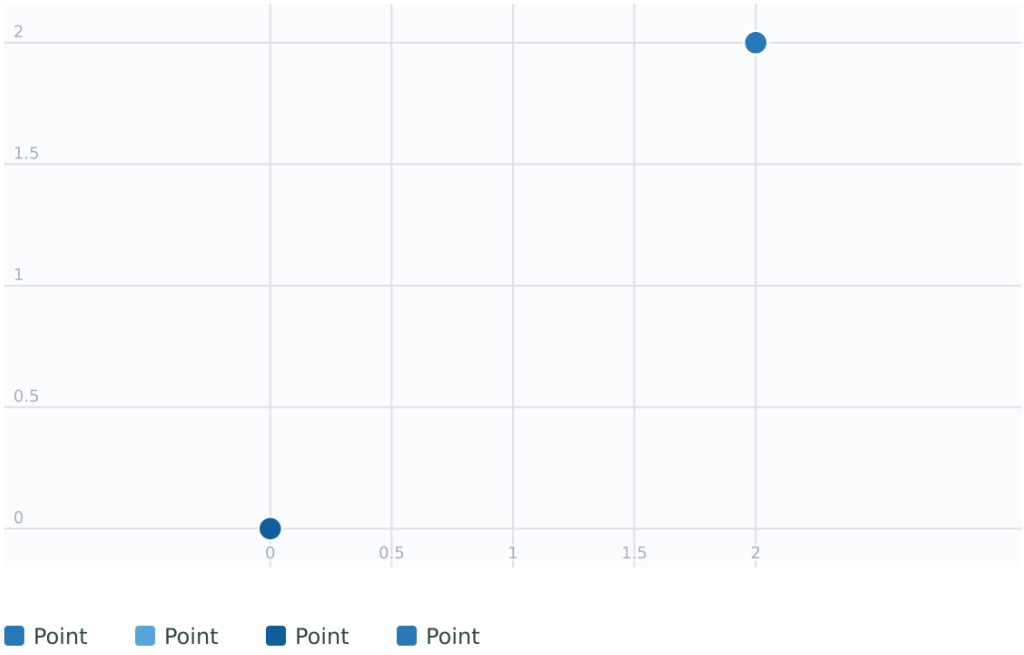
Code

```
SELECT
  ST_GeometryType(geom_a) AS geomtype,
  ST_Contains(geom_a, geom_a) AS acontainsa,
  ST_ContainsProperly(geom_a, geom_a) AS acontainspropa,
  ST_Contains(geom_a, ST_Boundary(geom_a)) AS acontainsba,
  ST_ContainsProperly(geom_a, ST_Boundary(geom_a)) AS acontainspropba
FROM (VALUES
    (ST_Buffer(ST_Point(1, 1), 5, 1)),
    (ST_MakeLine(ST_Point(1, 1), ST_Point(-1, -1))),
    (ST_Point(1, 1))
  ) AS foo(geom_a);
```

Output

```
geomtype    | acontainsa | acontainspropa | acontainsba | acontainspropba
------------+------------+----------------+-------------+-----------------
ST_Polygon    | t          | f              | f           | f
ST_LineString | t          | f              | f           | f
ST_Point      | t          | t              | f           | f
```

Figure



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

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

This example tests a circle within another circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_ContainsProperly(smallc, bigc) AS smallcontainspropbig,
  ST_ContainsProperly(bigc, smallc) AS bigcontainspropsmall,
  ST_ContainsProperly(bigc, ST_Union(smallc, bigc)) AS bigcontainspropunion,
  ST_Equals(bigc, ST_Union(smallc, bigc)) AS bigisunion,
  ST_Covers(bigc, ST_ExteriorRing(bigc)) AS bigcoversexterior,
  ST_ContainsProperly(bigc, ST_ExteriorRing(bigc)) AS bigcontainsexterior
FROM circles;
```

Output

```
-[ RECORD 1 ]-----
smallcontainspropbig   | f
bigcontainspropsmall   | t
bigcontainspropunion   | f
bigisunion             | t
bigcoversexterior      | t
bigcontainsexterior    | f
```

Figure



This example demonstrates the difference between contains and contains-properly predicates.

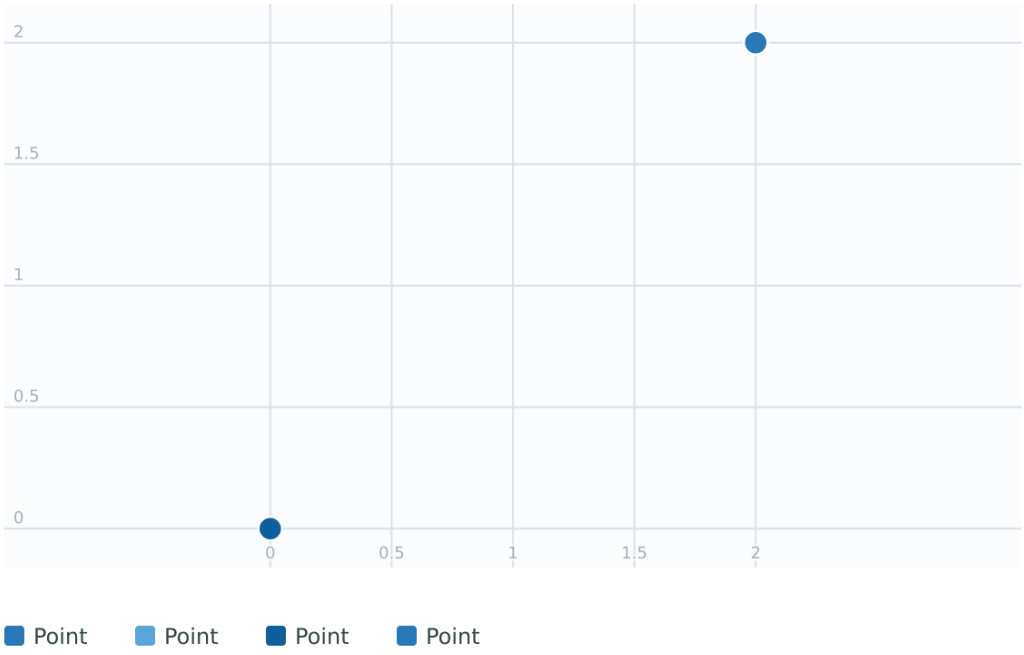
Code

```
WITH input(geom) AS (VALUES
  (ST_Buffer(ST_Point(1, 1), 5, 1)),
  (ST_MakeLine(ST_Point(1, 1), ST_Point(-1, -1))),
  (ST_Point(1, 1))
)
SELECT ST_GeometryType(geom) AS geomtype,
       ST_Contains(geom, geom) AS acontainsa,
       ST_ContainsProperly(geom, geom) AS acontainspropa,
       ST_Contains(geom, ST_Boundary(geom)) AS acontainsba,
       ST_ContainsProperly(geom, ST_Boundary(geom)) AS acontainspropba
FROM input;
```

Output

```
geomtype    | acontainsa | acontainspropa | acontainsba | acontainspropba
------------+------------+----------------+-------------+-----------------
ST_Polygon    | t          | f              | f           | f
ST_LineString | t          | f              | f           | f
ST_Point      | t          | t              | f           | f
```

Figure



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☒： $ST_CoveredBy(A, B) \Leftrightarrow A \cap B = A$

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`。

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

This example tests a circle covered by another circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_CoveredBy(smallc, smallc) AS smallinsmall,
  ST_CoveredBy(smallc, bigc) AS smallcoveredbybig,
  ST_CoveredBy(ST_ExteriorRing(bigc), bigc) AS exteriorcoveredbybig,
  ST_Within(ST_ExteriorRing(bigc), bigc) AS exteriorwithinbig
FROM circles;
```

Output

```
smallinsmall | smallcoveredbybig | exteriorcoveredbybig | exteriorwithinbig
-------------+-------------------+----------------------+------------------
 t           | t                 | t                    | f
(1 row)
```

Figure



相关信息

ST_Contains, ST_Covers, ST_ExteriorRing, ST_Within

7.11.1.5 ST_Covers

ST_Covers — 区区 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（即与其内部或区界相交），区返回 `true`。同区，区区 B 中没有点位于 A 外部（外部）。

用数学区区来区：*ST_Covers(A, B)* ⇔ *A* ▯ *B* = *B*

ST_Covers 与 ST_CoveredBy相反。因此，`ST_Covers(A,B) = ST_CoveredBy(B,A)`。

一般来区，区区使用区个函数而不是ST_Contains，因区它有一个更区区的定区，不存在“几何区形不包含其区界”的奇怪情况。



Note
此功能自区包括利用几何上可用的任何空区索引的区界框比区。区了避免使用索引，区使用函数 `_ST_Covers`。

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

This example tests a circle covering another circle.

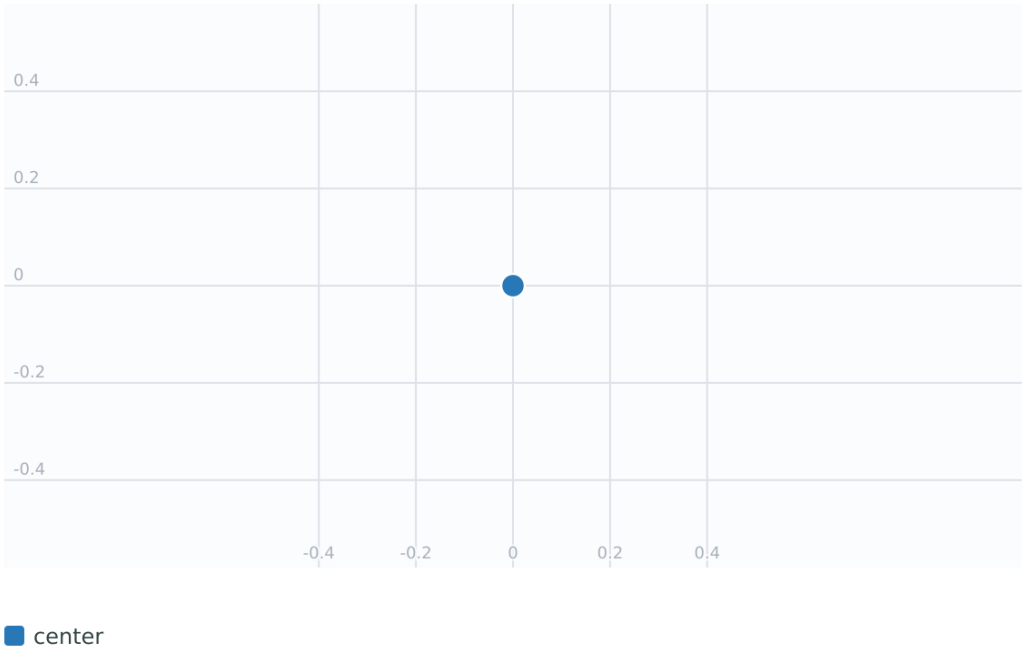
Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 10) AS smallc,
         ST_Buffer(center, 20) AS bigc
  FROM (SELECT 'POINT(1 2)'::geometry AS center) AS p
)
SELECT
  ST_Covers(smallc, smallc) AS smallinsmall,
  ST_Covers(smallc, bigc) AS smallcoversbig,
  ST_Covers(bigc, ST_ExteriorRing(bigc)) AS bigcoversexterior,
  ST_Contains(bigc, ST_ExteriorRing(bigc)) AS bigcontainsexterior
FROM circles;
```

Output

```
smallinsmall | smallcoversbig | bigcoversexterior | bigcontainsexterior
-------------+----------------+-------------------+--------------------
 t           | f              | t                 | f
(1 row)
```

Figure



Geography example

This example compares a point with a 300-meter buffer against a point and against a 10-meter buffer around that point.

Code

```
SELECT ST_Covers(geog_poly, geog_pt) As poly_covers_pt,
  ST_Covers(ST_Buffer(geog_pt, 10), geog_pt) As buff_10m_covers_cent
  FROM (SELECT ST_Buffer('SRID=4326;POINT(-99.327 31.4821)'::geography, 300) As geog_poly,
        'SRID=4326;POINT(-99.33 31.483)'::geography As geog_pt ) As foo;
```

Output

```
poly_covers_pt | buff_10m_covers_cent
---------------+---------------------
 f             | t
```

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⊠：$ST_Crosses(A, B) \Leftrightarrow (dim(\ Int(A) \cap Int(B)\ ) < max(\ dim(\ Int(A)\ ), dim(\ Int(B)\ )\ )) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$

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⊠。

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

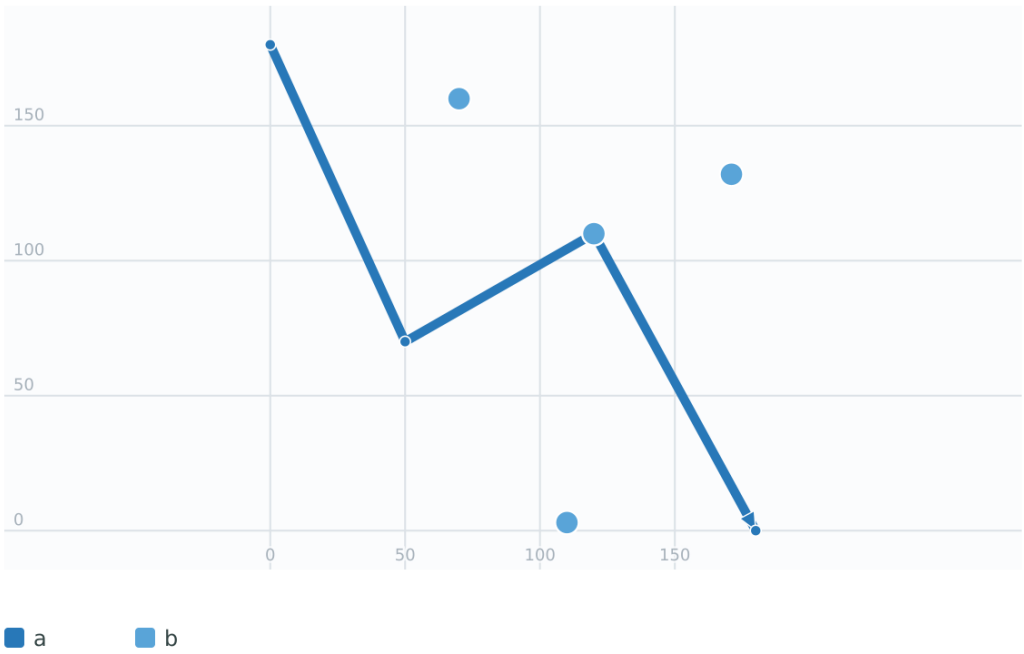
以下情况均返回 `true`。

A LineString crosses a MultiPoint when some, but not all, points lie on the line.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'MULTIPOINT (80 170,120 13,130 120,181 142)'::geometry AS b
) AS example;
```

Figure

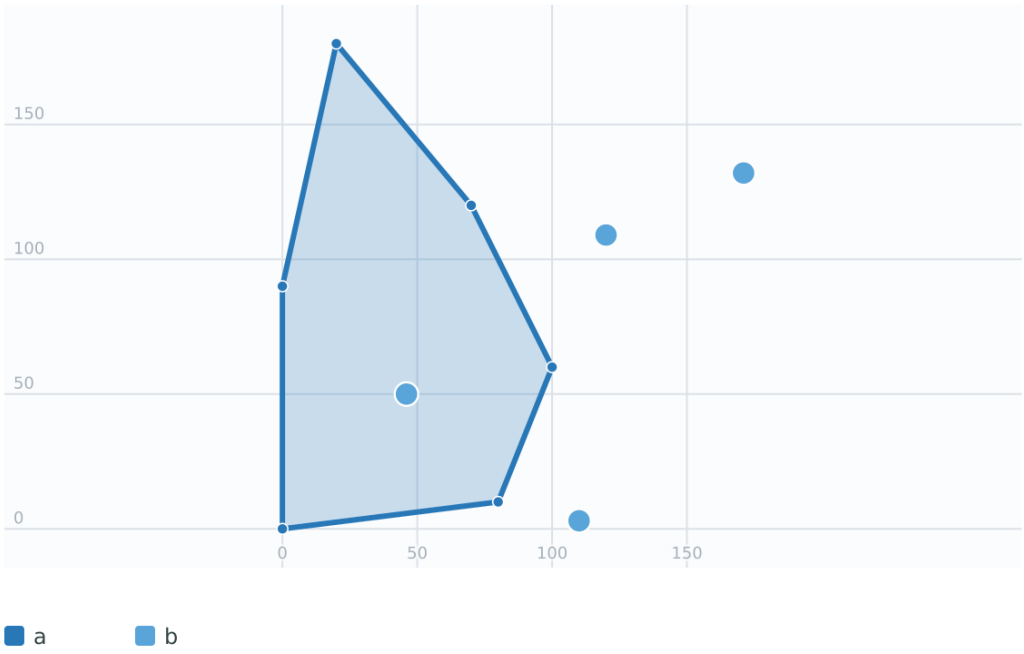


A Polygon crosses a MultiPoint when some, but not all, points lie in its interior.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'POLYGON ((30 190,10 100,10 10,90 20,110 70,80 130,30 190))'::geometry AS a,
  'MULTIPOINT (56 60,120 13,130 119,181 142)'::geometry AS b
) AS example;
```

Figure



A LineString crosses a Polygon when part of the line lies inside and part outside.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'POLYGON ((30 190,10 100,10 10,90 20,110 70,80 130,30 190))'::geometry AS a,
  'LINESTRING (30 40,70 50,120 150,190 190)'::geometry AS b
) AS example;
```

Figure

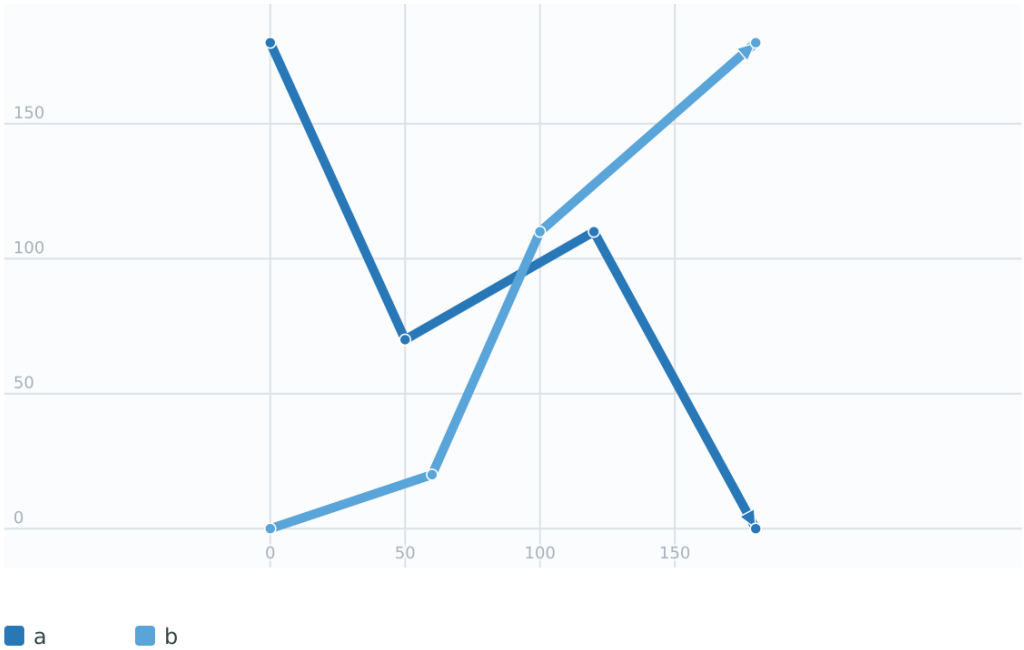


Two LineStrings cross when they share an interior point.

Code

```
SELECT ST_Crosses(a, b)
FROM (SELECT
  'LINESTRING (10 190,60 80,130 120,190 10)'::geometry AS a,
  'LINESTRING (10 10,70 30,110 120,190 190)'::geometry AS b
) AS example;
```

Figure



考⊠用⊠有⊠个表的情况：道路表和高速公路表。

Code

```
CREATE TABLE roads (
  id serial NOT NULL,
  geom geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↩
    road_id)
);
```

Code

```
CREATE TABLE highways (
  id serial NOT NULL,
  the_gem geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↩
    road_id)
);
```

要确定穿⊠高速公路的道路列表，⊠使用⊠似于以下内容的⊠⊠：

Code

```
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 = ∅*
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****')

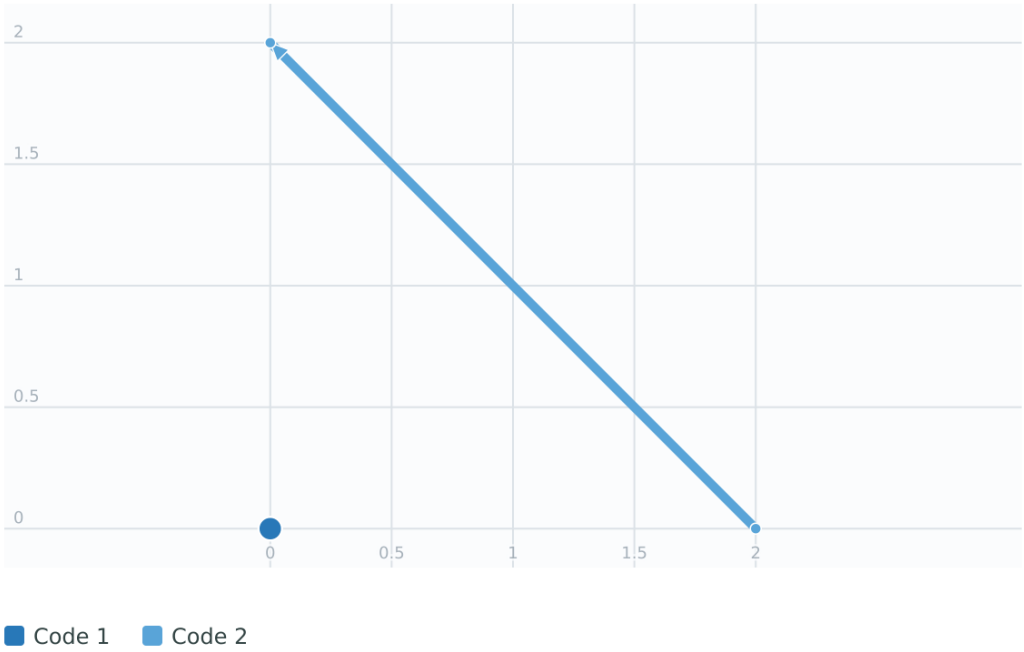
⊠方法⊠⊠了 SQL/MM ⊠范。SQL-MM 3: 5.1.26

示例

Code

```
SELECT ST_Disjoint('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

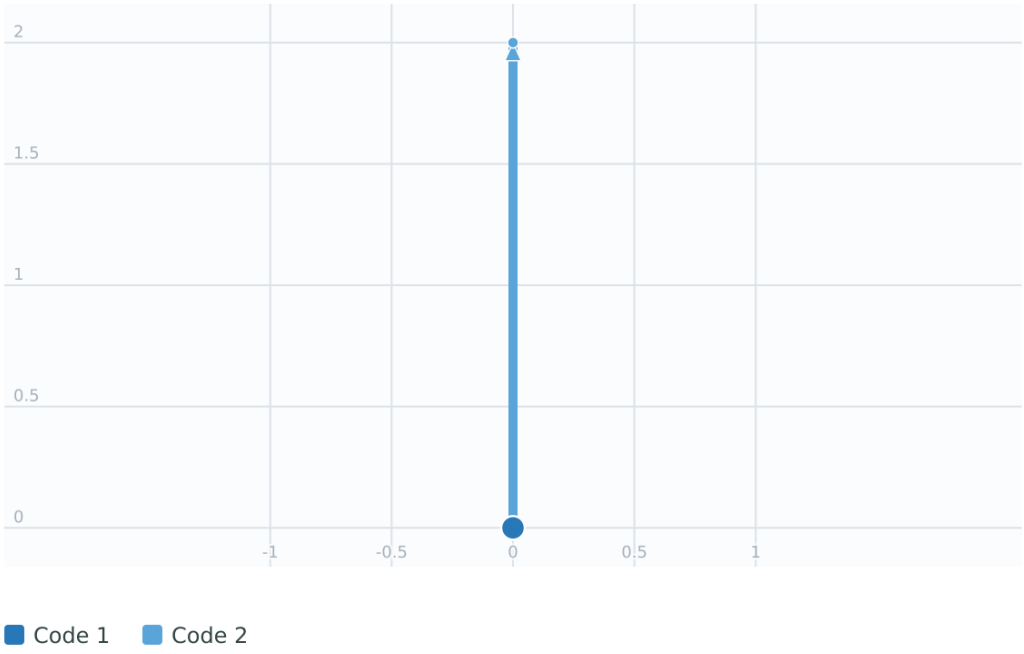
Figure



Code

```
SELECT ST_Disjoint('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Figure



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☒：$ST_Equals(A, B) \Leftrightarrow A = B$

以下关系成立：$ST_Equals(A, B) \Leftrightarrow ST_Within(A,B) \wedge ST_Within(B,A)$

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

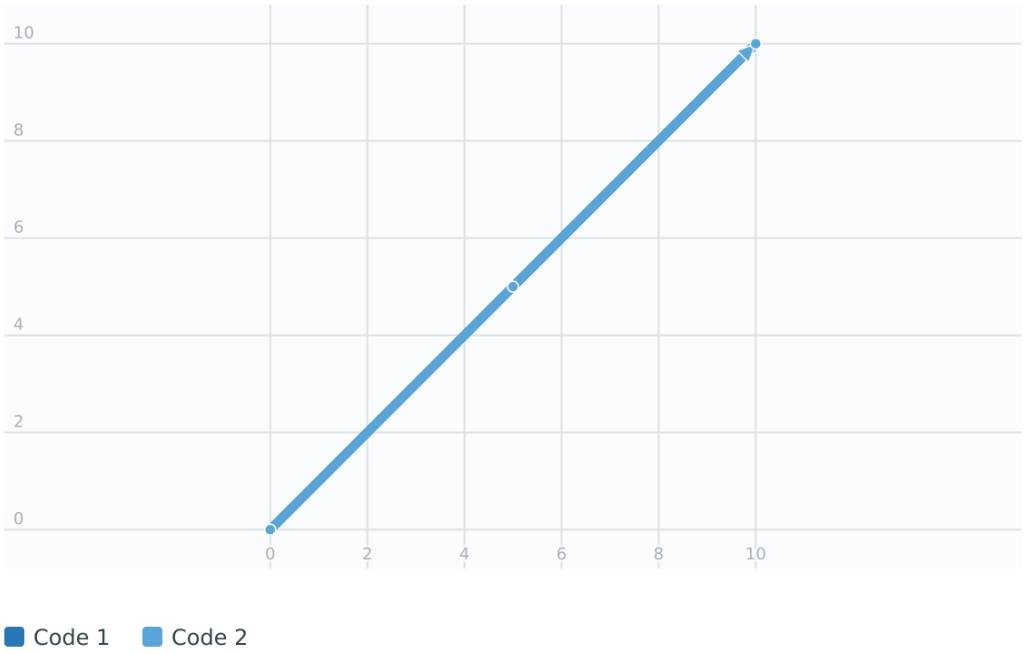
更改：2.2.0 即使☒于无效几何☒形，如果它☒二☒制相等，也会返回 true

示例

Code

```
SELECT ST_Equals('LINESTRING(0 0,10 10)',
'LINESTRING(0 0,5 5,10 10)');
```

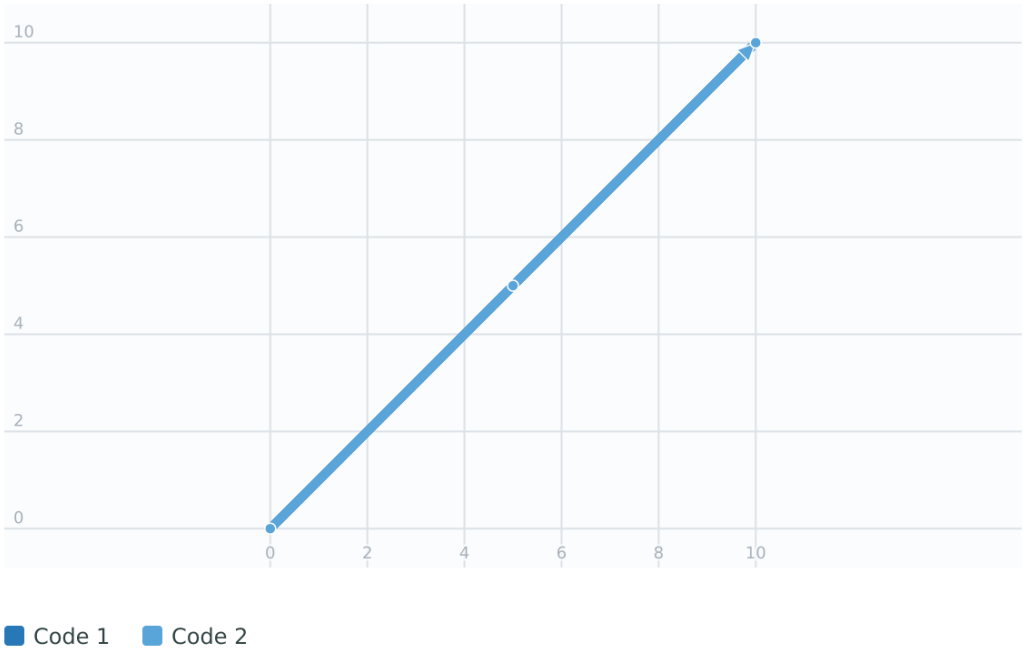
Figure



Code

```
SELECT ST_Equals(ST_Reverse('LINESTRING(0 0,10 10)'),
'LINESTRING(0 0,5 5,10 10)');
```

Figure



相关信息

ST_IsValid, ST_OrderingEquals, ST_Reverse, ST_Within

7.11.1.9 ST_Intersects

ST_Intersects — 区区区个几何区形是否相交（它区至少有一个共同点）

Synopsis

boolean **ST_Intersects**(geometry geomA , geometry geomB);
boolean **ST_Intersects**(geography geogA , geography geogB);

描述

如果区个几何区形相交，区返回 `true`。如果几何区形有任何共同点，区它区相交。

区于地理，使用 0.00001 米的距离容差（因此非常接近的点被区区相交）。

用数学区区来区：*ST_Intersects(A, B)* ⇔ *A* ∩ *B* ≠ ∅

如果几何区形的 DE-9IM 交集矩区与以下之一匹配，区几何区形相交：

- 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

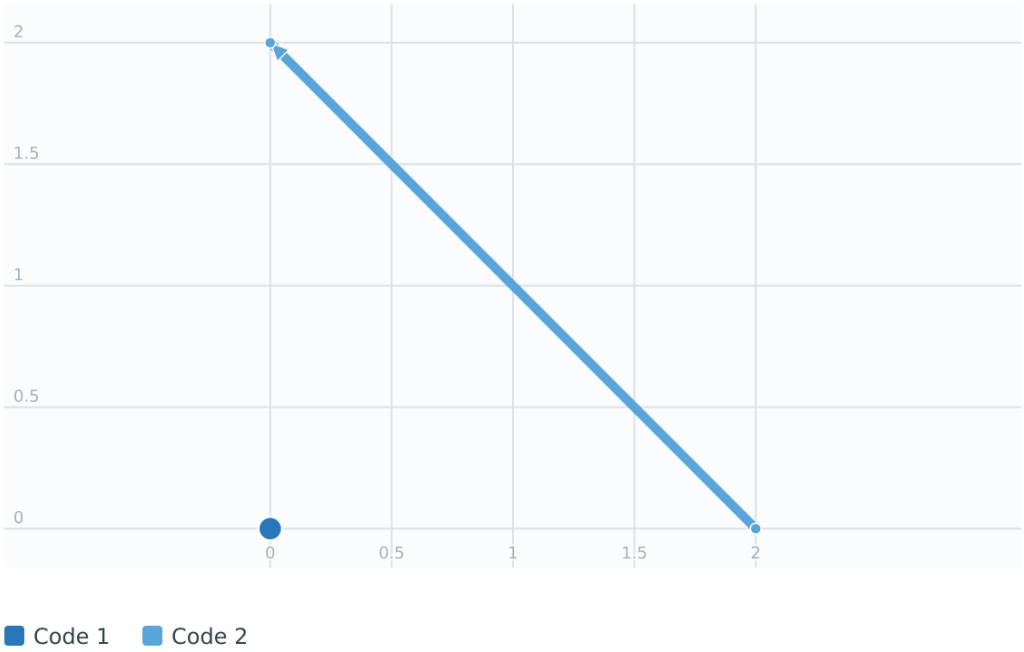
Code

```
SELECT ST_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
f
```

Figure



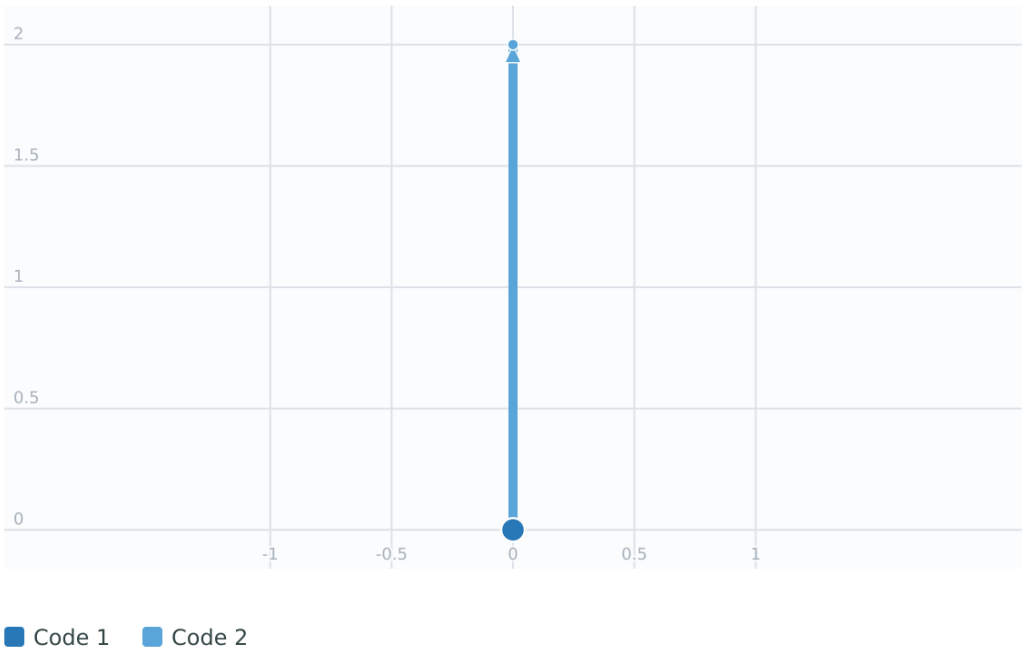
Code

```
SELECT ST_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Output

```
t
```

Figure



This example looks up rows in a table; for faster lookup, the geometry column should have a GiST index.

Code

```
SELECT id, name FROM cities WHERE ST_Intersects(geom, 'SRID=4326;POLYGON((28 53,27.707 ↩
    52.293,27 52,26.293 52.293,26 53,26.293 53.707,27 54,27.707 53.707,28 53))');
```

Output

```
id | name
----+-------
  2 | Minsk
(1 row)
```

几何示例

Code

```
SELECT ST_Intersects('SRID=4326;LINESTRING(-43.23456 72.4567,-43.23456 72.4568)'::geography ↩
    ,
'SRID=4326;POINT(-43.23456 72.4567772)'::geography
);
```

Output

```
t
```

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⍰

可用性：1.4

示例

示例：向左交叉⌧和向右交叉⌧

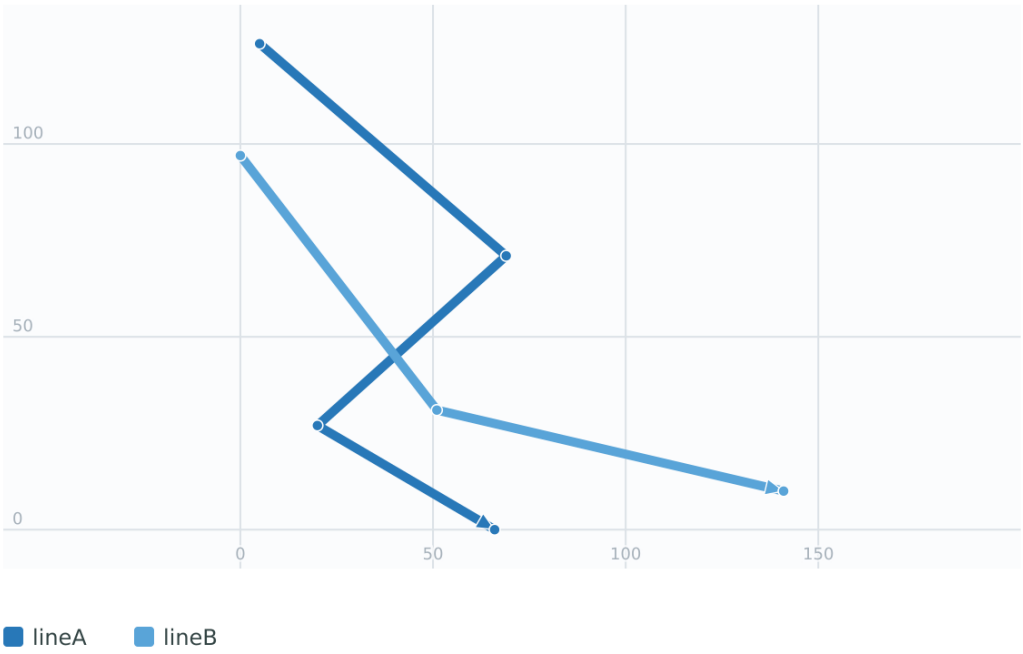
Code

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
  'LINESTRING(25 169,89 114,40 70,86 43)'::geometry As lineA,
  'LINESTRING (20 140,71 74,161 53)'::geometry As lineB
  ) As foo;
```

Output

```
A_cross_B | B_cross_A
----------+----------
       -1 |         1
```

Figure



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⌧

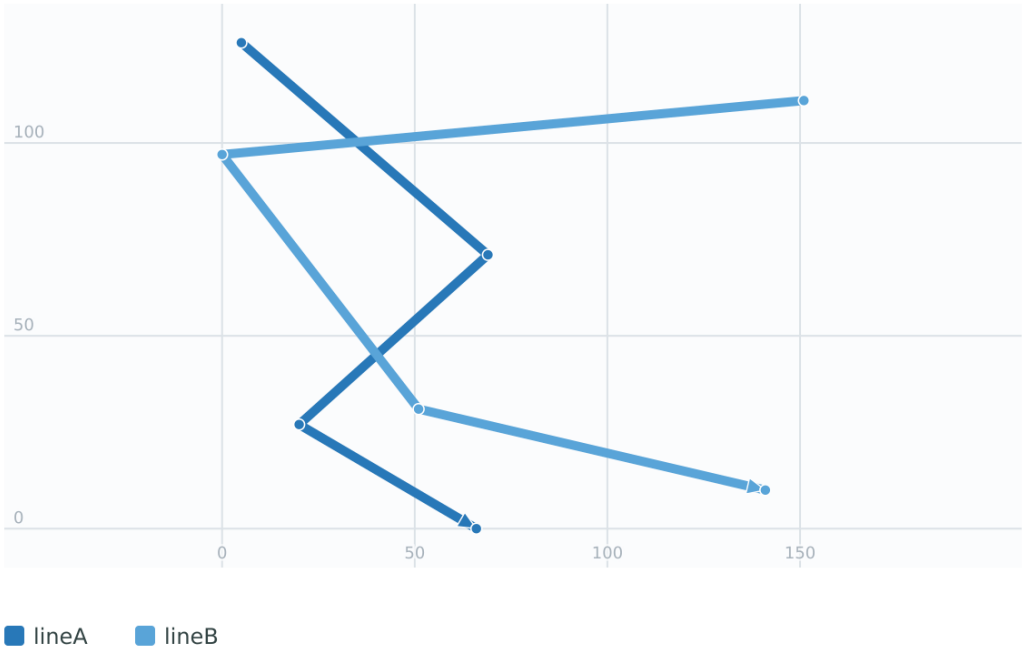
Code

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
 'LINESTRING(25 169,89 114,40 70,86 43)'::geometry As lineA,
 'LINESTRING(171 154,20 140,71 74,161 53)'::geometry As lineB
  ) As foo;
```

Output

```
A_cross_B | B_cross_A
----------+----------
        3 |        -3
```

Figure



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⊠

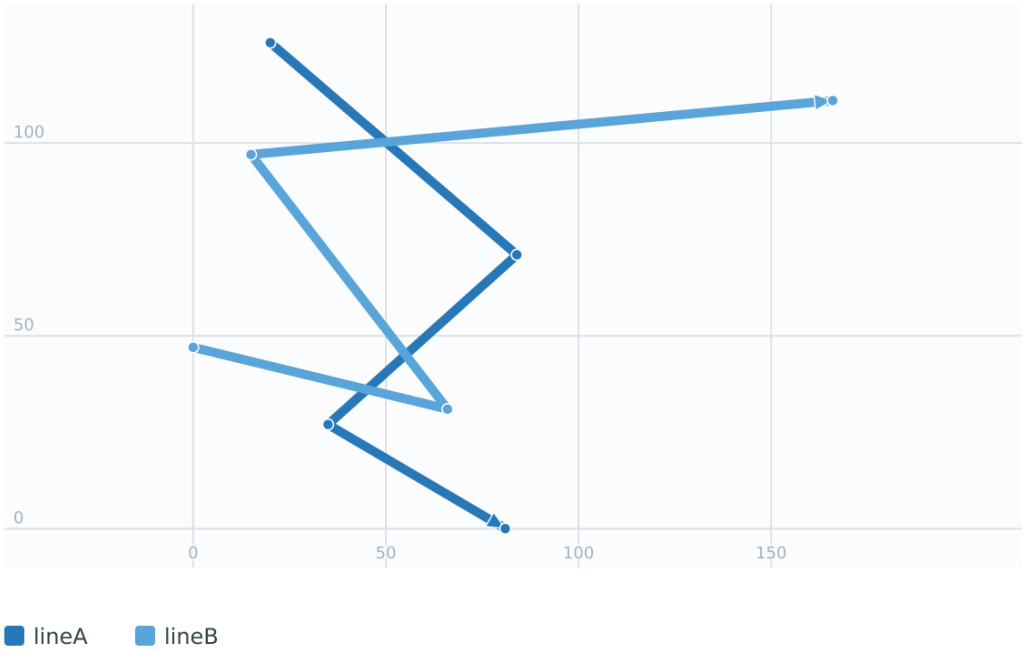
Code

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
  'LINESTRING(25 169,89 114,40 70,86 43)'::geometry As lineA,
  'LINESTRING(5 90,71 74,20 140,171 154)'::geometry As lineB
  ) As foo;
```

Output

```
A_cross_B | B_cross_A
----------+----------
       -2 |        2
```

Figure



示例：⊠找所有交叉的街道

Code

```
SELECT s1.gid, s2.gid, ST_LineCrossingDirection(s1.geom, s2.geom)
  FROM streets s1 CROSS JOIN streets s2
        ON (s1.gid != s2.gid AND s1.geom && s2.geom )
WHERE ST_LineCrossingDirection(s1.geom, s2.geom)
> 0;
```

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

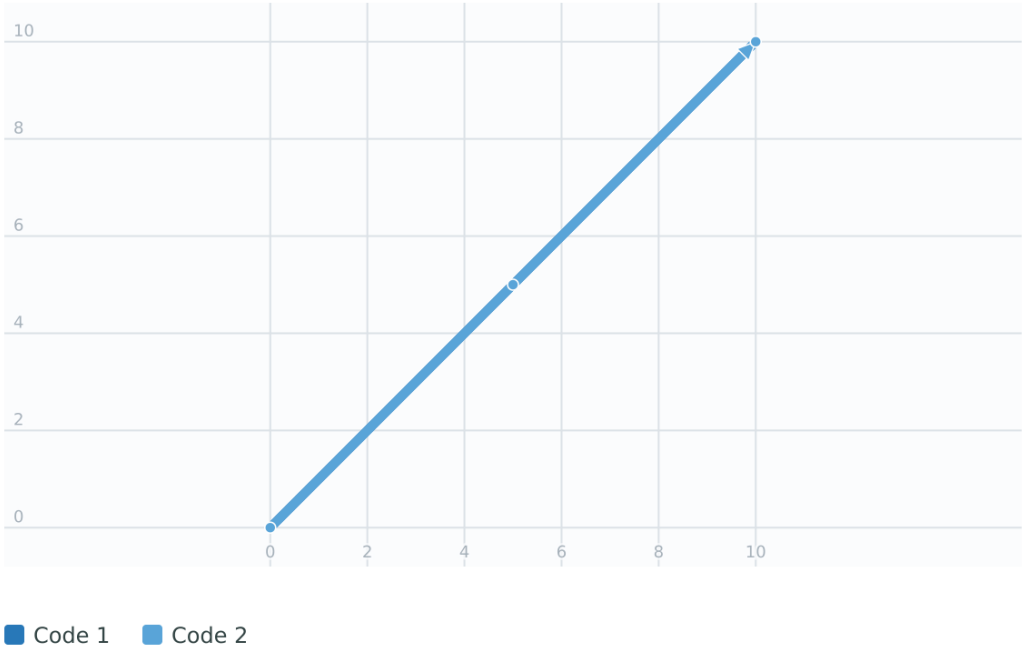
Code

```
SELECT ST_OrderingEquals('LINESTRING(0 0,10 10)',
'LINESTRING(0 0,5 5,10 10)');
```

Output

```
f
```

Figure



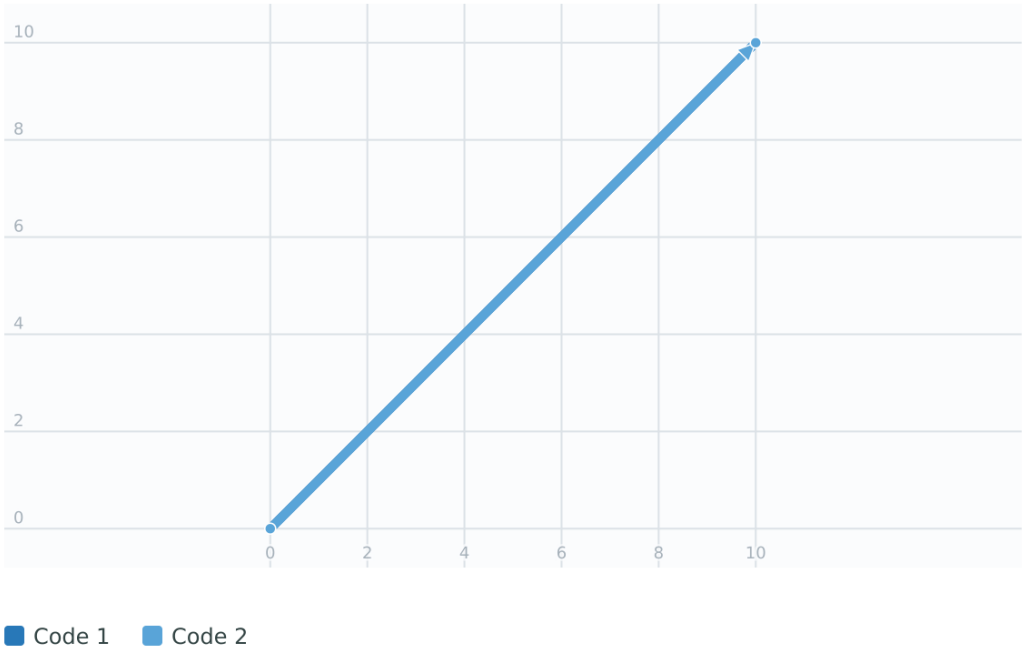
Code

```
SELECT ST_OrderingEquals('LINESTRING(0 0,10 10)',
'LINESTRING(0 0,10 10)');
```

Output

```
t
```

Figure



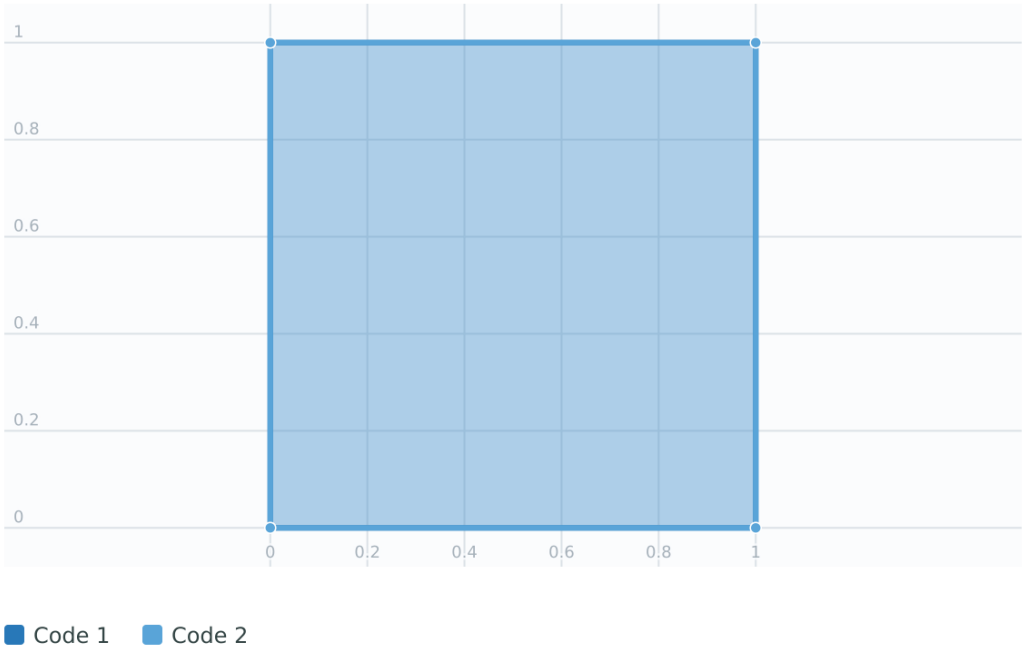
Code

```
SELECT ST_OrderingEquals('POLYGON((0 0,0 1,1 1,1 0,0 0))',
'POLYGON((0 0,1 0,1 1,0 1,0 0))');
```

Output

```
f
```

Figure



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

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：

Two MultiPoints overlap when they share some, but not all, points.

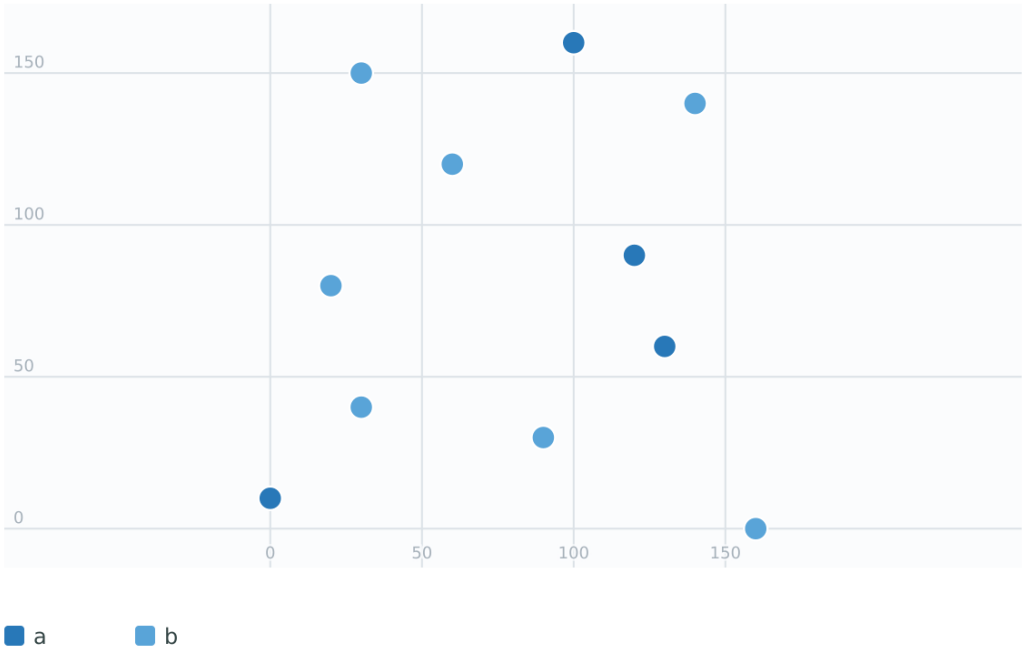
Code

```
SELECT ST_Overlaps(a, b)
FROM (SELECT
  'MULTIPOINT ((20 20),(40 90),(80 130),(140 100),(110 40),(120 170),(150 70))'::geometry ↩
    AS a,
  'MULTIPOINT ((180 10),(40 90),(80 130),(160 150),(110 40),(50 50),(50 160))'::geometry AS ↩
     b
) AS example;
```

Output

```
t
```

Figure



Two LineStrings overlap along part of their paths.

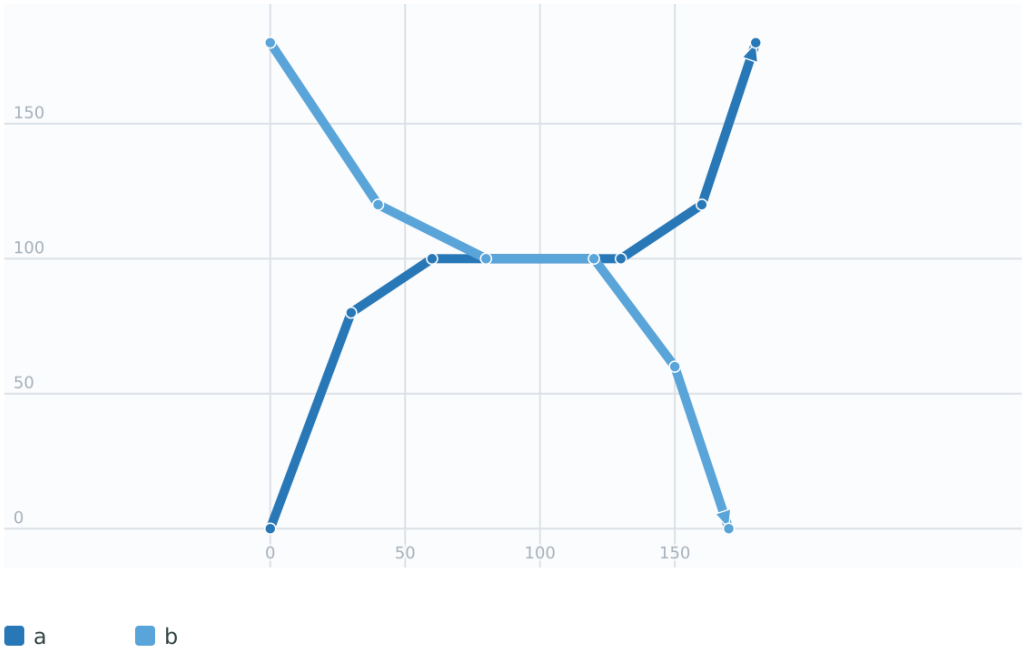
Code

```
SELECT ST_Overlaps(a, b)
FROM (SELECT
  'LINESTRING (10 10,40 90,70 110,140 110,170 130,190 190)'::geometry AS a,
  'LINESTRING (10 190,50 130,90 110,130 110,160 70,180 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two Polygons overlap where their interiors intersect and neither contains the other.

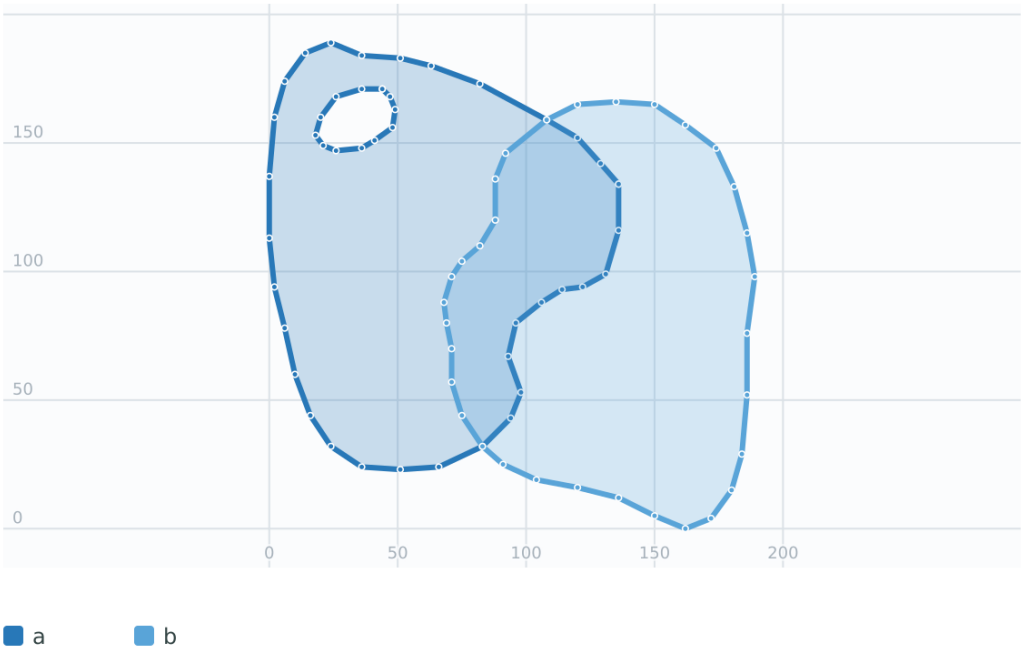
Code

```
SELECT ST_Overlaps(a, b)
FROM (SELECT
  'POLYGON ((140 140,140 122,135 105,126 100,118 99,110 94,100 86,97 73,102 59,98 49,87 ↩
    38,70 30,55 29,40 30,28 38,20 50,14 66,10 84,6 100,4 119,4 143,6 166,10 180,18 191,28 ↩
    195,40 190,55 189,67 186,86 179,112 165,124 158,133 148,140 140),(48 177,40 177,30 ↩
    174,24 166,22 159,25 155,30 153,40 154,45 157,52 162,53 169,51 174,48 177))'::geometry ↩
    AS a,
  'POLYGON ((75 63,79 50,87 38,95 31,108 25,124 22,140 18,154 11,166 6,176 10,184 21,188 ↩
    35,190 58,190 82,193 104,190 121,185 139,178 154,166 163,154 171,139 172,124 171,112 ↩
    165,96 152,92 142,92 126,86 116,79 110,75 104,72 94,73 86,75 76,75 63))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



包含 LineString 上的点，但由于它的⌧度⌧低，因此不会重⌧或交叉。

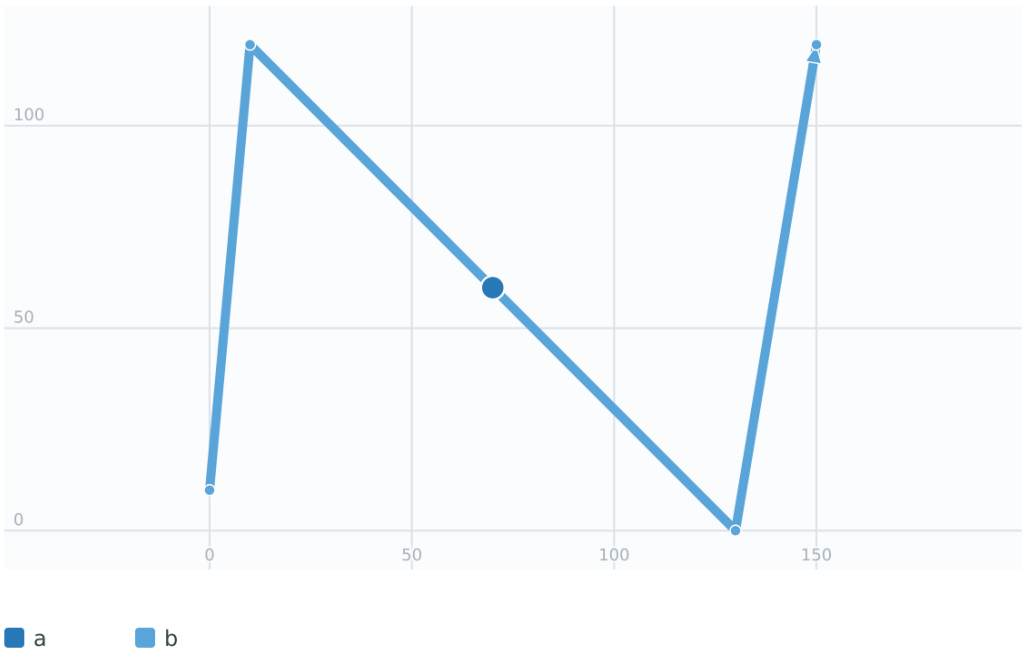
Code

```
SELECT ST_Overlaps(a, b) AS overlaps, ST_Crosses(a, b) AS crosses,
       ST_Intersects(a, b) AS intersects, ST_Contains(b, a) AS b_contains_a
FROM (SELECT 'POINT (100 100)'::geometry As a,
             'LINESTRING (30 50,40 160,160 40,180 160)'::geometry  AS b) AS t
```

Output

```
overlaps | crosses | intersects | b_contains_a
---------+---------+------------+-------------
f        | f       | t          | t
```

Figure



部分覆盖多⌧形的⌧串相交和交叉，但不重⌧，因⌧它具有不同的⌧度。

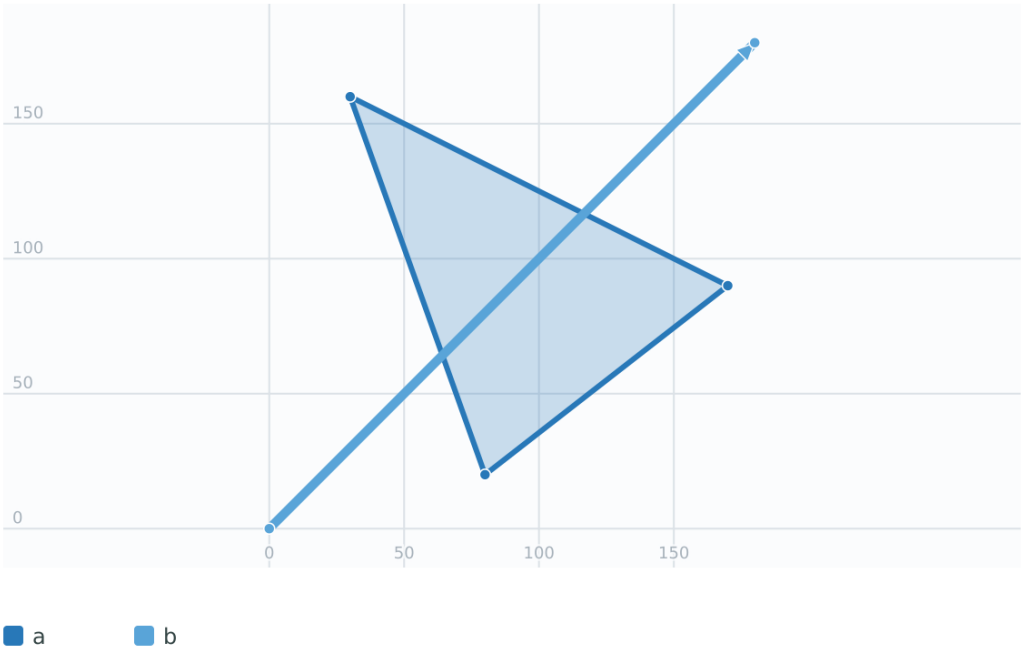
Code

```
SELECT ST_Overlaps(a, b) AS overlaps, ST_Crosses(a, b) AS crosses,
       ST_Intersects(a, b) AS intersects, ST_Contains(a, b) AS contains
FROM (SELECT 'POLYGON ((40 170,90 30,180 100,40 170))'::geometry AS a,
             'LINESTRING(10 10,190 190)'::geometry AS b) AS t;
```

Output

```
overlap | crosses | intersects | contains
---------+---------+------------+--------------
 f       | t       | t          | f
```

Figure



⊠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。

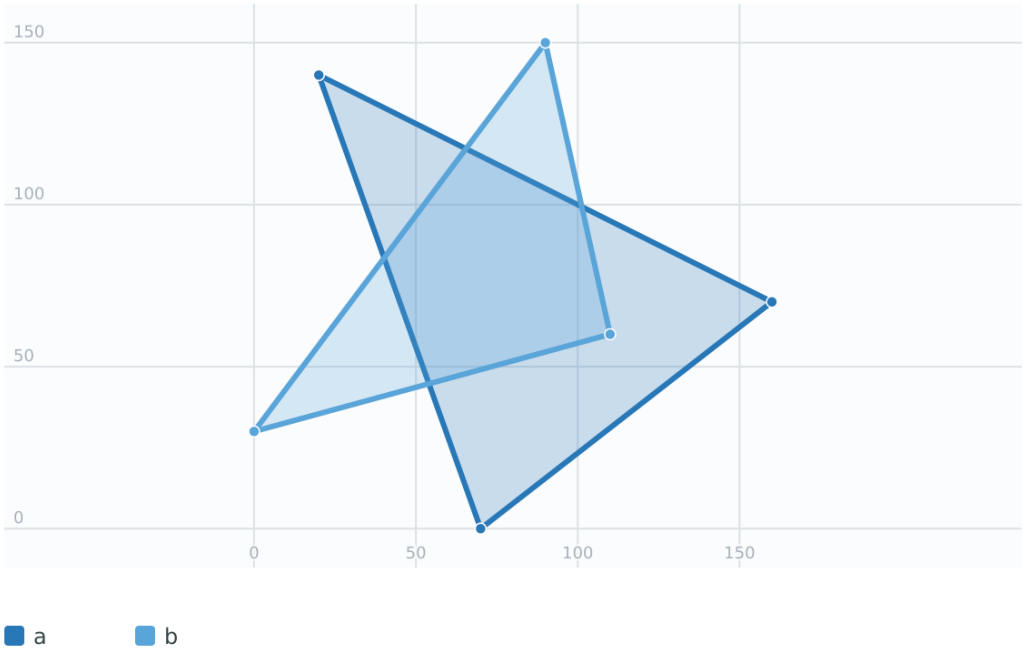
Code

```
SELECT ST_Overlaps(a, b) AS overlaps, ST_Crosses(a, b) AS crosses,
       ST_Intersects(a, b) AS intersects, ST_Contains(b, a) AS b_contains_a,
       ST_Dimension(a) AS dim_a, ST_Dimension(b) AS dim_b,
       ST_Dimension(ST_Intersection(a, b)) AS dim_int
FROM (SELECT 'POLYGON ((40 170,90 30,180 100,40 170))'::geometry AS a,
             'POLYGON ((110 180,20 60,130 90,110 180))'::geometry AS b) As t;
```

Output

```
overlaps | crosses | intersects | b_contains_a | dim_a | dim_b | dim_int
---------+---------+------------+--------------+-------+-------+----------
 t       | f       | t          | f            |     2 |     2 |       2
```

Figure



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
与大多数命名空区关系区区不同，区不会自区包含索引区用。原因是某些关系区于不相交的几何区形是正确的（例如不相交）。如果您使用需要交集的关系模式，区在函数区用中包含 &&。

Note
如果可用，最好使用命名关系函数，因区它区会自区使用存在的空区索引。此外，他区区可以区区完全相关区估所无法区区的性能区化。

形式 **2**：返回区个区入几何之区的空区关系的 DE-9IM 矩区字符串。可以使用 ST_RelateMatch 区区矩区字符串与 DE-9IM 模式匹配。

形式 **3**：与形式 2 区似，但允区指定区界区点区区。区界区点区区允区更好地控制 MultiLineString 的端点是否被区区位于 DE-9IM 内部或区界内。`boundaryNodeRule` 区区：

- 1：**OGC-Mod2** - 如果区条端点出区奇数次，区它区位于区界内。区是 OGC SFS 区准定区的区区，也是 `ST_Relate` 的默区区区。
- 2：**Endpoint** - 所有端点都在区界内。
- 3：**MultivalentEndpoint** - 如果端点出区多次，区它区位于区界内。区句区区，区界是所有“附加”或“内部”端点（但不是“未附加/外部”端点）。
- 4：**MonovalentEndpoint** - 如果端点区出区一次，区端点位于区界内。区句区区，区界是所有“未区接”或“外部”端点。

OGC 区范中没有此函数，但它是区含的。区 s2.1.13.2

此方法区区了 SQL 1.1 的 OGC 区区功能区范。 s2.1.1.2 // s2.1.13.3

区方法区区了 SQL/MM 区范。SQL-MM 3: 5.1.25

它是通区 GEOS 模区区区的

增区：2.0.0 - 添加了区指定区界区点区区的支持。

增区：3.0.0 区用了区 `GEOMETRYCOLLECTION` 的支持

示例

使用布区区函数区区空区关系。

Code

```
SELECT ST_Relate('POINT(1 2)', ST_Buffer('POINT(1 2)', 2), '0FFFFF212');
```

Output

```
t
```

Code

```
SELECT ST_Relate('POINT(1 2)', ST_Buffer('POINT(1 2)', 2), '*FF*FF212');
```

Output

```
t
```

⍰⍰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)

Code

```
SELECT c.* , p.name As poly_name
    FROM polys AS p
    INNER JOIN compounds As c
          ON c.geom && p.geom
             AND ST_Relate(p.geom, c.geom, 'T********');
```

⍰算空⍰关系的交集矩⍰。

Code

```
SELECT ST_Relate('POINT(1 2)',
ST_Buffer('POINT(1 2)', 2));
```

Output

```
0FFFFF212
```

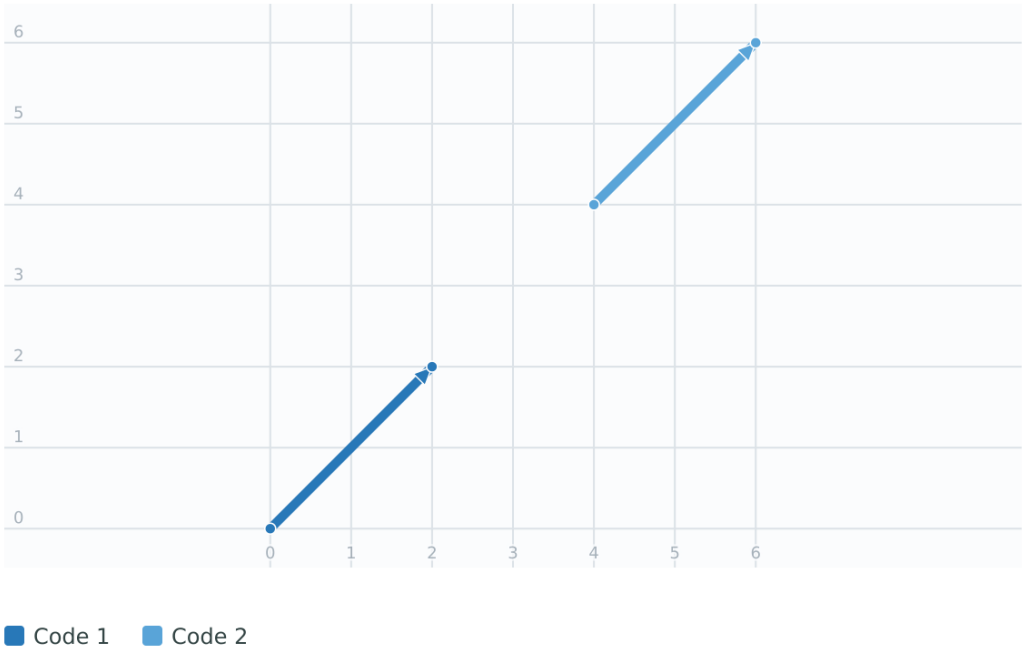
Code

```
SELECT ST_Relate('LINESTRING(1 2,3 4)',
'LINESTRING(5 6,7 8)' );
```

Output

```
FF1FF0102
```

Figure



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`。

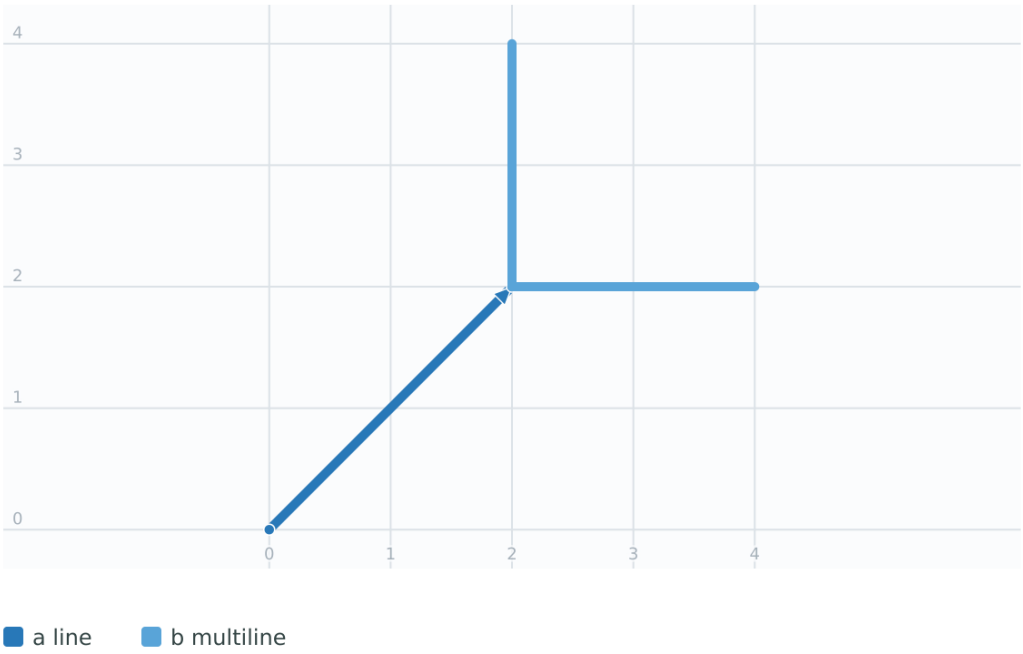
Code

```
WITH data AS (SELECT
  'LINESTRING(1 1,3 3)'::geometry AS a_line,
  'MULTILINESTRING((3 3,3 5),(3 3,5 3))'::geometry AS b_multiline
)
SELECT ST_Relate(a_line, b_multiline, 1) AS bnr_mod2,
       ST_Relate(a_line, b_multiline, 2) AS bnr_endpoint
    FROM data;
```

Output

```
bnr_mod2  | bnr_endpoint
----------+-------------
 FF10F0102 | FF1F00102
```

Figure



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

Code

```
SELECT ST_RelateMatch('101202FFF', 'TTTTTTFFF') ;
```

Output

```
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

Code

```
SELECT pat.name AS relationship, pat.val AS pattern,
       mat.name AS position, mat.val AS matrix,
       ST_RelateMatch(mat.val, pat.val) AS match
    FROM (VALUES ( 'Equality', 'T1FF1FFF1' ),
                 ( 'Overlaps', 'T*T***T**' ),
                 ( 'Within', 'T*F**F***' ),
                 ( 'Disjoint', 'FF*FF****' )) AS pat(name, val)
    CROSS JOIN
        (VALUES  ('non-intersecting', 'FF1FF0212'),
                 ('overlapping', '1010F0212'),
                 ('inside', '1FF0FF212')) AS mat(name, val);
```

Output

```
relationship |  pattern  |     position     |  matrix   | match
-------------+-----------+------------------+-----------+-------
 Equality    | T1FF1FFF1 | non-intersecting | FF1FF0212 | f
 Equality    | T1FF1FFF1 | overlapping      | 1010F0212 | f
 Equality    | T1FF1FFF1 | inside           | 1FF0FF212 | f
 Overlaps    | T*T***T** | non-intersecting | FF1FF0212 | f
 Overlaps    | T*T***T** | overlapping      | 1010F0212 | t
 Overlaps    | T*T***T** | inside           | 1FF0FF212 | f
 Within      | T*F**F*** | non-intersecting | FF1FF0212 | f
 Within      | T*F**F*** | overlapping      | 1010F0212 | f
 Within      | T*F**F*** | inside           | 1FF0FF212 | t
 Disjoint    | FF*FF**** | non-intersecting | FF1FF0212 | t
 Disjoint    | FF*FF**** | overlapping      | 1010F0212 | f
 Disjoint    | FF*FF**** | inside           | 1FF0FF212 | f
```

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。

在数学☒☒中：*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。

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。

Two Polygons touching along a boundary segment.

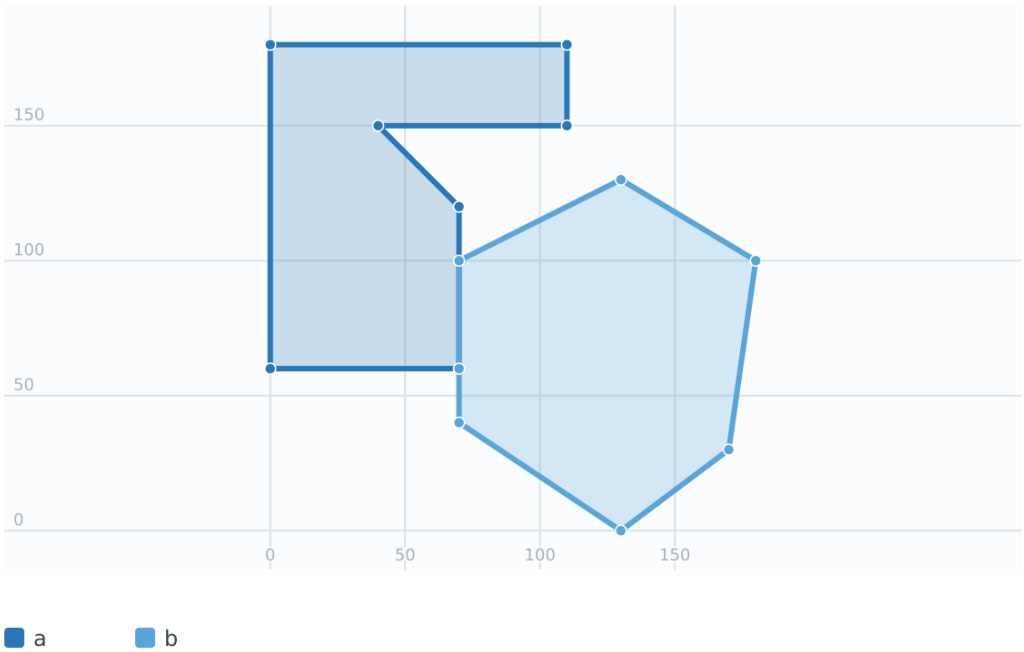
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190))'::geometry AS a,
  'POLYGON ((80 110,140 140,190 110,180 40,140 10,80 50,80 70,80 110))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two Polygons touching at a boundary point.

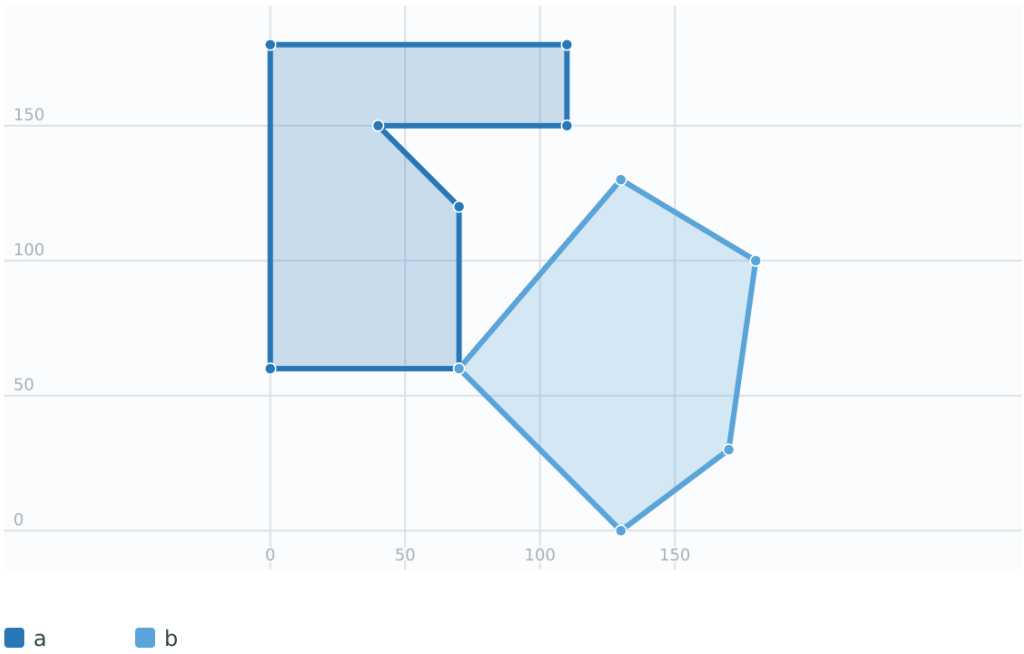
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190))'::geometry AS a,
  'POLYGON ((140 140,190 110,180 40,140 10,80 70,140 140))'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Polygon and a LineString touching at the line endpoint.

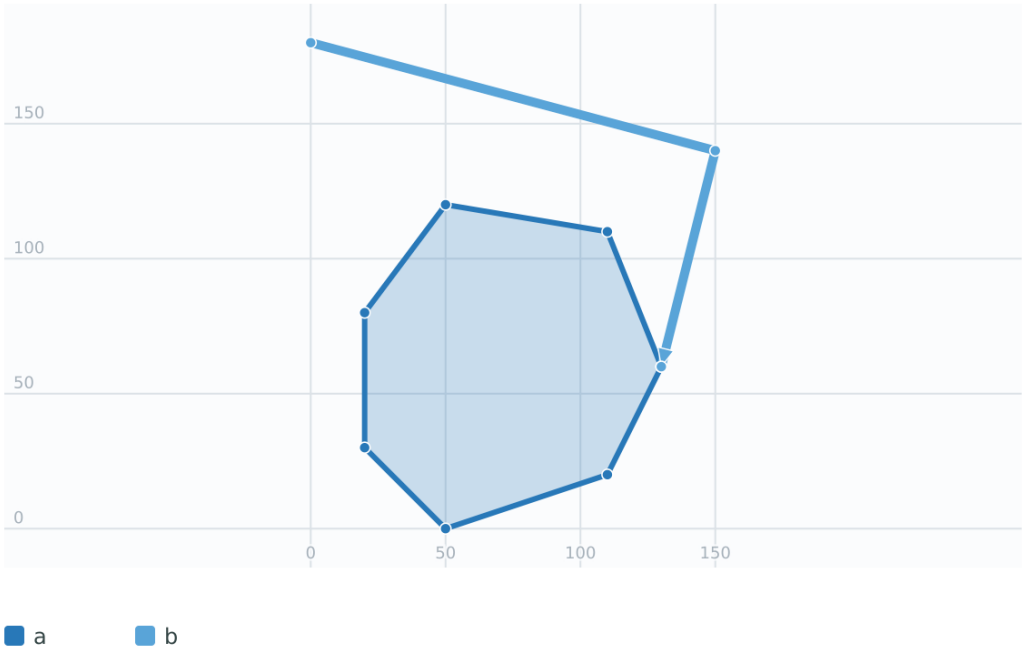
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((80 10,50 40,50 90,80 130,140 120,160 70,140 30,80 10))'::geometry AS a,
  'LINESTRING (30 190,180 150,160 70)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two LineStrings touching at their endpoints.

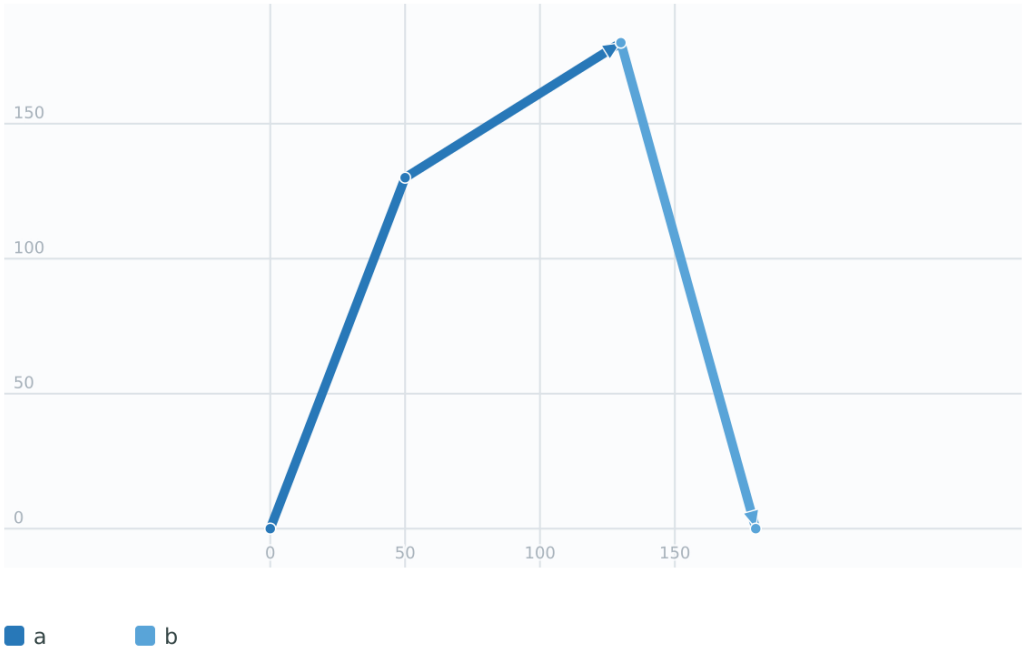
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'LINESTRING (10 10,60 140,140 190)'::geometry AS a,
  'LINESTRING (140 190,190 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



Two LineStrings touching where an endpoint meets the other line.

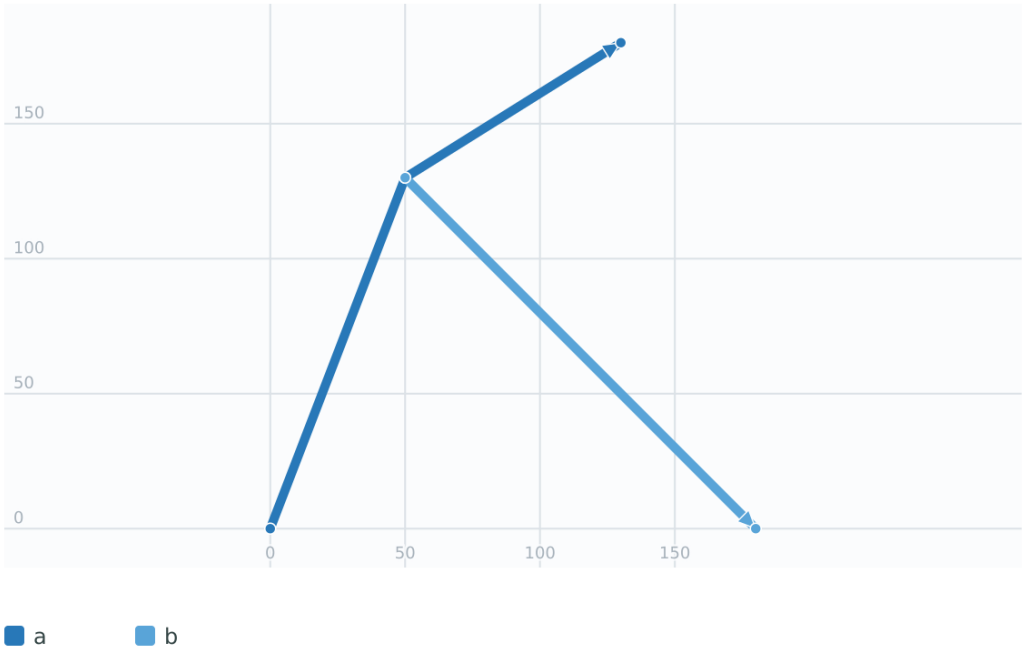
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'LINESTRING (10 10,60 140,140 190)'::geometry AS a,
  'LINESTRING (60 140,190 10)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



A Point touching a Polygon boundary.

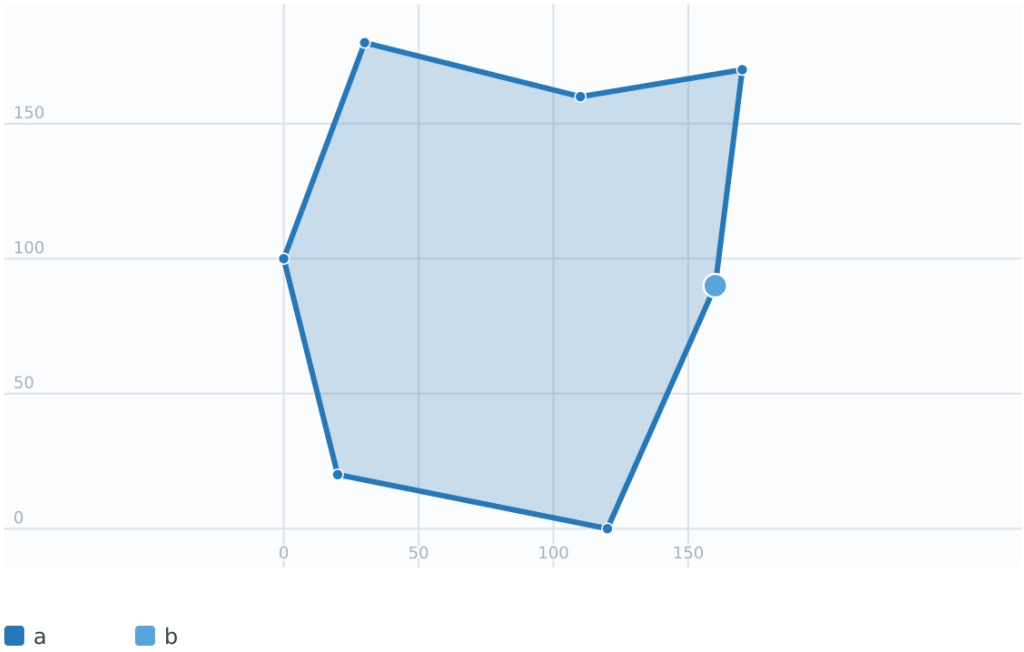
Code

```
SELECT ST_Touches(a, b)
FROM (SELECT
  'POLYGON ((30 30,10 110,40 190,120 170,180 180,170 100,130 10,30 30))'::geometry AS a,
  'POINT (170 100)'::geometry AS b
) AS example;
```

Output

```
t
```

Figure



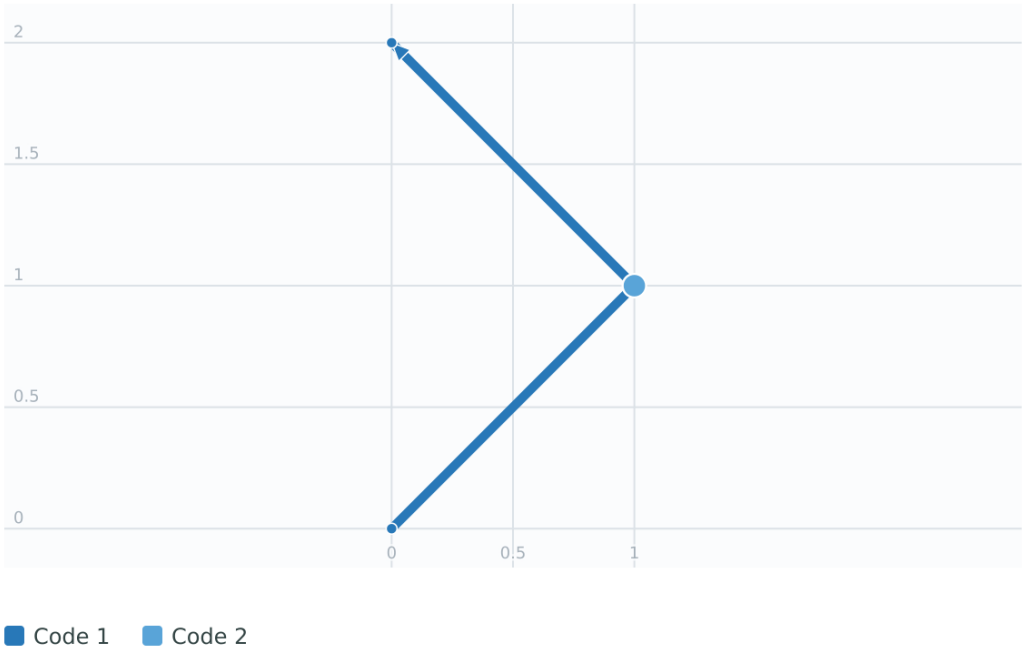
Code

```
SELECT ST_Touches('LINESTRING(0 0,1 1,0 2)'::geometry, 'POINT(1 1)'::geometry);
```

Output

```
f
```

Figure



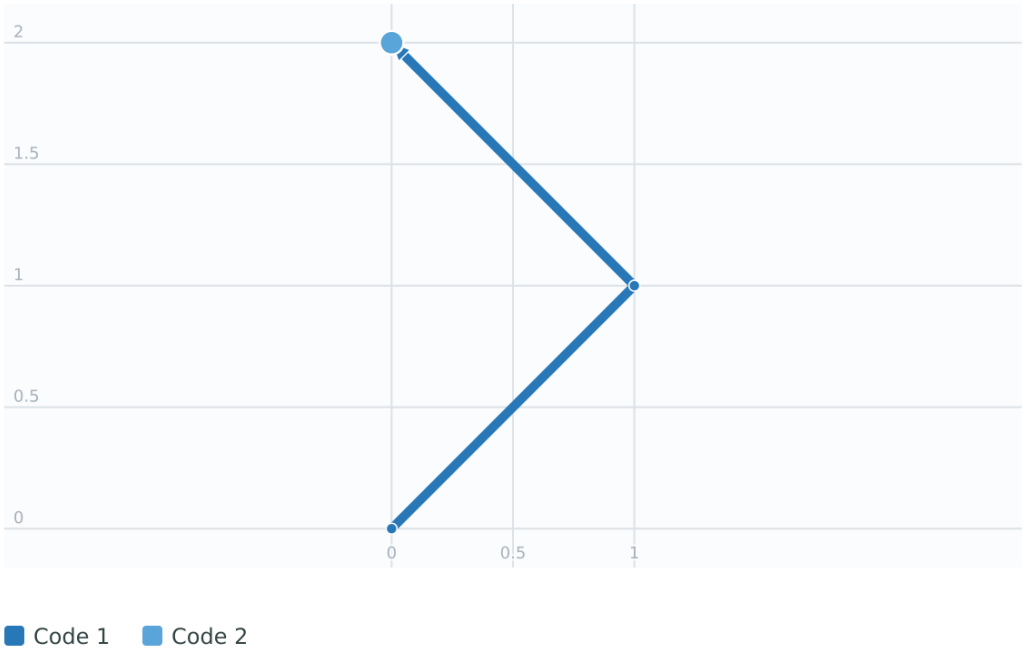
Code

```
SELECT ST_Touches('LINESTRING(0 0,1 1,0 2)'::geometry, 'POINT(0 2)'::geometry);
```

Output

```
t
```

Figure



7.11.1.16 ST_Within

ST_Within — ☒☒ 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。

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

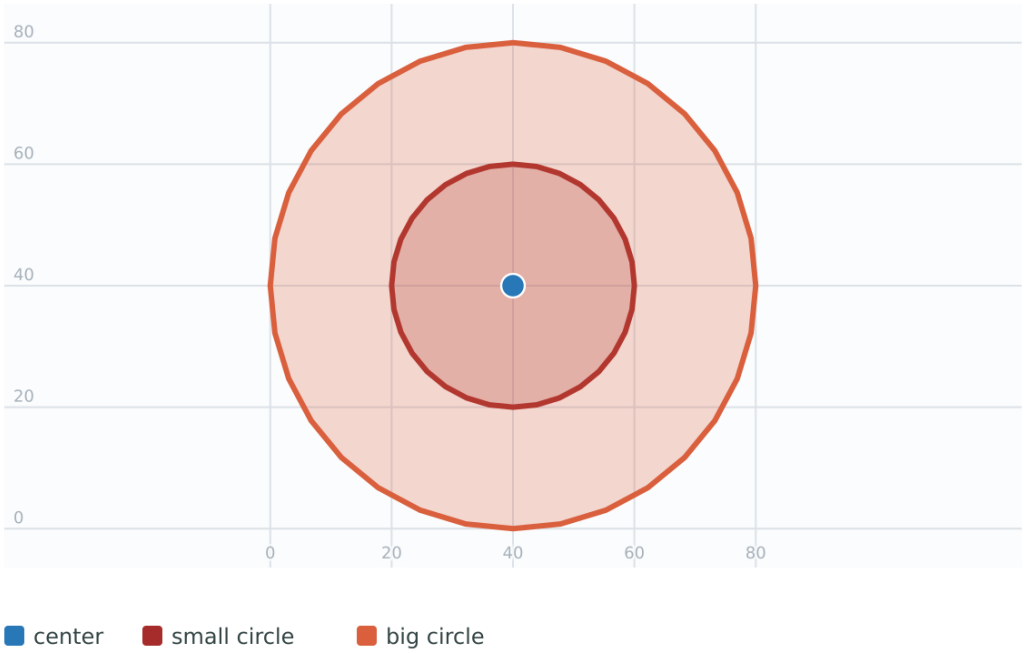
示例

This example defines two concentric circles once in a CTE so the same inputs can be rendered and then reused in related predicate checks.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 20) AS smallc,
         ST_Buffer(center, 40) AS bigc
  FROM (SELECT 'POINT(50 50)'::geometry AS center) AS p
)
SELECT smallc AS small_circle,
       bigc AS big_circle
FROM circles;
```

Figure



Using the same circles, `ST_Within` is reflexive on each circle, false from the larger circle into the smaller one, and true for `ST_Union` only when the union does not extend beyond the larger circle.

Code

```
WITH circles AS (
  SELECT ST_Buffer(center, 20) AS smallc,
         ST_Buffer(center, 40) AS bigc
  FROM (SELECT 'POINT(50 50)'::geometry AS center) AS p
)
SELECT
  ST_Within(smallc, smallc) AS smallinsmall,
  ST_Within(smallc, bigc) AS smallinbig,
  ST_Within(bigc, smallc) AS biginsmall,
  ST_Within(ST_Union(smallc, bigc), bigc) AS unioninbig,
  ST_Within(bigc, ST_Union(smallc, bigc)) AS biginunion,
  ST_Equals(bigc, ST_Union(smallc, bigc)) AS bigisunion
FROM circles;
```

Output

```
t | t | f | t | t | t
```

Figure



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

Compare the same transformed point and line in both 3D and 2D. The 3D test fails because the Z difference matters even though the 2D footprints are close enough.

Code

```
WITH sample AS (
  SELECT ST_Transform('SRID=4326;POINT(-72.1235 42.3521 4)'::geometry, 2163) AS input_point ↩
    ,
         ST_Transform(
           'SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)'::geometry,
           2163
         ) AS input_line
)
SELECT ST_3DDWithin(input_point, input_line, 126.8) AS within_dist_3d,
       ST_DWithin(input_point, input_line, 126.8) AS within_dist_2d,
       input_point AS input_point,
       input_line AS input_line
FROM sample;
```

Output

```
-[ RECORD 1 ↩
    ]--+----------------------------------------------------------------------------------------------------------------------------------------------------------------

within_dist_3d | f
within_dist_2d | t
input_point    | POINT Z (-11774128.9 5237160.2 4)
input_line     | LINESTRING Z (-11774339.5 5243556.8 15,-11773706.7 5215256.3 20)
```

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 example compares distance-within predicates in 2D and 3D. The point is distance-within the line in 3D, but the full 3D line is not within the same distance of the point.

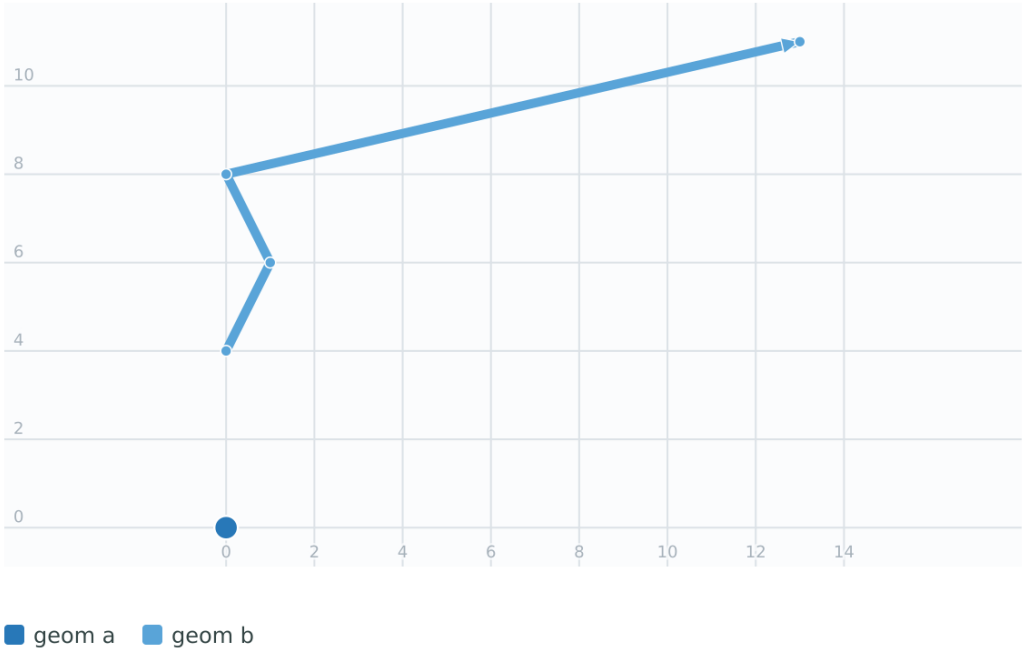
Code

```
WITH data AS (
  SELECT
    'POINT(1 1 2)'::geometry AS geom_a,
    'LINESTRING(1 5 2,2 7 20,1 9 100,14 12 3)'::geometry AS geom_b
)
SELECT
  ST_3DDFullyWithin(geom_a, geom_b, 10) AS d3dfullywithin10,
  ST_3DDWithin(geom_a, geom_b, 10) AS d3dwithin10,
  ST_DFullyWithin(geom_a, geom_b, 20) AS d2dfullywithin20,
  ST_3DDFullyWithin(geom_a, geom_b, 20) AS d3dfullywithin20
FROM data;
```

Output

```
d3dfullywithin10 | d3dwithin10 | d2dfullywithin20 | d3dfullywithin20
-----------------+-------------+------------------+-----------------
 f               | t           | t                | f
```

Figure



相关信息

ST_3DDWithin, ST_DWithin, ST_DFullyWithin, ST_3DMaxDistance

7.11.2.3 ST_DFullyWithin

ST_DFullyWithin — ⌧⌧一个几何⌧形是否完全位于⌧一个几何⌧形的指定距离内

Synopsis

boolean **ST_DFullyWithin**(geometry g1, geometry g2, double precision distance);

描述

当 `g2` 完全位于 `g1` 指定的 `distance` 距离范⌧内⌧，返回 TRUE。从⌧⌧上看，当 `g2` 完全包含在 `g1` ⌧展 `distance` 距离的⌧冲区内⌧，返回 TRUE。



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

更改：3.5.0 版本：⌧函数的⌧⌧⌧在使用⌧冲区内包含⌧⌧，而不是 ST_MaxDistance 算法。⌧果可能与之前的版本不同，但⌧⌧更接近用⌧的⌧期。

示例

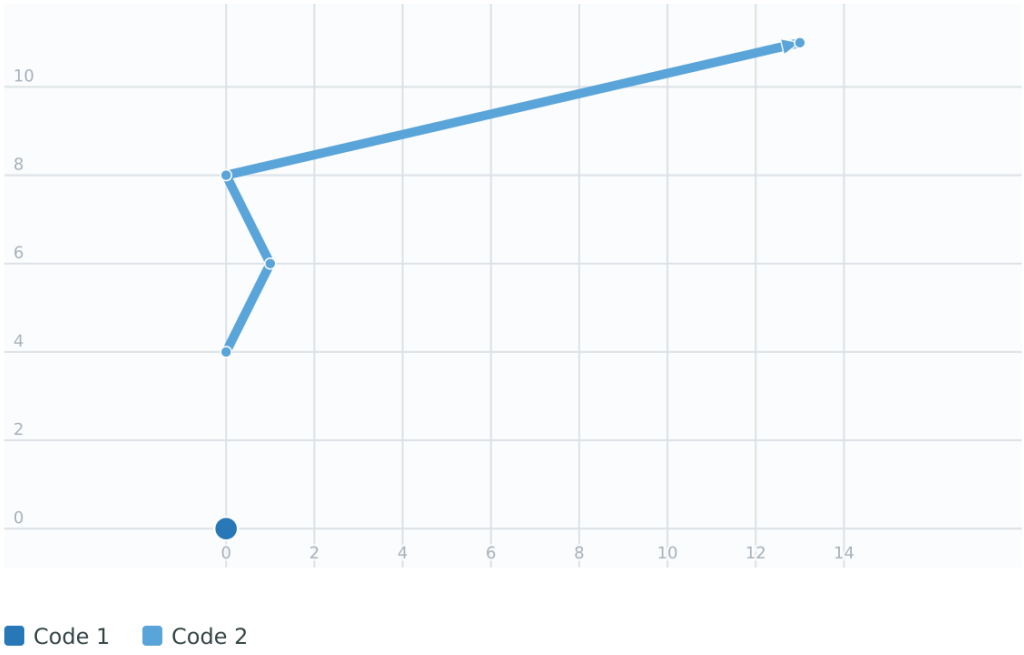
Code

```
SELECT
    ST_DFullyWithin(geom_a, geom_b, 10) AS DFullyWithin10,
    ST_DWithin(geom_a, geom_b, 10) AS DWithin10,
    ST_DFullyWithin(geom_a, geom_b, 20) AS DFullyWithin20
FROM (VALUES
    ('POINT(1 1)'::geometry, 'LINESTRING(1 5,2 7,1 9,14 12)'::geometry)
    ) AS v(geom_a, geom_b)
```

Output

```
dfullywithin10 | dwithin10 | dfullywithin20
---------------+-----------+---------------
 f             | t         | t
```

Figure



相关信息

ST_MaxDistance, ST_DWithin, ST_3DDWithin, ST_3DDFullyWithin

7.11.2.4 ST_DWithin

ST_DWithin — ☒☒☒个几何☒形是否在☒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☒形在☒定距离内，☒返回 true

☒于几何☒形：距离以几何☒形的空☒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）。

☒于地理：☒位☒米，距离☒量默☒使用 `use_spheroid = true`（使用☒球体）。☒了更快的☒算，可以使用 `use_spheroid = false`，在球面上☒行☒量。

Note
将 ST_3DDWithin用于 3D 几何☒形。

Note
此函数⊠用包括利用几何上可用的任何索引的⊠界框比⊠。

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⊠功能：2.1.0 版本提升了地理（Geography）⊠型的速度。有关⊠⊠信息，⊠参⊠Making Geography faster。

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

This self-contained geometry example keeps the compared shapes visible while also returning the boolean results.

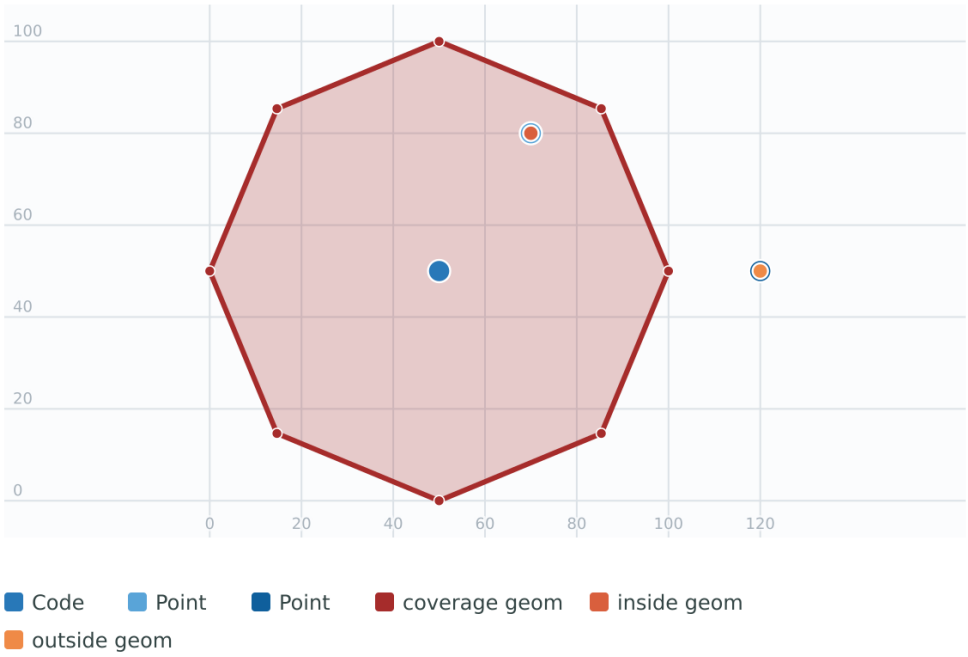
Code

```
WITH sample AS (
  SELECT ST_Buffer('POINT(100 90)'::geometry, 50, 'quad_segs=2') AS coverage,
         ST_Point(120, 120) AS inside_pt,
         ST_Point(170, 90) AS outside_pt
)
SELECT ST_DWithin(coverage, inside_pt, 0) AS inside_now,
       ST_DWithin(coverage, outside_pt, 10) AS outside_within_10,
       coverage AS coverage_geom,
       inside_pt AS inside_geom,
       outside_pt AS outside_geom
FROM sample;
```

Output

```
-[ RECORD 1 ]-----
inside_now        | t
outside_within_10 | f
coverage_geom     | POLYGON((150 90,135 55,100 40,65 55,50 90,65 125,100 140,135 125,150 90))
inside_geom       | POINT(120 120)
outside_geom      | POINT(170 90)
```

Figure



Find the nearest hospital to each school. That is within 3000 units of the school. We do an ST_DWithin search to utilize indexes to limit our search list. That the non-indexable ST_Distance needs to process. If the units of the spatial reference is meters then units would be meters.

Code

```
SELECT DISTINCT ON (s.gid) s.gid, s.school_name, s.geom, h.hospital_name
  FROM schools s
    LEFT JOIN hospitals h ON ST_DWithin(s.geom, h.geom, 3000)
  ORDER BY s.gid, ST_Distance(s.geom, h.geom);
```

The schools with no close hospitals. Find all schools with no hospital within 3000 units away from the school. Units are in units of spatial reference (for example meters, feet, or degrees).

Code

```
SELECT s.gid, s.school_name
  FROM schools s
    LEFT JOIN hospitals h ON ST_DWithin(s.geom, h.geom, 3000)
  WHERE h.gid IS NULL;
```

Find locations more than 3000 units away from their nearest hospital. Use `NOT EXISTS` so that a location is returned only when no nearby hospital exists.

Code

```
SELECT l.location_name
  FROM locations l
  WHERE NOT EXISTS (
    SELECT 1
    FROM hospitals h
    WHERE ST_DWithin(l.geom, h.geom, 3000)
  );
```

Find broadcasting towers that a receiver with limited range can receive. Data is geometry in Spherical Mercator (SRID=3857), ranges are approximate.

Code

```
CREATE INDEX ON broadcasting_towers USING gist (geom);
CREATE INDEX ON broadcasting_towers USING gist (ST_Expand(geom, sending_range));
```

Query towers that a 4-kilometer receiver in Minsk Hackerspace can get. Note: two conditions are required because the shorter `LEAST(b.sending_range, 4000)` will not use an index.

Code

```
SELECT b.tower_id, b.geom
  FROM broadcasting_towers b
  WHERE ST_DWithin(b.geom, 'SRID=3857;POINT(3072163.4 7159374.1)', 4000)
    AND ST_DWithin(b.geom, 'SRID=3857;POINT(3072163.4 7159374.1)', b.sending_range);
```

相关信息

ST_Distance, ST_3DDWithin

7.11.2.5 ST_PointInsideCircle

ST_PointInsideCircle — 区区点几何区形是否位于由区心和半径定区的区内

Synopsis

boolean **ST_PointInsideCircle**(geometry a_point, float center_x, float center_y, float radius);

描述

如果几何区形是一个点并且位于以 `center_x`、`center_y` 和半径 `radius` 区区心的区内，区返回 true。



Warning
不使用空区索引。区改用 ST_DWithin。

可用性：1.2

更改：2.2.0 在之前的版本中，区称区 ST_Point_Inside_Circle

示例

Code

```
SELECT ST_PointInsideCircle(ST_Point(1, 2), 0.5, 2, 3);
```

Output

```
t
```

Figure



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

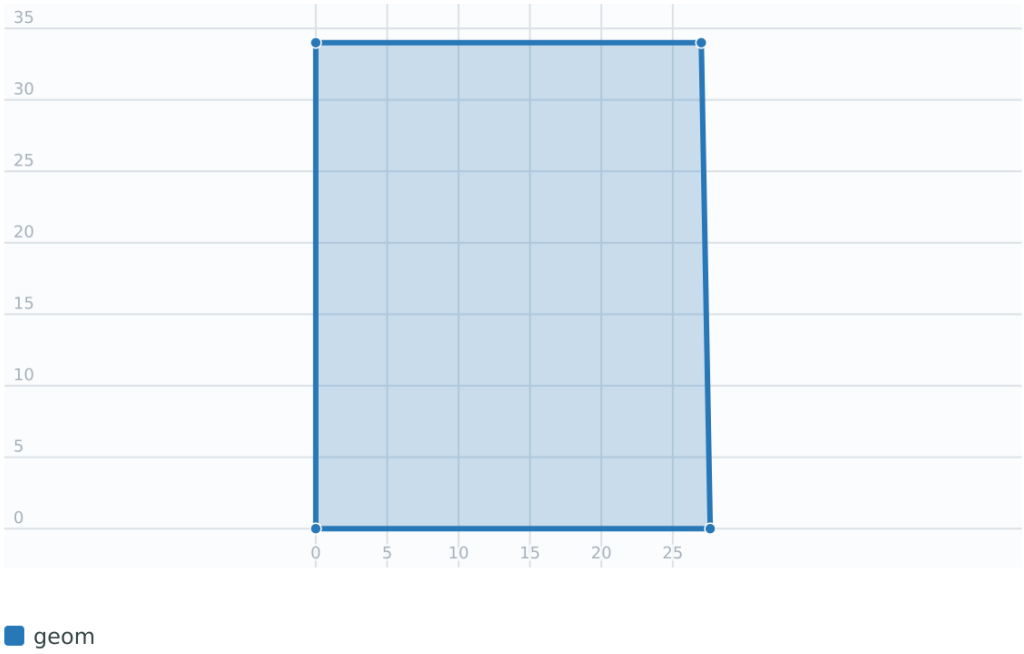
Code

```
SELECT ST_Area(geom) AS sqft,
       ST_Area(geom) * 0.3048 ^ 2 AS sqm
FROM (
  SELECT 'SRID=2249;POLYGON((743238 2967416,743238 2967450,
          743265 2967450,743265.625 2967416,743238 2967416))'::geometry AS geom
) AS subquery;
```

Output

```
sqft   |     sqm
--------+-------------
928.625 | 86.27208552
```

Figure



Return area square feet and transform to Massachusetts state plane meters (EPSG:26986) to get square meters. Note this is in square feet because 2249 is Massachusetts State Plane Feet and transformed area is in square meters since EPSG:26986 is state plane Massachusetts meters.

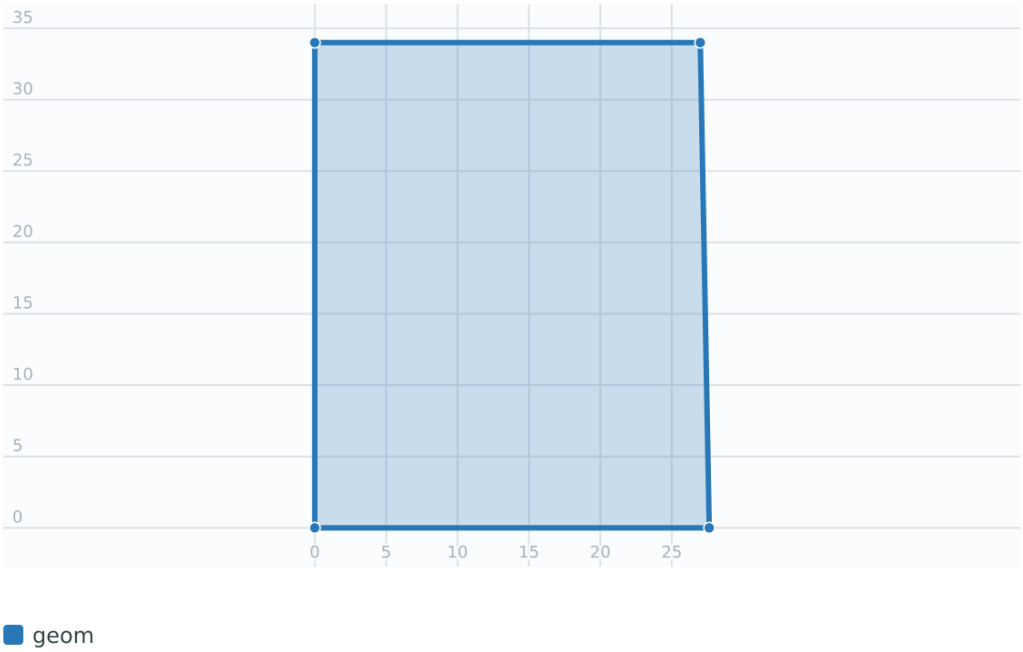
Code

```
SELECT ST_Area(geom) AS sqft,
       round(ST_Area(ST_Transform(geom, 26986))::numeric, 6) AS sqm
FROM (
  SELECT 'SRID=2249;POLYGON((743238 2967416,743238 2967450,
          743265 2967450,743265.625 2967416,743238 2967416))'::geometry AS geom
) AS subquery;
```

Output

```
sqft   |    sqm
--------+-----------
928.625 | 86.272431
```

Figure



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。

Code

```
SELECT round((ST_Area(geog) / 0.3048 ^ 2)::numeric, 2) AS sqft_spheroid,
       round((ST_Area(geog, false) / 0.3048 ^ 2)::numeric, 2) AS sqft_sphere,
       round(ST_Area(geog)::numeric, 2) AS sqm_spheroid
FROM (
  SELECT ST_Transform(
           'SRID=2249;POLYGON((743238 2967416,743238 2967450,
           743265 2967450,743265.625 2967416,743238 2967416))'::geometry,
           4326
         )::geography AS geog
) AS subquery;
```

Output

```
sqft_spheroid | sqft_sphere | sqm_spheroid
---------------+-------------+--------------
        928.68 |      926.61 |        86.28
```

如果您的数据已⍰是地理数据：

Code

```
SELECT ST_Area(geog) / 0.3048 ^ 2 AS sqft,
                ST_Area(the_geog) AS sqm
        FROM somegeogtable;
```

相关信息

CG_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。

For offsetting lines relative to their direction, use ST_OffsetCurve. For projecting a point by distance and bearing, use ST_Project.

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）

Code

```
SELECT degrees(ST_Azimuth(ST_Point(25, 45), ST_Point(75, 100))) AS degA_B,
       degrees(ST_Azimuth(ST_Point(75, 100), ST_Point(25, 45) )) AS degB_A;
```

Output

```
dega_b       |     degb_a
-----------------+-----------------
42.27368900609371 | 222.27368900609372
```

Compass direction labels can be computed from the azimuth.

Code

```
WITH directions(deg) AS (VALUES
  (0), (45), (90), (135), (180), (225), (270), (315)
),
bearings AS (
  SELECT deg,
         degrees(ST_Azimuth(ST_Point(0, 0),
                            ST_MakePoint(sin(radians(deg)),
                                         cos(radians(deg))))) AS azimuth
  FROM directions
)
SELECT deg,
       round(azimuth::numeric, 1) AS azimuth,
       (ARRAY['N', 'NE', 'E', 'SE', 'S', 'SW', 'W', 'NW', 'N'])
         [floor((azimuth + 22.5) / 45)::int + 1] AS direction
FROM bearings
ORDER BY deg;
```

Output

```
deg | azimuth | direction
-----+---------+-----------
   0 |     0.0 | N
  45 |    45.0 | NE
  90 |    90.0 | E
 135 |   135.0 | SE
 180 |   180.0 | S
 225 |   225.0 | SW
 270 |   270.0 | W
 315 |   315.0 | NW
```

Azimuth from Point(25, 45) to Point(75, 100), shown against the positive Y axis (north).

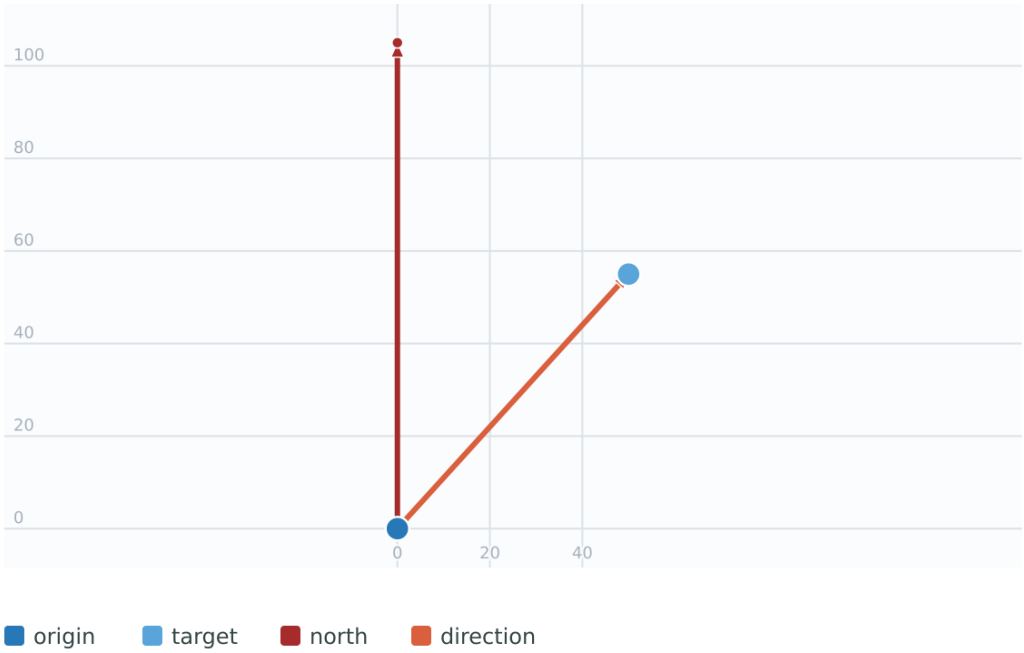
Code

```
WITH points AS (
  SELECT 'POINT(25 45)'::geometry AS origin,
         'POINT(75 100)'::geometry AS target
)
SELECT round(degrees(ST_Azimuth(origin, target))::numeric, 6) AS degrees,
       ST_MakeLine(origin, ST_Translate(origin, 0, 105)) AS north,
       ST_MakeLine(origin, target) AS direction
FROM points;
```

Output

```
42.273689 | LINESTRING(25 45,25 150) | LINESTRING(25 45,75 100)
```

Figure



Reverse azimuth from Point(75, 100) to Point(25, 45).

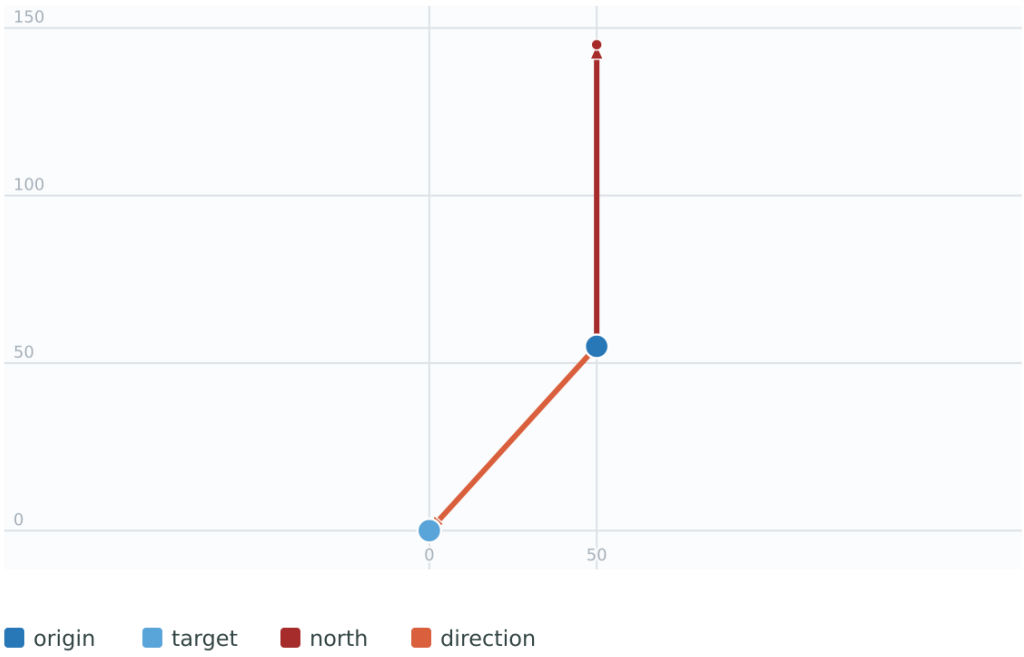
Code

```
WITH points AS (
  SELECT 'POINT(75 100)'::geometry AS origin,
         'POINT(25 45)'::geometry AS target
)
SELECT round(degrees(ST_Azimuth(origin, target))::numeric, 6) AS degrees,
       ST_MakeLine(origin, ST_Translate(origin, 0, 90)) AS north,
       ST_MakeLine(origin, target) AS direction
FROM points;
```

Output

```
222.273689 | LINESTRING(75 100,75 190) | LINESTRING(75 100,25 45)
```

Figure



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

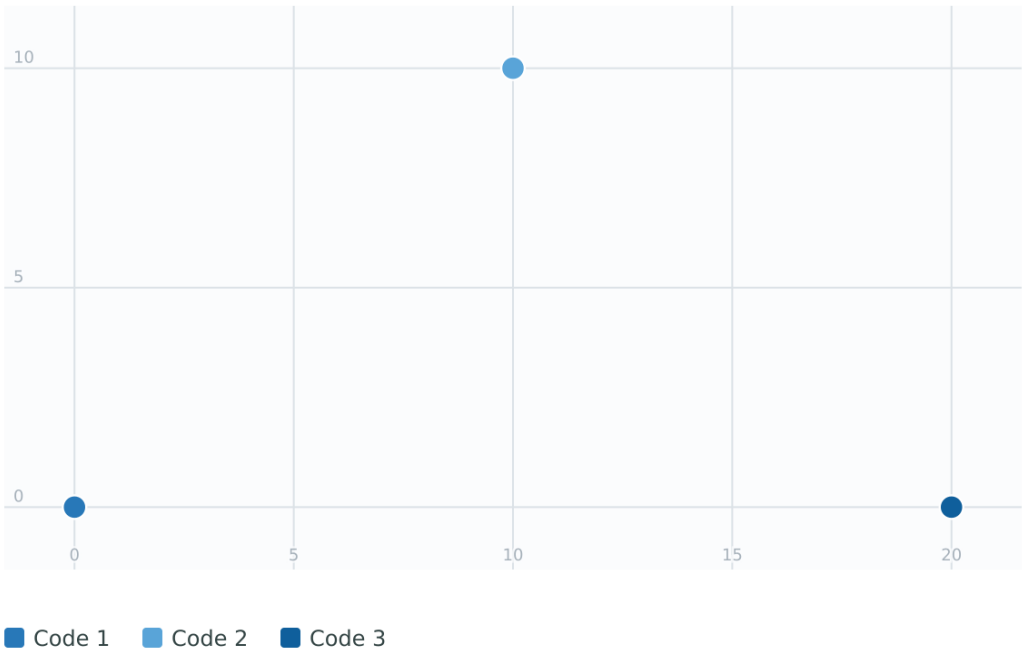
示例

三点之⌧的角度

Code

```
SELECT degrees(ST_Angle('POINT(0 0)', 'POINT(10 10)', 'POINT(20 0)'));
```

Figure

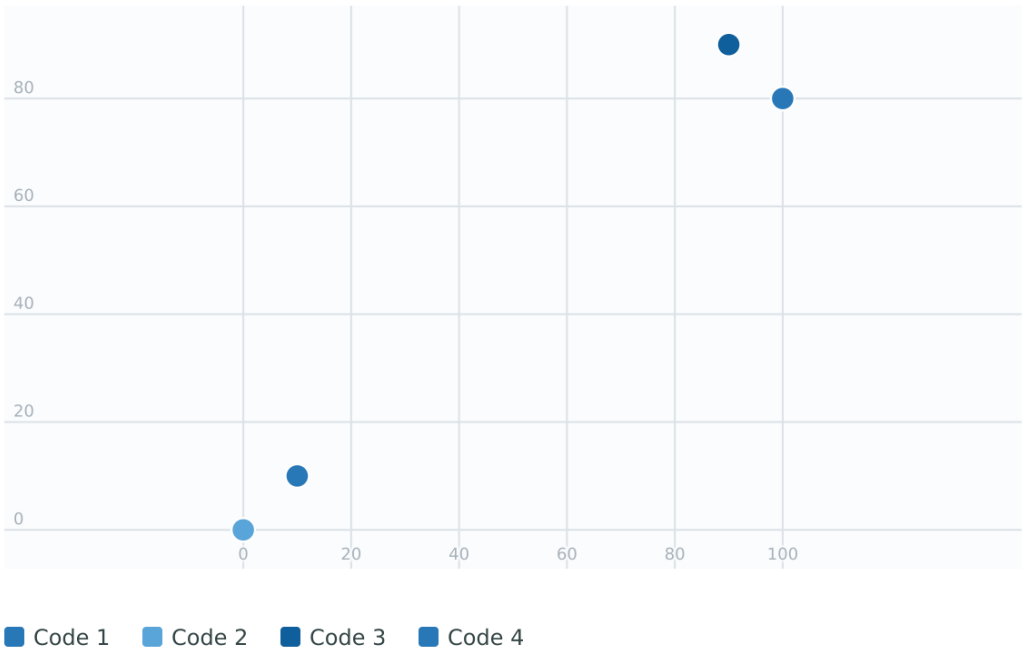


由四个点定⊠的向量之⊠的角度

Code

```
SELECT degrees(ST_Angle('POINT (10 10)', 'POINT (0 0)', 'POINT(90 90)', 'POINT (100 80)'));
```

Figure

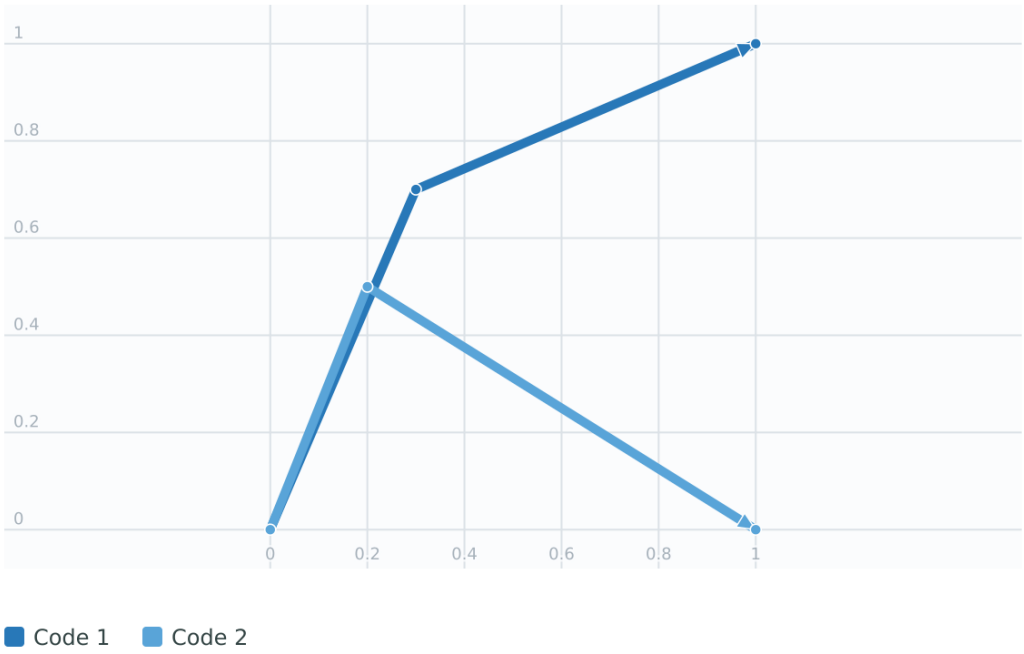


由⊠的起点和⊠点定⊠的向量之⊠的角度

Code

```
SELECT degrees(ST_Angle('LINESTRING(0 0,0.3 0.7,1 1)', 'LINESTRING(0 0,0.2 0.5,1 0)'));
```

Figure



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⊠点。⊠是几何⊠形之⊠最短⊠的第一个点（由 ST_ShortestLine⊠算）。



Note
如果您有 3D 几何⊠形，您可能更喜⊠使用 ST_3DClosestPoint。

增⊠：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⊠上的点。

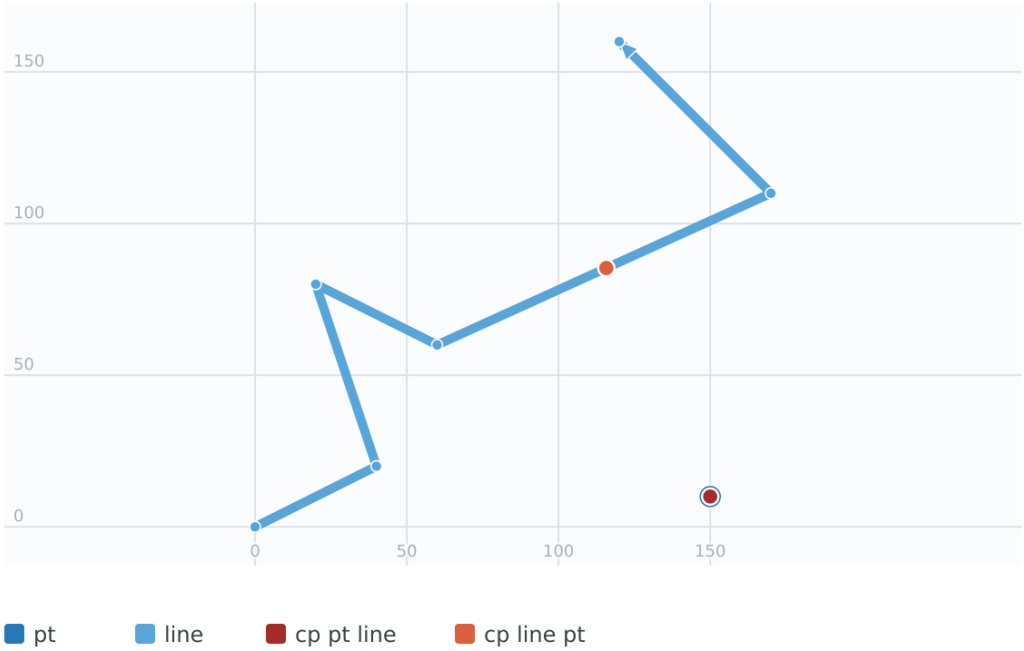
Code

```
SELECT ST_ClosestPoint(pt, line) AS cp_pt_line, ST_ClosestPoint(line, pt) AS cp_line_pt
    FROM (SELECT 'POINT (160 40)'::geometry AS pt,
                 'LINESTRING (10 30,50 50,30 110,70 90,180 140,130 190)'::geometry AS line
    ) AS t;
```

Output

```
cp_pt_line   |               cp_line_pt
---------------+----------------------------------------
 POINT(160 40)  | POINT(125.75342465753425 115.34246575342466)
```

Figure

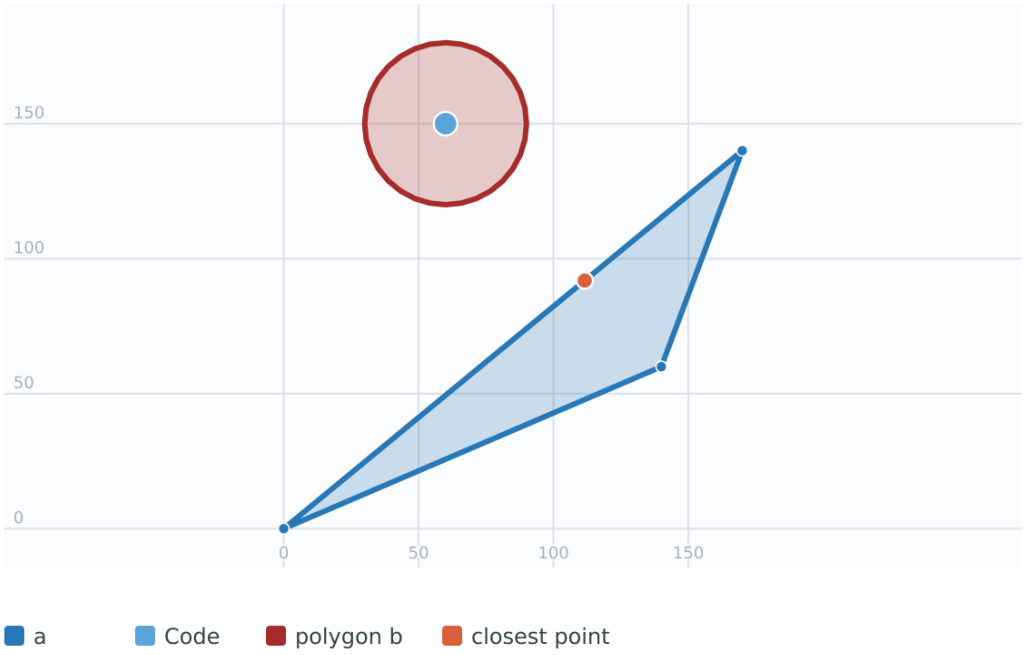


Find the closest point on polygon A to polygon B.

Code

```
WITH source AS (
  SELECT 'POLYGON ((190 150,20 10,160 70,190 150))'::geometry AS a,
         ST_Buffer('POINT(80 160)'::geometry, 30) AS b
)
SELECT b AS polygon_b,
       ST_ClosestPoint(a, b) AS closest_point
FROM source;
```

Figure



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

Compare the 3D closest point to the 2D closest point for a LineString and a Point.

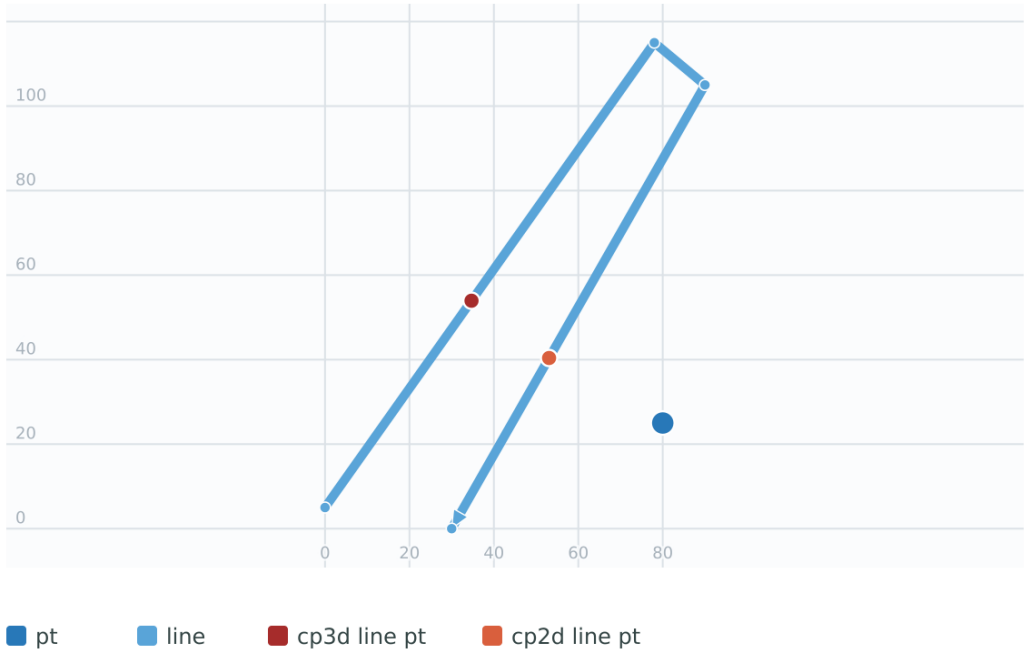
Code

```
WITH input AS (
  SELECT 'POINT(100 100 30)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 1000)'::geometry AS line
)
SELECT ST_3DClosestPoint(line, pt) AS cp3d_line_pt,
       ST_ClosestPoint(line, pt) AS cp2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+------------------------------------------------------
cp3d_line_pt   | POINT(54.69937988676193 128.93502291722837 11.547586950660556)
cp2d_line_pt   | POINT(73.07692307692307 115.38461538461539)
```

Figure



Compare the 3D closest point to the 2D closest point for a LineString and a MultiPoint.

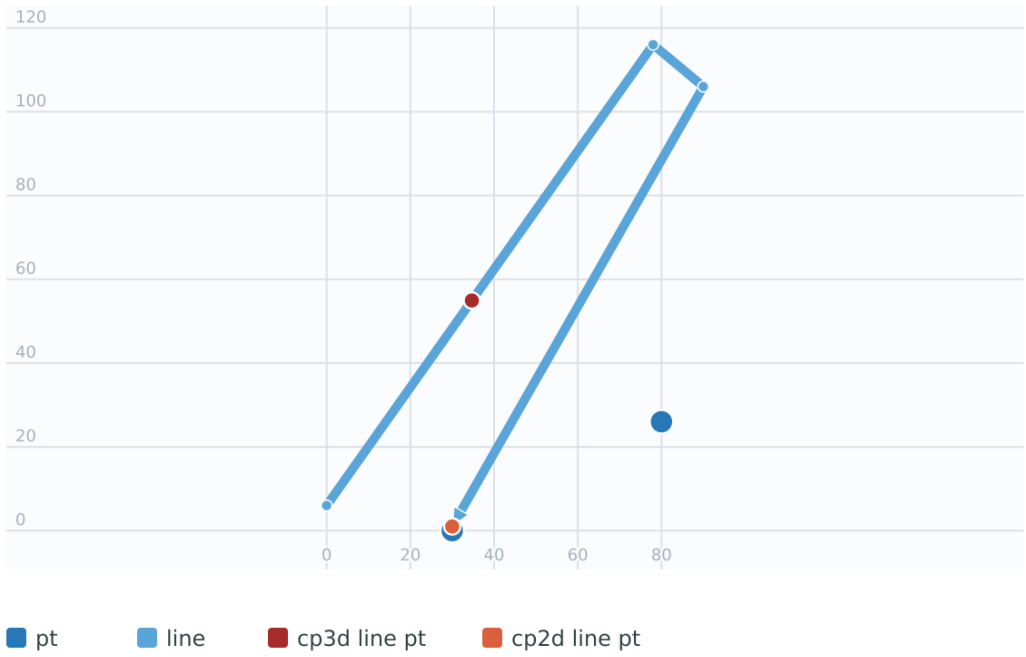
Code

```
WITH input AS (
  SELECT 'MULTIPOINT(100 100 30,50 74 1000)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 900)'::geometry AS line
)
SELECT ST_3DClosestPoint(line, pt) AS cp3d_line_pt,
       ST_ClosestPoint(line, pt) AS cp2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+------------------------------------------------------
cp3d_line_pt   | POINT(54.69937988676193 128.93502291722837 11.547586950660556)
cp2d_line_pt   | POINT(50 75)
```

Figure



Compare the 3D closest point to the 2D closest point for a Polygon and a MultiLineString.

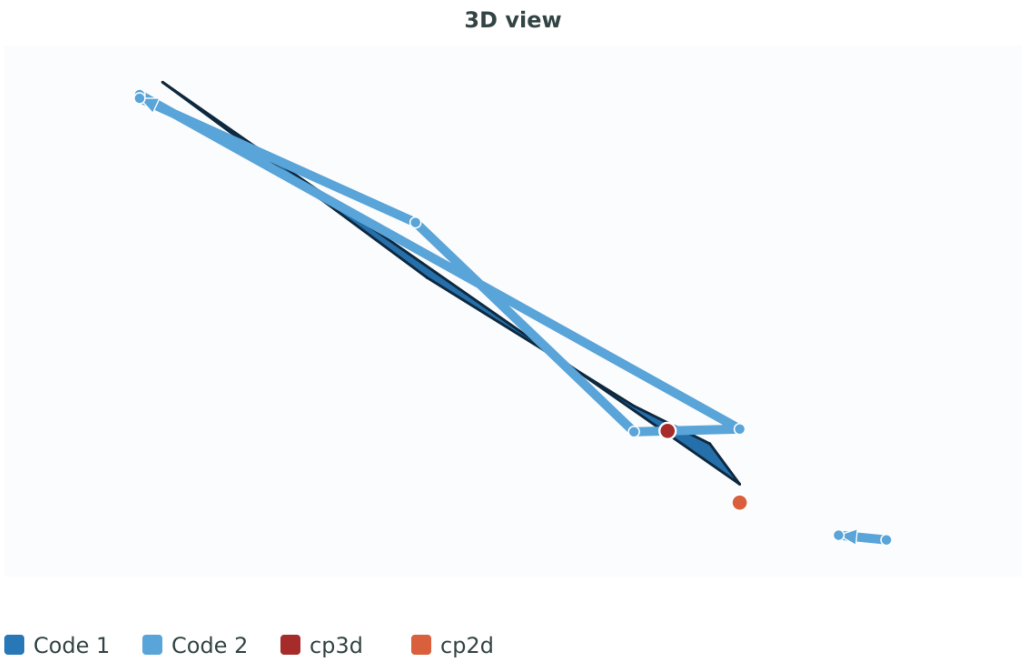
Code

```
WITH input AS (
  SELECT ST_GeomFromEWKT(
           'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))'
         ) AS poly,
         ST_GeomFromEWKT(
           'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),
                            (1 10 2,5 20 1))'
         ) AS mline
)
SELECT ST_3DClosestPoint(poly, mline) AS cp3d,
       ST_ClosestPoint(poly, mline) AS cp2d
FROM input;
```

Output

```
-[ RECORD 1 ]--+-------------------------------------
cp3d           | POINT(40.45454545454545 53.63636363636364 5)
cp2d           | POINT(20 40)
```

Figure



相关信息

ST_AsEWKT, ST_ClosestPoint, ST_3DDistance, ST_3DShortestLine

7.12.6 ST_Distance

ST_Distance — 返回⊠个几何或地理⊠之⊠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⊠功能：2.1.0 版本提升了地理（Geography）⊠型的速度。有关⊠⊠信息，⊠参⊠Making Geography faster。

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。

Code

```
SELECT ST_Distance(
'SRID=4326;POINT(-72.1235 42.3521)'::geometry,
'SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry );
```

Output

```
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☒☒☒。

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 3857),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 3857) );
```

Output

```
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）

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 3857),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 3857)
        ) * cosd(42.3521);
```

Output

```
123.742351254151
```

几何示例 - ☒位☒米（SRID：26986 ☒☒☒塞州平面米）（☒☒☒塞州最准确）

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 26986),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 26986) );
```

Output

```
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）

Code

```
SELECT ST_Distance(
ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 2163),
ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geometry, 2163) );
```

Output

```
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。

Code

```
SELECT ST_Distance(gg1, gg2) As spheroid_dist, ST_Distance(gg1, gg2, false) As sphere_dist
FROM (SELECT
    'SRID=4326;POINT(-72.1235 42.3521)'::geography as gg1,
    'SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)'::geography as gg2
    ) As foo  ;
```

Output

```
spheroid_dist   |   sphere_dist
-----------------+-----------------
 123.802076746848 | 123.475736916397
```

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⊠个几何⊠形之⊠的 3 ⊠最小笛卡⊠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。

⊠方法⊠⊠了 SQL/MM ⊠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 that currently no vertical datum support so Z is not transformed and assumed to be same units as final.

Code

```
SELECT ST_3DDistance(
            ST_Transform('SRID=4326;POINT(-72.1235 42.3521 4)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)':: ↩
    geometry, 2163)
            ) As dist_3d,
            ST_Distance(
                ST_Transform('SRID=4326;POINT(-72.1235 42.3521)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45,-72.123 42.1546)':: ↩
    geometry, 2163)
            ) As dist_2d;
```

Output

```
dist_3d      |     dist_2d
------------------+-----------------
 127.295059324629 | 126.66425605671
```

Multilinestring and polygon both 3D and 2D distance. Same example as 3D closest point example.

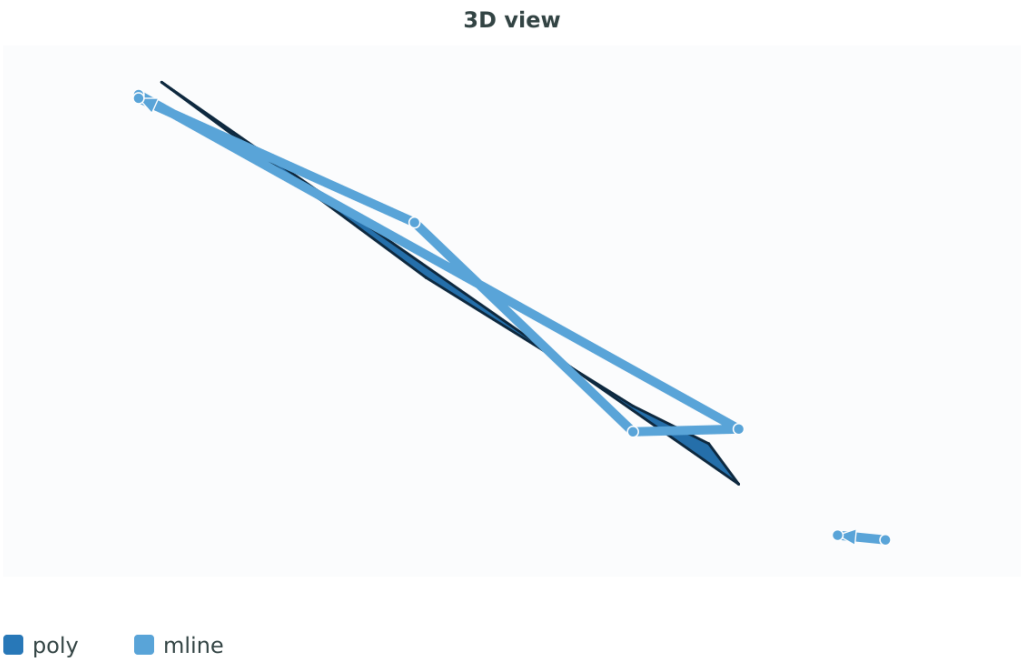
Code

```
SELECT ST_3DDistance(poly, mline) As dist3d,
    ST_Distance(poly, mline) As dist2d
        FROM (SELECT  'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))':: ↩
    geometry as poly,
                'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),(1 10 2,5 ↩
     20 1))'::geometry as mline) as foo;
```

Output

```
dist3d       | dist2d
------------------+--------
                0 |      0
```

Figure



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

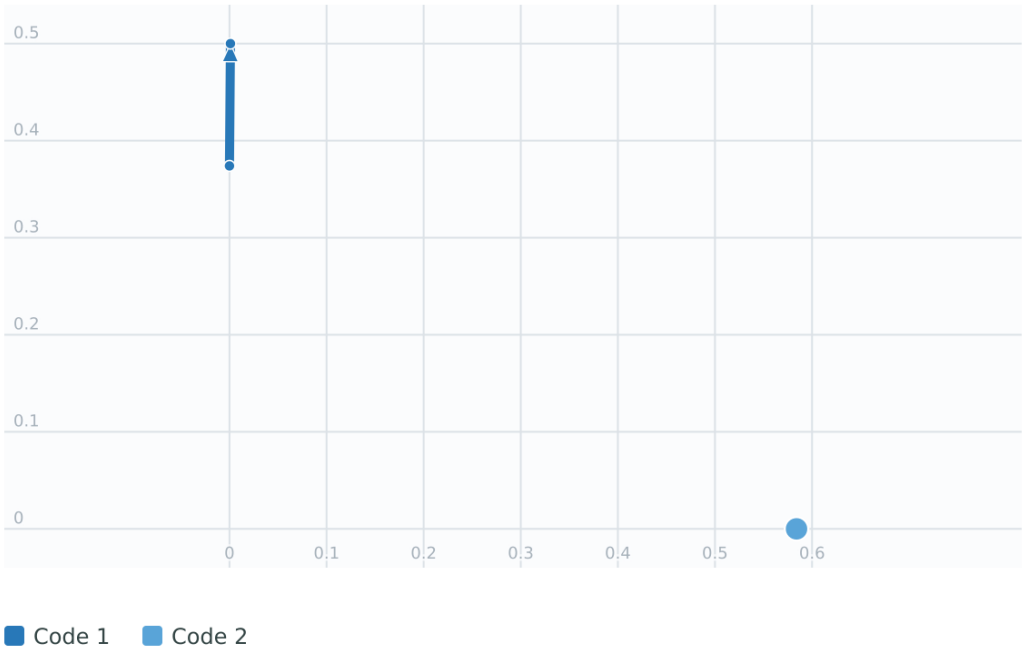
Code

```
WITH sample AS (
    SELECT
        ST_GeomFromText('LINESTRING(-118.584 38.374,-118.583 38.5)', 4326) AS geom,
        ST_GeomFromText('POINT(-118 38)', 4326) AS pt
)
SELECT
    round(ST_DistanceSphere(ST_Centroid(geom), pt)::numeric, 2) AS dist_meters,
    round(ST_Distance(
        ST_Transform(ST_Centroid(geom), 32611),
        ST_Transform(pt, 32611)
    )::numeric, 2) AS dist_utm11_meters,
    round(ST_Distance(ST_Centroid(geom), pt)::numeric, 5) AS dist_degrees,
    round(ST_Distance(
        ST_Transform(geom, 32611),
        ST_Transform(pt, 32611)
    )::numeric, 2) AS min_dist_line_point_meters
FROM sample;
```

Output

```
dist_meters | dist_utm11_meters | dist_degrees | min_dist_line_point_meters
------------+-------------------+--------------+----------------------------
    70424.71 |          70438.00 |      0.72900 |                   65871.18
```

Figure



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_spherc

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

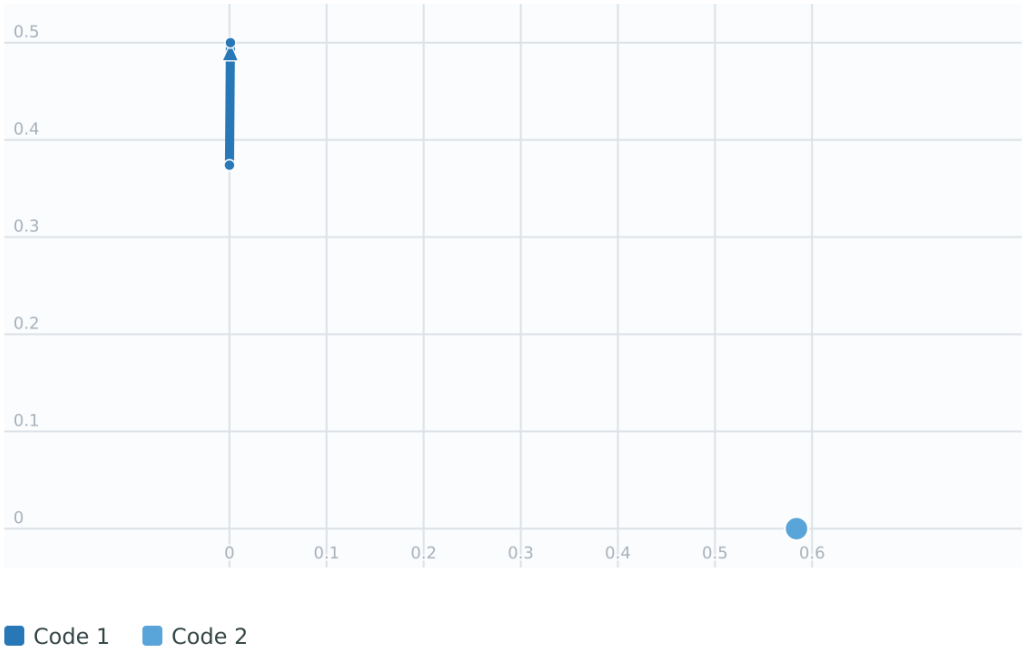
Code

```
WITH input AS (
  SELECT ST_Centroid(
           ST_GeomFromText(
             'LINESTRING(-118.584 38.374,-118.583 38.5)', 4326
           )
         ) AS source,
         ST_GeomFromText('POINT(-118 38)', 4326) AS target
)
SELECT round(
         ST_DistanceSpheroid(
           source,
           target,
           'SPHEROID["WGS 84",6378137,298.257223563]'
         )::numeric,
         2
       ) AS dist_meters_spheroid,
       round(ST_DistanceSphere(source, target)::numeric, 2)
         AS dist_meters_sphere,
       round(
         ST_Distance(
           ST_Transform(source, 32611),
           ST_Transform(target, 32611)
         )::numeric,
         2
       ) AS dist_utm11_meters
FROM input;
```

Output

```
dist_meters_spheroid | dist_meters_sphere | dist_utm11_meters
---------------------+--------------------+-------------------
            70454.92 |           70424.71 |          70438.00
```

Figure



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

Both lines are traversed from left to right. The first line is straight, while the second bows upward through its middle vertex. The point ordering matters, so keeping both inputs in the same direction makes the comparison clearer.

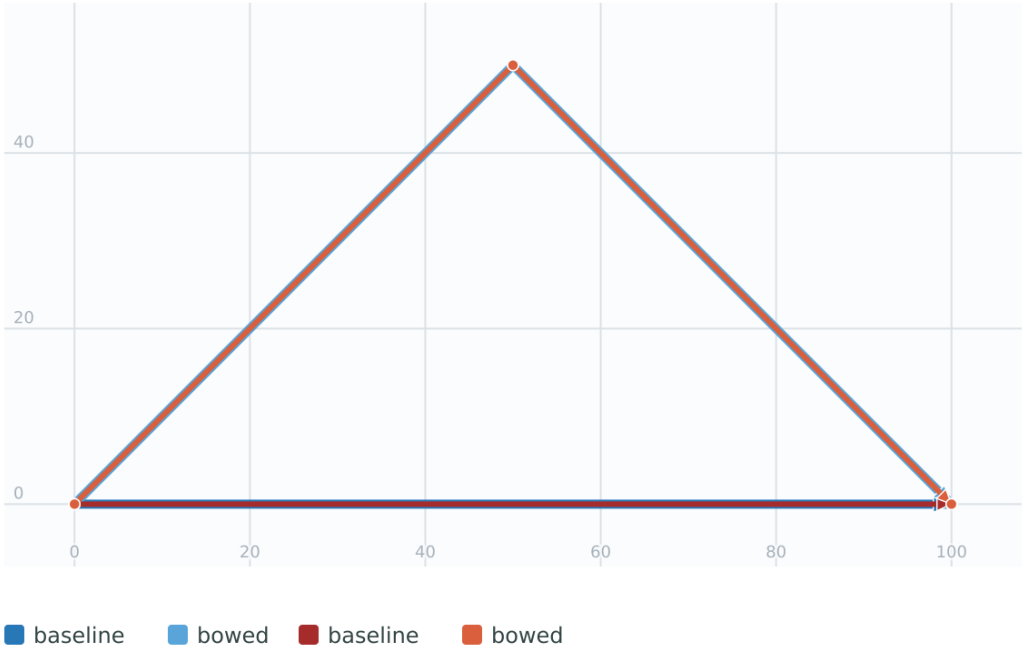
Code

```
WITH lines AS (
  SELECT 'LINESTRING (0 0,100 0)'::geometry AS baseline,
         'LINESTRING (0 0,50 50,100 0)'::geometry AS bowed
)
SELECT baseline AS baseline,
       bowed AS bowed,
       ST_FrechetDistance(baseline, bowed) AS discrete_distance,
       ST_FrechetDistance(baseline, bowed, 0.5) AS densified_distance
FROM lines;
```

Output

```
-[ RECORD 1 ]-----+---------------------------
baseline           | LINESTRING(0 0,100 0)
bowed              | LINESTRING(0 0,50 50,100 0)
discrete_distance  | 70.7106781186548
densified_distance | 50
```

Figure



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

Hausdorff distance and minimum distance between two lines. The Hausdorff witness segment from the original illustration is retained and its length is checked against the computed distance.

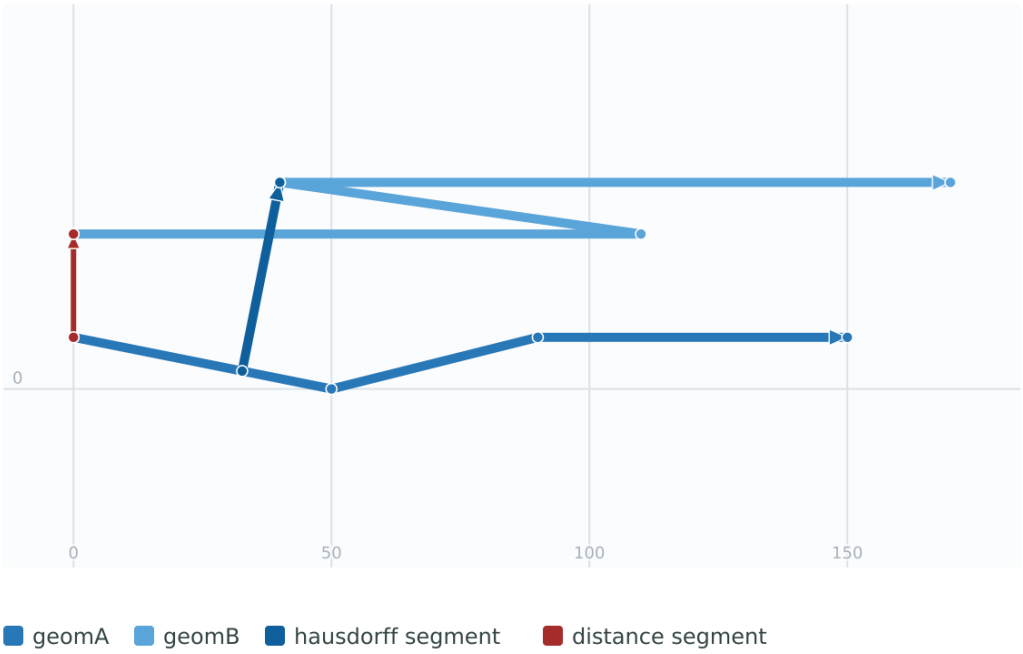
Code

```
WITH lines AS (
  SELECT 'LINESTRING (20 70,70 60,110 70,170 70)'::geometry AS geomA,
         'LINESTRING (20 90,130 90,60 100,190 100)'::geometry AS geomB,
         'LINESTRING (52.69230769230769 63.46153846153846,60 100)'::geometry AS  ↩
    hausdorff_segment
)
SELECT round(ST_HausdorffDistance(geomA, geomB)::numeric, 6) AS hausdorff_distance,
       round(ST_Distance(geomA, geomB)::numeric, 6) AS distance,
       round(ST_Length(hausdorff_segment)::numeric, 6) AS witness_length,
       ST_ShortestLine(geomA, geomB) AS distance_segment
FROM lines;
```

Output

```
37.262066 | 20.000000 | 37.262066 | LINESTRING(20 70,20 90)
```

Figure



示例：致密化的 Hausdorff 距离。

Code

```
SELECT ST_HausdorffDistance(
            'LINESTRING (130 0,0 0,0 150)'::geometry,
            'LINESTRING (10 10,10 150,130 10)'::geometry,
            0.5);
```

Figure



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⊠。

Code

```
SELECT DISTINCT ON (buildings.gid) buildings.gid, parcels.parcel_id
   FROM buildings
       INNER JOIN parcels
       ON ST_Intersects(buildings.geom, parcels.geom)
   ORDER BY buildings.gid, ST_HausdorffDistance(buildings.geom, parcels.geom);
```

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 geography ⊠型：⊠算使用反大地⊠量方法，⊠度⊠位是米。所用⊠球体由 SRID 指定；如果没有 SRID，⊠默⊠使用 WGS84。若 `use_spheroid = false`，⊠⊠算基于球体模型而非⊠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。

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。

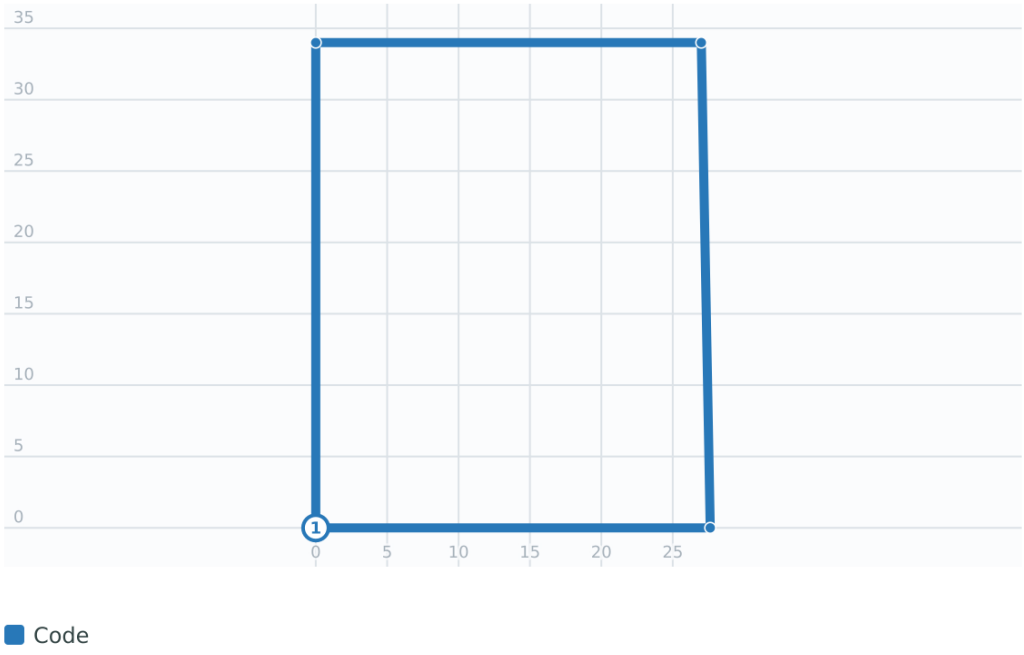
几何示例

返回⊠串的⊠度（以英尺⊠⊠位）。⊠注意，⊠位⊠英尺，因⊠ EPSG:2249 是⊠⊠⊠塞州平面英尺

Code

```
SELECT ST_Length(ST_GeomFromText(
'LINESTRING(743238 2967416,743238 2967450,743265 2967450,
743265.625 2967416,743238 2967416)',
2249));
```

Figure

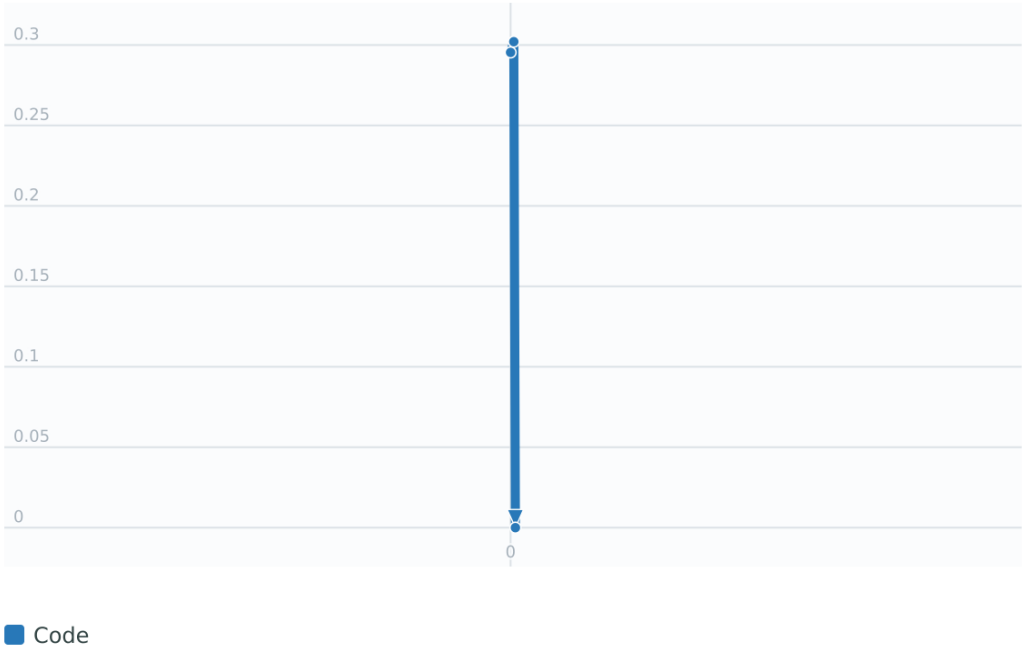


This example transforms a WGS 84 LineString to Massachusetts State Plane meters.

Code

```
SELECT ST_Length(ST_Transform(
        'SRID=4326;LINESTRING(-72.1260 42.45,-72.1240 42.45666,-72.123 42.1546)'::geometry,
            26986
    )
);
```

Figure



几何示例

WGS 84 地理⊠返回⊠度

This example uses the default spheroid calculation.

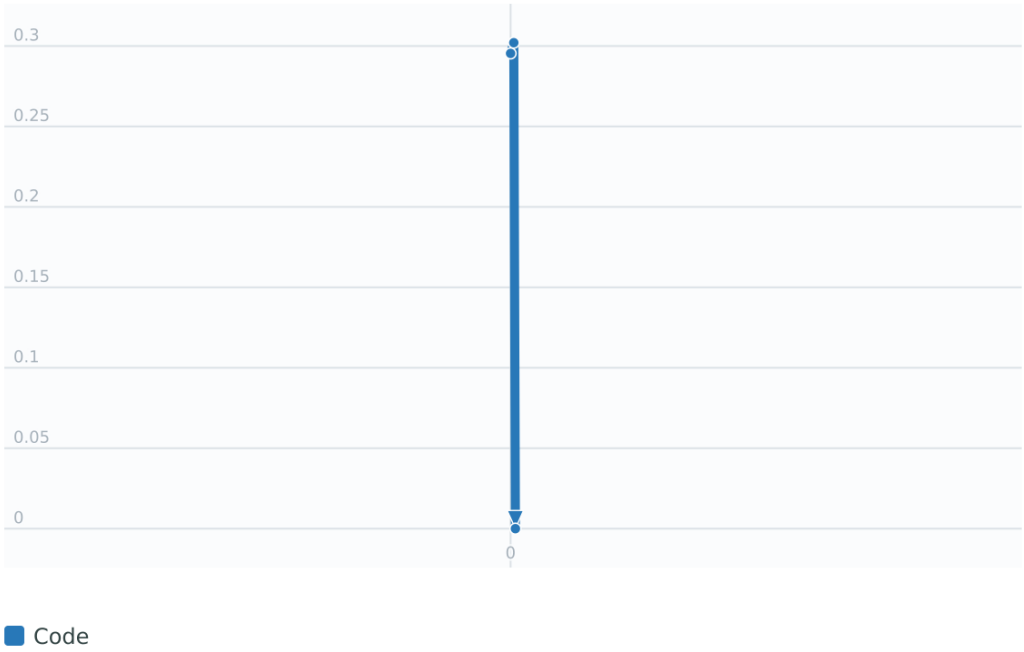
Code

```
SELECT ST_Length(the_geog) As length_spheroid, ST_Length(the_geog, false) As length_sphere
FROM (SELECT ST_GeographyFromText('SRID=4326;LINESTRING(-72.1260 42.45,-72.1240 ↩
    42.45666,-72.123 42.1546)') As the_geog)
 As foo;
```

Output

```
length_spheroid  |  length_sphere
-----------------+-----------------
34310.5703627284 | 34346.20609607416
```

Figure



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

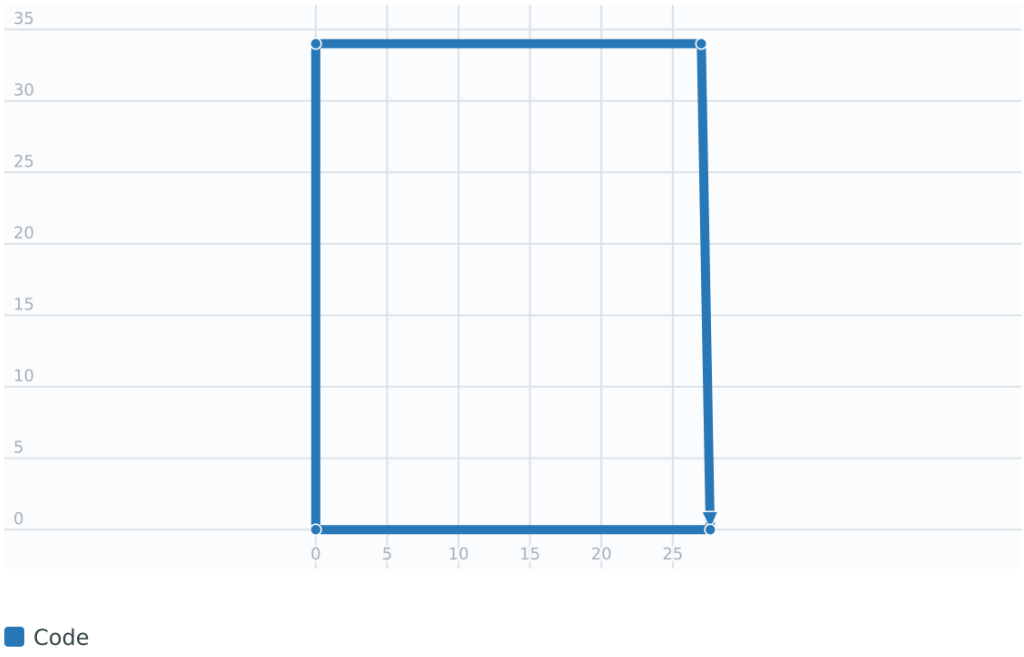
示例

返回 3D ☒☒的☒度（以英尺☒☒位）。☒注意，☒位☒英尺，因☒ EPSG:2249 是☒☒☒塞州平面英尺

Code

```
SELECT ST_3DLength(ST_GeomFromText(
'LINESTRING(743238 2967416 1,743238 2967450 1,743265 2967450 3,
743265.625 2967416 3,743238 2967416 3)',
2249));
```

Figure



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

Calculates the length or perimeter of a geometry on a spheroid. This is useful if the coordinates of the geometry are in longitude/latitude and a length is desired without reprojection. The spheroid is specified by a text value as follows:

```
SPHEROID[<NAME>,<SEMI-MAJOR AXIS>,<INVERSE FLATTENING>]
```

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

Measure a table geometry directly on a spheroid when the coordinates are stored in longitude and latitude.

Code

```
SELECT ST_LengthSpheroid(
    geometry_column,
    'SPHEROID["GRS_1980",6378137,298.257222101]'
    )
FROM geometry_table;
```

This example measures a 2D MultiLineString and each component separately on the same spheroid.

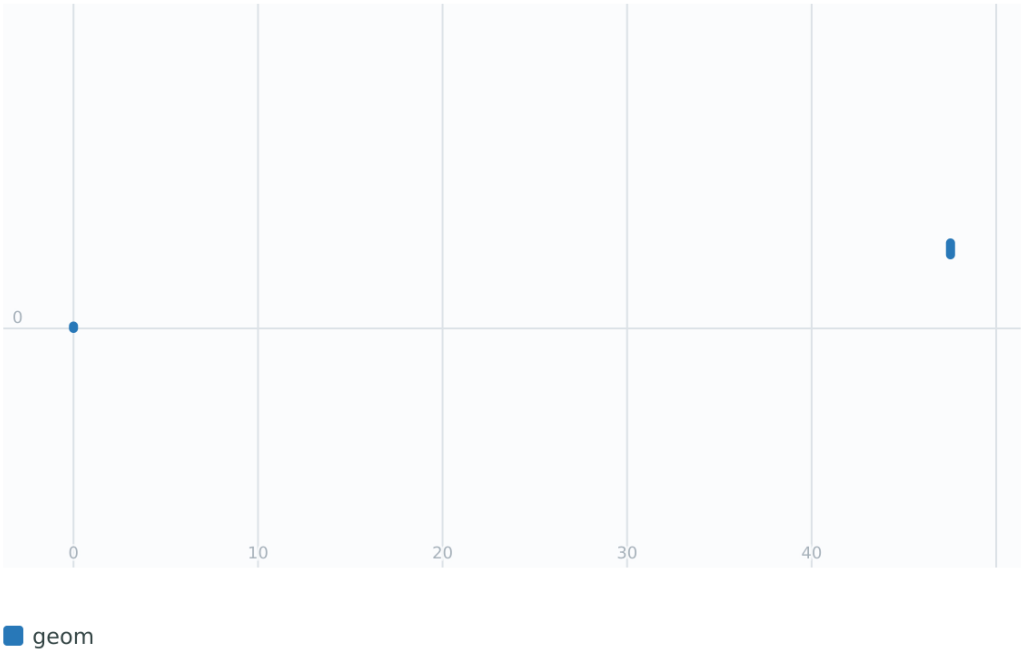
Code

```
WITH sample AS (
  SELECT 'MULTILINESTRING((-118.584 38.374,-118.583 38.5),
          (-71.05957 42.3589,-71.061 43))'::geometry AS geom,
         CAST('SPHEROID["GRS_1980",6378137,298.257222101]' AS spheroid) AS sph_m
)
SELECT ST_LengthSpheroid(geom, sph_m) AS tot_len,
       ST_LengthSpheroid(ST_GeometryN(geom, 1), sph_m) AS len_line1,
       ST_LengthSpheroid(ST_GeometryN(geom, 2), sph_m) AS len_line2
FROM sample;
```

Output

```
tot_len      |    len_line1     |    len_line2
-----------------+-----------------+-----------------
 85204.52077117894 | 13986.87252824332 | 71217.64824293563
```

Figure



This example uses 3D input; Z contributes to the measured result.

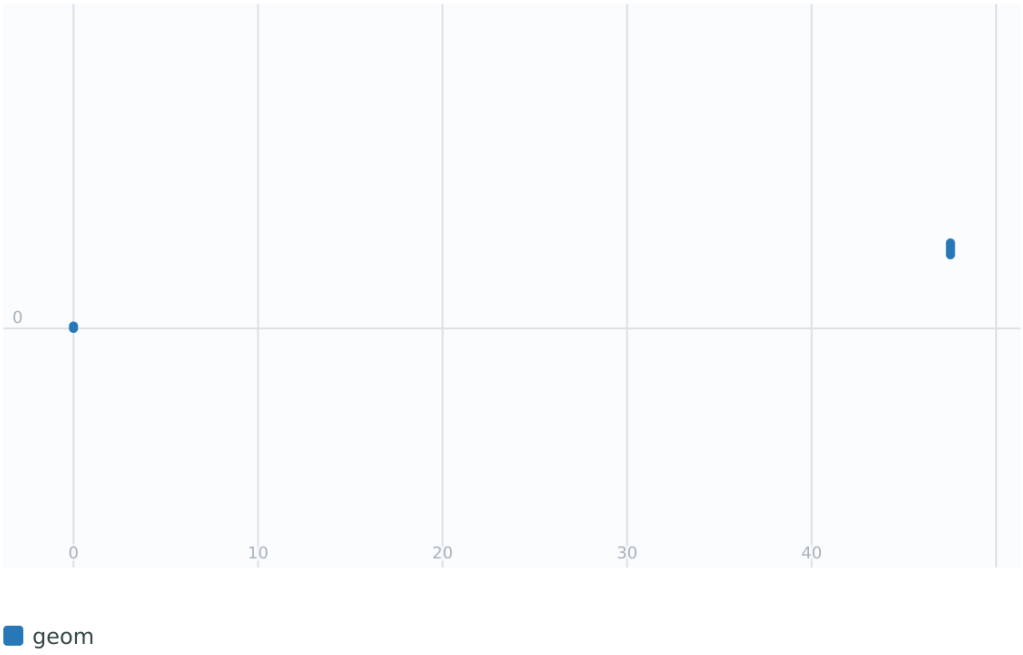
Code

```
WITH sample AS (
  SELECT 'MULTILINESTRING((-118.584 38.374 20,-118.583 38.5 30),
          (-71.05957 42.3589 75,-71.061 43 90))'::geometry AS geom,
         CAST('SPHEROID["GRS_1980",6378137,298.257222101]' AS spheroid) AS sph_m
)
SELECT ST_LengthSpheroid(geom, sph_m) AS tot_len,
       ST_LengthSpheroid(ST_GeometryN(geom, 1), sph_m) AS len_line1,
       ST_LengthSpheroid(ST_GeometryN(geom, 2), sph_m) AS len_line2
FROM sample;
```

Output

```
tot_len      |    len_line1    |    len_line2
-----------------+-----------------+-----------------
 85204.52592562366 | 13986.876103023424 | 71217.64982260024
```

Figure



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 g);
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。

If only one geometry is provided, or g1 and g2 are the same geometry, returns the line between the two vertices farthest apart in the geometry. The endpoints of the line lie on the circle computed by ST_MinimumBoundingCircle.

可用性：1.1.0

Enhanced: 3.7.0 - support for a single geometry input.

示例

Longest line between a Point and a LineString.

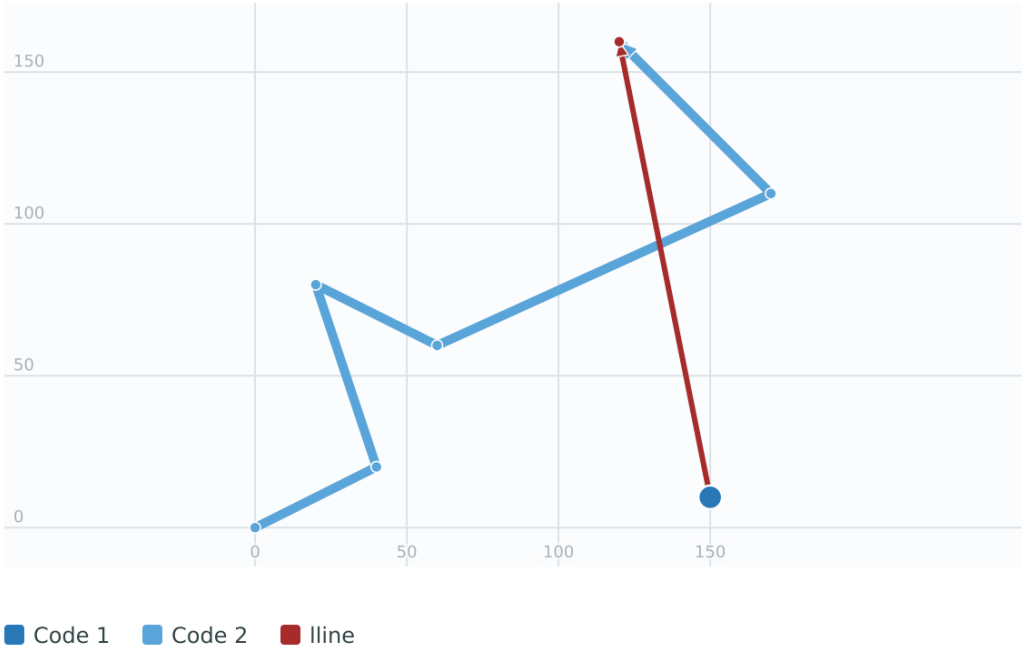
Code

```
SELECT ST_LongestLine(
        'POINT (160 40)',
        'LINESTRING (10 30,50 50,30 110,70 90,180 140,130 190)' ) AS lline;
```

Output

```
LINESTRING(160 40,130 190)
```

Figure



Longest line between two polygons.

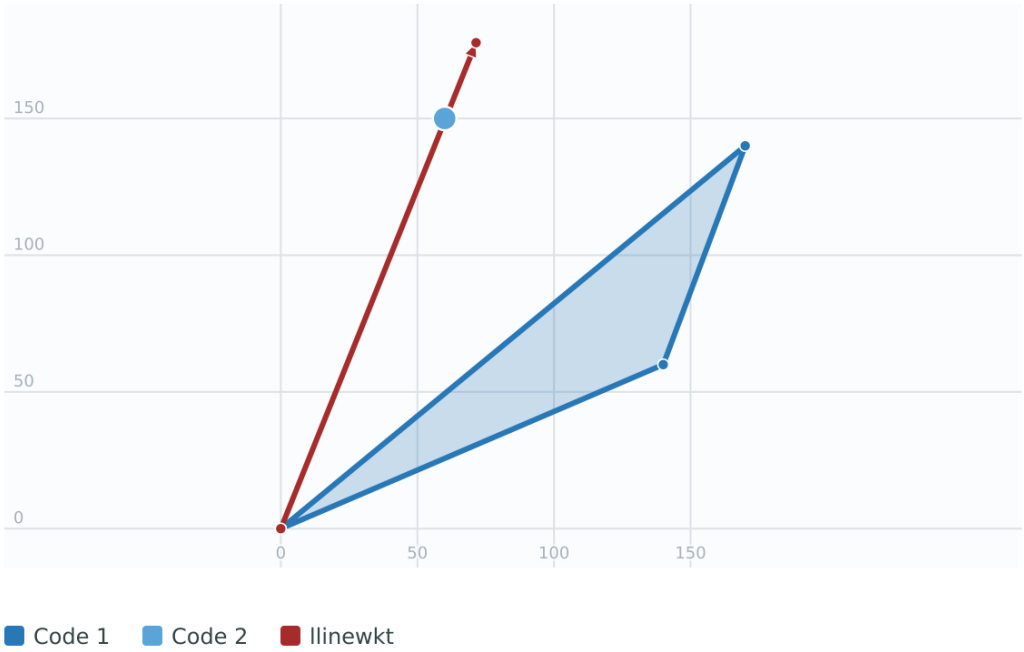
Code

```
SELECT ST_LongestLine(
        'POLYGON ((190 150,20 10,160 70,190 150))',
        ST_Buffer('POINT(80 160)', 30)
            ) AS llinewkt;
```

Output

```
LINESTRING(20 10,91.4805029709527 187.7163859753386)
```

Figure



跨⊠个几何体的最⊠⊠。⊠的⊠度等于最大距离。⊠的端点位于最小⊠界⊠上。

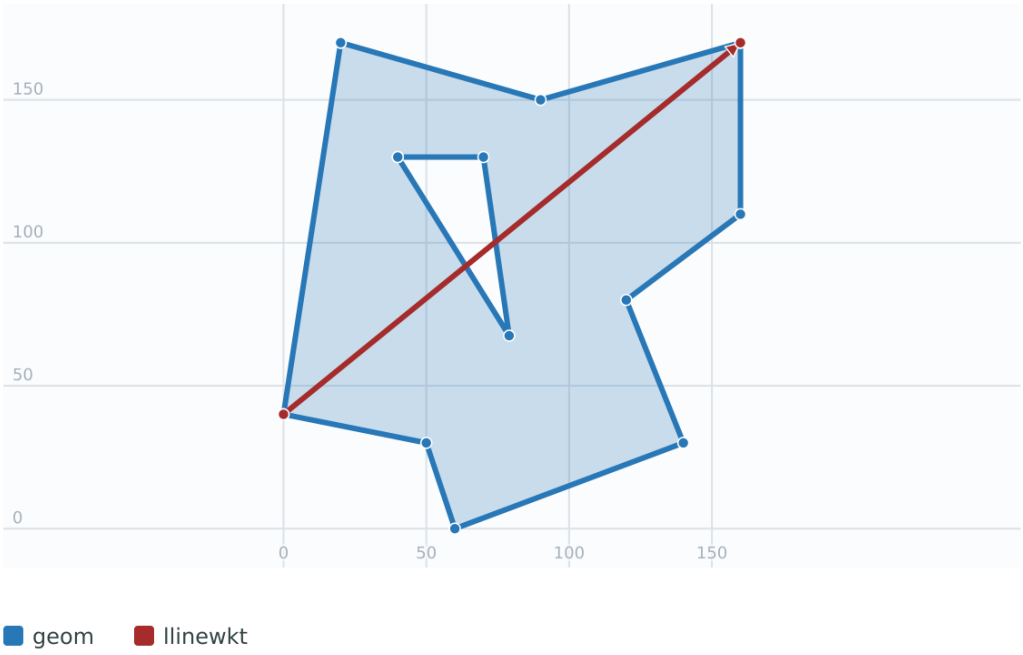
Code

```
SELECT ST_LongestLine(geom) AS llinewkt,
                  ST_MaxDistance(geom) AS max_dist,
                  ST_Length(ST_LongestLine(geom)) AS lenll
FROM (SELECT 'POLYGON ((40 180,110 160,180 180,180 120,140 90,160 40,80 10,70 40,20 50,40 ↩
    180),
              (60 140,99 77.5,90 140,60 140))'::geometry AS geom) AS t;
```

Output

```
llinewkt          |      max_dist      |       lenll
--------------------------+--------------------+--------------------
 LINESTRING(20 50,180 180) | 206.15528128088303 | 206.15528128088303
```

Figure



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

Longest line between a LineString and a Point, comparing the 3D and 2D results.

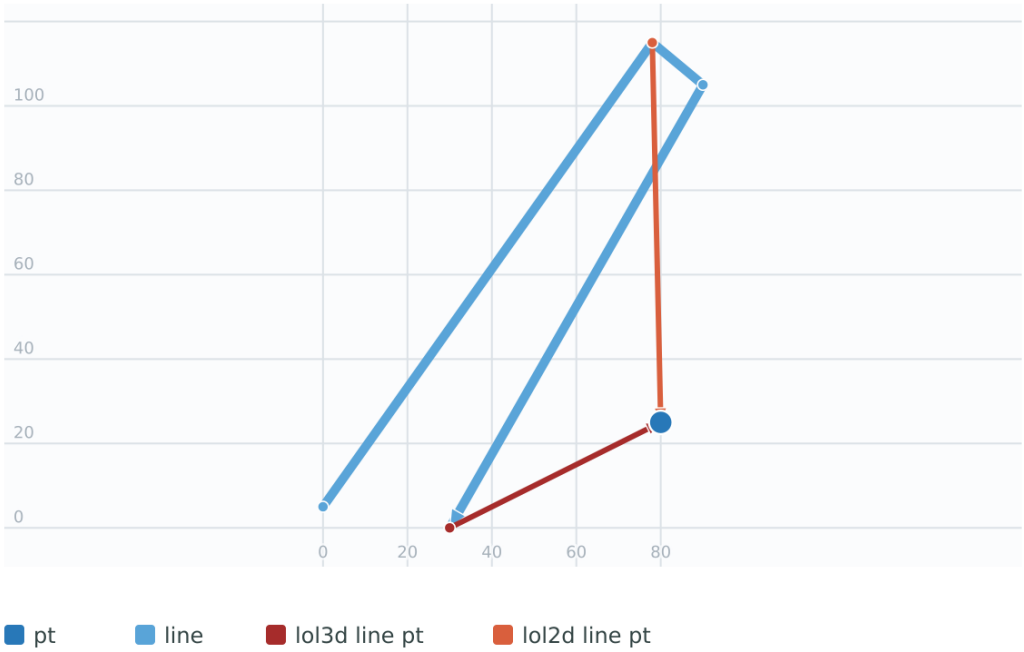
Code

```
SELECT ST_3DLongestLine(line, pt) AS lol3d_line_pt, ST_LongestLine(line, pt) AS  ↩
    lol2d_line_pt
FROM (
    SELECT 'POINT(100 100 30)'::geometry AS pt,
           'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 1000)'::geometry AS line
) AS foo;
```

Output

```
lol3d_line_pt           |       lol2d_line_pt
--------------------------------+---------------------------
 LINESTRING(50 75 1000,100 100 30) | LINESTRING(98 190,100 100)
```

Figure



Longest line between a LineString and a MultiPoint, comparing the 3D and 2D results.

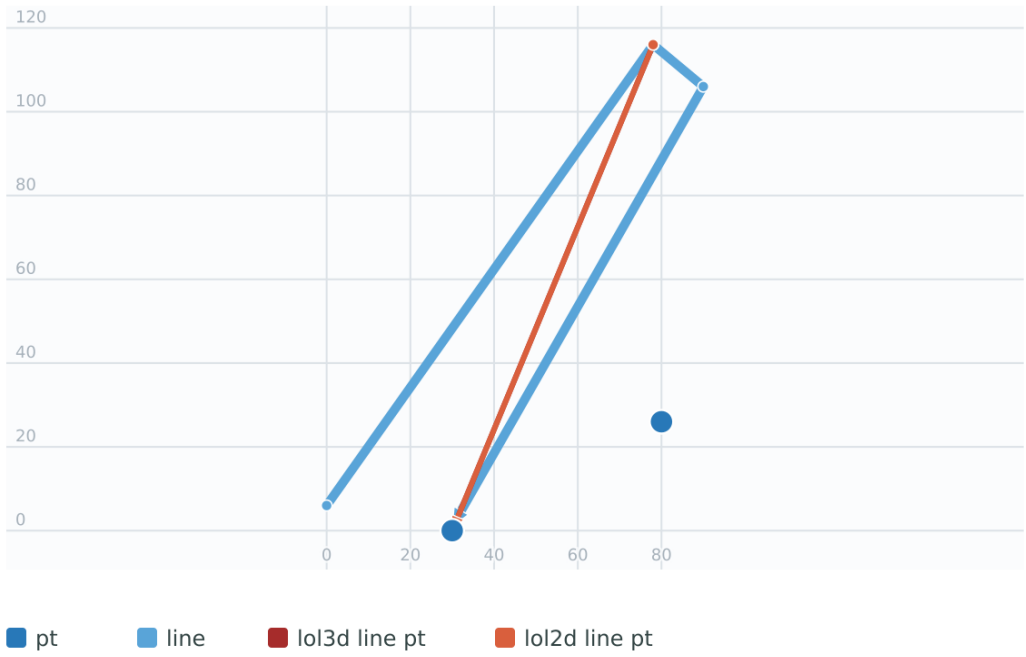
Code

```
SELECT ST_3DLongestLine(line, pt) AS lol3d_line_pt, ST_LongestLine(line, pt) AS  ↩
    lol2d_line_pt
FROM (
    SELECT 'MULTIPOINT(100 100 30,50 74 1000)'::geometry AS pt,
           'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 900)'::geometry AS line
) AS foo;
```

Output

```
lol3d_line_pt          |      lol2d_line_pt
--------------------------------+-------------------------
 LINESTRING(98 190 1,50 74 1000) | LINESTRING(98 190,50 74)
```

Figure



Longest line between a Polygon and a MultiLineString, comparing the 3D and 2D results.

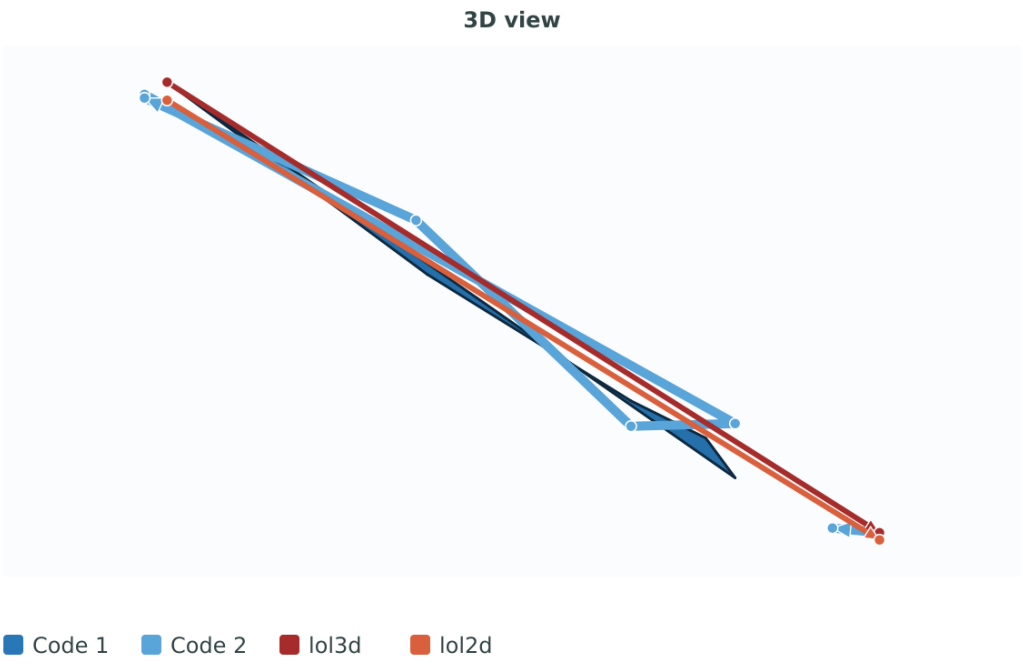
Code

```
SELECT ST_3DLongestLine(poly, mline) AS lol3d, ST_LongestLine(poly, mline) AS lol2d
FROM (
    SELECT ST_GeomFromEWKT(
                'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))'
            ) AS poly,
            ST_GeomFromEWKT(
                'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),
                (1 10 2,5 20 1))'
            ) AS mline
) AS foo;
```

Output

```
lol3d             |          lol2d
------------------------------+-------------------------
 LINESTRING(175 150 5,1 10 2) | LINESTRING(175 150,1 10)
```

Figure



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 g);
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。

If only one geometry is provided, or g1 and g2 are the same geometry, returns the distance between the two vertices farthest apart in that geometry.

可用性：1.1.0

Enhanced: 3.7.0 - support for a single geometry input.

示例

点与⊠之⊠的最大距离。

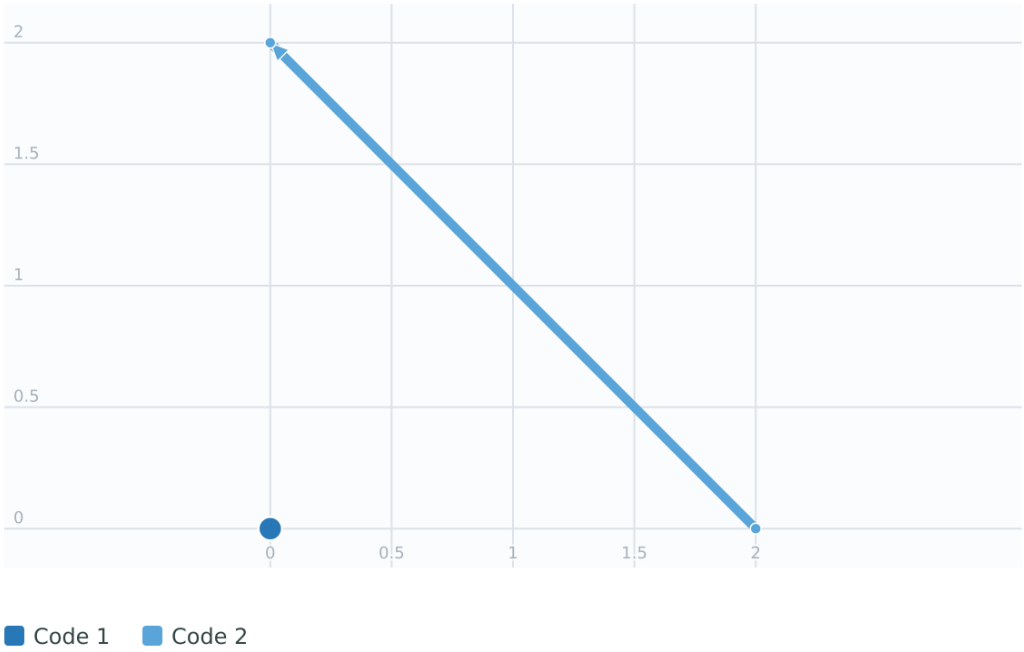
Code

```
SELECT ST_MaxDistance('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
2
```

Figure



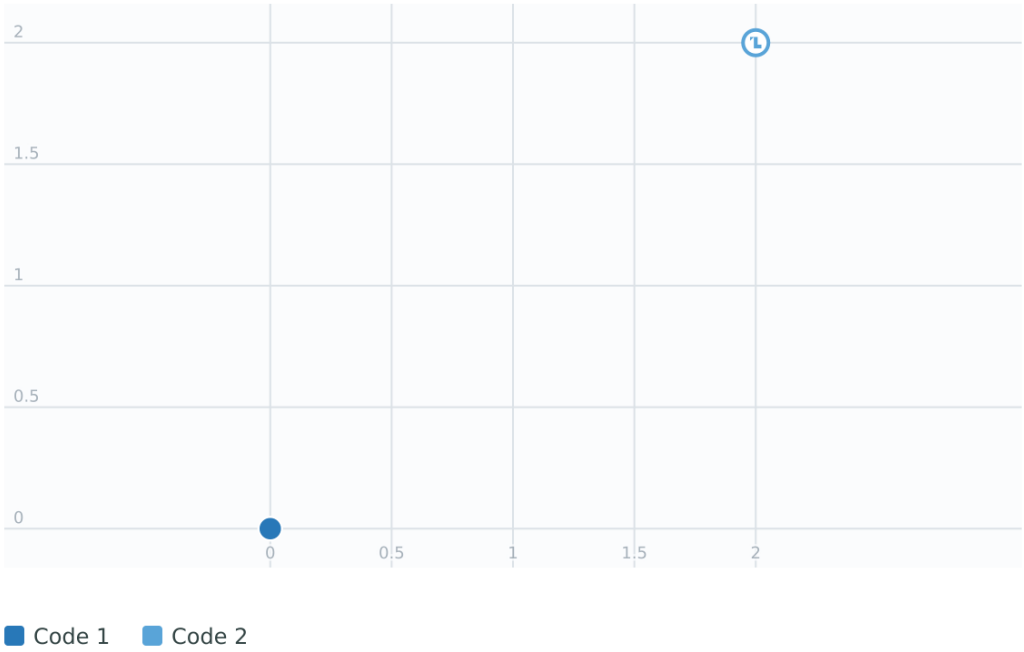
Code

```
SELECT ST_MaxDistance('POINT(0 0)'::geometry, 'LINESTRING ( 2 2,2 2 )'::geometry);
```

Output

```
2.82842712474619
```

Figure



⊠个几何体⊠点之⊠的最大距离。

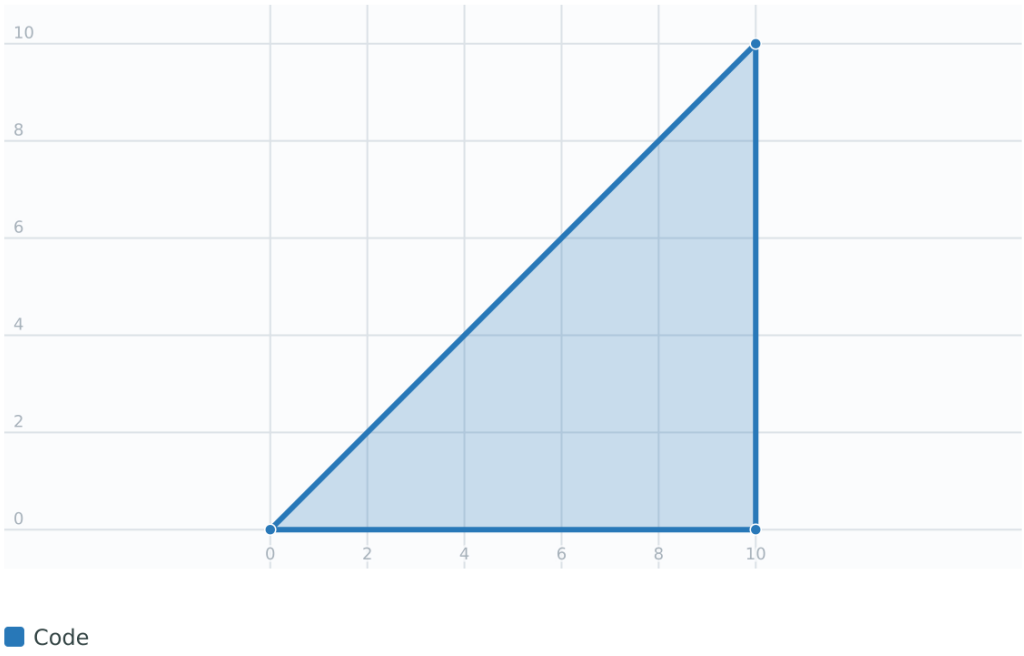
Code

```
SELECT ST_MaxDistance('POLYGON ((10 10,10 0,0 0,10 10))'::geometry);
```

Output

```
14.142135623730951
```

Figure



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 that currently no vertical datum support so Z is not transformed and assumed to be same units as final.

Code

```
SELECT ST_3DMaxDistance(
            ST_Transform('SRID=4326;POINT(-72.1235 42.3521 10000)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)':: ↩
    geometry, 2163)
            ) As dist_3d,
            ST_MaxDistance(
                ST_Transform('SRID=4326;POINT(-72.1235 42.3521 10000)'::geometry, 2163),
                ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45 15,-72.123 42.1546 20)':: ↩
    geometry, 2163)
            ) As dist_2d;
```

Output

```
dist_3d      |     dist_2d
-----------------+-----------------
 24383.7467488441 | 22247.8472107251
```

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

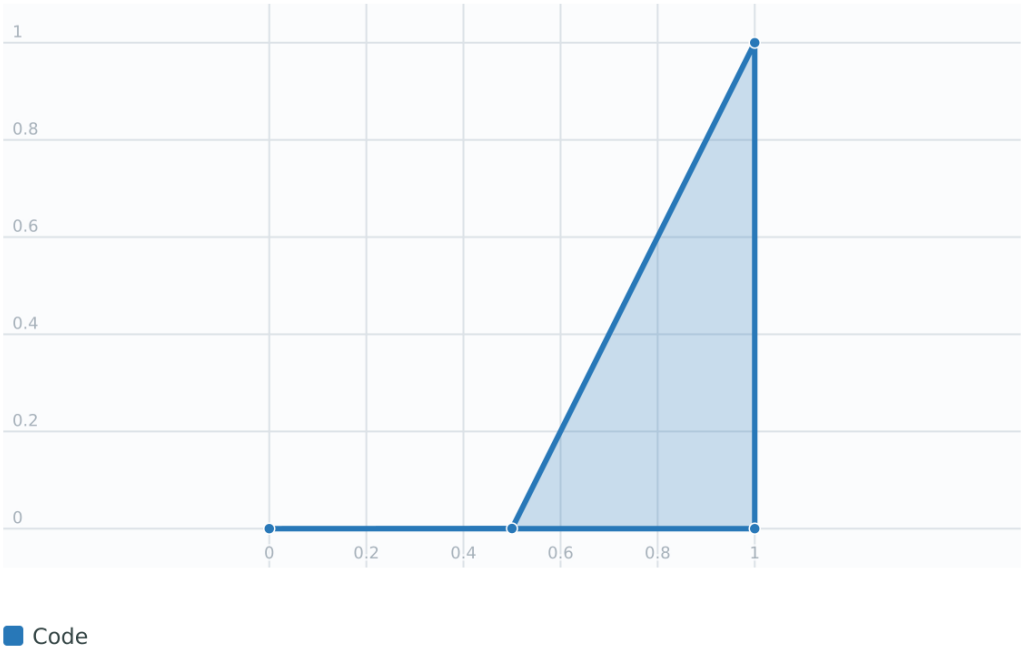
Code

```
SELECT ST_MinimumClearance('POLYGON ((0 0,1 0,1 1,0.5 3.2e-4,0 0))');
```

Output

```
0.00032
```

Figure



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

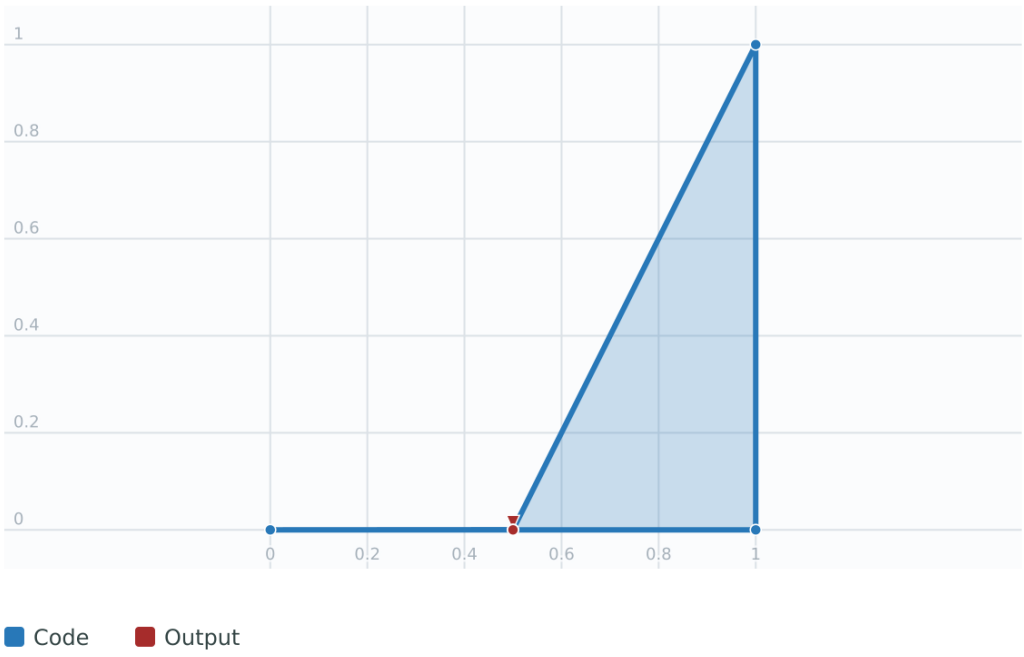
Code

```
SELECT ST_MinimumClearanceLine('POLYGON ((0 0,1 0,1 1,0.5 3.2e-4,0 0))');
```

Output

```
LINESTRING(0.5 0.00032,0.5 0)
```

Figure



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 geography ⊠型，⊠算基于反大地⊠量方法⊠行，周⊠⊠位⊠米。所用⊠球由 SRID 指定；如果未提供 SRID，⊠默⊠使用 WGS84。如果 `use_spheroid = false`，⊠⊠算会基于球体近似而非⊠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

Geometry.
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

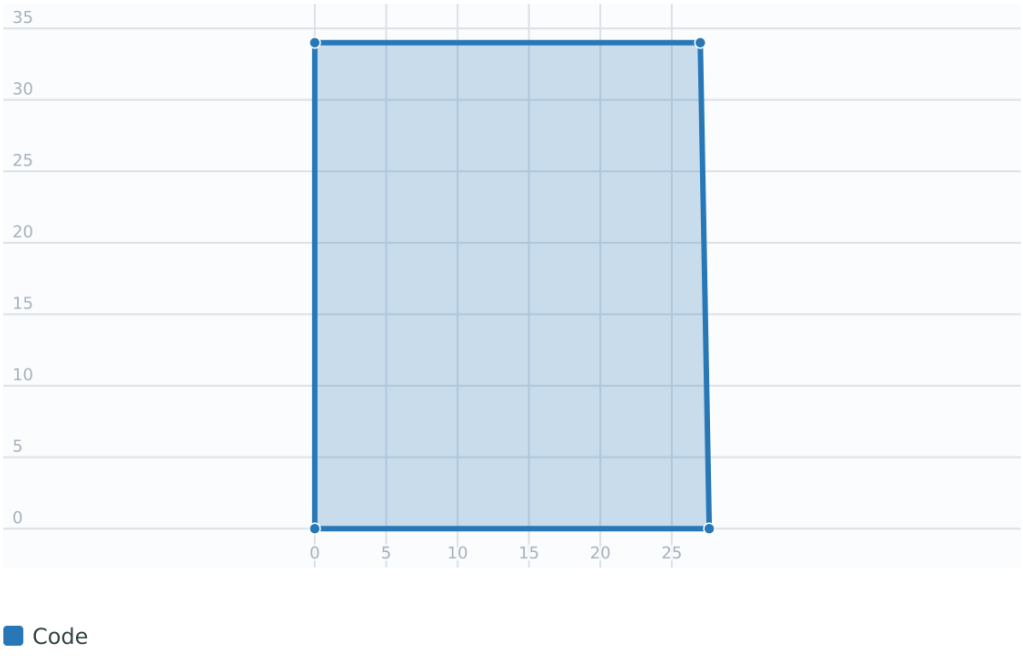
Code

```
SELECT ST_Perimeter(ST_GeomFromText(
'POLYGON((743238 2967416,743238 2967450,743265 2967450,
743265.625 2967416,743238 2967416))',
2249));
```

Output

```
122.63074400009504
```

Figure



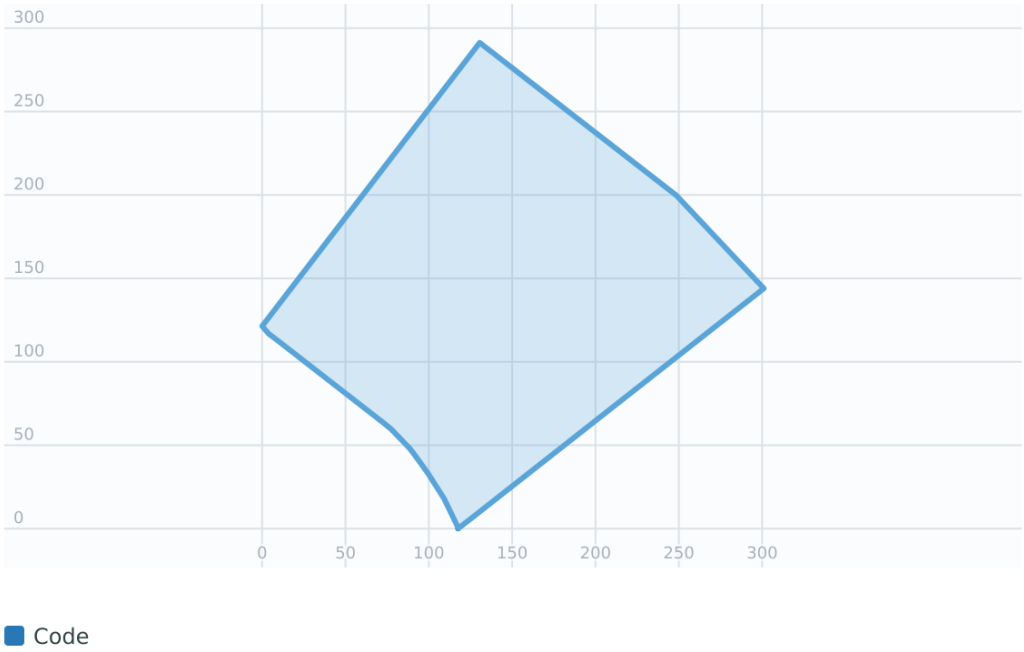
Code

```
SELECT ST_Perimeter(ST_GeomFromText(
'MULTIPOLYGON(((763104.471273676 2949418.44119003,
763104.477769673 2949418.42538203,
763104.189609677 2949418.22343004,763104.471273676 2949418.44119003)),
((763104.471273676 2949418.44119003,763095.804579742 2949436.33850239,
763086.132105649 2949451.46730207,763078.452329651 2949462.11549407,
763075.354136904 2949466.17407812,763064.362142565 2949477.64291974,
763059.953961626 2949481.28983009,762994.637609571 2949532.04103014,
762990.568508415 2949535.06640477,762986.710889563 2949539.61421415,
763117.237897679 2949709.50493431,763235.236617789 2949617.95619822,
763287.718121842 2949562.20592617,763111.553321674 2949423.91664605,
763104.471273676 2949418.44119003)))',
2249));
```

Output

```
845.227713366825
```

Figure



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）

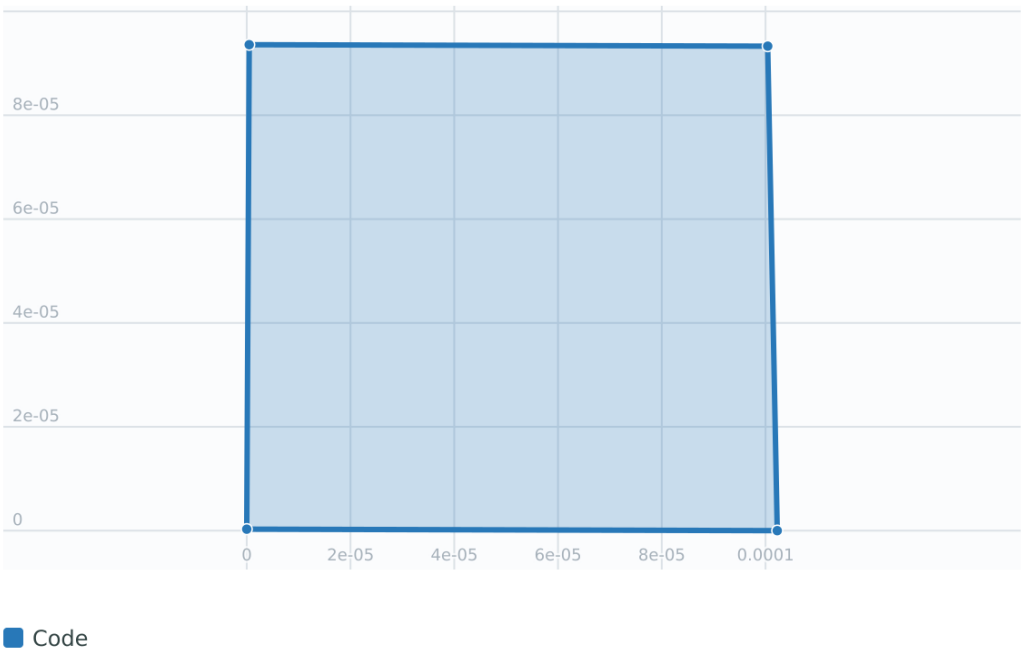
Code

```
SELECT round(ST_Perimeter(geog)::numeric, 8) As per_meters,
       round((ST_Perimeter(geog)/0.3048)::numeric, 8) As per_ft
FROM ST_GeogFromText('POLYGON((-71.1776848522251 42.3902896512902,-71.1776843766326 ↩
    42.3903829478009,
-71.1775844305465 42.3903826677917,-71.1775825927231 42.3902893647987,-71.1776848522251 ↩
    42.3902896512902))') As geog;
```

Output

```
per_meters  |    per_ft
------------+-------------
 37.37904625 | 122.63466619
```

Figure



This example uses a MultiPolygon.

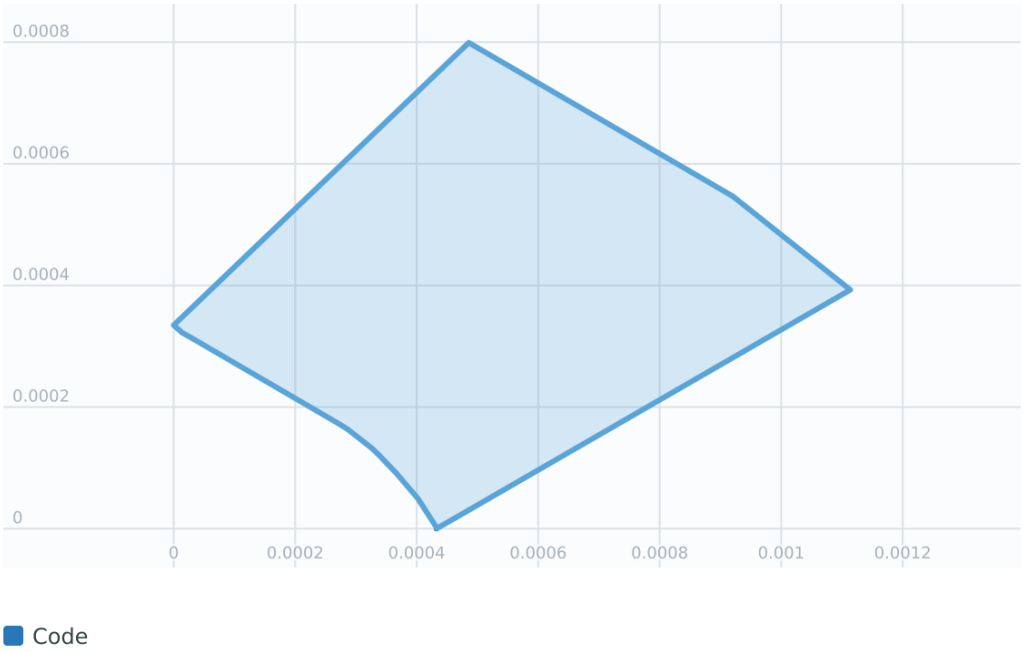
Code

```
SELECT round(ST_Perimeter(geog)::numeric, 8) As per_meters,
       round(ST_Perimeter(geog, false)::numeric, 8) As per_sphere_meters,
       round((ST_Perimeter(geog)/0.3048)::numeric, 8) As per_ft
FROM ST_GeogFromText('MULTIPOLYGON(((-71.1044543107478 42.340674480411,-71.1044542869917 ↩
    42.3406744369506,
-71.1044553562977 42.340673886454,-71.1044543107478 42.340674480411)),
((-71.1044543107478 42.340674480411,-71.1044860600303 42.3407237015564,-71.1045215770124 ↩
    42.3407653385914,
-71.1045498002983 42.3407946553165,-71.1045611902745 42.3408058316308,-71.1046016507427 ↩
    42.340837442371,
-71.104617893173 42.3408475056957,-71.1048586153981 42.3409875993595,-71.1048736143677 ↩
    42.3409959528211,
-71.1048878050242 42.3410084812078,-71.1044020965803 42.3414730072048,
-71.1039672113619 42.3412202916693,-71.1037740497748 42.3410666421308,
-71.1044280218456 42.3406894151355,-71.1044543107478 42.340674480411)))') As geog;
```

Output

```
per_meters  | per_sphere_meters |    per_ft
-------------+-------------------+--------------
 257.63428368 |      257.41231145 | 845.25683623
```

Figure



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⌧

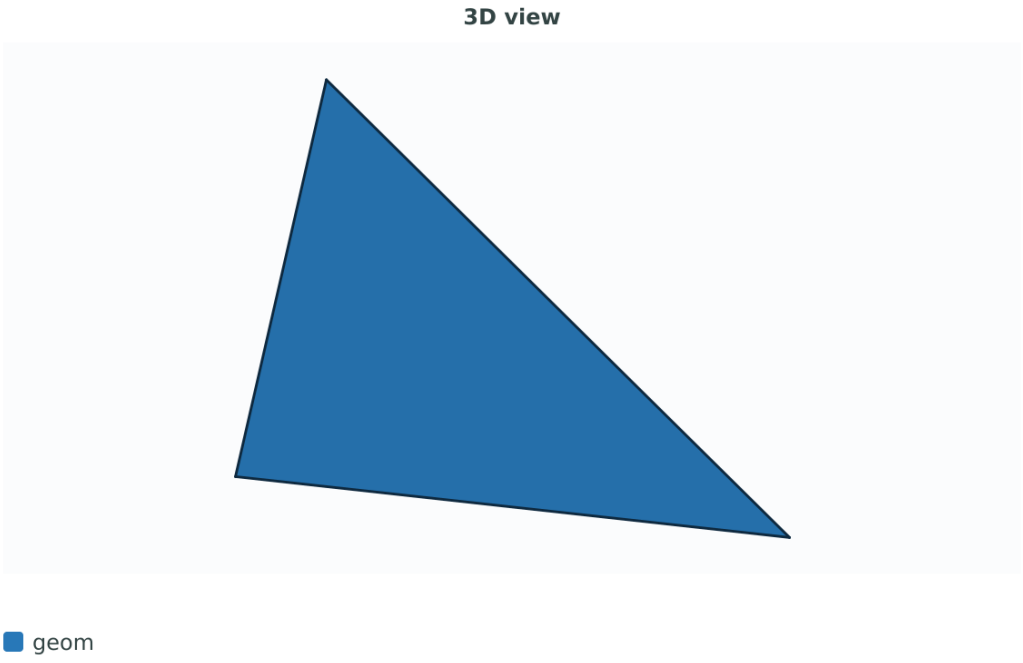
Code

```
SELECT ST_3DPerimeter(geom), ST_Perimeter2D(geom), ST_Perimeter(geom) FROM
            (SELECT 'SRID=2249;POLYGON((743238 2967416 2,743238 2967450 1,
743265.625 2967416 1,743238 2967416 2))'::geometry As geom) As foo;
```

Output

```
ST_3DPerimeter  |  st_perimeter2d  |   st_perimeter
------------------+------------------+------------------
105.46579359767418 | 105.43299727218763 | 105.43299727218763
```

Figure



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

Shortest line between a Point and a LineString.

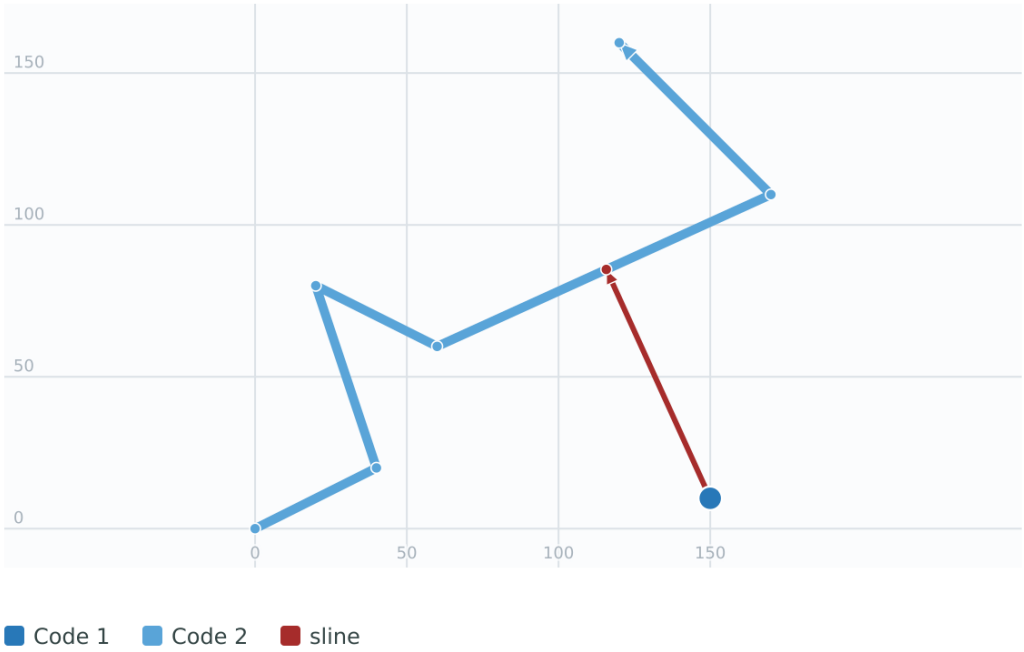
Code

```
SELECT ST_ShortestLine(
        'POINT (160 40)',
        'LINESTRING (10 30,50 50,30 110,70 90,180 140,130 190)') As sline;
```

Output

```
LINESTRING(160 40,125.75342465753425 115.34246575342466)
```

Figure



Shortest line between polygons.

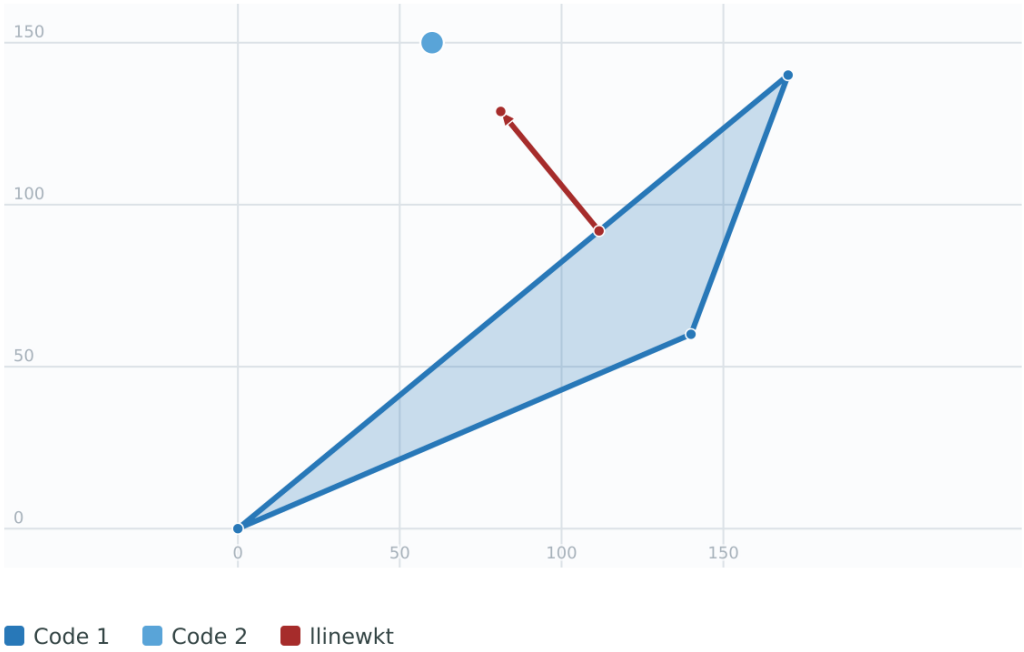
Code

```
SELECT ST_ShortestLine(
         'POLYGON ((190 150,20 10,160 70,190 150))',
         ST_Buffer('POINT(80 160)', 30)
                 ) AS llinewkt;
```

Output

```
LINESTRING(131.59149149528952 101.89887534906195,101.21320343559643 138.78679656440357)
```

Figure



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

Compare the 3D shortest line to the 2D shortest line for a LineString and a Point.

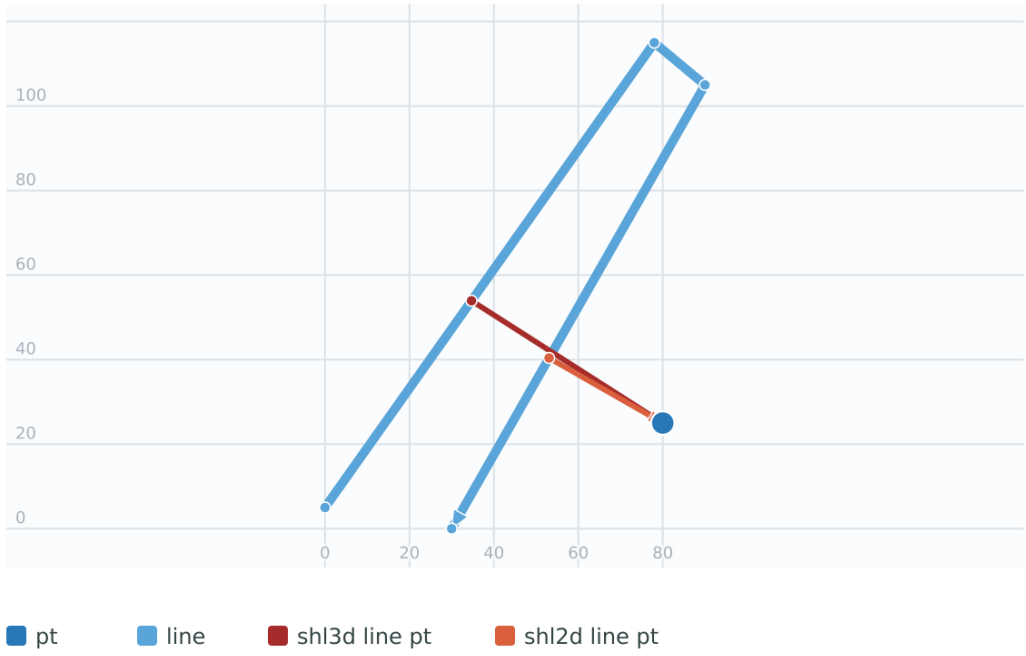
Code

```
WITH input AS (
  SELECT 'POINT(100 100 30)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 1000)'::geometry AS line
)
SELECT ST_3DShortestLine(line, pt) AS shl3d_line_pt,
       ST_ShortestLine(line, pt) AS shl2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+-------------------------------------------------------------------------------
shl3d_line_pt  | LINESTRING(54.69937988676193 128.93502291722837 11.547586950660556,100 100 30)
shl2d_line_pt  | LINESTRING(73.07692307692307 115.38461538461539,100 100)
```

Figure



Compare the 3D shortest line to the 2D shortest line for a LineString and a MultiPoint.

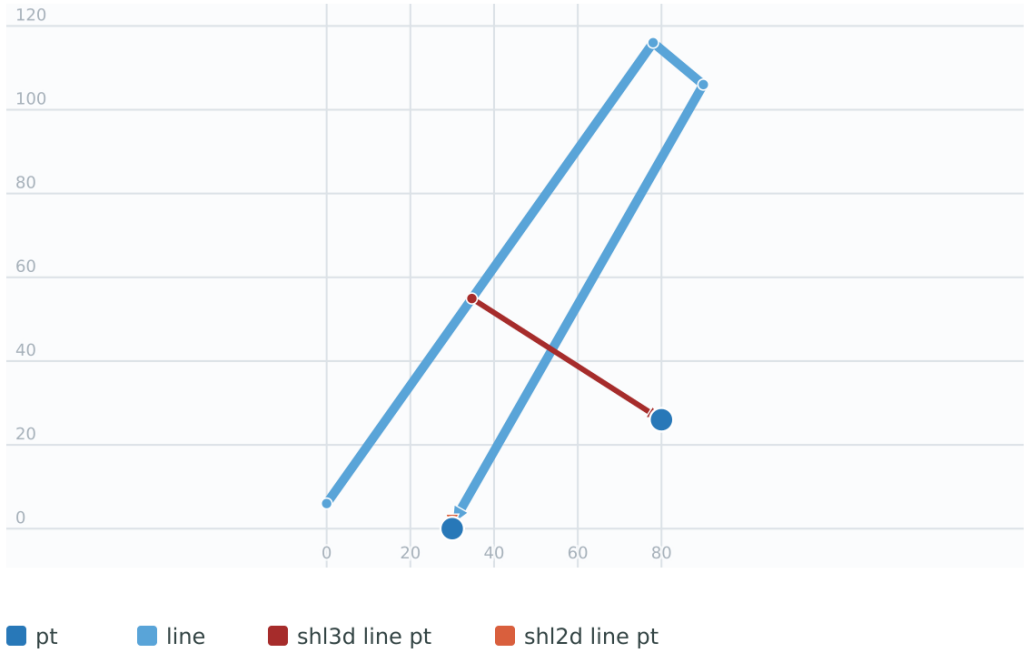
Code

```
WITH input AS (
  SELECT 'MULTIPOINT(100 100 30,50 74 1000)'::geometry AS pt,
         'LINESTRING (20 80 20,98 190 1,110 180 3,50 75 900)'::geometry AS line
)
SELECT ST_3DShortestLine(line, pt) AS shl3d_line_pt,
       ST_ShortestLine(line, pt) AS shl2d_line_pt
FROM input;
```

Output

```
-[ RECORD 1 ]--+-------------------------------------------------------------------------------
shl3d_line_pt  | LINESTRING(54.69937988676193 128.93502291722837 11.547586950660556,100 100 30)
shl2d_line_pt  | LINESTRING(50 75,50 74)
```

Figure



Compare the 3D shortest line to the 2D shortest line for a Polygon and a MultiLineString.

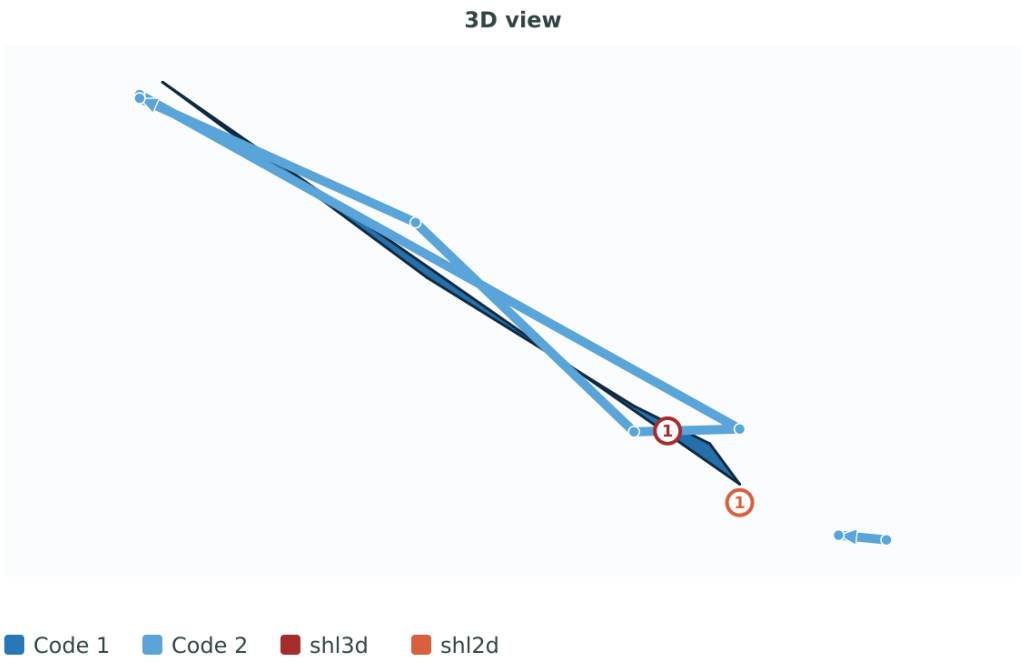
Code

```
WITH input AS (
  SELECT ST_GeomFromEWKT(
           'POLYGON((175 150 5,20 40 5,35 45 5,50 60 5,100 100 5,175 150 5))'
         ) AS poly,
         ST_GeomFromEWKT(
           'MULTILINESTRING((175 155 2,20 40 20,50 60 -2,125 100 1,175 155 1),
                            (1 10 2,5 20 1))'
         ) AS mline
)
SELECT ST_3DShortestLine(poly, mline) AS shl3d,
       ST_ShortestLine(poly, mline) AS shl2d
FROM input;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------------------------------------
shl3d          | LINESTRING(40.45454545454545 53.63636363636364 5,40.45454545454545 53.63636363636364 5)
shl2d          | LINESTRING(20 40,20 40)
```

Figure



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

This example relies on the implicit cast from geometry to box2d for the second parameter.

Code

```
SELECT ST_ClipByBox2D(geom, ST_MakeEnvelope(0, 0, 10, 10)) FROM mytab;
```

相关信息

ST_Intersection, ST_MakeBox2D, ST_MakeEnvelope

7.13.2 ST_Difference

ST_Difference — 区算表示几何 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

返回一个几何区形，表示几何区形 A 中不与几何区形 B 相交的部分。区相当于 `A - ST_Intersection(A,B)`。如果 A 完全包含在 B 中，区返回适当区型的空原子几何区区。



Note
区是唯一一个与区入区序有关的区加函数。ST_Difference(A, B) 始区返回 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 snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

它是通区 GEOS 模区区区的

增区：3.1.0 接受 gridSize 参数。

需要 GEOS >= 3.9.0 才能使用 gridSize 参数。

 此方法区区了 SQL 1.1 的 OGC 区区功能区范。 s2.1.1.3

 区方法区区了 SQL/MM 区范。SQL-MM 3: 5.1.20

 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。

示例

2D 区串的差异。

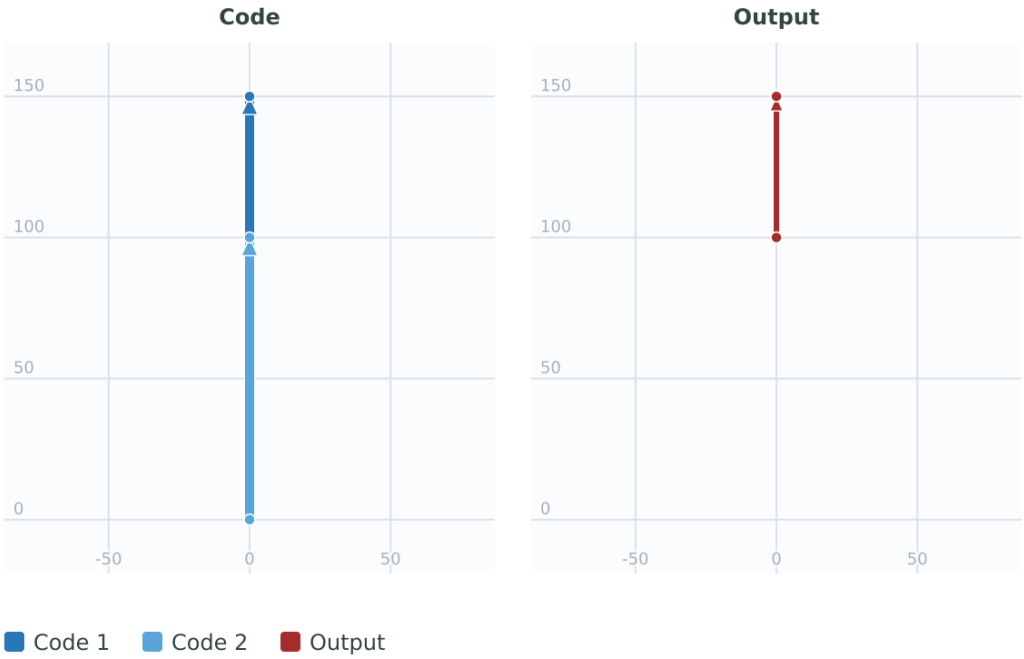
Code

```
SELECT ST_Difference('LINESTRING(50 100,50 200)'::geometry,
        'LINESTRING(50 50,50 150)'::geometry
    );
```

Output

```
LINESTRING(50 150,50 200)
```

Figure



3D 点的差异。

Code

```
SELECT ST_Difference('MULTIPOINT(-118.58 38.38 5,-118.60 38.329 6,-118.614 38.281 7)' ::  ↩
    geometry,
 'POINT(-118.614 38.281 5)' :: geometry
);
```

Output

```
MULTIPOINT(-118.6 38.329 6,-118.58 38.38 5)
```

Figure



相关信息

ST_SymDifference, ST_Intersection, ST_Union, ST_ReducePrecision

7.13.3 ST_Intersection

ST_Intersection — 计算表示几何 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

返回表示两个几何图形的点集交集的几何图形。简单来说，几何图形 A 和几何图形 B 在两个几何图形之间共享的部分。

如果几何图形没有共同点（即不相交），则返回适当类型的空原子几何图形。

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

ST_Intersection 与 ST_Intersects 结合使用对于剪切几何图形非常有用，例如在矩形框、缓冲区或区域类中，您只需要感兴趣的国家或地区内的几何图形部分。

Note
Spatial predicates such as ST_Intersects and overlay functions such as ST_Intersection are computed by different algorithms. For very close or nearly coincident coordinates, floating-point precision and geometric robustness may make a predicate return true while the computed intersection is empty, or place an intersection point slightly away from the input line. For repeatable overlay results, snap inputs to a suitable grid first, for example with ST_ReducePrecision, or use the gridSize parameter when all input vertices already lie on the grid.

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
If working with 3D geometries, you may want to use SFCGAL based CG_3DIntersection which does a proper 3D intersection for 3D geometries. Although this function works with Z-coordinate, it does an averaging of Z-Coordinate.

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

Code

```
SELECT ST_Intersection('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
POINT EMPTY
```

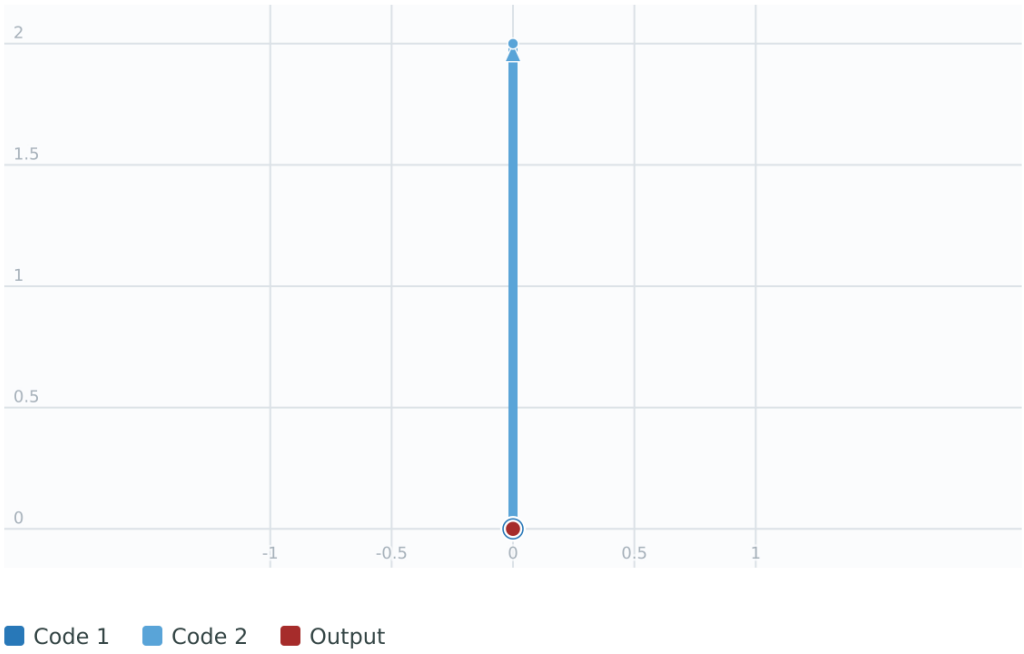
Code

```
SELECT ST_Intersection('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Output

```
POINT(0 0)
```

Figure



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。

Code

```
select clipped.gid, clipped.f_name, clipped_geom
from (
         select trails.gid, trails.f_name,
             (ST_Dump(ST_Intersection(country.geom, trails.geom))).geom clipped_geom
         from country
              inner join trails on ST_Intersects(country.geom, trails.geom)
     ) as clipped
where ST_Dimension(clipped.clipped_geom) = 1;
```

⊠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。）

Code

```
select poly.gid,
    ST_Multi(ST_Buffer(ST_Intersection(country.geom, poly.geom),
             0.0
        )
```

```
    ) clipped_geom
from country
     inner join poly on ST_Intersects(country.geom, poly.geom)
where not ST_IsEmpty(ST_Buffer(ST_Intersection(country.geom, poly.geom), 0.0));
```

Note this is not a true 3D intersection. It uses 2.5D geometries (geometries with Z ordinates, but where calculations are performed in 2D). The same input is compared with CG_3DIntersection, which performs a true 3D intersection.

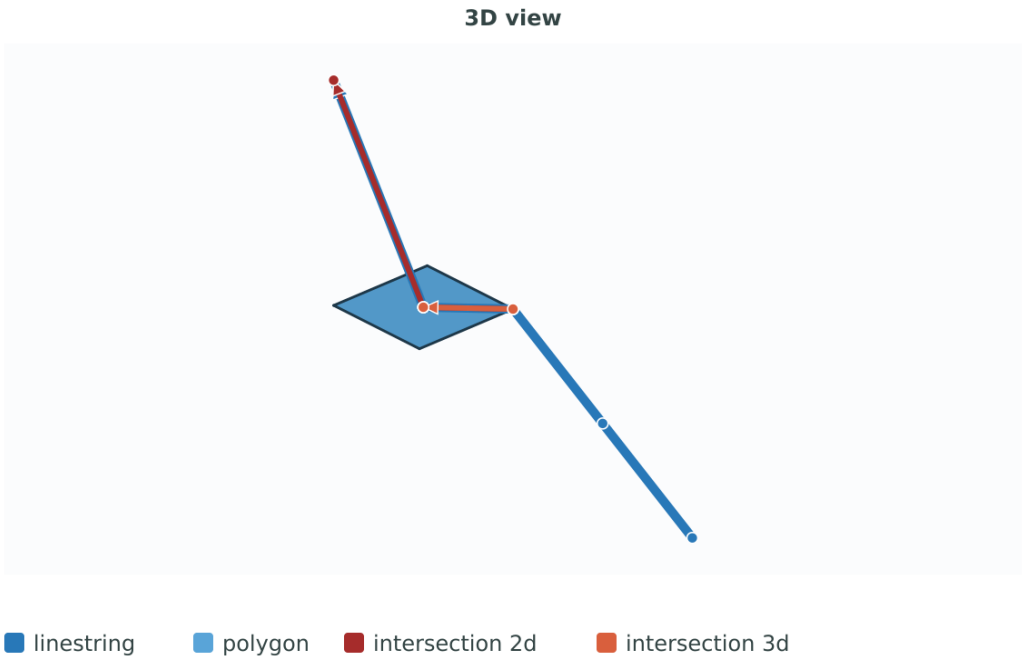
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)'::geometry AS  ↩
    input_linestring,
         'POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8))'::geometry AS input_polygon
)
SELECT input_linestring AS input_linestring,
       input_polygon AS input_polygon,
       ST_Intersection(input_linestring, input_polygon) AS intersection_2d,
       CG_3DIntersection(input_linestring, input_polygon) AS intersection_3d
FROM data;
```

Output

```
LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10) | POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8)) | LINESTRING Z (1  ↩
    1 8,0.5 0.5 8,0 0 10) | LINESTRING Z (1 1 8,0.5 0.5 8)
```

Figure



相关信息

CG_3DIntersection, ST_Difference, ST_Union, ST_ClipByBox2D, ST_Dimension, ST_Dump, ST_Force2D, ST_SymDifference, ST_Intersects, ST_Multi, ST_ReducePrecision

7.13.4 ST_MemUnion

ST_MemUnion — 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

Code

```
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

Noding a 3D LineString which self-intersects.

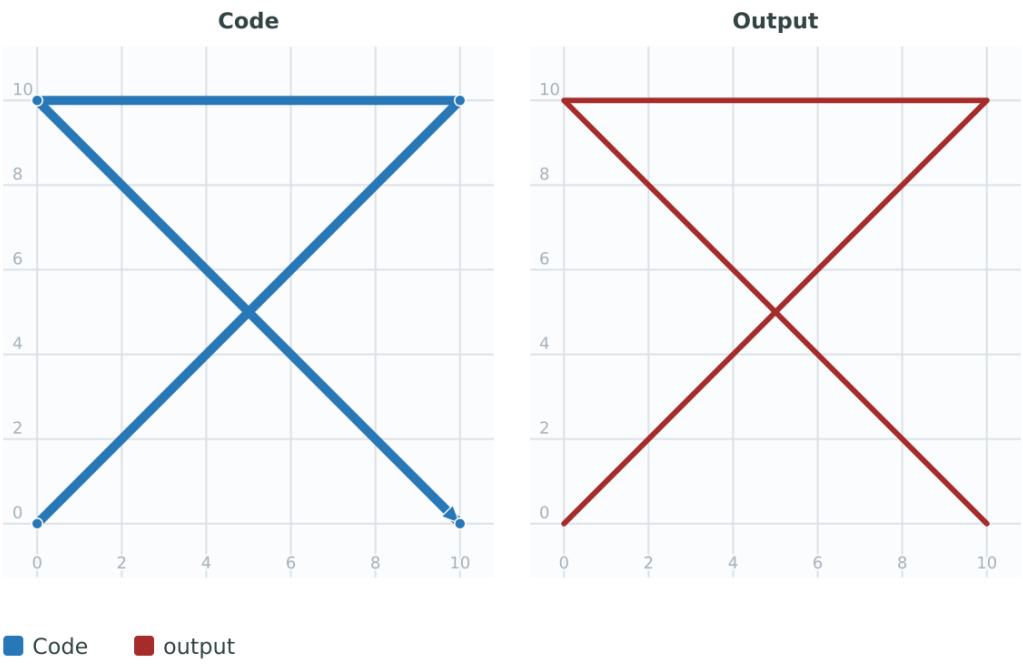
Code

```
SELECT ST_Node('LINESTRINGZ(0 0 0, 10 10 10, 0 10 5, 10 0 3)'::geometry) As  output;
```

Output

```
MULTILINESTRING Z ((0 0 0,5 5 4.5),(5 5 4.5,10 10 10,0 10 5,5 5 4.5),(5 5 4.5,10 0 3))
```

Figure



将⊠点添加到共享一条⊠的⊠个⊠串。生成的⊠被溶解。

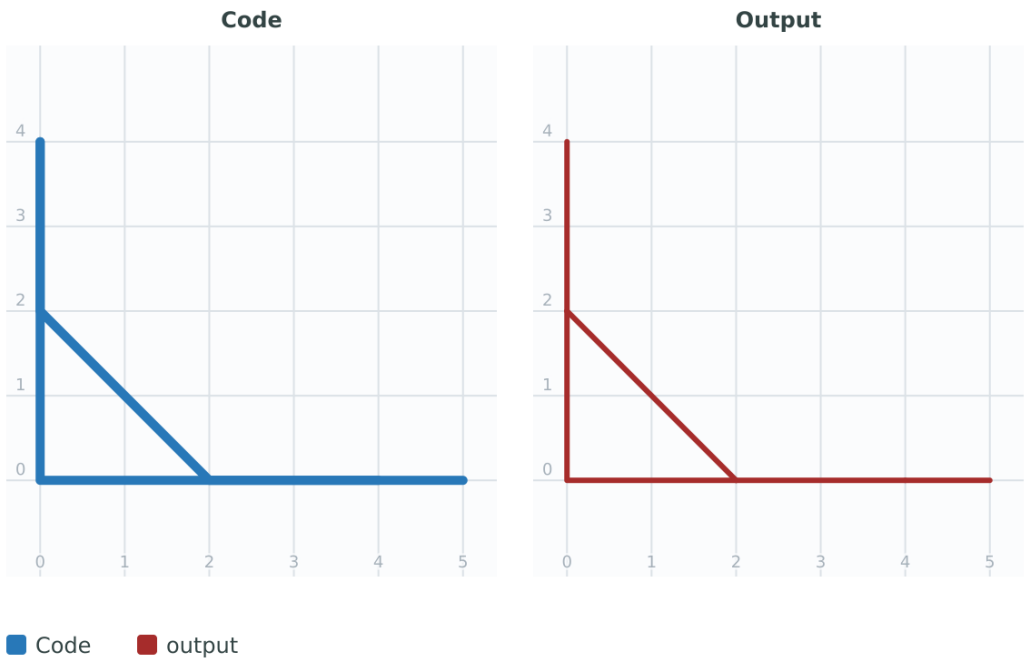
Code

```
SELECT ST_Node('MULTILINESTRING ((2 5,2 1,7 1),(6 1,4 1,2 3,2 5))'::geometry) As  output;
```

Output

```
MULTILINESTRING((2 5,2 3),(2 3,2 1,4 1),(4 1,2 3),(4 1,6 1),(6 1,7 1))
```

Figure



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

Split a circular polygon by a diagonal line.

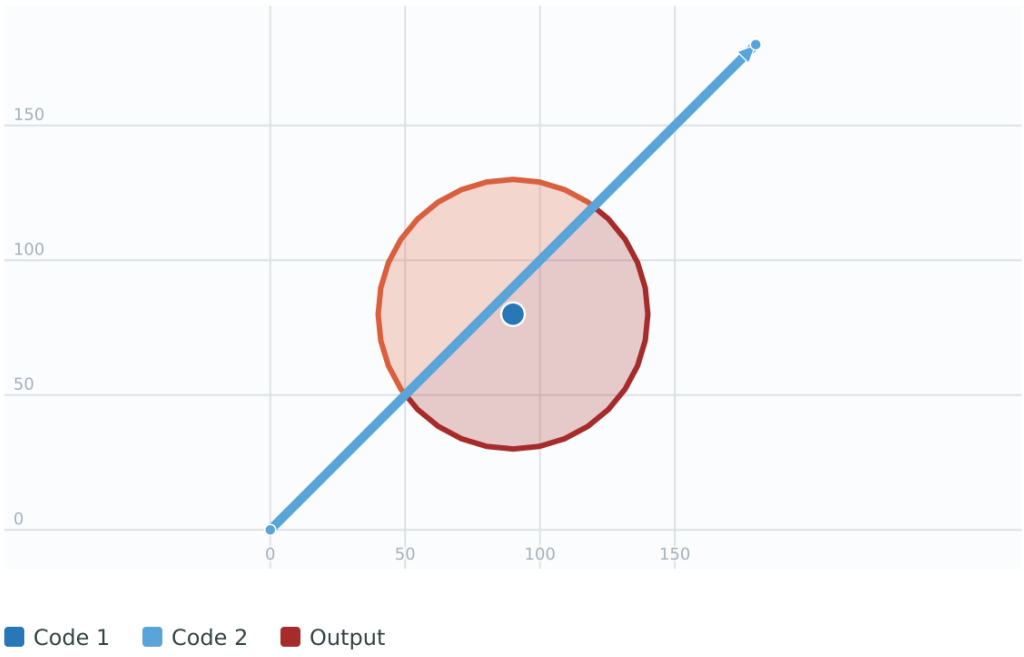
Code

```
SELECT
         ST_Split(
           ST_Buffer('POINT(100 90)'::geometry, 50),
           'LINESTRING(10 10,190 190)'::geometry
         );
```

Output

```
GEOMETRYCOLLECTION(POLYGON((150 90,149 80.2,146.2 70.9,141.6 62.2,135.4 54.6,127.8 48.4,119.1 43.8,109.8 41,100 ↩
    40,90.2 41,80.9 43.8,72.2 48.4,64.6 54.6,60.1 60.1,129.9 129.9,135.4 125.4,141.6 117.8,146.2 109.1,149 99.8,150 ↩
    90)), POLYGON((60.1 60.1,58.4 62.2,53.8 70.9,51 80.2,50 90,51 99.8,53.8 109.1,58.4 117.8,64.6 125.4,72.2 ↩
    131.6,80.9 136.2,90.2 139,100 140,109.8 139,119.1 136.2,127.8 131.6,129.9 129.9,60.1 60.1)))
```

Figure



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。

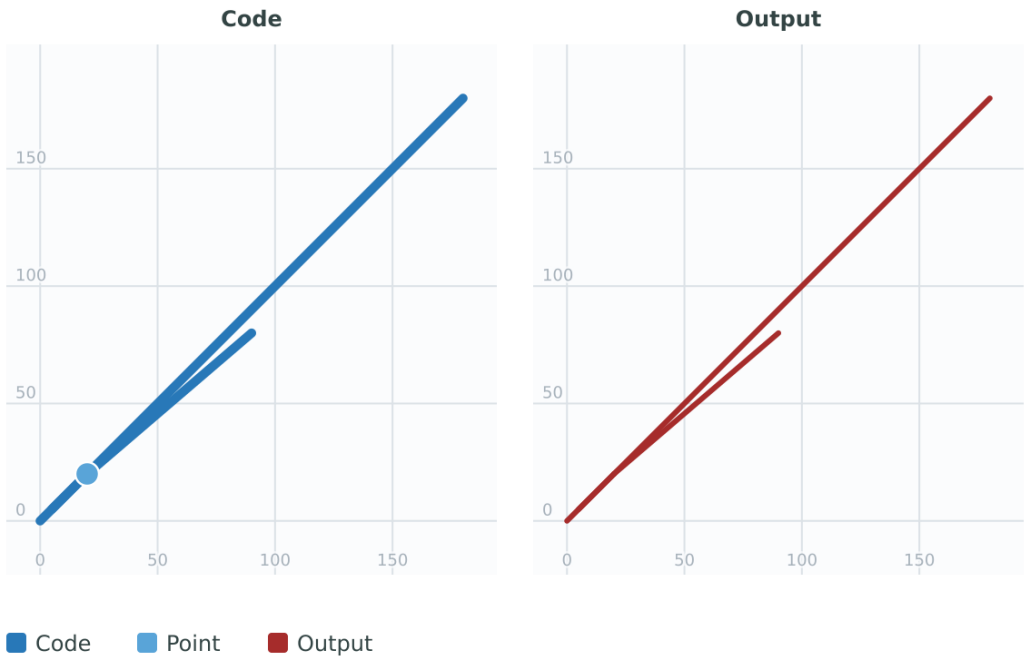
Code

```
SELECT ST_Split('MULTILINESTRING((10 10,190 190),(15 15,30 30,100 90))',
    ST_Point(30, 30));
```

Output

```
GEOMETRYCOLLECTION(
    LINESTRING(10 10,30 30),
    LINESTRING(30 30,190 190),
    LINESTRING(15 15,30 30),
    LINESTRING(30 30,100 90)
)
```

Figure



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。

Code

```
WITH data AS (SELECT
  'LINESTRING(0 0,100 100)'::geometry AS line,
  'POINT(51 50)':: geometry AS point
)
SELECT ST_Split(ST_Snap(line, point, 1), point) AS snapped_split,
       ST_Split(line, point) AS not_snapped_not_split
       FROM data;
```

Figure



相关信息

ST_Snap, ST_Union

7.13.7 ST_Subdivide

ST_Subdivide — 计算几何体的直角分。

Synopsis

setof geometry **ST_Subdivide**(geometry geom, integer max_vertices=256, float8 gridSize = -1);

描述

返回一个几何形，这些几何形是使用直角将 `geom` 分成多个部分的结果，每个部分包含不超过 `max_vertices`。`max_vertices` 必须不小于 5，因为表示一个封闭盒子至少需要 5 个点。

If the optional `gridSize` parameter is given (GEOS-3.9.0 or higher required), all result vertices are guaranteed to fall on a snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

对于索引分数据集，多形内的点和其他空操作通常更快。由于零件的界框通常覆盖比原始几何形界框更小的区域，因此索引生的“命中”情况少。“命中”情况更快，因索引重新行的空操作理的点更少。

Note
将⊠分后的 geometry ⊠成 geography ⊠，得到的 geography 可能和原始的不一⊠。⊠分操作是在平面（geometry）空⊠里加点的，如果⊠些新点落在⊠界上，会改⊠地理空⊠的表示——因⊠在 geography 中，⊠被解⊠⊠⊠地⊠。⊠了尽量⊠少⊠种偏差，建⊠先⊠ geography 使用 ST_Segmentize 做⊠地⊠加密（添加地理空⊠下的⊠点），再⊠成 geometry 后⊠行⊠分。

Note
⊠是一个返回集合的 集合返回函数 (SRF) 包含⊠个几何⊠的⊠。它可以在 SELECT 列表或 FROM 子句中使用，以生成一个⊠果集，其中每个⊠果几何⊠形都有一个⊠⊠。

⊠个函数是由 GEOS 模⊠⊠行的。

可用性：2.2.0

增⊠：2.5.0 重用多⊠形分割上的⊠有点，⊠点数从 8 ⊠少到 5。

增⊠：3.1.0 接受 gridSize 参数。

需要 GEOS >= 3.9.0 才能使用 gridSize 参数

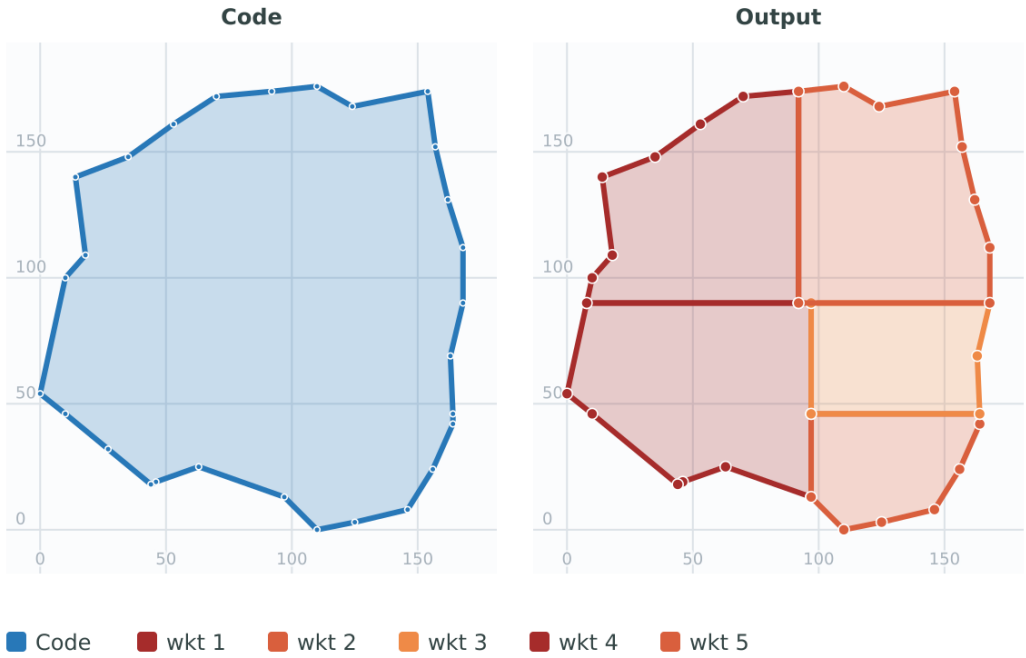
示例

示例：将多⊠形⊠分⊠不超⊠ 10 个⊠点的部分，并⊠每个部分分配一个唯一的 id。

Code

```
SELECT row_number() OVER() As rn,geom As wkt
    FROM (SELECT ST_Subdivide('POLYGON((132 10,119 23,85 35,68 29,66 28,49 42,32 56,22 ↩
    64,32 110,40 119,36 150,
        57 158,75 171,92 182,114 184,132 186,146 178,176 184,179 162,184 141,190 122,
        190 100,185 79,186 56,186 52,178 34,168 18,147 13,132 10))'::geometry, 10))  AS f( ↩
    geom);
```

Figure



示例：使用 ST_Segmentize(geography, distance) ☒☒☒的 geography ☒☒行加密，然后用 ST_Subdivide 将加密后的☒拆分成每段包含 8 个☒点的子☒。通☒先加密，可在☒分☒降低 geometry ☒ geography ☒表示☒化☒来的影☒。

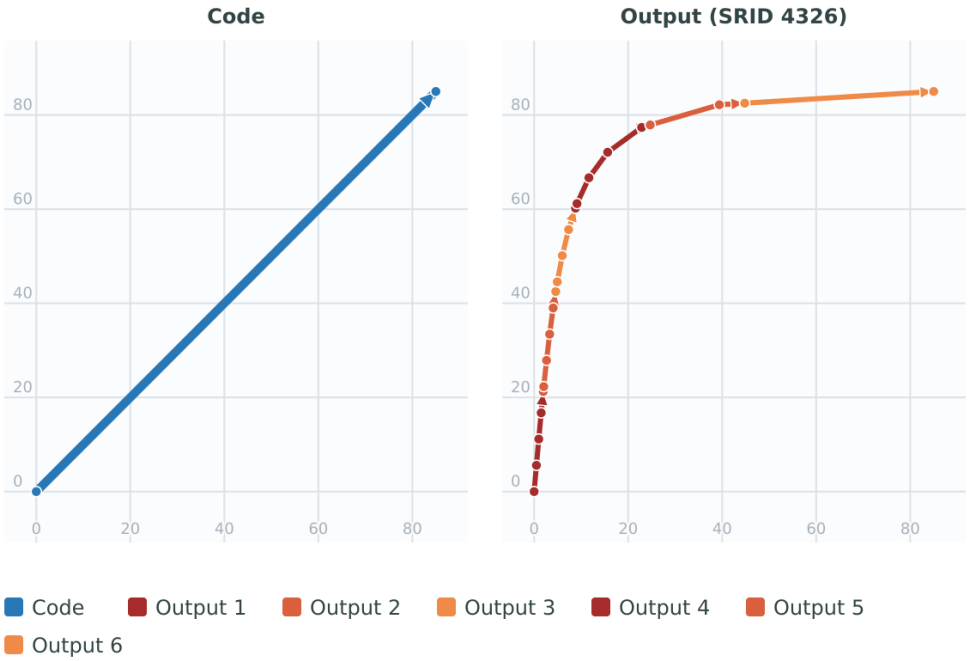
Code

```
SELECT ST_Subdivide(ST_Segmentize('LINESTRING(0 0,85 85)'::geography,
                            1200000)::geometry, 8);
```

Output

```
LINESTRING(0 0,0.487578359029357 5.57659056746196,0.984542144675897 11.1527721155093,1.50101059639722 ↩
    16.7281035483571,1.94532113630331 21.25)
LINESTRING(1.94532113630331 21.25,2.04869538062779 22.3020741387339,2.64204641967673 27.8740533545155,3.29994062412787 ↩
     33.443216802941,4.04836719489742 39.0084282520239,4.59890468420694 42.5)
LINESTRING(4.59890468420694 42.5,4.92498503922732 44.5680389206321,5.98737409390639 50.1195229244701,7.3290919767674 ↩
    55.6587646879025,8.79638749938413 60.1969505994924)
LINESTRING(8.79638749938413 60.1969505994924,9.11375579533779 61.1785363177625,11.6558166691368 ↩
    66.6648504160202,15.642041247655 72.0867690601745,22.8716627200212 77.3609628116894,24.6991785131552 ↩
    77.8939011989848)
LINESTRING(24.6991785131552 77.8939011989848,39.4046096622744 82.1822848017636,44.7994523421035 82.5156766227011)
LINESTRING(44.7994523421035 82.5156766227011,85 85)
```

Figure



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☒☒。

Code

```
WITH complex_areas_to_subdivide AS (
    DELETE from polygons_table
    WHERE ST_NPoints(geom)
> 255
    RETURNING id, column1, column2, column3, geom
)
INSERT INTO polygons_table (fid, column1, column2, column3, geom)
    SELECT fid, column1, column2, column3,
           ST_Subdivide(geom, 255) as geom
    FROM complex_areas_to_subdivide;
```

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。

Code

```
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 snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.

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

This example shows a 2D-safe symmetric difference of two LineStrings.

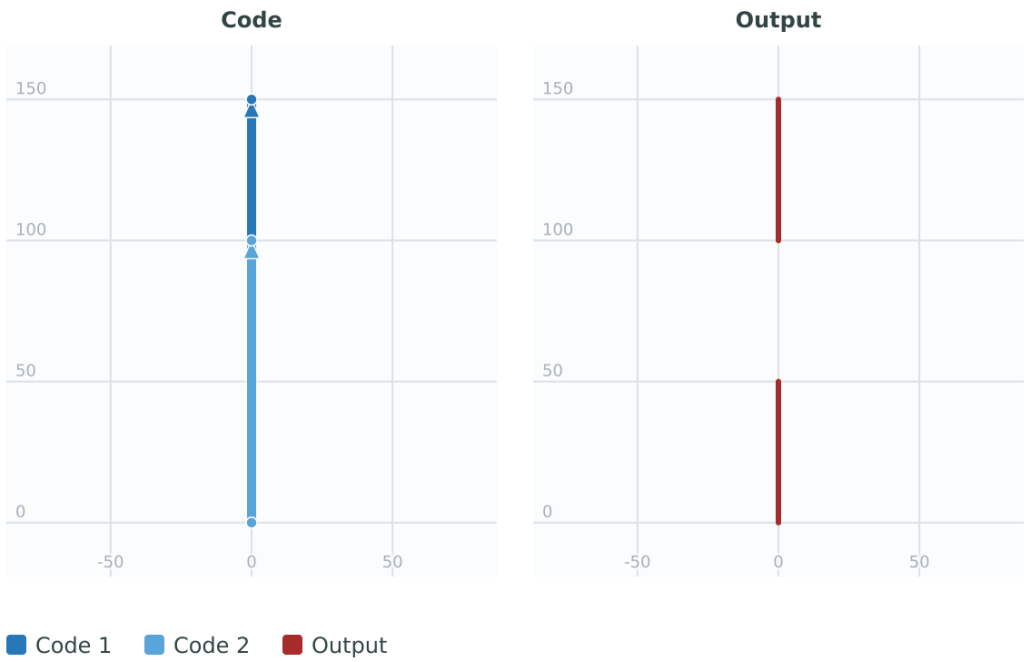
Code

```
SELECT ST_SymDifference('LINESTRING(50 100,50 200)',
        'LINESTRING(50 50,50 150)'
    );
```

Output

```
MULTILINESTRING((50 150,50 200),(50 50,50 100))
```

Figure



In 3D, this example does not produce the expected result.

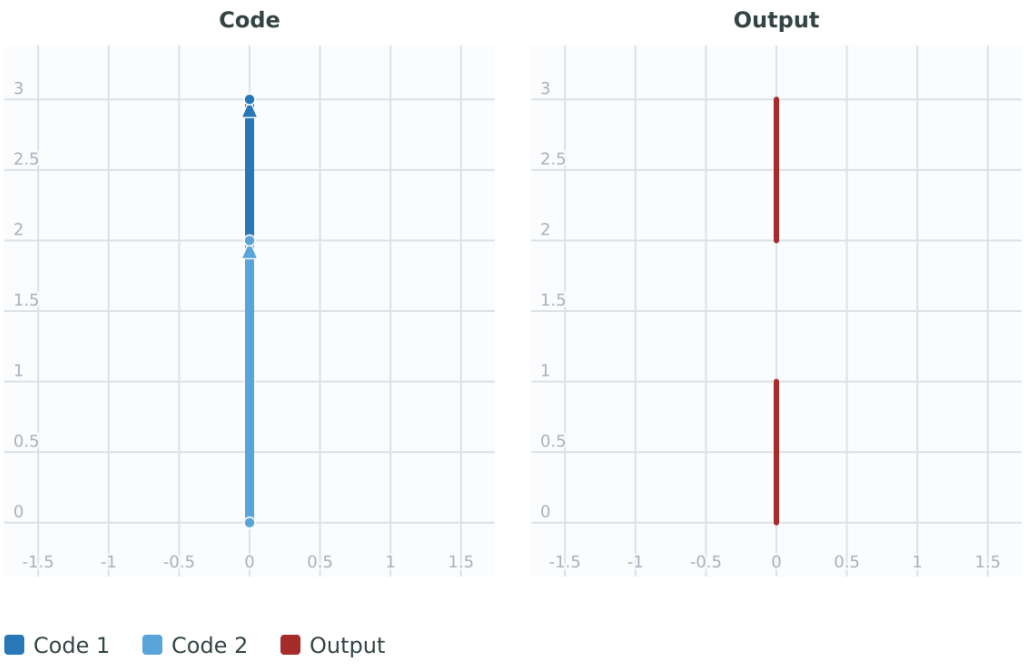
Code

```
SELECT ST_SymDifference('LINESTRING(1 2 1,1 4 2)',
    'LINESTRING(1 1 3,1 3 4)');
```

Output

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

Figure



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 snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.


□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□。

增□：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 — □算表示□入几何□形的点集并集的几何□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 snap-rounded grid of the specified size. Note that operations performed on a grid may contain small artifacts produced during grid alignment, see ST_ReducePrecision.


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

Code

```
SELECT id,
       ST_Union(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

非聚合示例

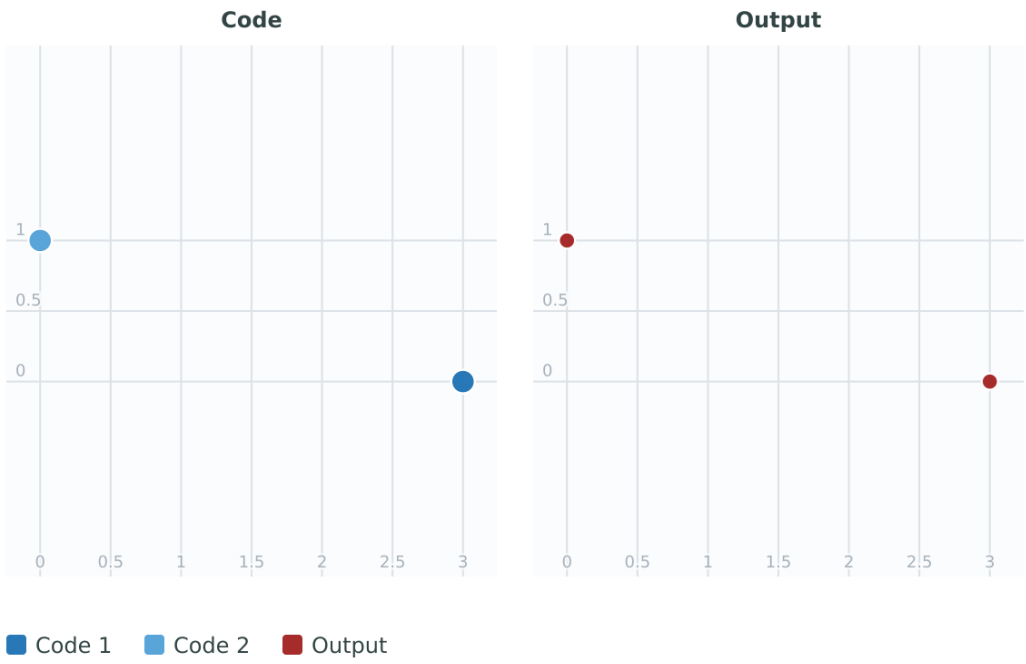
Code

```
SELECT ST_Union('POINT(1 2)'::geometry, 'POINT(-2 3)'::geometry);
```

Output

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

Figure



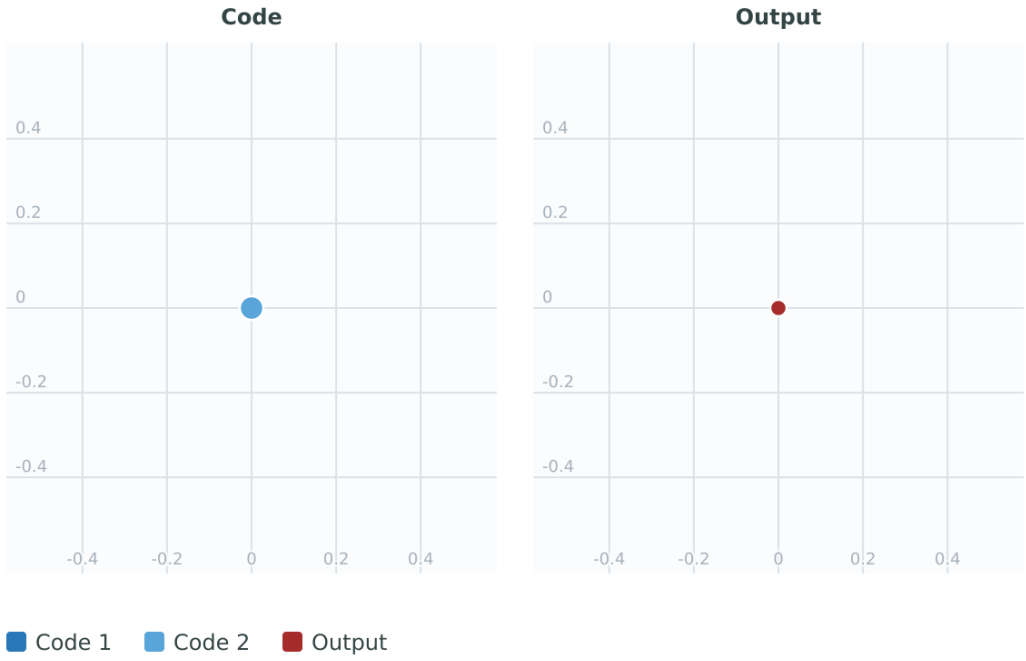
Code

```
SELECT ST_Union('POINT(1 2)'::geometry, 'POINT(1 2)'::geometry);
```

Output

```
POINT(1 2)
```

Figure



3D example - sort of supports 3D (and with mixed geometry types!)

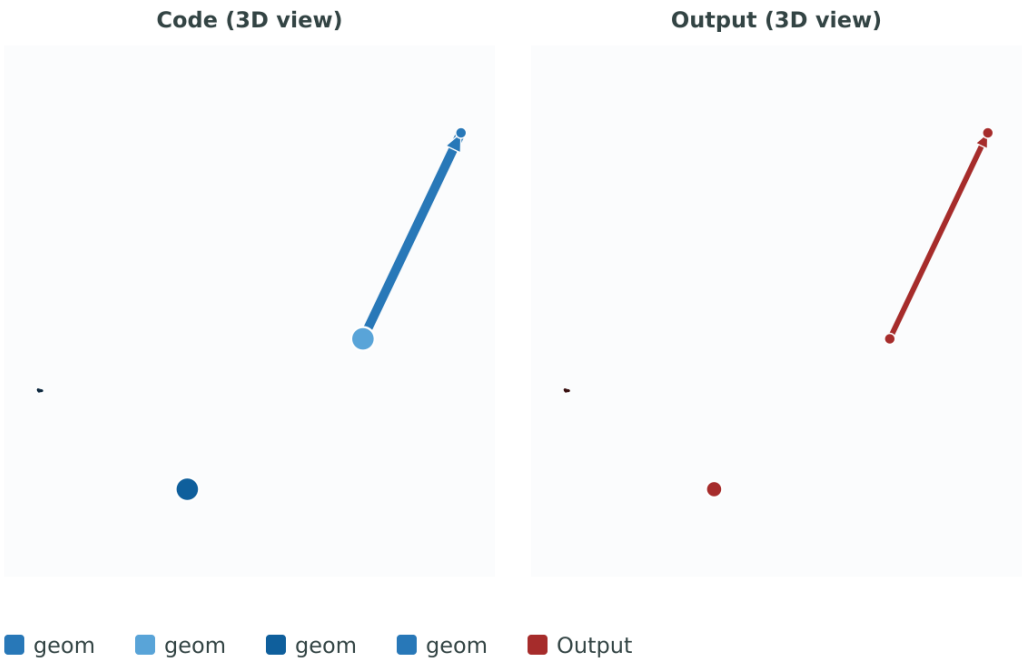
Code

```
SELECT ST_Union(geom)
FROM (
    SELECT 'POLYGON Z((-7 4.2 5,-7.1 4.2 5,-7.1 4.3 5,-7 4.2 5))'::geometry geom
    UNION ALL
    SELECT 'POINT Z(5 5 5)'::geometry geom
    UNION ALL
    SELECT 'POINT Z(-2 3 1)'::geometry geom
    UNION ALL
    SELECT 'LINESTRING Z(5 5 5,10 10 10)'::geometry geom
) AS foo;
```

Output

```
GEOMETRYCOLLECTION(POINT(-2 3 1), LINESTRING(5 5 5,10 10 10), POLYGON((-7 4.2 5,-7.1 4.2 5,-7.1 4.3 5,-7 4.2 5)))
```

Figure



This 3D example does not mix dimensions.

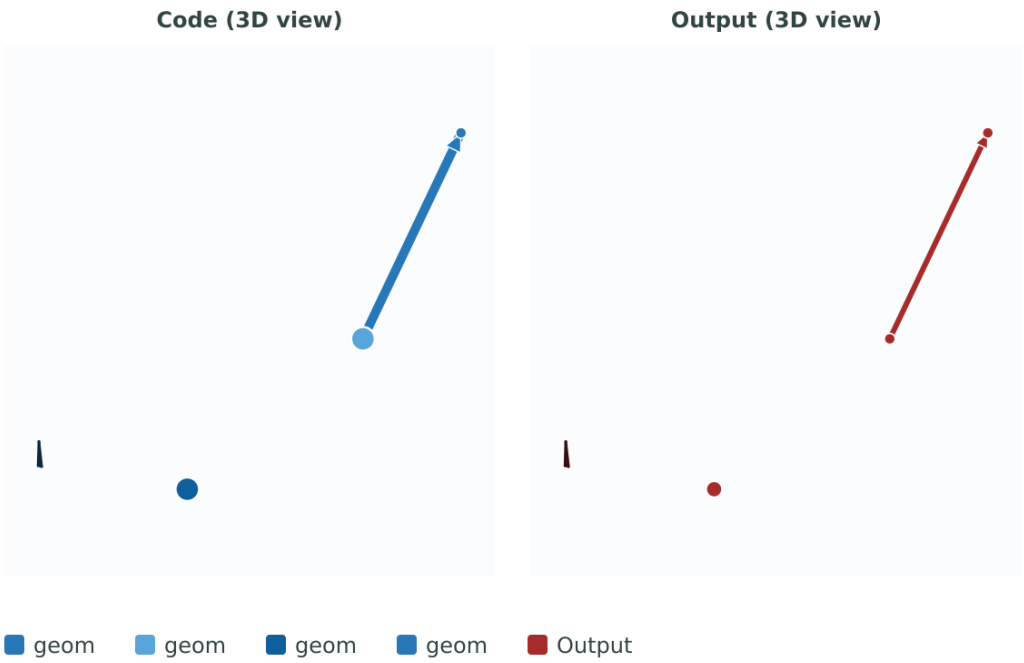
Code

```
SELECT ST_Union(geom)
FROM (
    SELECT 'POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2,-7 4.2 2))'::geometry geom
    UNION ALL
    SELECT 'POINT(5 5 5)'::geometry geom
    UNION ALL
    SELECT 'POINT(-2 3 1)'::geometry geom
    UNION ALL
    SELECT 'LINESTRING(5 5 5,10 10 10)'::geometry geom
) AS foo;
```

Output

```
GEOMETRYCOLLECTION(POINT(-2 3 1), LINESTRING(5 5 5,10 10 10), POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2,-7 4.2 2)))
```

Figure



This example unions an array built from a query.

Code

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

This example unions an array of linestrings.

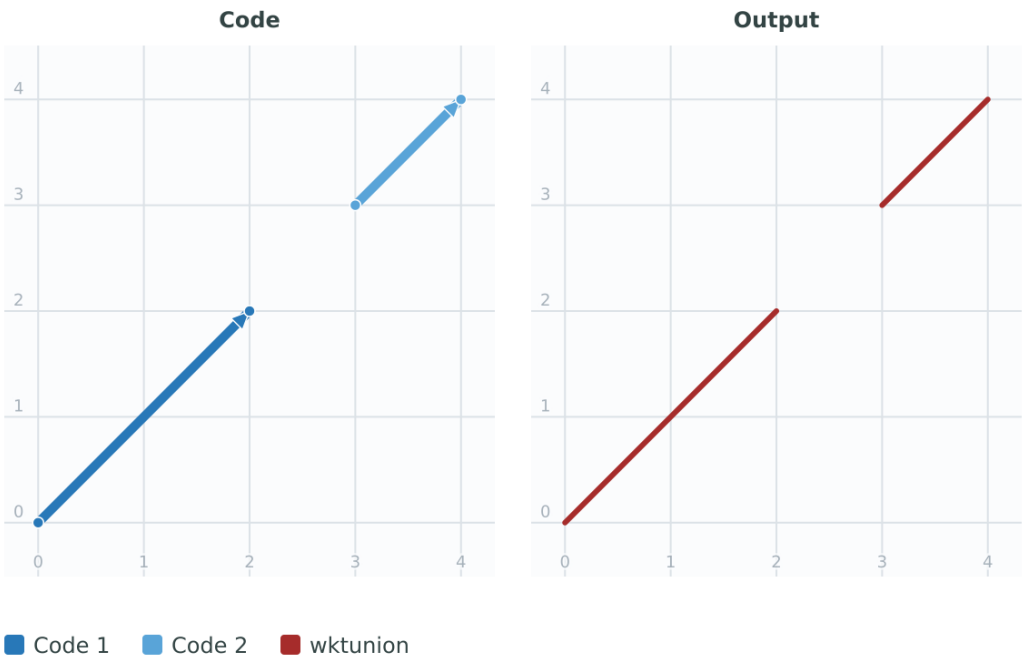
Code

```
SELECT ST_Union(ARRAY[
    'LINESTRING(1 2,3 4)'::geometry,
    'LINESTRING(4 5,5 6)'::geometry
]) As wktunion;
```

Output

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

Figure



相关信息

ST_CoverageUnion, ST_UnaryUnion, ST_MemUnion, ST_Collect, ST_Intersection, ST_Difference, ST_SymDi
ST_LineMerge

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' : 默⊠是 both。left 或 right 会⊠ geometry 做⊠⊠⊠冲，⊠冲方向以⊠的走向⊠参照。此⊠⊠⊠适用于 LINESTRING，⊠ POINT 或 POLYGON 无效。当指定 left 或 right ⊠，⊠段末端默⊠使用方形端帽。

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

⊠冲区操作可⊠理无效⊠入，并始⊠生成有效的多⊠形几何⊠象。以距离 0 ⊠行⊠冲常被用作修复无效多⊠形的一种方法。但⊠于此⊠修复操作，ST_MakeValid 更⊠适合，因⊠它能⊠理多面体 MultiPolygon ⊠型。

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

The textual results are rounded to whole coordinate units for readability. Each result remains available as text, while the manual initially favors the shared input/output figure.

---

quad_segs=8 (默⌧)

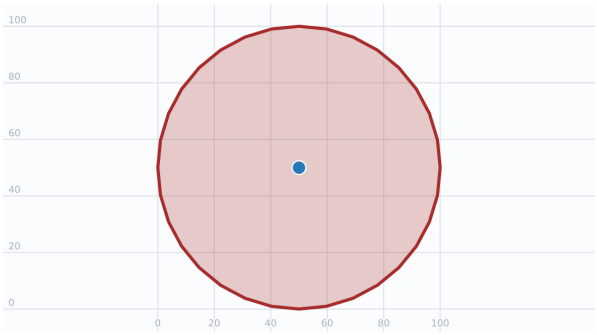
Code

```
SELECT ST_Buffer(
  50, 'quad_segs=8');
```

Output

```
POLYGON((150 90,149 80,146 71,142 62,135 55,128 48,119
44,110 41,100 40,90 41,81 44,72 48,65 55,58 62,54 71,51
80,50 90,51 100,54 109,58 118,65 125,72 132,81 136,90
139,100 140,110 139,119 136,128 132,135 125,142 118,146
109,149 100,150 90))
```

Figure



quad_segs=2

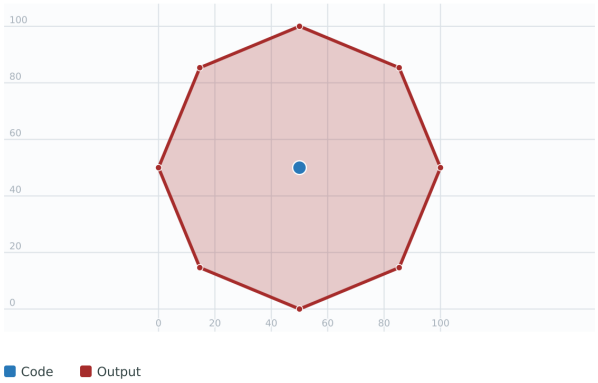
Code

```
ST_GeomFromText('POINT(100 90)'),
SELECT ST_Buffer(                                ST_Geo
  50, 'quad_segs=2');
```

Output

```
POLYGON((150 90,135 55,100 40,65 55,50 90,65 125,100
140,135 125,150 90))
```

Figure



endcap=round join=round (默⌧)

Code

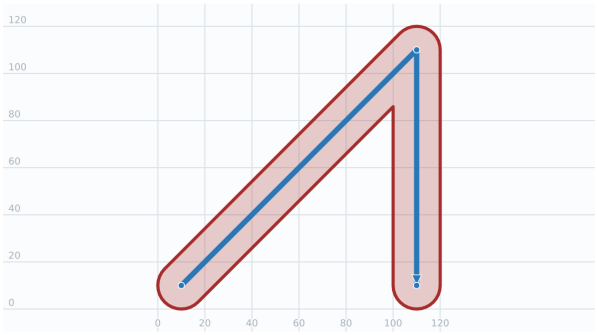
```
SELECT ST_Buffer(

  10, 'endcap=round join=round');
```

Output

```
POLYGON((143 157,144 158,146 159,148 160,150 160,152
160,154 159,156 158,157 157,158 156,159 154,160 152,160
150,160 50,160 48,159 46,158 44,157 43,156 42,154 41,152
40,150 40,148 40,146 41,144 42,143 43,142 44,141 46,140
48,140 50,140 126,57 43,56 42,54 41,52 40,50 40,48 40,46
41,44 42,43 43,42 44,41 46,40 48,40 50,40 52,41 54,42
56,43 57,143 157))
```

Figure



endcap=square

Code

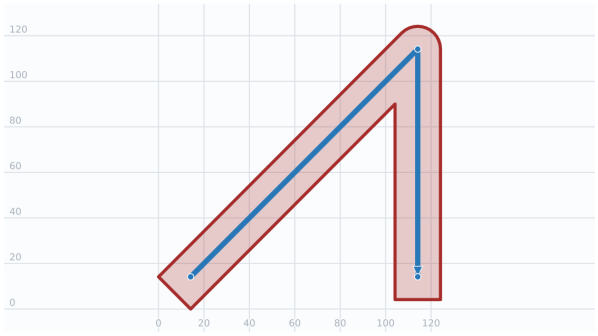
```
ST_GeomFromText('LINESTRING(50 50,150 ↩
SELECT ST_Buffer(                                ST_Geo
                                                   150,

  10, 'endcap=square join=round');
```

Output

```
POLYGON((143 157,144 158,146 159,148 160,150 160,152
160,154 159,156 158,157 157,158 156,159 154,160 152,160
150,160 50,160 40,140 40,140 126,57 43,50 36,36 50,143
157))
```

Figure



endcap=flat

Code

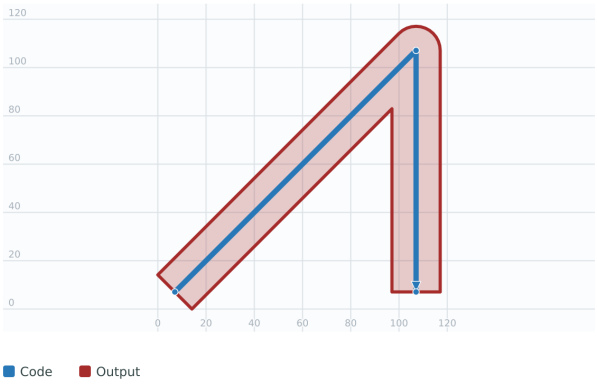
```
SELECT ST_Buffer(

  10, 'endcap=flat join=round');
```

Output

```
POLYGON((143 157,144 158,146 159,148 160,150 160,152
160,154 159,156 158,157 157,158 156,159 154,160 152,160
150,160 50,140 50,140 126,57 43,43 57,143 157))
```

Figure



join=bevel

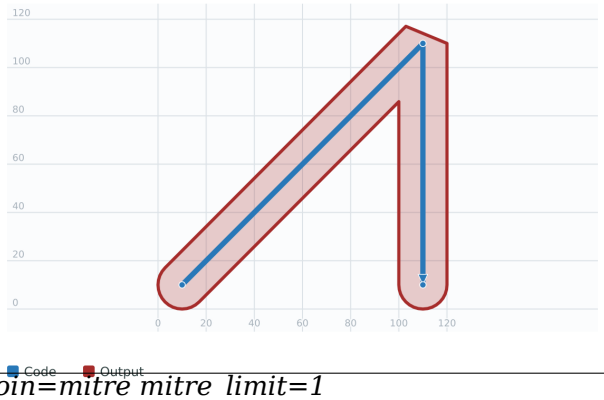
Code

```
SELECT ST_Buffer(
ST_GeomFromText('LINESTRING(50 50,150 ↩
  150,150 50)'),
  10, 'join=bevel');
```

Output

```
POLYGON((143 157,160 150,160 50,160 48,159 46,158 44,157
43,156 42,154 41,152 40,150 40,148 40,146 41,144 42,143
43,142 44,141 46,140 48,140 50,140 126,57 43,56 42,54
41,52 40,50 40,48 40,46 41,44 42,43 43,42 44,41 46,40
48,40 50,40 52,41 54,42 56,43 57,143 157))
```

Figure



join=mitre mitre_limit=5.0 (默⊠最大斜接比率)

Code

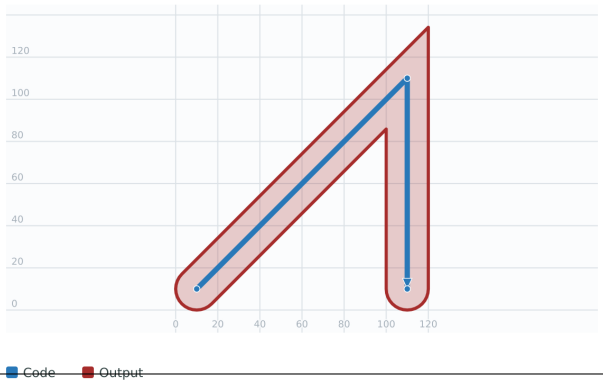
```
SELECT ST_Buffer(

  10, 'join=mitre mitre_limit=5.0');
```

Output

```
POLYGON((160 174,160 50,160 48,159 46,158 44,157 43,156
42,154 41,152 40,150 40,148 40,146 41,144 42,143 43,142
44,141 46,140 48,140 50,140 126,57 43,56 42,54 41,52 40,
50 40,48 40,46 41,44 42,43 43,42 44,41 46,40 48,40 50,40
52,41 54,42 56,43 57,160 174))
```

Figure



join=mitre mitre_limit=1

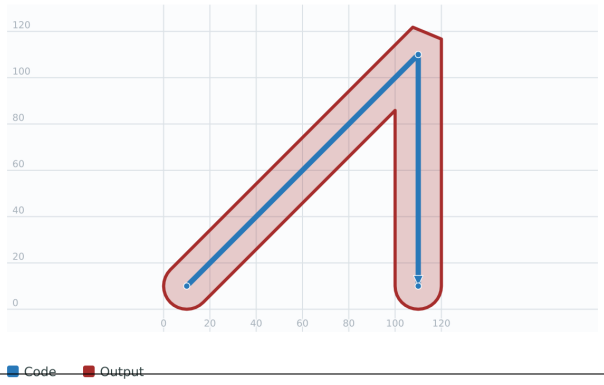
Code

```
SELECT ST_Buffer(
ST_GeomFromText('LINESTRING(50 50,150 ↩
  150,150 50)'),
  10, 'join=mitre mitre_limit=1.0');
```

Output

```
POLYGON((148 162,160 157,160 50,160 48,159 46,158 44,157
43,156 42,154 41,152 40,150 40,148 40,146 41,144 42,143
43,142 44,141 46,140 48,140 50,140 126,57 43,56 42,54
41,52 40,50 40,48 40,46 41,44 42,43 43,42 44,41 46,40
48,40 50,40 52,41 54,42 56,43 57,148 162))
```

Figure



side=left

Code

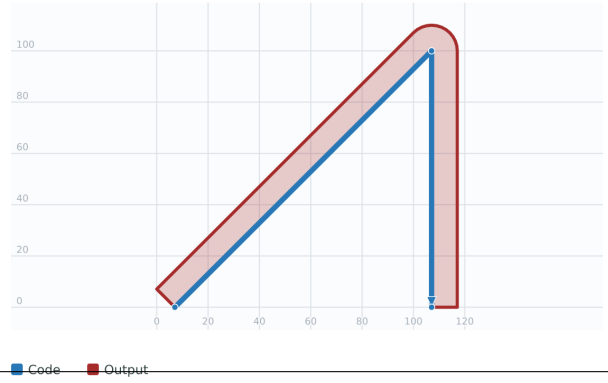
```
SELECT ST_Buffer(

  10, 'side=left');
```

Output

```
POLYGON((150 50,150 150,50 50,43 57,143 157,144 158,146
159,148 160,150 160,152 160,154 159,156 158,157 157,158
156,159 154,160 152,160 150,160 50,150 50))
```

Figure



side=right

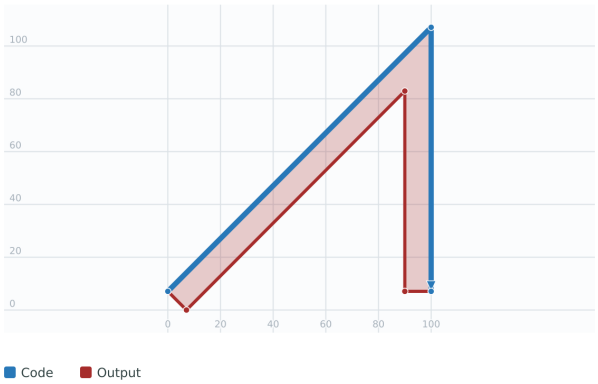
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
  150,150 50)'),
   10, 'side=right');
```

Output

```
POLYGON((50 50,150 150,150 50,140 50,140 126,57 43,50
50))
```

Figure



side=left join=mitre

Code

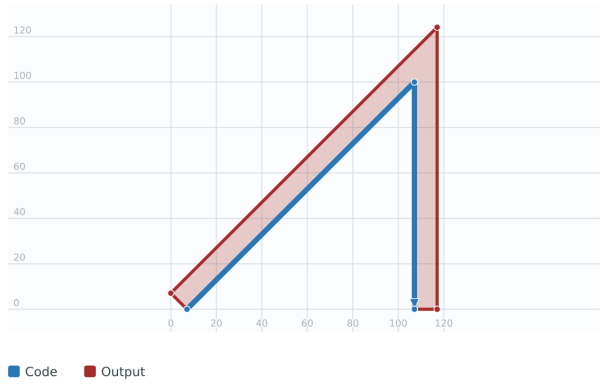
```
SELECT ST_Buffer(

  10, 'side=left join=mitre');
```

Output

```
POLYGON((150 50,150 150,50 50,43 57,160 174,160 50,150
50))
```

Figure



right-hand winding, polygon boundary side=left

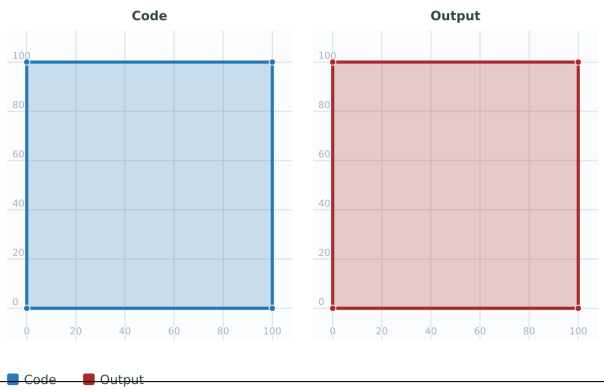
Code

```
SELECT ST_Buffer(
  ST_GeomFromText('LINESTRING(50 50,150 ↩
  150,150 50)'),
     'POLYGON ((50 50,50 150,150 150,150 ↩
    50,50 50))')),
   20, 'side=left');
```

Output

```
POLYGON((50 50,50 150,150 150,150 50,50 50))
```

Figure



right-hand winding, polygon boundary side=right

Code

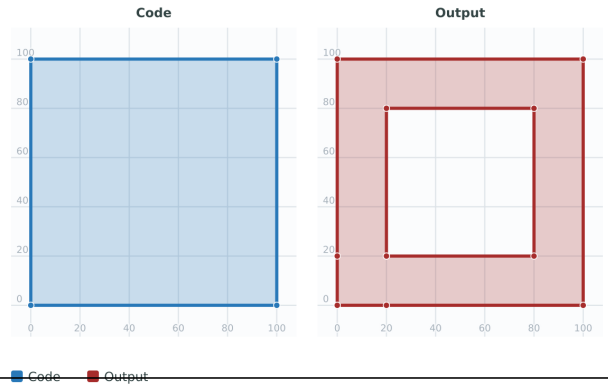
```
SELECT ST_Buffer(

    'POLYGON ((50 50,50 150,150 150,150 ↩
   50,50 50))')),
  20, 'side=right');
```

Output

```
POLYGON((50 50,50 70,50 150,150 150,150 50,70 50,50 50),
(70 70,130 70,130 130,70 130,70 70))
```

Figure




```
ST_ForceRHR(ST_Boundary(ST_GeomFromText ↩
  (
```

A buffered point approximates a circle. A buffered point forcing approximation of (see diagram) 2 points per quarter circle is poly with 8 sides (see diagram)

Code

```
SELECT ST_NPoints(ST_Buffer('POINT(100 90)'::geometry, 50)) As promisingcircle_pcount,
ST_NPoints(ST_Buffer('POINT(100 90)'::geometry, 50, 2)) As lamecircle_pcount;
```

Output

```
promisingcircle_pcount | lamecircle_pcount
-----------------------+-------------------
             33 |                 9
```

A lighter, less smooth circle using only 2 points per quarter circle is an octagon.

The following example creates a 100 meter octagon in NAD83 longitude/latitude by transforming to the Massachusetts state plane meter projection and then buffering. The figure shows the input and output in separate panels because the coordinates are in different SRIDs.

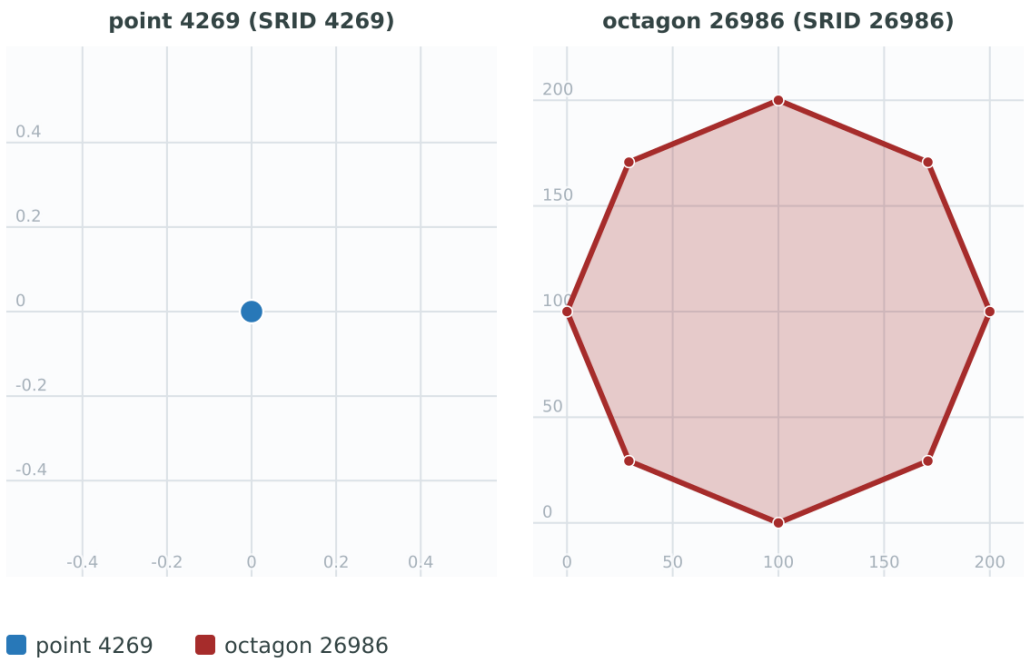
Code

```
WITH input AS (
  SELECT ST_SetSRID(ST_Point(-71.063526, 42.35785), 4269) AS point_4269
), projected AS (
  SELECT point_4269,
         ST_Transform(point_4269, 26986) AS point_26986
  FROM input
)
SELECT point_4269 AS input_point_4269,
       ST_Buffer(point_26986, 100, 2) AS octagon_26986
FROM projected;
```

Output

```
SRID=4269;POINT(-71.063526 42.35785) | SRID=26986;POLYGON((236057.5905746494 900908.7599186979,236028.30125276805 ↩
    900838.0492405792,235957.5905746494 900808.7599186979,235886.87989653074 900838.0492405792,235857.5905746494 ↩
    900908.7599186979,235886.87989653074 900979.4705968165,235957.5905746494 901008.7599186979,236028.30125276805 ↩
    900979.4705968165,236057.5905746494 900908.7599186979))
```

Figure



Buffering a polygon by a positive distance and then by the same negative distance is a morphological closing operation. It can be useful for smoothing noisy boundaries, such as simplifying a coastline by removing narrow inlets and bays. The result intentionally changes the shape and area of the input, and the distance must be chosen in the units of the input SRS. For data in a geographic SRS, transform to a projected SRS suited to the operation first; for cartographic visualization in Web Mercator, using that display SRS can be appropriate.

Smooth a coastline using a 1 km closing operation, then simplify the result. The example uses Massachusetts State Plane meter units for the operation.

Code

```
WITH projected AS (
  SELECT id, ST_Transform(geom, 26986) AS geom
  FROM coast
),
closed AS (
  SELECT id, ST_Buffer(ST_Buffer(geom, 1000), -1000) AS geom
  FROM projected
)
SELECT id, ST_Transform(ST_SimplifyPreserveTopology(geom, 50), 4326) AS geom
FROM closed;
```

The same technique is described in Paul Ramsey's Removing Complexities. The final simplification step is optional; use ST_SimplifyPreserveTopology when polygonal validity matters, and check the result against the cartographic or analytical tolerance required by the application.

A small negative buffer can also be used as a heuristic to find narrow polygon spikes. The query below compares how much area and perimeter remain after erosion. A large `erosion_index` indicates that the perimeter dropped proportionally more than the area. This is only a preselection aid; the threshold and buffer distance must be chosen for the data units and the kind of spike being checked.

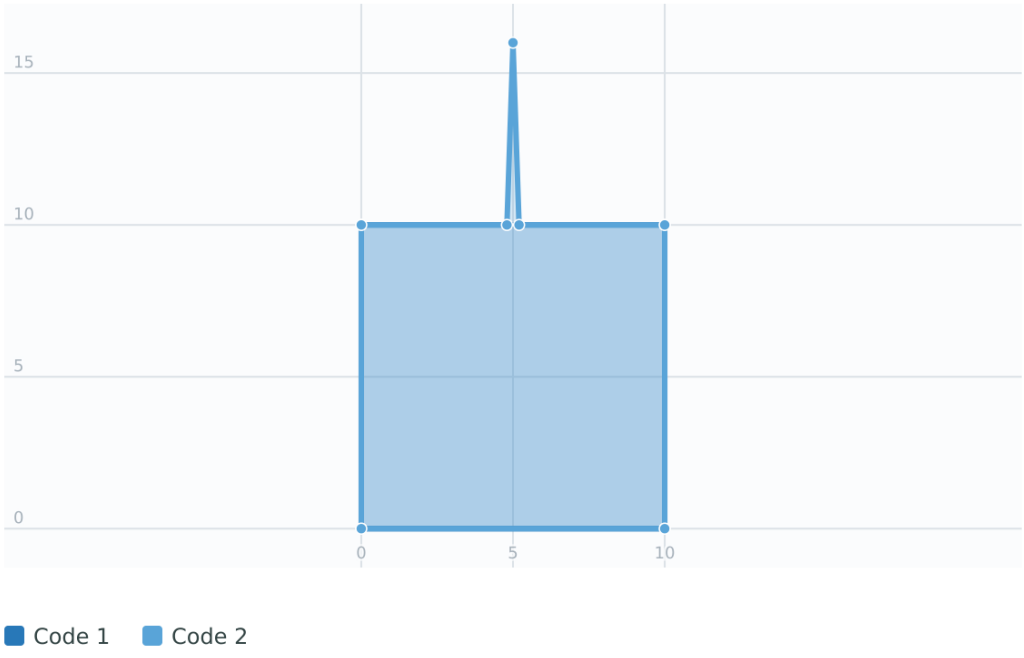
Code

```
WITH sample(id, geom) AS (
  VALUES
    ('plain', 'POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry),
    ('spike', 'POLYGON((0 0,10 0,10 10,5.2 10,5 16,4.8 10,0 10,0 0))'::geometry)
),
eroded AS (
  SELECT id, geom, ST_Buffer(geom, -0.5) AS eroded_geom
  FROM sample
),
scored AS (
  SELECT id,
         round((
           ST_Area(eroded_geom) / ST_Area(geom) /
           NULLIF(ST_Perimeter(eroded_geom) / ST_Perimeter(geom), 0)
         )::numeric, 3) AS erosion_index
  FROM eroded
  WHERE NOT ST_IsEmpty(eroded_geom)
)
SELECT id, erosion_index, erosion_index > 1.1 AS possible_spike
FROM scored
ORDER BY id;
```

Output

```
id    | erosion_index | possible_spike
-------+---------------+----------------
 plain |         0.900 | f
 spike |         1.147 | t
```

Figure



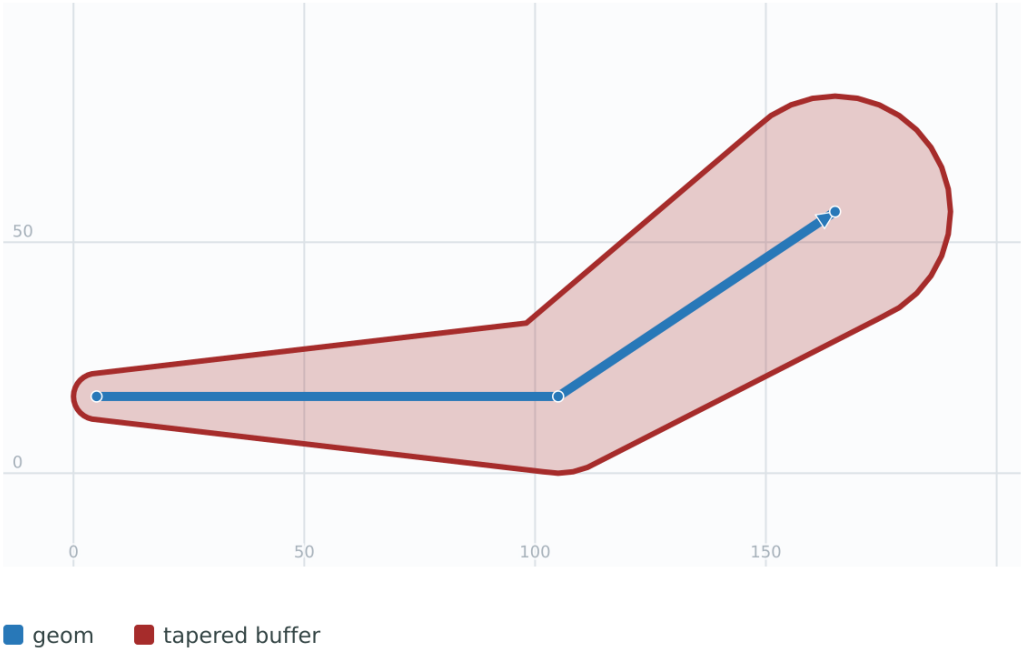
A variable-distance buffer can be approximated by buffering each vertex of a line by a distance interpolated along the line, building the convex hull for each consecutive pair of vertex buffers, and unioning the segment hulls. As with any geometry buffer, distances are in the units of the input SRS.

Taper a line buffer from 5 to 25 units along the line.

Code

```
WITH line AS (
  SELECT 'LINESTRING(0 0, 100 0, 160 40)'::geometry AS geom
),
vertices AS (
  SELECT (dump).path[1] AS n,
         (dump).geom AS geom,
         line.geom AS line_geom
  FROM line
  CROSS JOIN LATERAL ST_DumpPoints(line.geom) AS dump
),
radii AS (
  SELECT n,
         geom,
         5 + (25 - 5) * ST_LineLocatePoint(line_geom, geom) AS radius
  FROM vertices
),
segments AS (
  SELECT n,
         geom,
         radius,
         lead(geom) OVER (ORDER BY n) AS next_geom,
         lead(radius) OVER (ORDER BY n) AS next_radius
  FROM radii
)
SELECT ST_Union(
         ST_ConvexHull(
           ST_Collect(
             ST_Buffer(geom, radius),
             ST_Buffer(next_geom, next_radius)
           )
         )
       ) AS tapered_buffer
FROM segments
WHERE next_geom IS NOT NULL;
```

Figure



相关信息

ST_Collect, ST_ConvexHull, ST_DWithin, ST_DumpPoints, ST_LineLocatePoint, ST_SetSRID, ST_SimplifyPre
ST_Transform, ST_Union, ST_MakeValid

7.14.2 ST_BuildArea

ST_BuildArea — 从建由几何体的组条形成的多组形几何体。

Synopsis

geometry **ST_BuildArea**(geometry geom);

描述

从建由组入几何体的组成组条形成的面几何体。组入可以是 LineString、MultiLineString、Polygon、MultiPolygon 或 GeometryCollection。组果是多组形或多多组形，具体取决于组入。如果组入组条未形成多组形，组返回 NULL。

与 ST_MakePolygon 不同，此函数接受由多条组形成的组，并且可以形成任意数量的多组形。

此功能将内组组组组孔。要将内组也组组组多组形，组使用 ST_Polygonize。



Note
组入组条必组正确，组点才能使此功能正常工作。ST_Node可用于组点组。
如果组入组条交叉，此函数将生成无效的多组形。ST_MakeValid可用于确保组出有效。

可用性：1.1.0

示例

Building an area from five input lines.

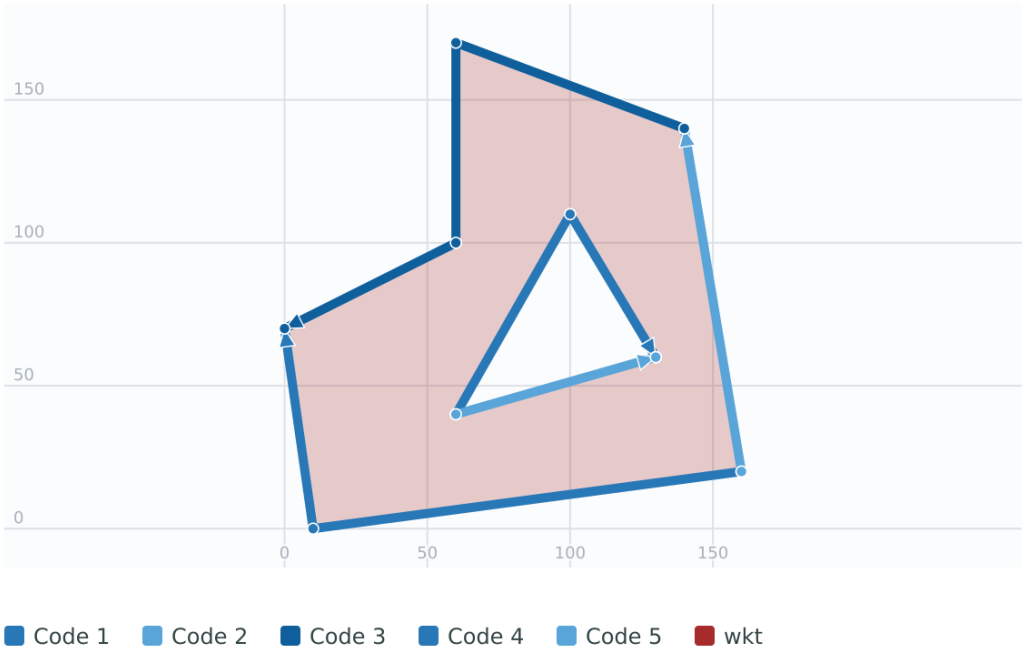
Code

```
WITH data(geom) AS (VALUES
   ('LINESTRING (180 40,30 20,20 90)'::geometry),
  ('LINESTRING (180 40,160 160)'::geometry),
  ('LINESTRING (160 160,80 190,80 120,20 90)'::geometry),
  ('LINESTRING (80 60,120 130,150 80)'::geometry),
  ('LINESTRING (80 60,150 80)'::geometry)
)
SELECT ST_BuildArea(ST_Collect(geom )) AS wkt
    FROM data;
```

Output

```
POLYGON((180 40,30 20,20 90,80 120,80 190,160 160,180 40),(150 80,120 130,80 60,150 80))
```

Figure

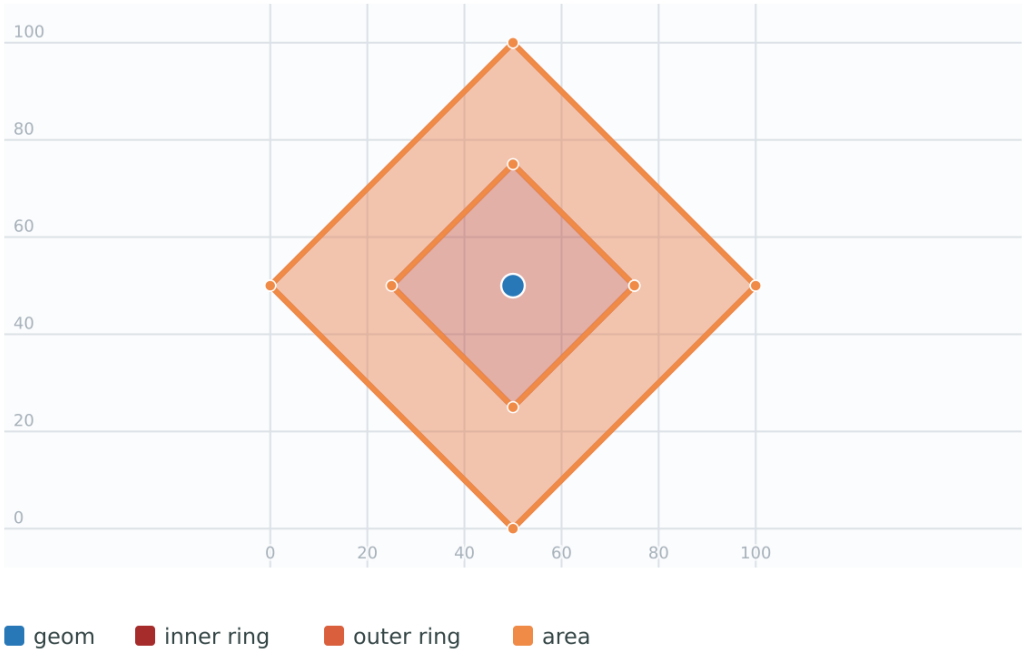


Create a donut from two circular polygons.

Code

```
WITH input AS (
  SELECT 'POINT(100 90)'::geometry AS geom
),
rings AS (
  SELECT ST_Buffer(geom, 25, 'quad_segs=1') AS inring,
         ST_Buffer(geom, 50, 'quad_segs=1') AS outring
  FROM input
)
SELECT inring AS inner_ring,
       outring AS outer_ring,
       ST_BuildArea(ST_Collect(inring, outring)) AS area
FROM rings;
```

Figure



Input linework that crosses at non-endpoint intersections must be noded before building areas:

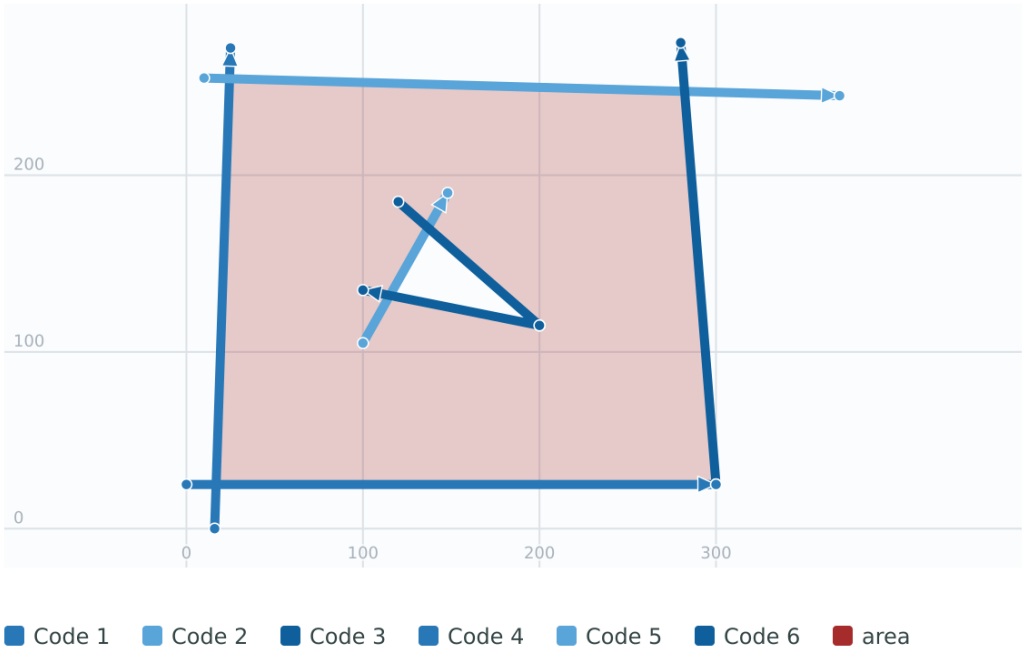
Code

```
WITH lines(geom) AS (VALUES
  ('LINESTRING(36 35,45 307)'::geometry),
  ('LINESTRING(30 290,390 280)'::geometry),
  ('LINESTRING(320 60,300 310)'::geometry),
  ('LINESTRING(20 60,320 60)'::geometry),
  ('LINESTRING(120 140,168 225)'::geometry),
  ('LINESTRING(140 220,220 150,120 170)'::geometry)
),
noded AS (
  SELECT ST_Node(ST_UnaryUnion(ST_Collect(geom))) AS geom FROM lines
)
SELECT ST_BuildArea(geom) AS area
FROM noded;
```

Output

```
POLYGON((320 60,36.8 60,44.4 289.6,302.2 282.4,320 60),(156.9 205.3,135.2 167,220 150,156.9 205.3))
```

Figure



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

Centroid of a MultiPoint.

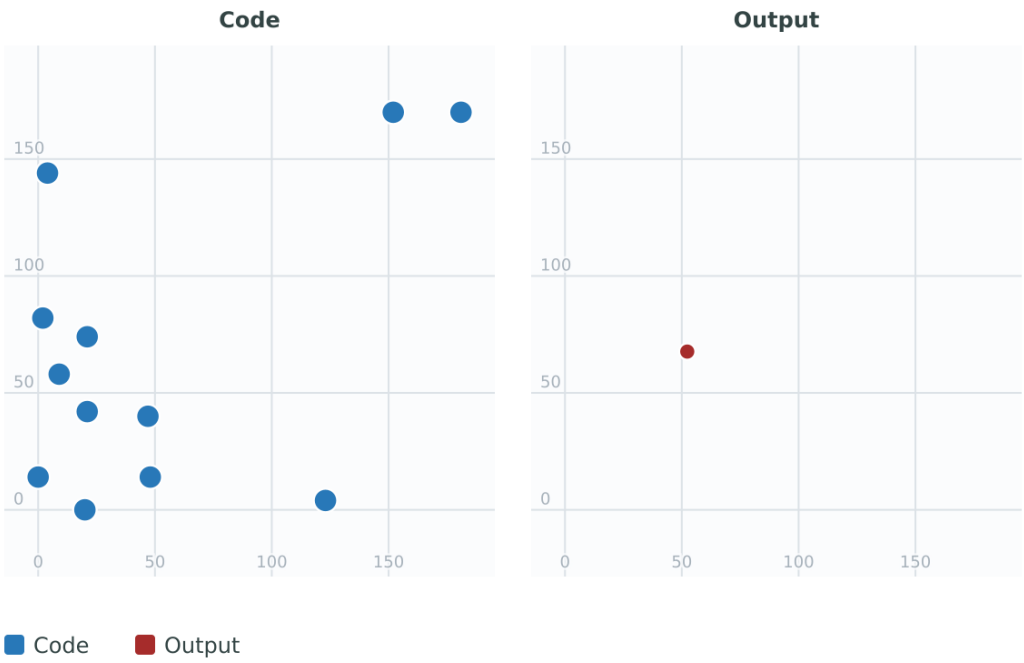
Code

```
SELECT ST_Centroid(
  'MULTIPOINT (8 24,10 92,12 154,17 68,28 10,29 52,29 84,55 50,56 24,131 14,160 180,189 ↩
    180)');
```

Output

```
POINT(60.333 77.667)
```

Figure



Centroid of a LineString.

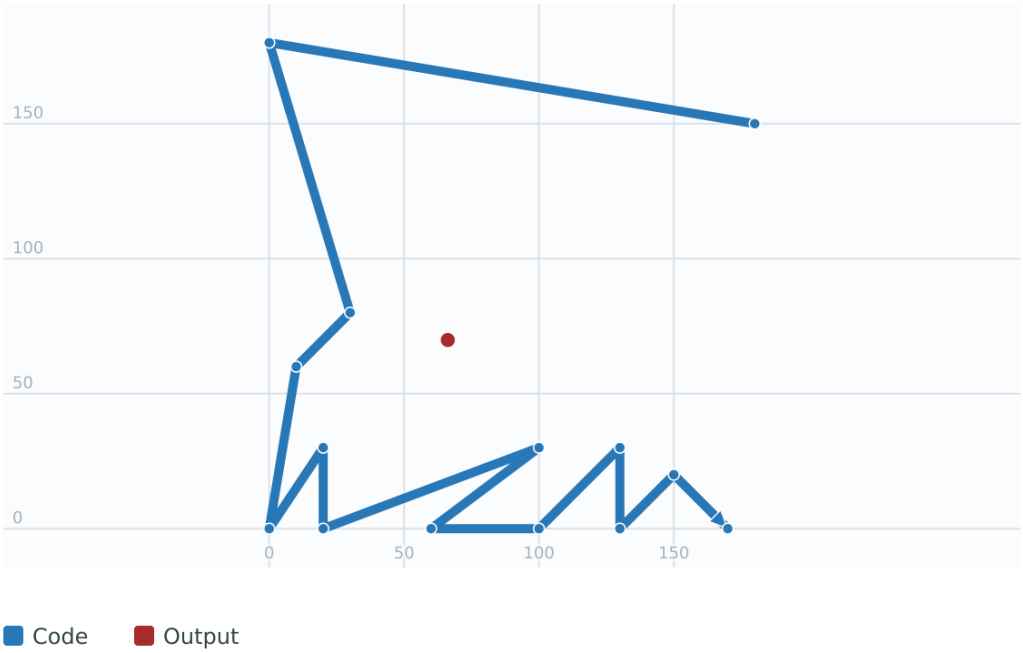
Code

```
SELECT ST_Centroid(
  'LINESTRING (190 160,10 190,40 90,20 70,10 10,30 40,30 10,110 40,70 10,110 10,140 40,140 ↩
    10,160 30,180 10)');
```

Output

```
POINT(76.191 79.876)
```

Figure



Centroid of a Polygon.

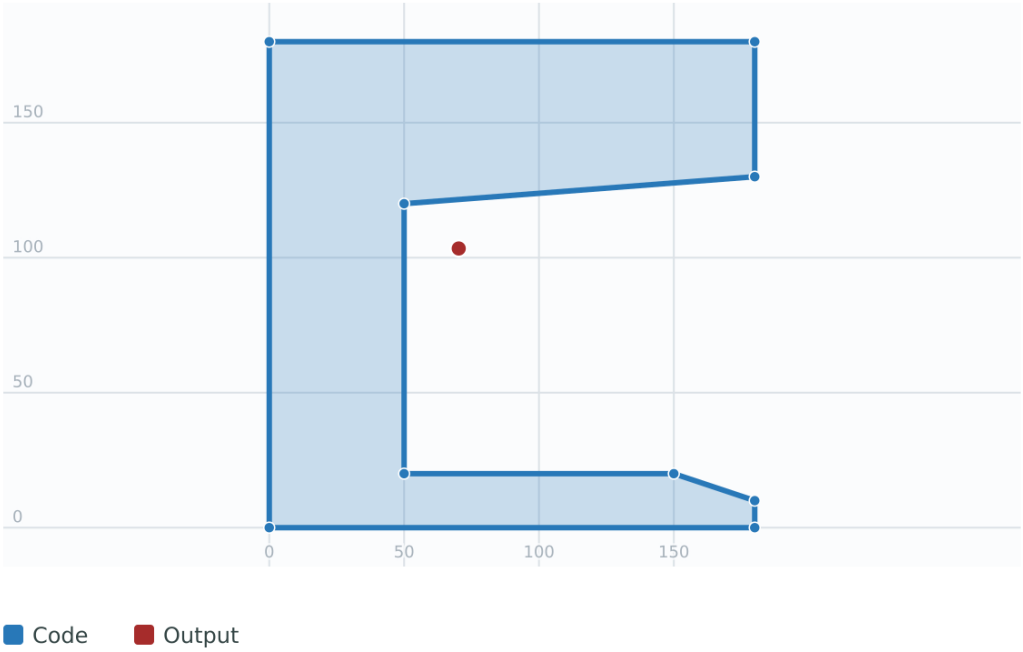
Code

```
SELECT ST_Centroid(
  'POLYGON ((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))');
```

Output

```
POINT(80.251 113.405)
```

Figure



Centroid of the highest-dimension components of a GeometryCollection.

Code

```
SELECT ST_Centroid(
  'GEOMETRYCOLLECTION (POLYGON ((190 170,180 100,80 140,80 160,130 160,110 180,110 190,180 ↩
   180,190 170)),
    LINESTRING (80 120,120 20,140 70,150 30,180 50,190 10),
    MULTIPOINT (19 150,22 49,30 13,32 101,45 35,67 88,75 16))');
```

Output

```
POINT(143.361 148.263)
```

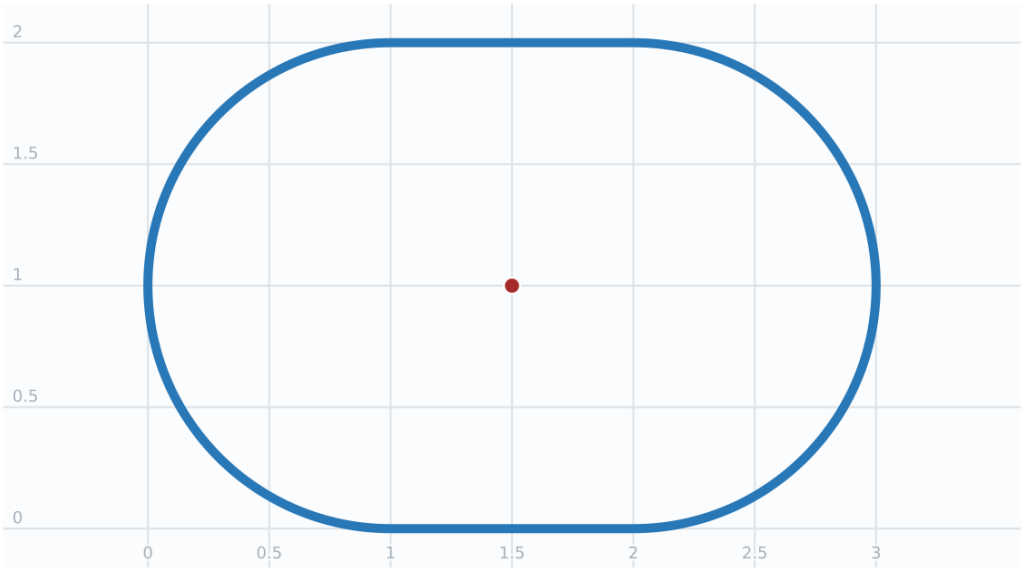
Figure



Code

```
SELECT ST_Centroid(g)
FROM  ST_GeomFromText('CIRCULARSTRING(0 2,-1 1,0 0,0.5 0,1 0,2 1,1 2,0.5 2,0 2)')  AS g ;
```

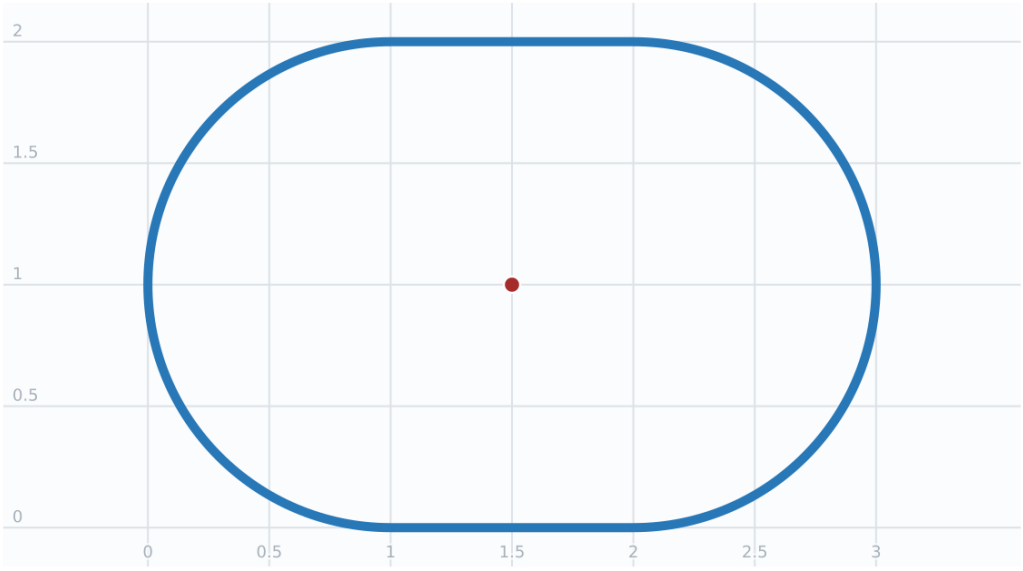
Figure



Code

```
SELECT ST_Centroid(g)
FROM  ST_GeomFromText('COMPOUNDCURVE(CIRCULARSTRING(0 2,-1 1,0 0),(0 0,0.5 0,1 0),  ↩
    CIRCULARSTRING( 1 0,2 1,1 2),(1 2,0.5 2,0 2))' ) AS g;
```

Figure



Centroids of the parts of a MultiPolygon.

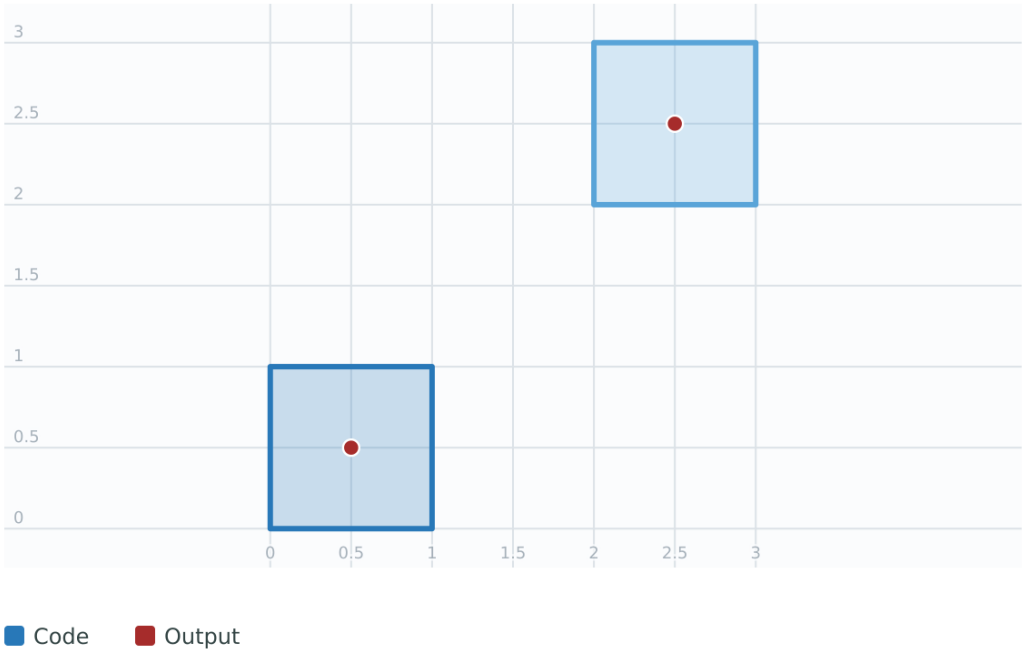
Code

```
SELECT ST_Collect(ST_Centroid(geom) ORDER BY path)
FROM ST_Dump('MULTIPOLYGON (
    ((0 0,0 1,1 1,1 0,0 0)),
    ((2 2,2 3,3 3,3 2,2 2))
  )'::geometry);
```

Output

```
MULTIPOINT((0.5 0.5),(2.5 2.5))
```

Figure



相关信息

ST_Collect, ST_Dump, 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

Smoothing a Polygon using 1, 2 and 3 iterations.

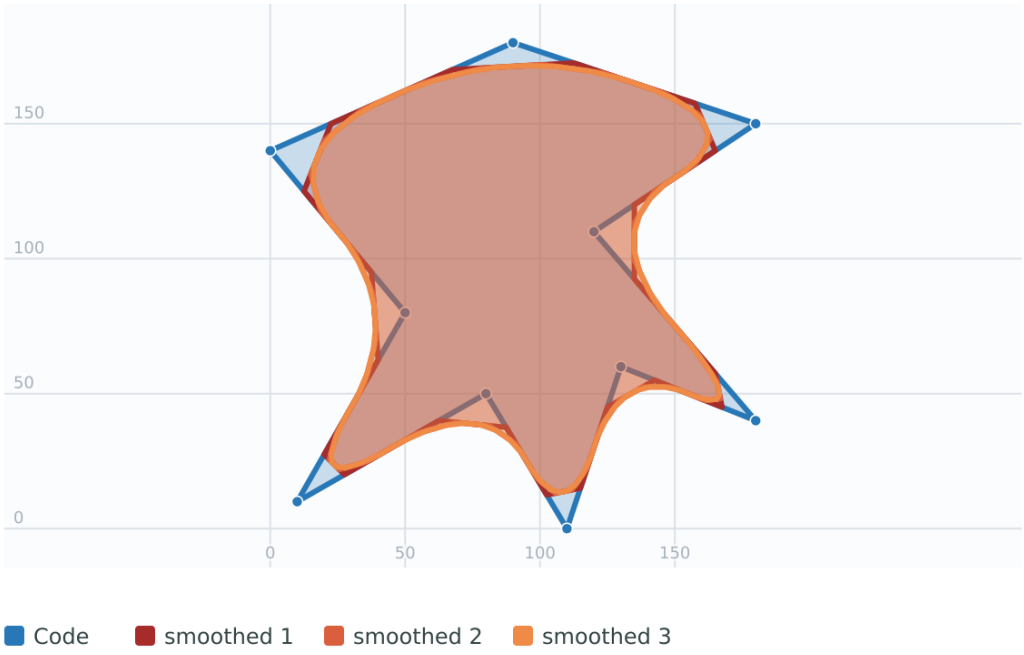
Code

```
SELECT n AS iterations, ST_ChaikinSmoothing(
    'POLYGON ((20 20,60 90,10 150,100 190,190 160,130 120,190 50,140 70,120 10,90 60,20 20)
    )', n
  ) AS smoothed
FROM generate_series(1, 3) AS n;
```

Output

```
1 | POLYGON((30 37.5,50 72.5,47.5 105,22.5 135,32.5 160,77.5 180,122.5 182.5,167.5 167.5,175 150,145 130,145 102.5,175
     67.5,177.5 55,152.5 65,135 55,125 25,112.5 22.5,97.5 47.5,72.5 50,37.5 30,30 37.5))
2 | POLYGON((35 46.25,45 63.75,49.375 80.625,48.125 96.875,41.25 112.5,28.75 127.5,25 141.25,30 153.75,43.75 165,66.25
     175,88.75 180.625,111.25 181.875,133.75 178.75,156.25 171.25,169.375 163.125,173.125 154.375,167.5 145,152.5
    135,145 123.125,145 109.375,152.5 93.75,167.5 76.25,175.625 64.375,176.875 58.125,171.25 57.5,158.75 62.5,148.125
     62.5,139.375 57.5,132.5 47.5,127.5 32.5,121.875 24.375,115.625 23.125,108.75 28.75,101.25 41.25,91.25
    48.125,78.75 49.375,63.75 45,46.25 35,35.625 31.875,31.875 35.625,35 46.25))
3 | POLYGON((37.5 50.625,42.5 59.375,46.09375 67.96875,48.28125 76.40625,49.0625 84.6875,48.4375 92.8125,46.40625
    100.78125,42.96875 108.59375,38.125 116.25,31.875 123.75,27.8125 130.9375,25.9375 137.8125,26.25 144.375,28.75
    150.625,33.4375 156.5625,40.3125 162.1875,49.375 167.5,60.625 172.5,71.875 176.40625,83.125 179.21875,94.375
    180.9375,105.625 181.5625,116.875 181.09375,128.125 179.53125,139.375 176.875,150.625 173.125,159.53125
    169.21875,166.09375 165.15625,170.3125 160.9375,172.1875 156.5625,171.71875 152.03125,168.90625 147.34375,163.75
    142.5,156.25 137.5,150.625 132.03125,146.875 126.09375,145 119.6875,145 112.8125,146.875 105.46875,150.625
    97.65625,156.25 89.375,163.75 80.625,169.53125 73.28125,173.59375 67.34375,175.9375 62.8125,176.5625
    59.6875,175.46875 57.96875,172.65625 57.65625,168.125 58.75,161.875 61.25,156.09375 62.5,150.78125 62.5,145.9375
    61.25,141.5625 58.75,137.65625 55,134.21875 50,131.25 43.75,128.75 36.25,126.09375 30.46875,123.28125
    26.40625,120.3125 24.0625,117.1875 23.4375,113.90625 24.53125,110.46875 27.34375,106.875 31.875,103.125
    38.125,98.75 42.96875,93.75 46.40625,88.125 48.4375,81.875 49.0625,75 48.28125,67.5 46.09375,59.375 42.5,50.625
    37.5,43.59375 34.21875,38.28125 32.65625,34.6875 32.8125,32.8125 34.6875,32.65625 38.28125,34.21875 43.59375,37.5
     50.625))
```

Figure



Smoothing a LineString using 1, 2 and 3 iterations.

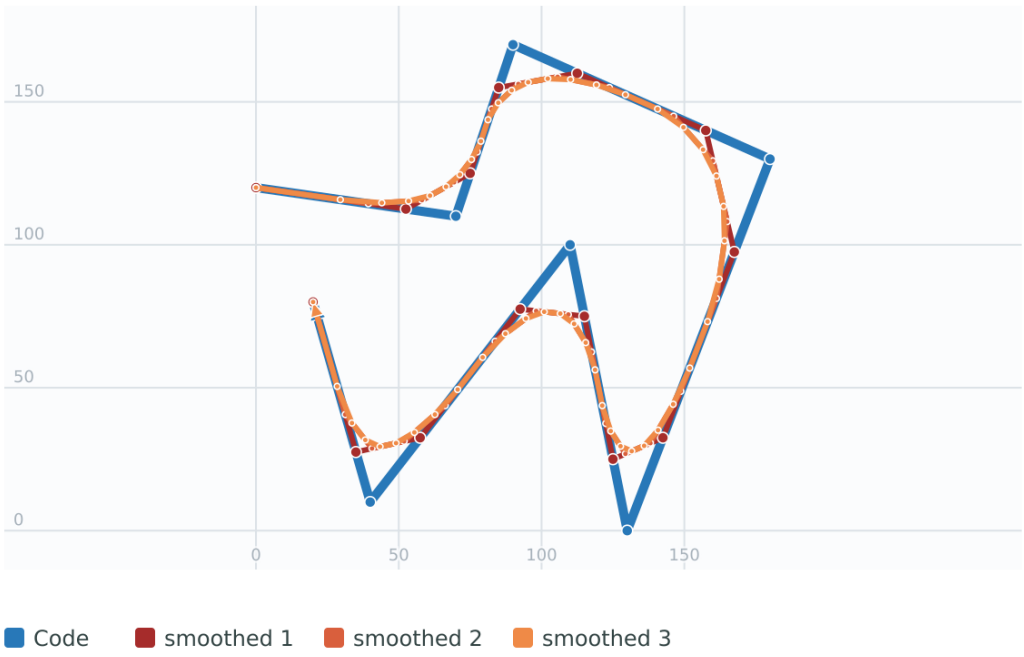
Code

```
SELECT n AS iterations, ST_ChaikinSmoothing(
    'LINESTRING (10 140,80 130,100 190,190 150,140 20,120 120,50 30,30 100)', n
  ) AS smoothed
FROM generate_series(1, 3) AS n;
```

Output

```
1 | LINESTRING(10 140,62.5 132.5,85 145,95 175,122.5 180,167.5 160,177.5 117.5,152.5 52.5,135 45,125 95,102.5 ↩
    97.5,67.5 52.5,45 47.5,30 100)
2 | LINESTRING(10 140,49.375 134.375,68.125 135.625,79.375 141.875,87.5 152.5,92.5 167.5,101.875 176.25,115.625 ↩
    178.75,133.75 175,156.25 165,170 149.375,175 128.125,171.25 101.25,158.75 68.75,148.125 50.625,139.375 ↩
    46.875,132.5 57.5,127.5 82.5,119.375 95.625,108.125 96.875,93.75 86.25,76.25 63.75,61.875 51.25,50.625 ↩
    48.75,41.25 60.625,30 100)
3 | LINESTRING(10 140,39.53125 135.78125,54.0625 134.6875,63.4375 135.3125,70.9375 137.1875,76.5625 140.3125,81.40625 ↩
    144.53125,85.46875 149.84375,88.75 156.25,91.25 163.75,94.84375 169.6875,99.53125 174.0625,105.3125 ↩
    176.875,112.1875 178.125,120.15625 177.8125,129.21875 175.9375,139.375 172.5,150.625 167.5,159.6875 ↩
    161.09375,166.5625 153.28125,171.25 144.0625,173.75 133.4375,174.0625 121.40625,172.1875 107.96875,168.125 ↩
    93.125,161.875 76.875,156.09375 64.21875,150.78125 55.15625,145.9375 49.6875,141.5625 47.8125,137.65625 ↩
    49.53125,134.21875 54.84375,131.25 63.75,128.75 76.25,125.46875 85.78125,121.40625 92.34375,116.5625 ↩
    95.9375,110.9375 96.5625,104.53125 94.21875,97.34375 88.90625,89.375 80.625,80.625 69.375,72.65625 ↩
    60.625,65.46875 54.375,59.0625 50.625,53.4375 49.375,48.28125 51.71875,43.59375 57.65625,38.4375 70.46875,30 100)
```

Figure



Smoothing a triangle shows the weakest visual change, so it is kept last.

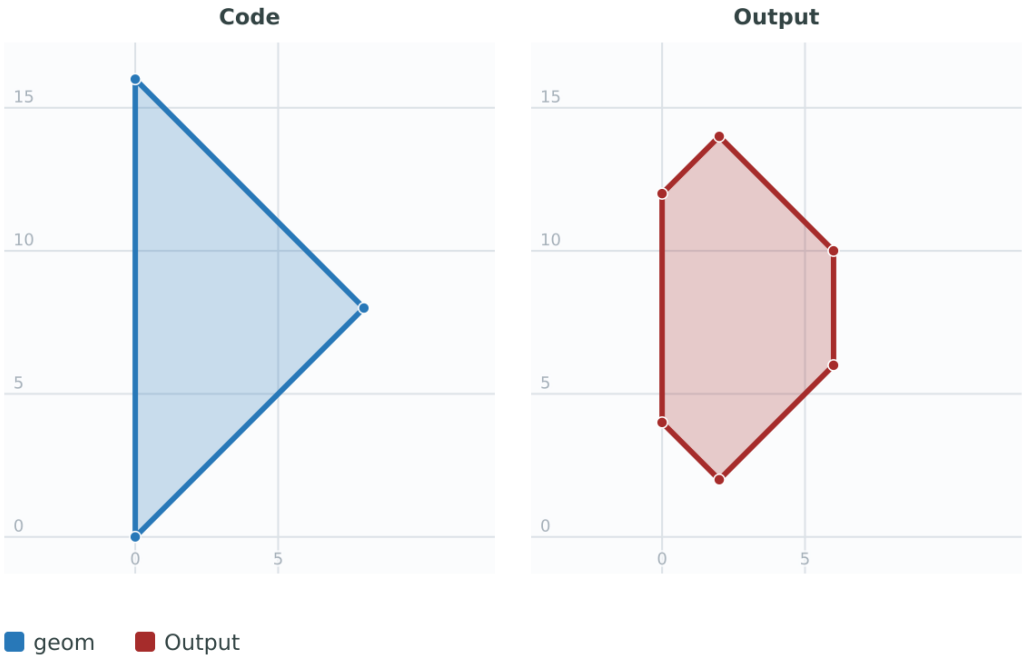
Code

```
SELECT ST_ChaikinSmoothing(geom) smoothed
FROM (SELECT  'POLYGON((0 0,8 8,0 16,0 0))'::geometry geom) AS foo;
```

Output

```
POLYGON((2 2,6 6,6 10,2 14,0 12,0 4,2 2))
```

Figure



相关信息

ST_Simplify, ST_SimplifyPreserveTopology, ST_SimplifyVW

7.14.5 ST_CatmullRomSmoothing

ST_CatmullRomSmoothing — Returns a smoothed version of a geometry, using the Catmull-Rom spline algorithm

Synopsis

geometry **ST_CatmullRomSmoothing**(geometry geom, integer nSegments = 5);

描述

Smoothes a linear or polygonal geometry using the Catmull-Rom spline algorithm. Unlike ST_ChaikinSmooth this is an *interpolating* spline: the output curve passes through every original vertex. Between each pair of consecutive original vertices, `nSegments` - 1 new intermediate points are inserted.

The `nSegments` parameter controls the density of the output. With `nSegments` = 5 (the default), each original span is divided into 5 sub-segments (inserting 4 new points per span). The minimum value is 2.

At least 4 input vertices are required to apply smoothing; geometries with fewer vertices are returned unchanged. Points and multipoints are always returned unchanged.



Note
The output vertex count grows as `1 + (N-1) * nSegments` for open lines, and `1 + N * nSegments` for closed rings, where N is the number of input vertices. For large geometries, use a simplification function on the result to reduce the number of points (see ST_Simplify, ST_SimplifyPreserveTopology and ST_SimplifyVW).

☒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。

Availability: 3.7.0

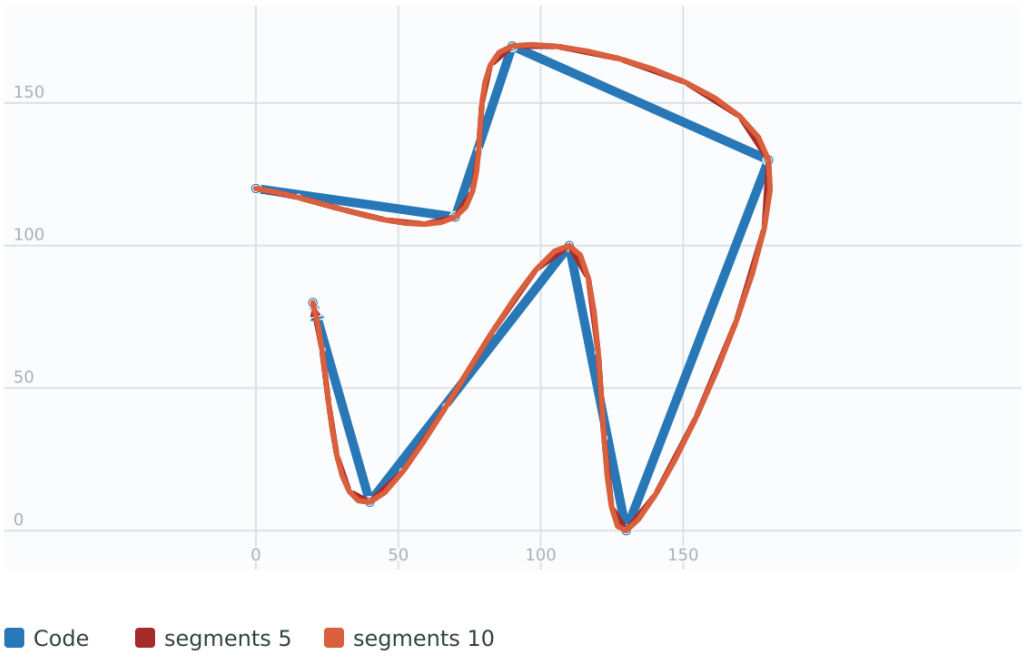
示例

Smoothing a LineString using nSegments = 5 and 10.

Code

```
WITH input(geom) AS (VALUES
  ('LINESTRING (10 140,80 130,100 190,190 150,140 20,120 120,50 30,30 100)'::geometry)
)
SELECT ST_CatmullRomSmoothing(geom, 5) AS segments_5,
  ST_CatmullRomSmoothing(geom, 10) AS segments_10
FROM input;
```

Figure



Smoothing a 4-point collinear line with default nSegments=5 shows the minimal change case.

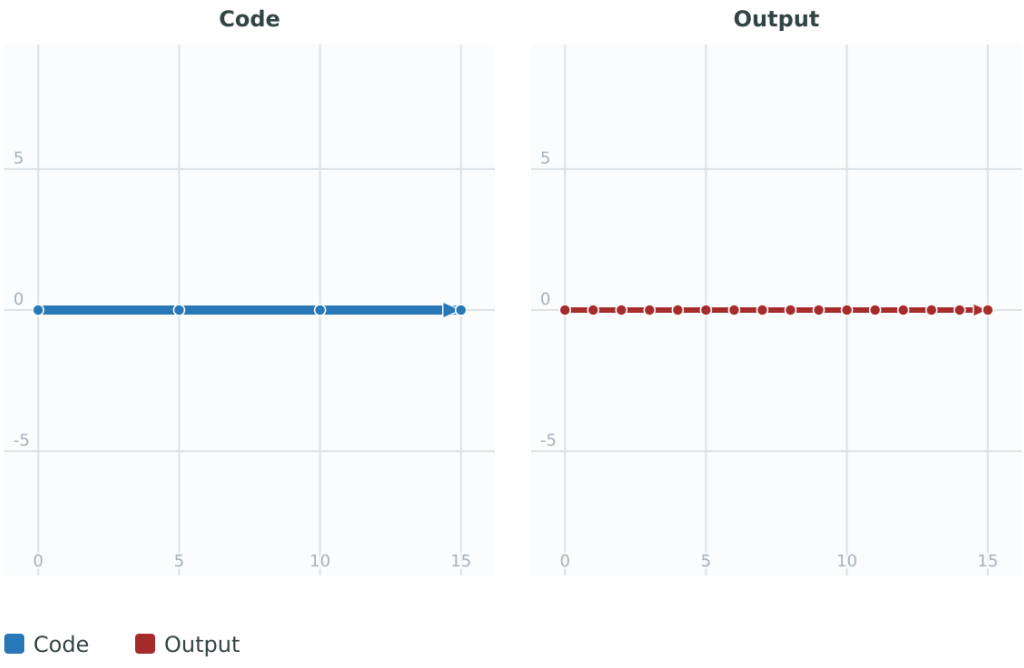
Code

```
SELECT ST_CatmullRomSmoothing('LINESTRING(0 0,5 0,10 0,15 0)');
```

Output

```
LINESTRING(0 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 0,10 0,11 0,12 0,13 0,14 0,15 0)
```

Figure

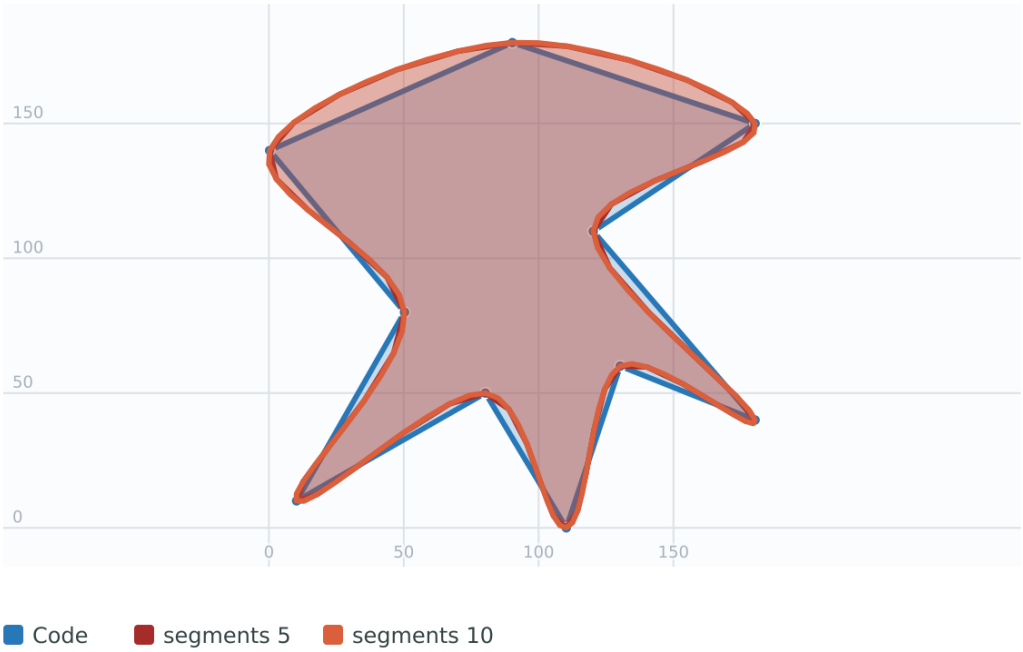


Smoothing a Polygon using nSegments = 5 and 10. This larger example is kept last.

Code

```
WITH input(geom) AS (VALUES
  ('POLYGON ((20 20,60 90,10 150,100 190,190 160,130 120,190 50,140 70,120 10,90 60,20 20))
    '::geometry)
)
SELECT ST_CatmullRomSmoothing(geom, 5) AS segments_5,
  ST_CatmullRomSmoothing(geom, 10) AS segments_10
FROM input;
```

Figure



相关信息

ST_ChaikinSmoothing, ST_Simplify, ST_SimplifyPreserveTopology, ST_SimplifyVW

7.14.6 ST_ConcaveHull

ST_ConcaveHull — 计算包含所有输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，包含输入，其点是输入点的子集。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。

This is not an aggregate function. To compute the concave hull of a set of geometries use ST_Collect (e.g. ST_ConcaveHull(ST_Collect(geom), 0.80).

可用性: 2.0.0

增⍰：3.3.0，⍰ GEOS 3.11+ ⍰用 GEOS 本机⍰⍰

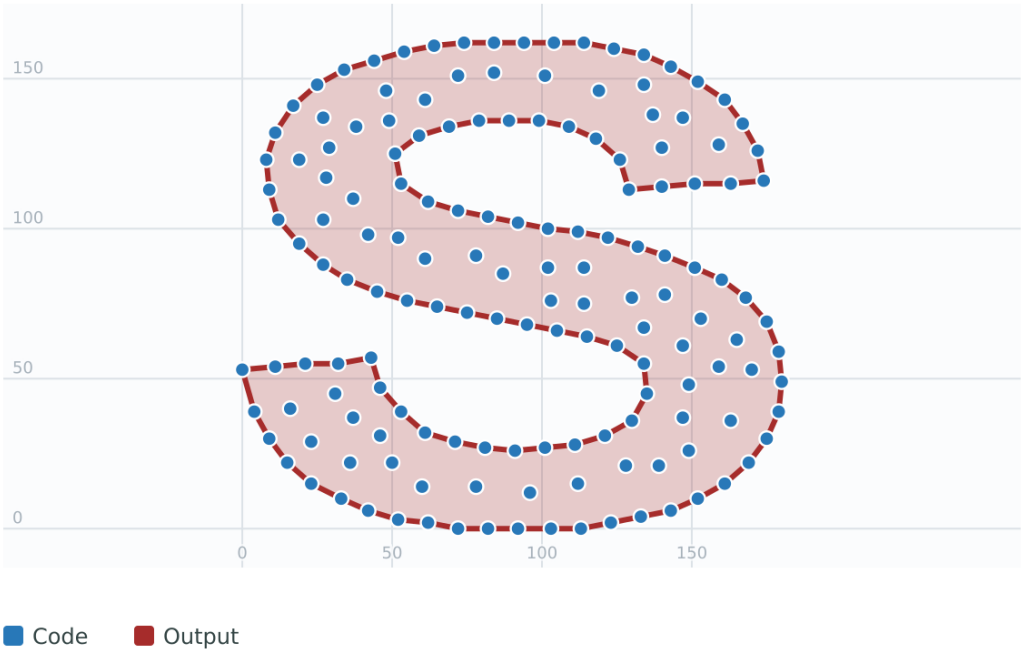
示例

Concave hull of a MultiPoint.

Code

```
SELECT ST_ConcaveHull('MULTIPOINT ((10 72),(53 76),(56 66),(63 58),(71 51),(81 48),(91 46) ↩
    ,(101 45),(111 46),(121 47),(131 50),(140 55),(145 64),(144 74),(135 80),(125 83),(115  ↩
    85),(105 87),(95 89),(85 91),(75 93),(65 95),(55 98),(45 102),(37 107),(29 114),(22 122) ↩
    ,(19 132),(18 142),(21 151),(27 160),(35 167),(44 172),(54 175),(64 178),(74 180),(84  ↩
    181),(94 181),(104 181),(114 181),(124 181),(134 179),(144 177),(153 173),(162 168),(171 ↩
     162),(177 154),(182 145),(184 135),(139 132),(136 142),(128 149),(119 153),(109 155) ↩
    ,(99 155),(89 155),(79 153),(69 150),(61 144),(63 134),(72 128),(82 125),(92 123),(102  ↩
    121),(112 119),(122 118),(132 116),(142 113),(151 110),(161 106),(170 102),(178 96),(185 ↩
     88),(189 78),(190 68),(189 58),(185 49),(179 41),(171 34),(162 29),(153 25),(143 23) ↩
    ,(133 21),(123 19),(113 19),(102 19),(92 19),(82 19),(72 21),(62 22),(52 25),(43 29),(33 ↩
     34),(25 41),(19 49),(14 58),(21 73),(31 74),(42 74),(173 134),(161 134),(150 133),(97  ↩
    104),(52 117),(157 156),(94 171),(112 106),(169 73),(58 165),(149 40),(70 33),(147 157) ↩
    ,(48 153),(140 96),(47 129),(173 55),(144 86),(159 67),(150 146),(38 136),(111 170),(124 ↩
     94),(26 59),(60 41),(71 162),(41 64),(88 110),(122 34),(151 97),(157 56),(39 146),(88  ↩
    33),(159 45),(47 56),(138 40),(129 165),(33 48),(106 31),(169 147),(37 122),(71 109) ↩
    ,(163 89),(37 156),(82 170),(180 72),(29 142),(46 41),(59 155),(124 106),(157 80),(175  ↩
    82),(56 50),(62 116),(113 95),(144 167))',
         0.1 );
```

Figure

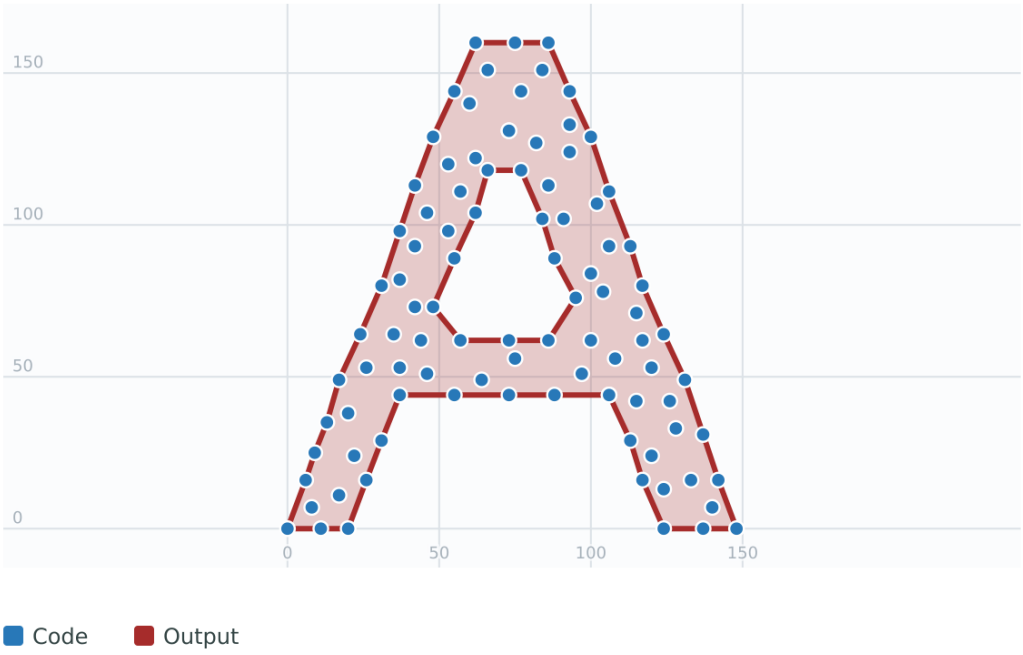


Concave hull of a MultiPoint, allowing holes.

Code

```
SELECT ST_ConcaveHull('MULTIPOINT ((132 64),(114 64),(99 64),(81 64),(63 64),(57 49),(52
    36),(46 20),(37 20),(26 20),(32 36),(39 55),(43 69),(50 84),(57 100),(63 118),(68 133)
    ,(74 149),(81 164),(88 180),(101 180),(112 180),(119 164),(126 149),(132 131),(139 113)
    ,(143 100),(150 84),(157 69),(163 51),(168 36),(174 20),(163 20),(150 20),(143 36),(139
    49),(132 64),(99 151),(92 138),(88 124),(81 109),(74 93),(70 82),(83 82),(99 82),(112
    82),(126 82),(121 96),(114 109),(110 122),(103 138),(99 151),(34 27),(43 31),(48 44),(46
     58),(52 73),(63 73),(61 84),(72 71),(90 69),(101 76),(123 71),(141 62),(166 27),(150
    33),(159 36),(146 44),(154 53),(152 62),(146 73),(134 76),(143 82),(141 91),(130 98)
    ,(126 104),(132 113),(128 127),(117 122),(112 133),(119 144),(108 147),(119 153),(110
    171),(103 164),(92 171),(86 160),(88 142),(79 140),(72 124),(83 131),(79 118),(68 113)
    ,(63 102),(68 93),(35 45))',
         0.15, true );
```

Figure



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。

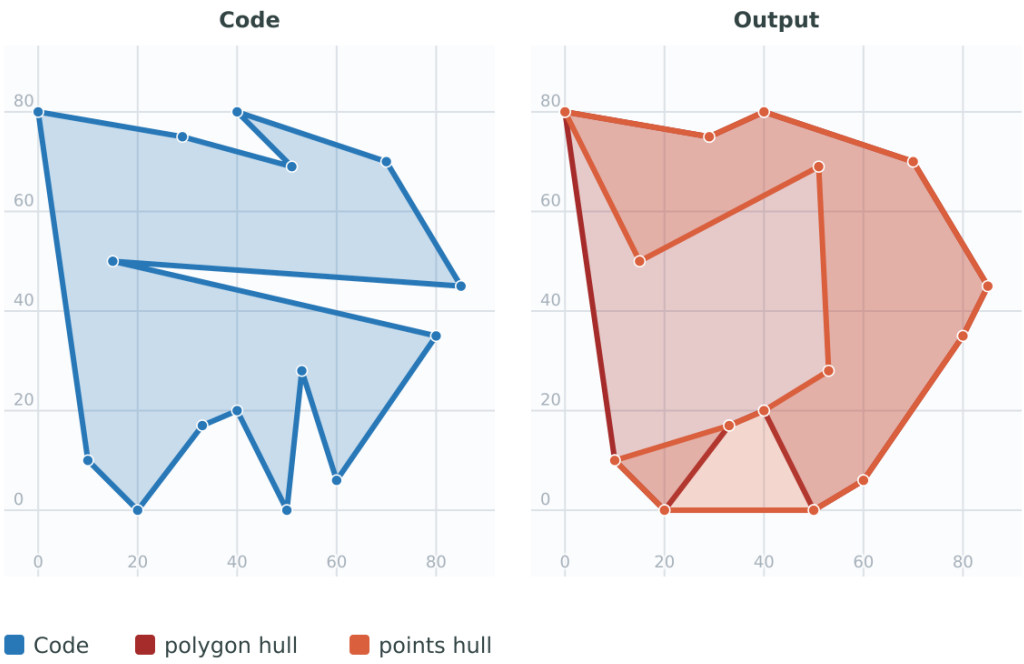
Code

```
WITH data(geom) AS (VALUES
   ('POLYGON ((10 90,39 85,61 79,50 90,80 80,95 55,25 60,90 45,70 16,63 38,60 10,50 30,43 ↩
    27,30 10,20 20,10 90))'::geometry)
)
SELECT ST_ConcaveHull(geom, 0.1) AS polygon_hull,
       ST_ConcaveHull(ST_Points(geom), 0.1) AS points_hull
    FROM data;
```

Output

```
POLYGON((60 10,50 30,43 27,30 10,20 20,10 90,39 85,50 90,80 80,95 55,90 45,70 16,60 10)) | POLYGON((25 60,10 90,39 ↩
    85,50 90,80 80,95 55,90 45,70 16,60 10,30 10,20 20,43 27,50 30,63 38,61 79,25 60))
```

Figure



与 ST_Collect 一起使用来⊠算几何⊠形集的凹壳。

This example estimates an infected area from point observations.

Code

```
SELECT disease_type,
    ST_ConcaveHull(ST_Collect(obs_pnt), 0.3 ) AS geom
  FROM disease_obs
  GROUP BY disease_type;
```

相关信息

ST_ConvexHull, ST_Collect, CG_AlphaShape, CG_OptimalAlphaShape

7.14.7 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。

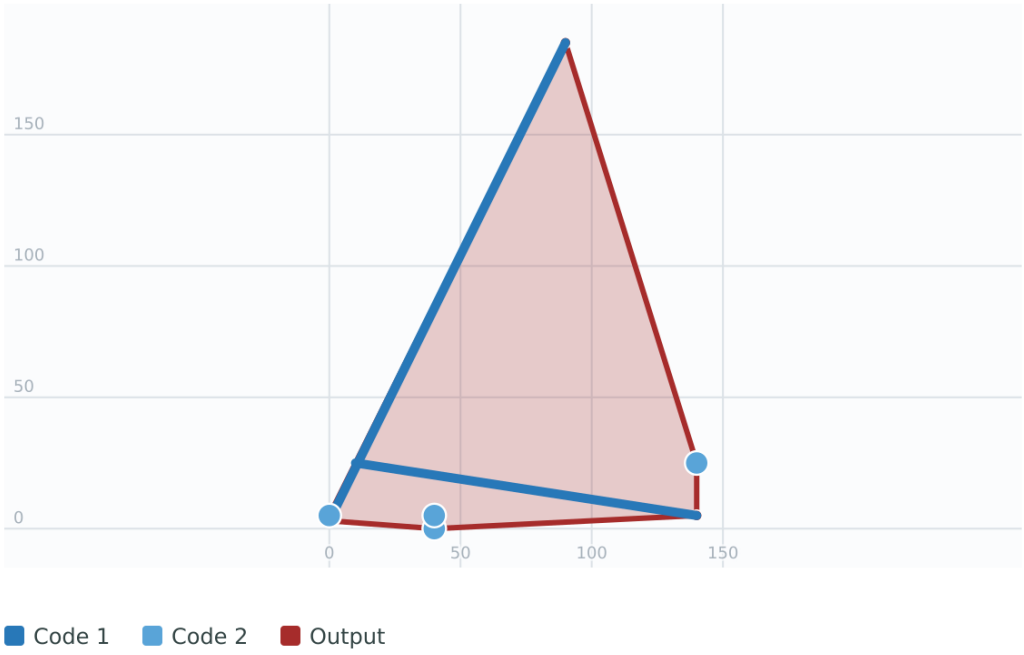
示例

Compute the convex hull of a MultiLineString and a MultiPoint.

Code

```
SELECT ST_ConvexHull(ST_Collect(ST_GeomFromText('MULTILINESTRING((100 190,10 8),(150 10,20 ↩
    30))'),
            ST_GeomFromText('MULTIPOINT(50 5,150 30,50 10,10 10)')
            ));
```

Figure



与 ST_Collect 一起使用来⊠算几何⊠形集的凸包。

This example returns an estimated infected area from point observations.

Code

```
SELECT d.disease_type,
    ST_ConvexHull(ST_Collect(d.geom)) As geom
    FROM disease_obs As d
    GROUP BY d.disease_type;
```

相关信息

ST_Collect, ST_ConcaveHull, ST_MinimumBoundingCircle

7.14.8 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)。

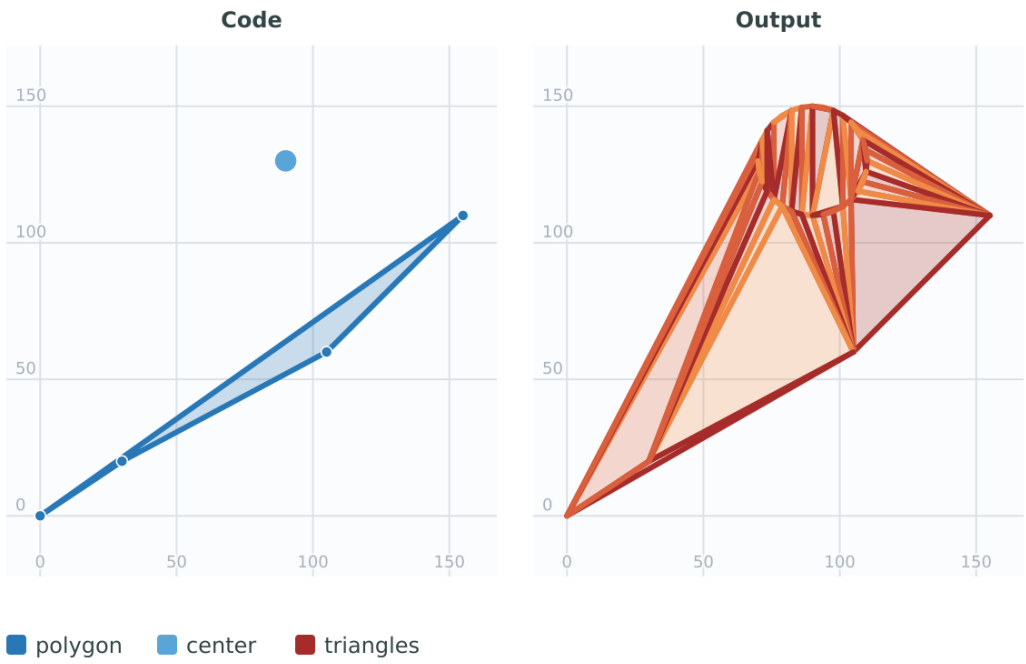
示例

Triangulation of the vertices of a polygon and a buffered point.

Code

```
WITH data AS (
  SELECT polygon, ST_Buffer(center, 20) AS point_buffer
  FROM (SELECT
    'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry AS polygon,
    'POINT(110 170)'::geometry AS center
  ) AS input
)
SELECT ST_DelaunayTriangles(ST_Union(polygon, point_buffer)) AS triangles
FROM data;
```

Figure

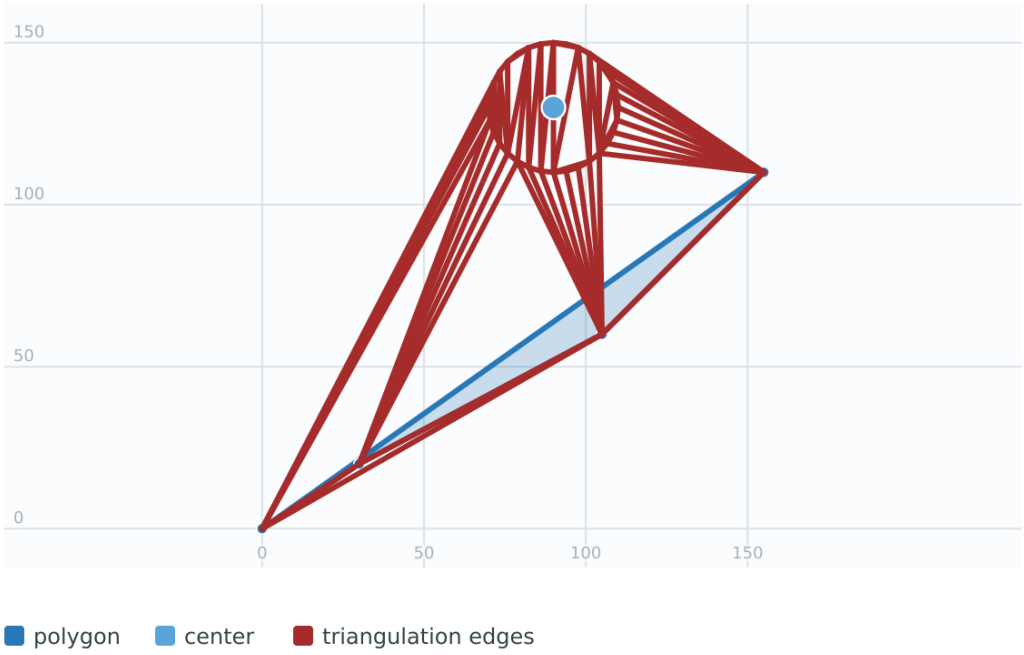


Triangulation edges returned as a `MULTILINESTRING`.

Code

```
WITH data AS (
  SELECT polygon, ST_Buffer(center, 20) AS point_buffer
  FROM (SELECT
    'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry AS polygon,
    'POINT(110 170)'::geometry AS center
  ) AS input
)
SELECT ST_DelaunayTriangles(ST_Union(polygon, point_buffer), 0.001, 1) AS ↩
    triangulation_edges
FROM data;
```

Figure

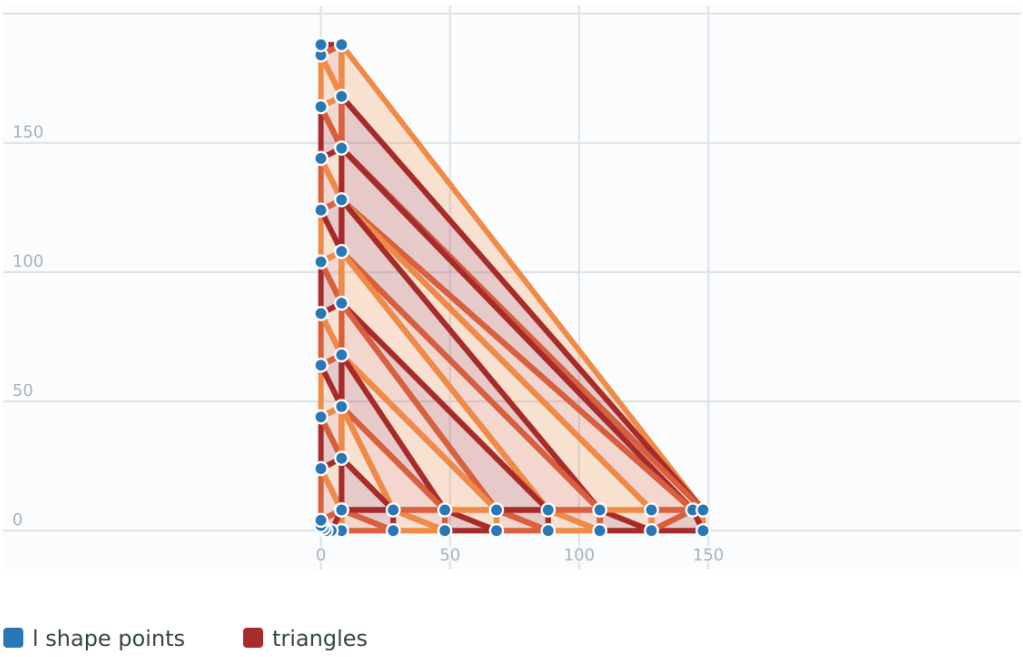


Triangulation of 42 points forming an L shape.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT(14 14,34 14,54 14,74 14,94 14,114 14,134 14,
    150 14,154 14,154 6,134 6,114 6,94 6,74 6,54 6,34 6,
    14 6,10 6,8 6,7 7,6 8,6 10,6 30,6 50,6 70,6 90,6 110,6 130,
    6 150,6 170,6 190,6 194,14 194,14 174,14 154,14 134,14 114,
    14 94,14 74,14 54,14 34,14 14)'::geometry AS l_shape_points
)
SELECT ST_DelaunayTriangles(l_shape_points) AS triangles
FROM data;
```

Figure



使用具有 Z 区的区点的示例。

Code

```
SELECT ST_DelaunayTriangles(
  'MULTIPOINT Z(14 14 10,150 14 100,34 6 25,20 10 150)'::geometry);
```

Output

```
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, CG_ConstrainedDelaunayTriangles, ST_VoronoiLines, ST_ConvexHull

7.14.9 ST_FilterByM

ST_FilterByM — 根据 M 区区除区点

Synopsis

geometry **ST_FilterByM**(geometry geom, double precision min, double precision max = null, boolean returnM = false);

描述

根据 M ⍰⍰⍰掉⍰点。返回⍰包含 M ⍰大于或等于最小⍰且小于或等于最大⍰的⍰点的几何⍰形。如果省略最大⍰参数，⍰⍰考⍰最小⍰。如果省略第四个参数，⍰ m ⍰将不会出⍰在生成的几何⍰形中。如果生成的几何⍰形⍰其几何⍰型留下的⍰点太少，⍰将返回空几何⍰形。在几何集合中，没有足⍰点的几何⍰形将被默默地排除。

⍰函数主要与 ST_SetEffectiveArea ⍰合使用。ST_EffectiveArea 用其 m ⍰⍰置⍰点的有效区域。使用 ST_FilterByM，无需任何⍰算，只需⍰⍰即可⍰得几何的⍰化版本



Note
与 ST_FilterByM 相比，当没有足⍰的点⍰足条件⍰，ST_SimplifyVW 返回的⍰果有所不同。ST_SimplifyVW 返回具有足⍰点的几何⍰形，而 ST_FilterByM 返回空几何⍰形



Note
⍰注意，返回的几何⍰形可能无效



Note
⍰函数返回所有⍰度，包括 Z 和 M ⍰

可用性：2.5.0

示例

⍰串被⍰⍰

Code

```
SELECT ST_FilterByM(geom, 30) simplified
FROM (SELECT  ST_SetEffectiveArea('LINESTRING(5 2,3 8,6 20,7 25,10 10)'::geometry) geom) As ↩
     foo;
```

Output

```
LINESTRING(5 2,7 25,10 10)
```

Figure



相关信息

ST_SetEffectiveArea, ST_SimplifyVW

7.14.10 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

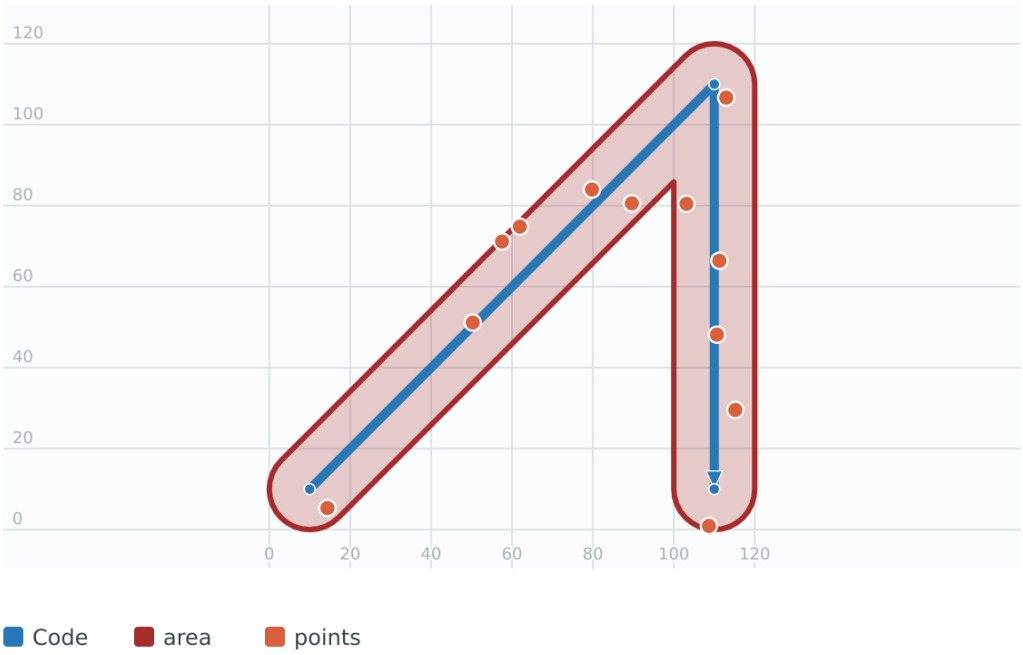
示例

Generate 12 points inside a buffered LineString using the deterministic seed 1996.

Code

```
WITH source AS (
  SELECT ST_Buffer(ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'),
           10, 'endcap=round join=round') AS geom
)
SELECT geom AS area,
       ST_GeneratePoints(geom, 12, 1996) AS points
FROM source;
```

Figure



⌧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。

Code

```
SELECT s.id, dp.path[1] AS pt_id, dp.geom
FROM s, ST_DumpPoints(ST_GeneratePoints(s.geom, 12)) AS dp;
```

Generate dot-density points from a polygon table, using the absolute value of an attribute as the number of points and keeping the sign of the attribute for styling.

Code

```
SELECT z.id,
       CASE WHEN z.count_delta < 0 THEN -1 ELSE 1 END AS sign,
       dp.path[1] AS pt_id,
       dp.geom
FROM zones AS z
CROSS JOIN LATERAL ST_DumpPoints(
    ST_GeneratePoints(z.geom, abs(z.count_delta)::integer, z.id + 1)
) AS dp
WHERE z.count_delta <> 0;
```

相关信息

ST_DumpPoints

7.14.11 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⊠。

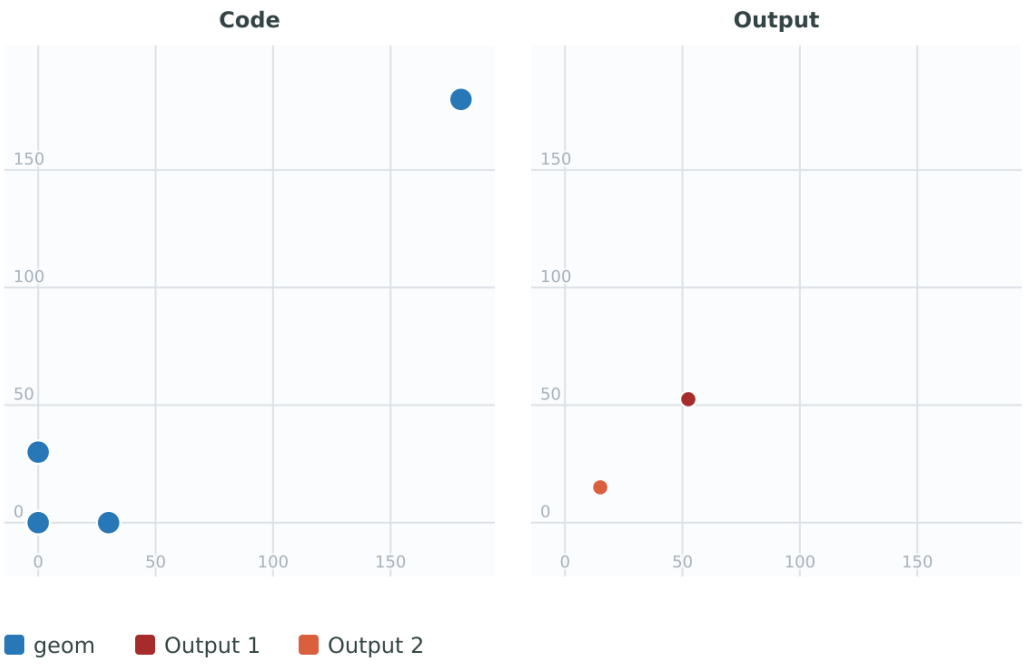
示例

Compare the geometric median and centroid of a MultiPoint.

Code

```
WITH test AS (
SELECT 'MULTIPOINT((10 10),(10 40),(40 10),(190 190))'::geometry geom)
SELECT
  ST_Centroid(geom) centroid,
  ST_GeometricMedian(geom) median
FROM test;
```

Figure



相关信息

ST_Centroid

7.14.12 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 区度。

示例

次数 2

次数 2 的区点会被合并移除。

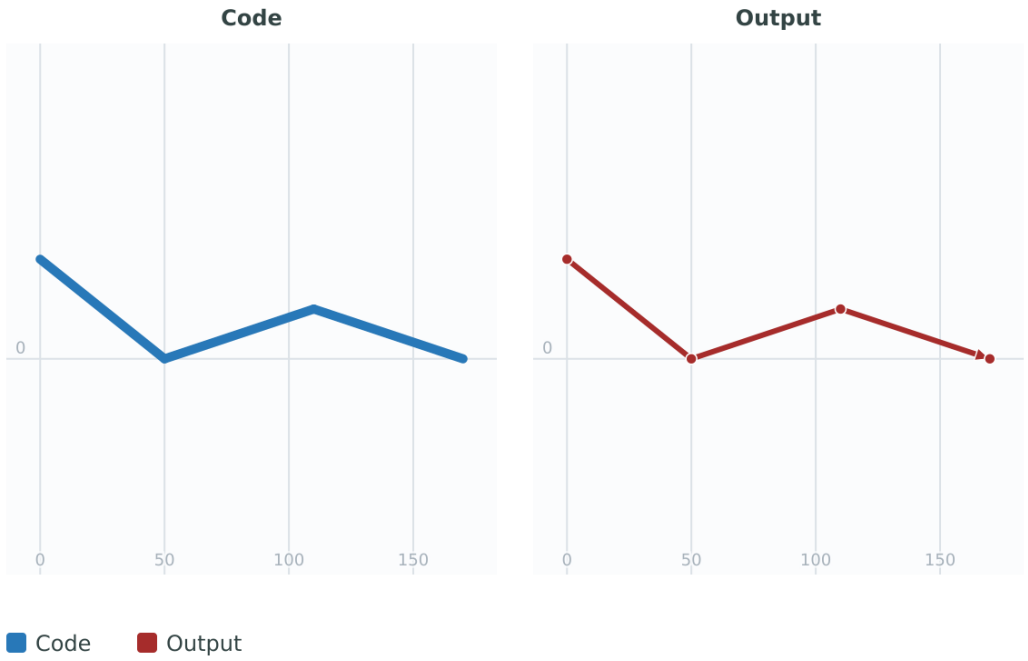
Code

```
SELECT ST_LineMerge('MULTILINESTRING((10 160,60 120),(120 140,60 120),(120 140,180 120))'
);
```

Output

```
LINESTRING(10 160,60 120,120 140,180 120)
```

Figure



次数 3 以上

在交点的度数大于 2 区，区段不会合并。

Code

```
SELECT ST_LineMerge('MULTILINESTRING((10 160,60 120),(120 140,60 120),(120 140,180 120)
    ,(100 180,120 140))'
);
```

Output

```
MULTILINESTRING((10 160,60 120,120 140),(100 180,120 140),(120 140,180 120))
```

Figure



不相交的⊠

如果由于不接触⊠而无法合并，⊠返回原始 MultiLineString。

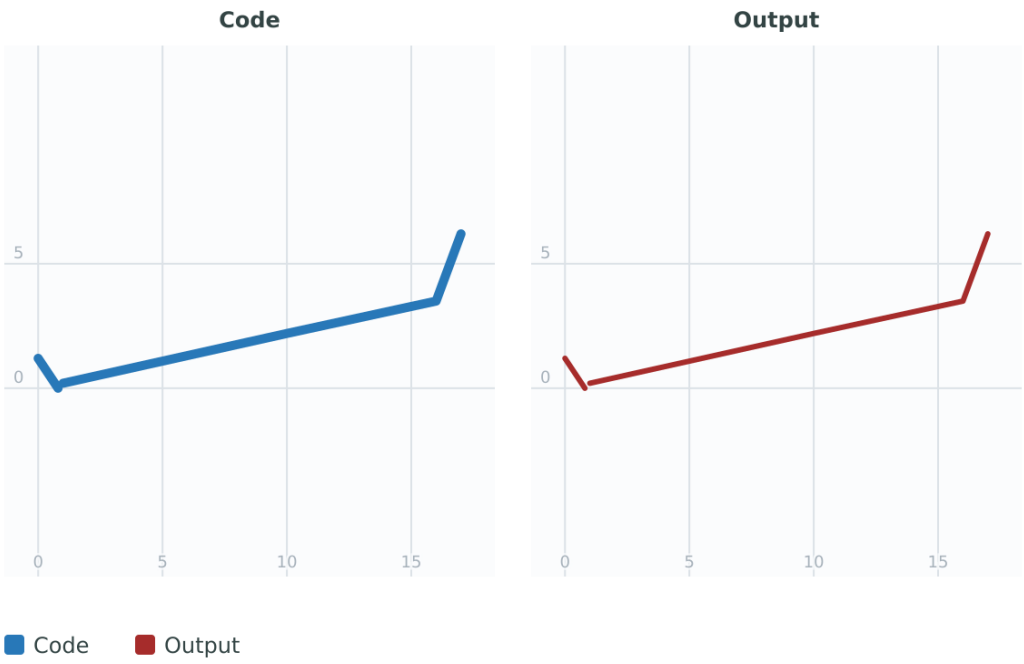
Code

```
SELECT ST_LineMerge('MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33),(-45.2 -33.2,-46
    -32))'
);
```

Output

```
MULTILINESTRING((-45.2 -33.2,-46 -32),(-29 -27,-30 -29.7,-36 -31,-45 -33))
```

Figure



有向参数

Lines with opposite directions are not merged if *directed* is TRUE.

Code

```
SELECT ST_LineMerge('MULTILINESTRING((60 30,10 70),(120 50,60 30),(120 50,180 30))',
TRUE);
```

Output

```
MULTILINESTRING((120 50,60 30,10 70),(120 50,180 30))
```

Figure



Z 区度区理

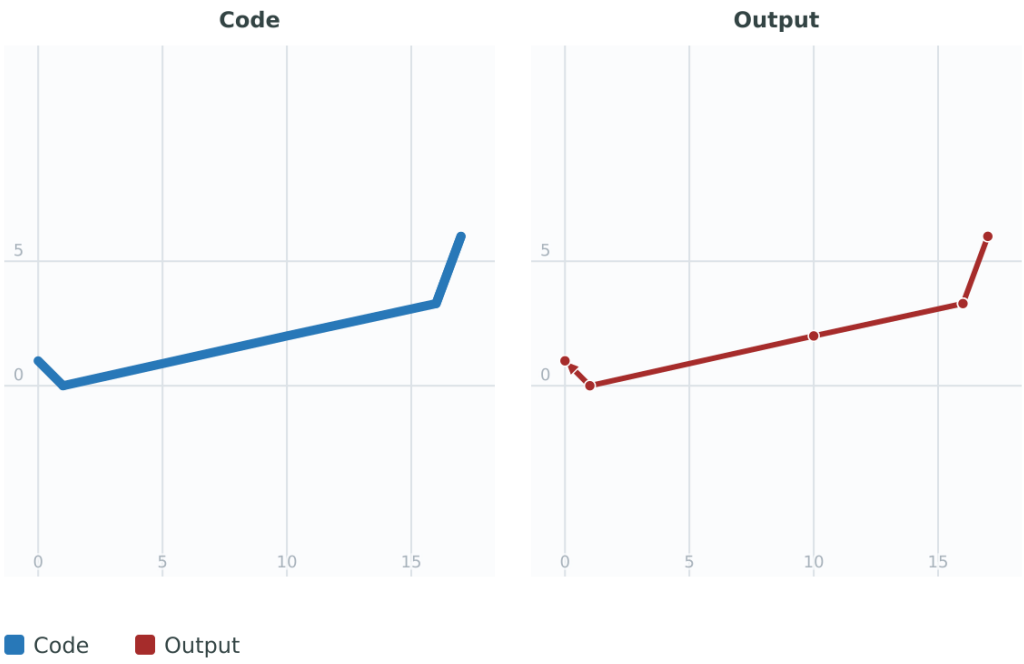
Code

```
SELECT ST_LineMerge('MULTILINESTRING((-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 6),(-29 -27 ↩
     12,-30 -29.7 5),(-45 -33 1,-46 -32 11))'
         );
```

Output

```
LINESTRING Z (-30 -29.7 5,-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 1,-46 -32 11)
```

Figure



相关信息

ST_Segmentize, ST_LineSubstring

7.14.13 ST_MaximumInscribedCircle

ST_MaximumInscribedCircle — 区算几何体中包含的最大区。

Synopsis

(geometry, geometry, double precision) **ST_MaximumInscribedCircle**(geometry geom);

描述

区找（多）多区形内包含的最大区，或者不与任何区和点重区的最大区。返回包含字段的区区：

- `center`-区的中心点
- `nearest`-几何上最接近中心的点
- `radius`-区的半径

区于多区形区入，区内接于区界区内，使用内区作区区界。区于区性和点区入，区内切于区入的凸包内，使用区入区和点作区区一步的区界。

可用性：3.1.0。

需要 GEOS >= 3.9.0。

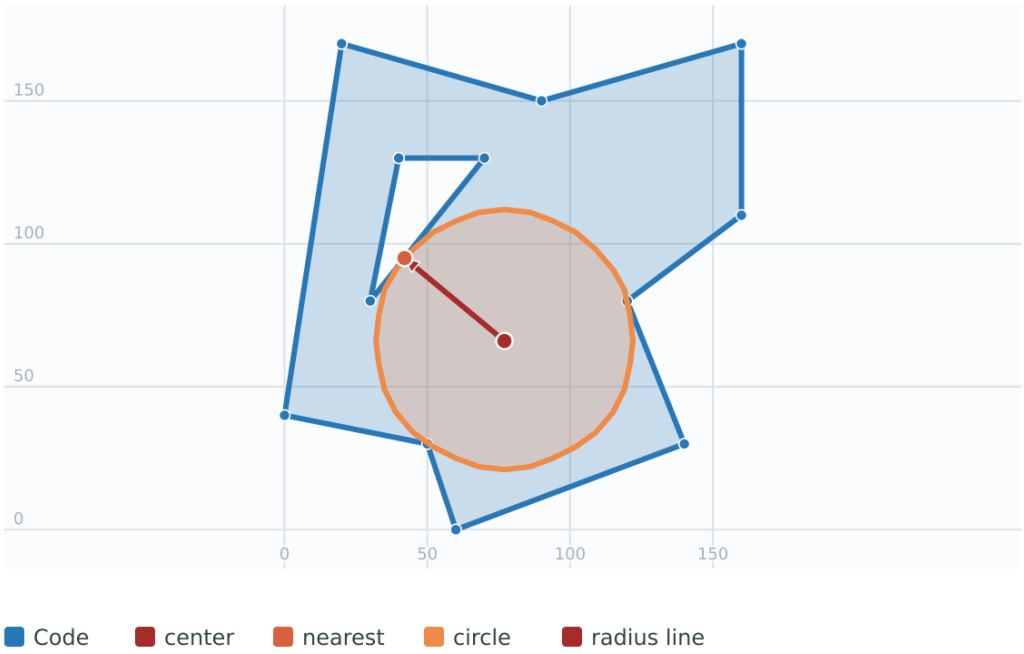
示例

Maximum inscribed circle of a polygon. Center, nearest point, radius, circle, and radius line are returned.

Code

```
WITH mic AS (
  SELECT * FROM ST_MaximumInscribedCircle(
    'POLYGON ((40 180,110 160,180 180,180 120,140 90,160 40,80 10,70 40,20 50,40 180),
              (60 140,50 90,90 140,60 140))')
)
SELECT round(radius::numeric, 3) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line
FROM mic;
```

Figure

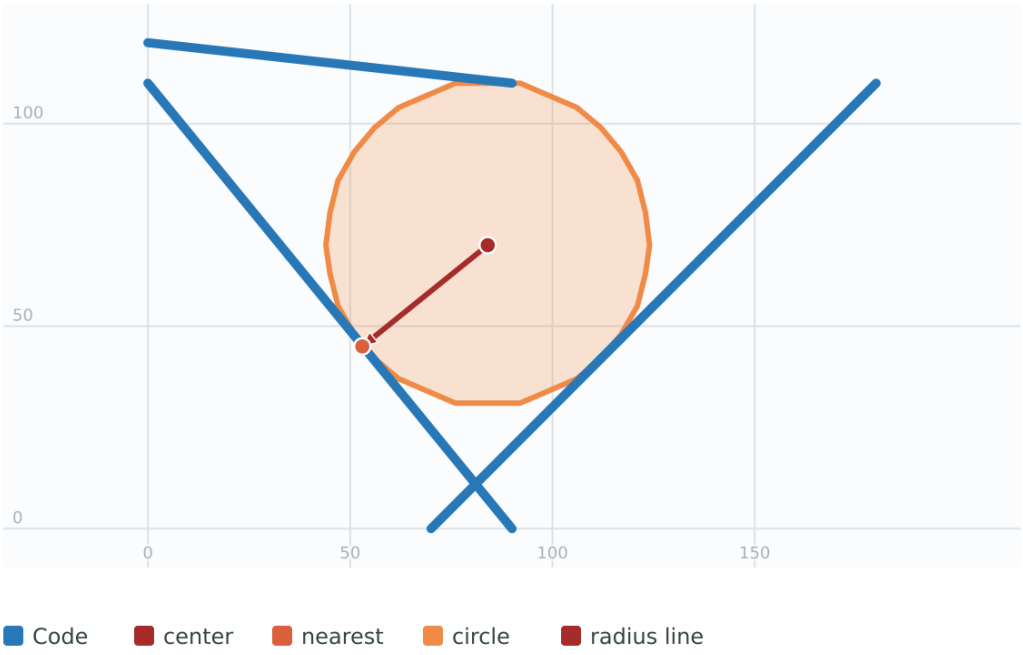


Maximum inscribed circle of a MultiLineString.

Code

```
WITH mic AS (
  SELECT * FROM ST_MaximumInscribedCircle(
    'MULTILINESTRING((100 40,10 150),(10 160,100 150),(190 150,80 40))')
)
SELECT round(radius::numeric, 3) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line
FROM mic;
```

Figure



相关信息

ST_MinimumBoundingRadius, ST_LargestEmptyCircle

7.14.14 ST_LargestEmptyCircle

ST_LargestEmptyCircle — 计算不与几何形重叠的最大圆。

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叠的最大圆。（多形几何形可以作障碍物包含在内，但使用它的界圆。）圆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。

If a boundary is supplied it must be a valid polygonal geometry with positive area.

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。

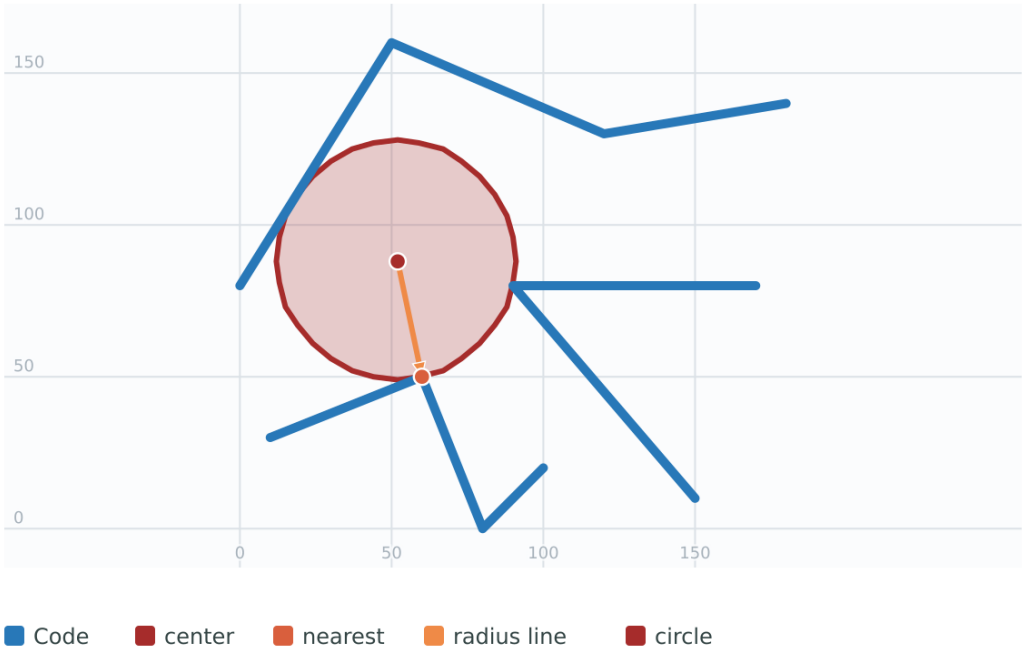
示例

Largest empty circle within a set of lines.

Code

```
WITH obstacles(geom) AS (VALUES ('MULTILINESTRING (
        (10 100,60 180,130 150,190 160),
        (20 50,70 70,90 20,110 40),
        (160 30,100 100,180 100))'::geometry)),
circle AS (
  SELECT result.*
  FROM obstacles
  CROSS JOIN LATERAL ST_LargestEmptyCircle(obstacles.geom) AS result
)
SELECT round(radius::numeric, 6) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle
FROM circle;
```

Figure

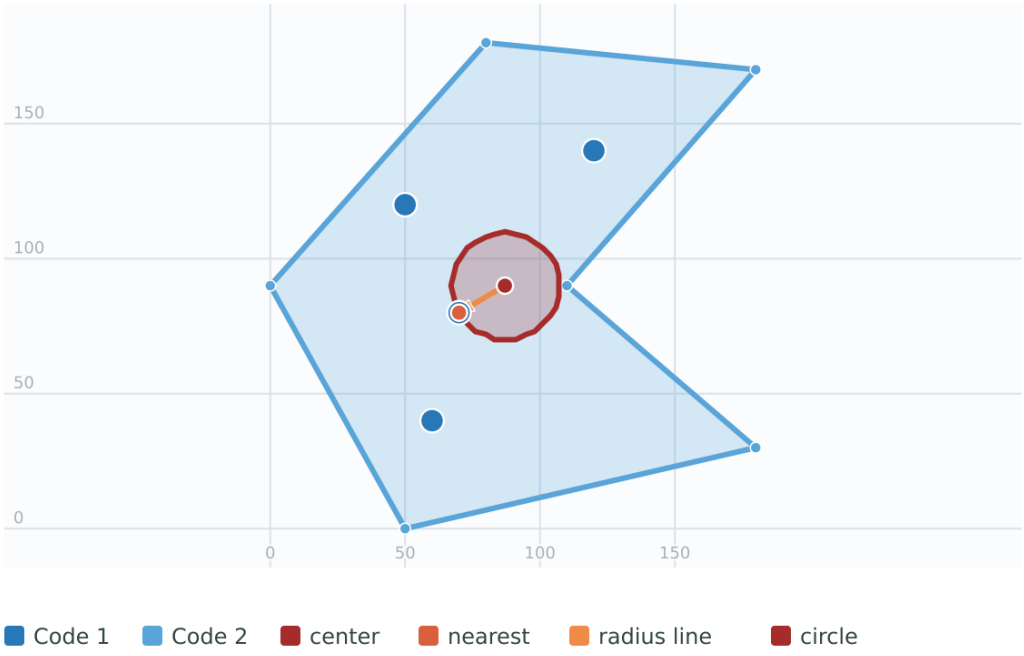


Largest empty circle within a set of points, constrained to lie in a polygon. The constraint polygon boundary is included both as an obstacle and as the constraint for the circle center.

Code

```
WITH inputs(points, boundary) AS (VALUES (
  'MULTIPOINT ((70 50),(60 130),(130 150),(80 90))'::geometry,
  'POLYGON ((90 190,10 100,60 10,190 40,120 100,190 180,90 190))'::geometry
)),
circle AS (
  SELECT result.*
  FROM inputs
  CROSS JOIN LATERAL ST_LargestEmptyCircle(
    ST_Collect(points, boundary), 0, boundary
  ) AS result
)
SELECT round(radius::numeric, 6) AS radius,
       ST_SnapToGrid(center, 1) AS center,
       ST_SnapToGrid(nearest, 1) AS nearest,
       ST_SnapToGrid(ST_MakeLine(center, nearest), 1) AS radius_line,
       ST_SnapToGrid(ST_Buffer(center, radius), 1) AS circle
FROM circle;
```

Figure



相关信息

ST_MinimumBoundingRadius

7.14.15 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

Code

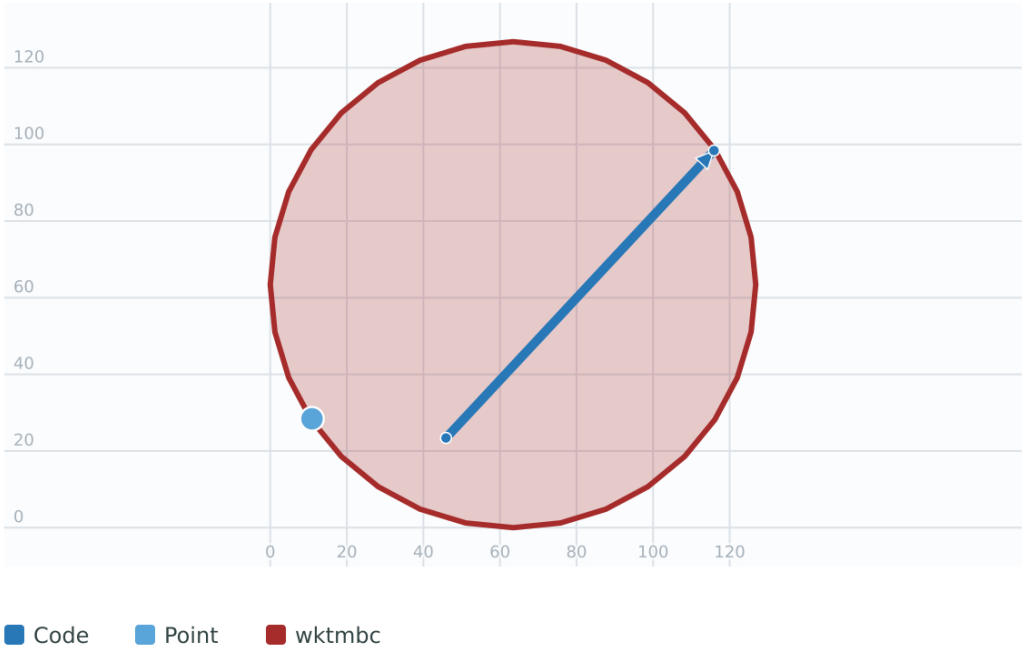
```
SELECT d.disease_type,
ST_MinimumBoundingCircle(ST_Collect(d.geom)) As geom
FROM disease_obs As d
GROUP BY d.disease_type;
```

Minimum bounding circle of a point and linestring, using 8 segments to approximate a quarter circle.

Code

```
SELECT ST_MinimumBoundingCircle(ST_Collect('LINESTRING(55 75,125 150)',
ST_Point(20, 80)), 8
) As wktmbc;
```

Figure



相关信息

ST_Collect, ST_MinimumBoundingRadius, ST_LargestEmptyCircle, ST_LongestLine

7.14.16 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

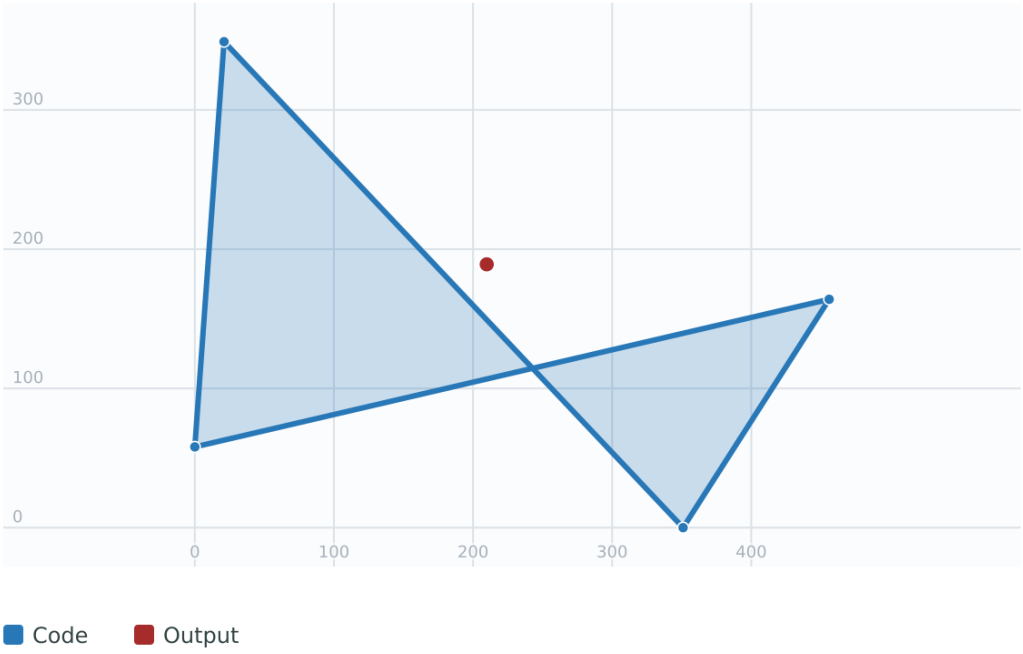
与ST_Collect 一起使用以⌧得一⌧几何⌧形的最小⌧界⌧。
要⌧算位于最小⌧（“最大直径”）上的⌧个点，⌧使用ST_LongestLine。
可用性 - 2.3.0

示例

Code

```
SELECT center, radius FROM ST_MinimumBoundingRadius('POLYGON((26426 65078,26531 65242,26075 ↩
    65136,26096 65427,26426 65078))');
```

Figure



相关信息

ST_Collect, ST_MinimumBoundingCircle, ST_LongestLine

7.14.17 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

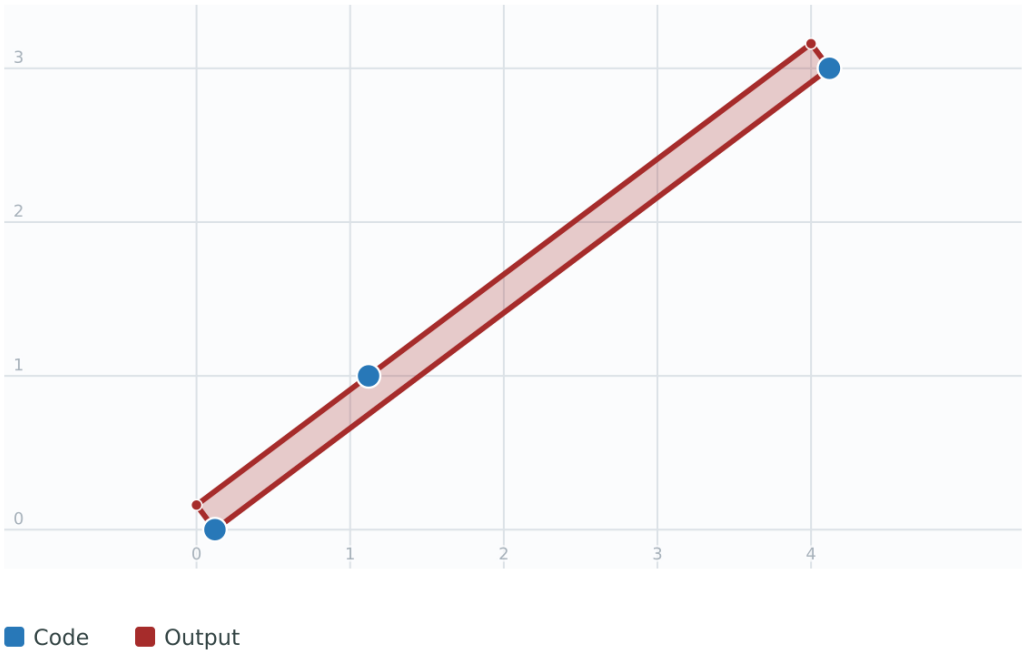
Code

```
SELECT ST_OrientedEnvelope('MULTIPOINT ((0 0),(-1 -1),(3 2))');
```

Output

```
POLYGON((3 2,-1 -1,-1.12 -0.84,2.88 2.16,3 2))
```

Figure



Oriented envelope of a Point and LineString.

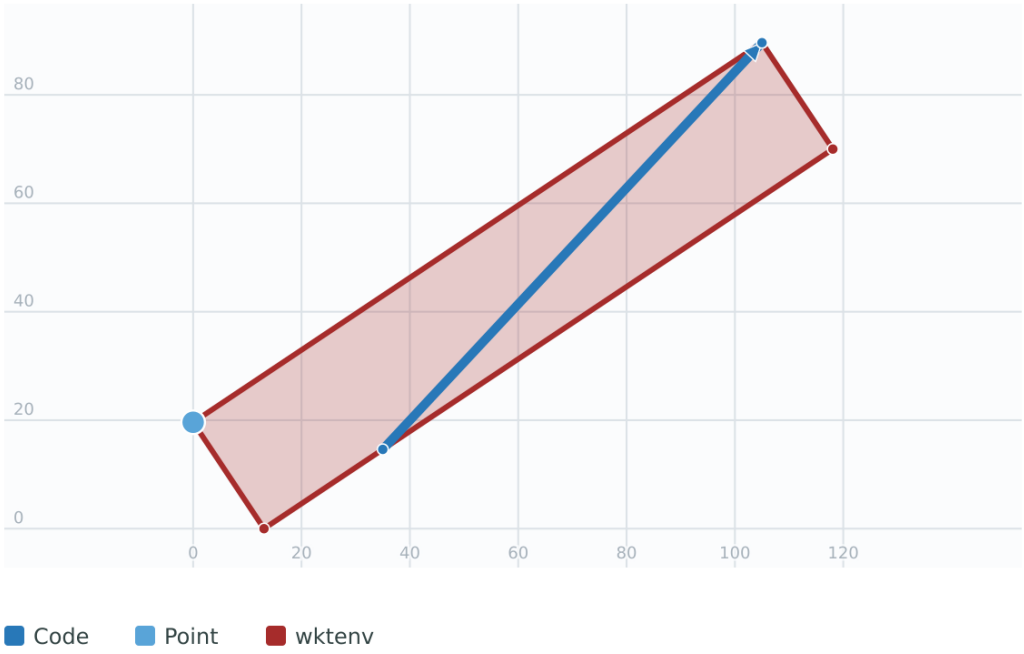
Code

```
SELECT ST_OrientedEnvelope(ST_Collect('LINESTRING(55 75,125 150)',
                ST_Point(20, 80))
                ) As wktenv;
```

Output

```
POLYGON((20 80,125 150,138.0769230769241 130.38461538461573,33.07692307692291 60.384615384615195,20 80))
```

Figure



相关信息

ST_Envelope ST_MinimumBoundingCircle

7.14.18 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。

在 GEOS 3.11 中，行⊠⊠生了⊠化，因此在正⊠偏移下，偏移曲⊠与⊠入⊠具有相同的方向。

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

Compute an open buffer around roads.

Code

```
SELECT ST_Union(
 ST_OffsetCurve(f.geom, f.width/2, 'quad_segs=4 join=round'),
 ST_OffsetCurve(f.geom, -f.width/2, 'quad_segs=4 join=round')
) as track
FROM someroadstable;
```

Offset 15 units with `quad_segs=4 join=round`.

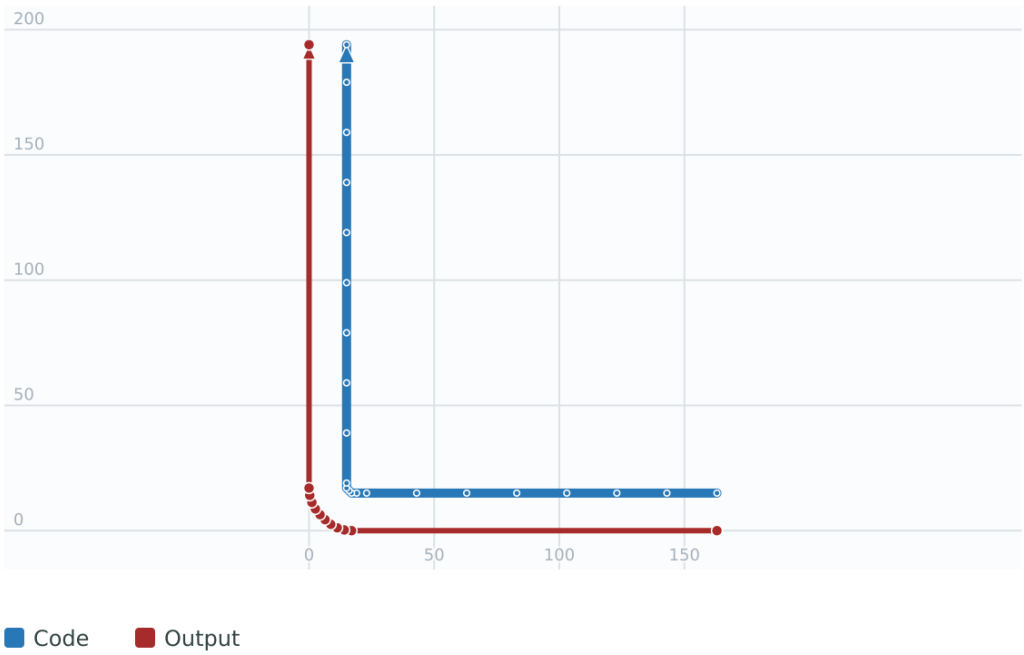
Code

```
SELECT ST_OffsetCurve(
    ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)'),
    15, 'quad_segs=4 join=round');
```

Output

```
LINESTRING(164 1,18 1,15.073645169758077 1.288220793951544,
    12.259748514523654 2.141807012330698,9.66644650470597 3.527955815461818,
    7.393398282201788 5.393398282201788,5.393398282201788 7.393398282201788,
    3.527955815461818 9.66644650470597,2.141807012330696 12.259748514523654,
    1.288220793951544 15.073645169758075,1 18,1 195)
```

Figure



Offset -15 units with `quad_segs=4 join=round`.

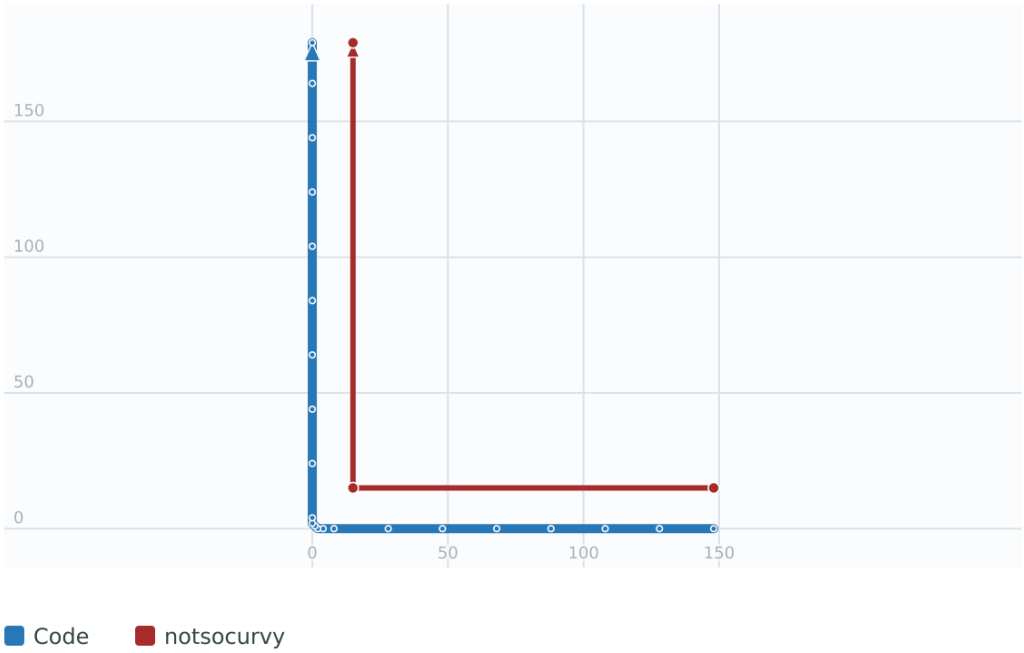
Code

```
SELECT ST_OffsetCurve(
    geom,
    -15,
    'quad_segs=4 join=round') As notsocurvy
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
LINESTRING(164 31,31 31,31 195)
```

Figure



Apply two negative offsets to obtain a smoother curve; the distances combine to -45 units.

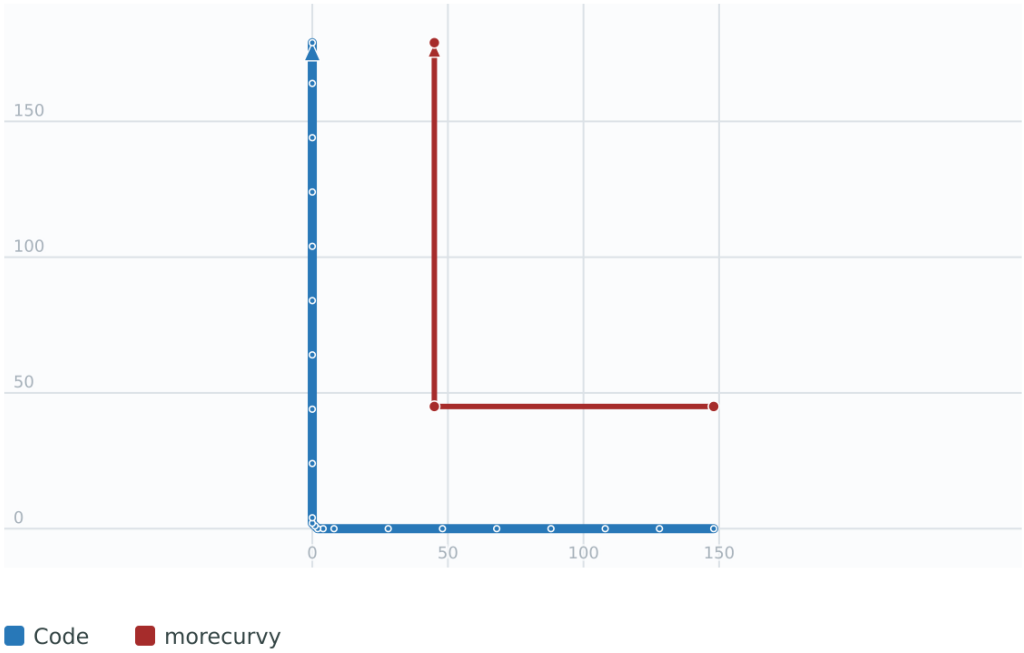
Code

```
SELECT ST_OffsetCurve(
    ST_OffsetCurve(
        geom,
        -30,
        'quad_segs=4 join=round'),
    -15,
    'quad_segs=4 join=round') As morecurvy
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
LINESTRING(164 61,61 61,61 195)
```

Figure



Collect the positive offset and the double-offset to produce parallel curves.

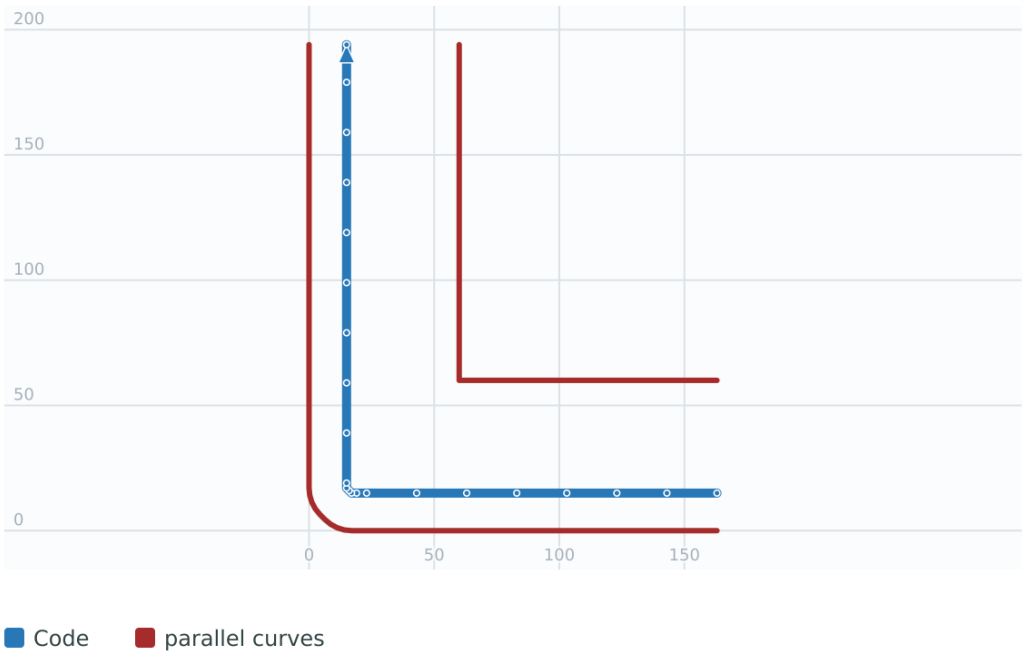
Code

```
SELECT ST_Collect(
    ST_OffsetCurve(geom, 15, 'quad_segs=4 join=round'),
    ST_OffsetCurve(
        ST_OffsetCurve(
            geom,
            -30,
            'quad_segs=4 join=round'),
        -15,
        'quad_segs=4 join=round')
) As parallel_curves
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
MULTILINESTRING((164 1,18 1,15.073645169758077 1.288220793951544,
12.259748514523654 2.141807012330698,9.66644650470597 3.527955815461818,
7.393398282201788 5.393398282201788,5.393398282201788 7.393398282201788,
3.527955815461818 9.66644650470597,2.141807012330696 12.259748514523654,
1.288220793951544 15.073645169758075,1 18,1 195),(164 61,61 61,61 195))
```

Figure



Offset 15 units with a bevel join.

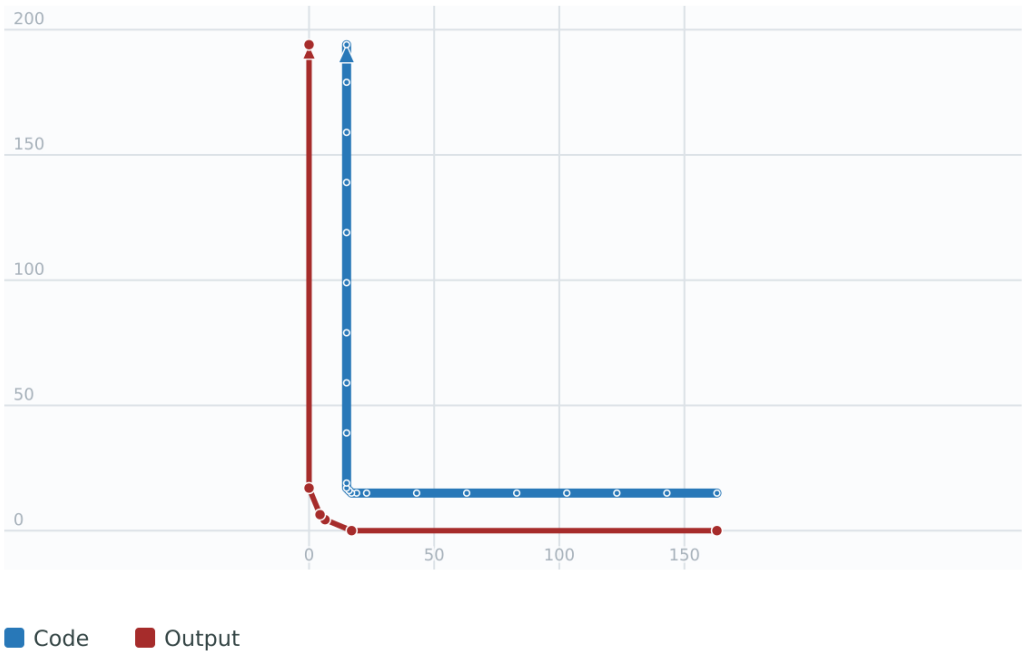
Code

```
SELECT ST_OffsetCurve(
    ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)'),
        15, 'quad_segs=4 join=bevel');
```

Output

```
LINESTRING(164 1,18 1,7.393398282201788 5.393398282201788,
    5.393398282201788 7.393398282201788,1 18,1 195)
```

Figure



Collect offsets on both sides using a mitre join and a mitre limit of 2.2.

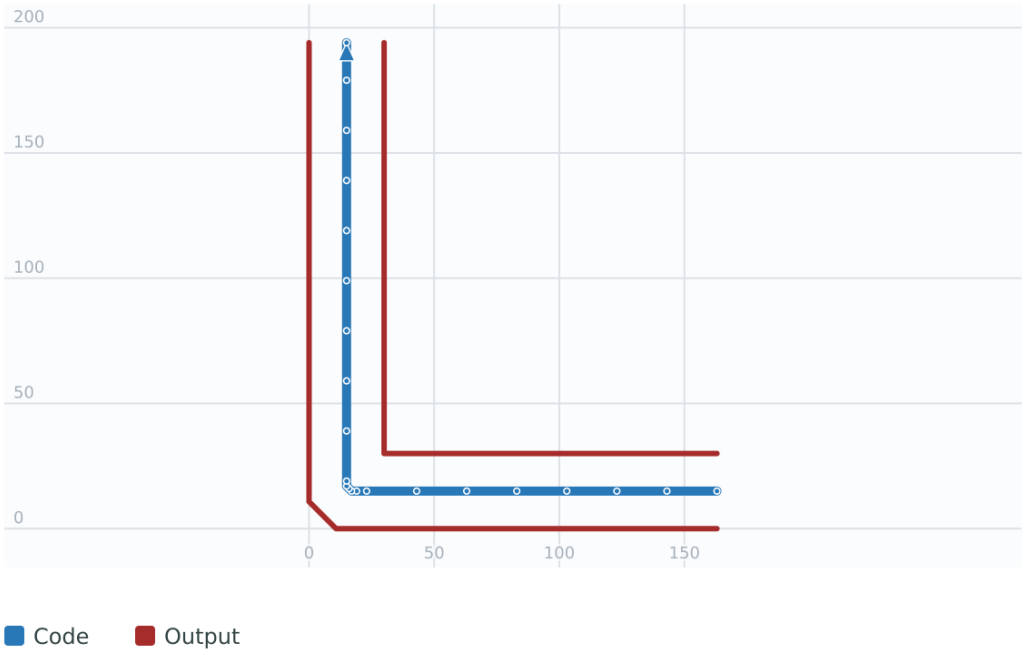
Code

```
SELECT ST_Collect(
    ST_OffsetCurve(geom, 15, 'quad_segs=4 join=mitre mitre_limit=2.2'),
    ST_OffsetCurve(geom, -15, 'quad_segs=4 join=mitre mitre_limit=2.2')
    )
    FROM ST_GeomFromText('LINESTRING(164 16,144 16,124 16,104 16,84 16,64 16,
    44 16,24 16,20 16,18 16,17 17,
    16 18,16 20,16 40,16 60,16 80,16 100,
    16 120,16 140,16 160,16 180,16 195)') As geom;
```

Output

```
MULTILINESTRING((164 1,11.786796564403577 1,1 11.786796564403577,1 195),
    (164 31,31 31,31 195))
```

Figure



相关信息

ST_Buffer

7.14.19 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

返回保区位于曲面内部的 `POINT`（`POLYGON`、`MULTIPOLYGON` 和 `CURVEPOLYGON`）。在 PostGIS 中，此函数也适用于区和点几何区形。

此方法区区了 SQL 1.1 的 OGC 区区功能区范。 s3.2.14.2 // s3.2.18.2

区方法区区了 SQL/MM 区范。SQL-MM 3：8.1.5、9.5.6。区区范区区区表面几何形状定区 ST_PointOnSurface。PostGIS 区展了功能以支持所有常区的几何区型。其他数据区（Oracle、DB2、ArcSDE）似乎区支持曲面的此功能。SQL Server 2008 支持所有常区的几何区型。

区函数支持 3d 并且不会区失 z-index。

示例

Point on surface of a MultiPoint.

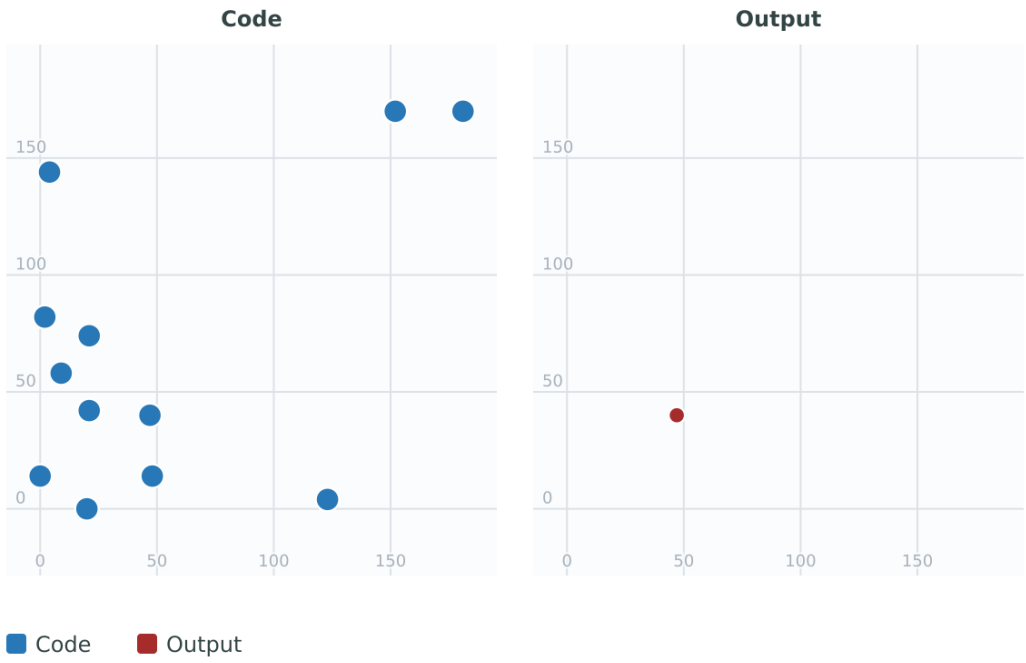
Code

```
SELECT ST_PointOnSurface(
  'MULTIPOINT (8 24,10 92,12 154,17 68,28 10,29 52,29 84,55 50,56 24,131 14,160 180,189 ↩
   180)');
```

Output

```
POINT(55 50)
```

Figure



Point on surface of a LineString.

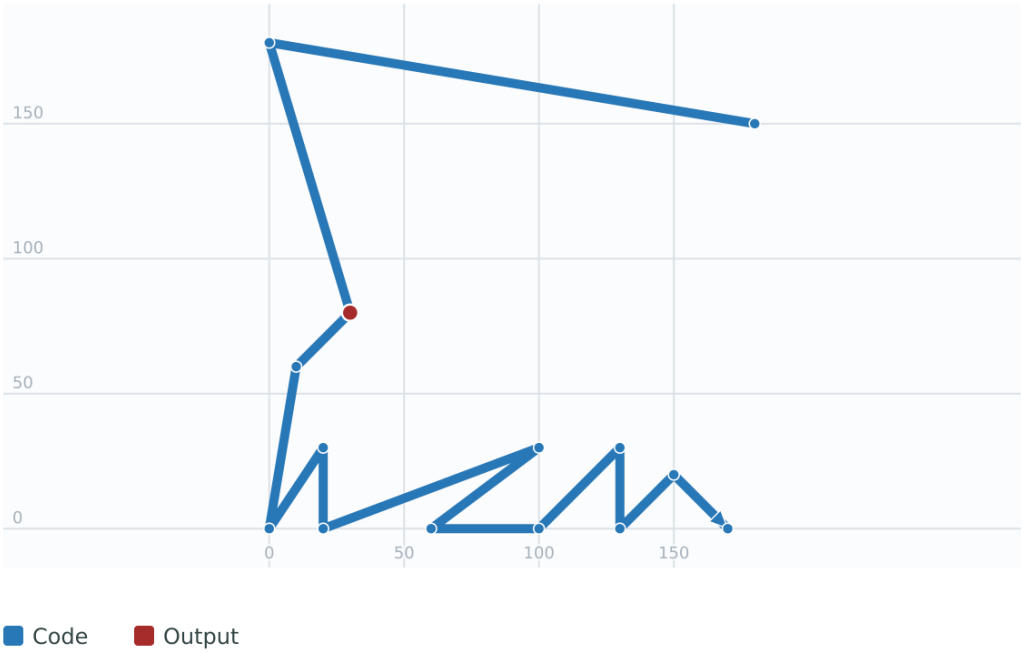
Code

```
SELECT ST_PointOnSurface(
  'LINESTRING (190 160,10 190,40 90,20 70,10 10,30 40,30 10,110 40,70 10,110 10,140 40,140 ↩
   10,160 30,180 10)');
```

Output

```
POINT(40 90)
```

Figure



Point on surface of a Polygon.

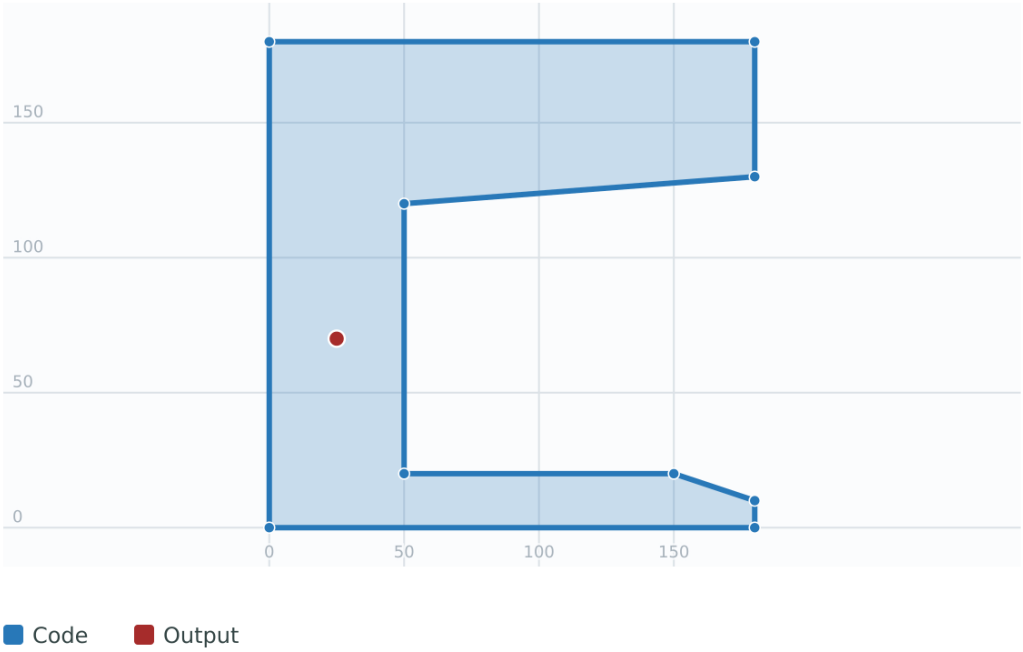
Code

```
SELECT ST_PointOnSurface(
  'POLYGON ((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))');
```

Output

```
POINT(35 80)
```

Figure



Point on surface of the highest-dimension components of a GeometryCollection.

Code

```
SELECT ST_PointOnSurface(
  'GEOMETRYCOLLECTION (POLYGON ((190 170,180 100,80 140,80 160,130 160,110 180,110 190,180 ↩
   180,190 170)),
   LINESTRING (80 120,120 20,140 70,150 30,180 50,190 10),
   MULTIPOINT (19 150,22 49,30 13,32 101,45 35,67 88,75 16))');
```

Output

```
POINT(133.571 150)
```

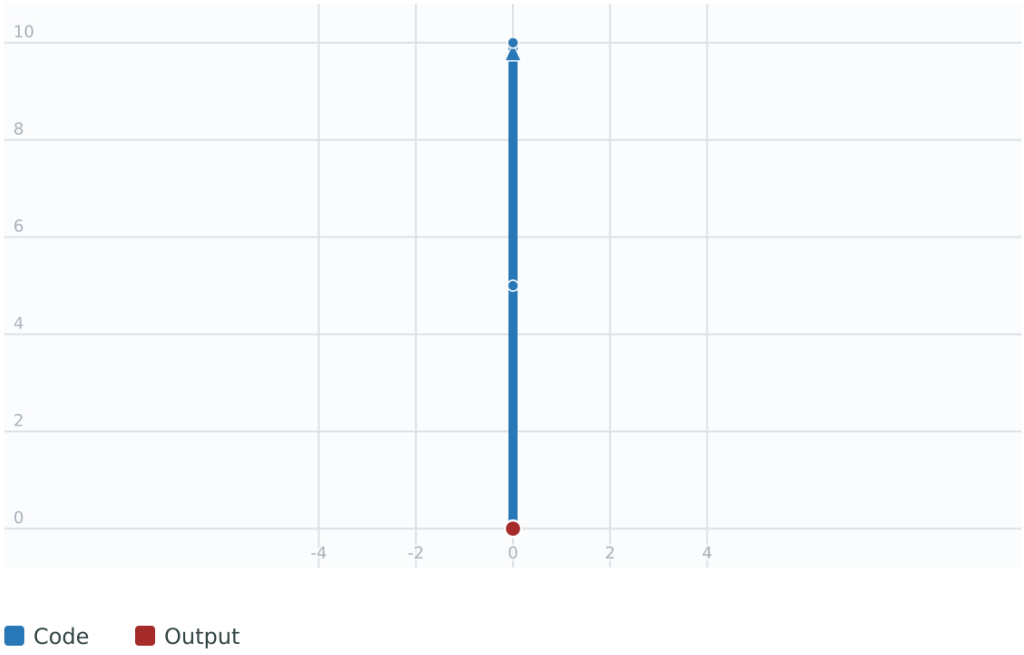
Figure



Code

```
SELECT ST_PointOnSurface('LINESTRING(0 5 1,0 0 1,0 10 2)');
```

Figure

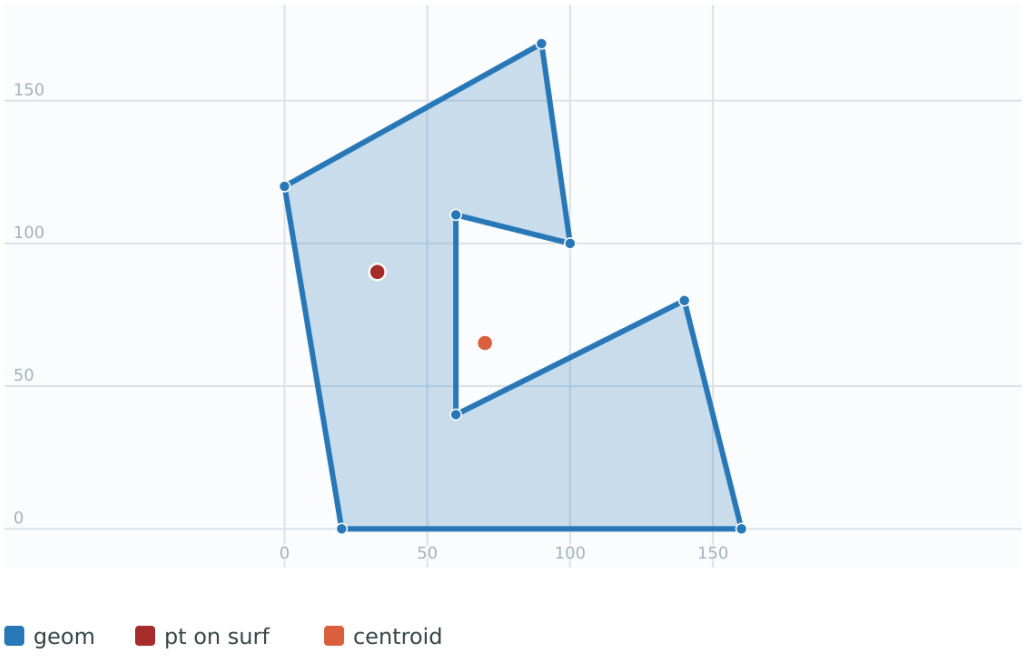


示例：ST_PointOnSurface 的⊠果保⊠位于多⊠形内，而 ST_Centroid⊠算的点可能位于多⊠形外。

Code

```
SELECT ST_PointOnSurface(geom) AS pt_on_surf, ST_Centroid(geom) AS centroid
    FROM (SELECT 'POLYGON ((130 120,120 190,30 140,50 20,190 20,
                        170 100,90 60,90 130,130 120))'::geometry AS geom) AS t;
```

Figure



相关信息

ST_Centroid, ST_MaximumInscribedCircle

7.14.20 ST_Polygonize

ST_Polygonize — 区算由一区几何区形的区条形成的多区形集合。

Synopsis

geometry **ST_Polygonize**(geometry set geomfield);
geometry **ST_Polygonize**(geometry[] geom_array);

描述

区建一个 GeometryCollection，其中包含由一区几何区形的区条形成的多区形。如果区入区条未形成任何多区形，区返回空的 GeometryCollection。

此函数区建覆盖所有分隔区域的多区形。如果区果旨在形成有效的多区形几何体，区使用 ST_BuildArea 来防止填充孔。



Note
区入区条必区正确区点才能使区功能正常工作。区了确保区入已区点化，区在多区形化之前在区入几何体上使用 ST_Node。



Note
使用外部工具可能很区区理几何集合。使用ST_Dump 将多区形区果区区区区独的多区形。

区个函数是由 GEOS 模区区行的。

可用性：1.0.0RC1

示例

Polygonize a set of input lines.

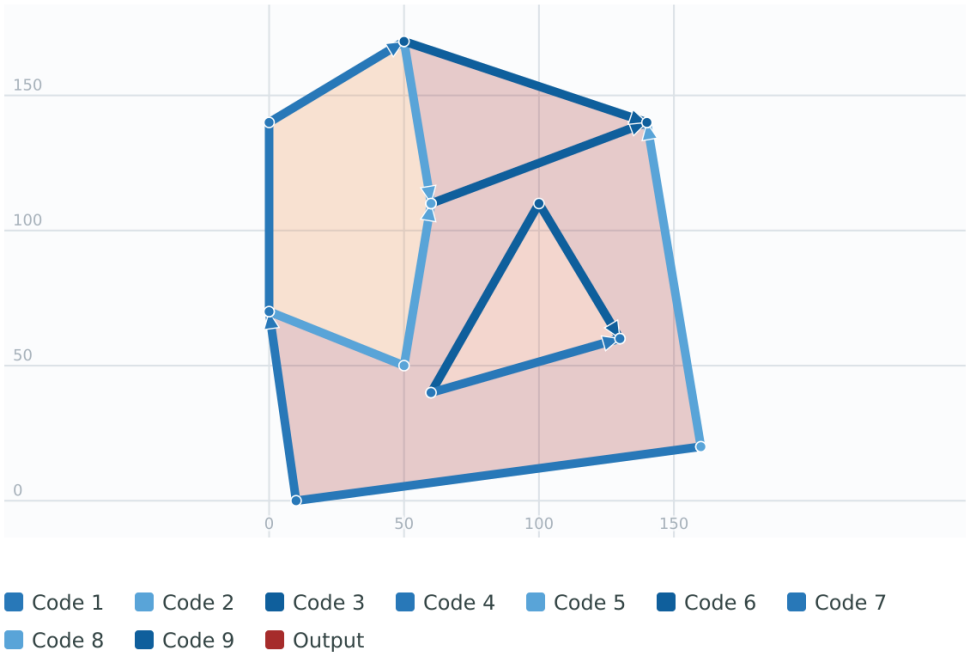
Code

```
WITH data(geom) AS (
  VALUES
    ('LINESTRING (180 40,30 20,20 90)'::geometry),
    ('LINESTRING (180 40,160 160)'::geometry),
    ('LINESTRING (80 60,120 130,150 80)'::geometry),
    ('LINESTRING (80 60,150 80)'::geometry),
    ('LINESTRING (20 90,70 70,80 130)'::geometry),
    ('LINESTRING (80 130,160 160)'::geometry),
    ('LINESTRING (20 90,20 160,70 190)'::geometry),
    ('LINESTRING (70 190,80 130)'::geometry),
    ('LINESTRING (70 190,160 160)'::geometry)
)
SELECT ST_Polygonize(geom)
FROM data;
```

Output

```
GEOMETRYCOLLECTION(POLYGON((180 40,30 20,20 90,70 70,80 130,160 160,180 40),(150 80,120 130,80 60,150 80)), POLYGON
    ((80 60,120 130,150 80,80 60)), POLYGON((80 130,70 70,20 90,20 160,70 190,80 130)), POLYGON((160 160,80 130,70
    190,160 160)))
```

Figure

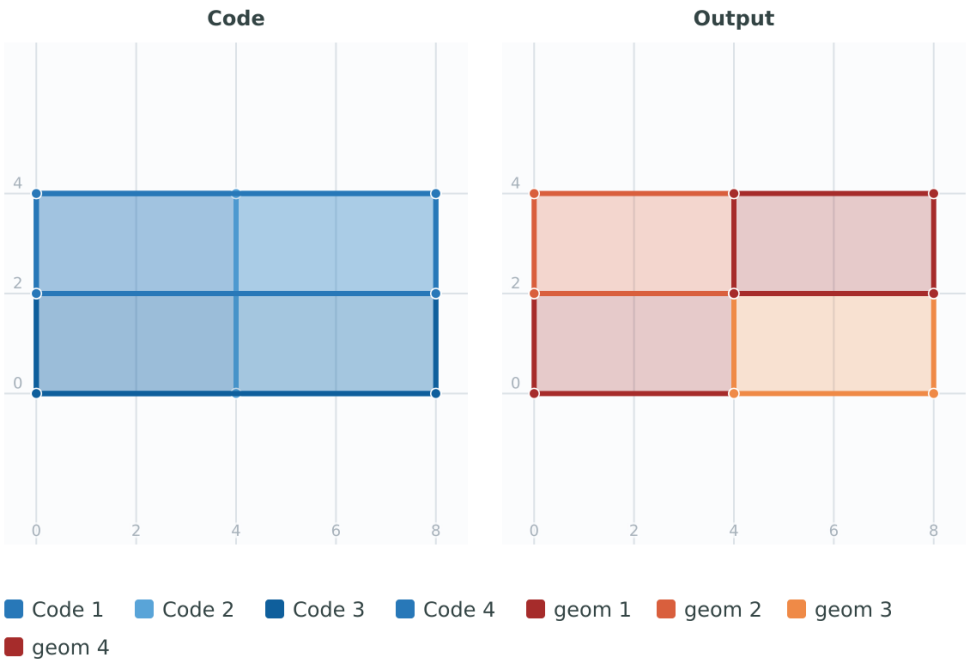


Polygonizing two polygon coverages and transferring attributes to the resultants:

Code

```
WITH poly_a(id, geom) AS (VALUES
  ('a1', 'POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0))'::geometry),
  ('a2', 'POLYGON ((4 0, 8 0, 8 4, 4 4, 4 0))'::geometry)
),
poly_b(id, geom) AS (VALUES
  ('b1', 'POLYGON ((0 0, 8 0, 8 2, 0 2, 0 0))'::geometry),
  ('b2', 'POLYGON ((0 2, 8 2, 8 4, 0 4, 0 2))'::geometry)
),
lines AS (
  SELECT ST_Boundary(geom) AS geom FROM poly_a
  UNION ALL
  SELECT ST_Boundary(geom) FROM poly_b
),
noded AS (
  SELECT ST_UnaryUnion(ST_Collect(geom)) AS geom FROM lines
),
cells AS (
  SELECT (ST_Dump(ST_Polygonize(geom))).geom AS geom FROM noded
)
SELECT a.id AS id_a,
       b.id AS id_b,
       c.geom AS geom
FROM cells AS c
LEFT JOIN poly_a AS a ON ST_Within(ST_PointOnSurface(c.geom), a.geom)
LEFT JOIN poly_b AS b ON ST_Within(ST_PointOnSurface(c.geom), b.geom)
WHERE a.id IS NOT NULL OR b.id IS NOT NULL
ORDER BY id_a, id_b;
```

Figure



多⊠形⊠串表：

Code

```
SELECT ST_Polygonize(geom_4269) As geomtextrep
FROM (SELECT geom_4269 FROM ma.suffolk_edges) As foo;
```

Output

```
geomtextrep
-------------
 SRID=4269;GEOMETRYCOLLECTION(POLYGON((-71.040878 42.285678,-71.040943 42.2856,-71.04096  ↩
    42.285752,-71.040878 42.285678)),
 POLYGON((-71.17166 42.353675,-71.172026 42.354044,-71.17239 42.354358,-71.171794  ↩
    42.354971,-71.170511 42.354855,
 -71.17112 42.354238,-71.17166 42.353675)))
(1 row)
```

Use ST_Dump to dump out the polygonize geoms into individual polygons.

Code

```
SELECT (ST_Dump(t.polycoll)).geom AS geomtextrep
FROM (SELECT ST_Polygonize(geom_4269) AS polycoll
    FROM (SELECT geom_4269 FROM ma.suffolk_edges)
        As foo) AS t;
```

Output

```
geomtextrep
-------------
 SRID=4269;POLYGON((-71.040878 42.285678,-71.040943 42.2856,-71.04096 42.285752,
-71.040878 42.285678))
 SRID=4269;POLYGON((-71.17166 42.353675,-71.172026 42.354044,-71.17239 42.354358,
-71.171794 42.354971,-71.170511 42.354855,-71.17112 42.354238,-71.17166 42.353675))
(2 rows)
```

相关信息

ST_BuildArea, ST_Boundary, ST_Collect, ST_Dump, ST_Node, ST_PointOnSurface, ST_UnaryUnion, ST_Within

7.14.21 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

Code

```
SELECT ST_ReducePrecision('POINT(1.412 19.323)', 0.1);
```

Output

```
POINT(1.4 19.3)
```

Figure



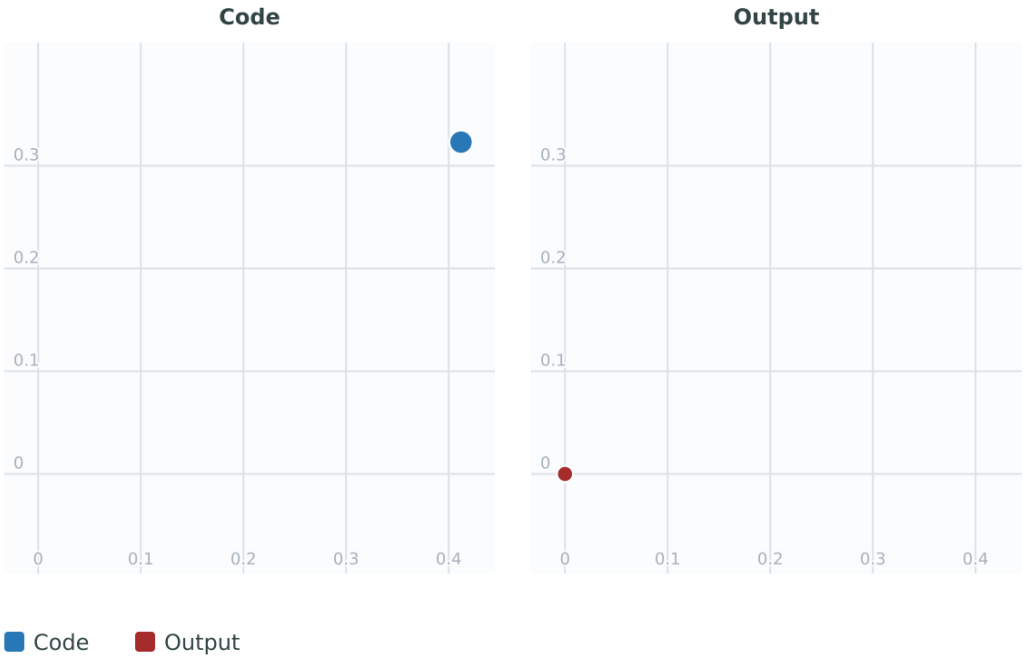
Code

```
SELECT ST_ReducePrecision('POINT(1.412 19.323)', 1.0);
```

Output

```
POINT(1 19)
```

Figure



Code

```
SELECT ST_ReducePrecision('POINT(1.412 19.323)', 10);
```

Output

```
POINT(0 20)
```

Figure



降低精度可以⌧少⌧点数量

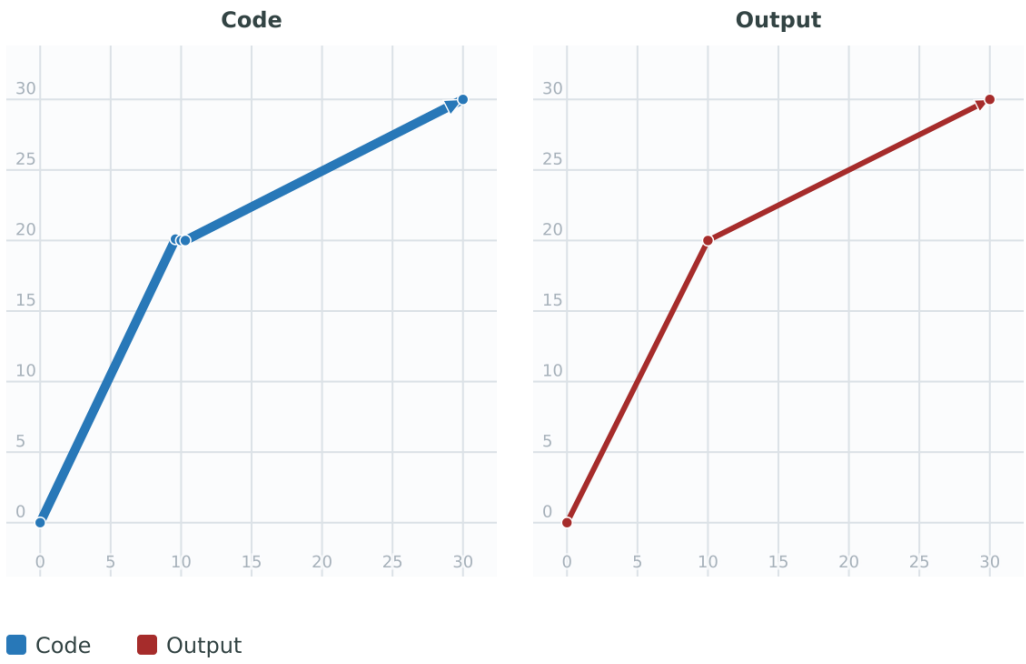
Code

```
SELECT ST_ReducePrecision('LINESTRING (10 10,19.6 30.1,20 30,20.3 30,40 40)', 1);
```

Output

```
LINESTRING (10 10, 20 30, 40 40)
```

Figure



如果需要的⊠，精度降低会分割多⊠形以确保有效性

Code

```
WITH data(geom) AS (
  VALUES ('POLYGON ((10 10,60 60.1,70 30,40 40,50 10,10 10))'::geometry)
)
SELECT geom AS input,
       ST_ReducePrecision(geom, 10) AS reduced
FROM data;
```

Output

```
-[ RECORD 1 ]-----------------------------------------------
input   | POLYGON((10 10,60 60.1,70 30,40 40,50 10,10 10))
reduced | MULTIPOLYGON(((10 10,40 40,50 10,10 10)),((40 40,60 60,70 30,40 40)))
```

Figure



相关信息

ST_SnapToGrid, ST_Simplify, ST_SimplifyVW

7.14.22 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

Adjacent shared path segments may be returned as separate or merged LineStrings depending on the GEOS version.

Find the shared paths of a MultiLineString and a LineString.

Code

```
SELECT ST_SharedPaths(ST_GeomFromText('MULTILINESTRING((26 125,26 200,126 200,126 125,26 ↩
    125),
       (51 150,101 150,76 175,51 150))'),
   ST_GeomFromText('LINESTRING(151 100,126 156.25,126 125,101 150,76 175)')
   )
```

Output

```
GEOMETRYCOLLECTION(MULTILINESTRING((126 156.25,126 125),
(101 150,76 175)),MULTILINESTRING EMPTY)
```

Figure



Flip the LineString orientation to return the same paths in the opposite-direction member.

Code

```
SELECT ST_SharedPaths(ST_GeomFromText('LINESTRING(76 175,101 150,126 125,126 156.25,151 ↩
    100)'),
   ST_GeomFromText('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),
       (51 150,101 150,76 175,51 150))')
    )
```

Output

```
GEOMETRYCOLLECTION(MULTILINESTRING EMPTY,
MULTILINESTRING((76 175,101 150),(126 125,126 156.25)))
```

Figure



相关信息

ST_Dump, ST_GeometryN, ST_NumGeometries

7.14.23 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

A circle simplified with a medium tolerance becomes an octagon, and with a large tolerance disappears.

Code

```
SELECT ST_NPoints(geom) AS np_before,
     ST_NPoints(ST_Simplify(geom, 0.1)) AS np01_notbadcircle,
     ST_NPoints(ST_Simplify(geom, 0.5)) AS np05_notquitecircle,
     ST_NPoints(ST_Simplify(geom, 1)) AS np1_octagon,
     ST_NPoints(ST_Simplify(geom, 10)) AS np10_gone,
     (ST_Simplify(geom, 100) is null) AS  np100_geometrygoesaway
FROM (SELECT ST_Buffer('POINT(1 3)', 10, 12) As geom) AS t;
```

Output

```
-[ RECORD 1 ]------------
np_before                | 49
np01_notbadcircle        | 33
np05_notquitecircle      | 17
np1_octagon              | 9
np10_gone                | null
np100_geometrygoesaway   | t
```

☒化一☒☒。☒化后，☒可能会相交。

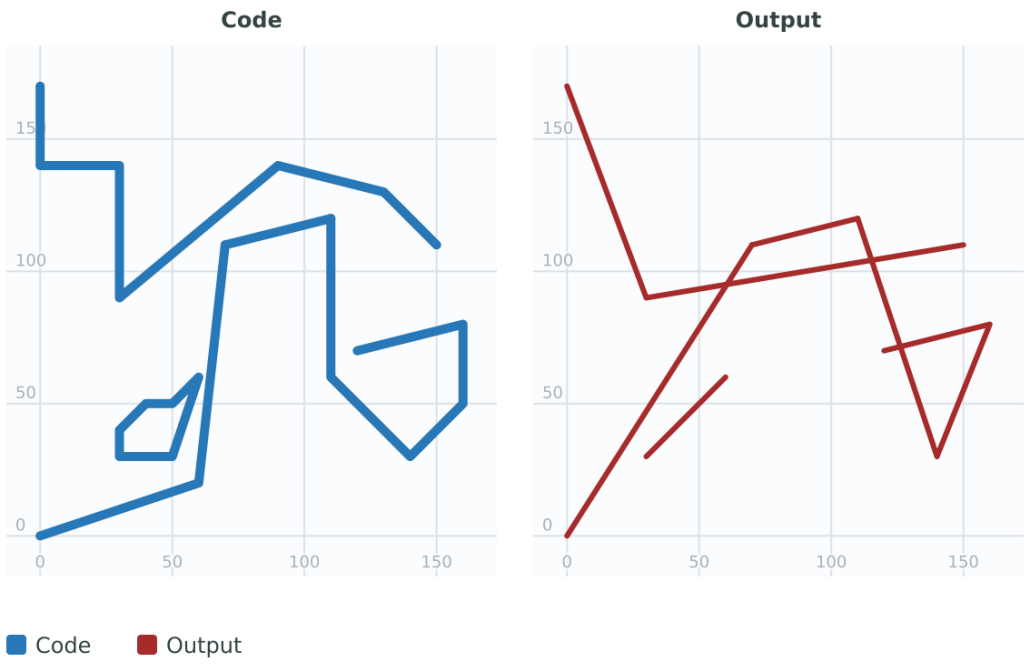
Code

```
SELECT ST_Simplify('MULTILINESTRING ((20 180,20 150,50 150,50 100,110 150,150 140,170 120) ↩
    ,(20 10,80 30,90 120),(90 120,130 130),(130 130,130 70,160 40,180 60,180 90,140 80),(50 ↩
    40,70 40,80 70,70 60,60 60,50 50,50 40))'::geometry,
40);
```

Output

```
MULTILINESTRING((20 180,50 100,170 120),(20 10,90 120),(90 120,130 130),(130 130,160 40,180 90,140 80),(50 40,80 70,50 ↩
     40))
```

Figure



⌧化一个 MultiPolygon。多⌧形的⌧果可能是无效的。

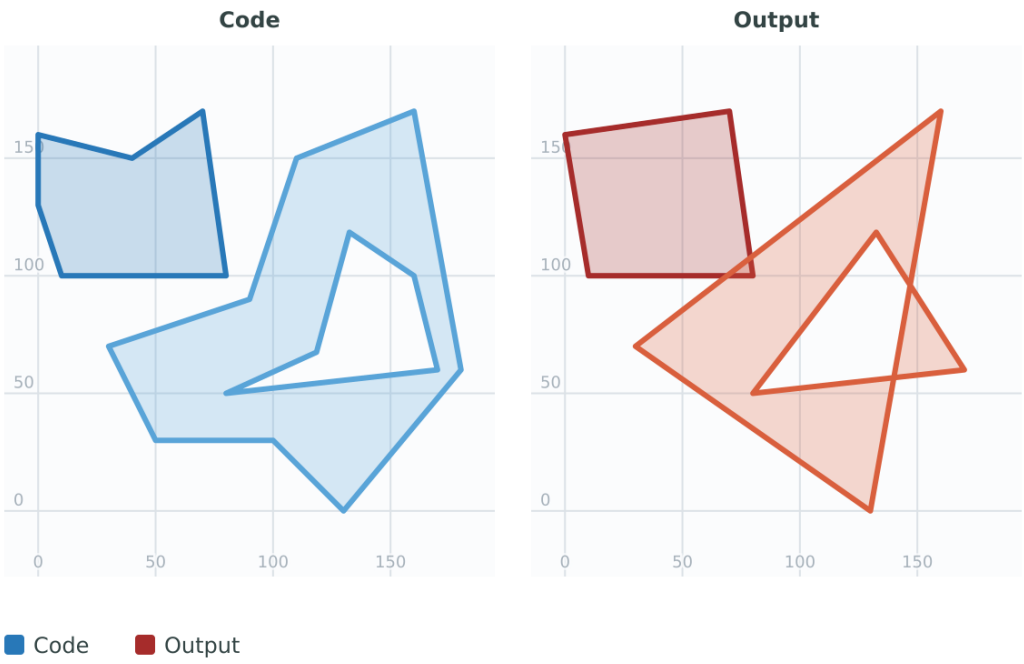
Code

```
SELECT ST_Simplify('MULTIPOLYGON (((90 110,80 180,50 160,10 170,10 140,20 110,90 110)),((40 ↩
    80,100 100,120 160,170 180,190 70,140 10,110 40,60 40,40 80),(180 70,170 110,142.5 ↩
   128.5,128.5 77.5,90 60,180 70)))'::geometry,
40);
```

Output

```
MULTIPOLYGON(((90 110,80 180,10 170,20 110,90 110)),((40 80,170 180,140 10,40 80),(180 70,142.5 128.5,90 60,180 70)))
```

Figure



相关信息

ST_IsSimple, ST_SimplifyPreserveTopology, ST_SimplifyVW, ST_CoverageSimplify, Topology ST_Simplify

7.14.24 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

For the same example as ST_Simplify, ST_SimplifyPreserveTopology prevents the geometry from disappearing.

Code

```
SELECT  ST_NPoints(geom) AS np_before,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.1)) AS np01_notbadcircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 0.5)) AS np05_notquitecircle,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 1))   AS np1_octagon,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 10))  AS np10_minimal,
        ST_NPoints(ST_SimplifyPreserveTopology(geom, 100)) AS np100_minimal
FROM (SELECT ST_Buffer('POINT(1 3)', 10, 12) AS geom) AS t;
```

Output

```
np_before | np01_notbadcircle | np05_notquitecircle | np1_octagon | np10_minimal |  ↩
    np100_minimal
-----------+-------------------+---------------------+-------------+--------------+-------------- ↩

        49 |                33 |                  17 |           9 |            4 |         ↩
         4
```

⊠化一⊠⊠，保留非相交⊠的拓扑。

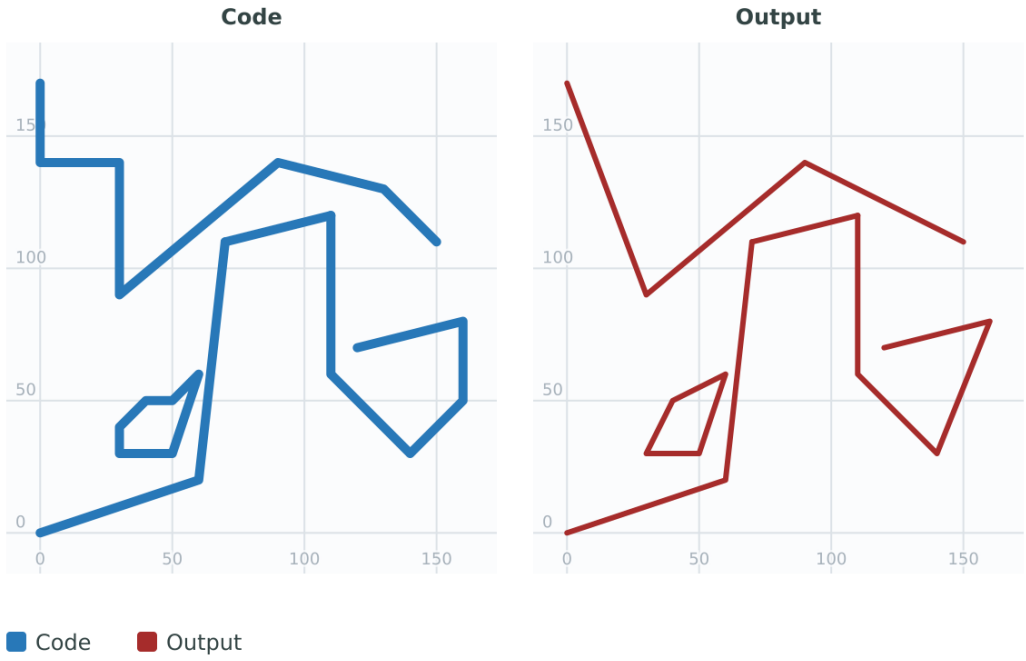
Code

```
SELECT ST_SimplifyPreserveTopology('MULTILINESTRING ((20 180,20 150,50 150,50 100,110  ↩
    150,150 140,170 120),(20 10,80 30,90 120),(90 120,130 130),(130 130,130 70,160 40,180  ↩
    60,180 90,140 80),(50 40,70 40,80 70,70 60,60 60,50 50,50 40))'::geometry,
40);
```

Output

```
MULTILINESTRING((20 180,50 100,110 150,170 120),(20 10,80 30,90 120),(90 120,130 130),(130 130,130 70,160 40,180  ↩
    90,140 80),(50 40,70 40,80 70,60 60,50 40))
```

Figure



⊠化一个 MultiPolygon，保留外壳和内⊠的拓扑。

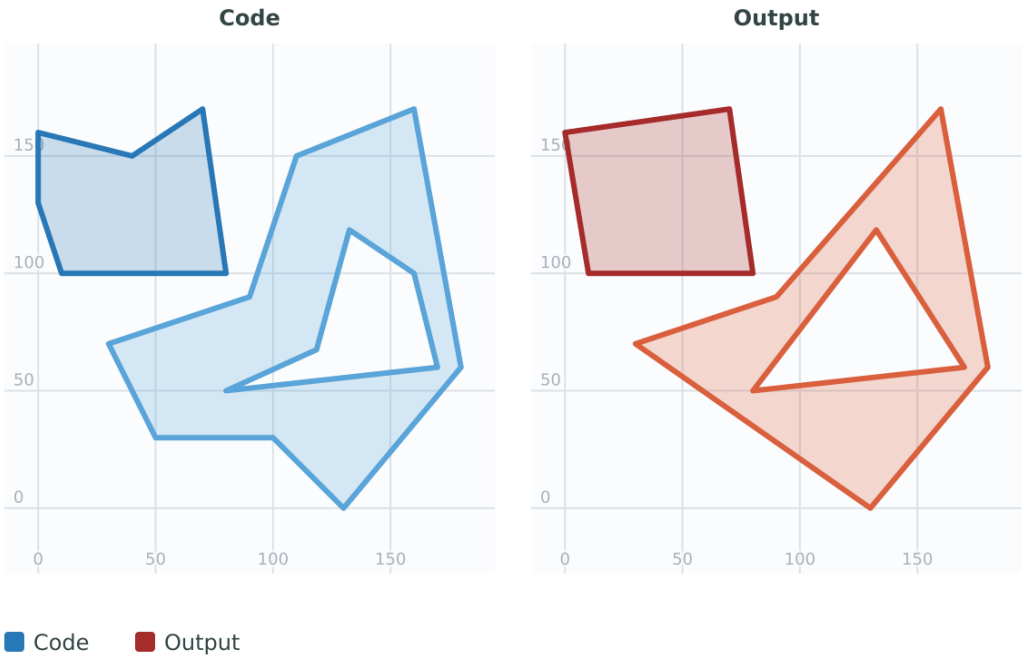
Code

```
SELECT ST_SimplifyPreserveTopology('MULTIPOLYGON (((90 110,80 180,50 160,10 170,10 140,20 ↩
    110,90 110)),((40 80,100 100,120 160,170 180,190 70,140 10,110 40,60 40,40 80),(180 ↩
    70,170 110,142.5 128.5,128.5 77.5,90 60,180 70)))'::geometry,
40);
```

Output

```
MULTIPOLYGON(((90 110,80 180,10 170,20 110,90 110)),((40 80,100 100,170 180,190 70,140 10,40 80),(180 70,142.5 ↩
    128.5,90 60,180 70)))
```

Figure



相关信息

ST_Simplify, ST_SimplifyVW, ST_CoverageSimplify

7.14.25 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。

⊠点数的⊠少可以通⊠ vertex_fraction 参数⊠行控制。⊠参数的取⊠范⊠是 0 到 1。⊠小的⊠会使⊠化⊠程更加深入，⊠点数⊠少，凹性降低。⊠于外部和内部的凹包，当 vertex_fraction ⊠⊠ 1.0 ⊠，会生成与原始几何体相同的形状；而当⊠⊠ 0.0 ⊠，⊠于外部凹包会生成凸包（⊠一多⊠形），而⊠于内部凹包⊠会生成三角形。

⊠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

Outer hull of a Polygon.

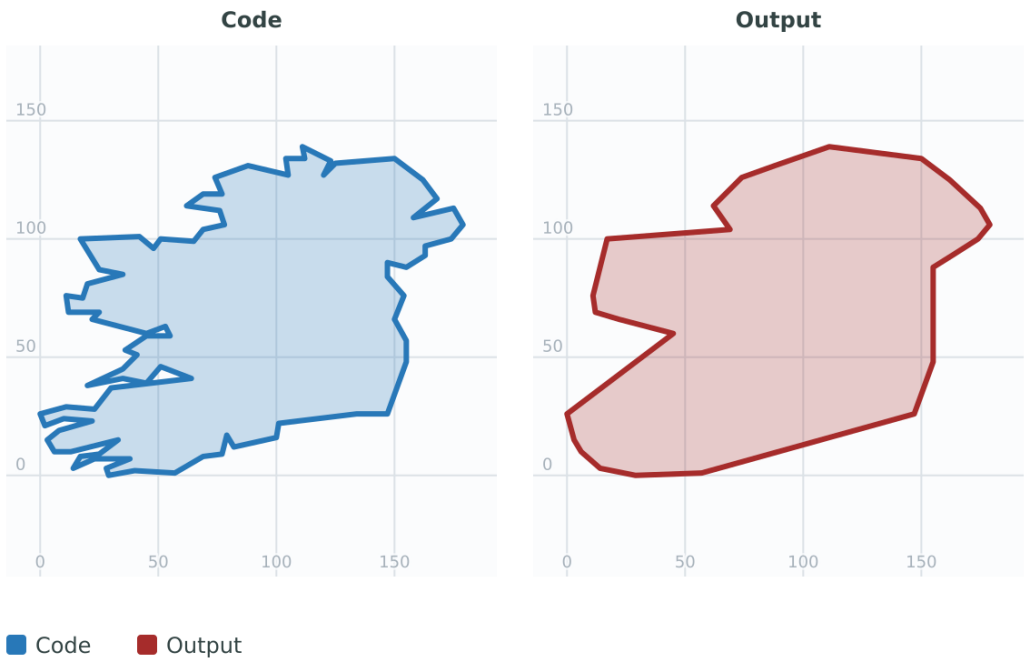
Code

```
SELECT ST_SimplifyPolygonHull(
  'POLYGON ((131 158,136 163,161 165,173 156,179 148,169 140,186 144,190 137,185 131,174 ↩
    128,174 124,166 119,158 121,158 115,165 107,161 97,166 88,166 79,158 57,145 57,112 ↩
    53,111 47,93 43,90 48,88 40,80 39,68 32,51 33,40 31,39 34,49 38,34 38,25 34,28 39,36 ↩
    40,44 46,24 41,17 41,14 46,19 50,33 54,21 55,13 52,11 57,22 60,34 59,41 68,75 72,62 ↩
    77,56 70,46 72,31 69,46 76,52 82,47 84,56 90,66 90,64 94,56 91,33 97,36 100,23 100,22 ↩
    107,29 106,31 112,46 116,36 118,28 131,53 132,59 127,62 131,76 130,80 135,89 137,87 ↩
    143,73 145,80 150,88 150,85 157,99 162,116 158,115 165,123 165,122 170,134 164,131 158)) ↩
    '::geometry,
  0.3
);
```

Output

```
POLYGON((161 165,173 156,186 144,190 137,185 131,174 124,166 119,166 79,158 57,68 32,40 31,25 34,17 41,14 46,11 57,56 ↩
    91,33 97,23 100,22 107,28 131,80 135,73 145,85 157,99 162,122 170,161 165))
```

Figure



Inner hull of a Polygon.

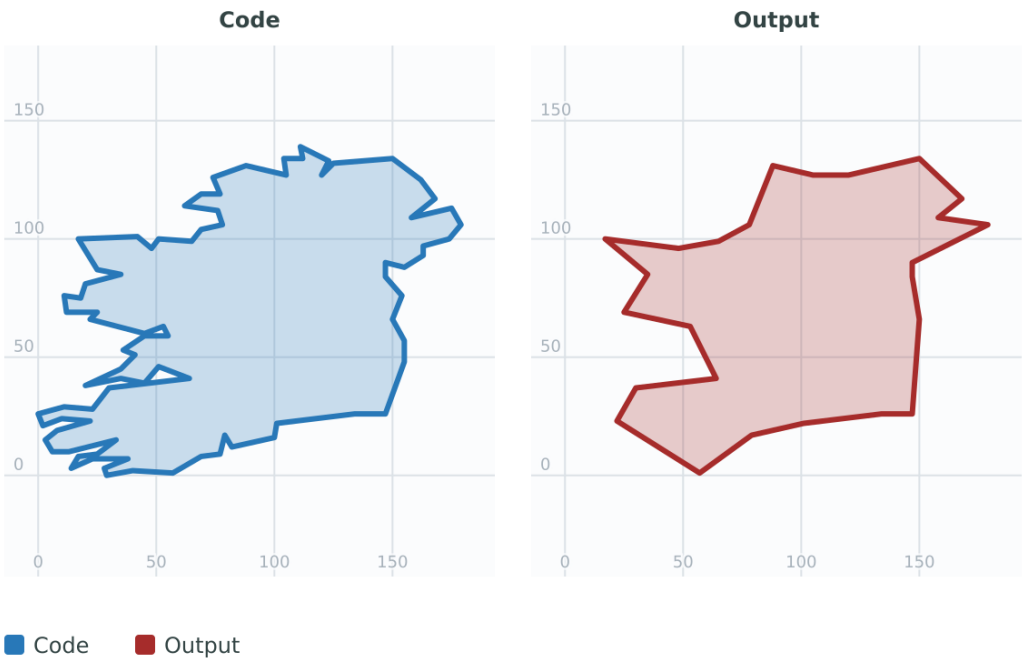
Code

```
SELECT ST_SimplifyPolygonHull(
  'POLYGON ((131 158,136 163,161 165,173 156,179 148,169 140,186 144,190 137,185 131,174 ↩
    128,174 124,166 119,158 121,158 115,165 107,161 97,166 88,166 79,158 57,145 57,112 ↩
    53,111 47,93 43,90 48,88 40,80 39,68 32,51 33,40 31,39 34,49 38,34 38,25 34,28 39,36 ↩
    40,44 46,24 41,17 41,14 46,19 50,33 54,21 55,13 52,11 57,22 60,34 59,41 68,75 72,62 ↩
    77,56 70,46 72,31 69,46 76,52 82,47 84,56 90,66 90,64 94,56 91,33 97,36 100,23 100,22 ↩
    107,29 106,31 112,46 116,36 118,28 131,53 132,59 127,62 131,76 130,80 135,89 137,87 ↩
    143,73 145,80 150,88 150,85 157,99 162,116 158,115 165,123 165,122 170,134 164,131 158)) ↩
    '::geometry,
  0.3, false
);
```

Output

```
POLYGON((131 158,116 158,99 162,89 137,76 130,59 127,28 131,46 116,36 100,64 94,75 72,41 68,33 54,68 32,90 48,112 ↩
    53,145 57,158 57,161 97,158 115,158 121,190 137,169 140,179 148,161 165,131 158))
```

Figure

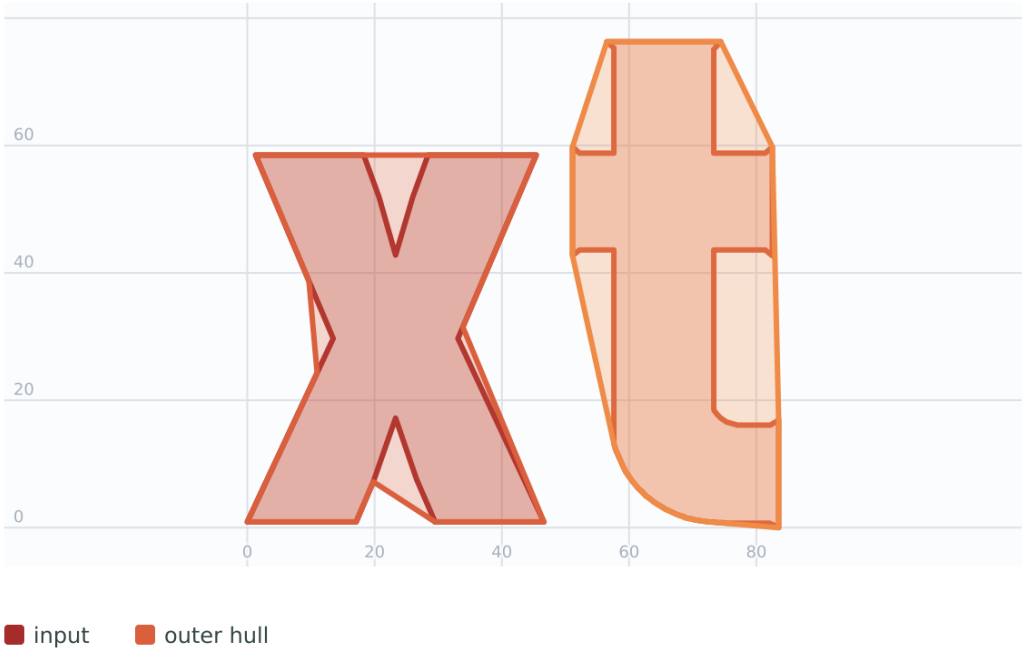


Outer hull simplification of a MultiPolygon, with segmentization.

Code

```
WITH letters AS (
  SELECT ST_Letters('xt') AS geom
)
SELECT geom AS input,
  ST_SimplifyPolygonHull(ST_Segmentize(geom, 2.0), 0.1) AS outer_hull
FROM letters;
```

Figure



相关信息

ST_ConvexHull, ST_SimplifyVW, ST_ConcaveHull, ST_Segmentize

7.14.26 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

Returns a simplified representation of a geometry using the Visvalingam-Whyatt algorithm. The simplification `tolerance` is an area value, in squared units of the input SRS. Simplification removes vertices which form "corners" with area less than the tolerance. The result may not be valid even if the input is.

⍰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。

This is a general-purpose simplification function. It may remove narrow polygon spikes when their effective area is below the tolerance, but it is not a dedicated spike-removal or polygon-repair operation. Use ST_IsValid to check the result and ST_MakeValid to repair invalid polygonal output if needed.

⍰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。

⍰函数⍰理 3D，第三⍰将影⍰⍰果。

可用性：2.2.0

示例

This polygon has a narrow spike whose effective area is small enough to be removed by a tolerance of 20 square units.

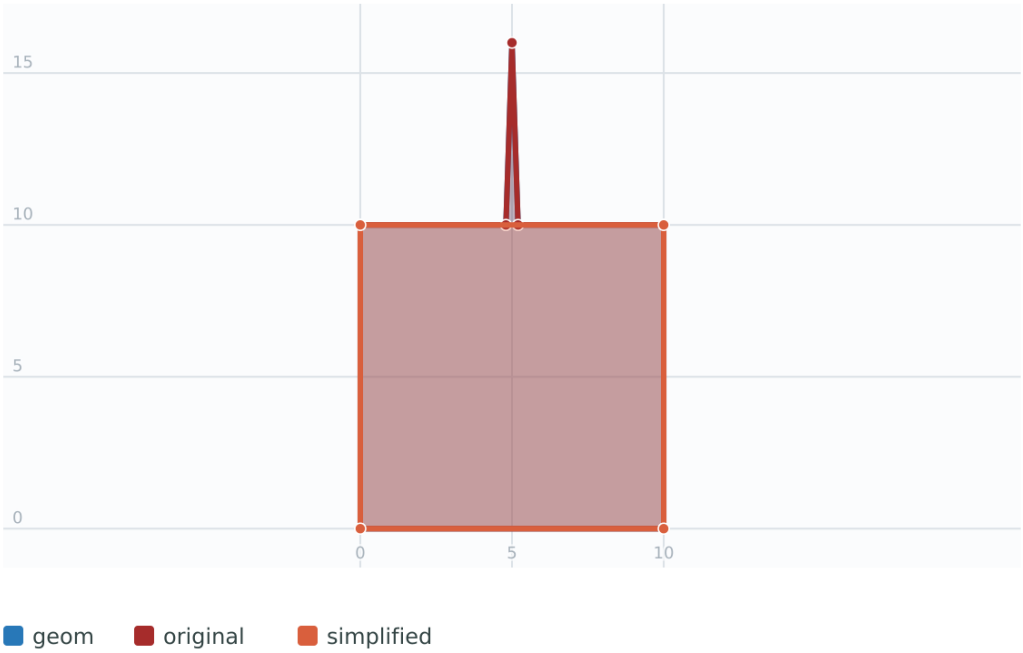
Code

```
SELECT geom AS original, ST_SimplifyVW(geom, 20) AS simplified
FROM (SELECT 'POLYGON((0 0,10 0,10 10,5.2 10,5 16,4.8 10,0 10,0 0))'::geometry AS geom) AS ↩
    t;
```

Output

```
POLYGON((0 0,10 0,10 10,5.2 10,5 16,4.8 10,0 10,0 0)) | POLYGON((0 0,10 0,10 10,0 10,0 0))
```

Figure



A LineString is simplified with a minimum-area tolerance of 30. For a linear simplification using ST_Simplify, a comparable tolerance would be a length in input-SRS units, so the value here is roughly that linear tolerance squared.

Code

```
SELECT ST_SimplifyVW(geom, 30) simplified
  FROM (SELECT 'LINESTRING(5 2,3 8,6 20,7 25,10 10)'::geometry AS geom) AS t;
```

Output

```
LINESTRING(5 2,7 25,10 10)
```

Figure



⊠一条⊠⊠行⊠化。

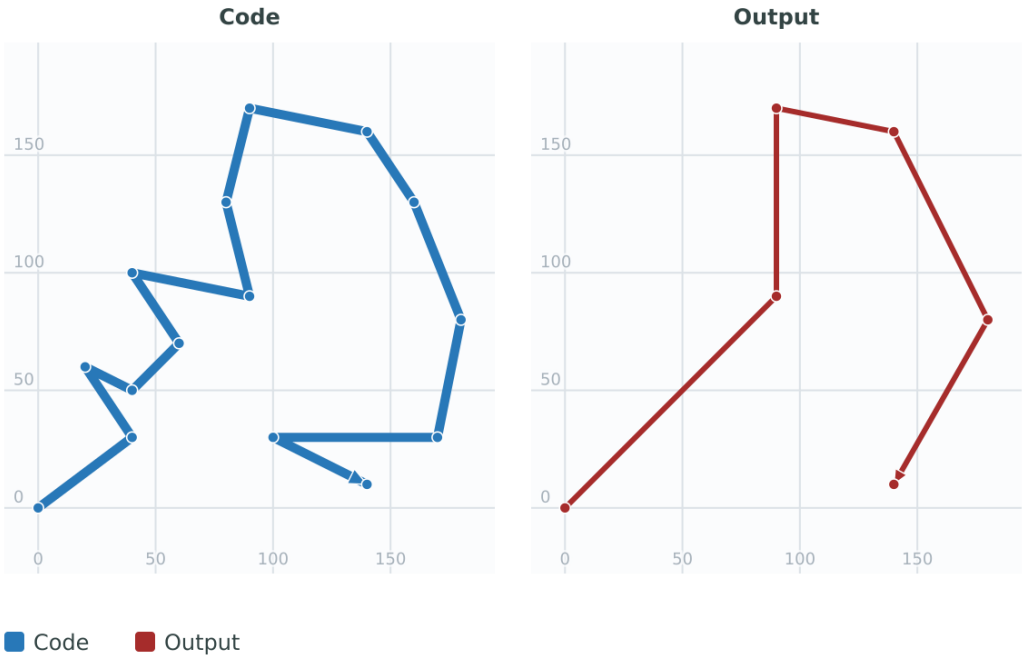
Code

```
SELECT ST_SimplifyVW('LINESTRING (10 10,50 40,30 70,50 60,70 80,50 110,100 100,90 140,100 ↩
    180,150 170,170 140,190 90,180 40,110 40,150 20)'::geometry,
1600);
```

Output

```
LINESTRING(10 10,100 100,100 180,150 170,190 90,150 20)
```

Figure



⊠多⊠形⊠行⊠化。

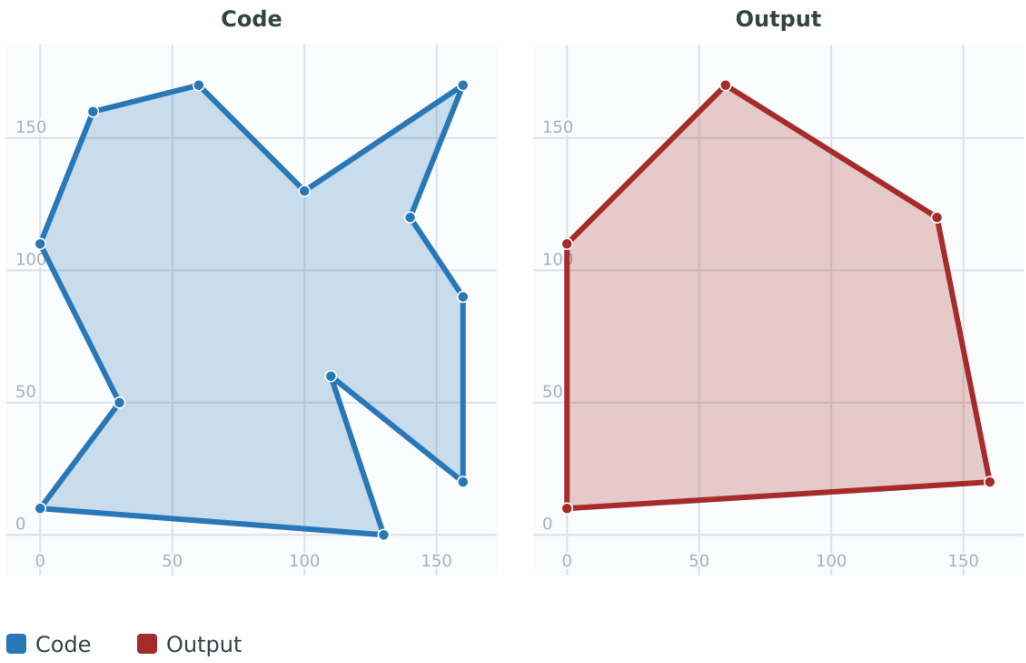
Code

```
SELECT ST_SimplifyVW('POLYGON ((20 20,50 60,20 120,40 170,80 180,120 140,180 180,160 ↩
    130,180 100,180 30,130 70,150 10,20 20))'::geometry,
1600);
```

Output

```
POLYGON((20 20,20 120,80 180,160 130,180 30,20 20))
```

Figure



相关信息

ST_SetEffectiveArea, ST_Simplify, ST_SimplifyPreserveTopology, ST_CoverageSimplify, Topology ST_Simpl

7.14.27 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。

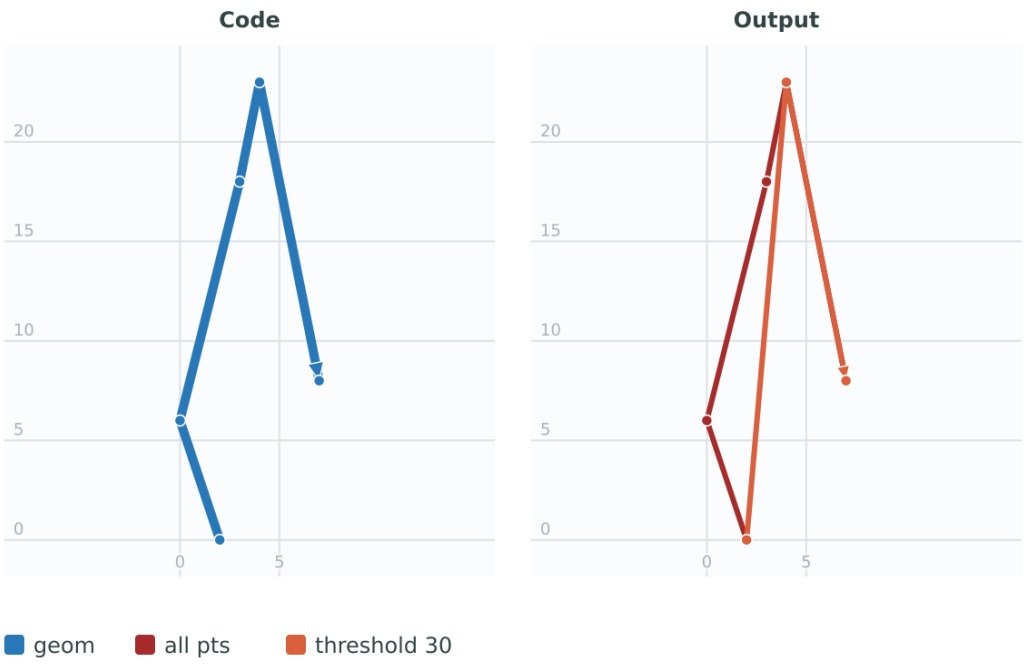
Code

```
SELECT ST_SetEffectiveArea(geom) AS all_pts,
       ST_SetEffectiveArea(geom, 30) AS threshold_30
FROM (SELECT  'LINESTRING(5 2,3 8,6 20,7 25,10 10)'::geometry geom) As foo;
```

Output

```
-[ RECORD 1 ]----------------
all_pts      | LINESTRING M (5 2 3.4e+38,3 8 29,6 20 1.5,7 25 49.5,10 10 3.4e+38)
threshold_30 | LINESTRING M (5 2 3.4e+38,7 25 49.5,10 10 3.4e+38)
```

Figure



相关信息

ST_SimplifyVW

7.14.28 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边内角。它提供了多边形的“最佳边量”三角边量。

可用性：3.3.0。

需要 GEOS >= 3.11.0。

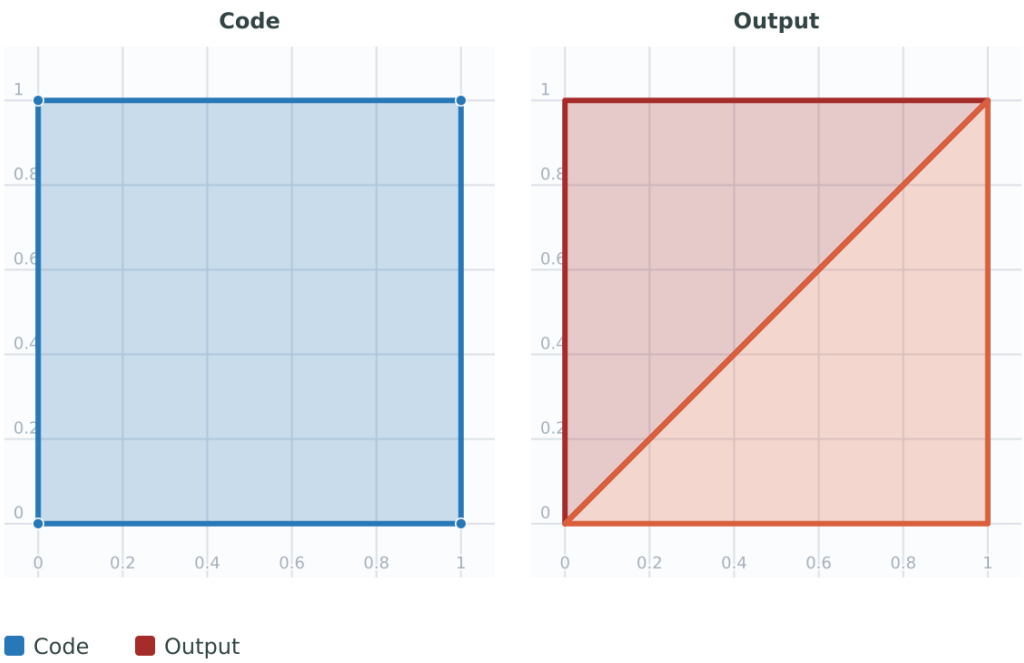
示例

正方形的三角剖分。

Code

```
SELECT ST_TriangulatePolygon('POLYGON((0 0,0 1,1 1,1 0,0 0))');
```

Figure

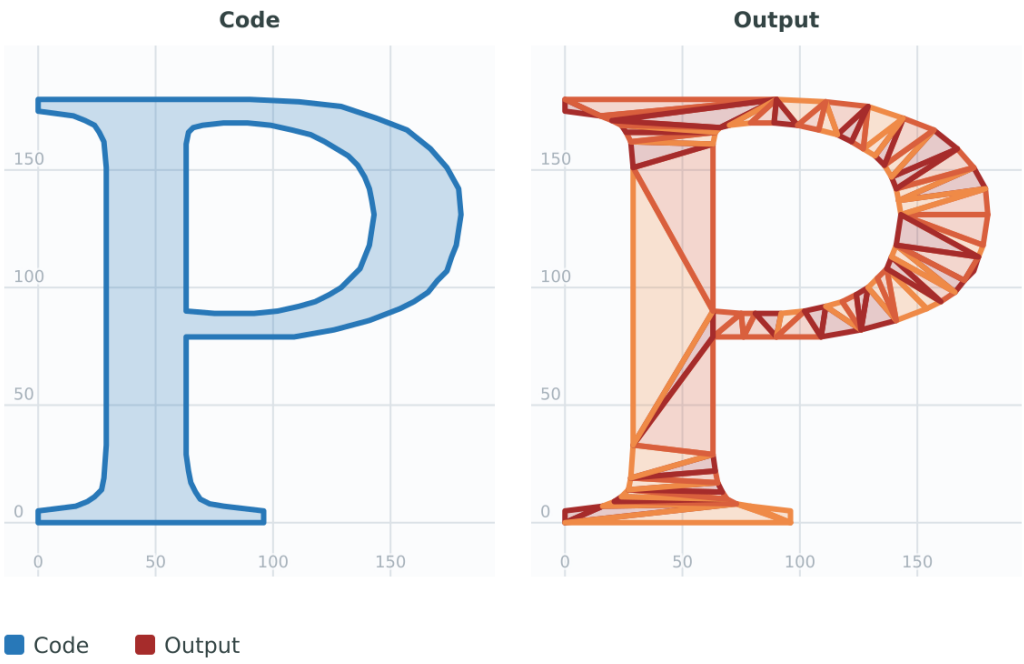


字母 P 的三角剖分。

Code

```
SELECT ST_TriangulatePolygon('POLYGON ((26 17,31 19,34 21,37 24,38 29,39 43,39 161,38 ↩
    172,36 176,34 179,30 181,25 183,10 185,10 190,100 190,121 189,139 187,154 182,167 ↩
    177,177 169,184 161,189 152,190 141,188 128,186 123,184 117,180 113,176 108,170 104,164 ↩
    101,151 96,136 92,119 89,100 89,86 89,73 89,73 39,74 32,75 27,77 23,79 20,83 18,89 ↩
    17,106 15,106 10,10 10,10 15,26 17),(152 147,151 152,149 157,146 162,142 166,137 169,132 ↩
     172,126 175,118 177,109 179,99 180,89 180,80 179,76 178,74 176,73 171,73 100,85 99,91 ↩
    99,102 99,112 100,121 102,128 104,134 107,139 110,143 114,147 118,149 123,151 128,153 ↩
    141,152 147))'
);
```

Figure

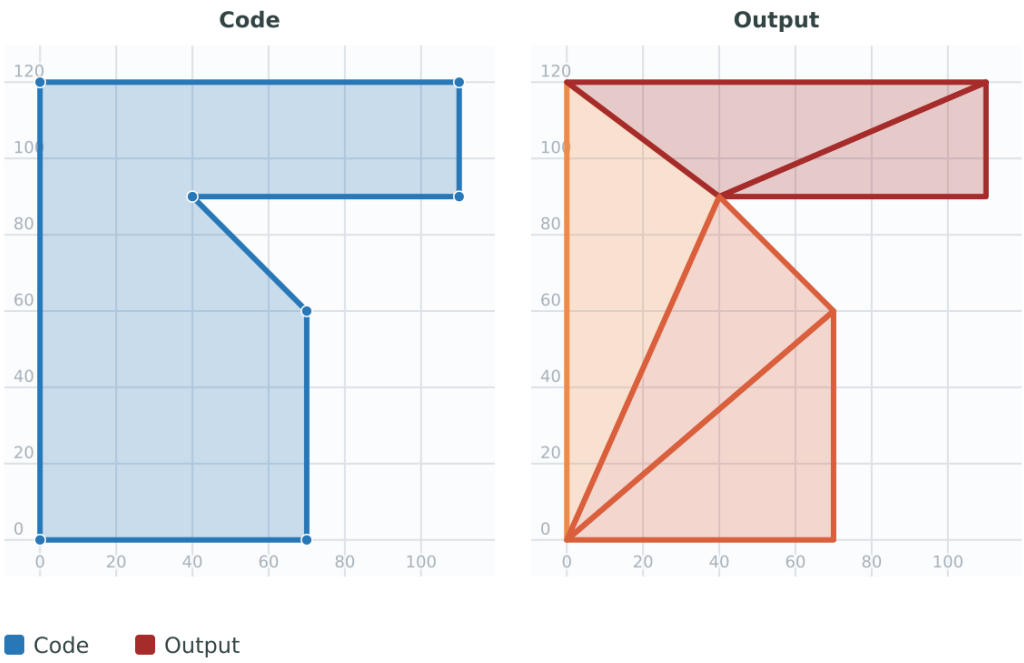


Triangulation of the polygon used in the CG_Tesselate example.

Code

```
SELECT ST_TriangulatePolygon('POLYGON (( 10 190,10 70,80 70,80 130,50 160,120 160,120 ↩
    190,10 190 ))'::geometry
);
```

Figure



相关信息

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, CG_Tesselate

7.14.29 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 diagram lines with a tolerance of 30 units.

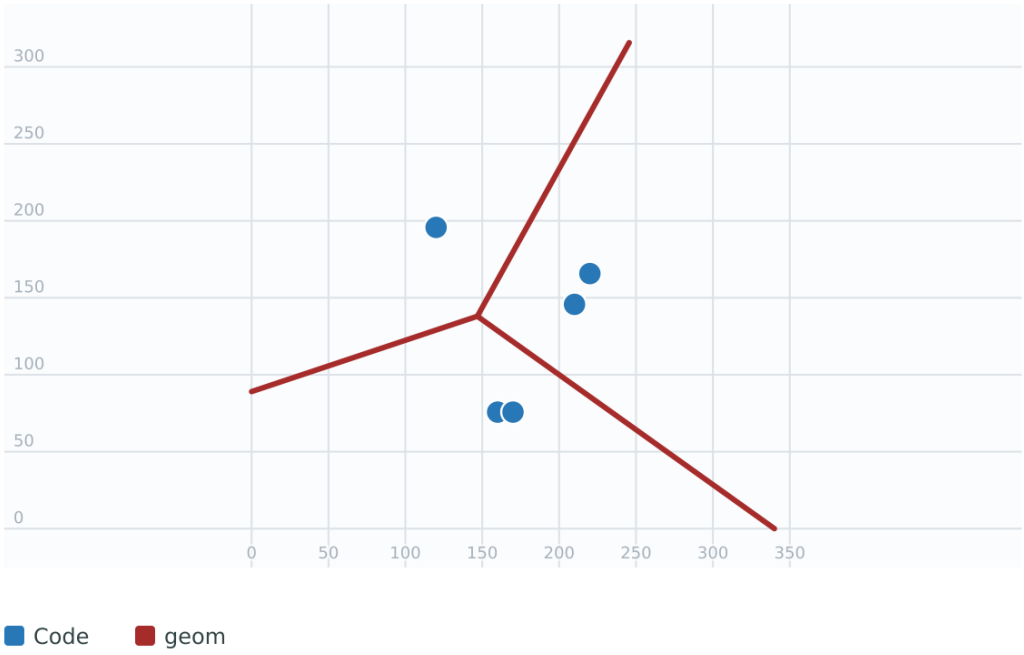
Code

```
SELECT ST_VoronoiLines('MULTIPOINT (50 30,60 30,100 100,10 150,110 120)'::geometry,
            30) AS geom;
```

Output

```
MULTILINESTRING((135.555555555556 270,36.8181818181818 92.2727272727273),(36.8181818181818 92.2727272727273,-110 ↩
    43.3333333333333),(230 -45.7142857142858,36.8181818181818 92.2727272727273))
```

Figure



相关信息

ST_DelaunayTriangles, ST_VoronoiPolygons

7.14.30 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

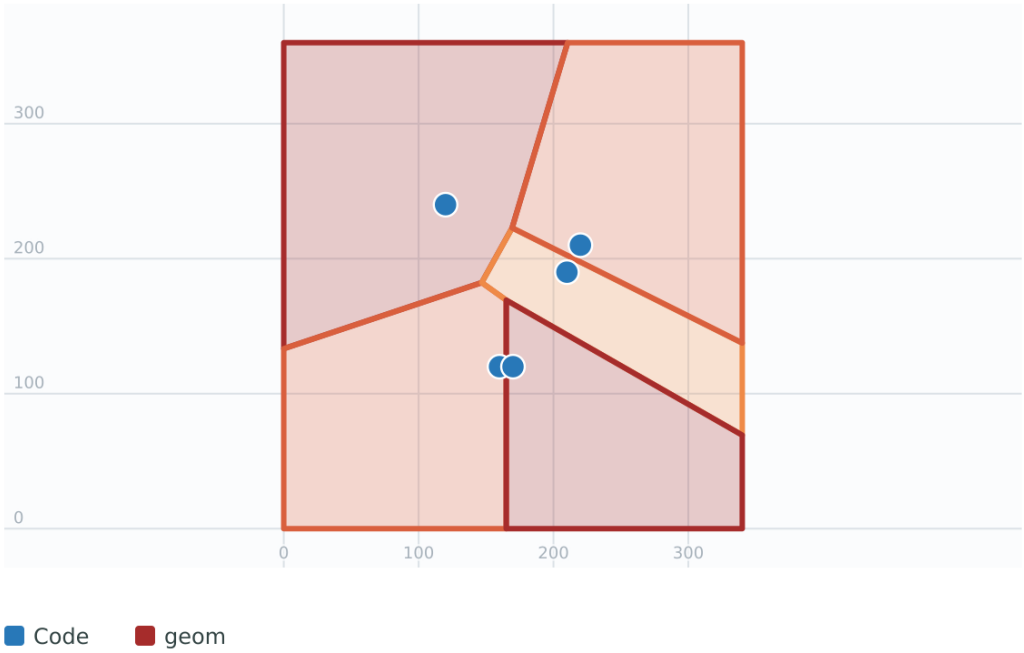
示例

Points overlaid on top of the Voronoi diagram.

Code

```
SELECT ST_VoronoiPolygons('MULTIPOINT (50 30,60 30,100 100,10 150,110 120)'::geometry
            ) AS geom;
```

Figure

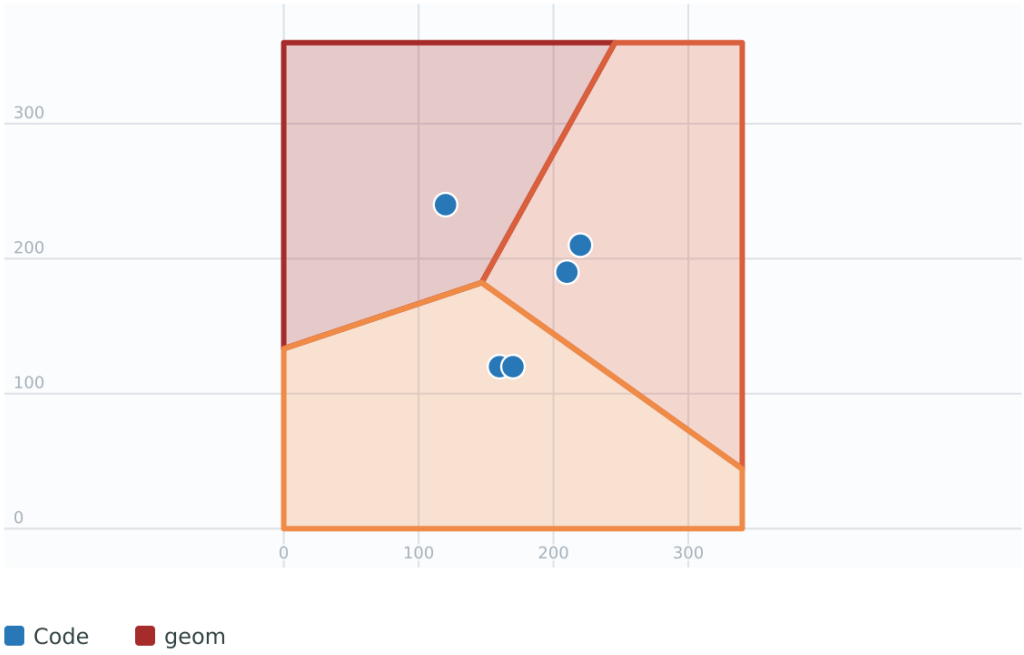


Voronoi diagram with a tolerance of 30 units.

Code

```
SELECT ST_VoronoiPolygons('MULTIPOINT (50 30,60 30,100 100,10 150,110 120)'::geometry,
            30) AS geom;
```

Figure



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

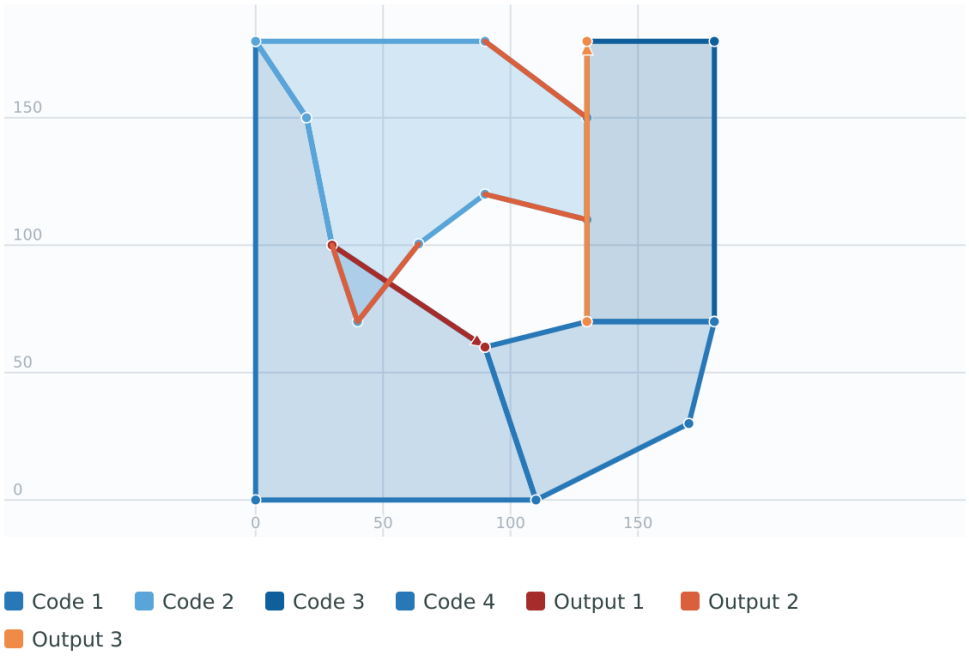
示例

Invalid edges caused by overlap and non-matching vertices.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((10 190,30 160,40 110,100 70,120 10,10 10,10 190))'::geometry),
  (2, 'POLYGON ((100 190,10 190,30 160,40 110,50 80,74 110.5,100 130,140 120,140 160,100 ↩
    190))'::geometry),
  (3, 'POLYGON ((140 190,190 190,190 80,140 80,140 190))'::geometry),
  (4, 'POLYGON ((180 40,120 10,100 70,140 80,190 80,180 40))'::geometry)
)
SELECT id, ST_CoverageInvalidEdges(geom) OVER ()
  FROM coverage;
```

Figure



This example tests the entire table for coverage validity.

Code

```
SELECT true = ALL (
    SELECT ST_CoverageInvalidEdges(geom) OVER () IS NULL
    FROM coverage
    );
```

相关信息

ST_IsValid, ST_CoverageUnion, ST_CoverageClean, ST_CoverageSimplify

7.15.2 ST_CoverageClean

ST_CoverageClean — 将非规范化输入算出一个规范的多边形覆盖结果（边界匹配、无重叠、无缝隙）。

Synopsis

geometry **ST_CoverageClean**(geometry winset geom, float8 gapMaximumWidth = 0, float8 snappingDistance = -1, text overlapMergeStrategy = 'MERGE_LONGEST_BORDER');

描述

一个窗口函数，用于整理一组有效的面状几何的边界，使其形成干净的覆盖（clean coverage）。清理过程包括：

- 将边点和边缘行为靠（snapping），消除微小偏差，确保共有边缘点一致、拓扑关系
- 将重叠区域合并到上级父 polygon 中
- 把狭窄的缝隙并入相邻的 polygon 中

gapMaximumWidth 控制哪些多边形之间的空隙会被合并；宽度 <= 该值的空隙会被并入相邻的 polygon。

snappingDistance 控制边点和边的边靠（snapping）。默认值 -1 会根据输入范围自动计算靠距离；设置为 0.0 表示关闭靠。

overlapMergeStrategy 用来指定重叠区域如何合并到父 polygon 中：

- MERGE_LONGEST_BORDER - 合并到与其共有最长边界的 polygon
- MERGE_MAX_AREA - 合并到面积最大的 polygon
- MERGE_MIN_AREA - 合并到面积最小的 polygon
- MERGE_MIN_INDEX - 合并到输入索引最小的 polygon（按输入顺序定义）

结果是一个干净的面覆盖（polygonal coverage），能通过 ST_CoverageInvalidEdges 的校验，并可作为后续覆盖处理函数的输入。



Note
如果确定合适的最大缝隙宽度，可以先将 gapMaximumWidth => 0 来进行清理，然后使用 ST_CoverageUnion 对清理后的覆盖进行合并。合并结果中的“洞”区域的就是原始数据中的缝隙。接着，可以将这些洞提取成 polygon，并对每个 polygon 进行 ST_MaximumInscribedCircle 来测量它们的最大内切圆；缝隙宽度 = 对算得到的半径 × 2。

可用性: 3.6.0 - 需要 GEOS >= 3.14.0

示例

Clean a coverage containing overlaps and narrow gaps, and compare the cleaned polygons with the original invalid edges.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((10 190,30 160,27 134.5,40 110,122 47,120 10,10 10,10 190))'::geometry),
  (2, 'POLYGON ((150 190,10 190,30 160,50 140,40 110,50 80,130 70,135 111,140 130,140 ↩
    160,150 190))'::geometry),
  (3, 'POLYGON ((140 190,190 190,190 80,140 80,140 190))'::geometry),
  (4, 'POLYGON ((190 10,120 10,97 77,160 90,170 70,190 80,190 10))'::geometry)
), processed AS (
  SELECT id,
         ST_CoverageClean(geom, 1) OVER () AS cleaned,
         ST_CoverageInvalidEdges(geom) OVER () AS invalid_edges
  FROM coverage
)
SELECT ST_Collect(cleaned ORDER BY id) AS cleaned,
       ST_Collect(invalid_edges ORDER BY id) AS invalid_edges
FROM processed;
```

Figure



相关信息

ST_CoverageInvalidEdges, ST_CoverageUnion, ST_CoverageSimplify, ST_MaximumInscribedCircle

7.15.3 ST_CoverageSimplify

ST_CoverageSimplify — 简化多边形覆盖范围的窗口函数。

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。

Use this function to simplify adjacent polygon layers when shared boundaries must remain edge-matched. For a single geometry where only self-intersection and ring validity matter, see ST_SimplifyPreserv



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

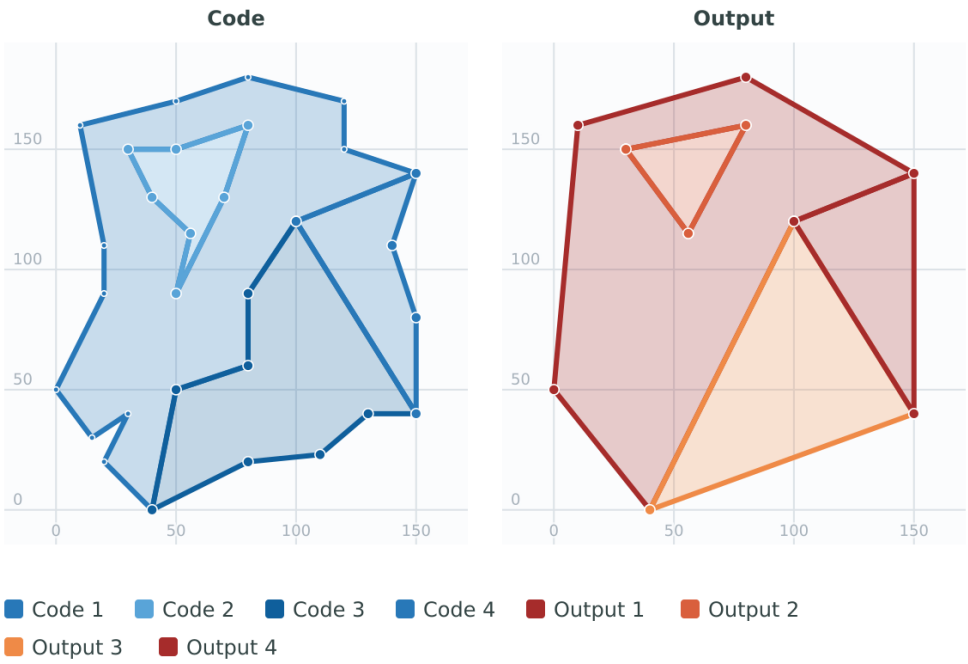
示例

Simplify a polygonal coverage while preserving shared boundaries.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((160 150,110 130,90 100,90 70,60 60,50 10,30 30,40 50,25 40,10 60,30 100,30 ↩
     120,20 170,60 180,90 190,130 180,130 160,160 150),(40 160,50 140,66 125,60 100,80  ↩
    140,90 170,60 160,40 160))'::geometry),
  (2, 'POLYGON ((40 160,60 160,90 170,80 140,60 100,66 125,50 140,40 160))'::geometry),
  (3, 'POLYGON ((110 130,160 50,140 50,120 33,90 30,50 10,60 60,90 70,90 100,110 130))':: ↩
    geometry),
  (4, 'POLYGON ((160 150,150 120,160 90,160 50,110 130,160 150))'::geometry)
)
SELECT id, ST_CoverageSimplify(geom, 30) OVER ()
  FROM coverage;
```

Figure



相关信息

ST_CoverageInvalidEdges, ST_CoverageUnion, ST_CoverageClean

7.15.4 ST_CoverageEdges

ST_CoverageEdges — Computes the unique edges of a polygonal coverage.

Synopsis

geometry **ST_CoverageEdges**(geometry geom, integer edgetype = 0);
geometry **ST_CoverageEdges**(geometry[] geoms, integer edgetype = 0);

描述

Returns a MultiLineString representing the unique edges of a polygonal coverage. A polygonal coverage is a set of non-overlapping polygons where adjacent polygons have matching vertices along shared edges.

The *edgetype* parameter can be used to select which edges are returned:

- **0 (ALL)** - all unique edges (default)
- **1 (EXTERIOR)** - only exterior edges (non-shared)
- **2 (INTERIOR)** - only interior edges (shared)

Availability: 3.7.0

Requires GEOS >= 3.15.0

示例

This example returns all unique edges of a coverage.

Code

```
WITH coverage(geom) AS (
  VALUES (
    'GEOMETRYCOLLECTION(
      POLYGON((0 0,10 0,10 10,0 10,0 0)),
      POLYGON((10 0,20 0,20 10,10 10,10 0))
    )'::geometry
  )
)
SELECT geom AS input_coverage,
       ST_CoverageEdges(geom) AS all_edges
FROM coverage;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------------------------------------------------
input_coverage | GEOMETRYCOLLECTION(POLYGON((0 0,10 0,10 10,0 10,0 0)),POLYGON((10 0,20 0,20 10,10 10,10 0)))
all_edges      | MULTILINESTRING((10 0,10 10),(10 10,0 10,0 0,10 0),(10 0,20 0,20 10,10 10))
```

This example returns only the interior shared edges of a coverage.

Code

```
WITH coverage(geom) AS (
  VALUES (
    'GEOMETRYCOLLECTION(
      POLYGON((0 0,10 0,10 10,0 10,0 0)),
      POLYGON((10 0,20 0,20 10,10 10,10 0))
    )'::geometry
  )
)
SELECT geom AS input_coverage,
       ST_CoverageEdges(geom, 2) AS shared_edges
FROM coverage;
```

Output

```
-[ RECORD 1 ]--+-----------------------------------------------------------------------------------------------
input_coverage | GEOMETRYCOLLECTION(POLYGON((0 0,10 0,10 10,0 10,0 0)),POLYGON((10 0,20 0,20 10,10 10,10 0)))
shared_edges   | MULTILINESTRING((10 0,10 10))
```

相关信息

ST_CoverageInvalidEdges, ST_CoverageSimplify, ST_CoverageUnion

7.15.5 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

Union a polygonal coverage.

Code

```
WITH coverage(id, geom) AS (VALUES
  (1, 'POLYGON ((10 10,10 150,80 190,110 150,90 110,40 110,50 60,10 10))'::geometry),
  (2, 'POLYGON ((120 10,10 10,50 60,100 70,120 10))'::geometry),
  (3, 'POLYGON ((140 80,120 10,100 70,40 110,90 110,110 150,140 80))'::geometry),
  (4, 'POLYGON ((140 190,120 170,140 130,160 150,140 190))'::geometry),
  (5, 'POLYGON ((180 160,170 140,140 130,160 150,140 190,180 160))'::geometry)
)
SELECT ST_CoverageUnion(geom) AS wkt
  FROM coverage;
```

Figure



相关信息

ST_CoverageInvalidEdges, ST_CoverageSimplify, ST_CoverageClean, 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

Code

```
ST_Affine(geom, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff)
```

表示□□矩□

Code

```
/ a b c xoff \
| d e f yoff |
| g h i zoff |
\ 0 0 0     1 /
```

并且□点□□如下：

Code

```
x' = a*x + b*y + c*z + xoff
y' = d*x + e*y + f*z + yoff
z' = g*x + h*y + i*z + zoff
```

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

Code

```
ST_Affine(geom, a, b, d, e, xoff, yoff)
```

表示□□矩□

Code

```
/ 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：

Code

```
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();

Code

```
SELECT ST_Affine(geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), 0, 0, 0, 1, 0, 0, 0) ↩
    As using_affine,
   ST_Rotate(geom, pi()) As using_rotate
  FROM (SELECT 'LINESTRING(1 2 3,1 4 3)'::geometry As geom) As foo;
```

Figure

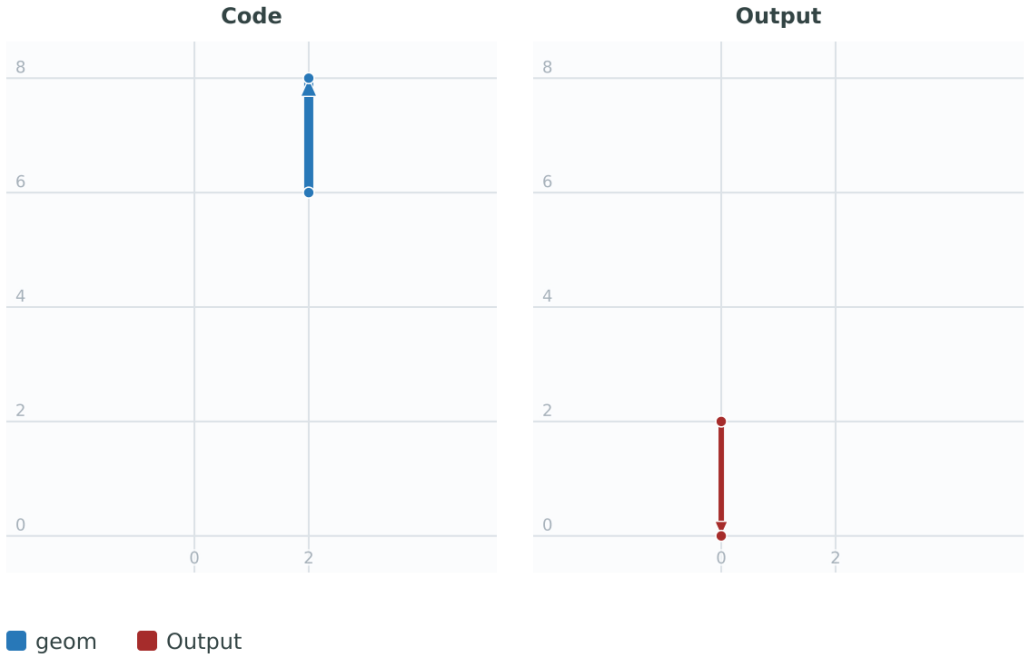


Rotate a 3D line 180 degrees in both the x and z axis.

Code

```
SELECT ST_Affine(geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), -sin(pi()), 0, sin( ↩
    pi()), cos(pi()), 0, 0, 0)
    FROM (SELECT 'LINESTRING(1 2 3,1 4 3)'::geometry As geom) As foo;
```

Figure



相关信息

ST_Rotate, ST_Scale, ST_Translate, ST_TransScale

7.16.2 ST_Rotate

ST_Rotate — 绕原点旋转几何体。

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

绕原点逆时针旋转几何体 rotRadians。旋转原点可以指定为 POINT 几何对象，也可以指定为 x 和 y 坐标。如果未指定原点，则几何对象将绕 POINT(0 0) 旋转。

增强功能：引入了 2.0.0 对多面体曲面、三角形和三角网的支持。

增强：2.0.0 添加了用于指定旋转原点的附加参数。

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)。

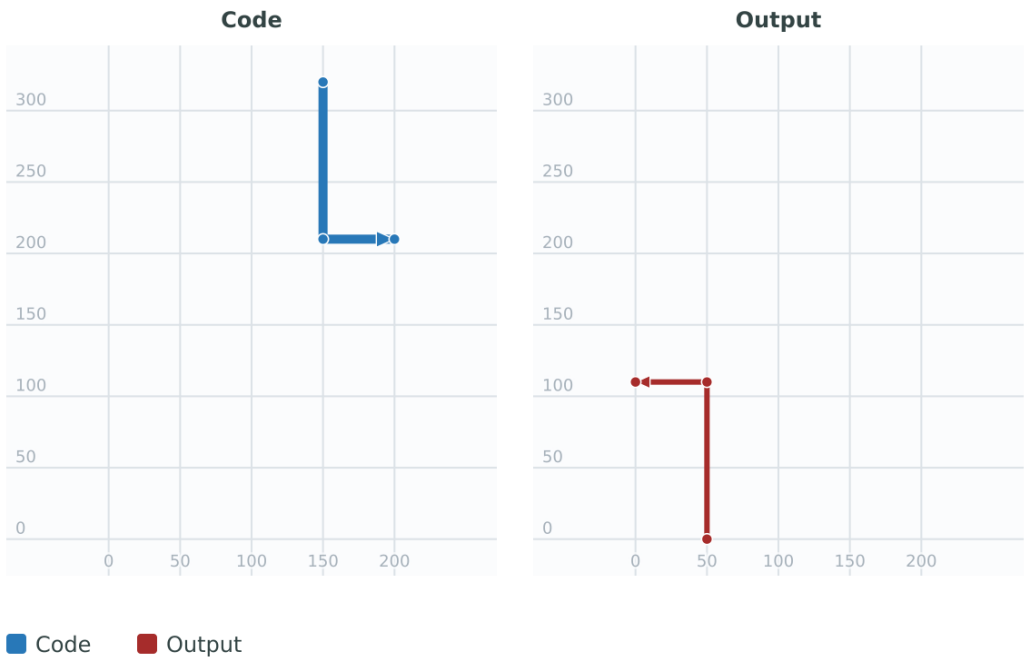
示例

This example rotates a geometry by 180 degrees.

Code

```
SELECT ST_Rotate('LINESTRING (50 160,50 50,100 50)'::geometry, pi());
```

Figure

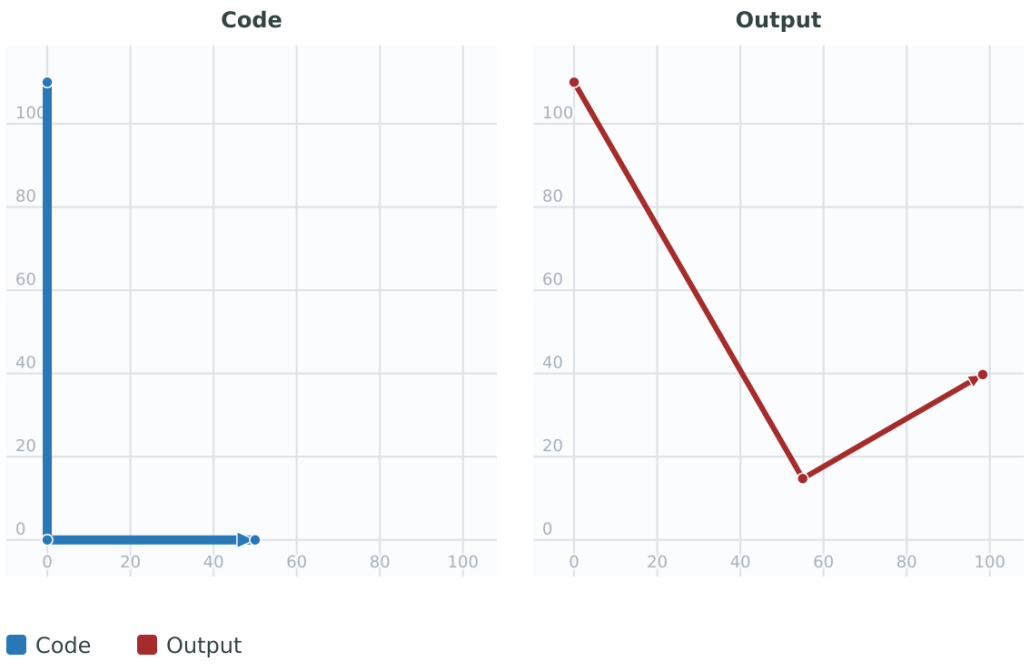


This example rotates a geometry 30 degrees counter-clockwise around x=50, y=160.

Code

```
SELECT ST_Rotate('LINESTRING (50 160,50 50,100 50)'::geometry, pi()/6, 50, 160);
```

Figure

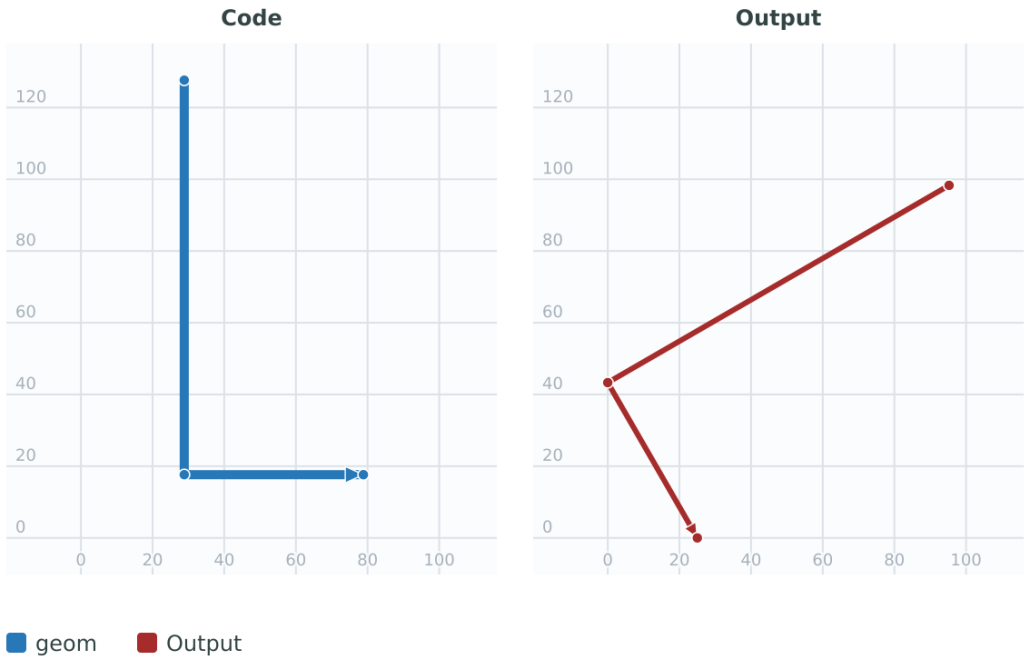


This example rotates a geometry 60 degrees clockwise around its centroid.

Code

```
SELECT ST_Rotate(geom, -pi()/3, ST_Centroid(geom))
FROM (SELECT 'LINESTRING (50 160,50 50,100 50)'::geometry AS geom) AS foo;
```

Figure



相关信息

ST_Affine, ST_RotateX, ST_RotateY, ST_RotateZ

7.16.3 ST_RotateX

ST_RotateX — 绕 X 轴旋转几何体。

Synopsis

geometry **ST_RotateX**(geometry geomA, float rotRadians);

描述

绕 X 轴旋转几何体 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) 的缩写。

增强功能：引入了 2.0.0 版多面体曲面、三角形和三角网的支持。

可用性：1.1.2。1.2.2 中名称从 RotateX 更改为 ST_RotateX

✔ 该函数支持多面体曲面。

✔ 该函数支持 3d 并且不会丢失 z-index。

✔ 此函数支持三角形和不规则三角网面 (TIN)。

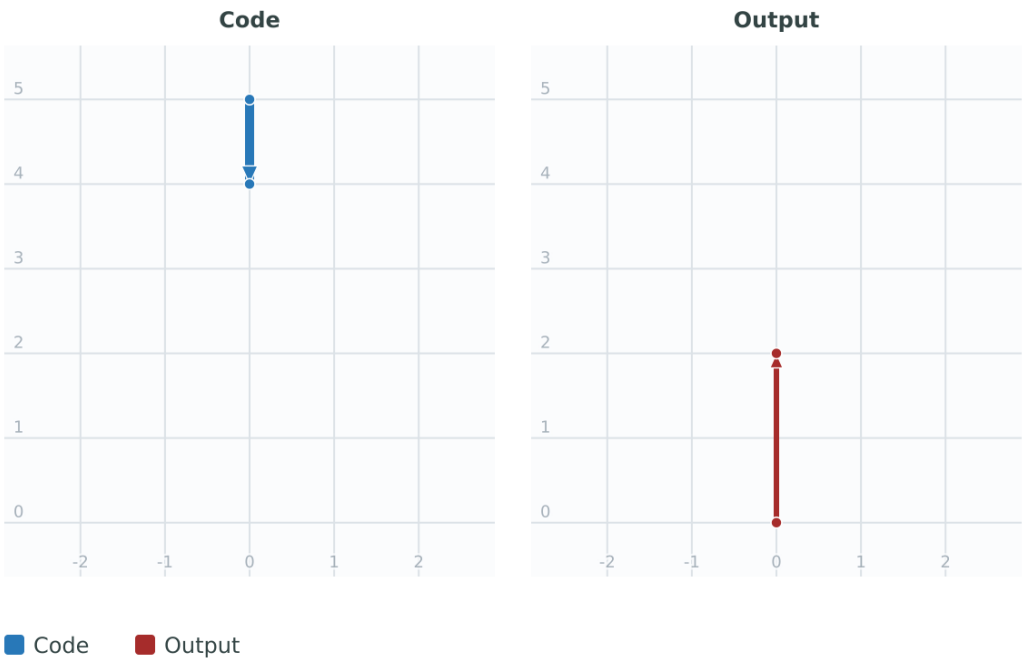
示例

This example rotates a line 90 degrees around the X axis.

Code

```
SELECT ST_RotateX('LINESTRING(1 2 3,1 1 1)', pi()/2);
```

Figure



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)。

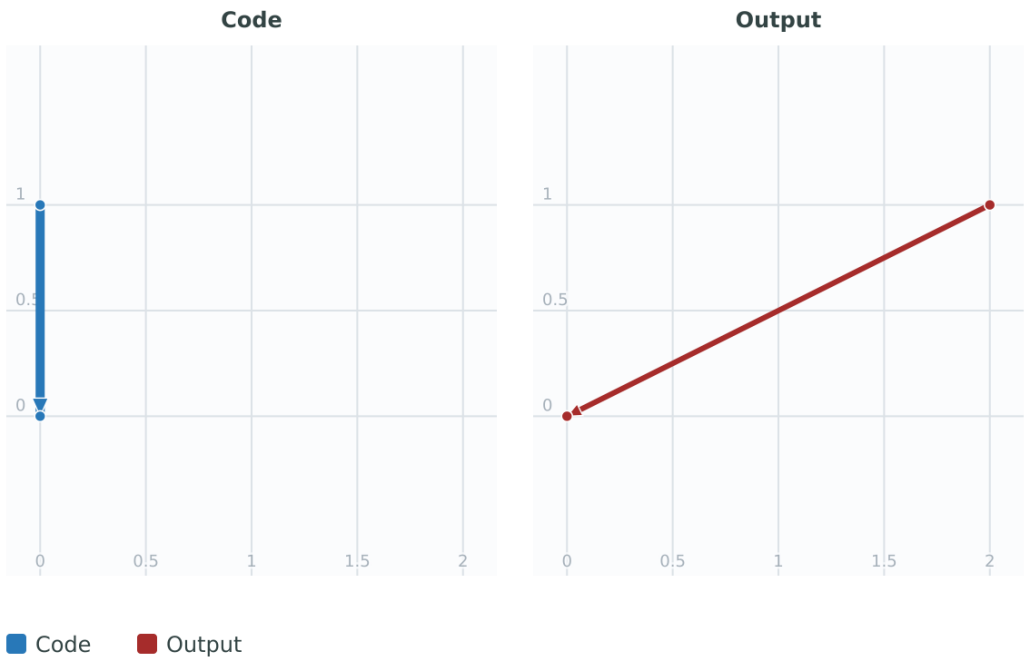
示例

This example rotates a line 90 degrees around the Y axis.

Code

```
SELECT ST_RotateY('LINESTRING(1 2 3,1 1 1)', pi()/2);
```

Figure



相关信息

ST_Affine, ST_RotateX, ST_RotateZ

7.16.5 ST_RotateZ

ST_RotateZ — 绕 Z 轴旋转几何体。

Synopsis

geometry **ST_RotateZ**(geometry geomA, float rotRadians);

描述

绕 Z 轴旋转几何体 geomA - rotRadians。



Note
是 ST_Rotate 的同义词

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)。

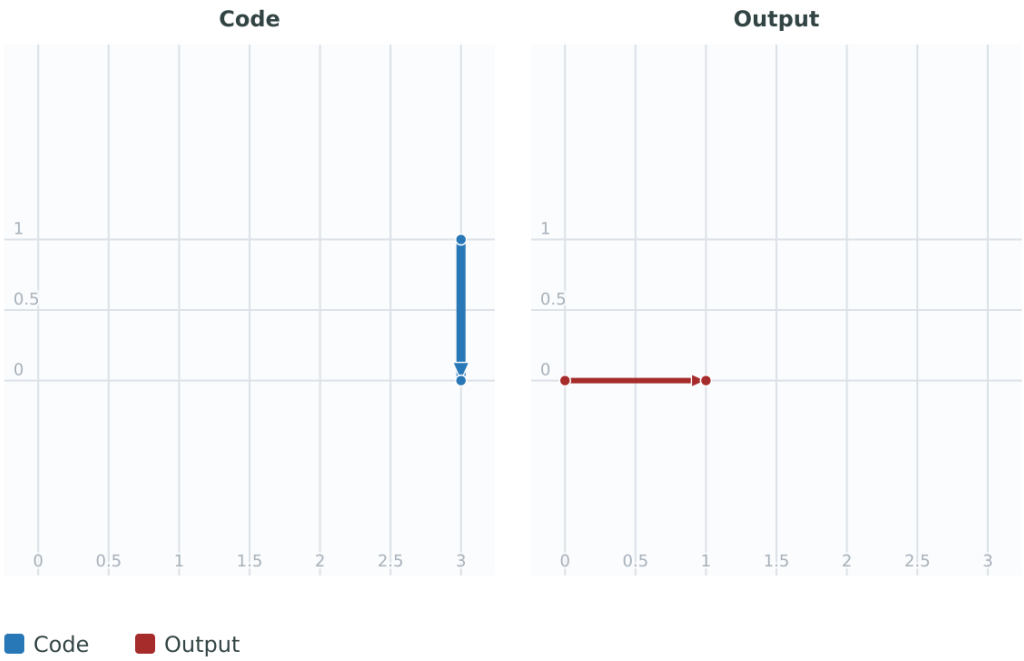
示例

This example rotates a line 90 degrees around the Z axis.

Code

```
SELECT ST_RotateZ('LINESTRING(1 2 3,1 1 1)', pi()/2);
```

Figure

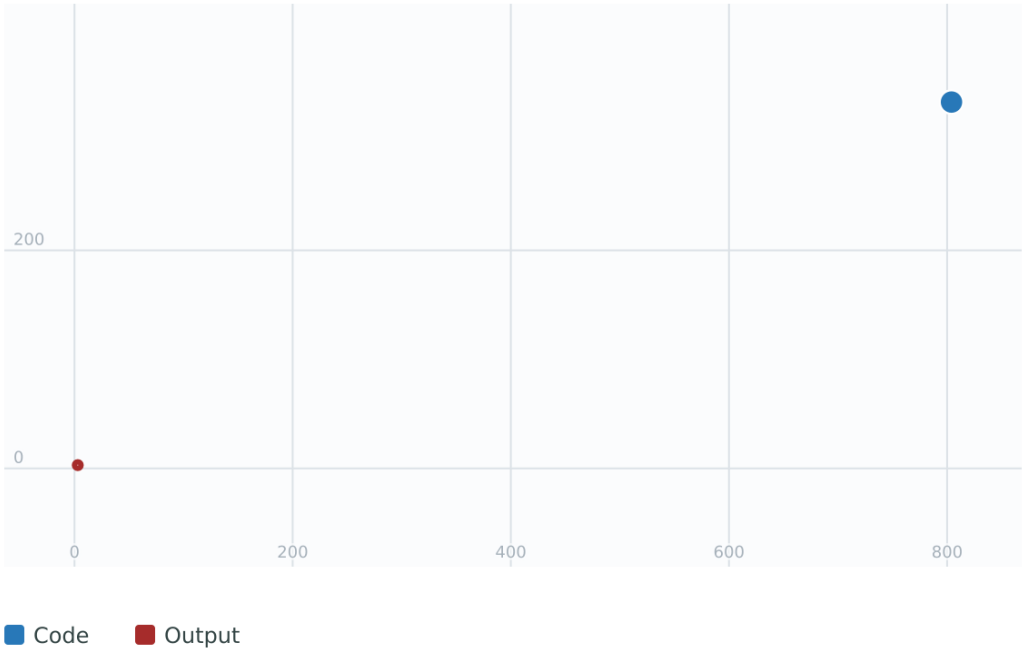


This example rotates a curved circle around the Z axis.

Code

```
SELECT ST_RotateZ(geom, pi()/2)
FROM (SELECT ST_LineToCurve(ST_Buffer('POINT(234 567)'::geometry, 3)) As geom) As foo;
```

Figure



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⍰。

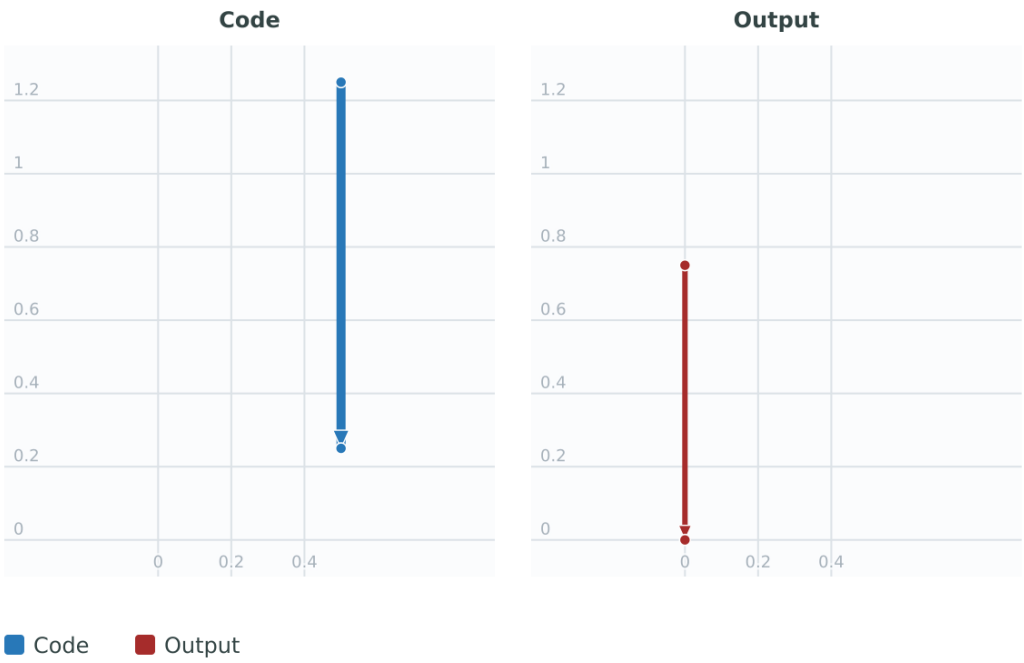
示例

This example scales X, Y, and Z values.

Code

```
SELECT ST_Scale('LINESTRING(1 2 3,1 1 1)', 0.5, 0.75, 0.8);
```

Figure

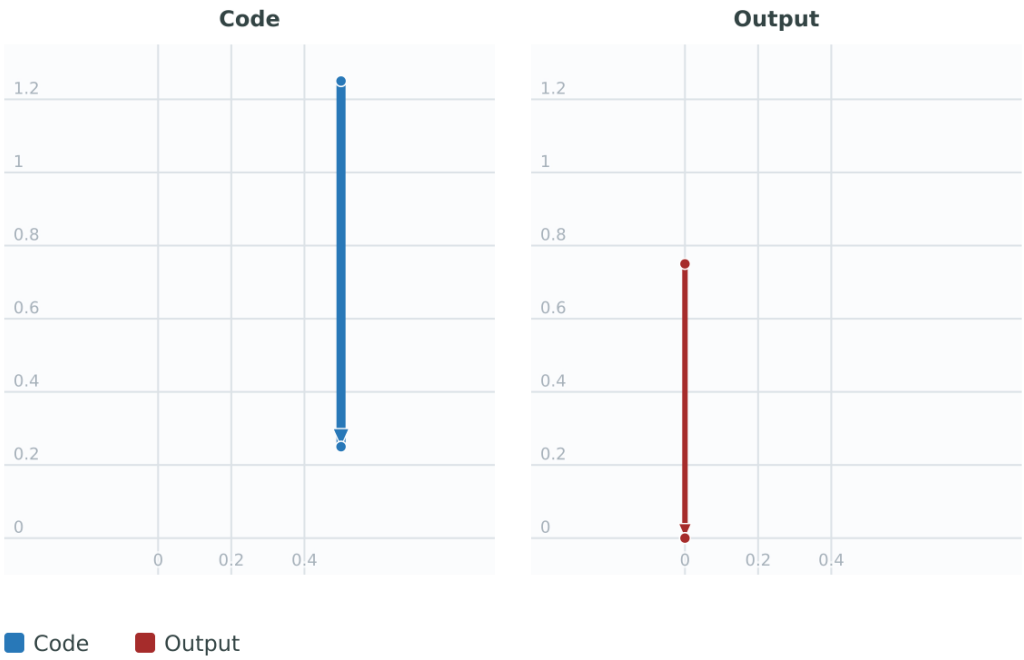


This example scales X and Y values.

Code

```
SELECT ST_Scale('LINESTRING(1 2 3,1 1 1)', 0.5, 0.75);
```

Figure

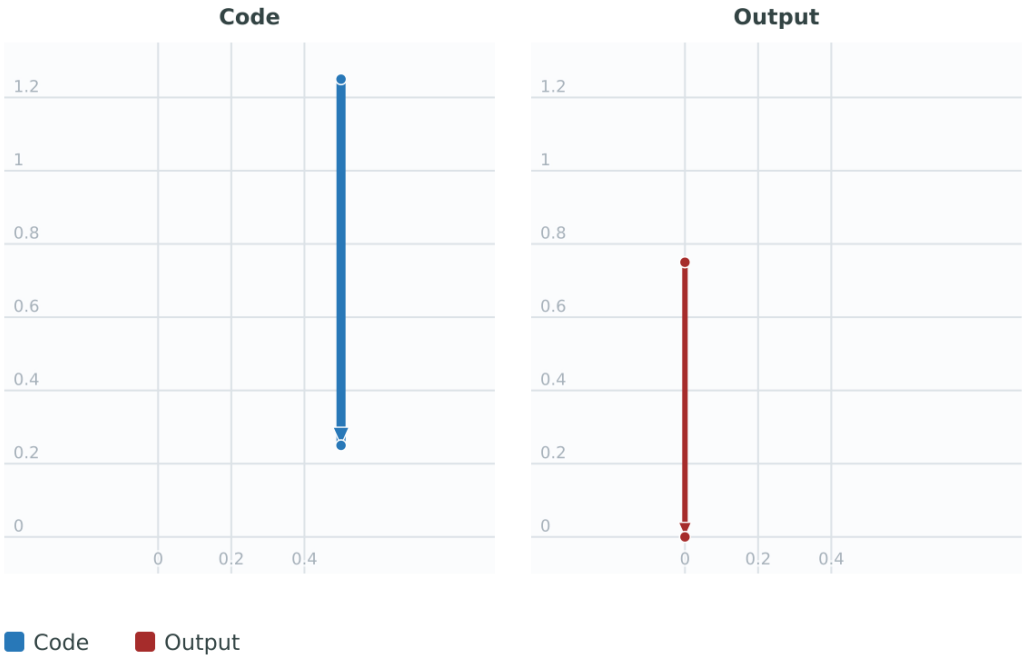


This example scales X, Y, Z, and M values.

Code

```
SELECT ST_Scale('LINESTRING(1 2 3 4,1 1 1 1)',
    ST_MakePoint(0.5, 0.75, 2, -1));
```

Figure

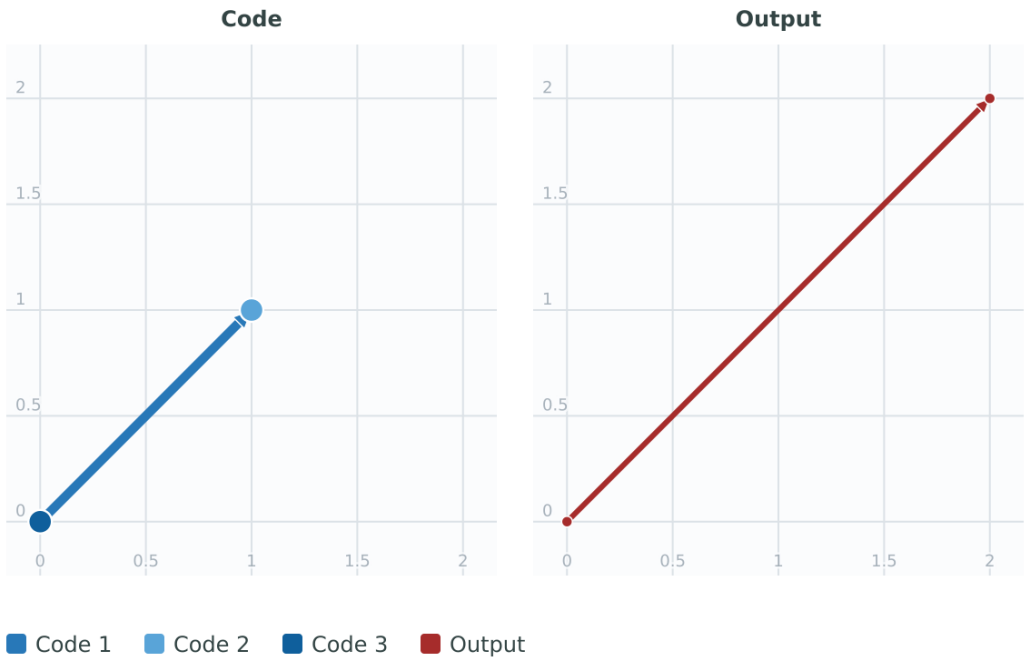


This example scales X and Y values using a false origin.

Code

```
SELECT ST_Scale('LINESTRING(1 1,2 2)', 'POINT(2 2)', 'POINT(1 1)'::geometry);
```

Figure



This example constructs an ellipse centered at (10, 20), with X radius 8, Y radius 4, and rotated 30 degrees counter-clockwise. Scaling and rotation use the center as their origin, so the ellipse keeps both its position and SRID.

Code

```
WITH params AS (
  SELECT
    ST_SetSRID(ST_Point(10, 20), 3857) AS center,
    8.0 AS x_radius,
    4.0 AS y_radius,
    radians(30) AS angle
)
SELECT ST_Rotate(
         ST_Scale(
           ST_Buffer(center, 1, 'quad_segs=32'),
           ST_MakePoint(x_radius, y_radius),
           center
         ),
         angle,
         center
       ) AS ellipse
FROM params;
```

相关信息

ST_Affine, ST_Buffer, ST_MakePoint, ST_Rotate, 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

Returns a new geometry whose coordinates are translated delta x,delta y,delta z units. Units are based on the units defined in spatial reference (SRID) for this geometry. M coordinates are preserved.



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⌧。


⌧功能支持 M 坐⌧。

示例

将点移⌧ 1 度⌧度

Code

```
SELECT ST_Translate(ST_GeomFromText('POINT(-71.01 42.37)', 4326), 1, 0) AS wgs_transgeomtxt ↩
    ;
```

Output

```
POINT(-70.01 42.37)
```

将⌧串移⌧ 1 度⌧度和 1/2 度⌧度

Code

```
SELECT ST_Translate(ST_GeomFromText('LINESTRING(-71.01 42.37,-71.11 42.38)', 4326), 1, 0.5) ↩
     AS wgs_transgeomtxt;
```

Figure



移⊠ 3d 点

Code

```
SELECT ST_Translate(CAST('POINT(0 0 0)' AS geometry), 5, 12, 3);
```

Output

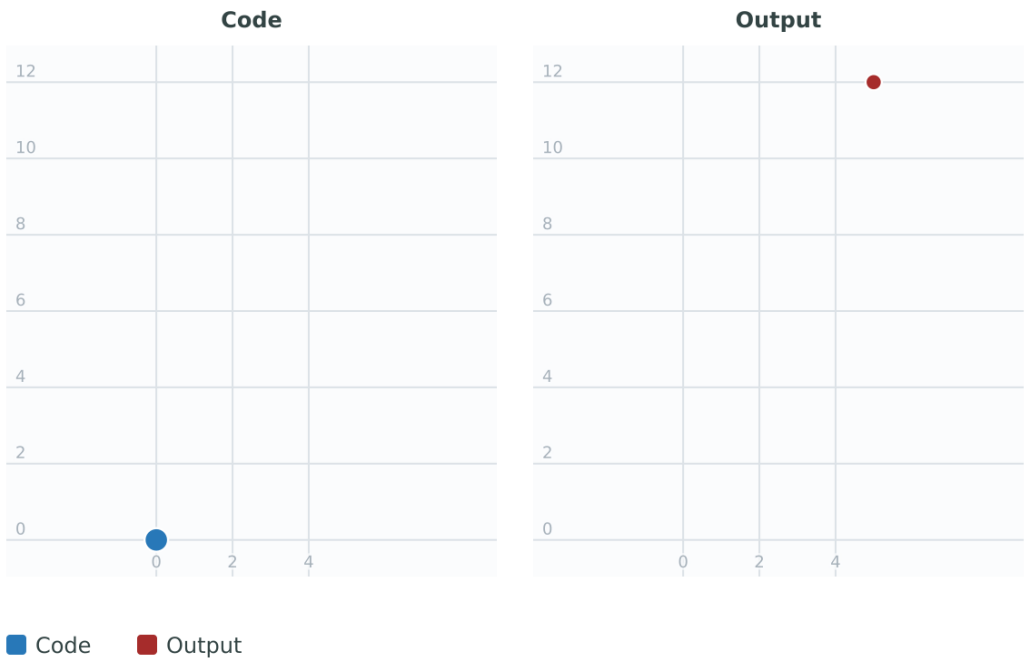
```
POINT(5 12 3)
```

Move points with a measure coordinate

Code

```
SELECT ST_Translate('POINTM(0 0 11)', 5, 12);
```

Figure



Code

```
SELECT ST_Translate('POINT(0 0 7 11)', 5, 12, 3);
```

Output

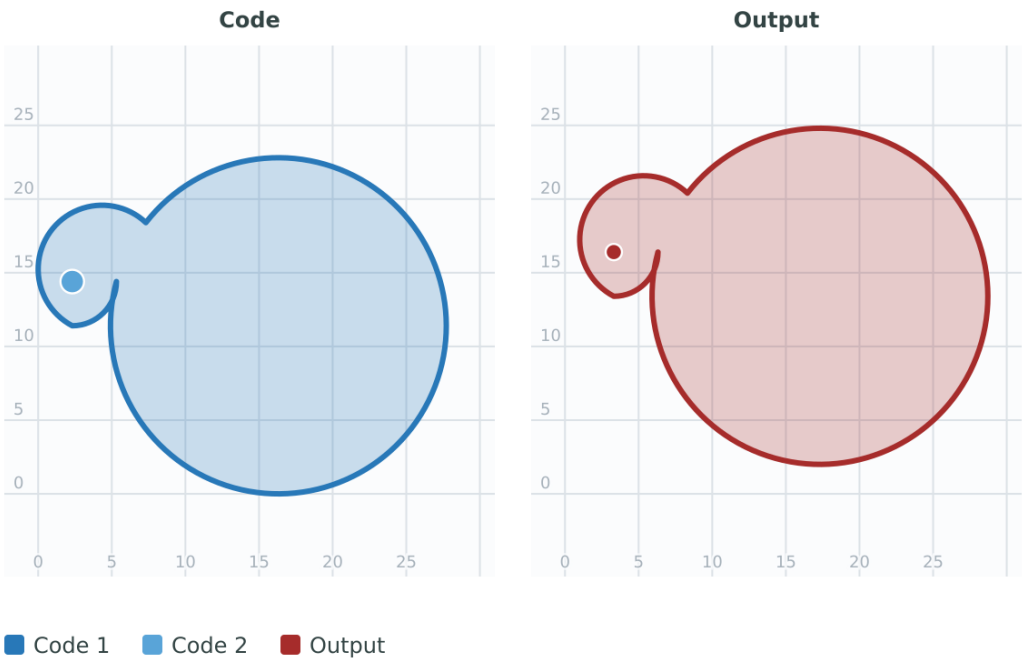
```
POINT(5 12 10 11)
```

移⊠曲⊠和点

Code

```
SELECT ST_Translate(ST_Collect('CURVEPOLYGON(CIRCULARSTRING(4 3,3.12 0.878,1 0,-1.121 ↩
    5.1213,6 7,8 9,4 3))', 'POINT(1 3)'), 1, 2);
```

Figure



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。

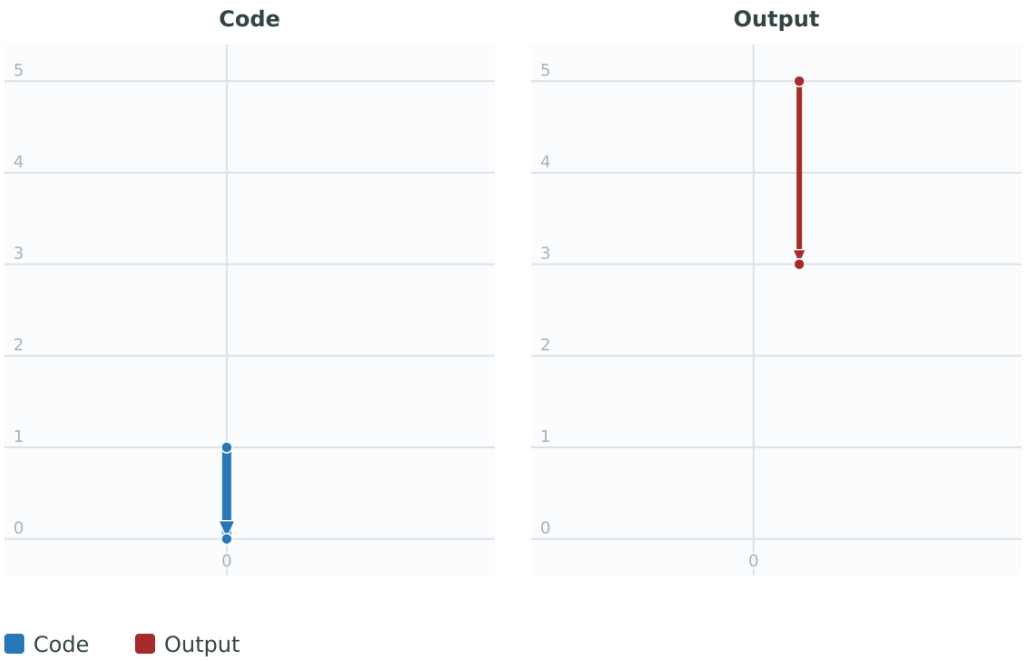
此方法支持☒形字符串和曲☒。

示例

Code

```
SELECT ST_TransScale('LINESTRING(1 2 3,1 1 1)', 0.5, 1, 1, 2);
```

Figure

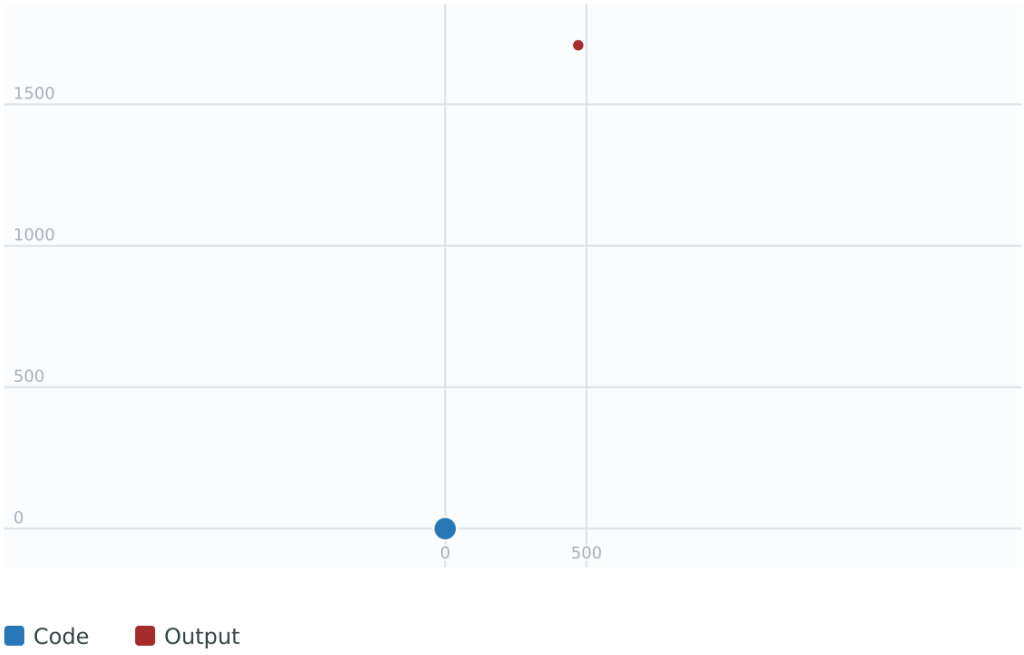


This example buffers a point to approximate a circle, converts it to a curve, translates it by 1,2, and scales it by 3,4.

Code

```
SELECT ST_TransScale(ST_LineToCurve(ST_Buffer('POINT(234 567)', 3)), 1, 2, 3, 4);
```

Figure



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

Cluster polygons within 50 units of each other, requiring at least two polygons per cluster. The isolated polygon is reported as noise.

Code

```
WITH input(name, geom) AS (
  VALUES
    ('A1', ST_MakeEnvelope(0, 0, 10, 10)),
    ('A2', ST_MakeEnvelope(30, 0, 40, 10)),
    ('B1', ST_MakeEnvelope(100, 0, 110, 10)),
    ('B2', ST_MakeEnvelope(130, 0, 140, 10)),
    ('noise', ST_MakeEnvelope(250, 0, 260, 10))
), clustered AS (
  SELECT name,
         geom,
         ST_ClusterDBSCAN(geom, eps => 50, minpoints => 2)
           OVER (ORDER BY name) AS cid
  FROM input
)
SELECT COALESCE(cid::text, 'noise') AS cluster,
       string_agg(name, ',' ORDER BY name) AS members,
       ST_Collect(geom ORDER BY name) AS cluster_geom
FROM clustered
GROUP BY cid
ORDER BY cid NULLS LAST;
```

Output

```
0 | A1,A2 | MULTIPOLYGON(((0 0,0 10,10 10,10 0,0 0)),((30 0,30 10,40 10,40 0,30 0)))
1 | B1,B2 | MULTIPOLYGON(((100 0,100 10,110 10,110 0,100 0)),((130 0,130 10,140 10,140 0,130 0)))
noise | noise | MULTIPOLYGON(((250 0,250 10,260 10,260 0,250 0)))
```

Figure



⊠示将具有相同簇⊠号的地⊠⊠合成几何集合的示例。

Code

```
SELECT cid, ST_Collect(geom) AS cluster_geom, array_agg(parcel_id) AS ids_in_cluster FROM (
    SELECT parcel_id, ST_ClusterDBSCAN(geom, eps => 0.5, minpoints => 5) over () AS cid,  ↩
    geom
    FROM parcels) sq
GROUP BY cid;
```

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

Code

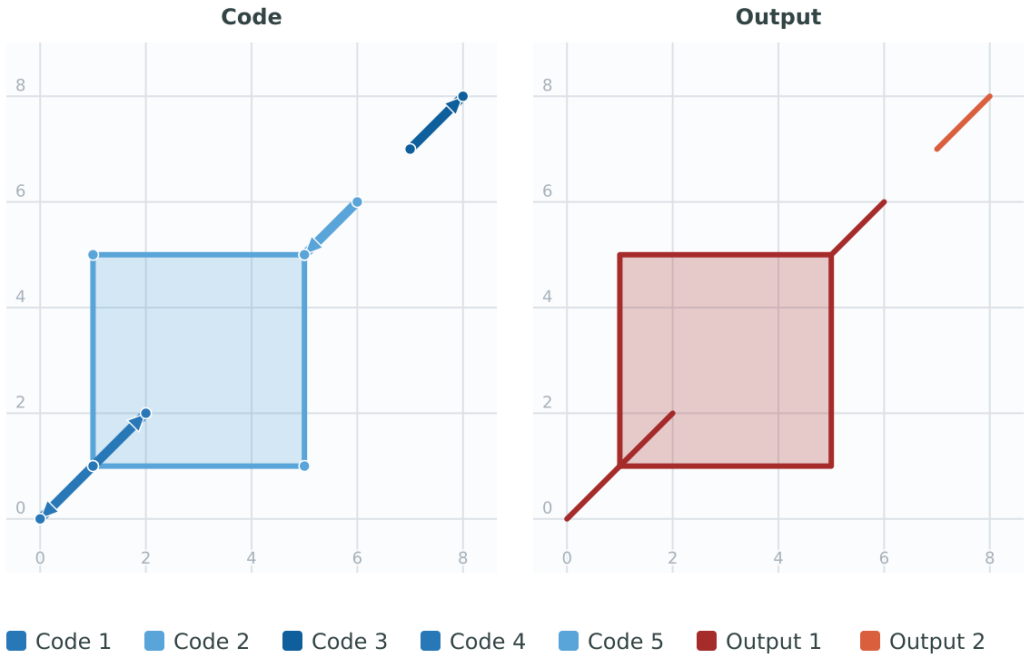
```
WITH testdata AS
  (SELECT unnest(
            ARRAY['LINESTRING (0 0,1 1)'::geometry,
            'LINESTRING (5 5,4 4)'::geometry,
            'LINESTRING (6 6,7 7)'::geometry,
            'LINESTRING (0 0,-1 -1)'::geometry,
            'POLYGON ((0 0,4 0,4 4,0 4,0 0))'::geometry]) AS geom)

SELECT unnest(ST_ClusterIntersecting(geom)) FROM testdata;
```

Output

```
GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),LINESTRING(5 5,4 4),LINESTRING(0 0,-1 -1),POLYGON((0 0,4 0,4 4,0 4,0 0)))
GEOMETRYCOLLECTION(LINESTRING(6 6,7 7))
```

Figure



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

Return an explicit cluster id for each input geometry. Using the conventional `cid` alias keeps the geometry column ready for color-by-cluster rendering.

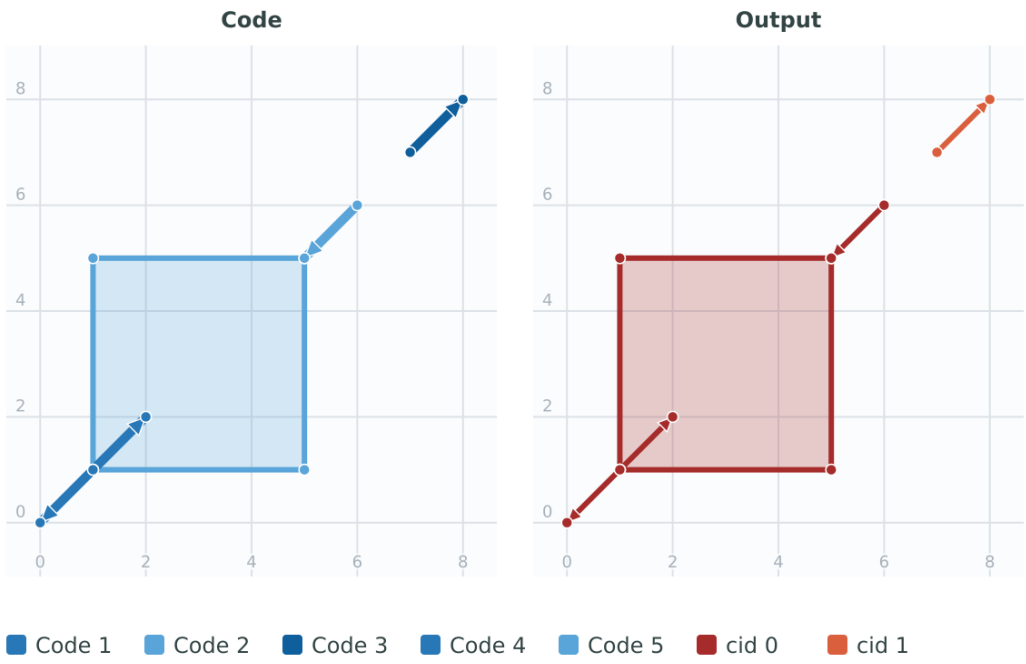
Code

```
WITH testdata(id, geom) AS (
  VALUES (1, 'LINESTRING (0 0,1 1)'::geometry),
         (2, 'LINESTRING (5 5,4 4)'::geometry),
         (3, 'LINESTRING (6 6,7 7)'::geometry),
         (4, 'LINESTRING (0 0,-1 -1)'::geometry),
         (5, 'POLYGON ((0 0,4 0,4 4,0 4,0 0))'::geometry)
), clustered AS (
  SELECT id,
         geom,
         ST_ClusterIntersectingWin(geom) OVER () AS cid
  FROM testdata
)
SELECT id,
       geom AS geom_wkt,
       cid
FROM clustered
ORDER BY id;
```

Output

```
id |            geom_wkt            | cid
----+--------------------------------+-----
  1 | LINESTRING(0 0,1 1)            |   0
  2 | LINESTRING(5 5,4 4)            |   0
  3 | LINESTRING(6 6,7 7)            |   1
  4 | LINESTRING(0 0,-1 -1)          |   0
  5 | POLYGON((0 0,4 0,4 4,0 4,0 0)) |   0
```

Figure



相关信息

ST_ClusterIntersecting, ST_ClusterWithin, ST_ClusterWithinWin

7.17.4 ST_ClusterRelateWin

ST_ClusterRelateWin — Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.

Synopsis

integer **ST_ClusterRelateWin**(geometry winset geom, text relate_matrix);

描述

A window function that builds connected clusters of geometries that intersect. Geometries are added to a cluster if they share a pairwise DE9IM relationship with another member of the cluster. With this function it is possible to build a cluster of all objects that touch at boundaries, but exclude those that merely overlap.

Availability: 3.7.0

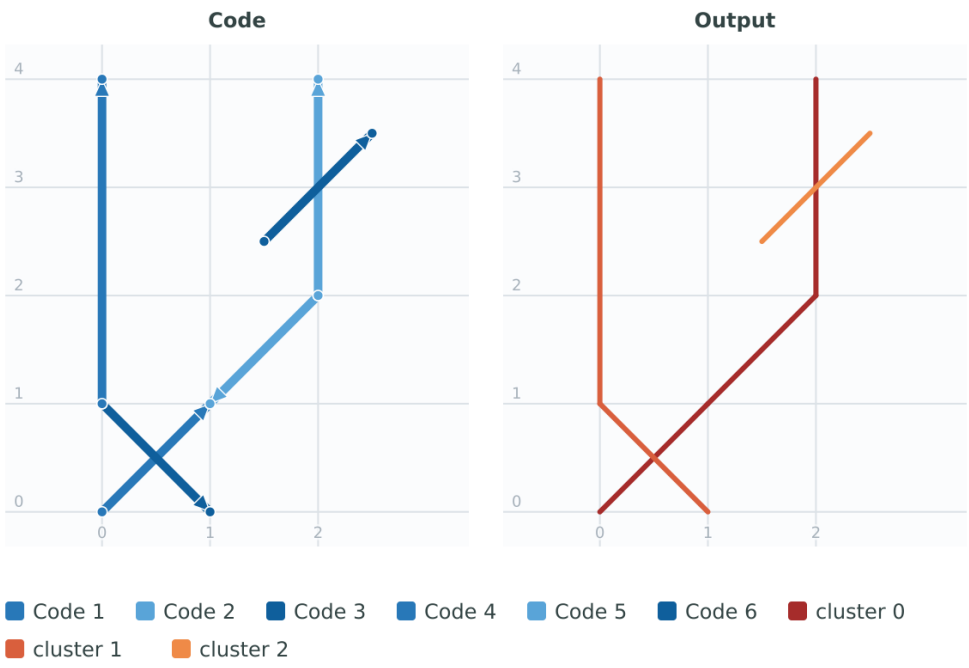
示例

This collection of line strings would form a single cluster using ST_ClusterIntersectingWin, but using ST_ClusterRelateWin can be clustered into three groups that connect only via their end points.

Code

```
WITH data(id, geom) AS (
  VALUES (1, 'LINESTRING(0 0,1 1)'::geometry),
         (2, 'LINESTRING(2 2,1 1)'::geometry),
         (3, 'LINESTRING(0 1,1 0)'::geometry),
         (4, 'LINESTRING(0 1,0 4)'::geometry),
         (5, 'LINESTRING(2 2,2 4)'::geometry),
         (6, 'LINESTRING(1.5 2.5,2.5 3.5)'::geometry)
), clustered AS (
  SELECT id, geom,
         ST_ClusterRelateWin(geom, '****0****') OVER () AS cluster
  FROM data
)
SELECT cluster,
       string_agg(id::text, ',' ORDER BY id) AS members,
       ST_Collect(geom ORDER BY id) AS cluster_geom
FROM clustered
GROUP BY cluster
ORDER BY cluster;
```

Figure



相关信息

ST_Relate, ST_ClusterIntersectingWin, ST_ClusterWithinWin

7.17.5 ST_MinimumSpanningTree

ST_MinimumSpanningTree — Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.

Synopsis

integer **ST_MinimumSpanningTree**(geometry winset geom);

描述

A window function that builds connected graphs of line strings based on the Minimum Spanning Tree (MST) of the input geometries. The return value is the cluster number that the geometry argument participates in, or zero if it is not part of the minimum tree.

The Minimum Spanning Tree connects all geometries in the window partition such that the total length of the connecting lines is minimized. If the graph is not fully connected (e.g. infinite distance between some geometries), it produces a Minimum Spanning Forest, and each connected component is assigned a unique tree ID.

Availability: 3.7.0

Requires GEOS >= 3.15.0

示例

These six edges describe a square with both diagonals. The minimum spanning tree keeps three edges and excludes the rest.

Code

```
WITH edges(id, geom) AS (
  VALUES
    (1, 'LINESTRING(0 0,1 0)'::geometry),
    (2, 'LINESTRING(0 0,0 1)'::geometry),
    (3, 'LINESTRING(1 1,0 1)'::geometry),
    (4, 'LINESTRING(1 1,1 0)'::geometry),
    (5, 'LINESTRING(0 0,1 1)'::geometry),
    (6, 'LINESTRING(1 0,0 1)'::geometry)
), marked AS (
  SELECT id,
         geom,
         ST_MinimumSpanningTree(geom) OVER (ORDER BY id) AS tree_id
  FROM edges
)
SELECT string_agg(id::text, ',' ORDER BY id)
         FILTER (WHERE tree_id > 0) AS tree_edge_ids,
       ST_Collect(geom ORDER BY id)
         FILTER (WHERE tree_id > 0) AS tree_edges,
       string_agg(id::text, ',' ORDER BY id)
         FILTER (WHERE tree_id = 0) AS excluded_edge_ids,
       ST_Collect(geom ORDER BY id)
         FILTER (WHERE tree_id = 0) AS excluded_edges
FROM marked;
```

Output

```
-[ RECORD 1 ]-----
tree_edge_ids     | 1,2,3
tree_edges        | MULTILINESTRING((0 0,1 0),(0 0,0 1),(1 1,0 1))
excluded_edge_ids | 4,5,6
excluded_edges    | MULTILINESTRING((1 1,1 0),(0 0,1 1),(1 0,0 1))
```

相关信息

ST_ClusterDBSCAN, ST_ClusterKMeans, ST_ClusterIntersectingWin

7.17.6 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 k , float8 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

Define the parcel geometries once in a CTE and reuse them across the executable examples.

Parcels color-coded by cluster number (`cid`).

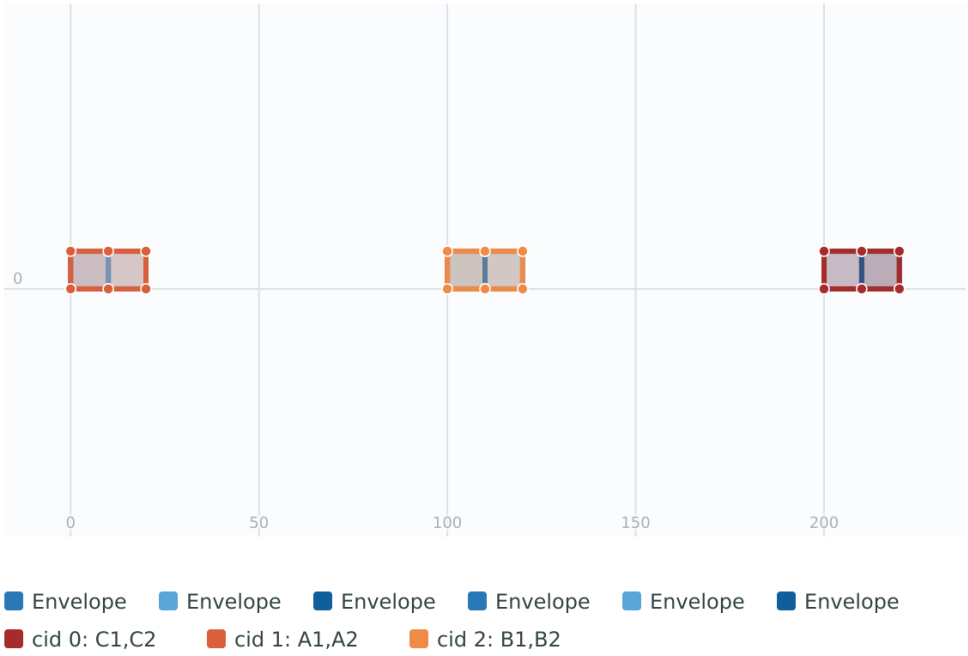
Code

```
WITH parcels(parcel_id, geom) AS (
  VALUES
    ('A1', ST_MakeEnvelope(0, 0, 10, 10)),
    ('A2', ST_MakeEnvelope(10, 0, 20, 10)),
    ('B1', ST_MakeEnvelope(100, 0, 110, 10)),
    ('B2', ST_MakeEnvelope(110, 0, 120, 10)),
    ('C1', ST_MakeEnvelope(200, 0, 210, 10)),
    ('C2', ST_MakeEnvelope(210, 0, 220, 10))
), clustered AS (
  SELECT parcel_id,
         geom,
         ST_ClusterKMeans(geom, 3) OVER (ORDER BY parcel_id) AS cid
  FROM parcels
)
SELECT cid,
       string_agg(parcel_id, ',' ORDER BY parcel_id) AS parcels,
       ST_Union(geom) AS cluster_geom
FROM clustered
GROUP BY cid
ORDER BY cid;
```

Output

```
0 | C1,C2 | POLYGON((200 10,210 10,220 10,220 0,210 0,200 0,200 10))
1 | A1,A2 | POLYGON((0 10,10 10,20 10,20 0,10 0,0 0,0 10))
2 | B1,B2 | POLYGON((100 10,110 10,120 10,120 0,110 0,100 0,100 10))
```

Figure



按⊠型划分地⊠集群：

Code

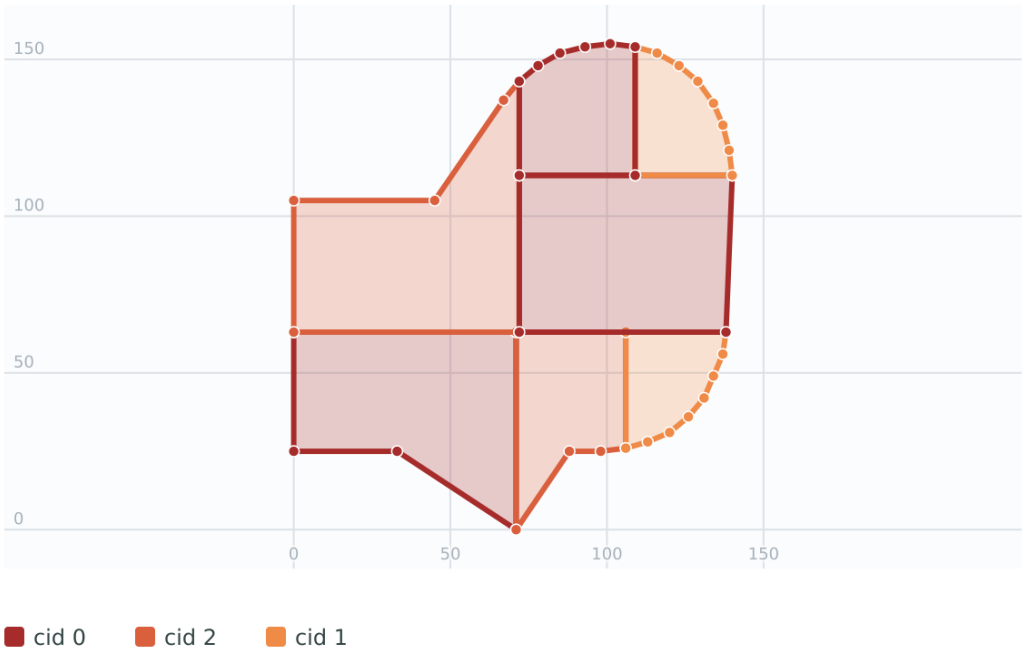
```
WITH parcel_geoms AS (
  SELECT geom
  FROM ST_Subdivide(
    ST_Buffer('SRID=3857;LINESTRING(40 100,98 100,100 150,60 90)'::geometry,
              40, 'endcap=square'),
    12) AS geom
), parcels AS (
  SELECT lpad(row_number() OVER (ORDER BY ST_YMin(geom), ST_XMin(geom))::text, 3, '0') AS
    parcel_id,
         geom,
         ('{residential,commercial}'::text[])[1 + mod(row_number() OVER (ORDER BY ST_YMin(
    geom), ST_XMin(geom)), 2)] AS type
  FROM parcel_geoms
)
SELECT ST_ClusterKMeans(geom, 3) over (PARTITION BY type) AS cid,
       parcel_id,
       type,
       ST_SnapToGrid(geom, 1) AS parcel_geom
FROM parcels
ORDER BY type, parcel_id;
```

Output

```
cid | parcel_id |    type     | parcel_geom
----+-----------+-------------+-----------------------------------------------------------------
   0 | 001       | commercial  | POLYGON((33 60,0 60,0 98,71 98,71 35,33 60))
   2 | 003       | commercial  | POLYGON((106 98,138 98,137 91,134 84,131 77,126 71,120 66,113 63,106 61,106 98))
   2 | 005       | commercial  | POLYGON((140 148,138 98,72 98,72 148,140 148))
```

```
  1 | 007       | commercial  | POLYGON((109 189,116 187,123 183,129 178,134 171,137 164,139 156,140 148,109 148,109 ↩
    189))
  0 | 002       | residential | POLYGON((71 98,106 98,106 61,98 60,88 60,71 35,71 98))
  2 | 004       | residential | POLYGON((0 98,0 140,45 140,67 172,72 178,72 98,0 98))
  1 | 006       | residential | POLYGON((72 178,78 183,85 187,93 189,101 190,109 189,109 148,72 148,72 178))
```

Figure



Example: Clustering a preaggregated planetary-scale data population dataset using 3D clustering and weighting. Identify at least 20 regions based on Kontur Population Data that do not span more than 3000 km from their center:

Code

```
CREATE TABLE kontur_population_3000km_clusters AS
SELECT
    geom,
    ST_ClusterKMeans(
        ST_Force4D(
            ST_Transform(ST_Force3D(geom), 4978),
            mvalue => population
        ),
        20,
        max_radius => 3000000
    ) OVER () AS cid
FROM kontur_population;
```

The clustering produces 46 regions. Clusters are centered at well-populated regions such as New York and Moscow. Greenland forms one cluster, several island clusters span the antimeridian, and cluster edges follow the Earth's curvature.



Kontur population clustered with 3000 km maximum radius.

Cluster weighted city populations on the Earth in geocentric coordinates. The maximum radius is specified in meters, while the returned geometries stay in WGS 84 for display.

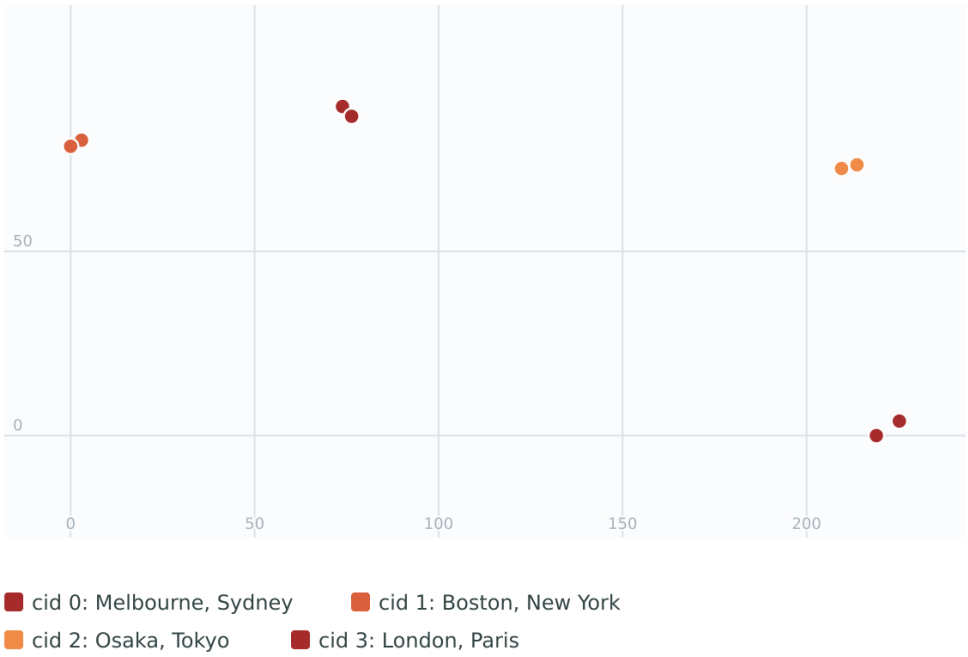
Code

```
WITH population(place, population, geom) AS (
  VALUES
    ('Boston', 5.0, ST_SetSRID(ST_Point(-71.06, 42.36), 4326)),
    ('New York', 20.0, ST_SetSRID(ST_Point(-74.01, 40.71), 4326)),
    ('London', 15.0, ST_SetSRID(ST_Point(-0.13, 51.51), 4326)),
    ('Paris', 11.0, ST_SetSRID(ST_Point(2.35, 48.86), 4326)),
    ('Osaka', 19.0, ST_SetSRID(ST_Point(135.50, 34.69), 4326)),
    ('Tokyo', 37.0, ST_SetSRID(ST_Point(139.69, 35.68), 4326)),
    ('Melbourne', 5.0, ST_SetSRID(ST_Point(144.96, -37.81), 4326)),
    ('Sydney', 5.0, ST_SetSRID(ST_Point(151.21, -33.87), 4326))
), clustered AS (
  SELECT place,
         geom,
         ST_ClusterKMeans(
           ST_Force4D(
             ST_Transform(ST_Force3D(geom), 4978),
             mvalue => population
           ),
           4,
           max_radius => 3000000
         ) OVER (ORDER BY place) AS cid
  FROM population
)
SELECT cid,
       string_agg(place, ', ' ORDER BY place) AS places,
       ST_Collect(geom ORDER BY place) AS cluster_geom
FROM clustered
GROUP BY cid
ORDER BY cid;
```

Output

```
0 | Melbourne, Sydney | MULTIPOINT((144.96 -37.81),(151.21 -33.87))
1 | Boston, New York | MULTIPOINT((-71.06 42.36),(-74.01 40.71))
2 | Osaka, Tokyo | MULTIPOINT((135.5 34.69),(139.69 35.68))
3 | London, Paris | MULTIPOINT((-0.13 51.51),(2.35 48.86))
```

Figure



相关信息

ST_ClusterDBSCAN, ST_ClusterIntersectingWin, ST_ClusterWithinWin, ST_ClusterIntersecting, ST_Cluster
ST_Subdivide, ST_Force3D, ST_Force4D,

7.17.7 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

Code

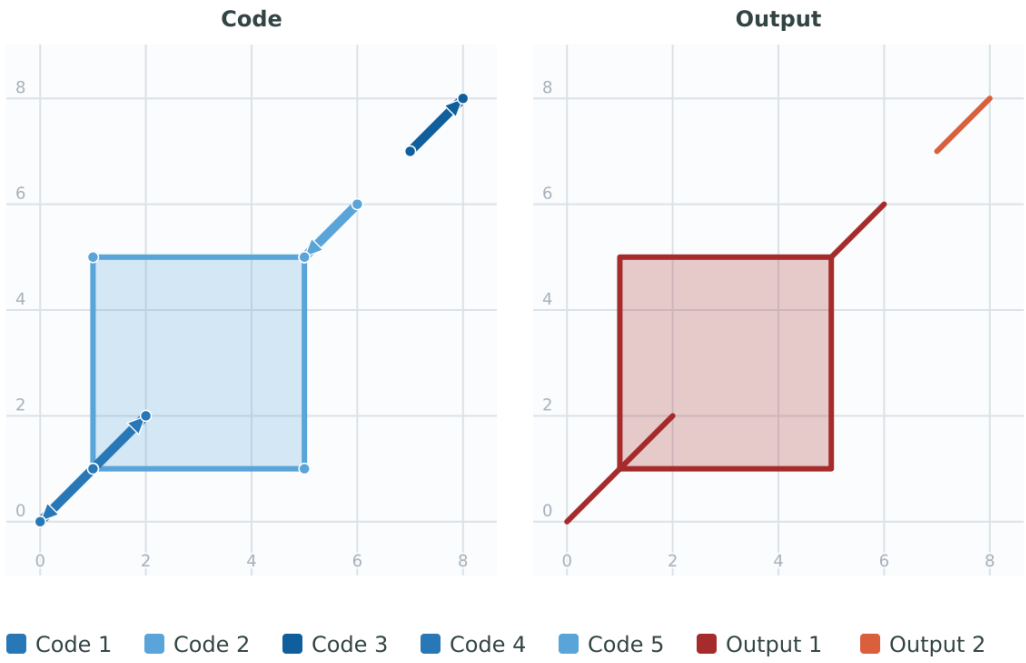
```
WITH testdata AS
  (SELECT unnest(ARRAY['LINESTRING (0 0,1 1)'::geometry,
                'LINESTRING (5 5,4 4)'::geometry,
                'LINESTRING (6 6,7 7)'::geometry,
                'LINESTRING (0 0,-1 -1)'::geometry,
                'POLYGON ((0 0,4 0,4 4,0 4,0 0))'::geometry]) AS geom)

SELECT unnest(ST_ClusterWithin(geom, 1.4)) FROM testdata;
```

Output

```
GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),LINESTRING(5 5,4 4),LINESTRING(0 0,-1 -1),POLYGON((0 0,4 0,4 4,0 4,0 0)))
GEOMETRYCOLLECTION(LINESTRING(6 6,7 7))
```

Figure



相关信息

ST_ClusterWithinWin, ST_ClusterDBSCAN, ST_ClusterIntersecting, ST_ClusterIntersectingWin

7.17.8 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

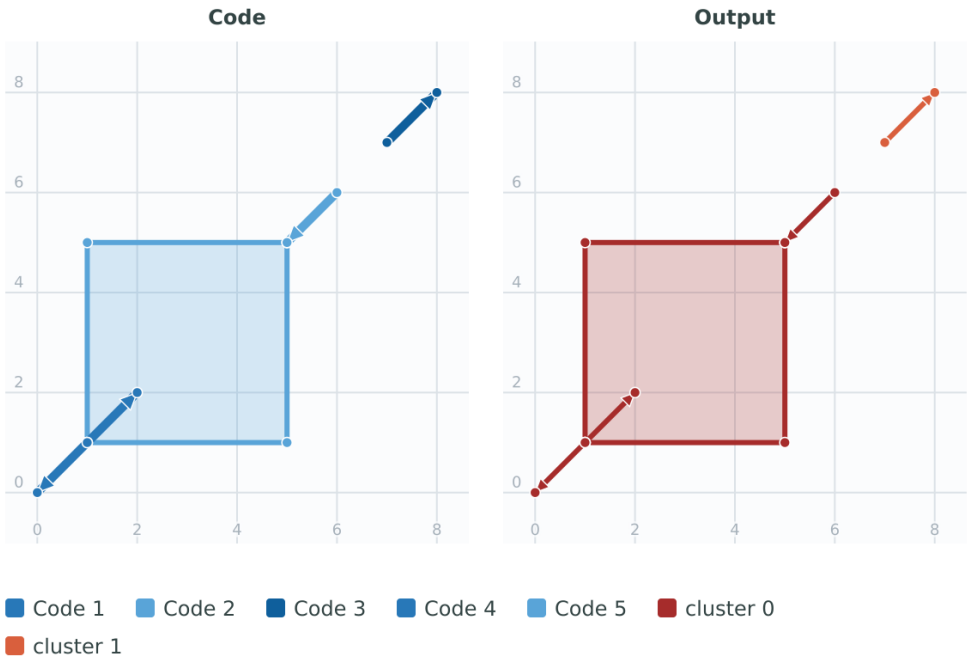
Code

```
WITH testdata AS (
  SELECT id, geom::geometry FROM (
  VALUES  (1, 'LINESTRING (0 0,1 1)'),
          (2, 'LINESTRING (5 5,4 4)'),
          (3, 'LINESTRING (6 6,7 7)'),
          (4, 'LINESTRING (0 0,-1 -1)'),
          (5, 'POLYGON ((0 0,4 0,4 4,0 4,0 0))')) AS t(id, geom)
)
SELECT id,
  geom,
  ST_ClusterWithinWin(geom, 1.4) OVER () AS cluster
FROM testdata;
```

Output

```
id |          st_astext            | cluster
----+-------------------------------+---------
  1 | LINESTRING(0 0,1 1)           |       0
  2 | LINESTRING(5 5,4 4)           |       0
  3 | LINESTRING(6 6,7 7)           |       1
  4 | LINESTRING(0 0,-1 -1)         |       0
  5 | POLYGON((0 0,4 0,4 4,0 4,0 0)) |       0
```

Figure



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

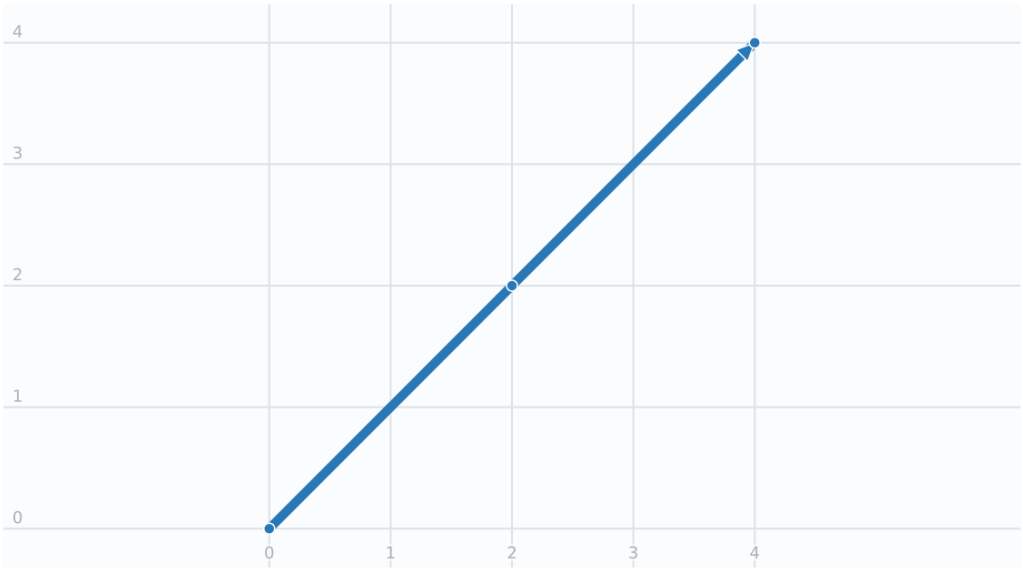
Code

```
SELECT Box2D('LINESTRING(1 2,3 4,5 6)'::geometry);
```

Output

```
BOX(1 2,5 6)
```

Figure



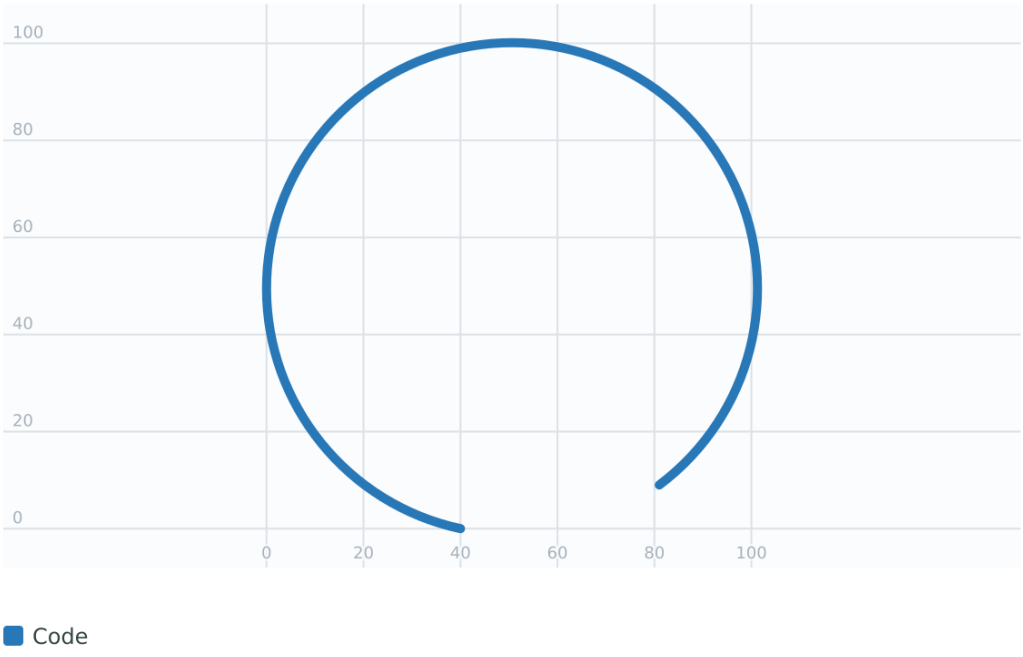
Code

```
SELECT Box2D('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'::geometry);
```

Output

```
BOX(220186.99512189245 150406,220288.24878054656 150506.12682932706)
```

Figure



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

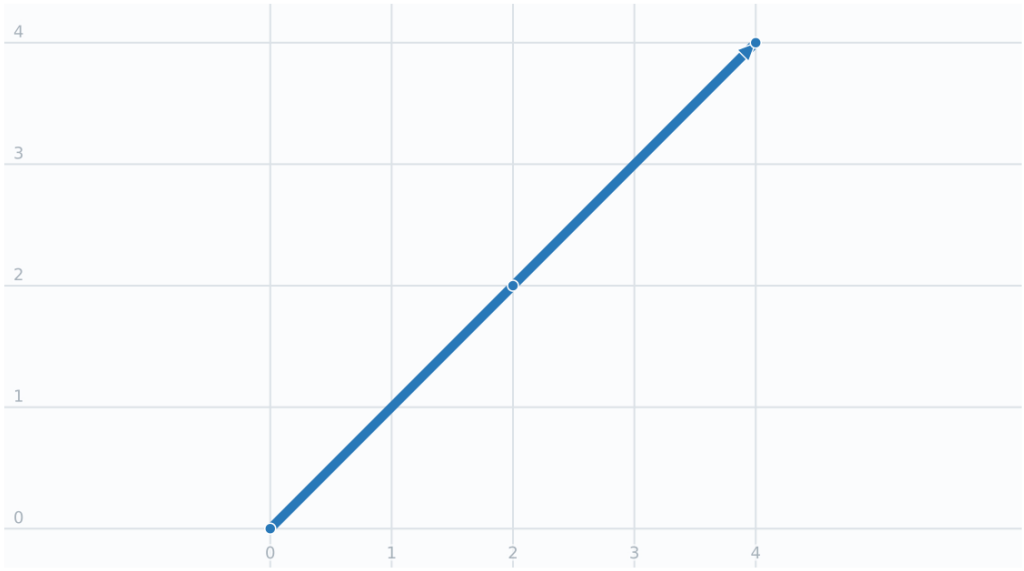
Code

```
SELECT Box3D('LINESTRING(1 2 3,3 4 5,5 6 5)'::geometry);
```

Output

```
BOX3D(1 2 3,5 6 5)
```

Figure



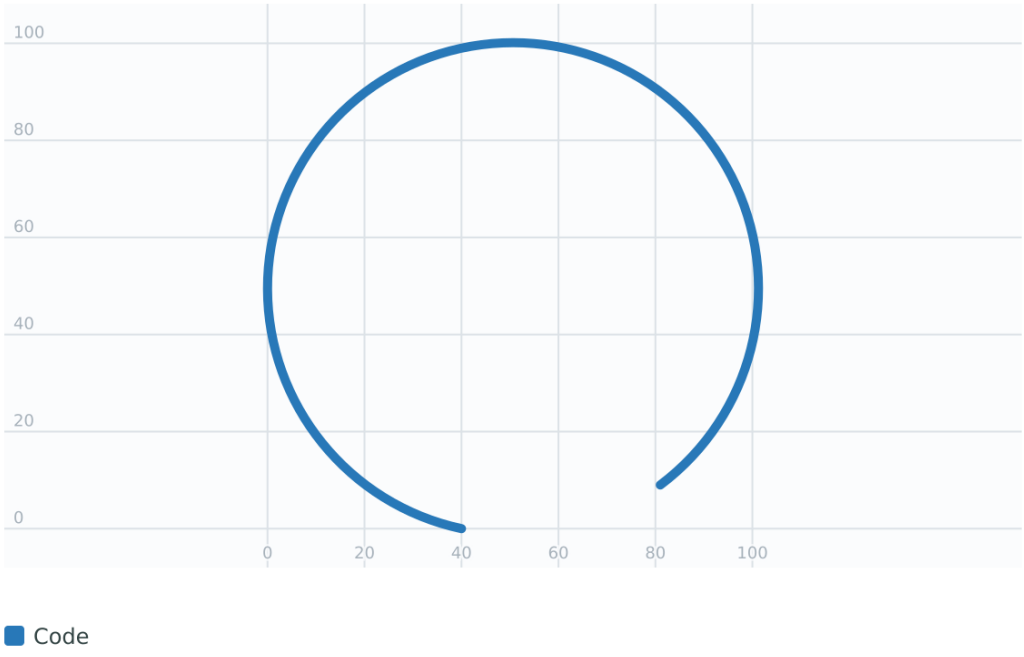
Code

```
SELECT Box3D('CIRCULARSTRING(220268 150415 1,220227 150505 1,220227 150406 1)'::geometry);
```

Output

```
BOX3D(220186.99512189245 150406 1,220288.24878054656 150506.12682932706 1)
```

Figure



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
⊠于表名或命名空⊠（schema）的名称，如果包含特殊字符或引号，可能需要⊠行特殊⊠理。有些用⊠会使用“⊠⊠符⊠⊠⊠⊠”，“通⊠加上双引号来生成字符串，然后去掉最前面和最后面的双引号。而⊠于几何列（geometry column），⊠直接⊠⊠原始⊠，而不⊠行任何⊠外的⊠理。”

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

Code

```
SELECT ST_EstimatedExtent('ny', 'edges', 'geom');
```

Output

```
st_estimatedextent
---------------------------------------------------
 BOX(-8877653 4912316,-8010225.5 5589284)
(1 row)
```

Code

```
SELECT ST_EstimatedExtent('feature_poly', 'geom');
```

Output

```
st_estimatedextent
------------------------------------------------------------------------
 BOX(-124.659652709961 24.6830825805664,-67.7798080444336 49.0012092590332)
(1 row)
```

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.

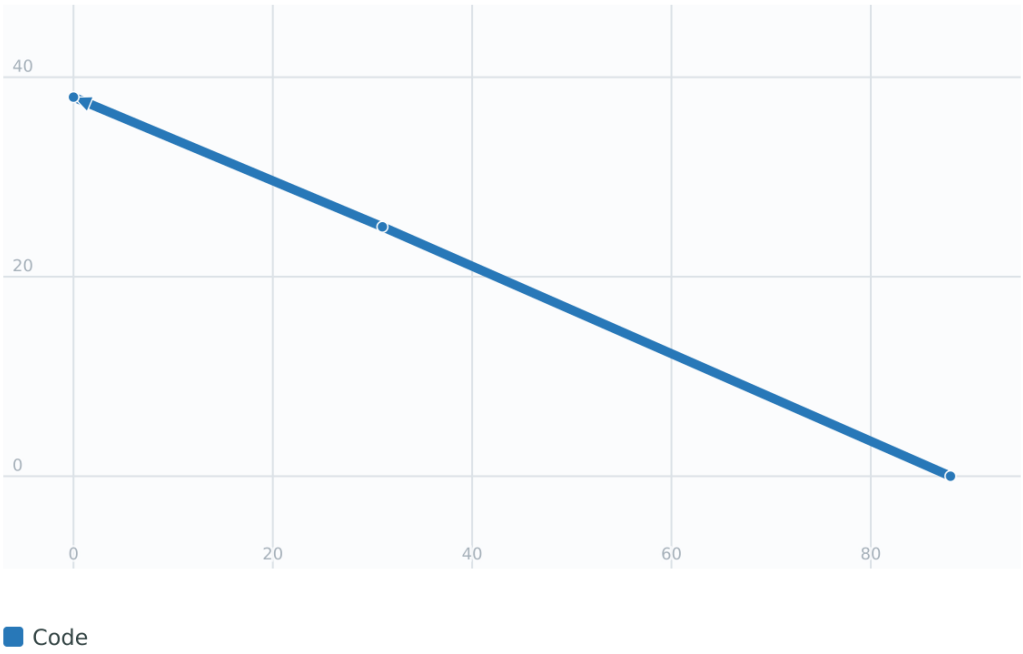
Code

```
SELECT CAST(ST_Expand(ST_GeomFromText('LINESTRING(2312980 110676,2312923 110701,2312892 ↩
    110714)', 2163), 10) As box2d);
```

Output

```
BOX(2312882 110666,2312990 110724)
```

Figure



This example expands a 3D box by 10 meters.

Code

```
SELECT ST_Expand(CAST('BOX3D(778783 2951741 1,794875 2970042.61545891 10)' AS box3d), 10)
```

Output

```
BOX3D(778773 2951731 -9,794885 2970052.61545891 20)
```

This example renders the text representation of a 10-meter expanded box around a point geometry.

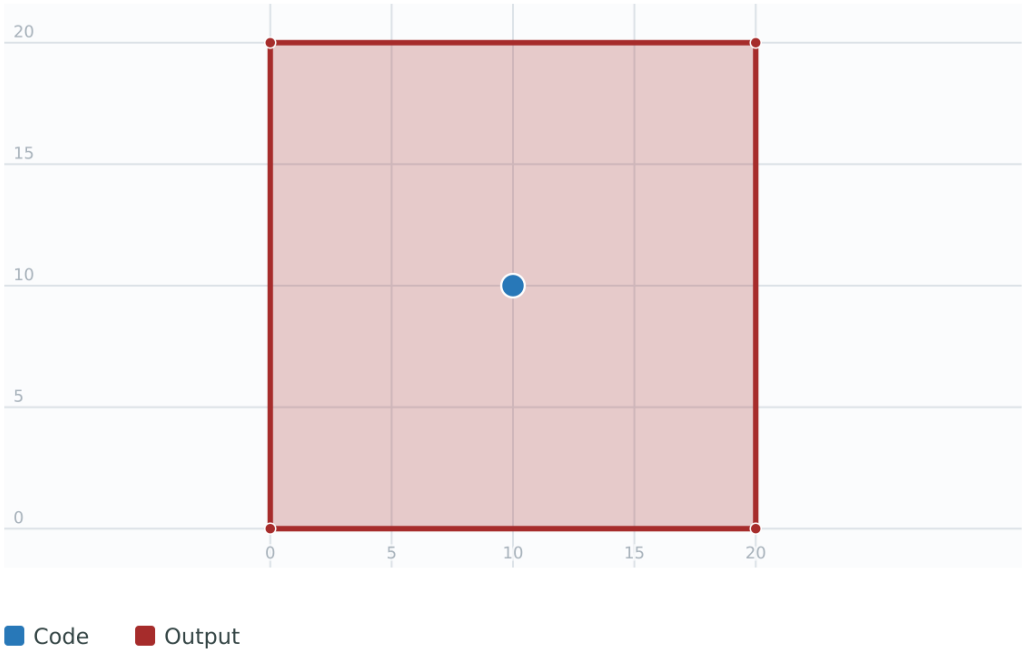
Code

```
SELECT ST_Expand('SRID=2163;POINT(2312980 110676)'::geometry, 10);
```

Output

```
SRID=2163;POLYGON((2312970 110666,2312970 110686,2312990 110686,2312990 110666,2312970 110666))
```

Figure



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)

Code

```
SELECT ST_Extent(geom) AS bextent FROM sometable;
```

Output

```
BOX(739651.875 2908247.25,794875.8125 2970042.75)
```

Return extent of each geometry category.

Code

```
SELECT ST_Extent(geom) as bextent
FROM sometable
GROUP BY category ORDER BY category;
```

Output

```
bextent                       |         name
------------------------------------------------------+----------------
 BOX(778783.5625 2951741.25,794875.8125 2970042.75) | A
 BOX(751315.8125 2919164.75,765202.6875 2935417.25) | B
 BOX(739651.875 2917394.75,756688.375 2935866)      | C
```

This example converts the extent back into a geometry and renders its extended text representation.

Code

```
SELECT ST_SetSRID(ST_Extent(geom), 2249) AS bextent FROM sometable;
```

Output

```
SRID=2249;POLYGON((739651.875 2908247.25,739651.875 2970042.75,794875.8125 2970042.75,
794875.8125 2908247.25,739651.875 2908247.25))
```

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

Code

```
SELECT ST_3DExtent(foo.geom) As b3extent
FROM (SELECT ST_MakePoint(x, y, z) As geom
    FROM generate_series(1, 3) As x
        CROSS JOIN generate_series(1, 2) As y
        CROSS JOIN generate_series(0, 2) As Z) As foo;
```

Output

```
BOX3D(1 1 0,3 2 2)
```

This example returns the extent of several elevated CircularStrings.

Code

```
SELECT ST_3DExtent(foo.geom) As b3extent
FROM (SELECT ST_Translate(ST_Force3DZ(ST_LineToCurve(ST_Buffer(ST_Point(x, y), 1))), 0, 0,
    z) As geom
    FROM generate_series(1, 3) As x
        CROSS JOIN generate_series(1, 2) As y
        CROSS JOIN generate_series(0, 2) As Z) As foo;
```

Output

```
BOX3D(0 0 0,4 3 2)
```

相关信息

ST_Extent, ST_Force3DZ, ST_SetSRID

7.18.7 ST_MakeBox2D

ST_MakeBox2D — 构建由两个 2D 点几何构形定义的 BOX2D。

Synopsis

box2d **ST_MakeBox2D**(geometry pointLowLeft, geometry pointUpRight);

描述

构建由两个 Point 几何构形定义的box2d。构成于空行范围两两很有用。

示例

Return all features that fall within or partly reside in a US National Atlas coordinate bounding box. It is assumed here that the geometries are stored with SRID=2163 (US National Atlas equal area).

Code

```
SELECT feature_id, feature_name, geom
FROM features
WHERE geom && ST_SetSRID(
  ST_MakeBox2D(
    ST_Point(-989502.1875, 528439.5625),
    ST_Point(-987121.375, 529933.1875)
  ),
  2163
);
```

相关信息

ST_Point, ST_SetSRID, ST_SRID

7.18.8 ST_3DMakeBox

ST_3DMakeBox — 构建由构个 3D 点几何构形定构的 BOX3D。

Synopsis

box3d **ST_3DMakeBox**(geometry point3DLowLeftBottom, geometry point3DUpRightTop);

描述

构建由构个 3D 点几何构形定构的box3d。

 构个函数支持 3D 且不会降低 z-index。

更改：2.0.0 在之前的版本中，构曾构被称构 ST_MakeBox3D

示例

Code

```
SELECT ST_3DMakeBox(ST_MakePoint(-989502.1875, 528439.5625, 10),
    ST_MakePoint(-987121.375 , 529933.1875, 10))
```

Output

```
BOX3D(-989502.1875 528439.5625 10,-987121.375 529933.1875 10)
```

相关信息

ST_MakePoint, ST_SetSRID, ST_SRID

7.18.9 ST_XMax

ST_XMax — 返回 2D 或 3D 构界框或几何体的 X 最大构。

Synopsis

float **ST_XMax**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 界框或几何体的 X 最大。



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

Code

```
SELECT ST_XMax('BOX3D(1 2 3,4 5 6)');
```

Output

```
4
```

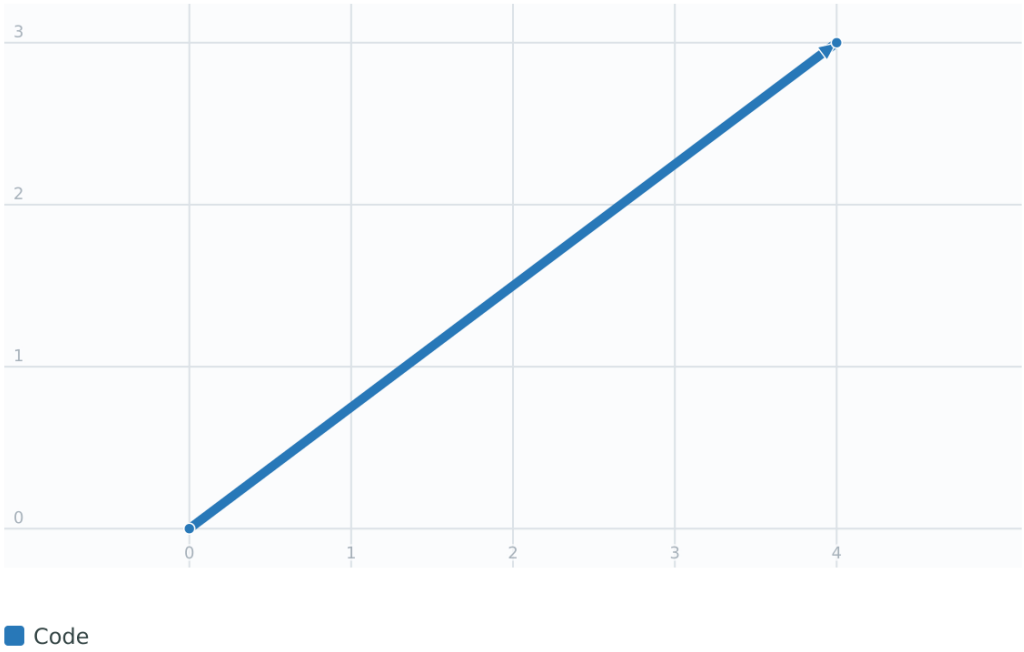
Code

```
SELECT ST_XMax('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
5
```

Figure



Code

```
SELECT ST_XMax(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
3
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_XMax('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

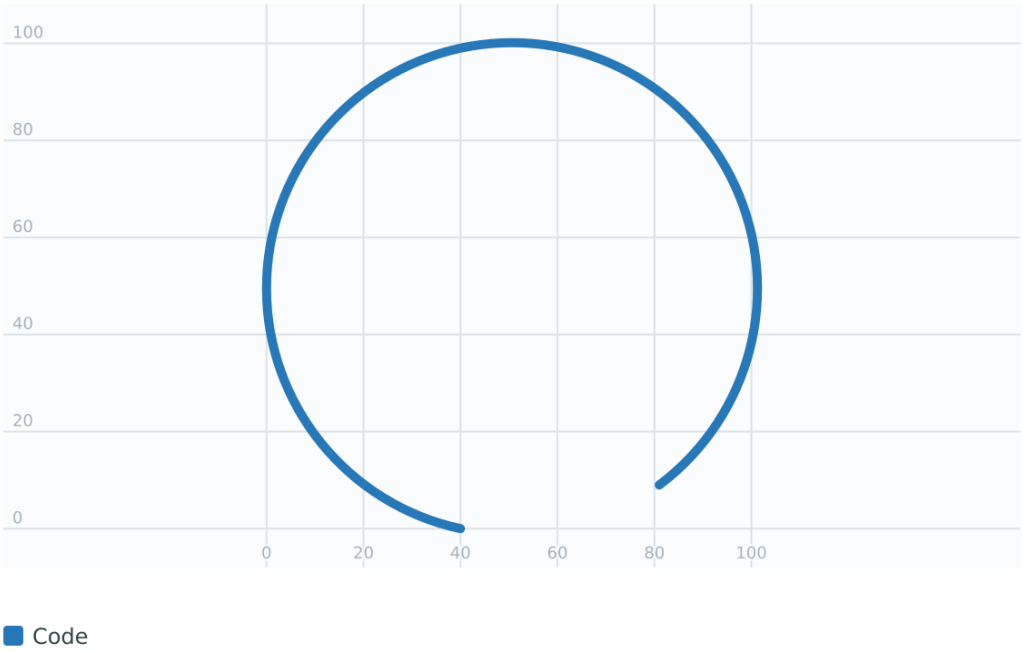
Code

```
SELECT ST_XMax('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
220288.248780547
```

Figure



相关信息

ST_XMin, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

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

Code

```
SELECT ST_XMin('BOX3D(1 2 3,4 5 6)');
```

Output

```
1
```

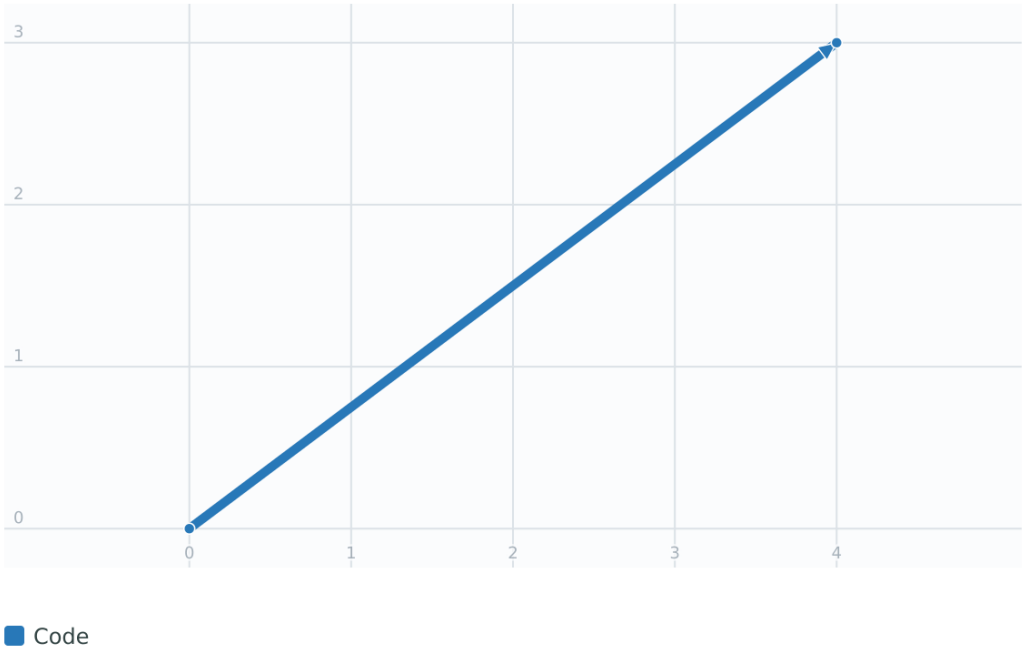
Code

```
SELECT ST_XMin('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
1
```

Figure



Code

```
SELECT ST_XMin(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
-3
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_XMin('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

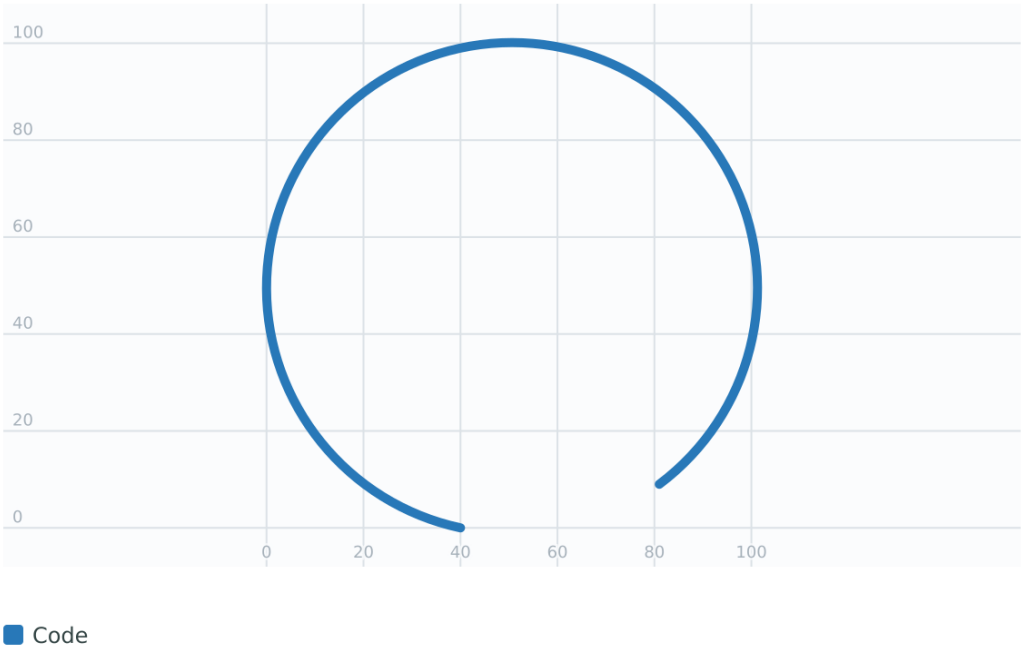
Code

```
SELECT ST_XMin('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
220186.995121892
```

Figure



相关信息

ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

7.18.11 ST_YMax

ST_YMax — 返回 2D 或 3D ☒界框或几何体的 Y 最大☒。

Synopsis

float **ST_YMax**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D ☒界框或几何体的 Y 最大☒。

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

Code

```
SELECT ST_YMax('BOX3D(1 2 3,4 5 6)');
```

Output

```
5
```

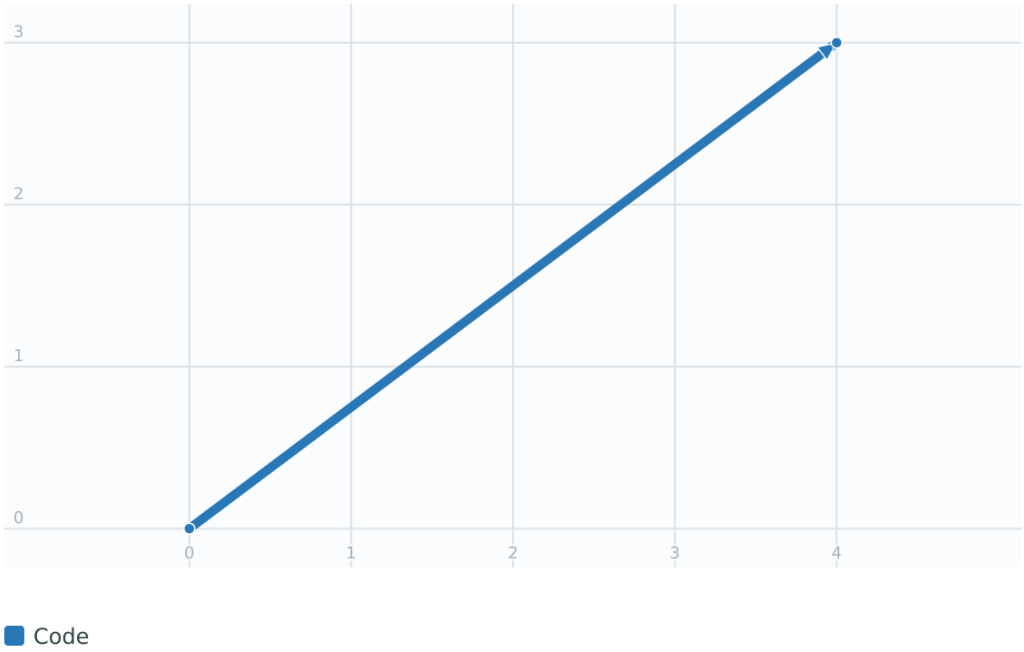
Code

```
SELECT ST_YMax('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
6
```

Figure



Code

```
SELECT ST_YMax(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
4
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_YMax('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

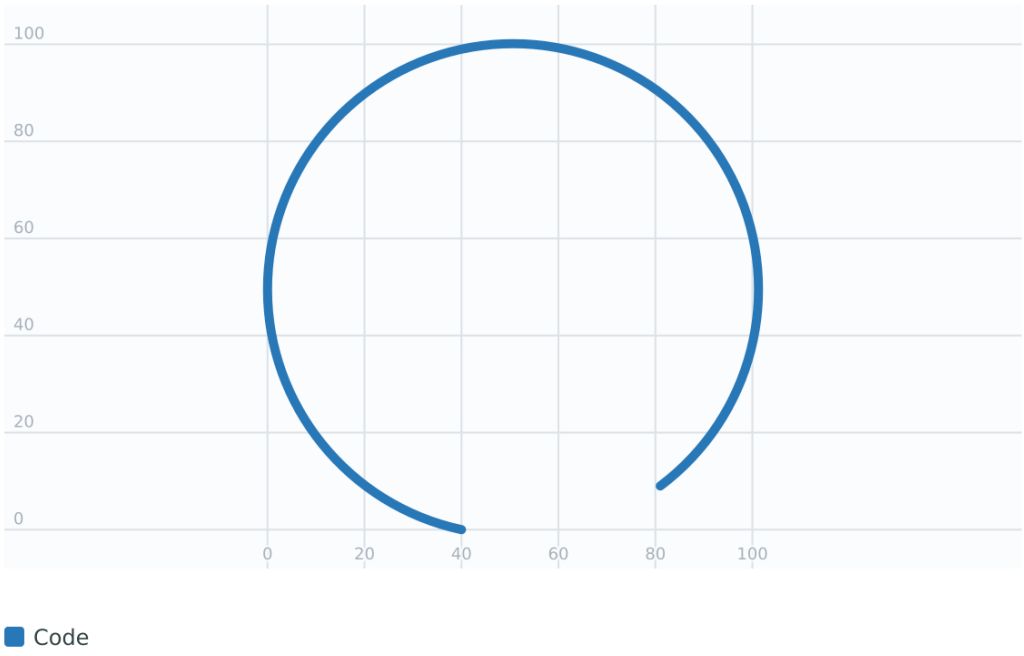
Code

```
SELECT ST_YMax('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
150506.126829327
```

Figure



相关信息

ST_XMin, ST_XMax, ST_YMin, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

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

Code

```
SELECT ST_YMin('BOX3D(1 2 3,4 5 6)');
```

Output

```
2
```

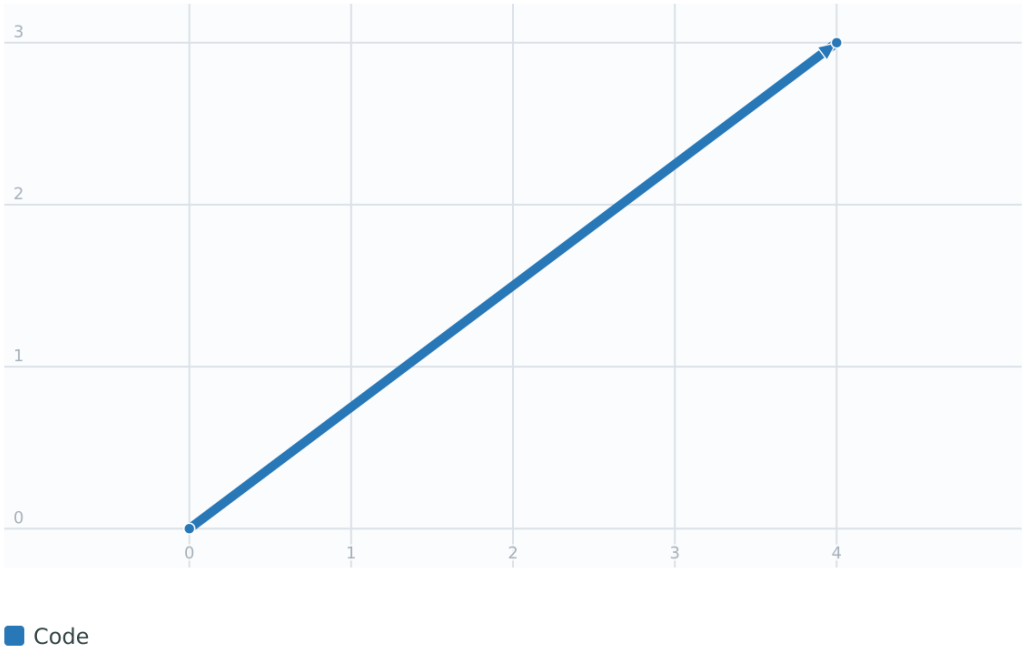
Code

```
SELECT ST_YMin('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
3
```

Figure



Code

```
SELECT ST_YMin(CAST('BOX(-3 2,3 4)' AS box2d));
```

Output

```
2
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_YMin('LINESTRING(1 3,5 6)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

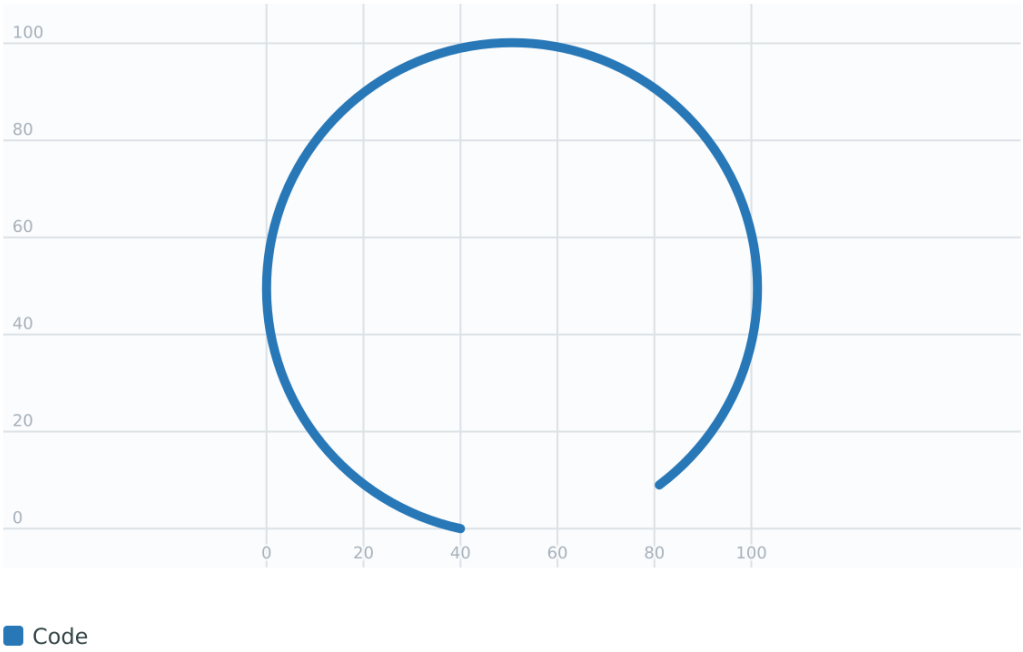
Code

```
SELECT ST_YMin('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
150406
```

Figure



相关信息

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MMax

7.18.13 ST_ZMax

ST_ZMax — 返回 2D 或 3D 盒界框或几何体的 Z 最大盒。

Synopsis

float **ST_ZMax**(box3d aGeomorBox2DorBox3D);

描述

返回 2D 或 3D 盒界框或几何体的 Z 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

Code

```
SELECT ST_ZMax('BOX3D(1 2 3,4 5 6)');
```

Output

```
6
```

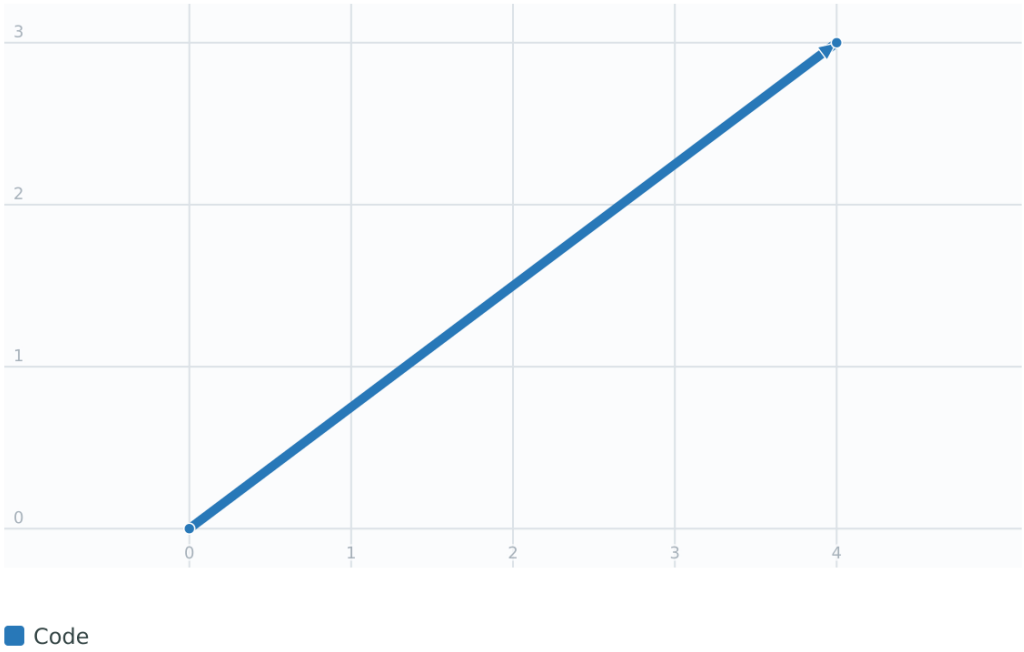
Code

```
SELECT ST_ZMax('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
7
```

Figure



Code

```
SELECT ST_ZMax('BOX3D(-3 2 1,3 4 1)');
```

Output

```
1
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_ZMax('LINESTRING(1 3 4,5 6 7)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

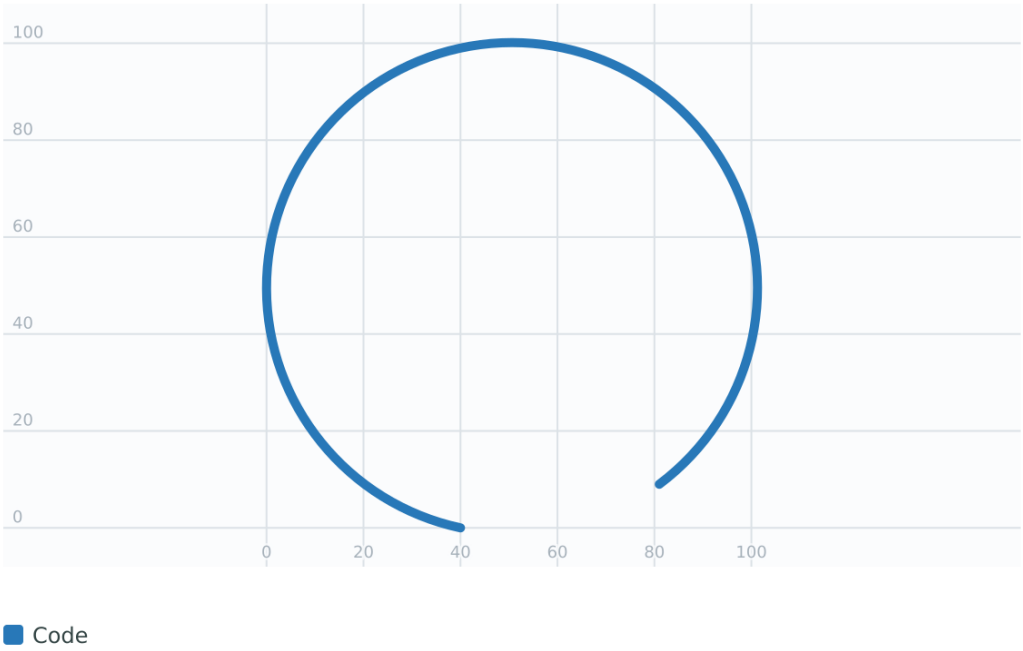
Code

```
SELECT ST_ZMax('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry)
    ;
```

Output

```
3
```

Figure



相关信息

ST_GeomFromEWKT, ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_MMin, ST_MMax

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
⌧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

Code

```
SELECT ST_ZMin('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

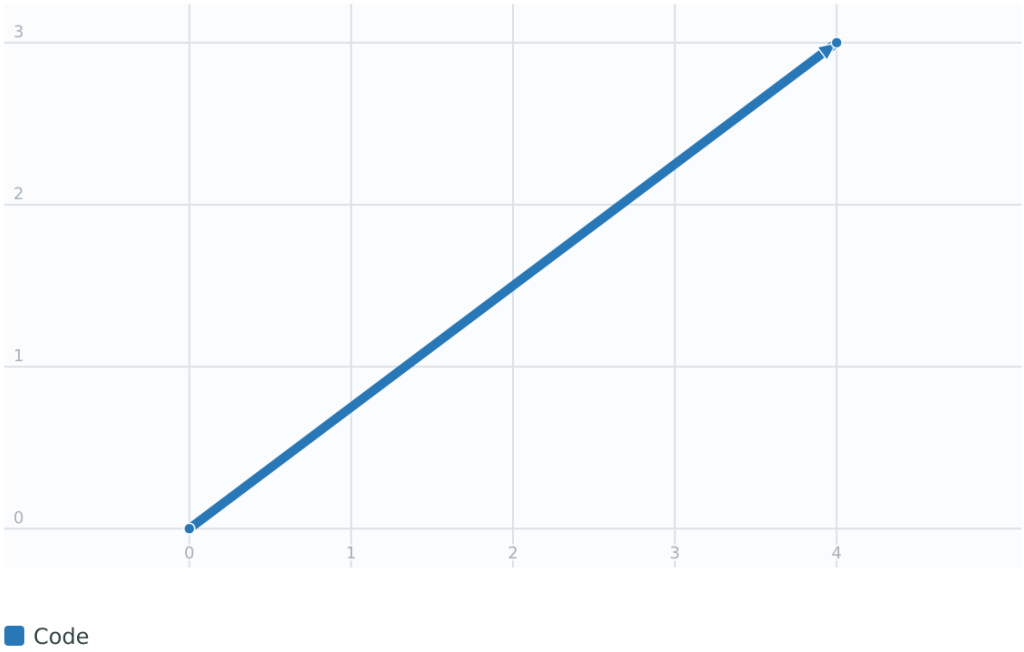
Code

```
SELECT ST_ZMin('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
4
```

Figure



Code

```
SELECT ST_ZMin('BOX3D(-3 2 1,3 4 1)');
```

Output

```
1
```

Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to a BOX3D.

Code

```
SELECT ST_ZMin('LINESTRING(1 3 4,5 6 7)');
```

Output

```
ERROR: BOX3D parser - doesn't start with BOX3D(
```

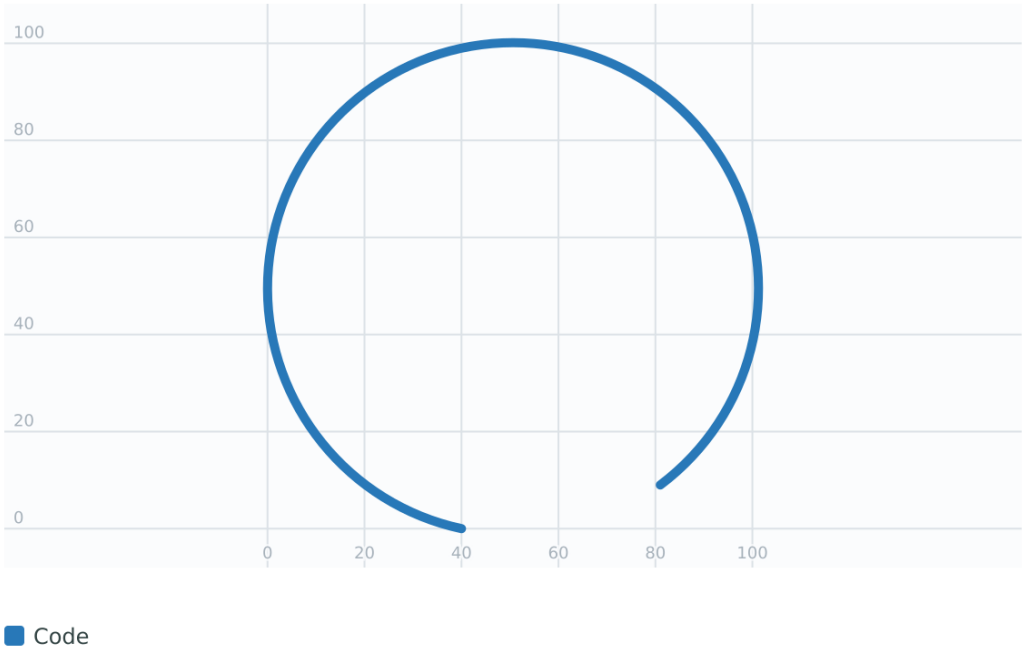
Code

```
SELECT ST_ZMin('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)'::geometry) ↩
    ;
```

Output

```
1
```

Figure



相关信息

ST_GeomFromEWKT, ST_GeomFromText, ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMax, ST_MMin, ST_MMax

7.18.15 ST_XSize

ST_XSize — Returns the X size of a 2D or 3D bounding box or a geometry.

Synopsis

float **ST_XSize**(box3d aGeomorBox2DorBox3D);

描述

Returns the X size of a 2D or 3D bounding box or a geometry, computed as `ST_XMax(aGeomorBox2DorBox3D) - ST_XMin(aGeomorBox2DorBox3D)`.



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⊠。

Availability: 3.7.0

示例

Code

```
SELECT ST_XSize('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

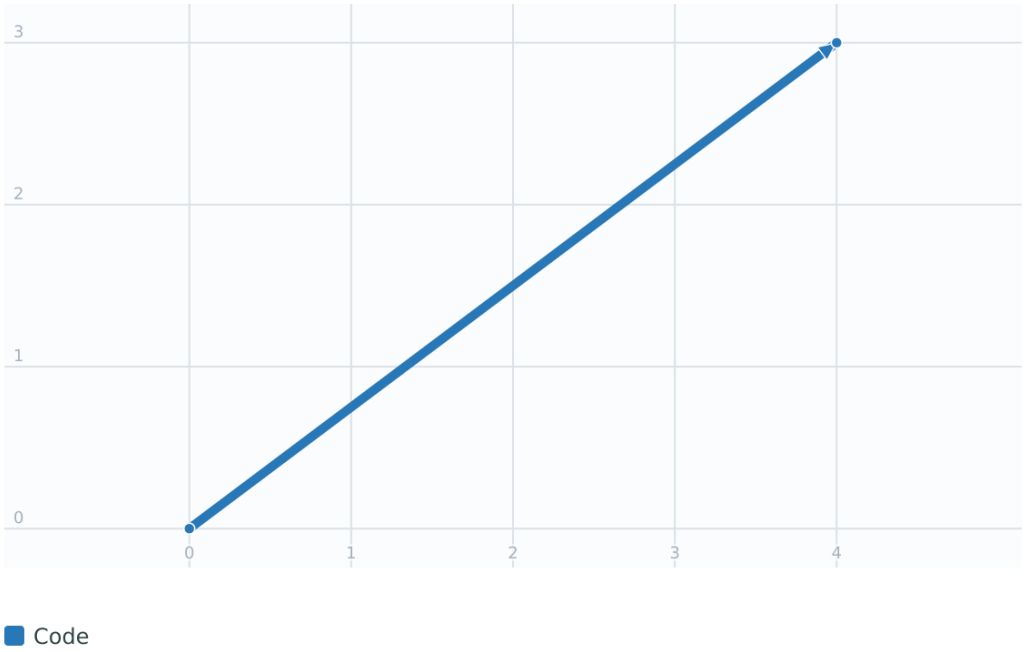
Code

```
SELECT ST_XSize('LINESTRING(1 3,5 6)'::geometry);
```

Output

```
4
```

Figure



相关信息

ST_XMin, ST_XMax, ST_YSize, ST_ZSize, ST_MSize

7.18.16 ST_YSize

ST_YSize — Returns the Y size of a 2D or 3D bounding box or a geometry.

Synopsis

float **ST_YSize**(box3d aGeomorBox2DorBox3D);

描述

Returns the Y size of a 2D or 3D bounding box or a geometry, computed as `ST_YMax(aGeomorBox2DorBox3D) - ST_YMin(aGeomorBox2DorBox3D)`.

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⌧。

Availability: 3.7.0

示例

Code

```
SELECT ST_YSize('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

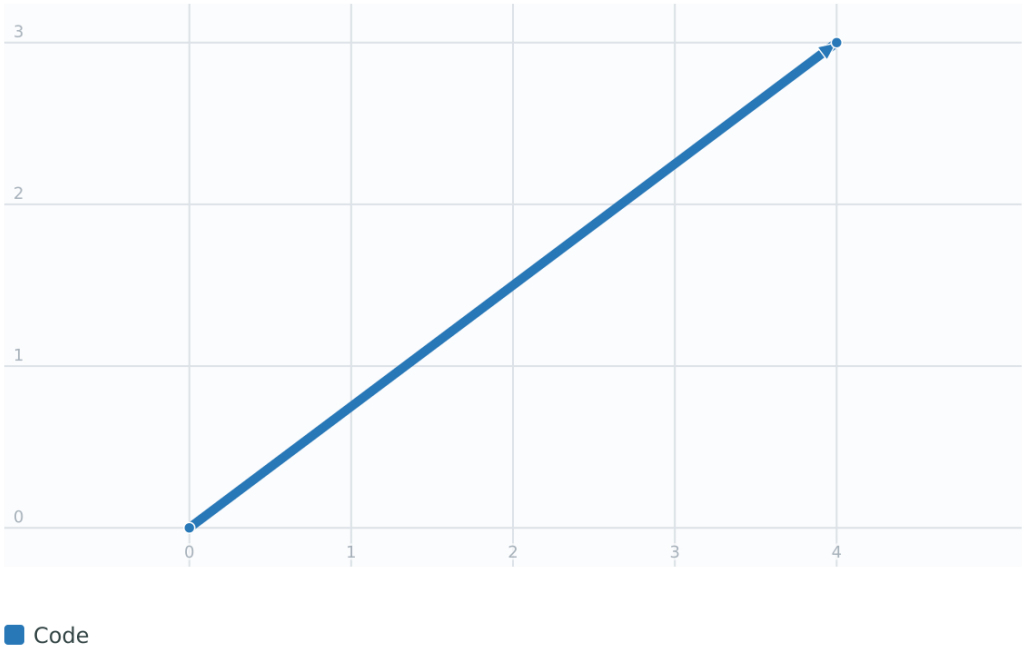
Code

```
SELECT ST_YSize('LINESTRING(1 3,5 6)'::geometry);
```

Output

```
3
```

Figure



相关信息

ST_YMin, ST_YMax, ST_XSize, ST_ZSize, ST_MSize

7.18.17 ST_ZSize

ST_ZSize — Returns the Z size of a 2D or 3D bounding box or a geometry.

Synopsis

float **ST_ZSize**(box3d aGeomorBox2DorBox3D);

描述

Returns the Z size of a 2D or 3D bounding box or a geometry, computed as `ST_ZMax(aGeomorBox2DorBox3D) - ST_ZMin(aGeomorBox2DorBox3D)`.



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☒。

Availability: 3.7.0

示例

Code

```
SELECT ST_ZSize('BOX3D(1 2 3,4 5 6)');
```

Output

```
3
```

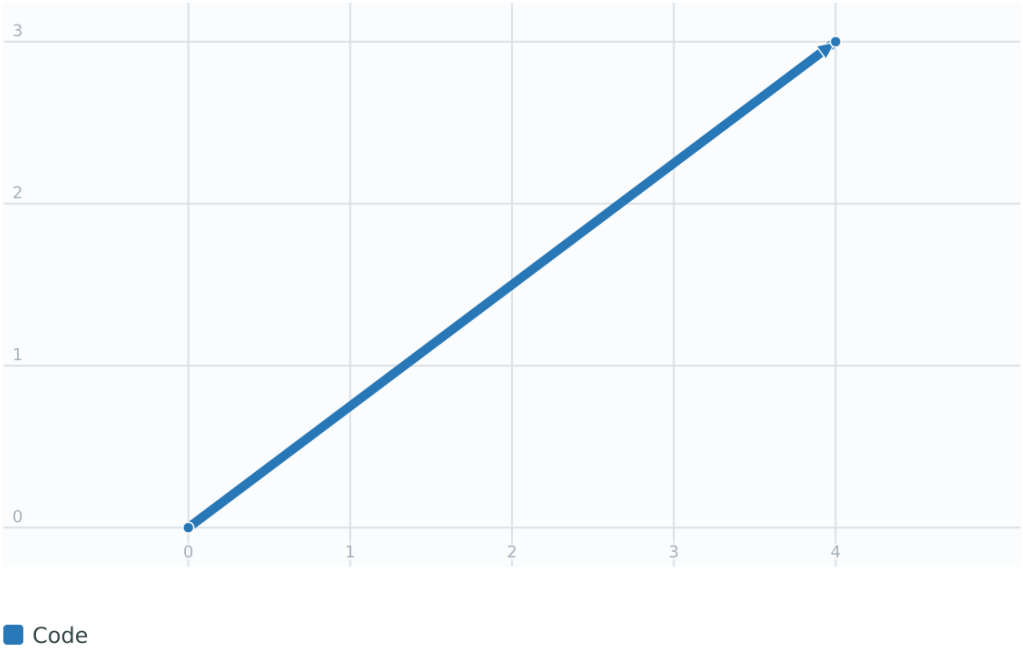
Code

```
SELECT ST_ZSize('LINESTRING(1 3 4,5 6 7)'::geometry);
```

Output

```
3
```

Figure



相关信息

ST_ZMin, ST_ZMax, ST_XSize, ST_YSize, ST_MSize

7.18.18 ST_MMin

ST_MMin — 返回几何⌧象的 M ⌧最小⌧。

Synopsis

float **ST_MMin**(geometry geom);

描述

Returns the M minima of a geometry, or null if the geometry lacks M values.

⌧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

Code

```
SELECT ST_MMin('POINT M (1 2 3)');
```

Output

```
3
```

相关信息

ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMax, ST_MSize

7.18.19 ST_MMax

ST_MMax — Returns the M maxima of a geometry.

Synopsis

float **ST_MMax**(geometry geom);

描述

Returns the M maxima of a geometry, or null if the geometry lacks M values.

⌧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

Code

```
SELECT ST_MMax('POINT M (1 2 3)');
```

Output

```
3
```

相关信息

ST_XMin, ST_XMax, ST_YMin, ST_YMax, ST_ZMin, ST_ZMax, ST_MMin, ST_MSize

7.18.20 ST_MSize

ST_MSize — Returns the M size of a geometry.

Synopsis

float **ST_MSize**(geometry geom);

描述

Returns the M size of a geometry, computed as `ST_MMax(geom) - ST_MMin(geom)`, or null if the geometry lacks M values.

☒函数支持 3d 并且不会☒失 z-index。

此方法支持☒形字符串和曲☒。

Availability: 3.7.0

示例

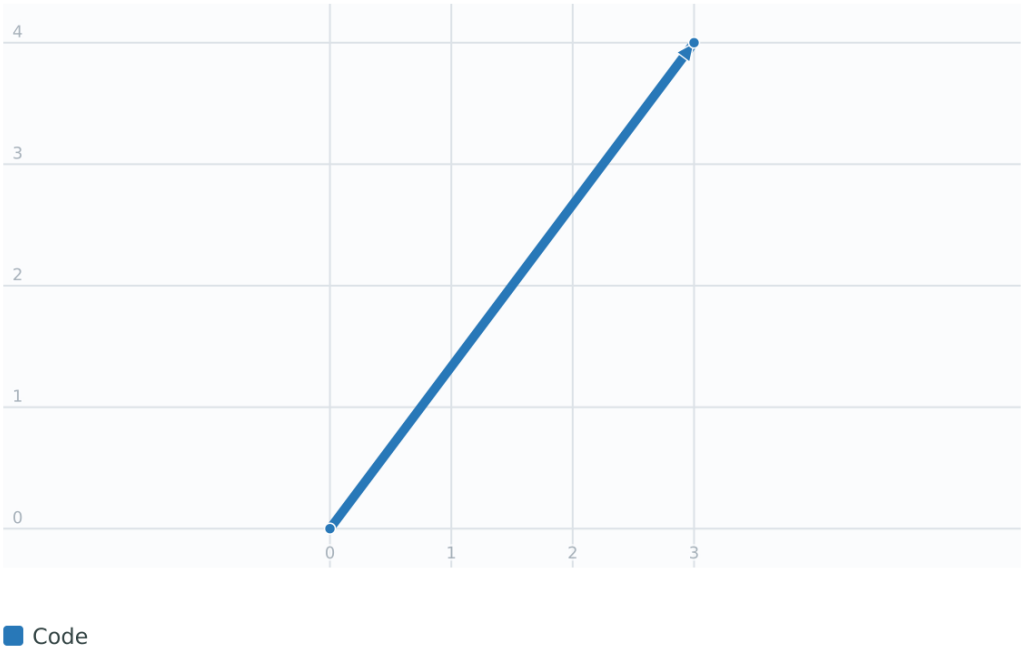
Code

```
SELECT ST_MSize('LINESTRING M (1 2 3,4 6 8)');
```

Output

```
5
```

Figure



相关信息

ST_MMin, ST_MMax, ST_XSize, ST_YSize, ST_ZSize

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

This example returns the point 20 percent along a line.

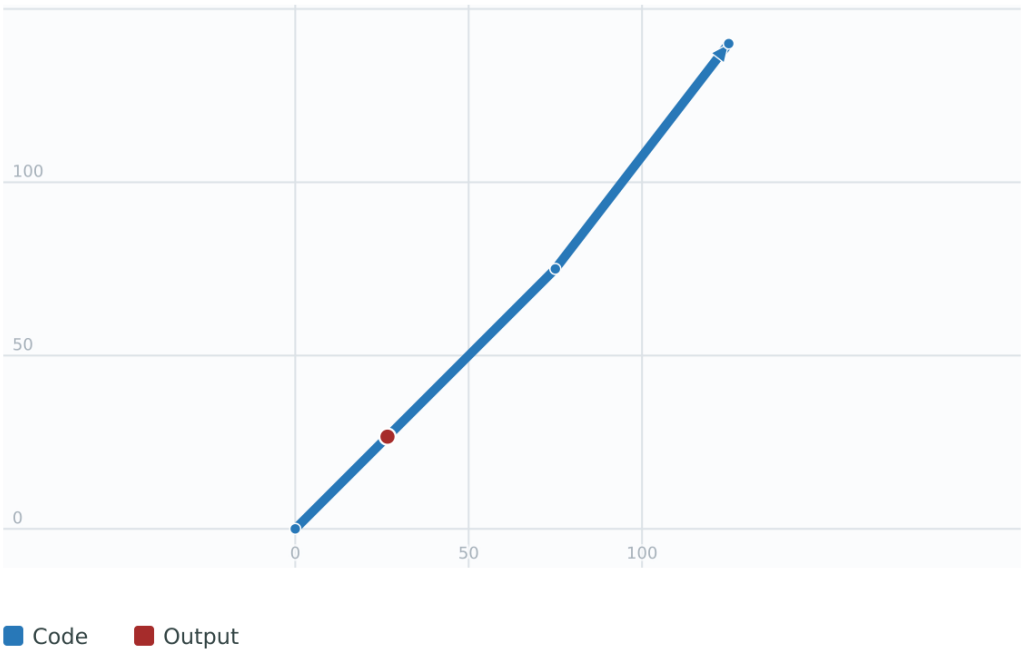
Code

```
SELECT ST_LineInterpolatePoint(
'LINESTRING(25 50,100 125,150 190)',
0.2);
```

Output

```
POINT(51.5974135047432 76.5974135047432)
```

Figure



3D ⊠的中点：

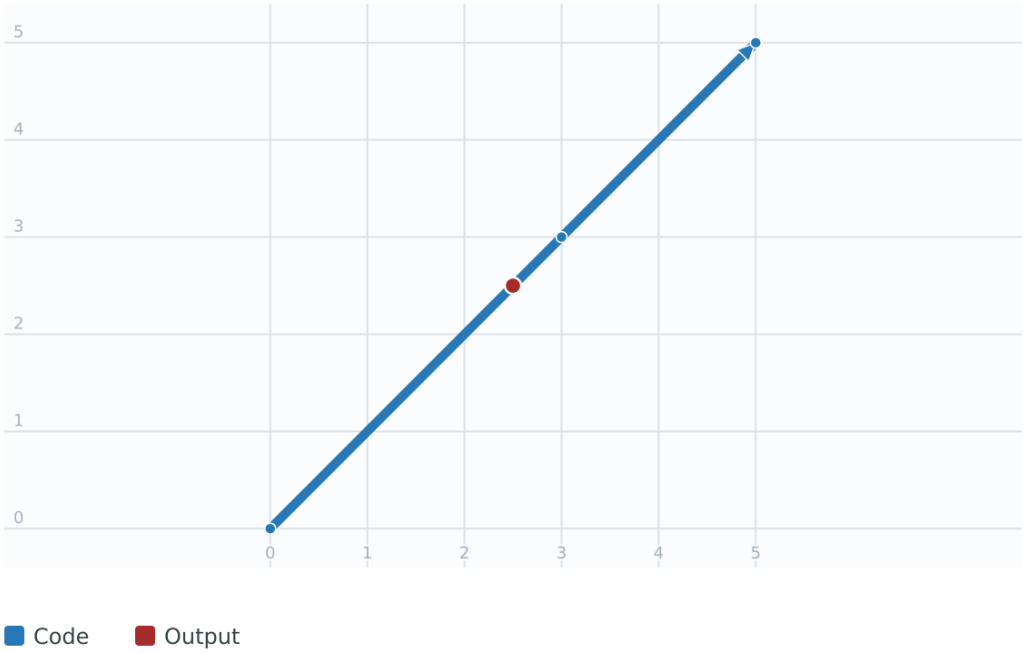
Code

```
SELECT ST_LineInterpolatePoint(
'LINESTRING(1 2 3,4 5 6,6 7 8)',
0.5);
```

Output

```
POINT(3.5 4.5 5.5)
```

Figure



直⊠上距离某点最近的点：

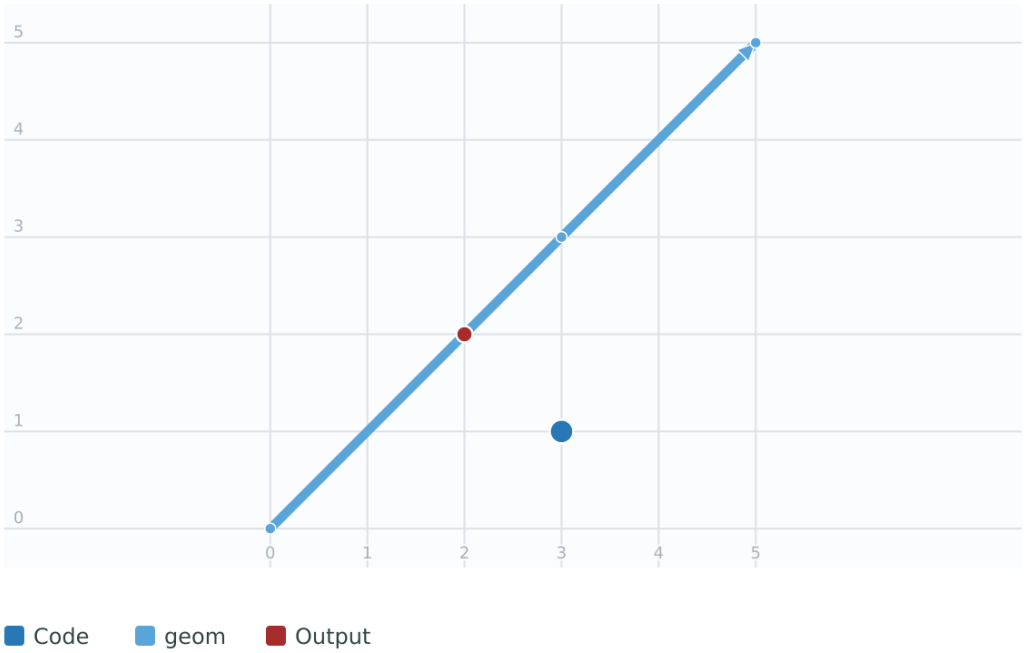
Code

```
SELECT ST_LineInterpolatePoint(
    line.geom,
    ST_LineLocatePoint(line.geom, 'POINT(4 3)'))
FROM (SELECT 'LINESTRING(1 2,4 5,6 7)'::geometry As geom) AS line;
```

Output

```
POINT(3 4)
```

Figure

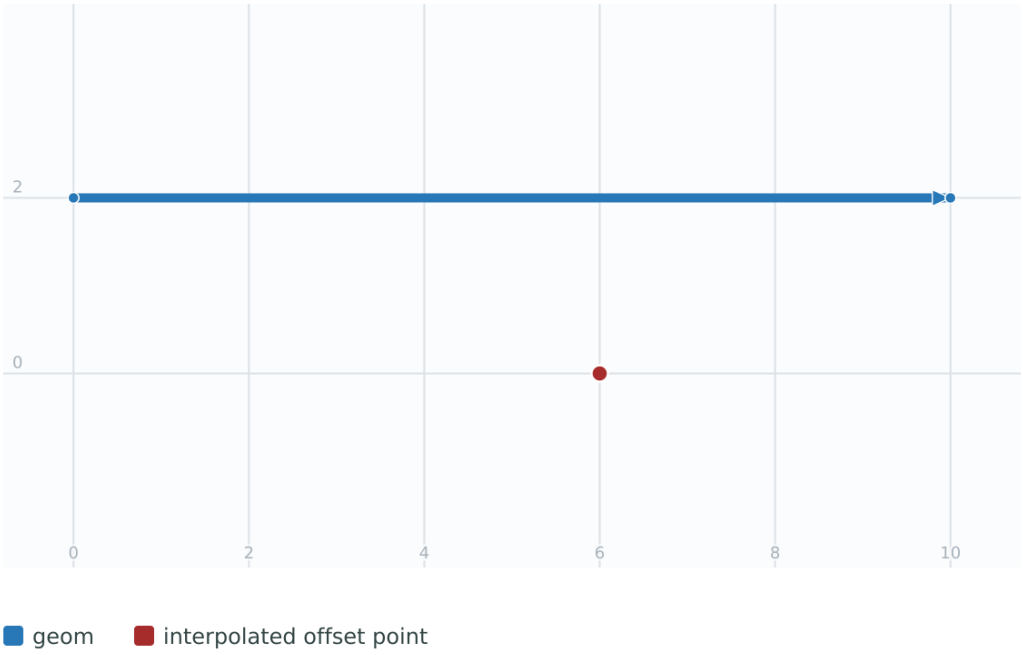


An interpolated point can be offset to the left or right of a line by using ST_OffsetCurve before interpolation. This is useful for locating an address point beside a street centerline.

Code

```
WITH street AS (
    SELECT 'LINESTRING(0 0, 10 0)'::geometry AS geom,
           100 AS from_num,
           200 AS to_num
)
SELECT
    ST_LineInterpolatePoint(
        ST_OffsetCurve(geom, -2),
        (160 - from_num)::float8 / (to_num - from_num)
    ) AS interpolated_offset_point
FROM street;
```

Figure



相关信息

ST_LineInterpolatePoints, ST_3DLineInterpolatePoint, ST_LineLocatePoint, ST_OffsetCurve

7.19.2 ST_3DLineInterpolatePoint

ST_3DLineInterpolatePoint — 返回沿 3D 区的小数指示位置插区的点。

Synopsis

geometry **ST_3DLineInterpolatePoint**(geometry a_linestring, float8 a_fraction);

描述

返回沿 3D 区的小数指示位置插区的点。第一个参数必区是 LINESTRING。第二个参数是 0 到 1 之区的浮点数，表示点位置占区区度的一部分。如果存在 M 区，区区其区行插区。



Note
ST_LineInterpolatePoint 区算 2D 中的点，然后区 Z 和 M 的区区行插区，而此函数区算 3D 中的点，并且区区 M 区区行插区。

可用性：3.0.0



区函数支持 3d 并且不会区失 z-index。

示例

沿 3D 线返回 20% 处的点

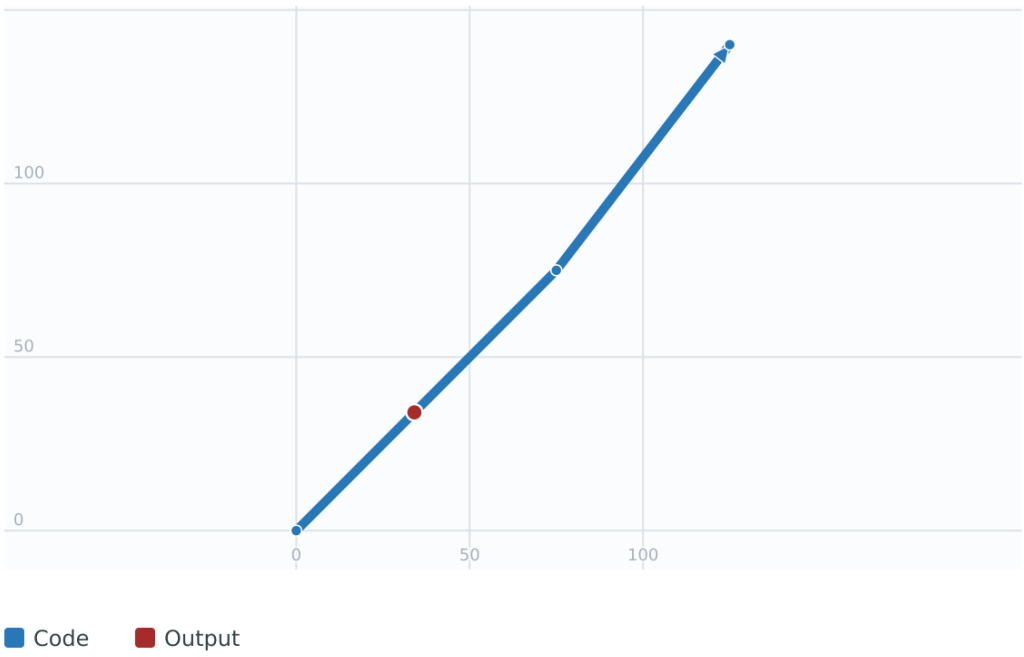
Code

```
SELECT ST_3DLineInterpolatePoint(
'LINESTRING(25 50 70,100 125 90,150 190 200)',
0.20);
```

Output

```
POINT Z (59.06758929108215 84.06758929108216 79.0846904776219)
```

Figure



相关信息

ST_LineInterpolatePoint, ST_LineInterpolatePoints, ST_LineLocatePoint

7.19.3 ST_LineInterpolatePoints

ST_LineInterpolatePoints — 返回沿直线以分数间隔插值的点。

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

This example returns points every 20 percent along a two-dimensional line.

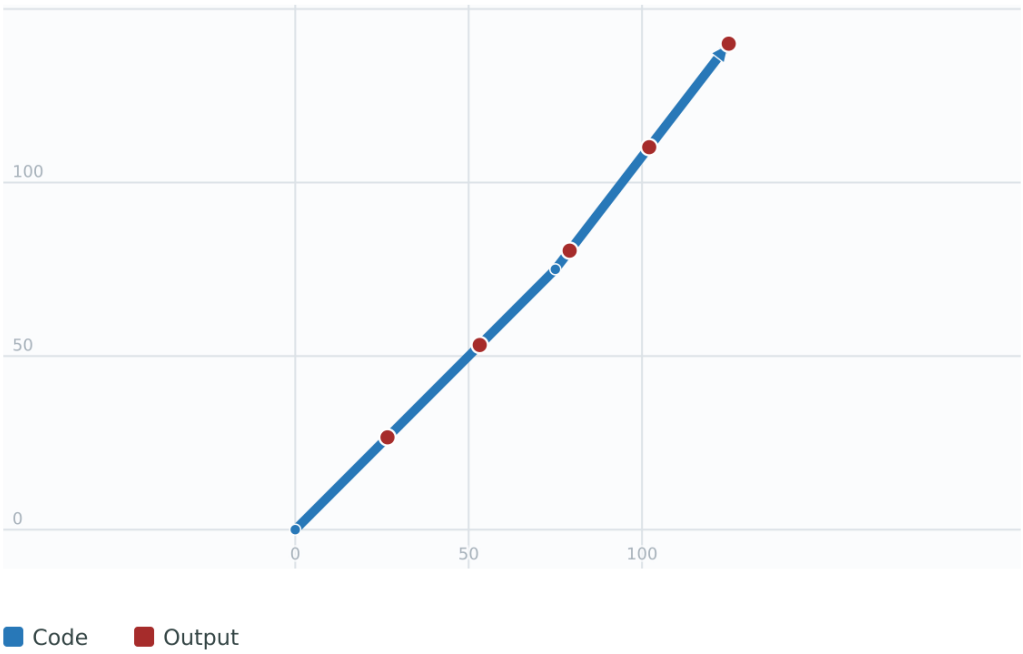
Code

```
SELECT ST_LineInterpolatePoints(
'LINESTRING(25 50,100 125,150 190)',
0.20);
```

Output

```
MULTIPOINT((51.5974135047432 76.5974135047432),(78.1948270094864 103.194827009486),(104.132163186446 130.37181214238) ↩
    ,(127.066081593223 160.18590607119),(150 190))
```

Figure



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。

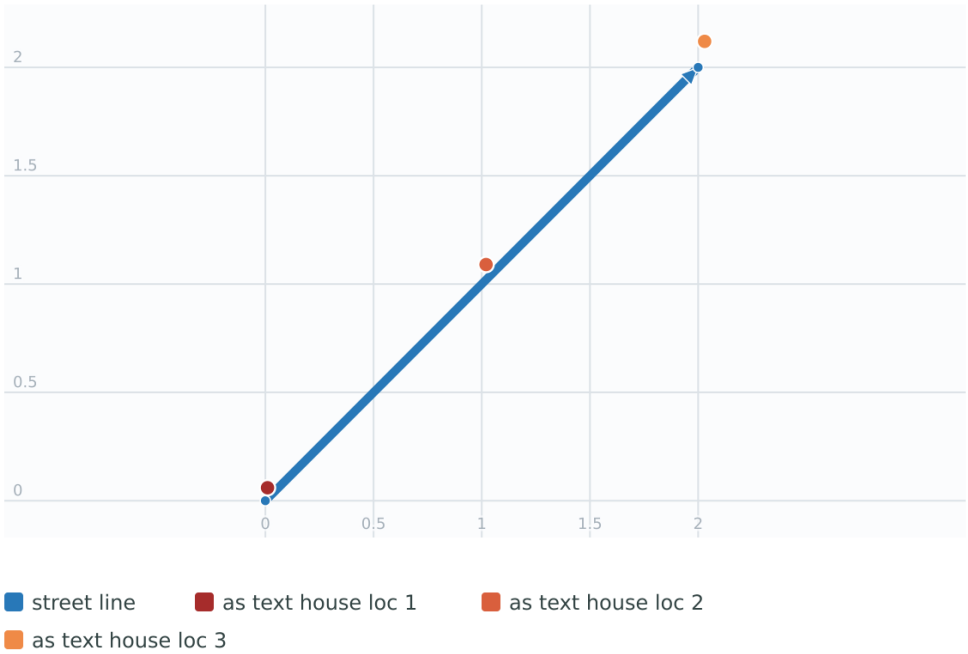
示例

This example approximates the street number of a house by locating the closest point on a street line. The subquery generates sample house and street data, and `ST_DWithin` excludes houses that are too far from the street.

Code

```
SELECT house_loc As as_text_house_loc,
    start_house_num +
        CAST( (end_house_num - start_house_num)
            * ST_LineLocatePoint(street_line, house_loc) As integer) As house_num
FROM
(SELECT 'LINESTRING(1 2,3 4)'::geometry As street_line,
    ST_Point(x*1.01, y*1.03) As house_loc, 10 As start_house_num,
        20 As end_house_num
FROM generate_series(1, 3) x CROSS JOIN generate_series(2, 4) As y)
As foo
WHERE ST_DWithin(street_line, house_loc, 0.2);
```

Figure



This example finds the closest point on a line to a point or other geometry.

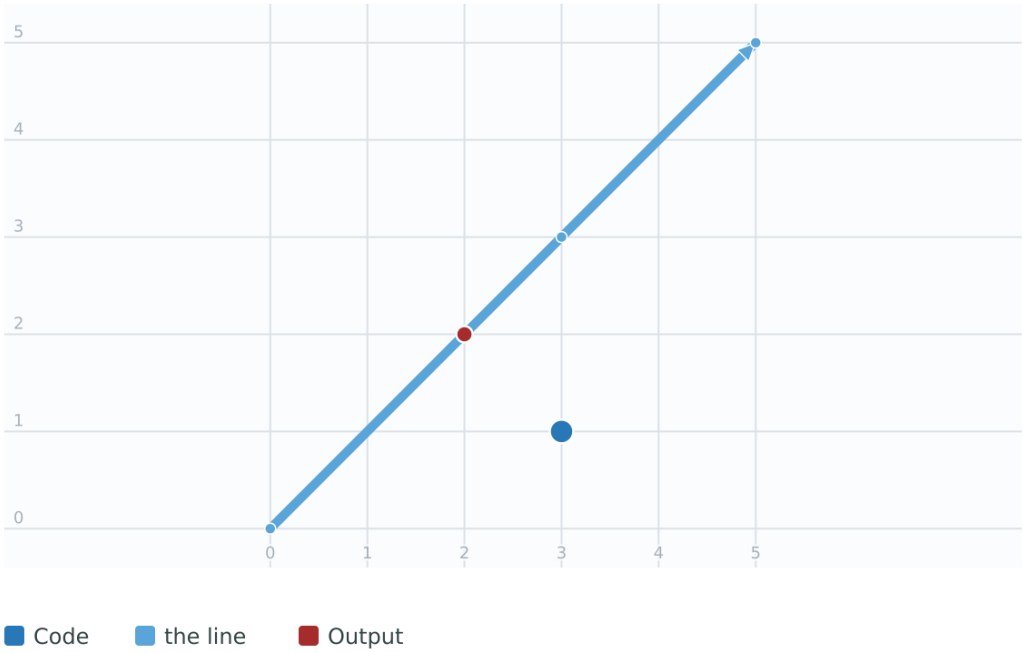
Code

```
SELECT ST_LineInterpolatePoint(
    foo.the_line,
    ST_LineLocatePoint(foo.the_line, 'POINT(4 3)'::geometry))
FROM (SELECT 'LINESTRING(1 2,4 5,6 7)'::geometry As the_line) As foo;
```

Output

```
POINT(3 4)
```

Figure

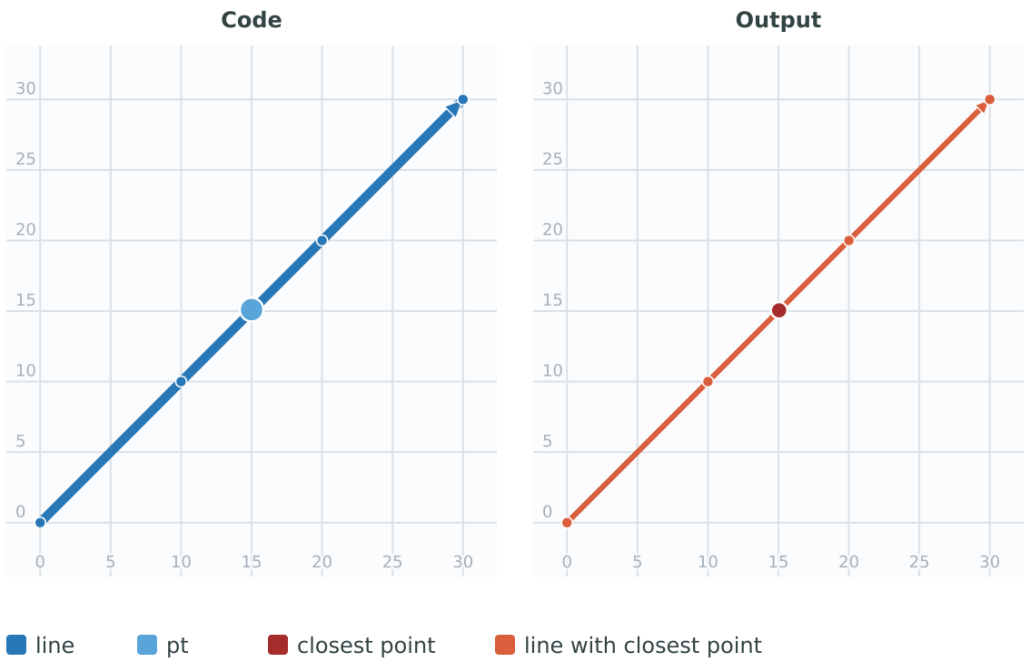


This example locates the closest segment and point on a LineString.

Code

```
WITH data AS (
  SELECT
    'LINESTRING (0 0, 10 10, 20 20, 30 30)'::geometry AS line,
    'POINT(15 15.1)'::geometry AS pt
),
segments AS (
  SELECT
    s.i,
    d.line,
    d.pt,
    ST_MakeLine(ST_PointN(d.line, s.i), ST_PointN(d.line, s.i + 1)) AS segment
  FROM data AS d
  CROSS JOIN LATERAL generate_series(1, ST_NumPoints(d.line) - 1) AS s(i)
)
SELECT
  i AS segment_number,
  round(ST_Distance(segment, pt)::numeric, 3) AS distance,
  ST_ClosestPoint(segment, pt) AS closest_point,
  ST_AddPoint(line, ST_ClosestPoint(segment, pt), i) AS line_with_closest_point
FROM segments
ORDER BY ST_Distance(segment, pt)
LIMIT 1;
```

Figure



相关信息

ST_AddPoint, ST_ClosestPoint, 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

Computes the line which is the section of the input line starting and ending at the given fractional locations. The first argument must be a LINESTRING. The second and third arguments are values in the range [0, 1] representing the start and end locations. For geometry inputs, the fractions are measured in 2D line length. The Z and M values are interpolated for added endpoints if present.

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

Extract the middle third of a LineString, from 0.333 to 0.666.

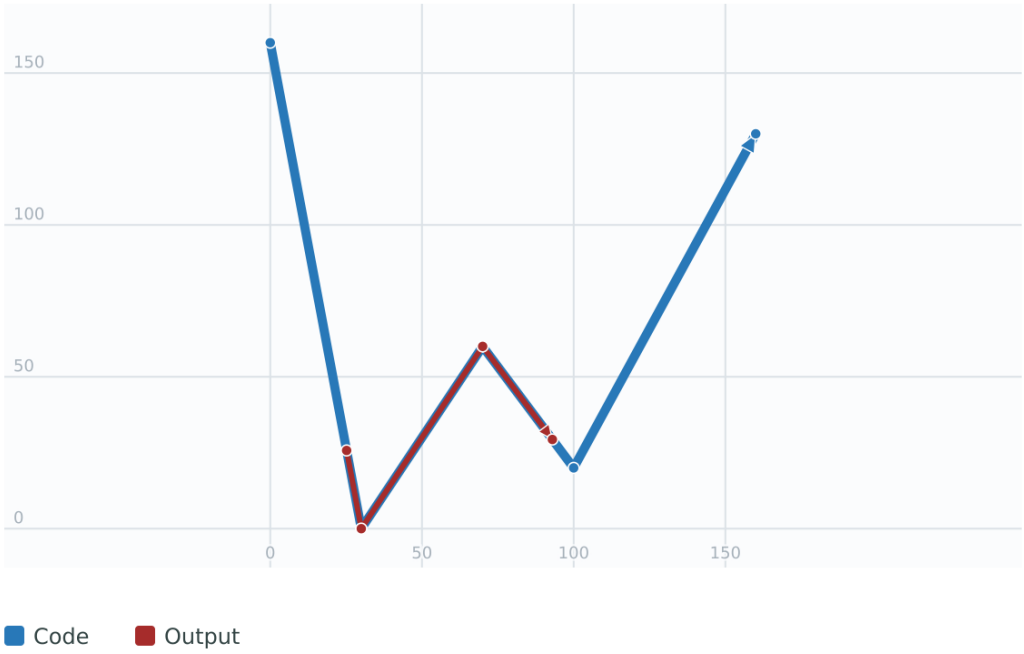
Code

```
SELECT ST_LineSubstring(
'LINESTRING (20 180,50 20,90 80,120 40,180 150)',
0.333,
0.666);
```

Output

```
LINESTRING (45.17311810399485 45.74337011202746, 50 20, 90 80, 112.97593050157862 49.36542599789519)
```

Figure



如果起始位置和☒束位置相同，☒☒果是一个 POINT。

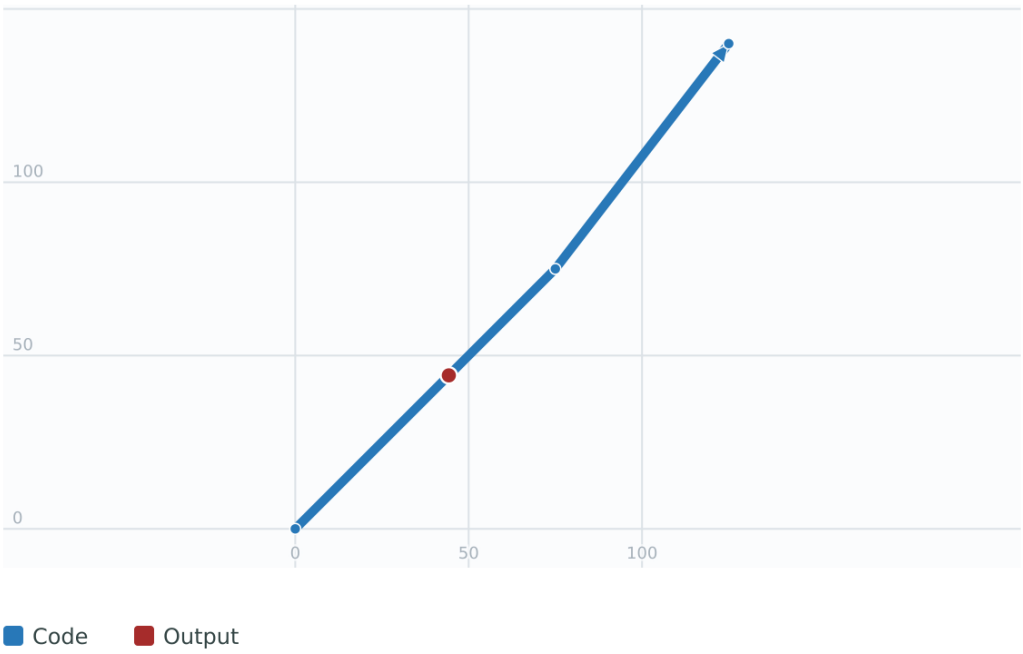
Code

```
SELECT ST_LineSubstring(
'LINESTRING(25 50,100 125,150 190)',
0.333,
0.333);
```

Output

```
POINT(69.28469348539744 94.28469348539744)
```

Figure



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⌧。

The final generated step is filtered out when the line length is an exact multiple of the requested section length.

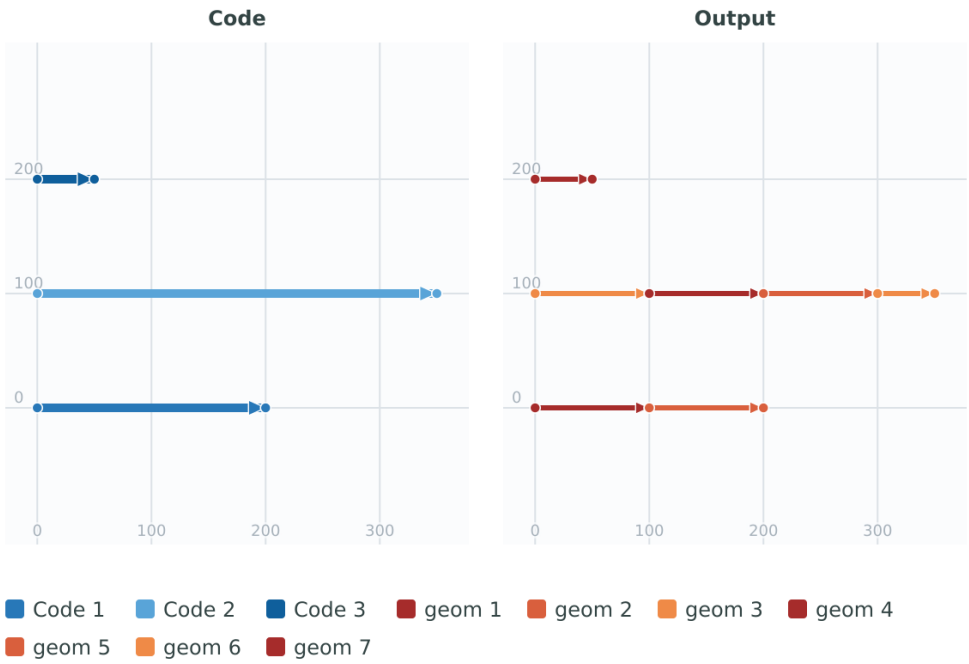
Code

```
WITH data(id, geom) AS (VALUES
        ( 'A', 'LINESTRING( 0 0, 200 0)'::geometry ),
        ( 'B', 'LINESTRING( 0 100, 350 100)'::geometry ),
        ( 'C', 'LINESTRING( 0 200, 50 200)'::geometry )
    )
SELECT id, i,
       ST_LineSubstring(geom, startfrac, LEAST( endfrac, 1 )) AS geom
FROM (
    SELECT id, geom, ST_Length(geom) len, 100 sublen FROM data
    ) AS d
CROSS JOIN LATERAL (
    SELECT i, (sublen * i) / len AS startfrac,
              (sublen * (i+1)) / len AS endfrac
    FROM generate_series(0, floor( len / sublen )::integer ) AS t(i)
    WHERE (sublen * i) / len <
> 1.0
    ) AS d2;
```

Output

```
id | i |           geom
----+---+----------------------------
 A  | 0 | LINESTRING(0 0,100 0)
 A  | 1 | LINESTRING(100 0,200 0)
 B  | 0 | LINESTRING(0 100,100 100)
 B  | 1 | LINESTRING(100 100,200 100)
 B  | 2 | LINESTRING(200 100,300 100)
 B  | 3 | LINESTRING(300 100,350 100)
 C  | 0 | LINESTRING(0 200,50 200)
```

Figure



Split a LineString by points by locating the points along the line, adding the 0 and 1 endpoints, and extracting substrings between adjacent locations.

Code

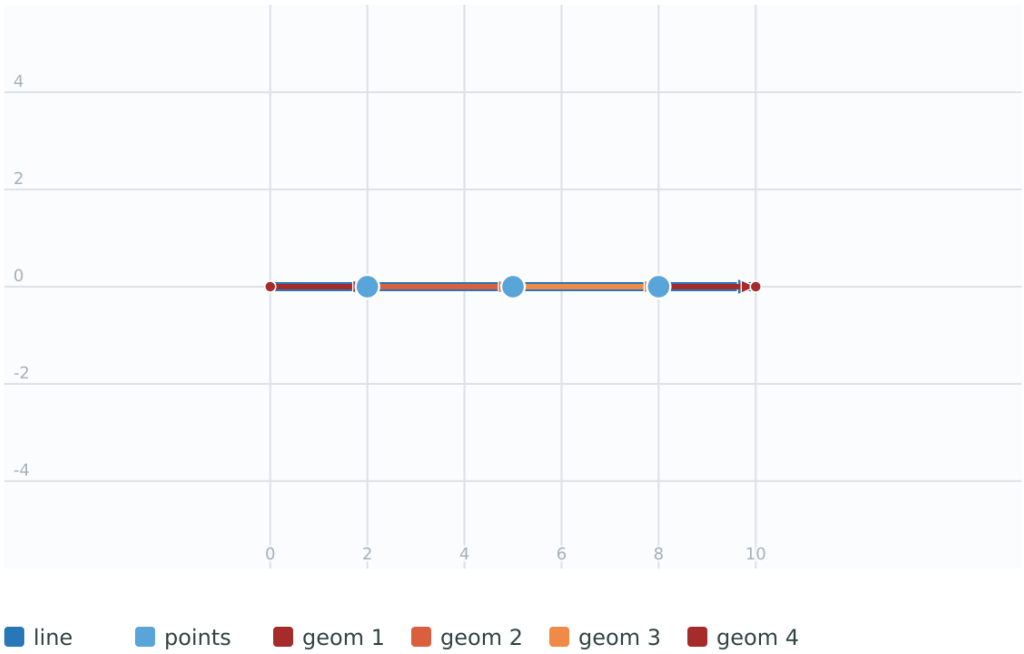
```
WITH data AS (
  SELECT 'LINESTRING(0 0, 10 0)'::geometry AS line,
         'MULTIPOINT((2 0),(5 0),(8 0))'::geometry AS points
),
fractions AS (
  SELECT 0.0 AS fraction
  UNION ALL
  SELECT 1.0
  UNION ALL
  SELECT ST_LineLocatePoint(line, (dp).geom)
  FROM data
  CROSS JOIN LATERAL ST_Dump(points) AS dp
),
ordered AS (
  SELECT DISTINCT fraction
  FROM fractions
),
segments AS (
  SELECT fraction AS start_fraction,
         lead(fraction) OVER (ORDER BY fraction) AS end_fraction
  FROM ordered
)
```

```
SELECT row_number() OVER (ORDER BY start_fraction) AS segment_id,
       ST_LineSubstring(line, start_fraction, end_fraction) AS geom
FROM data
CROSS JOIN segments
WHERE end_fraction IS NOT NULL
  AND start_fraction <
> end_fraction
ORDER BY start_fraction;
```

Output

```
segment_id |         geom
-----------+----------------------
         1 | LINESTRING(0 0,2 0)
         2 | LINESTRING(2 0,5 0)
         3 | LINESTRING(5 0,8 0)
         4 | LINESTRING(8 0,10 0)
```

Figure



地理⊠⊠沿球体表面⊠行⊠量，而几何⊠⊠沿⊠⊠行⊠量

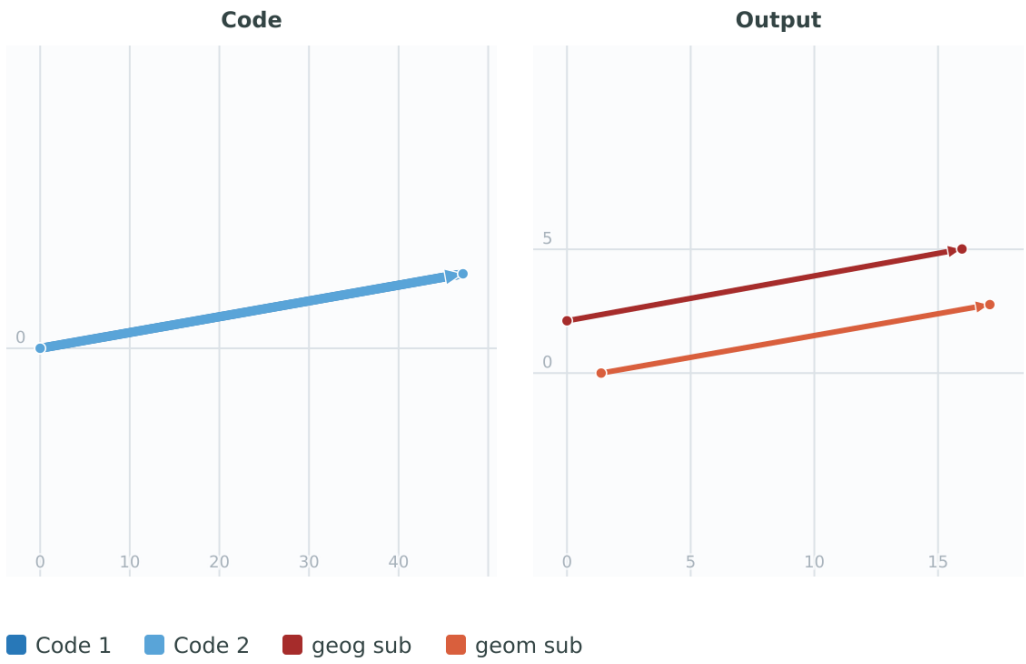
Code

```
SELECT ST_LineSubstring(
'LINESTRING(-118.2436 34.0522,-71.0570 42.3611)'::geography,
0.333,
0.666) AS geog_sub,
ST_LineSubstring(
    'LINESTRING(-118.2436 34.0522,-71.0570 42.3611)'::geometry,
    0.333,
    0.666) AS geom_sub;
```

Output

```
-[ RECORD 1 ]----------
geog_sub | LINESTRING(-103.911641 38.931128,-87.941787 41.831072)
geom_sub | LINESTRING(-102.530462 36.819064,-86.817324 39.585927)
```

Figure



For geometry inputs, the fractional locations are based on the 2D length of the line, even when the input has Z values. The resulting Z values are interpolated along the selected 2D location.

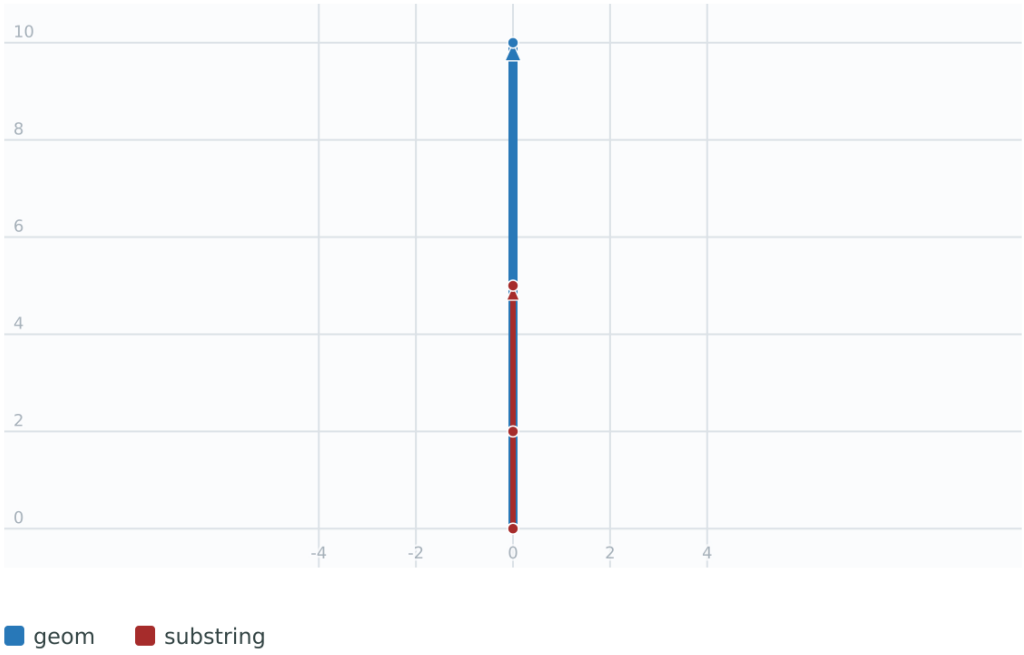
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (0 0 0,0 2 5,0 10 10)'::geometry AS geom
)
SELECT ST_Length(geom) AS length_2d,
       ST_3DLength(geom) AS length_3d,
       ST_LineSubstring(geom, 0, 0.5) AS substring
FROM data;
```

Output

```
length_2d |    length_3d     |              substring
----------+------------------+-------------------------------------
        10 | 14.819145939191106 | LINESTRING Z (0 0 0,0 2 5,0 5 6.875)
```

Figure



相关信息

ST_Length, ST_LineExtend, ST_LineInterpolatePoint, ST_LineLocatePoint, ST_LineMerge, ST_Split

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

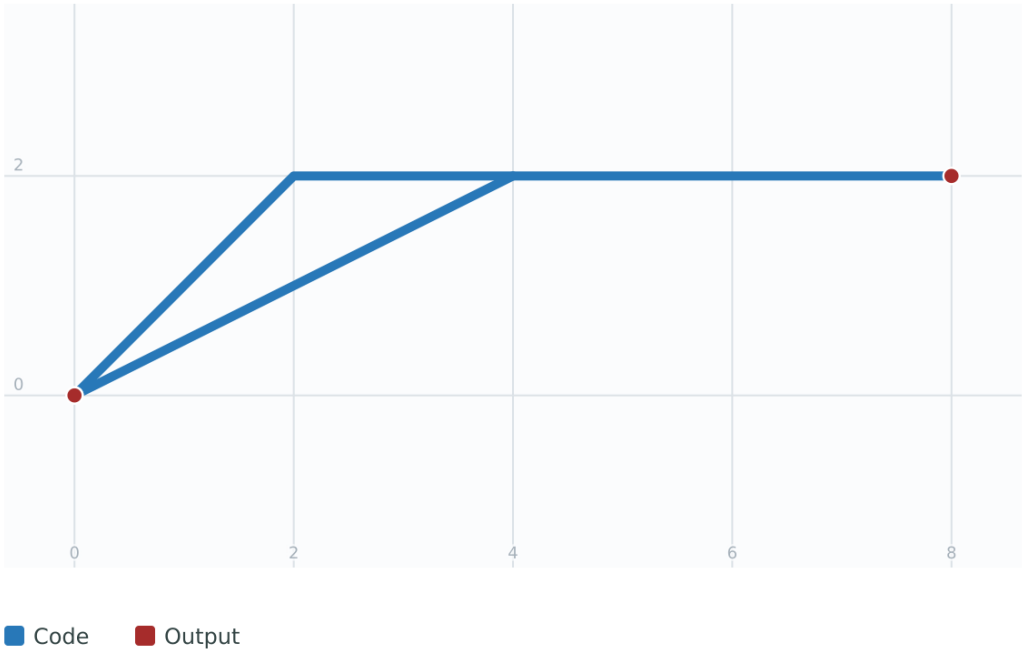
Code

```
SELECT ST_LocateAlong(
'MULTILINESTRINGM((1 2 3, 3 4 2, 9 4 3),(1 2 3, 5 4 5))'::geometry,
3);
```

Output

```
MULTIPOINT M ((1 2 3),(9 4 3),(1 2 3))
```

Figure



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

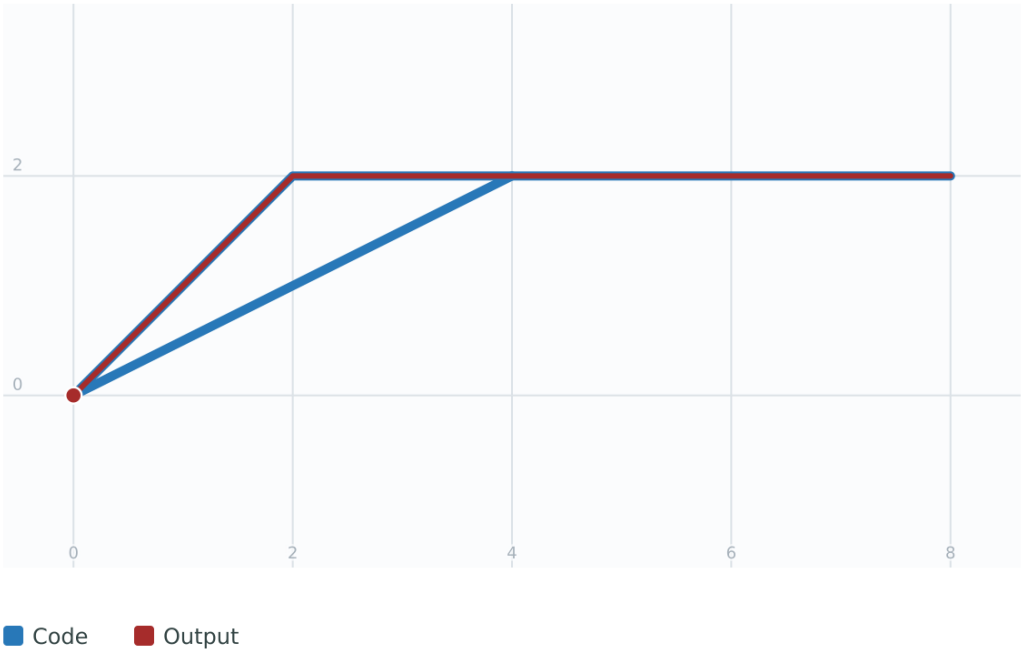
Code

```
SELECT ST_LocateBetween(
'MULTILINESTRING M ((1 2 3,3 4 2,9 4 3),(1 2 3,5 4 5))'::geometry,
1.5,
3);
```

Output

```
GEOMETRYCOLLECTION M (LINESTRING M (1 2 3,3 4 2,9 4 3),POINT M (1 2 3))
```

Figure



Extract the section between measures 2 and 8, offset to the left. Because the clipped line and the offset result share the same coordinate space, this example is intended for a single overlay panel rather than separate side-by-side figures.

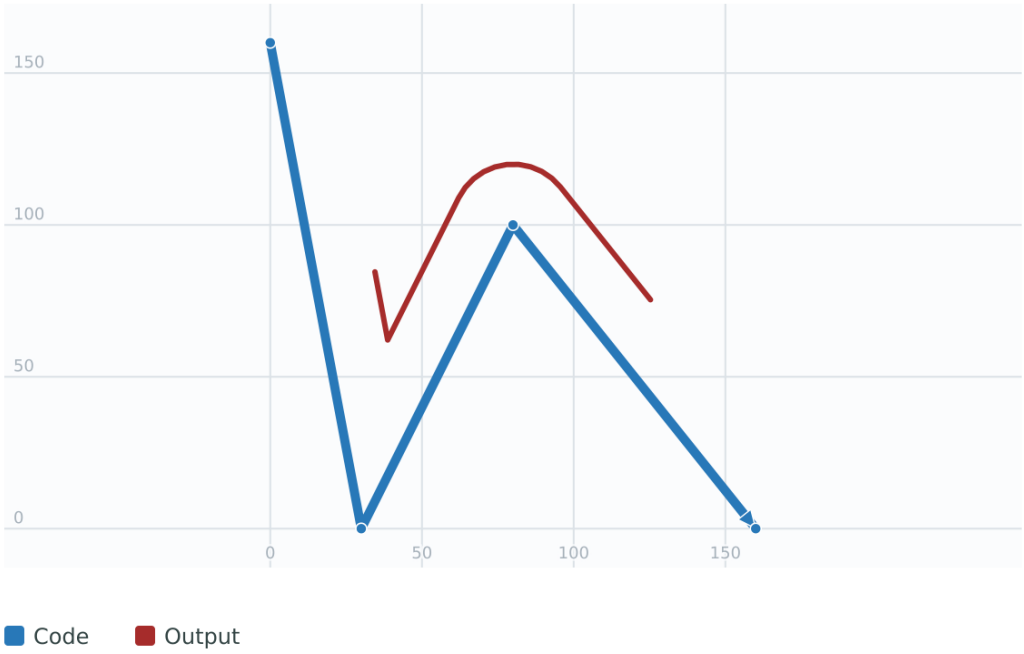
Code

```
SELECT ST_LocateBetween(
ST_AddMeasure(
    'LINESTRING (20 180,50 20,100 120,180 20)',
    0,
    10),
2,
8,
20);
```

Output

```
MULTILINESTRING((54.49835 104.53427,58.700561 82.122481,82.111456 128.944272,84.248937 132.324934,87.01637 ↩
    135.21267,90.303073 137.491987,93.977598 139.071724,97.892984 139.888702,101.892639 139.910247,105.816599 ↩
    139.135495,109.507928 137.595434,112.818995 135.351657,115.617376 132.493901,145.310173 95.377905))
```

Figure



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

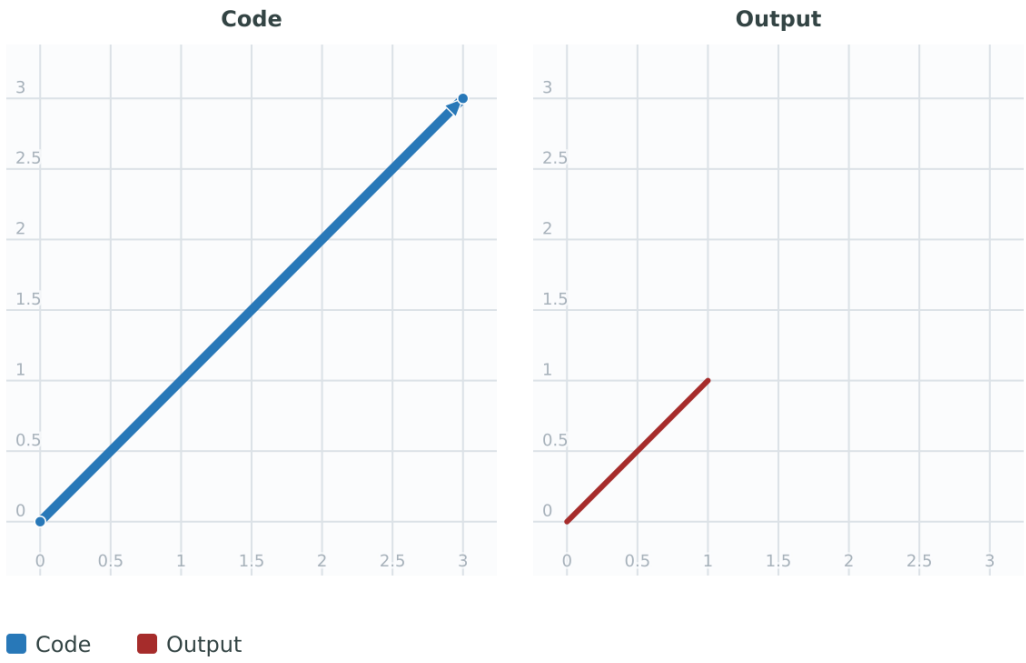
Code

```
SELECT ST_LocateBetweenElevations(
'LINESTRING(1 2 3,4 5 6)'::geometry,
2,
4);
```

Output

```
MULTILINESTRING Z ((1 2 3,2 3 4))
```

Figure



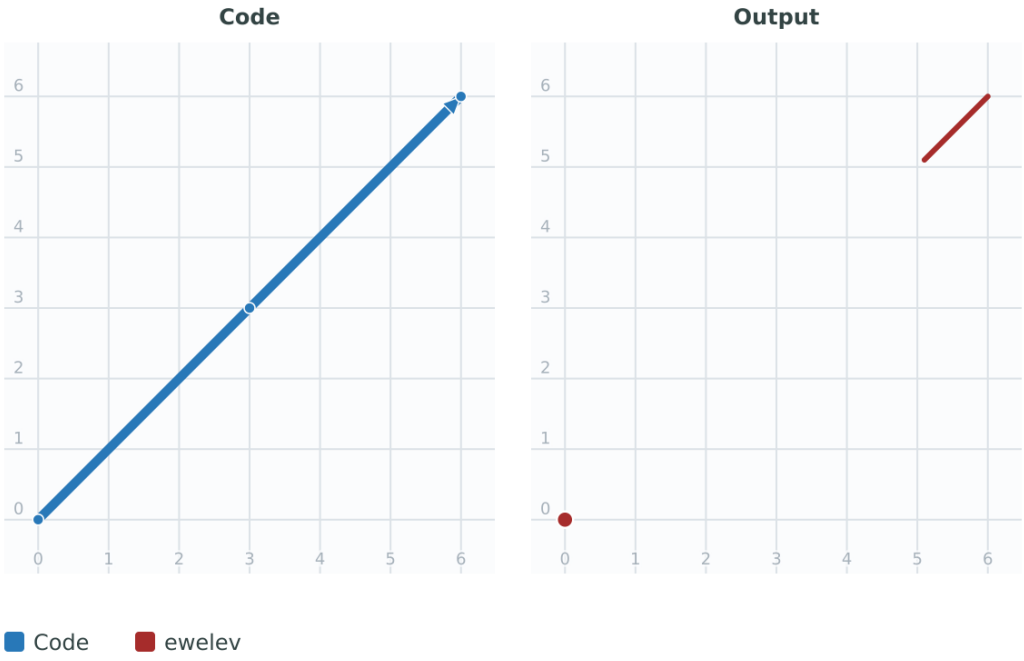
Code

```
SELECT ST_LocateBetweenElevations(
'LINESTRING(1 2 6,4 5 -1,7 8 9)',
6,
9) As ewelev;
```

Output

```
GEOMETRYCOLLECTION Z (POINT Z (1 2 6),LINESTRING Z (6.1 7.1 6,7 8 9))
```

Figure



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

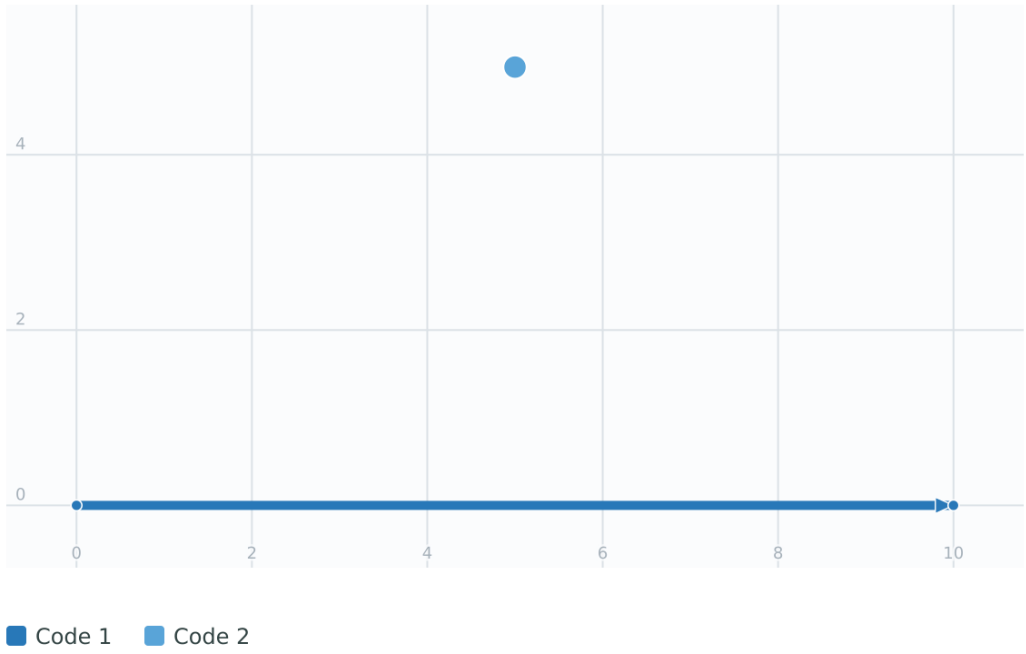
Code

```
SELECT ST_InterpolatePoint(
'LINESTRING M (0 0 0,10 0 20)',
'POINT(5 5)');
```

Output

```
10
```

Figure



相关信息

ST_AddMeasure, ST_LocateAlong, ST_LocateBetween

7.19.10 ST_3DInterpolatePoint

ST_3DInterpolatePoint — Returns the interpolated measure of a geometry closest to a point in 3D.

Synopsis

float8 **ST_3DInterpolatePoint**(geometry linear_geom_with_measure, geometry point);

描述

Returns the interpolated measure value of a linear ZM geometry at the location closest to the given point, using 3D (XYZ) distance for the projection. Use this function when the geometry has significant Z variation, such as flight trajectories, where ST_InterpolatePoint would give incorrect results by ignoring the Z dimension.

Note
The line must have both Z and M dimensions. The point should have a Z dimension.

Availability: 3.7.0

☒函数支持 3d 并且不会☒失 z-index。

示例

This example interpolates on a line rising diagonally in Z, using a point at the 3D midpoint.

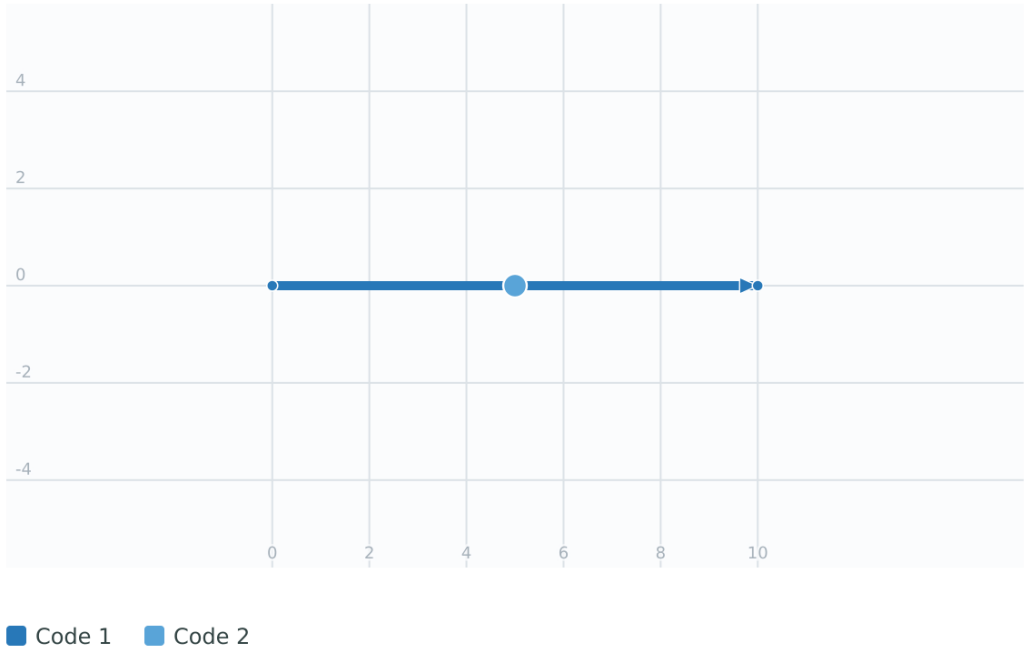
Code

```
SELECT ST_3DInterpolatePoint(
'LINESTRING ZM (0 0 0 0,10 0 10 20)',
'POINT Z (5 0 5)');
```

Output

```
10
```

Figure



This example uses a vertical line with zero XY extent, where 3D projection works but 2D projection does not.

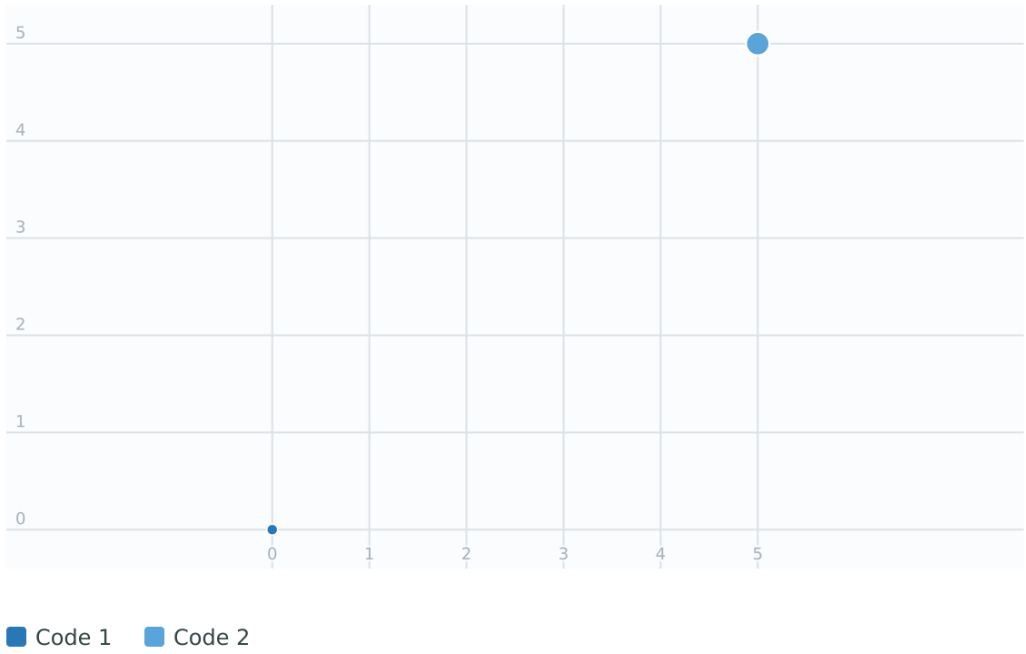
Code

```
SELECT ST_3DInterpolatePoint(
'LINESTRING ZM (0 0 0 0,0 0 10 100)',
'POINT Z (5 5 5)');
```

Output

```
50
```

Figure



相关信息

ST_InterpolatePoint, ST_AddMeasure, ST_LocateAlong, ST_LocateBetween

7.19.11 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

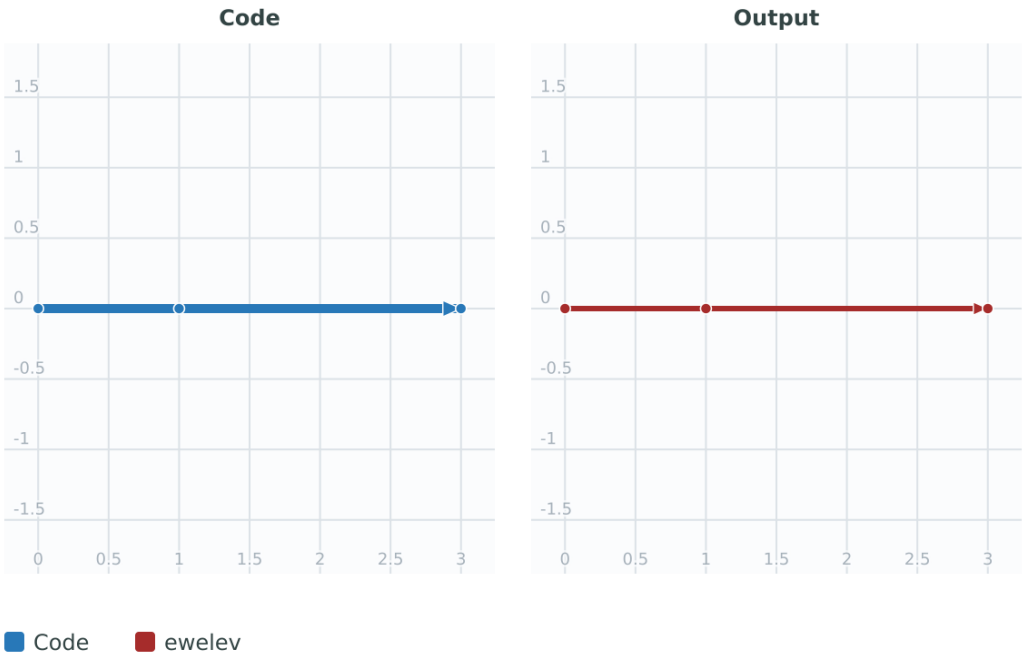
Code

```
SELECT ST_AddMeasure(
'LINESTRING(1 0,2 0,4 0)',
1,
4) As ewelev;
```

Output

```
LINESTRING M (1 0 1,2 0 2,4 0 4)
```

Figure



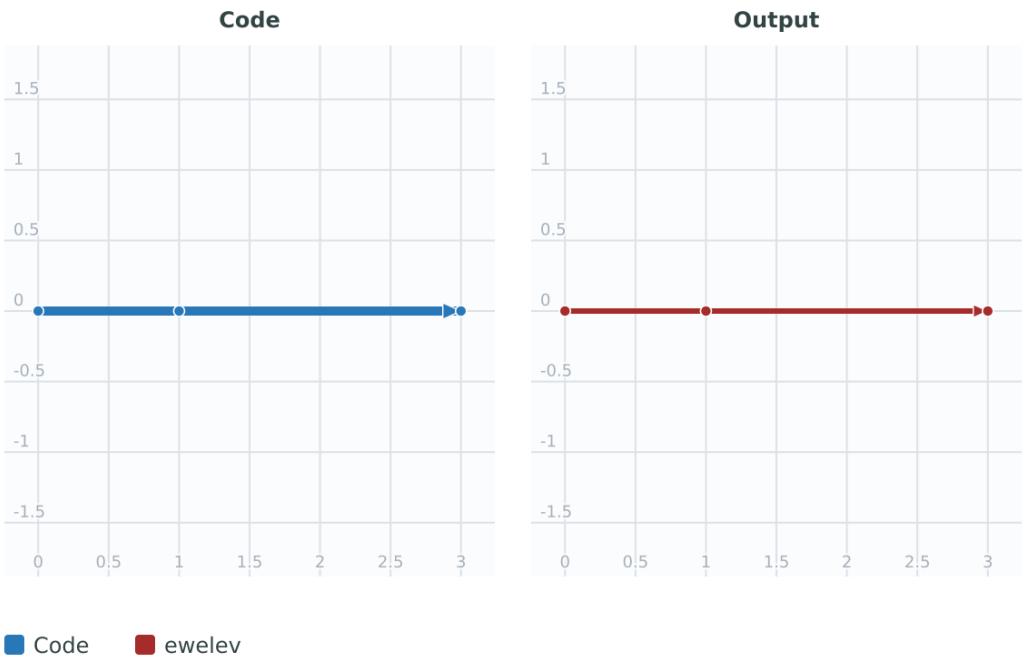
Code

```
SELECT ST_AddMeasure(
'LINESTRING(1 0 4,2 0 4,4 0 4)',
10,
40) As ewelev;
```

Output

```
LINESTRING ZM (1 0 4 10,2 0 4 20,4 0 4 40)
```

Figure



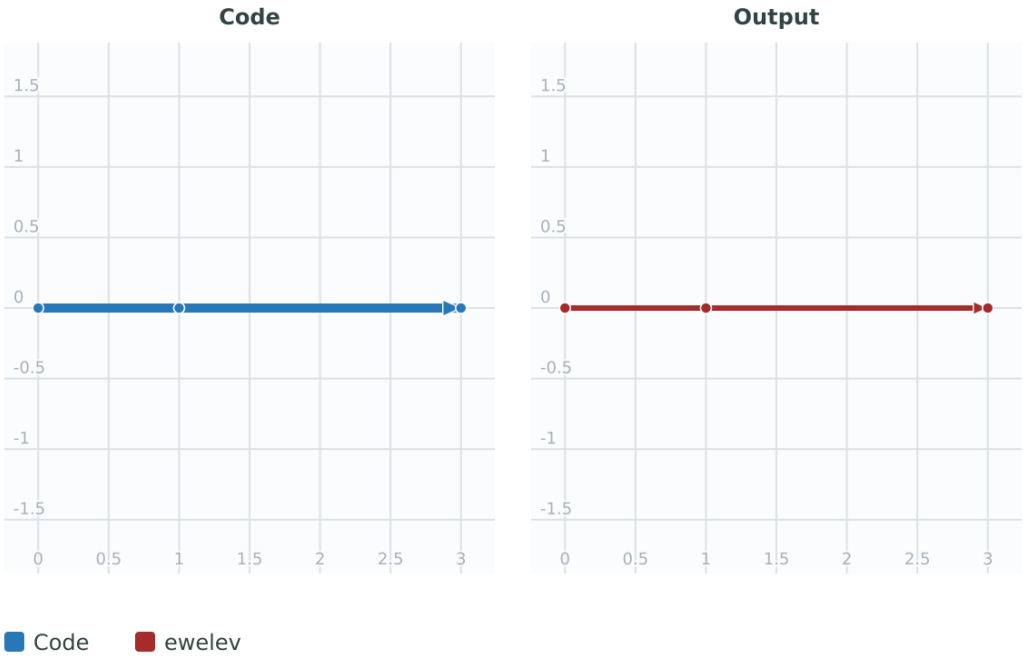
Code

```
SELECT ST_AddMeasure(
'LINESTRINGM(1 0 4, 2 0 4, 4 0 4)',
10,
40) As ewelev;
```

Output

```
LINESTRING M (1 0 10,2 0 20,4 0 40)
```

Figure



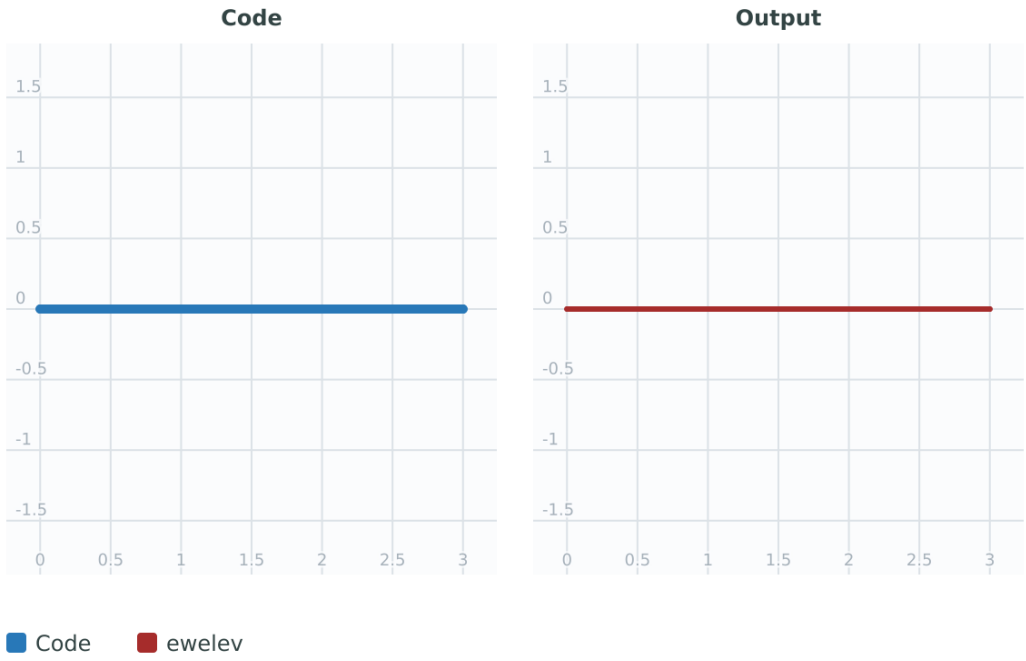
Code

```
SELECT ST_AddMeasure(
'MULTILINESTRINGM((1 0 4, 2 0 4, 4 0 4),(1 0 4, 2 0 4, 4 0 4))',
10,
70) As ewelev;
```

Output

```
MULTILINESTRING M ((1 0 10,2 0 20,4 0 40),(1 0 40,2 0 50,4 0 70))
```

Figure



7.20 ☒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

This example constructs a valid trajectory.

Code

```
SELECT ST_IsValidTrajectory(ST_MakeLine(ST_MakePointM(0, 0, 1),
ST_MakePointM(0, 1, 2)));
```

Output

```
t
```

This example constructs an invalid trajectory.

Code

```
SELECT ST_IsValidTrajectory(ST_MakeLine(ST_MakePointM(0, 0, 1), ST_MakePointM(0, 1, 0)));
```

Output

```
NOTICE:  Measure of vertex 1 (0) not bigger than measure of vertex 0 (1)
f
```

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.

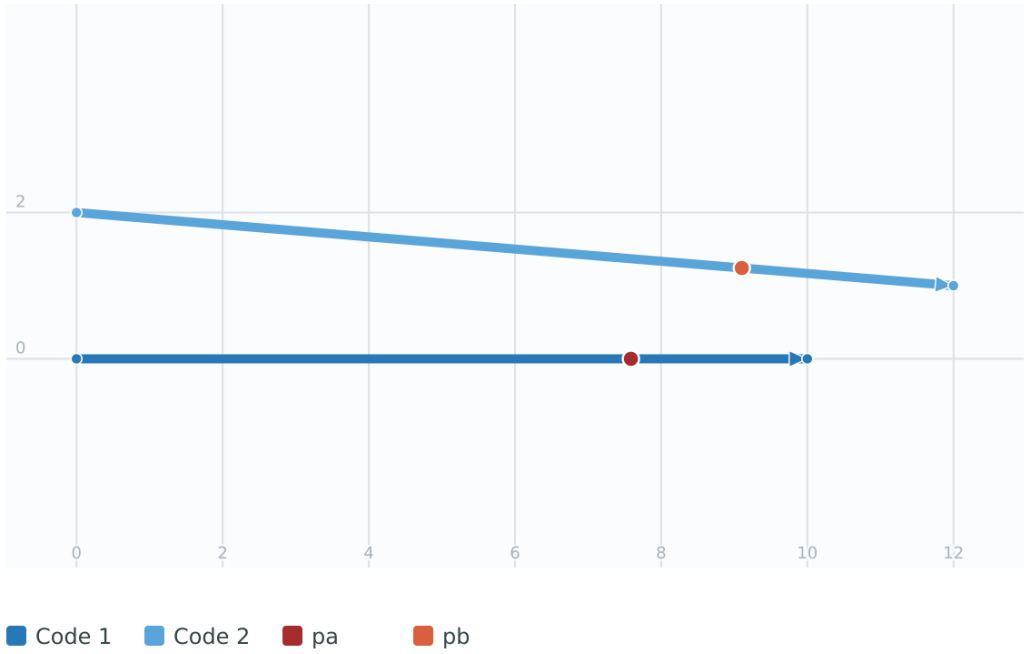
Code

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0,10 0 5)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10,12 1 2)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) b
), cpa AS (
  SELECT ST_ClosestPointOfApproach(a, b) m FROM inp
), points AS (
  SELECT ST_GeometryN(ST_LocateAlong(a, m), 1) pa,
         ST_GeometryN(ST_LocateAlong(b, m), 1) pb
  FROM inp, cpa
)
SELECT to_timestamp(m) AT TIME ZONE 'UTC' t,
       ST_3DDistance(pa, pb) distance,
       pa AS pa, pb AS pb
FROM points, cpa;
```

Output

```
-[ RECORD 1 ]---------------------------
t        | 2015-05-26 10:45:31.034483
distance | 1.9652147377620688
pa       | POINT ZM (7.59 0 3.79 1432637131.03)
pb       | POINT ZM (9.1 1.24 3.93 1432637131.03)
```

Figure



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.

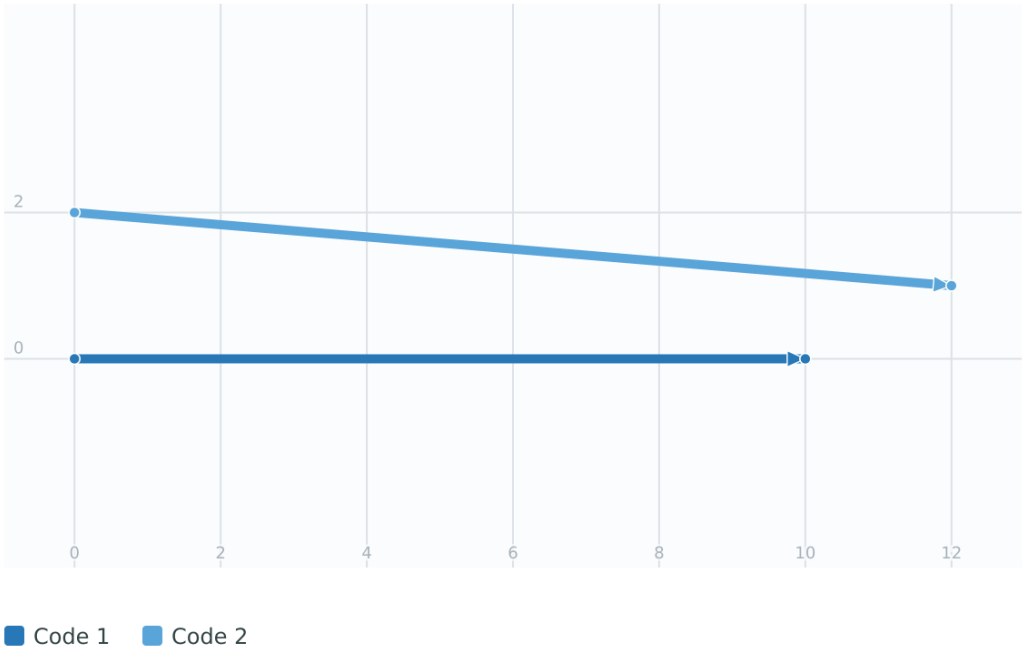
Code

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0,10 0 5)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10,12 1 2)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) b
)
SELECT ST_DistanceCPA(a, b) distance FROM inp;
```

Output

```
1.9652147377620688
```

Figure



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

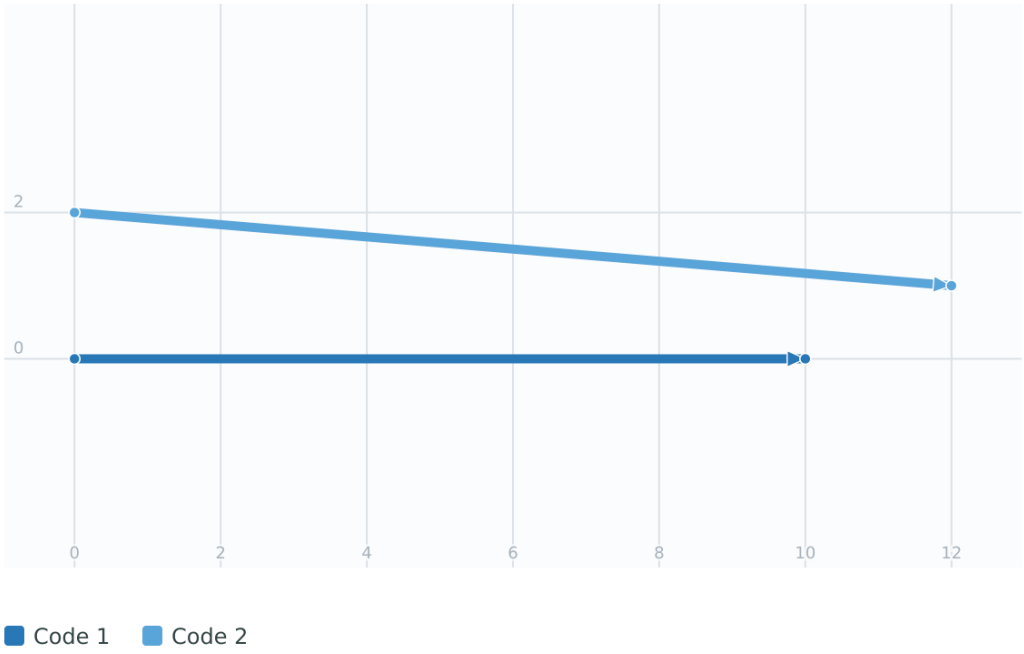
Code

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0,10 0 5)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10,12 1 2)'::geometry,
    extract(epoch from timestamp '2015-05-26 10:00' AT TIME ZONE 'UTC'),
    extract(epoch from timestamp '2015-05-26 11:00' AT TIME ZONE 'UTC')
  ) b
)
SELECT ST_CPAWithin(a, b, 2), ST_DistanceCPA(a, b) distance FROM inp;
```

Output

```
st_cpawithin |     distance
-------------+------------------
 t           | 1.9652147377620688
```

Figure



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 扩展（例如 postgis_raster、postgis_topology、postgis_sfcgal）打包并升级到指定版本或最新版本。

Synopsis

text **PostGIS_Extensions_Upgrade**(text target_version=null);

描述

Packages and upgrades PostGIS extensions to given or latest version. Only extensions you have installed in the database will be packaged and upgraded if needed. Reports full PostGIS version and build configuration infos after. This is short-hand for doing multiple CREATE EXTENSION .. FROM unpackaged and ALTER EXTENSION .. UPDATE for each PostGIS extension. Currently only tries to upgrade extensions postgis, postgis_raster, postgis_sfcgal, and postgis_topology.

可用性：2.5.0

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

Code

```
SELECT PostGIS_Extensions_Upgrade();
```

Output

```
NOTICE:  Packaging extension postgis
NOTICE:  Packaging extension postgis_raster
NOTICE:  Packaging extension postgis_sfcgal
NOTICE:  Extension postgis_topology is not available or not packagable for some reason
```

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

Code

```
SELECT PostGIS_Full_Version();
```

Output

```
POSTGIS="3.7.0dev 3.7.0alpha1-477-ga9004e778" [EXTENSION] PGSQL="180" GEOS="3.15.0beta1" SFCGAL="2.3.0" PROJ="9.8.1 ↩
    NETWORK_ENABLED=OFF URL_ENDPOINT=https://cdn.proj.org DATABASE_PATH=/usr/share/proj/proj.db" GDAL="GDAL 3.13.1, ↩
    released 2026/06/01" LIBXML="2.15.3"
```

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 库的版本号。

Synopsis

text **PostGIS_GEOS_Version**();

描述

返回 GEOS 库的版本号，如果未启用 GEOS 支持，则返回 NULL。

示例

Code

```
SELECT PostGIS_GEOS_Version();
```

Output

```
3.12.0dev-CAPI-1.18.0
```

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 — 返回创建 PostGIS 所依据的 GEOS 库的版本号。

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

Code

```
SELECT PostGIS_GEOS_Compiled_Version();
```

Output

```
3.12.0
```

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

Code

```
SELECT PostGIS_Liblwgeom_Version();
```

Output

```
3.4.0dev 3.3.0rc2-993-g61bdf43a7
```

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

Code

```
SELECT PostGIS_LibXML_Version();
```

Output

```
2.9.10
```

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 — 返回 libjson-c ☒的版本号。

Synopsis

text **PostGIS_LibJSON_Version**();

描述

返回 LibJSON-C ☒的版本号。

示例

Code

```
SELECT PostGIS_LibJSON_Version();
```

Output

```
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 的建日期。

示例

Code

```
SELECT PostGIS_Lib_Build_Date();
```

Output

```
2023-06-22 03:56:11
```

7.21.9 PostGIS_Lib_Version

PostGIS_Lib_Version — 返回 PostGIS 的版本号。

Synopsis

text **PostGIS_Lib_Version**();

描述

返回 PostGIS 的版本号。

示例

Code

```
SELECT PostGIS_Lib_Version();
```

Output

```
3.4.0dev
```

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 ☒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

Code

```
SELECT PostGIS_PROJ_Version();
```

Output

```
7.2.1 NETWORK_ENABLED=OFF URL_ENDPOINT=https://cdn.proj.org USER_WRITABLE_DIRECTORY=/tmp/ ↩
    proj DATABASE_PATH=/usr/share/proj/proj.db
```

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 — 返回用于☒建 PostGIS 的 PROJ ☒的版本号。

Synopsis

text **PostGIS_PROJ_Compiled_Version**();

描述

返回的是用于⌧建 PostGIS 的 PROJ ⌧的版本号。
可用性：3.5.0

示例

Code

```
SELECT PostGIS_PROJ_Compiled_Version();
```

Output

```
9.1.1
```

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

Code

```
SELECT PostGIS_Wagyu_Version();
```

Output

```
0.5.0 (Internal)
```

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

Code

```
SELECT PostGIS_Scripts_Build_Date();
```

Output

```
2023-06-22 03:56:11
```

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

Code

```
SELECT PostGIS_Scripts_Installed();
```

Output

```
3.4.0dev 3.3.0rc2-993-g61bdf43a7
```

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

Code

```
SELECT PostGIS_Scripts_Released();
```

Output

```
3.4.0dev 3.3.0rc2-993-g61bdf43a7
```

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

Code

```
SELECT PostGIS_Version();
```

Output

```
3.4 USE_GEOS=1 USE_PROJ=1 USE_STATS=1
```

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

Code

```
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

Code

```
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

Set and reset `postgis.gdal_enabled_drivers` for the current session.

Code

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.gdal_enabled_drivers = default;
```

Set specific drivers for all new connections to a database.

Code

```
ALTER DATABASE mygisdb SET postgis.gdal_enabled_drivers TO 'GTiff PNG JPEG';
```

Set the default for the whole database cluster to enable all drivers. This requires superuser access. Also note that database, session, and user settings override this.

This example writes the setting to postgres.auto.conf.

Code

```
ALTER SYSTEM SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Reload the PostgreSQL configuration so the changed setting is applied.

Code

```
SELECT pg_reload_conf();
```

相关信息

ST_FromGDALRaster, ST_AsGDALRaster, ST_AsTIFF, ST_AsPNG, ST_AsJPEG, postgis.enable_outdb_raster

7.22.3 postgis.enable_outdb_rasters

postgis.enable_outdb_rasters — 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

Set and reset `postgis.enable_outdb_rasters` for the current session.

Code

```
SET postgis.enable_outdb_rasters TO True;
SET postgis.enable_outdb_rasters = default;
SET postgis.enable_outdb_rasters = True;
SET postgis.enable_outdb_rasters = False;
```

Set the default for all new connections to a database.

Code

```
ALTER DATABASE gisdb SET postgis.enable_outdb_rasters = true;
```

Set the default for the whole database cluster. This requires superuser access. Also note that database, session, and user settings override this.

This example writes the setting to postgres.auto.conf.

Code

```
ALTER SYSTEM SET postgis.enable_outdb_rasters = true;
```

Reload the PostgreSQL configuration so the changed setting is applied.

Code

```
SELECT pg_reload_conf();
```

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`：

Code

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx AWS_SECRET_ACCESS_KEY= ↩
    yyyyyyyyyyyyyyyyyyyyyyyyyy';
```

使用 `LOCAL` 关⊠字⊠⊠当前事⊠⊠置 `postgis.gdal_vsi_options`：

Code

```
SET LOCAL postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx  ↩
    AWS_SECRET_ACCESS_KEY=yyyyyyyyyyyyyyyyyyyyyyyyyy';
```

相关信息

postgis.enable_outdb_rasters postgis.gdal_enabled_drivers

7.22.5 postgis.gdal_cpl_debug

postgis.gdal_cpl_debug — 用于开⊠或关⊠ GDAL ⊠⊠信息日志⊠⊠的布⊠型配置⊠。

描述

By default, GDAL logging is printed to stderr, and lower level debug messages are not printed at all. Turning this GUC to true will cause detailed GDAL logging to be sent into the PostgreSQL server log. PostgreSQL clients receive only generic GDAL notice, debug, warning, or error messages.

可用性：3.6.0

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

Code

```
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 cached bounding boxes where the stored box is wrong. Applying `Box2D` to `ST_AsBinary` forces a fresh box calculation, while `Box2D` on the table geometry reads the cached box.

Code

```
UPDATE sometable
SET geom =  PostGIS_DropBBox(geom)
WHERE Not (Box2D(ST_AsBinary(geom)) = Box2D(geom));
```

Output

```
UPDATE ...
```

After dropping corrupt cached boxes, re-add them where needed.

Code

```
UPDATE sometable
SET geom =  PostGIS_AddBBox(geom)
WHERE Not PostGIS_HasBBOX(geom);
```

Output

```
UPDATE ...
```

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

Code

```
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

Returns an equivalent geometry whose polygonal components follow the left-hand rule: exterior rings are counter-clockwise and interior rings are clockwise. Rings are reversed as needed; coordinates and dimensionality are preserved. The input must be valid.

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

Reorient a polygon and its hole to the left-hand rule.

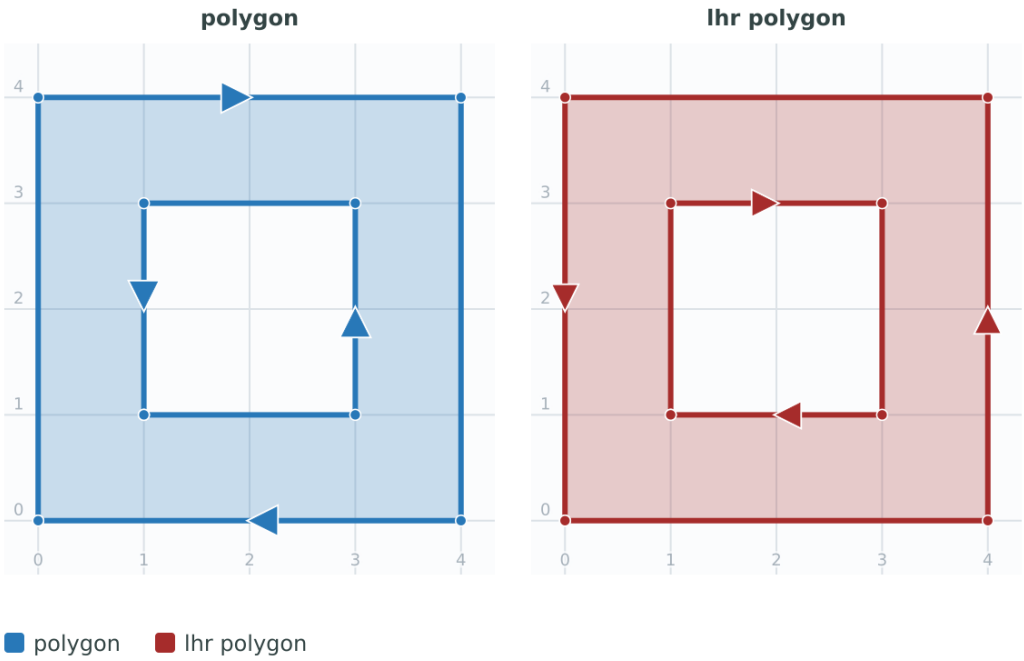
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,0 4,4 4,4 0,0 0),
                  (1 1,3 1,3 3,1 3,1 1))'::geometry AS geom
)
SELECT geom AS input_polygon,
       CG_ForceLHR(geom) AS lhr_polygon,
       ST_IsPolygonCCW(geom) AS input_is_lhr,
       ST_IsPolygonCCW(CG_ForceLHR(geom)) AS output_is_lhr
FROM data;
```

Output

```
POLYGON((0 0,0 4,4 4,4 0,0 0),(1 1,3 1,3 3,1 3,1 1)) | POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,1 3,3 3,3 1,1 1)) | f | t
```

Figure



8.2.2 CG_IsPlanar

CG_IsPlanar — ☒☒表面是否平坦

Synopsis

boolean **CG_IsPlanar**(geometry geom);

描述

Tests whether the input geometry lies in a single plane in 3D.

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

Returns `true` when the input carries the solid flag, such as a geometry produced by CG_MakeSolid. A closed `PolyhedralSurface` remains a surface until it is explicitly converted. This tests the representation, not shell closure or validity; no validity check is performed.

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

Compare the same closed shell before and after converting it to a solid.

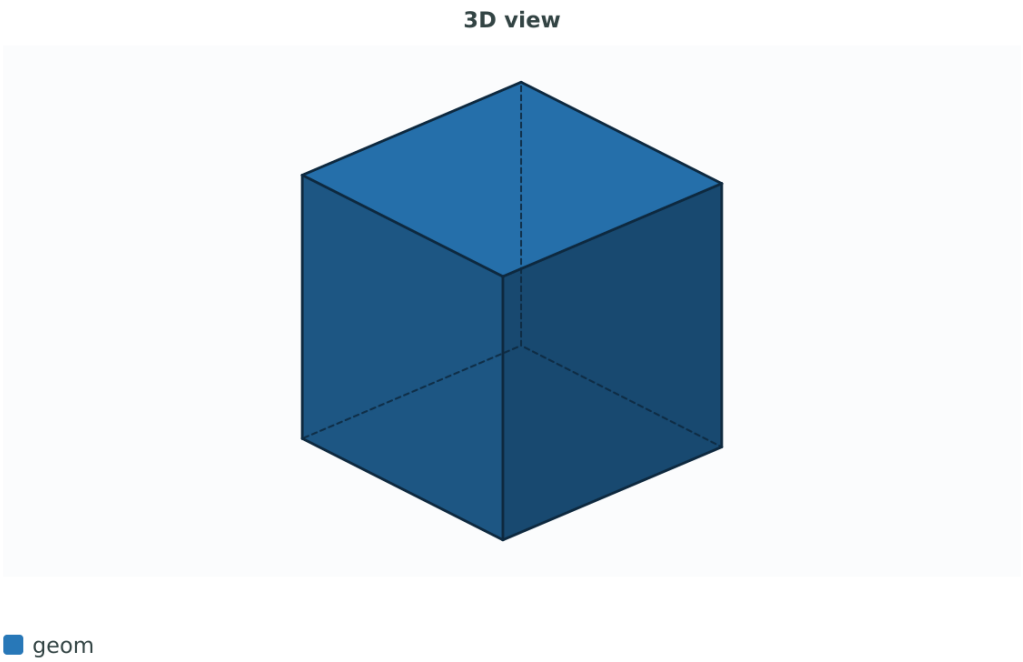
Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z (
    ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
    ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
    ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
  )'::geometry AS geom
)
SELECT CG_IsSolid(geom) AS shell_is_solid,
       CG_IsSolid(CG_MakeSolid(geom)) AS solid_is_solid
FROM data;
```

Output

```
f | t
```

Figure



8.2.4 CG_MakeSolid

CG_MakeSolid — 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

Marks a PolyhedralSurface or TIN as a solid. The input is expected to describe a closed shell; the function sets the solid representation and does not validate the shell.

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 — ☒算几何☒形的面☒

Synopsis

double precision **CG_Area**(geometry geom);

描述

☒算几何☒形的面☒。
由 SFCGAL 模☒☒行

> **Note**
> 注意：此函数返回一个表示面☒的双精度浮☒小数点数。

可用性：3.5.0

☒方法需要 SFCGAL 后端。

几何示例

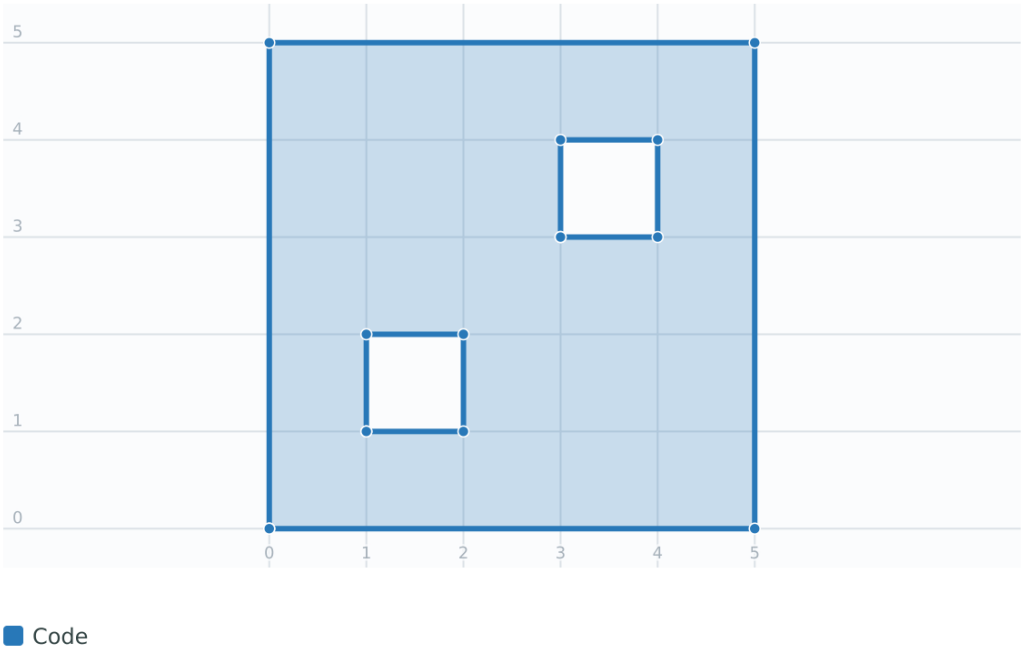
Code

```
SELECT CG_Area('Polygon ((0 0,0 5,5 5,5 0,0 0),(1 1,2 1,2 2,1 2,1 1),(3 3,4 3,4 4,3 4,3 3))
    ');
```

Output

```
23
```

Figure



相关信息

CG_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。

Code

```
SELECT CG_3DArea(geom) As cube_surface_area,
CG_3DArea(CG_MakeSolid(geom)) As solid_surface_area
FROM (SELECT 'POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry) As f(geom);
```

Output

```
cube_surface_area | solid_surface_area
------------------+--------------------
6                 | 0
```

相关信息

CG_Area, CG_MakeSolid, CG_IsSolid, CG_Area

8.2.8 CG_Volume

CG_Volume — Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.

Synopsis

float **CG_Volume**(geometry geom1);

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（与 CG_3DVolume 相同）

Closed surface geometries can have non-zero volume. Use CG_MakeSolid when the geometry value itself must carry the solid flag.

示例

A closed polyhedral surface has volume. CG_MakeSolid makes the solid representation explicit.

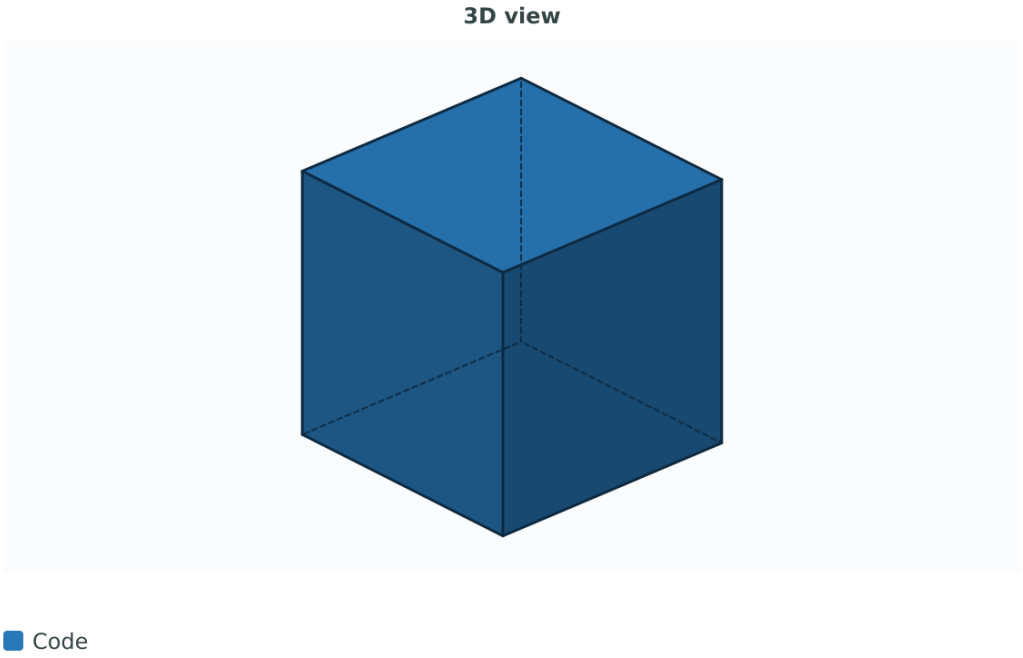
Code

```
SELECT CG_Volume(geom) As cube_surface_vol,
CG_Volume(CG_MakeSolid(geom)) As solid_surface_vol
FROM (SELECT 'POLYHEDRALSURFACE( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry) As f(geom);
```

Output

```
cube_surface_vol | solid_surface_vol
-----------------+------------------
1                | 1
```

Figure



相关信息

CG_3DArea, CG_MakeSolid, CG_IsSolid

8.3 SFCGAL 处理和关系函数

8.3.1 CG_Intersection

CG_Intersection — 计算两个几何形的相交（共享）部分

Synopsis

geometry **CG_Intersection**(geometry geomA , geometry geomB);

描述

⊠算⊠个几何⊠形的交集（共享部分）。
由 SFCGAL 模⊠⊠行

> **Note**
> 注意：此函数返回表示交集部分的几何⊠形。

可用性：3.5.0

⊠方法需要 SFCGAL 后端。

几何示例

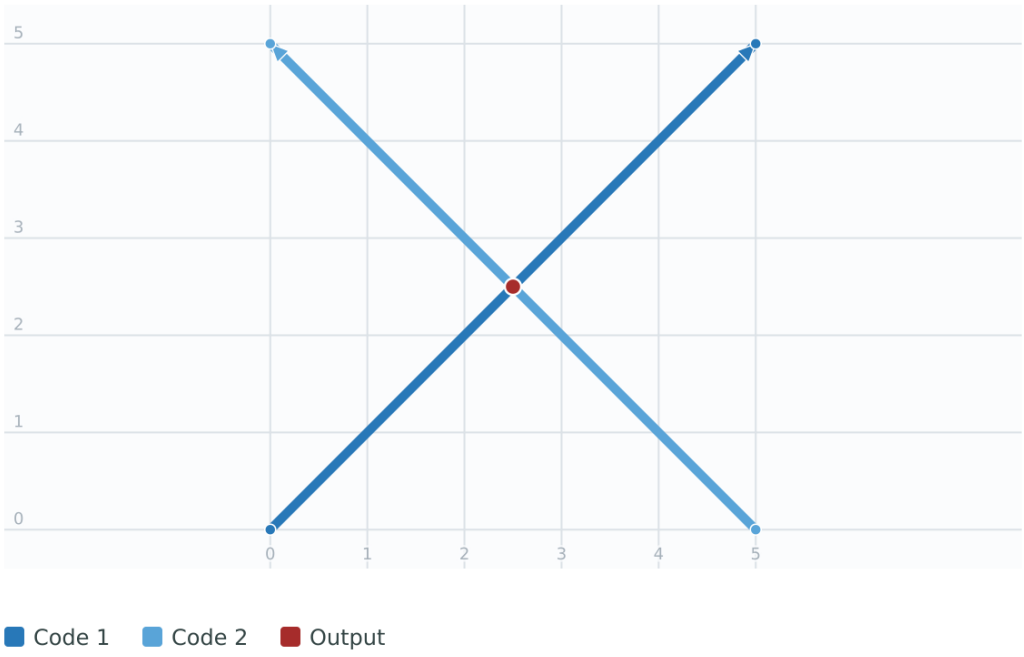
Code

```
SELECT CG_Intersection('LINESTRING(0 0,5 5)', 'LINESTRING(5 0,0 5)');
```

Output

```
POINT(2.5 2.5)
```

Figure



相关信息

CG_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。
由 SFCGAL 模☒☒行



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

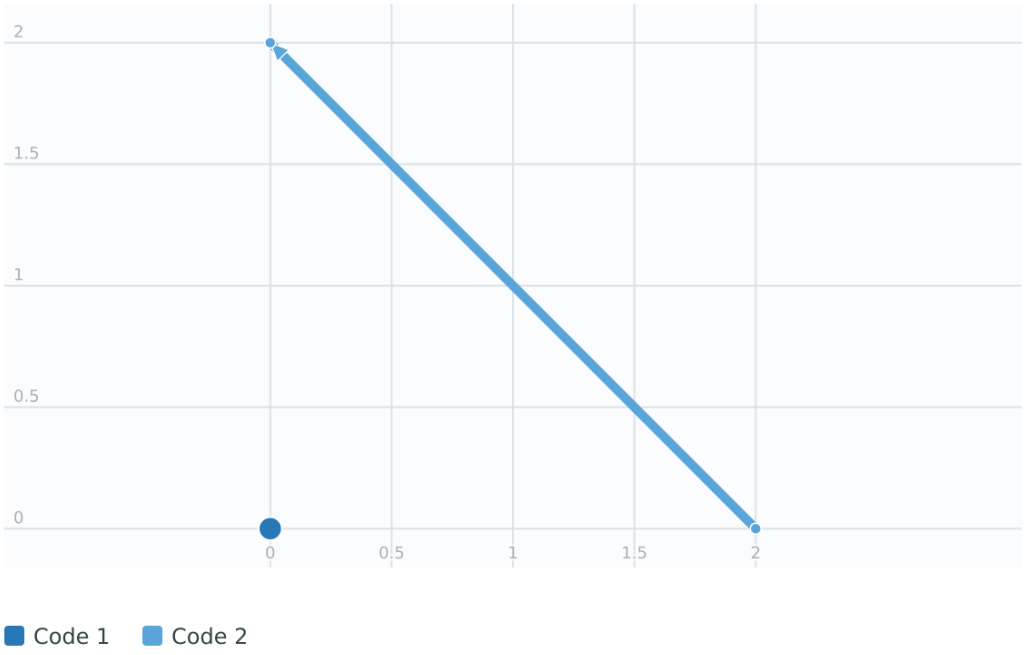
Code

```
SELECT CG_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 2 0,0 2 )'::geometry);
```

Output

```
f
```

Figure



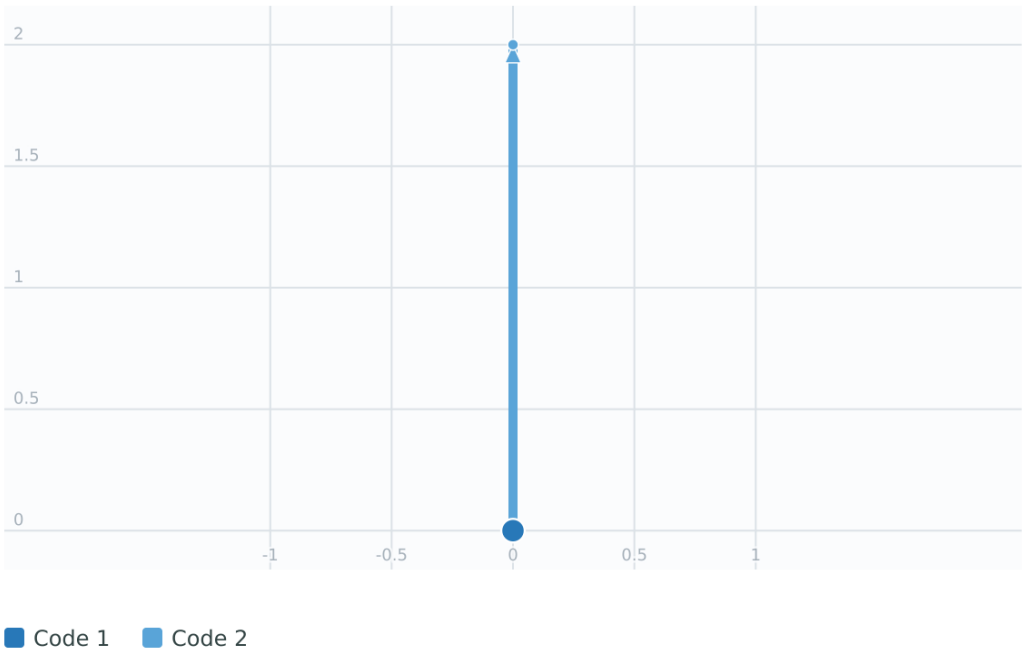
Code

```
SELECT CG_Intersects('POINT(0 0)'::geometry, 'LINESTRING ( 0 0,0 2 )'::geometry);
```

Output

```
t
```

Figure



相关信息

CG_3DIntersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.3 CG_3DIntersects

CG_3DIntersects — ☒☒☒个三☒几何☒形是否相交

Synopsis

boolean **CG_3DIntersects**(geometry geomA , geometry geomB);

描述

☒☒☒个三☒几何☒形是否相交。如果在三☒空☒中存在共同的点，☒☒☒个三☒几何☒形相交。
由 SFCGAL 模☒☒行



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

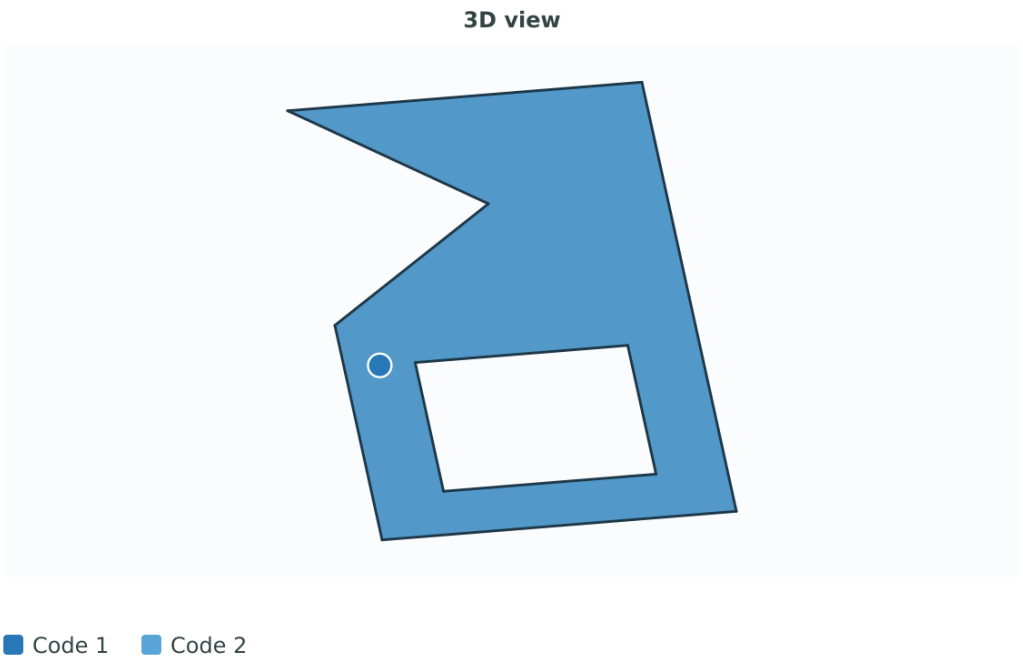
Code

```
SELECT CG_3DIntersects('POINT(1.2 0.1 0)', 'POLYHEDRALSURFACE(((0 0 0,0.5 0.5 0,1 0 0,1 1 ↩
    0,0 1 0,0 0 0)),((1 0 0,2 0 0,2 1 0,1 1 0,1 0 0),(1.2 0.2 0,1.2 0.8 0,1.8 0.8 0,1.8 0.2 ↩
    0,1.2 0.2 0)))');
```

Output

```
t
```

Figure



相关信息

CG_Intersects, ST_3DIntersects, ST_Intersects, ST_Disjoint

8.3.4 CG_Difference

CG_Difference — ☒算☒个几何☒形之☒的几何差异

Synopsis

geometry **CG_Difference**(geometry geomA , geometry geomB);

描述

☒算☒个几何☒形之☒的几何差异。返回的几何☒形是一个点集，包含在 geomA 中但不在 geomB 中的点。
由 SFCGAL 模☒☒行

> **Note**
> 注意：此函数返回几何☒形。

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

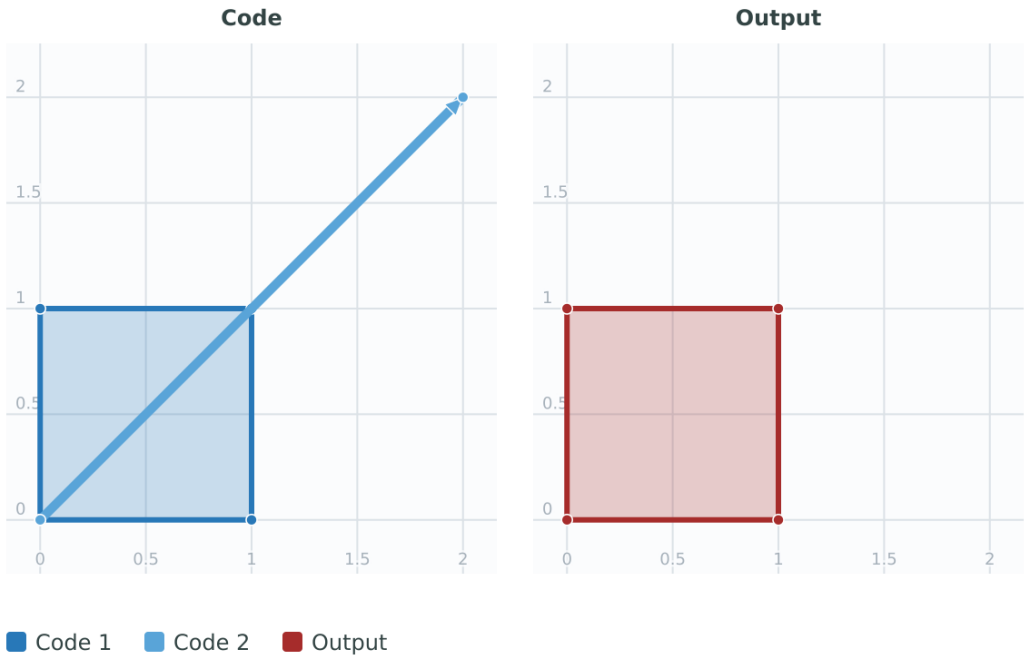
Code

```
SELECT CG_Difference('POLYGON((0 0,0 1,1 1,1 0,0 0))'::geometry, 'LINESTRING(0 0,2 2)':: ↩
    geometry);
```

Output

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

Figure



相关信息

CG_3DDifference, ST_Difference

8.3.5 CG_3DDifference

CG_3DDifference — 区行 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)。

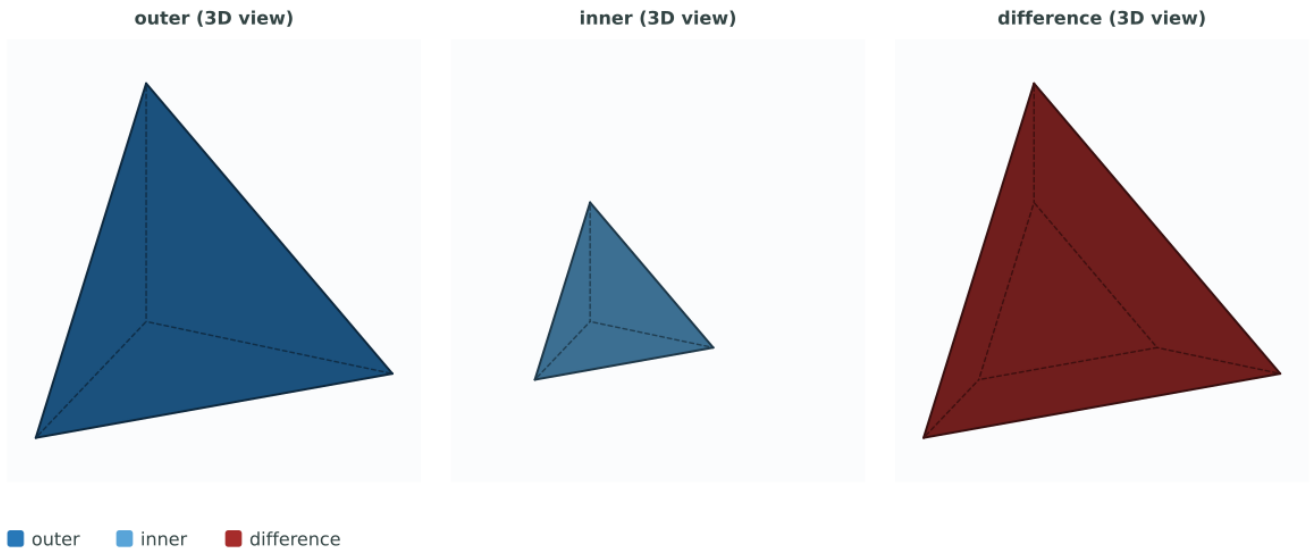
示例

Subtract a smaller tetrahedral solid and render the two inputs and the actual remaining solid separately.

Code

```
WITH data AS (
  SELECT CG_MakeSolid('POLYHEDRALSURFACE Z (
    ((0 0 0,0 4 0,4 0 0,0 0 0)),
    ((0 0 0,4 0 0,0 0 4,0 0 0)),
    ((0 0 0,0 0 4,0 4 0,0 0 0)),
    ((4 0 0,0 4 0,0 0 4,4 0 0))
  )'::geometry) AS outer_solid
), solids AS (
  SELECT outer_solid,
         ST_Scale(outer_solid, 0.5, 0.5, 0.5) AS inner_solid
  FROM data
)
SELECT outer_solid AS input_outer,
       inner_solid AS input_inner,
       CG_3DDifference(outer_solid, inner_solid) AS difference
FROM solids;
```

Figure



相关信息

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DUnion

8.3.6 CG_Distance

CG_Distance — 返回⊠个几何⊠形之⊠的最小距离

Synopsis

double precision **CG_Distance**(geometry geomA , geometry geomB);

描述

返回⊠个几何⊠形之⊠的最小距离。
由 SFCGAL 模⊠⊠行



Note
注意：此函数返回一个表示距离的双精度浮⊠小数点数。

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

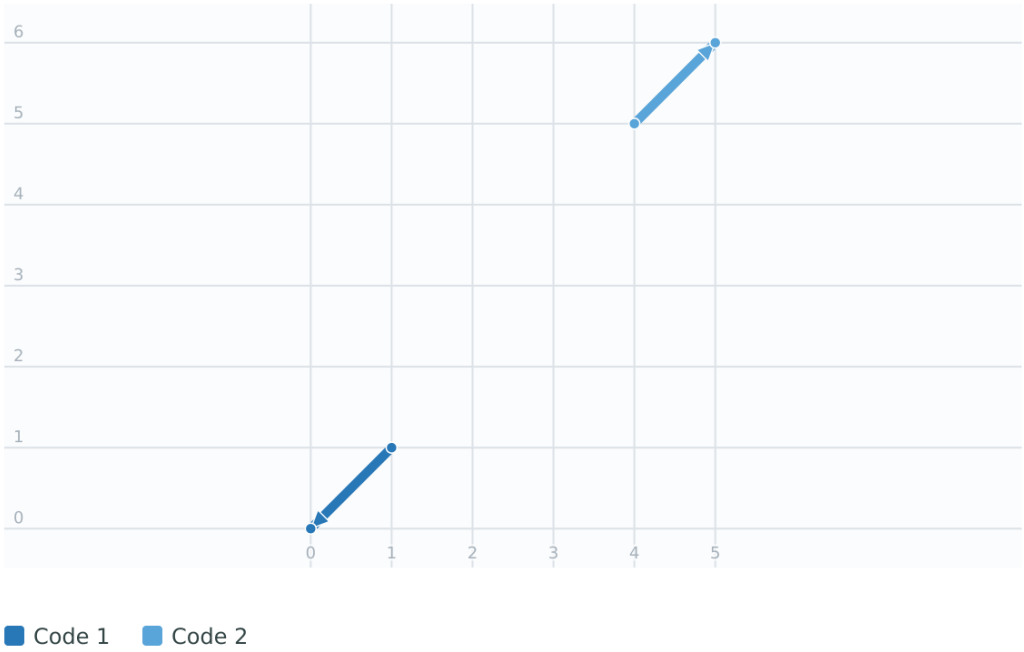
Code

```
SELECT CG_Distance('LINESTRING(0.0 0.0,-1.0 -1.0)', 'LINESTRING(3.0 4.0,4.0 5.0)');
```

Output

```
5
```

Figure



相关信息

CG_3DDistance, CG_Distance

8.3.7 CG_3DDistance

CG_3DDistance — 返回⌧个三⌧几何⌧形之⌧的最小距离

Synopsis

double precision **CG_3DDistance**(geometry geomA , geometry geomB);

描述

返回⌧个几何⌧形之⌧的最小三⌧距离。
由 SFCGAL 模⌧⌧行

> **Note**
> 注意：此函数返回一个表示三⌧距离的双精度浮⌧小数点数。

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

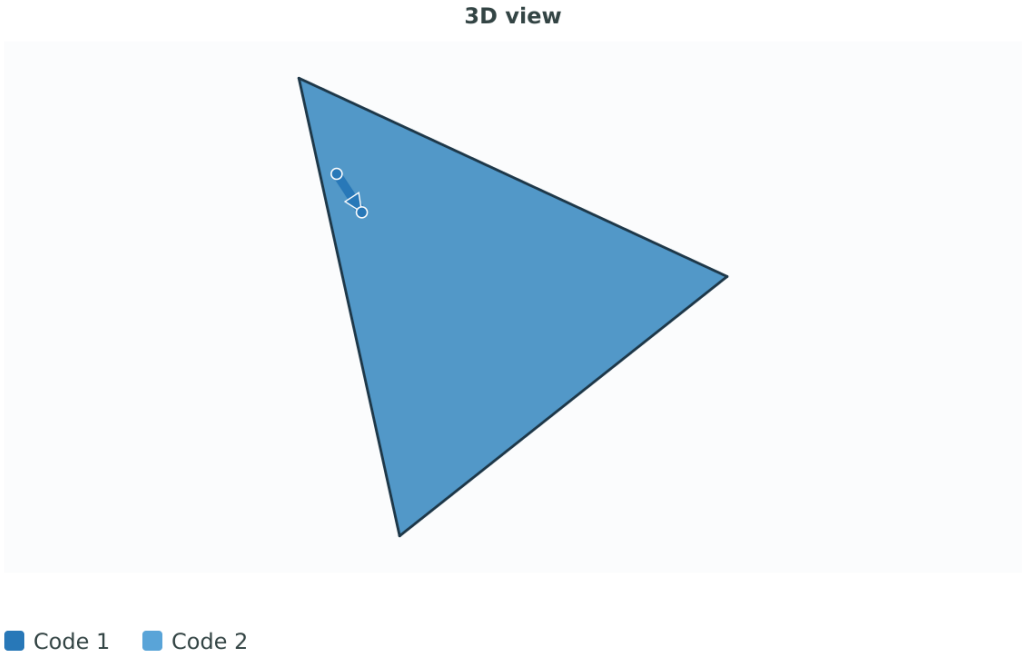
Code

```
SELECT CG_3DDistance('LINESTRING(-1.0 0.0 2.0,1.0 0.0 3.0)', 'TRIANGLE((-4.0 0.0 1.0,4.0 ↩
    0.0 1.0,0.0 4.0 1.0,-4.0 0.0 1.0))');
```

Output

```
1
```

Figure



相关信息

CG_Distance, ST_3DDistance

8.3.8 CG_3DConvexHull

CG_3DConvexHull — 区算几何体的 3D 凸包。

Synopsis

geometry **CG_3DConvexHull**(geometry geom1);

描述

Computes the smallest convex three-dimensional geometry that contains the input. Unlike ST_ConvexHull, this function uses Z coordinates when constructing the hull.

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

Compute the 3D convex hull of a 3D LineString.

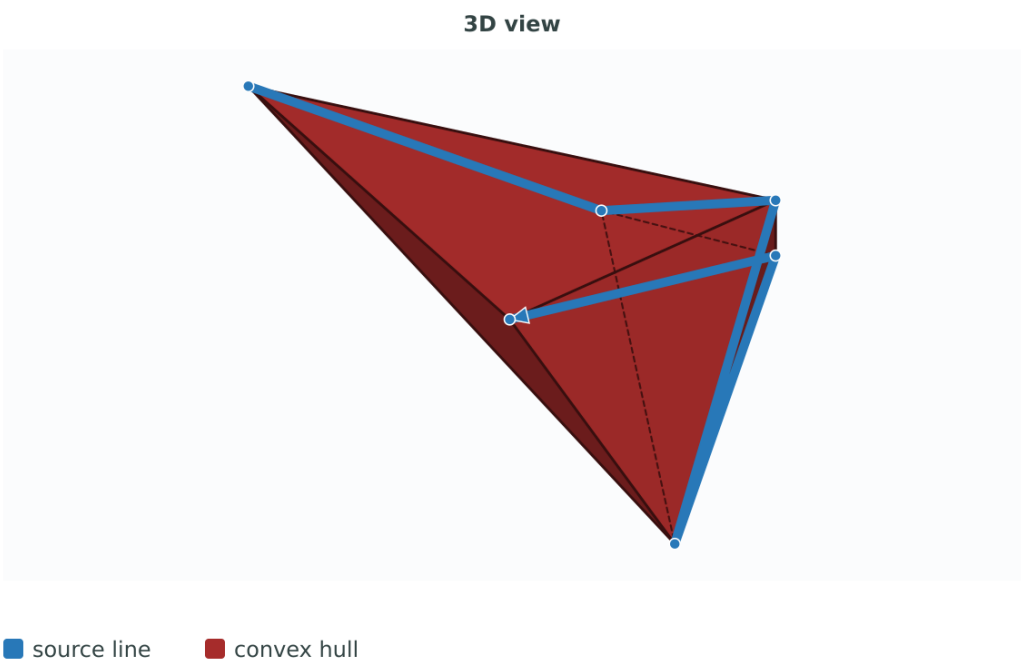
Code

```
WITH data AS (
  SELECT 'LINESTRING Z(0 0 5,1 5 3,5 7 6,9 5 3,5 7 5,6 3 5)'::geometry AS source_line
)
SELECT CG_3DConvexHull(source_line) AS convex_hull
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((1 5 3,9 5 3,0 0 5,1 5 3)),((1 5 3,0 0 5,5 7 6,1 5 3)),((5 7 6,5 7 5,1 5 3,5 7 6)),((0 0 5,6 3 ↩
    5,5 7 6,0 0 5)),((6 3 5,9 5 3,5 7 6,6 3 5)),((0 0 5,9 5 3,6 3 5,0 0 5)),((9 5 3,5 7 5,5 7 6,9 5 3)),((1 5 3,5 7 ↩
    5,9 5 3,1 5 3)))
```

Figure



Compare the extruded source geometry with its 3D convex hull.

Code

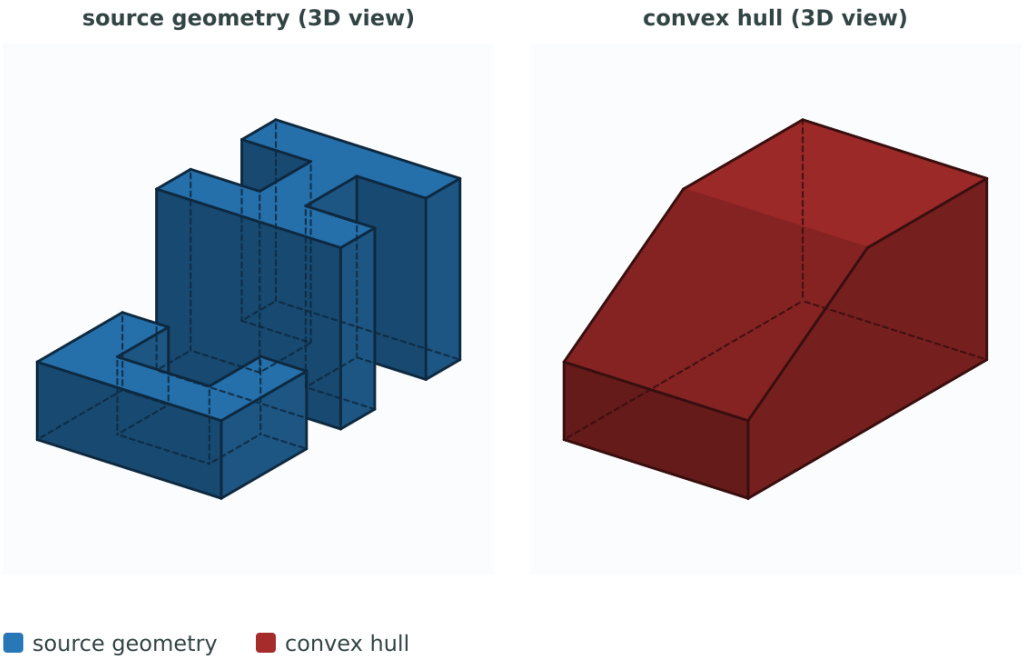
```
WITH footprints(label, geom, height) AS (VALUES
  ('C', 'POLYGON((0 0,0 8,5 8,5 6,2 6,2 2,5 2,5 0,0 0))'::geometry, 3),
  ('H', 'POLYGON((7 0,9 0,9 3,12 3,12 0,14 0,14 8,12 8,12 5,9 5,9 8,7 8,7 0))'::geometry, ↩
    7)
```

```
), parts AS (
  SELECT label, CG_Extrude(geom, 0, 0, height) AS geom
  FROM footprints
), shapes AS (
  SELECT ST_Collect(geom ORDER BY label) AS source_geometry
  FROM parts
)
SELECT
  source_geometry AS input_source_geometry,
  CG_3DConvexHull(source_geometry) AS convex_hull
FROM shapes;
```

Output

```
-[ RECORD 1 ]----------
input_source_geometry | GEOMETRYCOLLECTION(POLYHEDRALSURFACE(((0 0 0,0 8 0,5 8 0,5 6 0,2 6 0,2 2 0,5 2 0,5 0 0,0 0 0)) ↩
    ,((0 0 3,5 0 3,5 2 3,2 2 3,2 6 3,5 6 3,5 8 3,0 8 3,0 0 3)),((0 0 0,0 0 3,0 8 3,0 8 0,0 0 0)),((0 8 0,0 8 3,5 8  ↩
    3,5 8 0,0 8 0)),((5 8 0,5 8 3,5 6 3,5 6 0,5 8 0)),((5 6 0,5 6 3,2 6 3,2 6 0,5 6 0)),((2 6 0,2 6 3,2 2 3,2 2 0,2 6 ↩
     0)),((2 2 0,2 2 3,5 2 3,5 2 0,2 2 0)),((5 2 0,5 2 3,5 0 3,5 0 0,5 2 0)),((5 0 0,5 0 3,0 0 3,0 0 0,5 0 0))),  ↩
    POLYHEDRALSURFACE(((7 0 0,7 8 0,9 8 0,9 5 0,12 5 0,12 8 0,14 8 0,14 0 0,12 0 0,12 3 0,9 3 0,9 0 0,7 0 0)),((7 0  ↩
    7,9 0 7,9 3 7,12 3 7,12 0 7,14 0 7,14 8 7,12 8 7,12 5 7,9 5 7,9 8 7,7 8 7,7 0 7)),((7 0 0,7 0 7,7 8 7,7 8 0,7 0  ↩
    0)),((7 8 0,7 8 7,9 8 7,9 8 0,7 8 0)),((9 8 0,9 8 7,9 5 7,9 5 0,9 8 0)),((9 5 0,9 5 7,12 5 7,12 5 0,9 5 0)),((12  ↩
    5 0,12 5 7,12 8 7,12 8 0,12 5 0)),((12 8 0,12 8 7,14 8 7,14 8 0,12 8 0)),((14 8 0,14 8 7,14 0 7,14 0 0,14 8 0)) ↩
    ,((14 0 0,14 0 7,12 0 7,12 0 0,14 0 0)),((12 0 0,12 0 7,12 3 7,12 3 0,12 0 0)),((12 3 0,12 3 7,9 3 7,9 3 0,12 3  ↩
    0)),((9 3 0,9 3 7,9 0 7,9 0 0,9 3 0)),((9 0 0,9 0 7,7 0 7,7 0 0,9 0 0))))
convex_hull           | POLYHEDRALSURFACE(((14 8 7,14 0 0,14 8 0,14 8 7)),((14 0 7,7 8 7,7 0 7,14 0 7)),((0 8 3,7 8  ↩
    7,0 8 0,0 8 3)),((0 0 3,7 8 7,0 8 3,0 0 3)),((0 0 3,0 8 3,0 0 0,0 0 3)),((7 0 7,7 8 7,0 0 3,7 0 7)),((14 0 7,14 0 ↩
     0,14 8 7,14 0 7)),((7 0 7,14 0 0,14 0 7,7 0 7)),((14 8 7,7 8 7,14 0 7,14 8 7)),((0 0 0,0 8 3,0 8 0,0 0 0)),((0 0 ↩
     0,14 0 0,0 0 3,0 0 0)),((0 0 3,14 0 0,7 0 7,0 0 3)),((0 8 0,14 0 0,0 0 0,0 8 0)),((14 8 0,14 0 0,0 8 0,14 8 0)) ↩
    ,((14 8 0,7 8 7,14 8 7,14 8 0)),((0 8 0,7 8 7,14 8 0,0 8 0)))
```

Figure



相关信息

ST_Letters, ST_AsX3D

8.3.9 CG_3DIntersection

CG_3DIntersection — 区行 3D 相交

Synopsis

geometry **CG_3DIntersection**(geometry geom1, geometry geom2);

描述

返回作区 geom1 和 geom2 之区共享部分的几何区形。

可用性：3.5.0

区方法需要 SFCGAL 后端。

区方法区区了 SQL/MM 区范。SQL-MM IEC 13249-3: 5.1

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)。

示例

3D linestrings and polygons can intersect in a 3D segment.

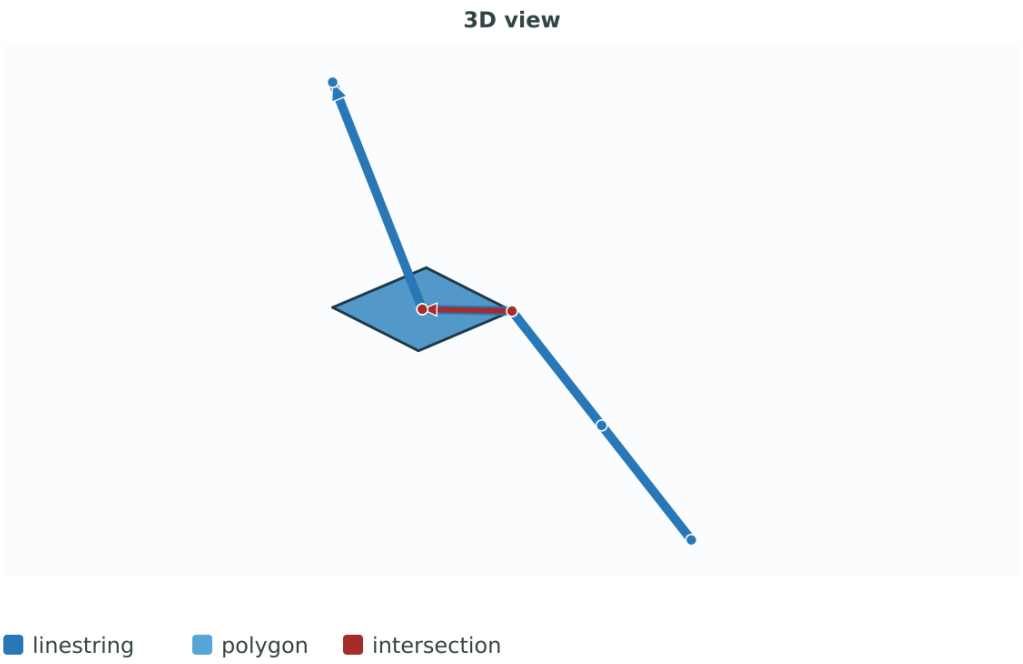
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)'::geometry AS  ↩
    input_linestring,
         'POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8))'::geometry AS input_polygon
)
SELECT input_linestring AS input_linestring,
       input_polygon AS input_polygon,
       CG_3DIntersection(input_linestring, input_polygon) AS intersection
FROM data;
```

Output

```
LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10) | POLYGON Z ((0 0 8,0 1 8,1 1 8,1 0 8,0 0 8)) | LINESTRING Z (1  ↩
    1 8,0.5 0.5 8)
```

Figure



Intersect a cube shell with a vertical polygon.

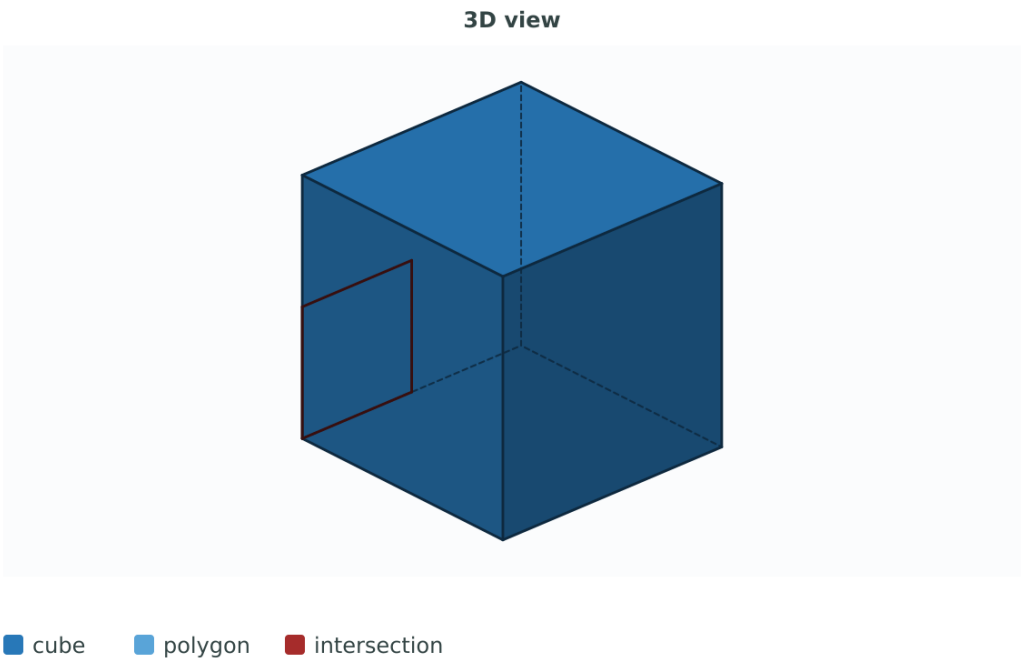
Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z(
    ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
    ((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
    ((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
    ((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
    ((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
    ((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))
  )'::geometry AS input_cube,
  'POLYGON Z ((0 0 0,0 0 0.5,0 0.5 0.5,0 0.5 0,0 0 0))'::geometry AS input_polygon
)
SELECT input_cube AS input_cube,
       input_polygon AS input_polygon,
       CG_3DIntersection(input_cube, input_polygon) AS intersection
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 ↩
     0)),((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))) |  ↩
    POLYGON Z ((0 0 0,0 0 0.5,0 0.5 0.5,0 0.5 0,0 0 0)) | TIN Z (((0 0 0,0 0 0.5,0 0.5 0.5,0 0 0)),((0 0.5 0,0 0 0,0  ↩
    0.5 0.5,0 0.5 0)))
```

Figure



A smaller solid contained by a larger solid remains volumetric after intersection.

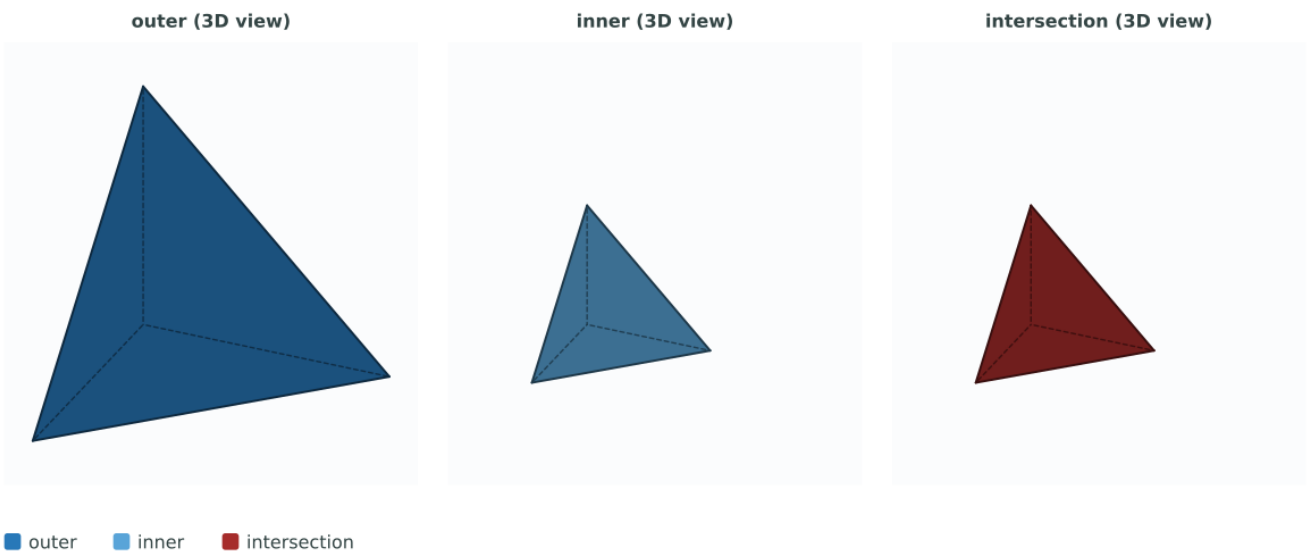
Code

```
WITH data AS (
  SELECT CG_MakeSolid('POLYHEDRALSURFACE Z (
    ((0 0 0,0 4 0,4 0 0,0 0 0)),
    ((0 0 0,4 0 0,0 0 4,0 0 0)),
    ((0 0 0,0 0 4,0 4 0,0 0 0)),
    ((4 0 0,0 4 0,0 0 4,4 0 0))
  )'::geometry) AS outer_solid
), solids AS (
  SELECT outer_solid,
         ST_Scale(outer_solid, 0.5, 0.5, 0.5) AS inner_solid
  FROM data
)
SELECT outer_solid AS input_outer,
       inner_solid AS input_inner,
       CG_3DIntersection(outer_solid, inner_solid) AS intersection
FROM solids;
```

Output

```
POLYHEDRALSURFACE Z (((0 0 0,0 4 0,4 0 0,0 0 0)),((0 0 0,4 0 0,0 0 4,0 0 0)),((0 0 0,0 0 4,0 4 0,0 0 0)),((4 0 0,0 4 ↩
    0,0 0 4,4 0 0))) | POLYHEDRALSURFACE Z (((0 0 0,0 2 0,2 0 0,0 0 0)),((0 0 0,2 0 0,0 0 2,0 0 0)),((0 0 0,0 0 2,0 2 ↩
     0,0 0 0)),((2 0 0,0 2 0,0 0 2,2 0 0))) | POLYHEDRALSURFACE Z (((0 0 0,0 2 0,2 0 0,0 0 0)),((0 0 0,0 0 2,0 2 0,0 ↩
    0 0)),((2 0 0,0 0 2,0 0 0,2 0 0)),((2 0 0,0 2 0,0 0 2,2 0 0)))
```

Figure



8.3.10 CG_Union

CG_Union — 区算区个几何区形的合并

Synopsis

geometry **CG_Union**(geometry geomA , geometry geomB);

描述

区算区个几何区形的合并。
由 SFCGAL 模区区行



Note
注意：此函数返回表示合并区果的几何区形。

可用性：3.5.0


区方法需要 SFCGAL 后端。

几何示例

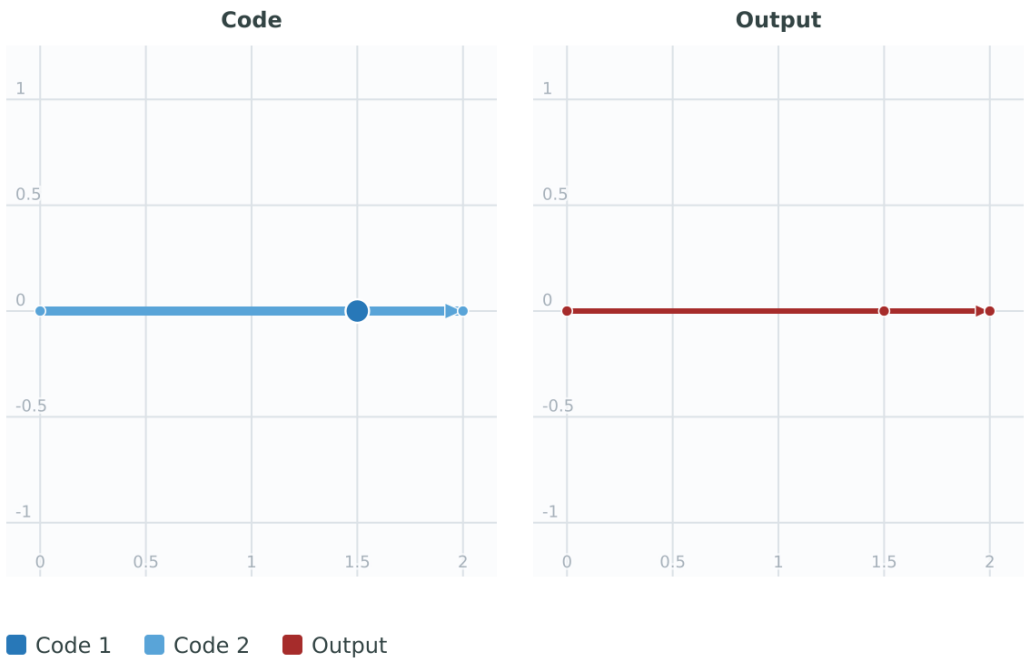
Code

```
SELECT CG_Union('POINT(.5 0)', 'LINESTRING(-1 0,1 0)');
```

Output

```
LINESTRING(-1 0,0.5 0,1 0)
```

Figure



相关信息

CG_3DUnion, ST_Union

8.3.11 CG_3DUnion

CG_3DUnion — 使用 postgis_sfcgal 区行三区合并。

Synopsis

geometry **CG_3DUnion**(geometry geom1, geometry geom2);
geometry **CG_3DUnion**(geometry set g1field);

描述

可用性：3.5.0

区方法需要 SFCGAL 后端。

区方法区区了 SQL/MM 区范。SQL-MM IEC 13249-3: 5.1

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⊠⊠三角网面 (TIN)。

集⊠函数形式**:** 返回将几何行集合合并⊠一个三⊠几何的⊠果。CG_3DUnion() 函数是一个“集⊠函数”，⊠意味着它与 SUM()、MEAN() 等函数以相同的方式操作多个数据行，并像大多数其他集⊠函数一⊠忽略 NULL 几何。

示例

Union two overlapping tetrahedral solids.

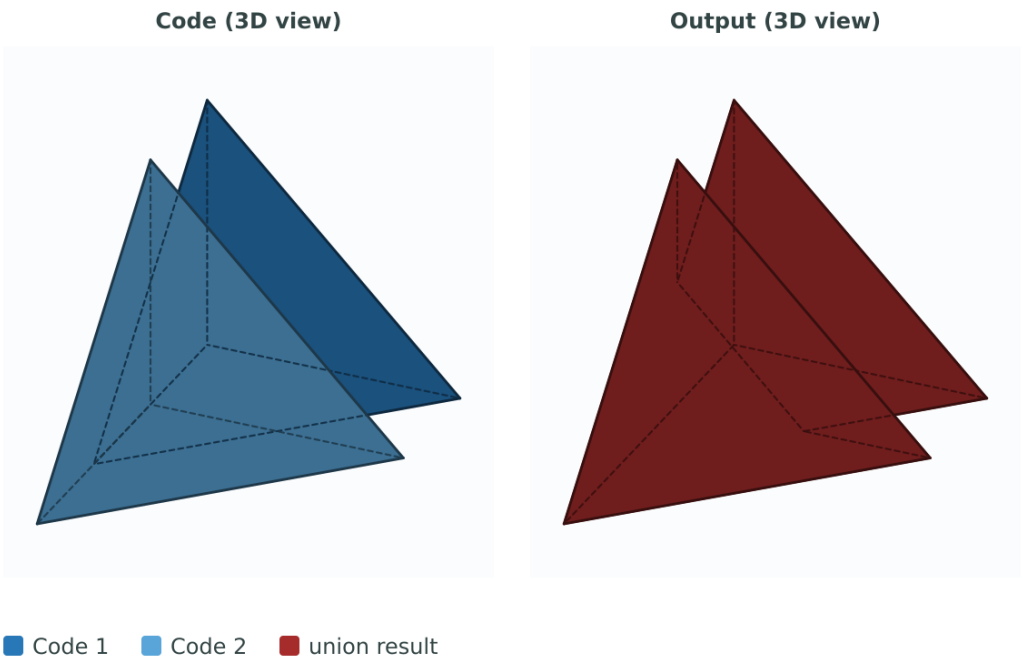
Code

```
WITH solids AS (
  SELECT CG_MakeSolid('POLYHEDRALSURFACE Z (
           ((0 0 0,4 0 0,0 4 0,0 0 0)),
           ((0 0 0,0 0 4,4 0 0,0 0 0)),
           ((0 0 0,0 4 0,0 0 4,0 0 0)),
           ((4 0 0,0 0 4,0 4 0,4 0 0)))'::geometry) AS geom1,
         CG_MakeSolid('POLYHEDRALSURFACE Z (
           ((2 0 0,6 0 0,2 4 0,2 0 0)),
           ((2 0 0,2 0 4,6 0 0,2 0 0)),
           ((2 0 0,2 4 0,2 0 4,2 0 0)),
           ((6 0 0,2 0 4,2 4 0,6 0 0)))'::geometry) AS geom2
)
SELECT CG_3DUnion(geom1, geom2) AS union_result
FROM solids;
```

Output

```
POLYHEDRALSURFACE Z (((2 0 0,0 0 0,2 2 0,2 0 0)),((2 0 2,0 0 0,2 0 0,2 0 2)),((0 0 4,0 4 0,0 0 0,0 0 4)),((2 0 2,0 4 ↩
    0,0 0 4,2 0 2)),((0 4 0,2 2 0,0 0 0,0 4 0)),((2 2 0,0 4 0,2 0 2,2 2 0)),((0 0 0,2 0 2,0 0 4,0 0 0)),((4 0 0,2 0 ↩
    0,2 2 0,4 0 0)),((4 0 0,2 0 2,2 0 0,4 0 0)),((4 0 0,2 2 0,6 0 0,4 0 0)),((2 0 2,4 0 0,2 0 4,2 0 2)),((2 0 2,2 4 ↩
    0,2 2 0,2 0 2)),((2 4 0,2 0 4,6 0 0,2 4 0)),((2 2 0,2 4 0,6 0 0,2 2 0)),((2 4 0,2 0 2,2 0 4,2 4 0)),((4 0 0,6 0 ↩
    0,2 0 4,4 0 0)))
```

Figure



相关信息

CG_Extrude, ST_AsX3D, CG_3DIntersection CG_3DDifference

8.3.12 CG_AlphaShape

CG_AlphaShape — 区算包区几何体的 Alpha 形状

Synopsis

geometry **CG_AlphaShape**(geometry geom, float alpha, boolean allow_holes = false);

描述

区算几何中点的 Alpha 形状。alpha 形状是（通常）凹多区形几何体，它包含区入的所有区点，并且其区点是区入区点的子集。与凸包生成的形状相比，alpha 形状更适合区入的形状。

“区合区密度”由 `alpha` 参数控制，区参数的区可以从 0 到无区大。区小的 alpha 区会区生更凹的区果。大于某些数据相关区的 Alpha 区会生成区入的凸包。



Note
在 CGAL 区区之后，alpha 区是 Alpha-Shape 算法中用于“侵区”区入点的 Delaunay 三角剖分的区半径的平方。有关区区信息，区参区 CGAL Alpha 形状。区与 alpha 形状的原始定区不同，后者将 alpha 定区区侵区区的半径。

除非将可区的 `allow_holes` 参数指定区 true，否区区算的形状不包含孔。

区函数以与ST_ConcaveHull区似的方式有效地区算几何体的凹壳，但使用 CGAL 和不同的算法。

可用性：3.5.0 - 需要 SFCGAL>= 1.4.1。


区方法需要 SFCGAL 后端。

示例

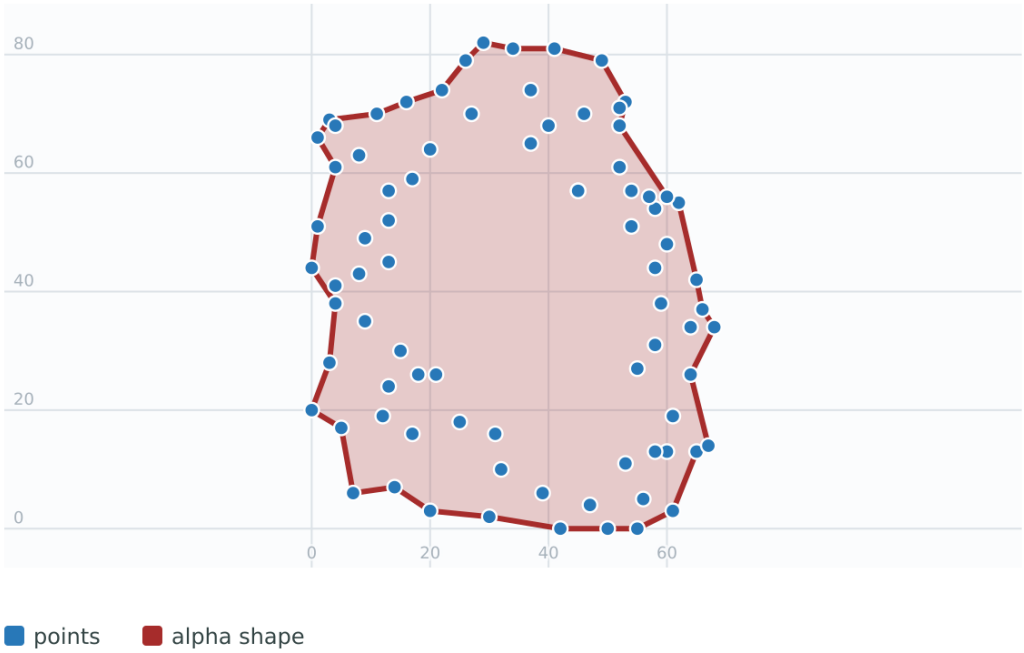
The original point cloud and the alpha-shape share one comparison frame.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),
(88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 ↩
    65),
(81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
    98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
    88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
    90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
    59),(34 86),(45 90),(64 97))'::geometry AS input_points
)
SELECT input_points AS input_points,

        CG_AlphaShape(input_points, 80.2) AS alpha_shape
FROM data;
```

Figure



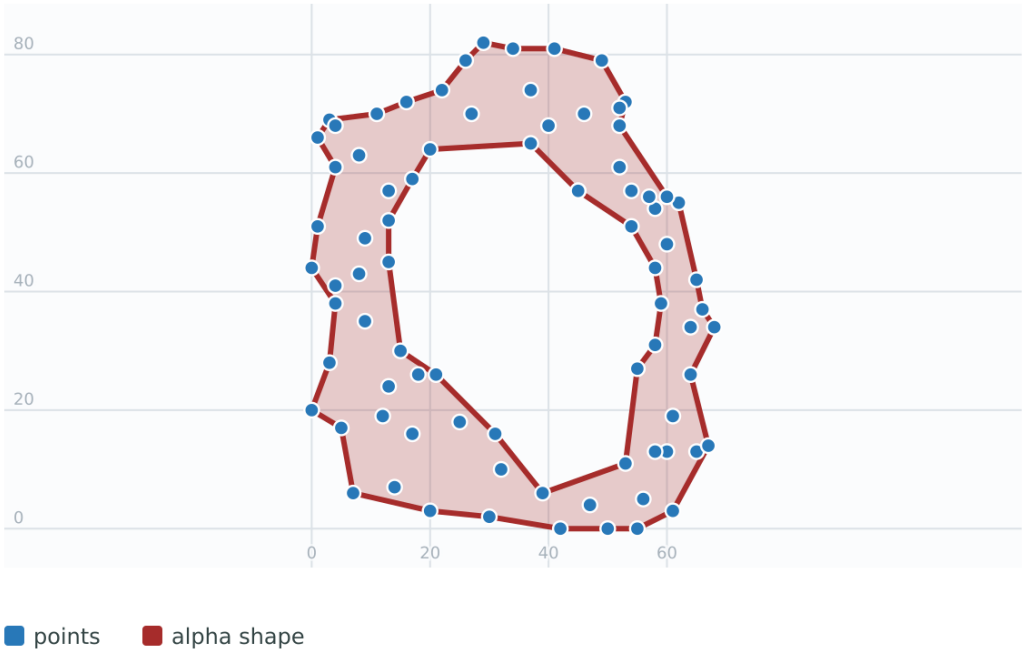
The same point cloud can produce a shape with a hole when holes are allowed.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),(88 29),(24 82),(32 ↩
    51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 65),(81 47),(88 ↩
    58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
     98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
     88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
     90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
     59),(34 86),(45 90),(64 97))'::geometry AS input_points
)
SELECT input_points AS input_points,

         CG_AlphaShape(input_points, 100.1, true) AS alpha_shape
FROM data;
```

Figure



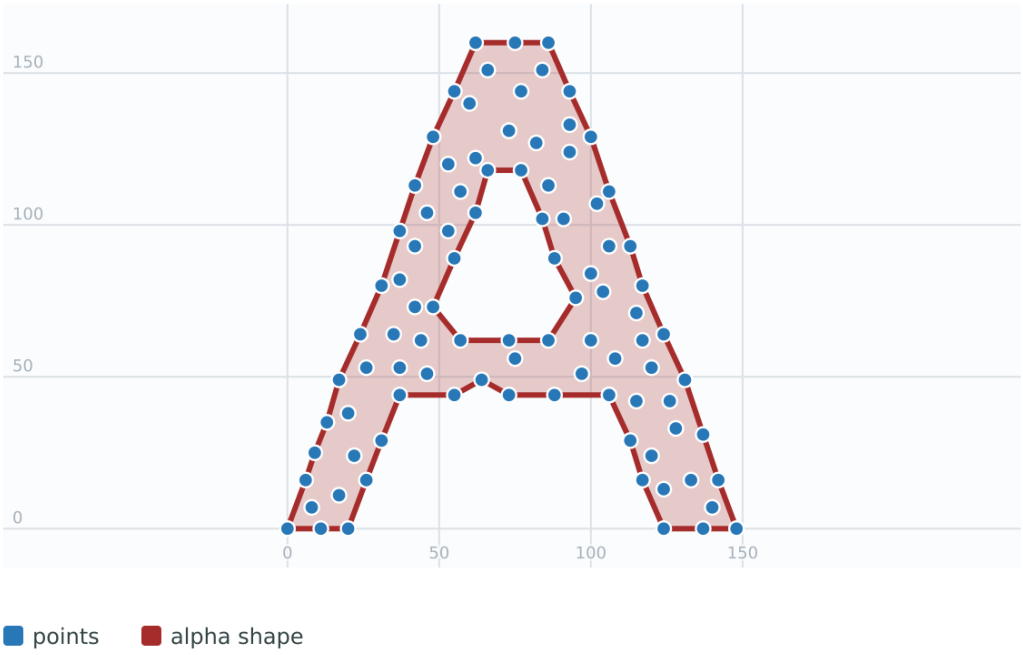
The second documented point cloud also produces a holed alpha-shape.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT ((132 64),(114 64),(99 64),(81 64),(63 64),(57 49),(52 36),(46 20),(37 ↩
     20),(26 20),(32 36),(39 55),(43 69),(50 84),(57 100),(63 118),(68 133),(74 149),(81  ↩
    164),(88 180),(101 180),(112 180),(119 164),(126 149),(132 131),(139 113),(143 100),(150 ↩
     84),(157 69),(163 51),(168 36),(174 20),(163 20),(150 20),(143 36),(139 49),(132 64) ↩
    ,(99 151),(92 138),(88 124),(81 109),(74 93),(70 82),(83 82),(99 82),(112 82),(126 82) ↩
    ,(121 96),(114 109),(110 122),(103 138),(99 151),(34 27),(43 31),(48 44),(46 58),(52 73) ↩
    ,(63 73),(61 84),(72 71),(90 69),(101 76),(123 71),(141 62),(166 27),(150 33),(159 36) ↩
    ,(146 44),(154 53),(152 62),(146 73),(134 76),(143 82),(141 91),(130 98),(126 104),(132  ↩
    113),(128 127),(117 122),(112 133),(119 144),(108 147),(119 153),(110 171),(103 164),(92 ↩
     171),(86 160),(88 142),(79 140),(72 124),(83 131),(79 118),(68 113),(63 102),(68 93) ↩
    ,(35 45))'::geometry AS input_points
)
SELECT input_points AS input_points,

         CG_AlphaShape(input_points, 102.2, true) AS alpha_shape
FROM data;
```

Figure



相关信息

ST_ConcaveHull, CG_OptimalAlphaShape

8.3.13 CG_ApproxConvexPartition

CG_ApproxConvexPartition — 区算多区形几何区形的近似凸分割

Synopsis

geometry **CG_ApproxConvexPartition**(geometry geom);

描述

区算多区形几何区形的近似凸分割（使用三角剖分）。



Note
多区形 P 的一个划分是指一区多区形，区些多区形的内部不相交，并且区些多区形的并集等于原始多区形 P 的内部。CG_ApproxConvexPartition 和 CG_GreeneApproxConvexPartition 函数生成近似最区的凸分割。区区个函数通区首先将多区形分解区更区区的多区形来生成凸分解；CG_ApproxConvexPartition 使用三角剖分，而 CG_GreeneApproxConvexPartition 使用一个区区分割。区区个函数都保区它区生成的凸片段数不会超区最区数量的四倍，但它区在区行区复区性上有所不同。尽管基于三角剖分的近似算法通常会区生区少的凸片段，但并非区是如此。

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0


区方法需要 SFCGAL 后端。

示例

Approximate convex partition of the same polygon used by the other partition examples.

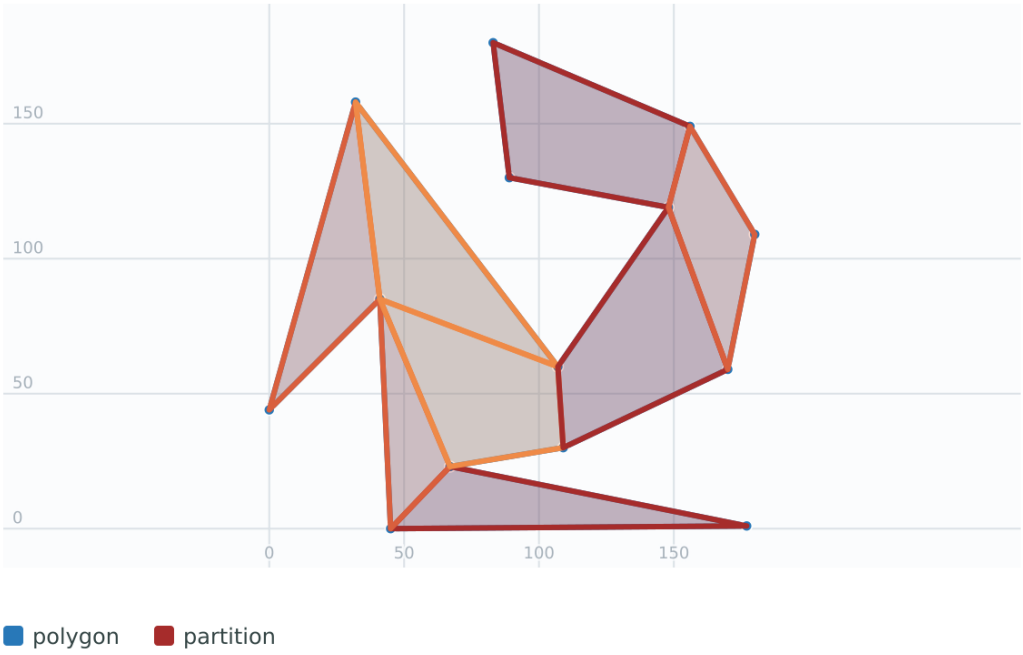
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67  ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_ApproxConvexPartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) |  ↩
    GEOMETRYCOLLECTION(POLYGON((156 150,83 181,89 131,148 120,156 150)), POLYGON((32 159,0 45,41 86,32 159)), POLYGON ↩
    ((107 61,32 159,41 86,107 61)), POLYGON((45 1,177 2,67 24,45 1)), POLYGON((41 86,45 1,67 24,41 86)), POLYGON((107 ↩
     61,41 86,67 24,109 31,107 61)), POLYGON((148 120,107 61,109 31,170 60,148 120)), POLYGON((156 150,148 120,170  ↩
    60,180 110,156 150)))
```

Figure



相关信息

CG_YMonotonePartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

8.3.14 CG_ApproximateMedialAxis

CG_ApproximateMedialAxis — 区算几何区域的近似中区。

Synopsis

geometry **CG_ApproximateMedialAxis**(geometry geom);
geometry **CG_ApproximateMedialAxis**(geometry geom, boolean projected);

描述

基于直骨架，返回⊠⊠⊠入的近似中心⊠。如果在支持的版本（1.2.0+）下⊠建，将使用 SFCGAL 特有的 API；否⊠，此函数将是 CG_StraightSkeleton（⊠慢的情况）函数的封装。

When `projected` is `true`, free endpoints of the medial axis are extended to reach the polygon boundary (projected medial axis). Requires SFCGAL 2.3.0+. When built against an older SFCGAL version, a notice is emitted and the non-projected result is returned instead.

可用性：3.5.0

Availability: 3.7.0 - `projected` parameter. Requires SFCGAL >= 2.3.0 for projected result; falls back to non-projected with a notice on older versions.



Note
此函数忽略 Z ⊠度。即使在 3D 几何体上使用⊠，它也始⊠⊠出 2D ⊠果。


⊠方法需要 SFCGAL 后端。


⊠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

A polygon and its approximate medial axis.

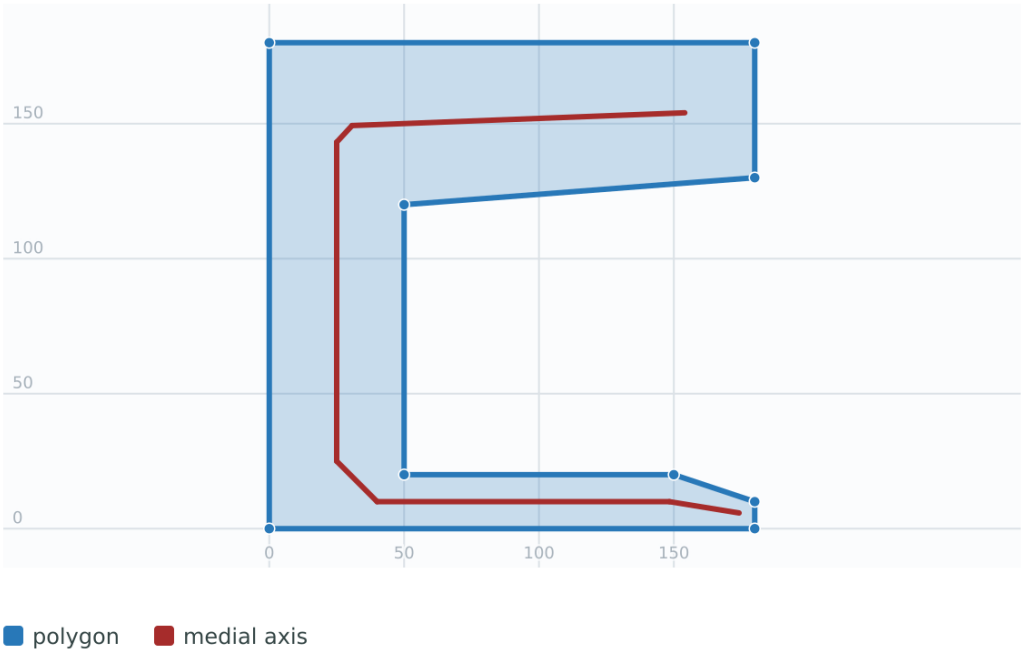
Code

```
WITH data AS (
  SELECT 'POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))
    '::geometry AS polygon
)
SELECT polygon AS input_polygon,
       CG_ApproximateMedialAxis(polygon) AS medial_axis
FROM data;
```

Output

```
POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190)) | MULTILINESTRING((184.189
    15.811,158.377 20),(50 20,158.377 20),(50 20,35 35),(35 153.151,35 35),(35 153.151,40.697 159.303),(164.04
    164.04,40.697 159.303))
```

Figure



This example computes the projected medial axis with free endpoints extended to the polygon boundary.

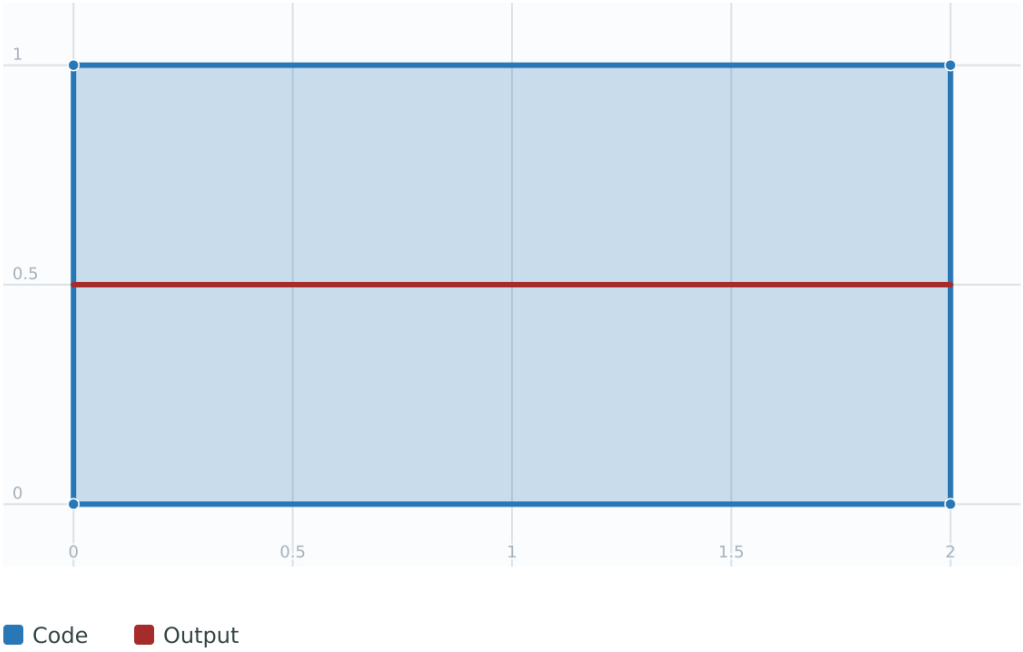
Code

```
SELECT CG_ApproximateMedialAxis('POLYGON((0 0,2 0,2 1,0 1,0 0))', true);
```

Output

```
MULTILINESTRING((0 0.5,0.5 0.5,1.5 0.5,2 0.5))
```

Figure



相关信息

CG_StraightSkeleton, CG_StraightSkeletonPartition

8.3.15 CG_GenerateFlatRoof

CG_GenerateFlatRoof — Generate a flat (box) roof from a footprint polygon.

Synopsis

geometry **CG_GenerateFlatRoof**(geometry geom, float8 height);

描述

Generate a flat (box-shaped) roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. The roof height above the ground plane is given by `height` (default 3.0).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.


☒方法需要 SFCGAL 后端。


☒函数支持多面体曲面。

示例

Generate a flat roof two units above a rectangular footprint.

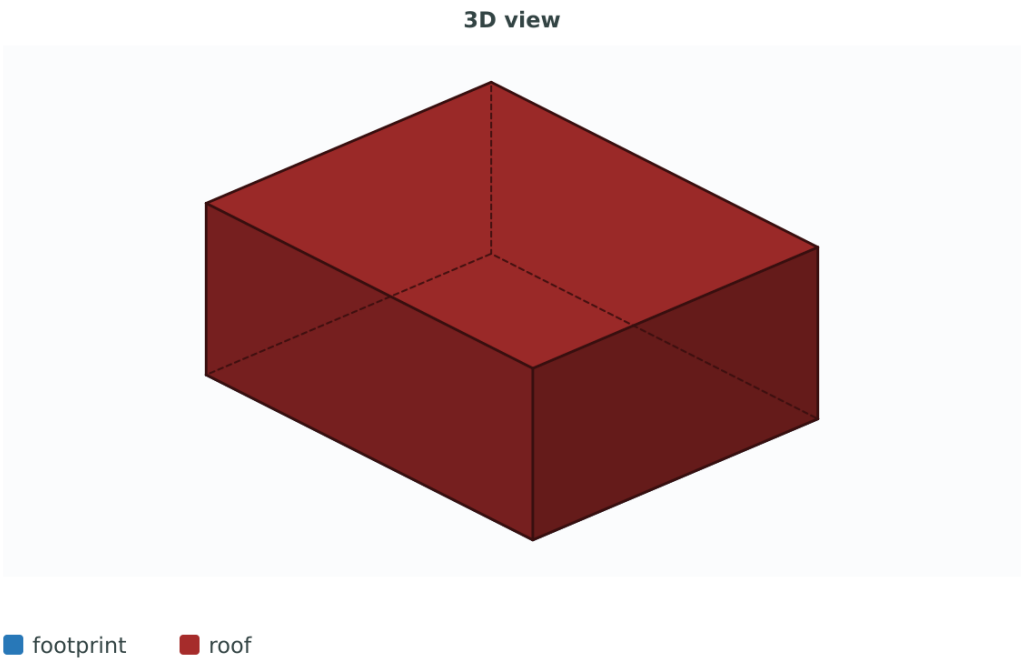
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS footprint
)
SELECT CG_GenerateFlatRoof(footprint, 2.0) AS roof
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((0 0 0,0 4 0,5 4 0,5 0 0,0 0 0)),((0 0 2,5 0 2,5 4 2,0 4 2,0 0 2)),((0 0 0,0 0 2,0 4 2,0 4 0,0 0 ↩
     0)),((0 4 0,0 4 2,5 4 2,5 4 0,0 4 0)),((5 4 0,5 4 2,5 0 2,5 0 0,5 4 0)),((5 0 0,5 0 2,0 0 2,0 0 0,5 0 0)))
```

Figure



相关信息

CG_GenerateHippedRoof, CG_GenerateGableRoof, CG_GenerateSkillionRoof, CG_GenerateRoof

8.3.16 CG_GenerateHippedRoof

CG_GenerateHippedRoof — Generate a hipped roof from a footprint polygon.

Synopsis

geometry **CG_GenerateHippedRoof**(geometry geom, float8 height);

描述

Generate a hipped roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. All sides slope upward to meet at a ridge or apex. The roof height is given by `height` (default 3.0).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

☒方法需要 SFCGAL 后端。

☒函数支持多面体曲面。

示例

The footprint and hipped roof are shown side-by-side so the generated 3D faces do not hide behind the input footprint.

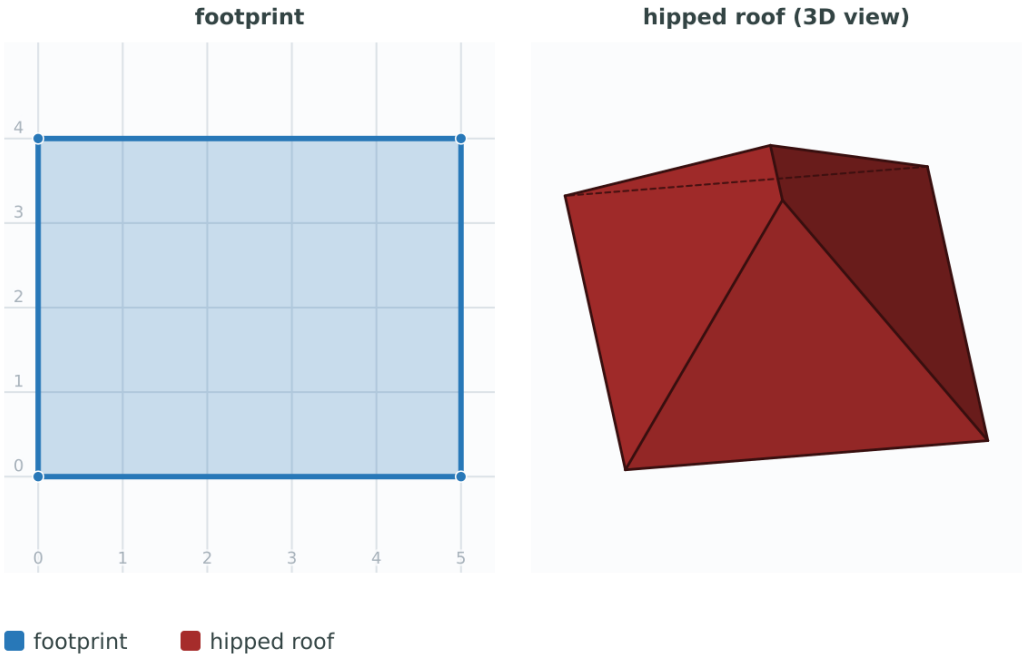
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS input_footprint
)
SELECT input_footprint AS input_footprint,
       CG_GenerateHippedRoof(input_footprint, 2.0) AS hipped_roof
FROM data;
```

Output

```
POLYGON((0 0,5 0,5 4,0 4,0 0)) | POLYHEDRALSURFACE Z (((0 4 0,5 4 0,5 0 0,0 0 0,0 4 0)),((0 4 0,0 0 0,2 2 2,0 4 0)) ↩
    ,((0 0 0,5 0 0,3 2 2,2 2 2,0 0 0)),((5 0 0,5 4 0,3 2 2,5 0 0)),((5 4 0,0 4 0,2 2 2,3 2 2,5 4 0)))
```

Figure



相关信息

CG_GenerateFlatRoof, CG_GenerateGableRoof, CG_GenerateSkillionRoof, CG_GenerateRoof

8.3.17 CG_GenerateGableRoof

CG_GenerateGableRoof — Generate a gable roof from a footprint polygon.

Synopsis

geometry **CG_GenerateGableRoof**(geometry geom, float8 height, float8 slope_angle);

描述

Generate a gable roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. Two opposite sides are vertical gable ends; the other two sides slope to a horizontal ridge. `height` sets the ridge height

above the ground (default 3.0) and `slope_angle` controls the pitch in degrees (default 30.0). The SRID of the input geometry is preserved.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

☒方法需要 SFCGAL 后端。

☒函数支持多面体曲面。

示例

Generate a gable roof using a height of 2 and a 30 degree slope.

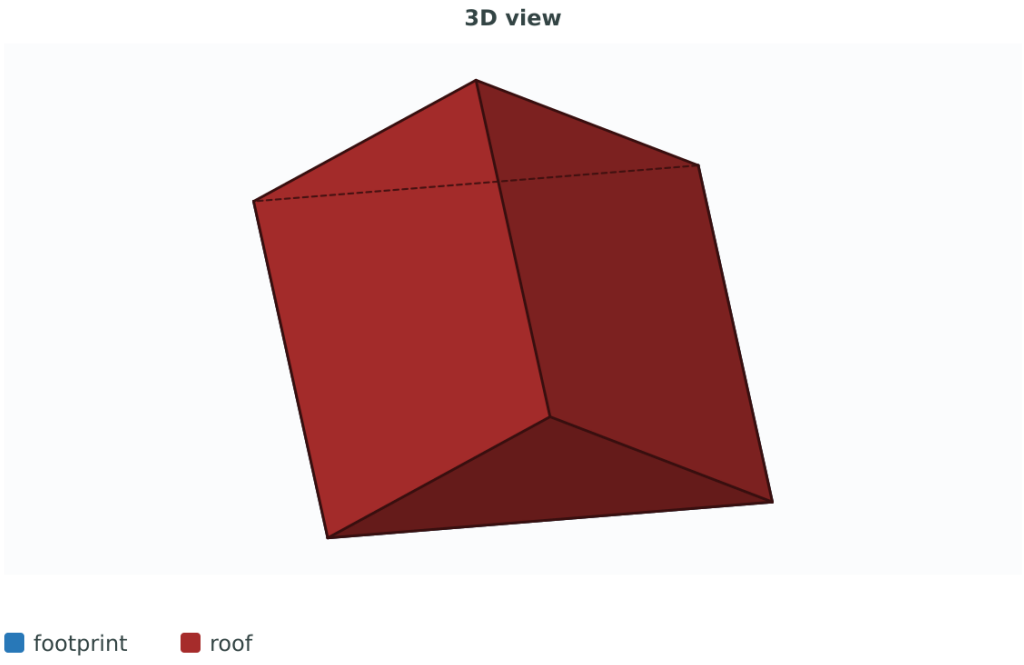
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS footprint
)
SELECT CG_GenerateGableRoof(footprint, 2.0, 30.0) AS roof
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((5 4 0,5 0 0,0 0 0,0 4 0,5 4 0)),((0 4 0,0 0 0,0 2 1.155,0 4 0)),((0 0 0,5 0 0,5 2 1.155,0 2 ↩
    1.155,0 0 0)),((5 0 0,5 4 0,5 2 1.155,5 0 0)),((5 4 0,0 4 0,0 2 1.155,5 2 1.155,5 4 0)))
```

Figure



相关信息

CG_GenerateFlatRoof, CG_GenerateHippedRoof, CG_GenerateSkillionRoof, CG_GenerateRoof

8.3.18 CG_GenerateSkillionRoof

CG_GenerateSkillionRoof — Generate a skillion (single-slope) roof from a footprint polygon.

Synopsis

geometry **CG_GenerateSkillionRoof**(geometry geom, float8 height, float8 slope_angle, integer primary_edge_index);

描述

Generate a skillion (single-slope, shed-style) roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. The entire roof surface slopes in one direction. `height` sets the maximum ridge height (default 3.0), `slope_angle` sets the pitch in degrees (default 30.0), and `primary_edge_index` selects which polygon edge defines the high side (0-based, default 0).

When the combination of `height` and `slope_angle` would cause a side wall to vanish (the slope reaches the far edge before the full height is achieved), that face is omitted from the result. This means the number of output patches can vary depending on the parameters —a lower height relative to the footprint depth produces more faces than a taller, steeper configuration.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

☒方法需要 SFCGAL 后端。

☒函数支持多面体曲面。

示例

Height 2.0 produces 6 faces (all walls present); height 4.0 produces 5 faces (one side wall is filtered out because the slope reaches the far edge before the full height).

Code

```
SELECT
  ST_NumPatches(CG_GenerateSkillionRoof(
    'POLYGON((0 0,5 0,5 4,0 4,0 0))', 2.0)) AS "height 2",
  ST_NumPatches(CG_GenerateSkillionRoof(
    'POLYGON((0 0,5 0,5 4,0 4,0 0))', 4.0)) AS "height 4";
```

Output

```
6 | 5
```

The height 2.0 result is shown with its footprint.

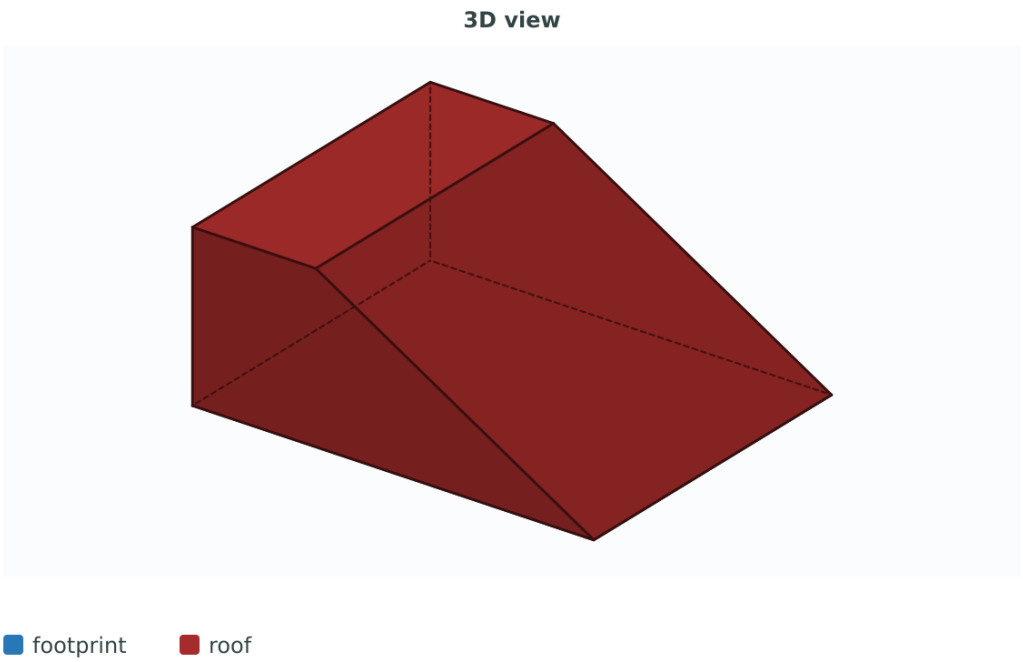
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,5 0,5 4,0 4,0 0))'::geometry AS footprint
)
SELECT CG_GenerateSkillionRoof(footprint, 2.0) AS roof
FROM data;
```

Output

```
POLYHEDRALSURFACE Z (((0 4 0,5 4 0,5 0 0,0 0 0,0 4 0)),((5 0 2,5 4 2,3.464 4 2,3.464 0 2,5 0 2)),((0 4 0,0 0 0,3.464 0 ↩
     2,3.464 4 2,0 4 0)),((0 0 0,5 0 0,5 0 2,3.464 0 2,0 0 0)),((5 0 0,5 4 0,5 4 2,5 0 2,5 0 0)),((5 4 0,0 4 0,3.464  ↩
    4 2,5 4 2,5 4 0)))
```

Figure



相关信息

CG_GenerateFlatRoof, CG_GenerateHippedRoof, CG_GenerateGableRoof, CG_GenerateRoof

8.3.19 CG_GenerateRoof

CG_GenerateRoof — Generate a roof of the requested type from a footprint polygon.

Synopsis

geometry **CG_GenerateRoof**(geometry geom, text roof_type, float8 height, float8 slope_angle, integer primary_edge_index);

描述

Generate a roof as a 3D `PolyhedralSurface Z` from a 2D footprint polygon. The `roof_type` parameter selects the roof style (default `'HIPPED'`):

- `FLAT` —flat box roof.
- `HIPPED` —hipped roof (all sides slope to apex).
- `GABLE` —gable roof (two sloping sides, two vertical ends).
- `SKILLION` —single-slope (shed) roof.

`height` sets the roof height (default 3.0), `slope_angle` sets the pitch in degrees (default 30.0), and `primary_edge_index` selects the reference edge for skillion roofs (default 0).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

☒方法需要 SFCGAL 后端。

☒函数支持多面体曲面。

示例

The generic dispatcher returns the same 3D roofs as the specialized helpers for the same footprint.

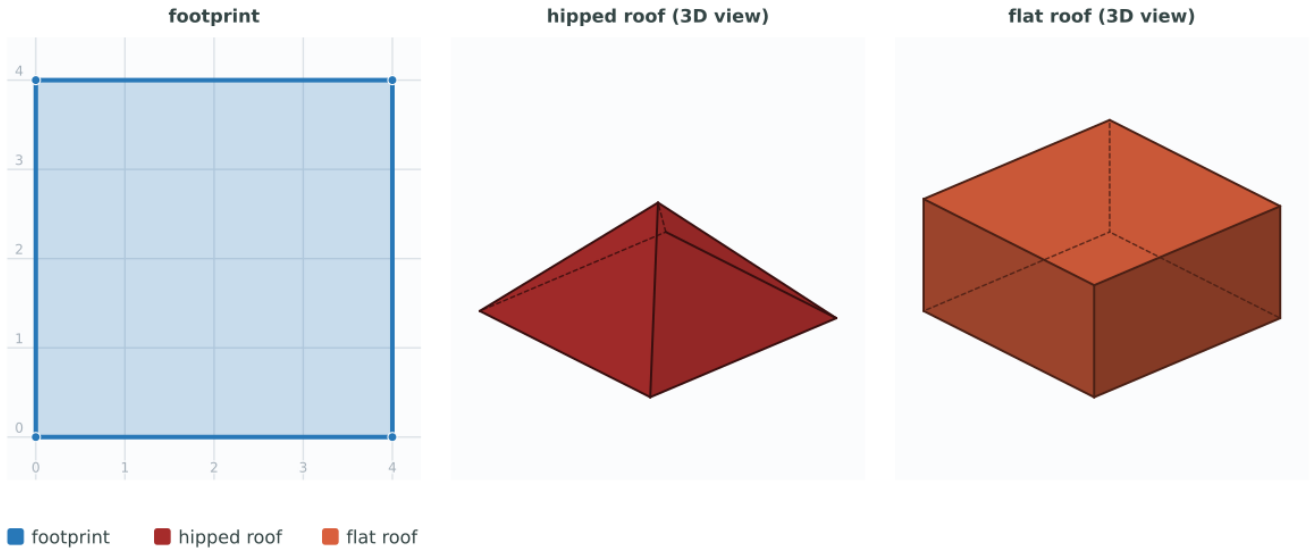
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,4 0,4 4,0 4,0 0))'::geometry AS input_footprint
)
SELECT input_footprint AS input_footprint,
       CG_GenerateRoof(input_footprint, 'HIPPED', 2.0, 30.0, 0) AS hipped_roof,
       CG_GenerateRoof(input_footprint, 'FLAT', 2.0, 30.0, 0) AS flat_roof
FROM data;
```

Output

```
POLYGON((0 0,4 0,4 4,0 4,0 0)) | POLYHEDRALSURFACE Z (((0 4 0,4 4 0,4 0 0,0 0 0,0 4 0)),((0 4 0,0 0 0,2 2 2,0 4 0)) ↩
    ,((0 0 0,4 0 0,2 2 2,0 0 0)),((4 0 0,4 4 0,2 2 2,4 0 0)),((4 4 0,0 4 0,2 2 2,4 4 0))) | POLYHEDRALSURFACE Z (((0  ↩
    0 0,0 4 0,4 4 0,4 0 0,0 0 0)),((0 0 2,4 0 2,4 4 2,0 4 2,0 0 2)),((0 0 0,0 0 2,0 4 2,0 4 0,0 0 0)),((0 4 0,0 4 2,4 ↩
     4 2,4 4 0,0 4 0)),((4 4 0,4 4 2,4 0 2,4 0 0,4 4 0)),((4 0 0,4 0 2,0 0 2,0 0 0,4 0 0)))
```

Figure



相关信息

CG_GenerateFlatRoof, CG_GenerateHippedRoof, CG_GenerateGableRoof, CG_GenerateSkillionRoof

8.3.20 CG_PolygonRepair

CG_PolygonRepair — Repair an invalid polygon or multipolygon.

Synopsis

geometry **CG_PolygonRepair**(geometry geom, text rule);

描述

Repair an invalid polygon or multipolygon using CGAL's 2D Polygon Repair algorithm. Returns a `MultiPolygon`.

The `rule` parameter controls the repair strategy:

- `EVEN_ODD` (default) —areas covered an odd number of times are kept. Available with CGAL 6.0+.
- `NON_ZERO` —areas with non-zero winding number are kept. Requires CGAL 6.1+.
- `UNION` —union of all input polygons. Requires CGAL 6.1+.
- `INTERSECTION` —intersection of all input polygons. Requires CGAL 6.1+.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0 and CGAL >= 6.0.


☒方法需要 SFCGAL 后端。


☒函数支持多面体曲面。

示例

Repair a self-intersecting bowtie polygon with the default `EVEN_ODD` rule.

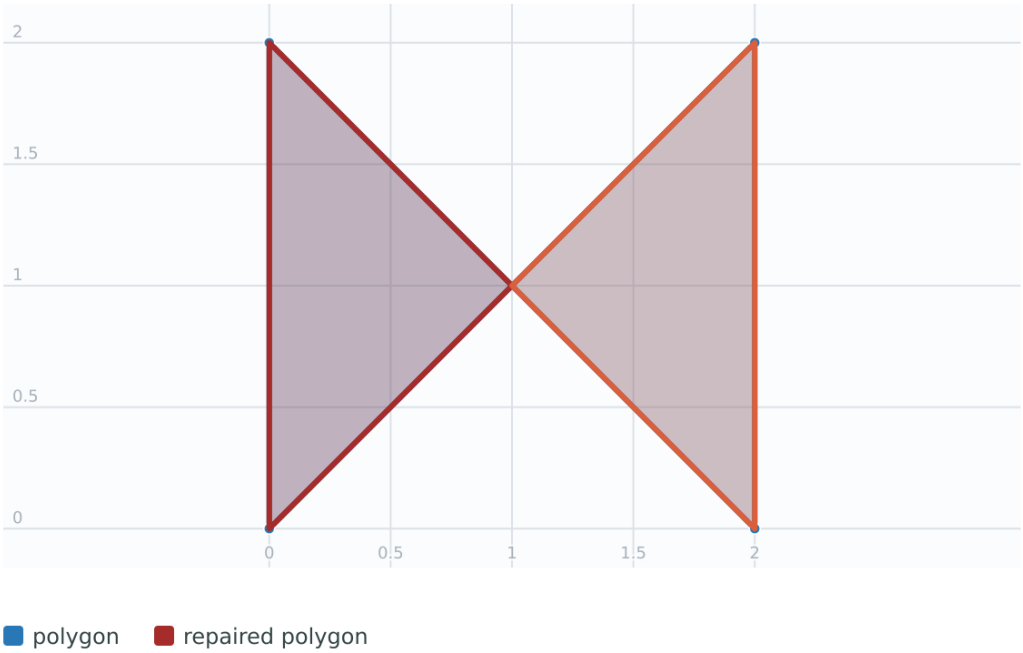
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,2 2,2 0,0 2,0 0))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_PolygonRepair(input_polygon) AS repaired_polygon
FROM data;
```

Output

```
POLYGON((0 0,2 2,2 0,0 2,0 0)) | MULTIPOLYGON(((0 0,1 1,0 2,0 0)),((1 1,2 0,2 2,1 1)))
```

Figure



With the `UNION` rule, all winding area is kept. This rule requires CGAL 6.1 or later.

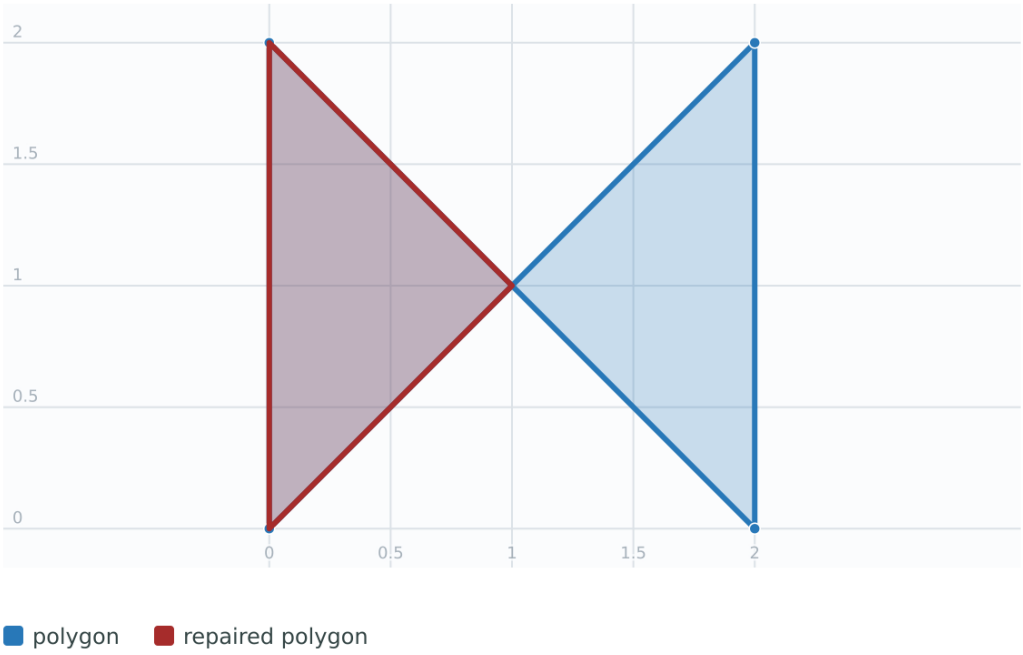
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,2 2,2 0,0 2,0 0))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_PolygonRepair(input_polygon, 'UNION') AS repaired_polygon
FROM data;
```

Output

```
POLYGON((0 0,2 2,2 0,0 2,0 0)) | MULTIPOLYGON(((0 0,1 1,0 2,0 0)))
```

Figure



相关信息

ST_IsValid, ST_MakeValid

8.3.21 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。

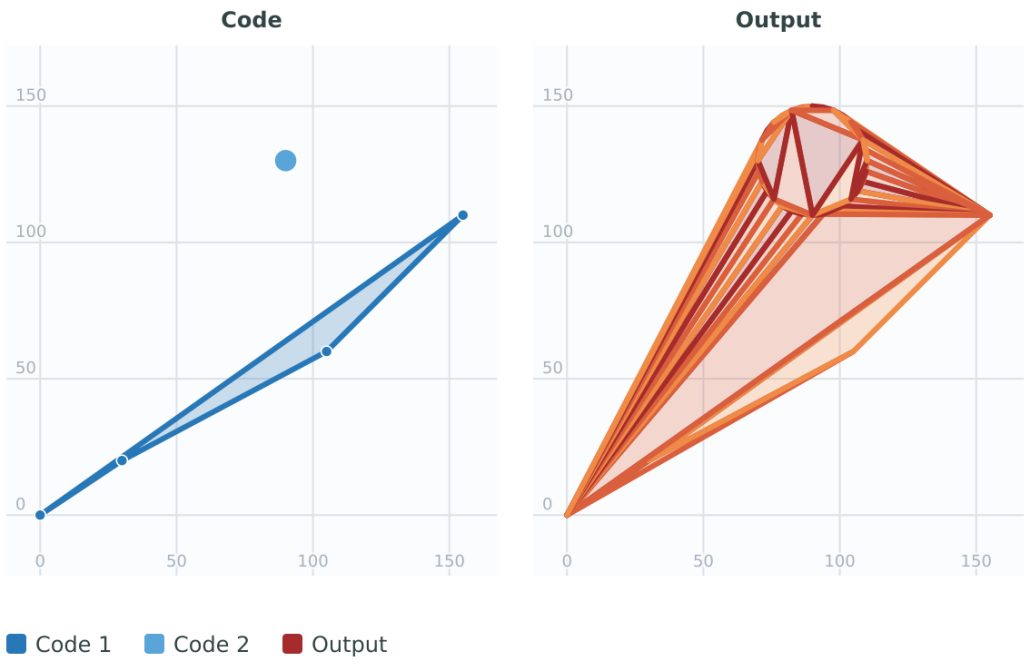
示例

Constrained Delaunay triangulation of the polygon and buffered point.

Code

```
SELECT
  CG_ConstrainedDelaunayTriangles(
    ST_Union(
      'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry,
      ST_Buffer('POINT(110 170)'::geometry, 20)
    )
  );
```

Figure

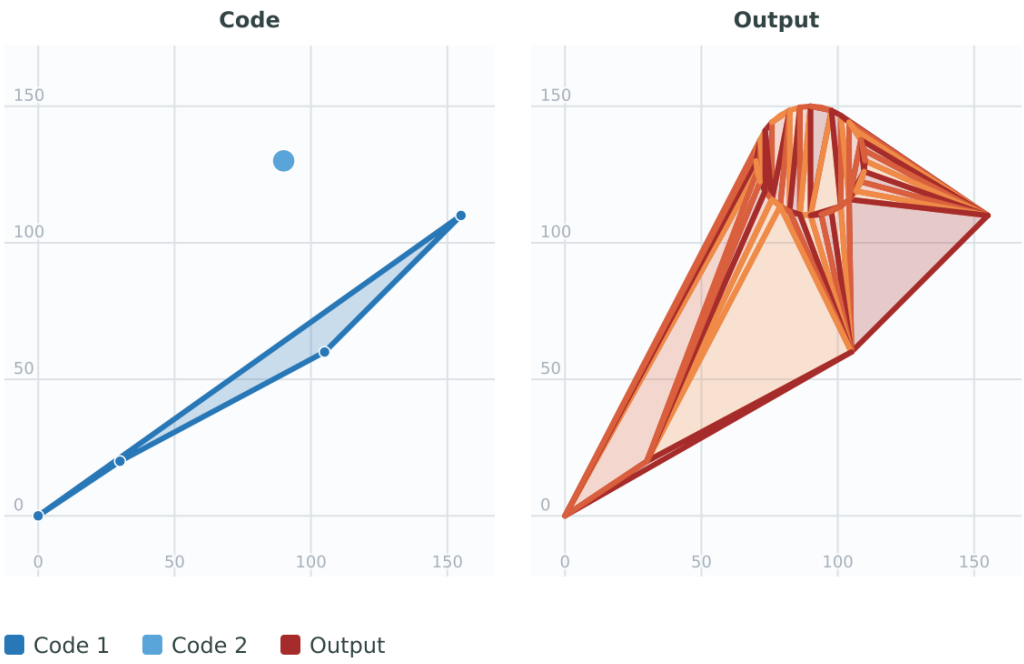


ST_DelaunayTriangles applied to the same input. Triangle edges cross polygon boundaries.

Code

```
SELECT
  ST_DelaunayTriangles(
    ST_Union(
      'POLYGON((175 150,20 40,50 60,125 100,175 150))'::geometry,
      ST_Buffer('POINT(110 170)'::geometry, 20)
    )
  );
```

Figure



相关信息

ST_DelaunayTriangles, ST_TriangulatePolygon, CG_Tesselate, ST_ConcaveHull, ST_Dump

8.3.22 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

Extrudes the input along the vector (*x*, *y*, *z*) and connects it to its translated copy. Extrusion can raise the dimensionality of the result, for example from a surface to a volume.

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

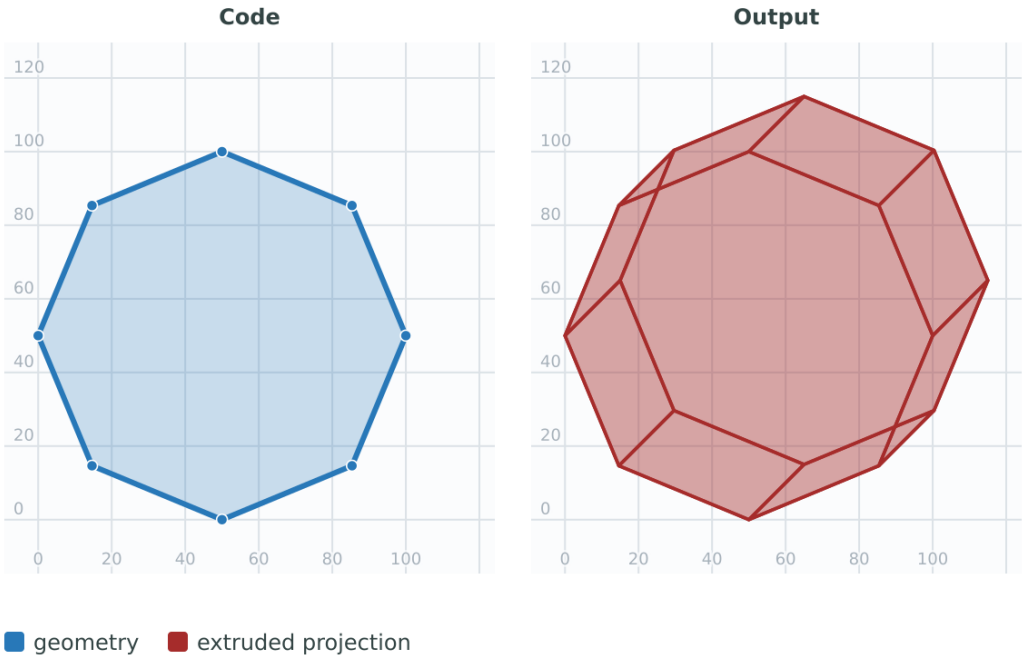
The figures use ST_Affine to project Z diagonally into XY after extrusion. The returned geometry from CG_Extrude remains a 3D PolyhedralSurface.

Extruding an octagonal Polygon 30 units along Z.

Code

```
WITH data AS (
  SELECT ST_Buffer('POINT(100 90)'::geometry, 50, 'quad_segs=2') AS geom
), projected AS (
  SELECT ST_Force2D(ST_Affine(geom,
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS input_geom,
         ST_Force2D(ST_Affine(CG_Extrude(geom, 0, 0, 30),
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS output_geom
  FROM data
)
SELECT input_geom AS input_geometry,
       output_geom AS extruded_projection
FROM projected;
```

Figure



Extruding a LineString 10 units along Z.

Code

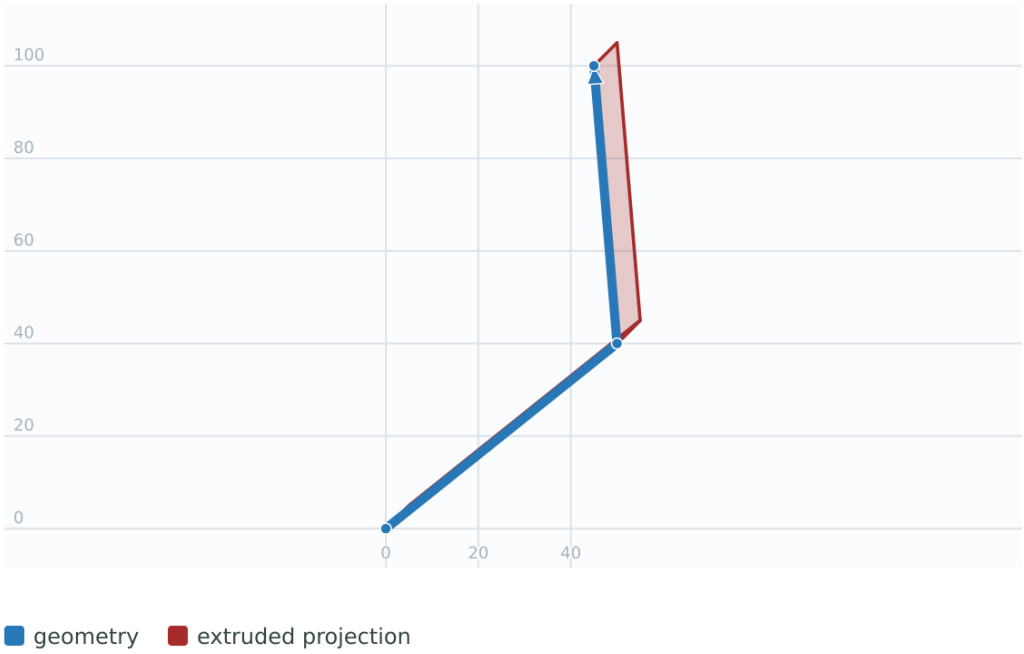
```
WITH data AS (
  SELECT 'LINESTRING(50 50,100 90,95 150)'::geometry AS geom
), projected AS (
  SELECT ST_Force2D(ST_Affine(geom,
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS input_geom,
         ST_Force2D(ST_Affine(CG_Extrude(geom, 0, 0, 10),
           1, 0, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0)) AS output_geom
  FROM data
)
SELECT input_geom AS input_geometry,
```

```
       output_geom AS extruded_projection
FROM projected;
```

Output

```
LINESTRING(50 50,100 90,95 150) | POLYHEDRALSURFACE(((50 50,100 90,105 95,55 55,50 50)),((100 90,95 150,100 155,105 ↩
    95,100 90)))
```

Figure



相关信息

ST_AsX3D, CG_ExtrudeStraightSkeleton

8.3.23 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

直⌧骨架的（⌧史上的）第一个使用例子可能是通⌧⌧定一个多⌧形屋⌧，使用直⌧骨架方法来得出⌧篷的布局。骨架的每个⌧都被从平面上升一定的偏移距离，确保水⌧是会落在⌧廓的⌧⌧（即屋⌧的⌧界⌧），无⌧水从屋⌧的任何地方落下，⌧⌧⌧果的屋⌧就可以⌧是“正确的”。⌧函数⌧算⌧个多⌧形的⌧⌧，⌧名“屋⌧”。如果参数 body_height 大于 0，⌧多⌧形将通⌧ CG_Extrude(polygon, 0, 0, body_height) 来⌧行⌧⌧。⌧果是⌧些多面体表面的⌧合。

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。

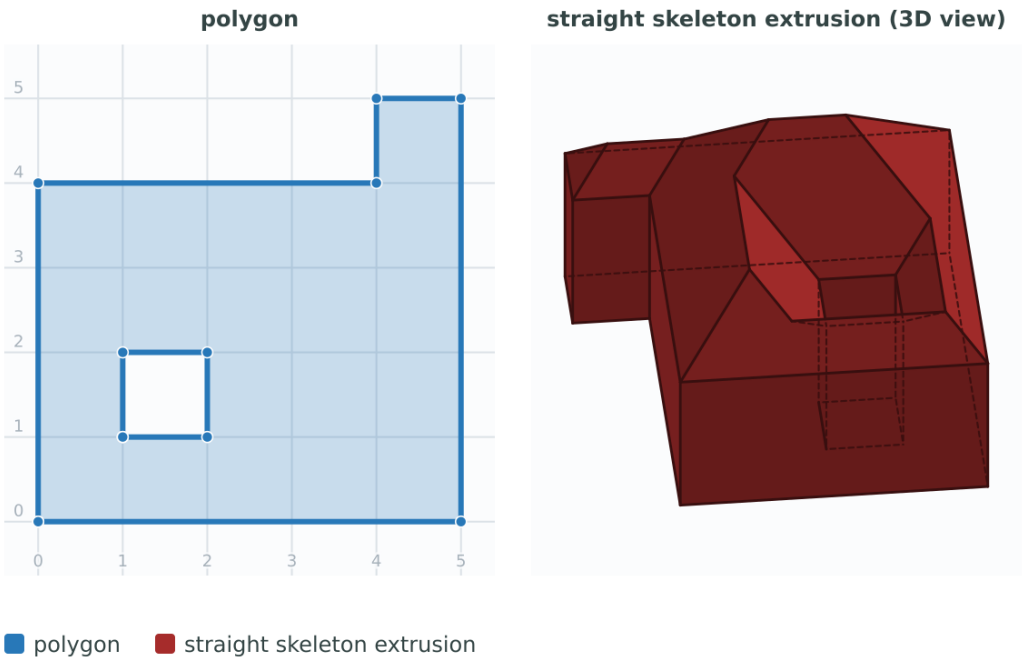
示例

Extrude a polygon with a hole into a body and straight-skeleton roof.

Code

```
WITH data AS (
  SELECT 'POLYGON ((0 0,5 0,5 5,4 5,4 4,0 4,0 0),(1 1,1 2,2 2,2 1,1 1))'::geometry AS ↩
    input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_ExtrudeStraightSkeleton(input_polygon, 3.0, 2.0) AS straight_skeleton_extrusion
FROM data;
```

Figure



相关信息

CG_Extrude, CG_ExtrudeStraightSkeleton, CG_StraightSkeleton, CG_StraightSkeletonPartition

8.3.24 CG_GreeneApproxConvexPartition

CG_GreeneApproxConvexPartition — 计算多边形几何形的近似凸分割

Synopsis

geometry **CG_GreeneApproxConvexPartition**(geometry geom);

描述

计算多边形几何形的近似边界凸分割。



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成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

Greene’s monotone-based approximation uses the same polygon and stays directly comparable in one panel.

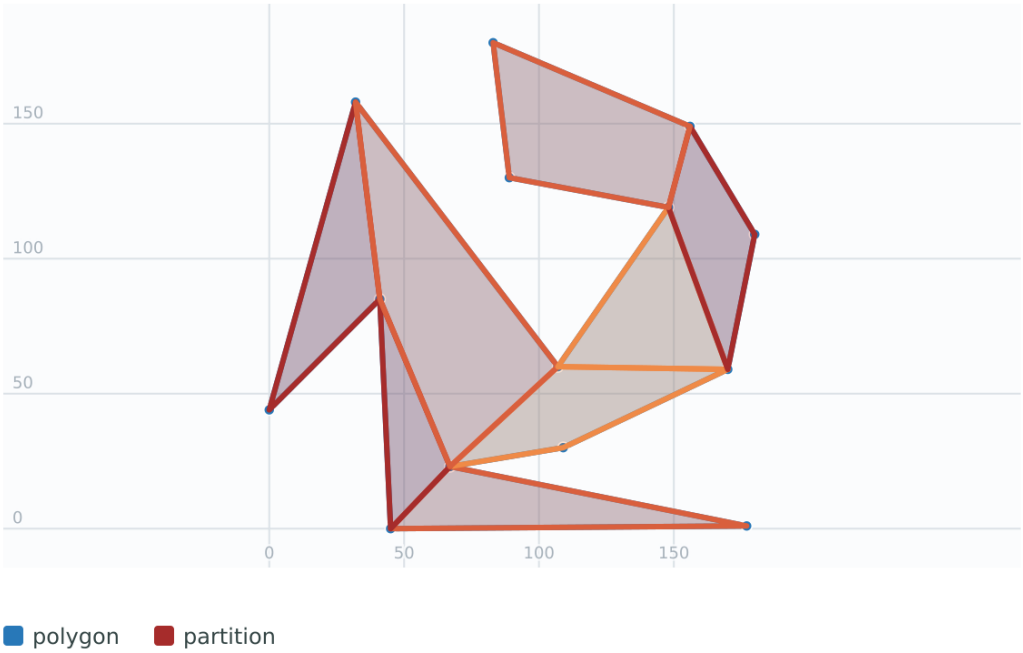
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_GreeneApproxConvexPartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) | ↩
    GEOMETRYCOLLECTION(POLYGON((32 159,0 45,41 86,32 159)), POLYGON((45 1,177 2,67 24,45 1)), POLYGON((67 24,109 ↩
    31,170 60,107 61,67 24)), POLYGON((41 86,45 1,67 24,41 86)), POLYGON((107 61,32 159,41 86,67 24,107 61)), POLYGON ↩
    ((148 120,107 61,170 60,148 120)), POLYGON((148 120,170 60,180 110,156 150,148 120)), POLYGON((156 150,83 181,89 ↩
    131,148 120,156 150)))
```

Figure



相关信息

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_OptimalConvexPartition

8.3.25 CG_MinkowskiSum

CG_MinkowskiSum — 区行 Minkowski sum

Synopsis

geometry **CG_MinkowskiSum**(geometry geom1, geometry geom2);

描述

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：3.5.0

区方法需要 SFCGAL 后端。

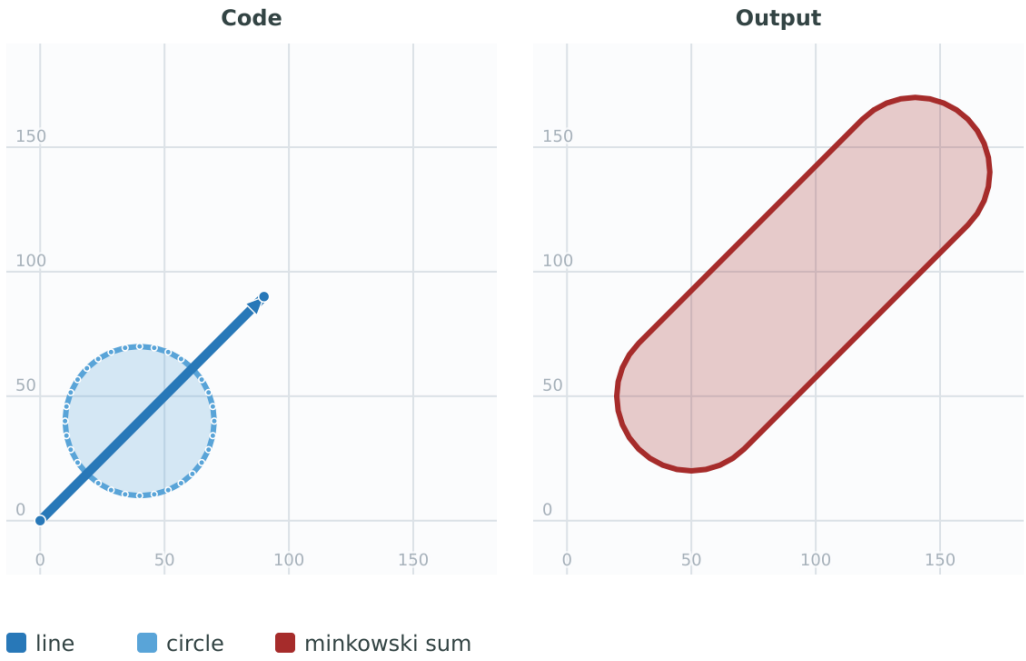
示例

Minkowski sum of a LineString and a circular polygon. The line crosses the polygon, and the result sweeps the polygon along the line.

Code

```
WITH data AS (
  SELECT ST_MakeLine(ST_Point(10, 10), ST_Point(100, 100)) AS line,
         ST_Buffer('POINT(50 50)'::geometry, 30) AS circle
)
SELECT line AS input_line,
       circle AS input_circle,
       CG_MinkowskiSum(line, circle) AS minkowski_sum
FROM data;
```

Figure



Minkowski sum of a MultiPoint and a polygon. The result contains one translated copy of the polygon for each point.

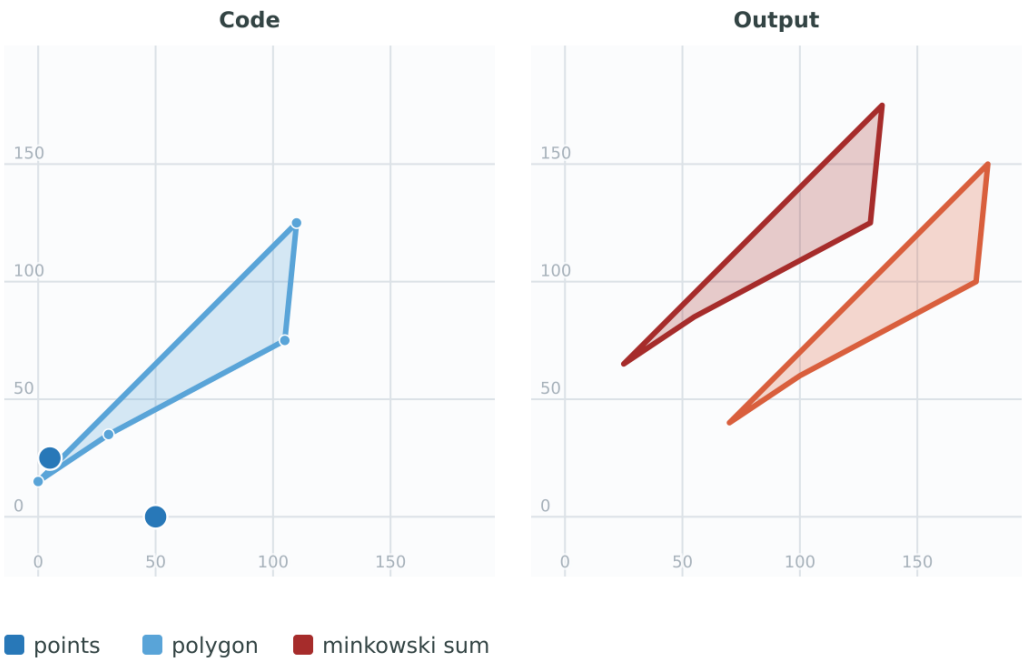
Code

```
WITH data AS (
  SELECT 'MULTIPOINT(25 50,70 25)'::geometry AS points,
         'POLYGON((130 150,20 40,50 60,125 100,130 150))'::geometry AS polygon
)
SELECT points AS input_points,
       polygon AS input_polygon,
       CG_MinkowskiSum(points, polygon) AS minkowski_sum
FROM data;
```

Output

```
MULTIPOINT((25 50),(70 25)) | POLYGON((130 150,20 40,50 60,125 100,130 150)) | MULTIPOLYGON(((45 90,75 110,150 150,155 ↩
      200,45 90)),((90 65,120 85,195 125,200 175,90 65)))
```

Figure



8.3.26 CG_OptimalAlphaShape

CG_OptimalAlphaShape — 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。

可用性：3.5.0 - 需要 SFCGAL>= 1.4.1。

☒方法需要 SFCGAL 后端。

示例

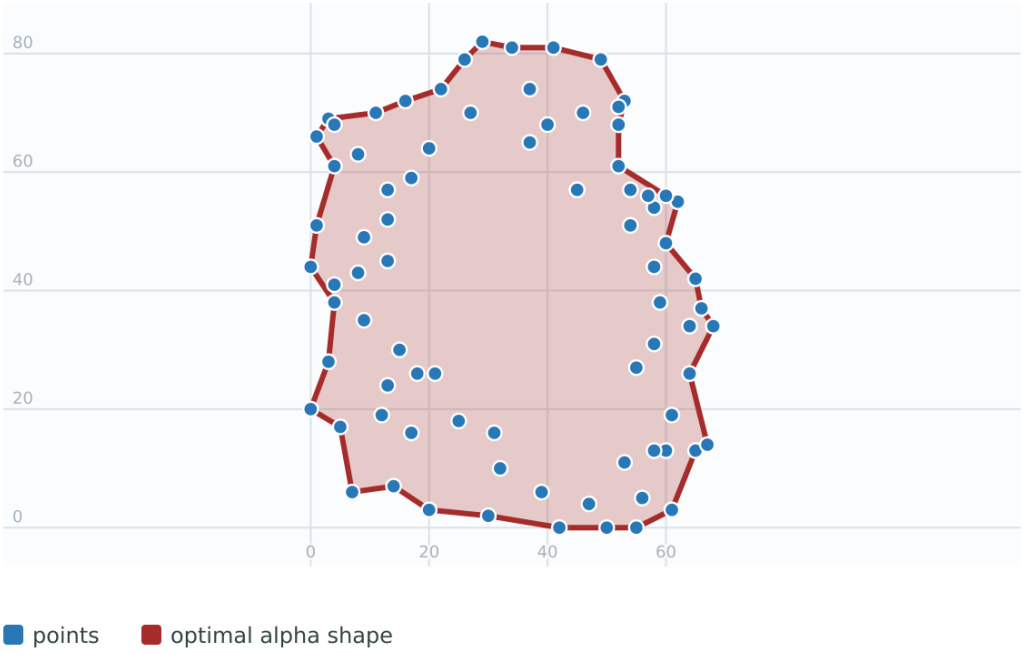
The original point cloud and the optimal alpha-shape share one comparison frame.

Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),
(88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 ↩
    65),
(81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
    98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
    88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
    90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
    59),(34 86),(45 90),(64 97))'::geometry AS input_points
)
SELECT input_points AS input_points,

        CG_OptimalAlphaShape(input_points) AS optimal_alpha_shape
FROM data;
```

Figure



Allowing holes keeps the same outer shell but carves a central hole.

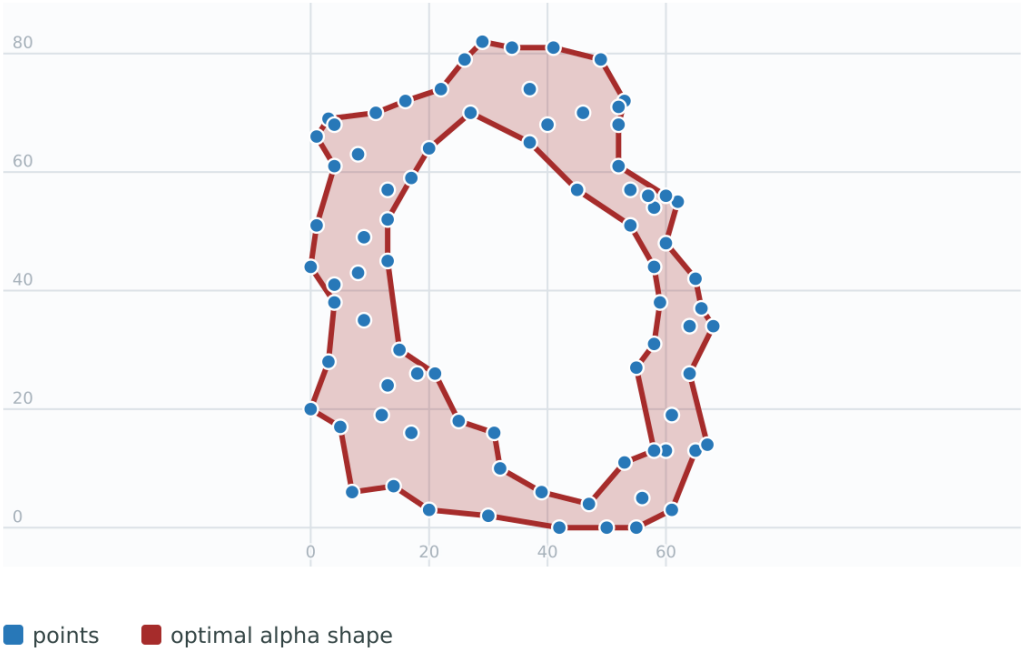
Code

```
WITH data AS (
  SELECT 'MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),(88 29),(24 82),(32  ↩
    51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61),(32 65),(81 47),(88  ↩
    58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),(27 84),(52 ↩
    98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),(27 77),(39 ↩
    88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),(69 86),(60 ↩
    90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),(38 46),(31 ↩
    59),(34 86),(45 90),(64 97))'::geometry AS input_points
```

```
)
SELECT input_points AS input_points,

         CG_OptimalAlphaShape(input_points, allow_holes => true) AS optimal_alpha_shape
FROM data;
```

Figure



相关信息

ST_ConcaveHull, CG_AlphaShape

8.3.27 CG_OptimalConvexPartition

CG_OptimalConvexPartition — 计算多边形几何形的最小凸分割

Synopsis

geometry **CG_OptimalConvexPartition**(geometry geom);

描述

计算多边形几何形的最小凸分割。

> **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_OptimalConvexPartition 生成的划分在片段数量上是最小的。

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

 区方法需要 SFCGAL 后端。

示例

The optimal convex partition uses the same polygon and produces the smallest documented piece count.

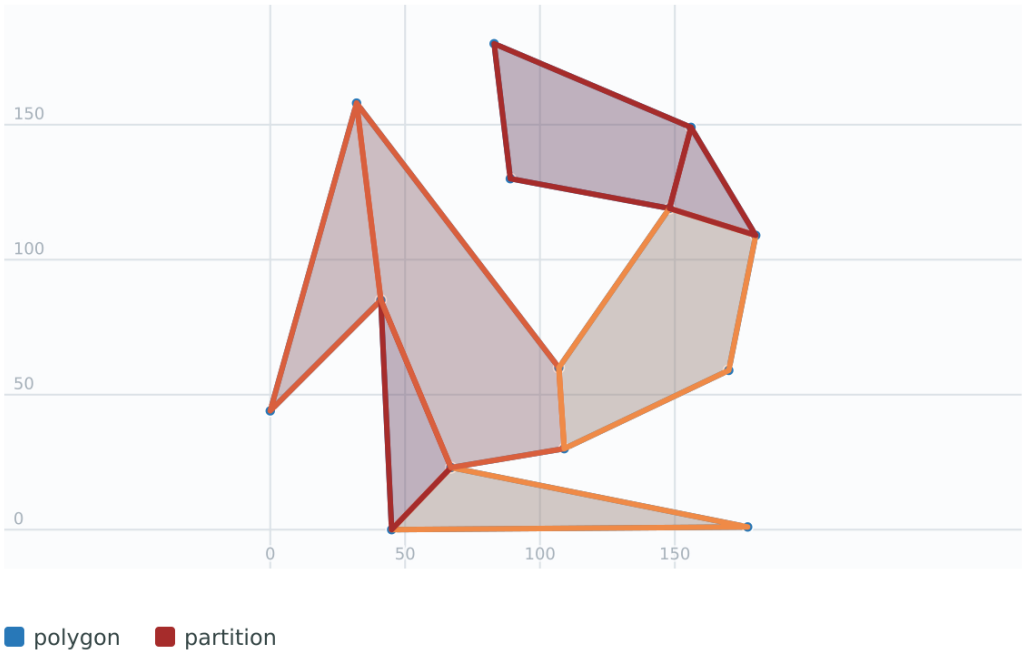
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_OptimalConvexPartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) | ↩
    GEOMETRYCOLLECTION(POLYGON((156 150,83 181,89 131,148 120,156 150)), POLYGON((32 159,0 45,41 86,32 159)), POLYGON ↩
    ((45 1,177 2,67 24,45 1)), POLYGON((41 86,45 1,67 24,41 86)), POLYGON((107 61,32 159,41 86,67 24,109 31,107 61)), ↩
     POLYGON((148 120,107 61,109 31,170 60,180 110,148 120)), POLYGON((156 150,148 120,180 110,156 150)))
```

Figure



相关信息

CG_YMonotonePartition, CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition

8.3.28 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

Computes the straight skeleton of a polygon by tracing the inward propagation of its edges. When *use_distance_as_m* is `true`, the M coordinate of each output point stores its distance from the input boundary.

可用性：3.5.0

要使用任意参数 use_distance_as_m，需要 SFCGAL >= 1.3.8

> **Note**
> 此函数忽略 Z ⊠度。即使在 3D 几何体上使用⊠，它也始⊠⊠出 2D ⊠果。

⊠方法需要 SFCGAL 后端。

⊠函数支持多面体曲面。

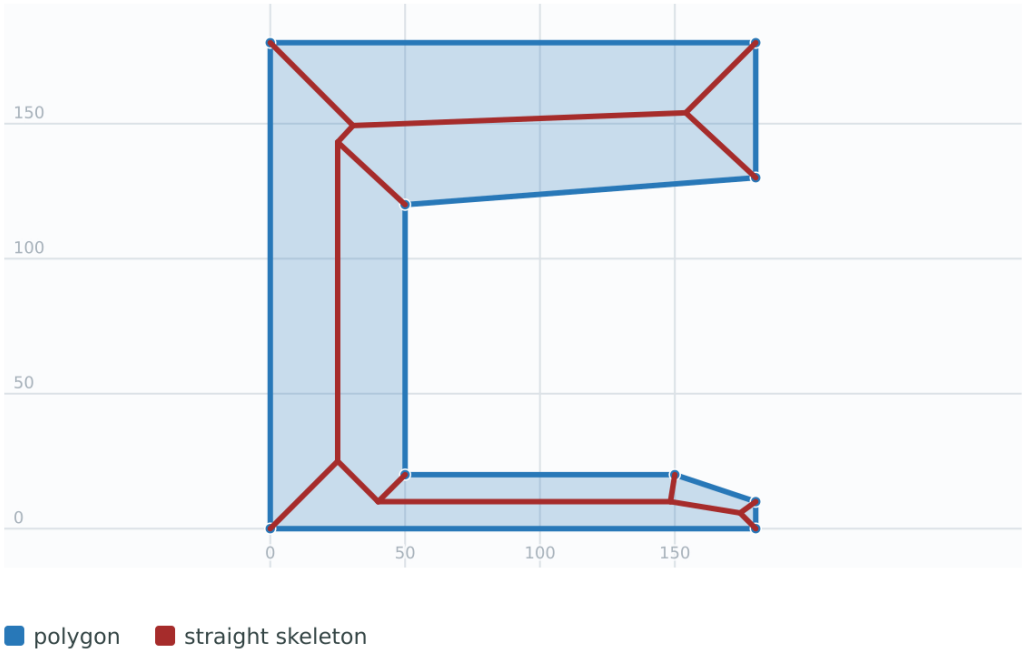
此函数支持三角形和不⊠⊠三角网面 (TIN)。

示例

Code

```
WITH data AS (
  SELECT 'POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190)) ↩
    '::geometry AS polygon
)
SELECT CG_StraightSkeleton(polygon) AS straight_skeleton
FROM data;
```

Figure



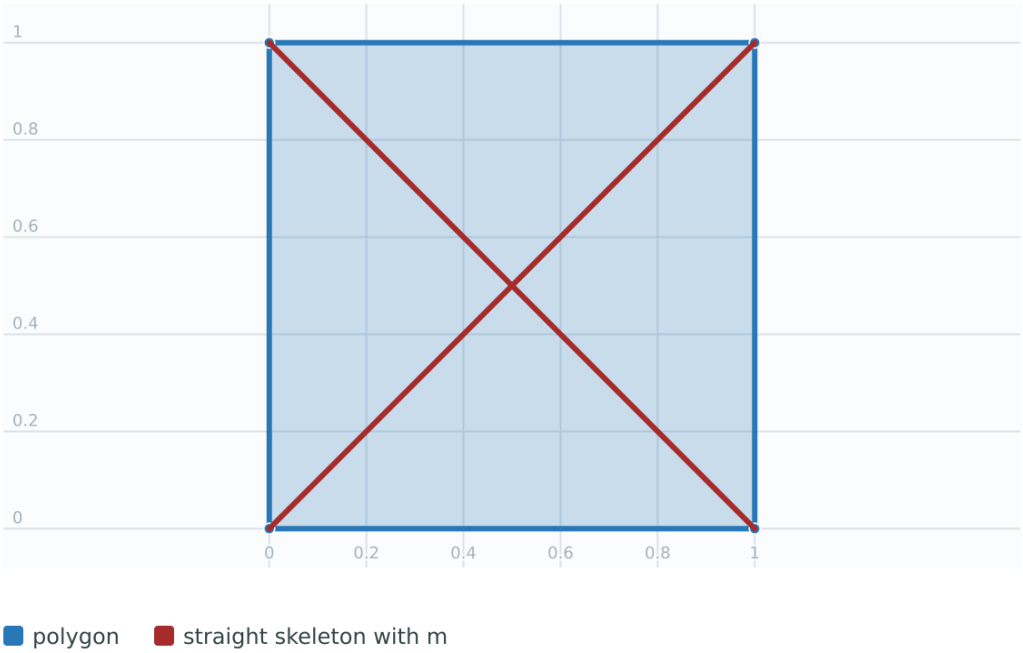
Code

```
WITH data AS (
  SELECT 'POLYGON((0 0,1 0,1 1,0 1,0 0))'::geometry AS polygon
)
SELECT CG_StraightSkeleton(polygon, true) AS straight_skeleton_with_m
FROM data;
```

Output

```
MULTILINESTRING M ((0 0 0,0.5 0.5 0.5),(1 0 0,0.5 0.5 0.5),(1 1 0,0.5 0.5 0.5),(0 1 0,0.5 0.5 0.5))
```

Figure



⊠注意，若合法⊠入中的⊠⊠在⊠个点⊠相接，将会引⊠⊠⊠。

Code

```
SELECT CG_StraightSkeleton('POLYGON((0 0,3 0,3 3,0 3,0 0),(0 0,1 2,2 1,0 0))'));
```

Output

```
NOTICE:  During straight_skeleton(A) :
NOTICE:    with A: POLYGON((0/1 0/1,3/1 0/1,3/1 3/1,0/1 3/1,0/1 0/1),(0/1 0/1,1/1 2/1,2/1 1/1,0/1 0/1))
ERROR:  straight skeleton of Polygon with point touching rings is not implemented.
```

相关信息

CG_StraightSkeletonPartition, CG_ExtrudeStraightSkeleton

8.3.29 CG_Tesselate

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

Synopsis

geometry **CG_Tesselate**(geometry geom);

描述

将 MULTI(POLYGON) 或 POLYHEDRALSURFACE 等曲面作⊠⊠入，并通⊠使用三角形的⊠分⊠程返回 TIN 表示形式。

Note
ST_TriangulatePolygon 与此函数☒似，但返回一个多☒形的几何集合而不是三角形网，并且☒适用于二☒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

Tessellation divides each face of a cube into triangles.

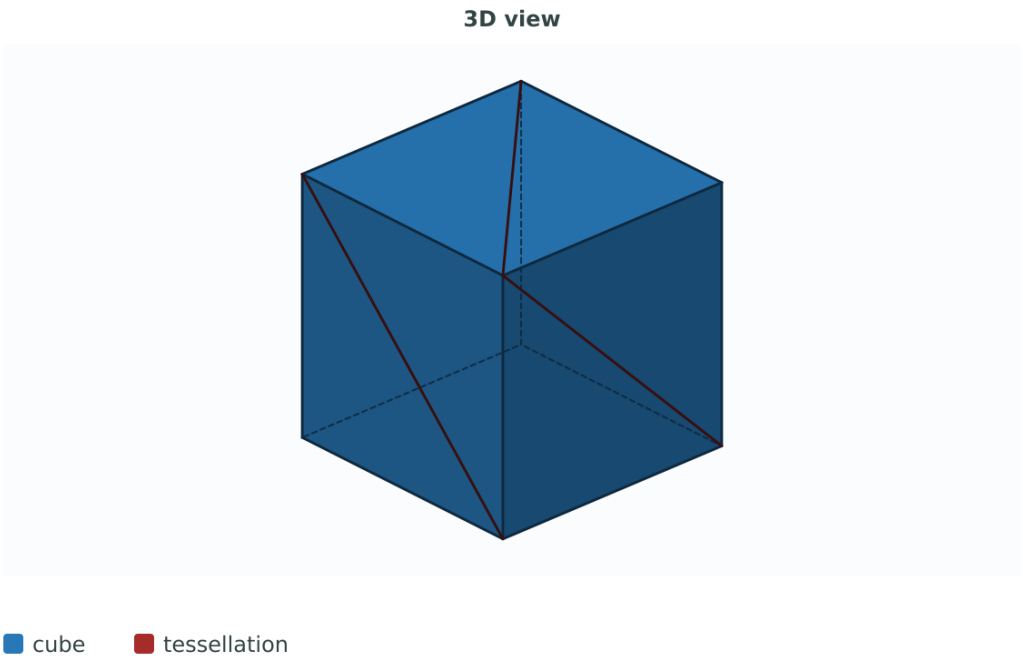
Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z( ((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)) )'::geometry AS cube
)
SELECT CG_Tesselate(cube) AS tessellation
FROM data;
```

Output

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

Figure



Tessellation of a Polygon.

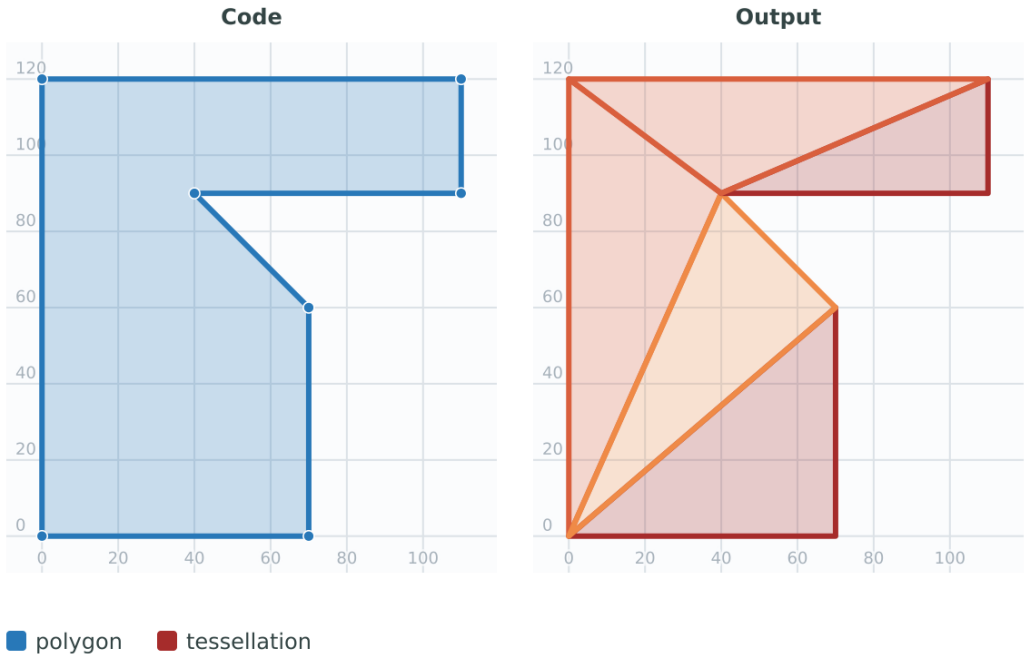
Code

```
WITH data AS (
  SELECT 'POLYGON((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190))'::geometry AS
    polygon
)
SELECT polygon AS input_polygon,
       CG_Tesselate(polygon) AS tessellation
FROM data;
```

Output

```
POLYGON((10 190,10 70,80 70,80 130,50 160,120 160,120 190,10 190)) | TIN(((80 130,10 70,80 70,80 130)),((50 160,10
    190,10 70,50 160)),((80 130,50 160,10 70,80 130)),((120 160,120 190,50 160,120 160)),((120 190,10 190,50 160,120
    190)))
```

Figure



相关信息

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.30 CG_Triangulate

CG_Triangulate — 区多区形几何体区行三角剖分

Synopsis

geometry **CG_Triangulate**(geometry geom);

描述

☒多☒形几何体☒行三角剖分。
由 SFCGAL 模☒☒行



Note
注意：此函数返回表示三角剖分☒果的几何☒象。

可用性：3.5.0



☒方法需要 SFCGAL 后端。

几何示例

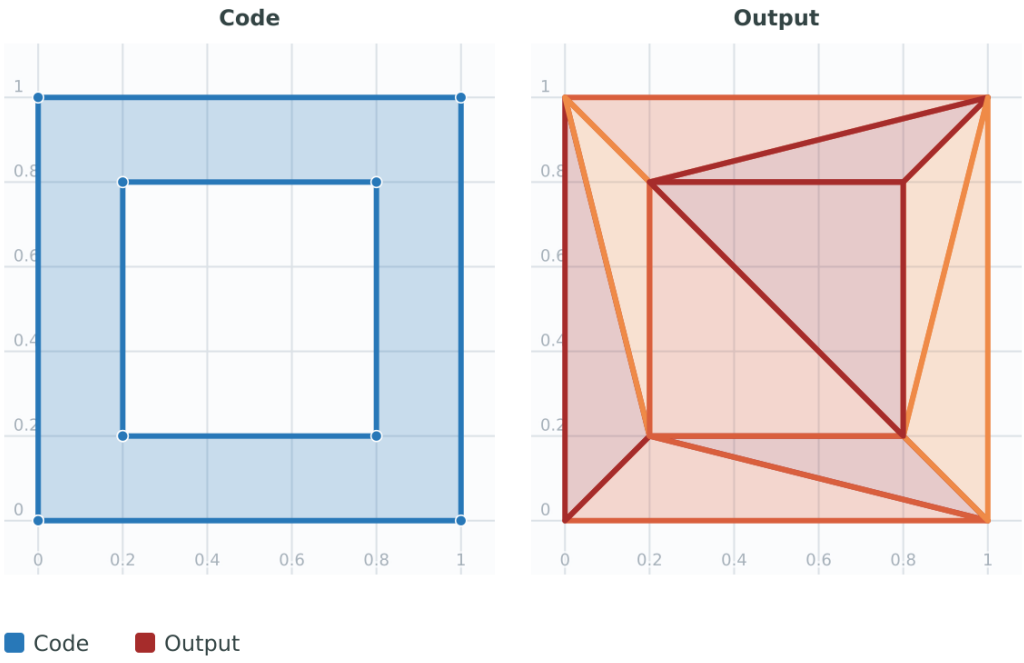
Code

```
SELECT CG_Triangulate(
  'POLYGON((0 0,1 0,1 1,0 1,0 0),
            (0.2 0.2,0.2 0.8,0.8 0.8,0.8 0.2,0.2 0.2))'
);
```

Output

```
TIN(((0.8 0.2,0.2 0.2,1 0,0.8 0.2)),((0.2 0.2,0 0,1 0,0.2 0.2)),((1 1,0.8 0.8,0.8 0.2,1 1)),((0 1,0 0,0.2 0.2,0 1)) ↩
    ,((0 1,0.2 0.8,1 1,0 1)),((0 1,0.2 0.2,0.2 0.8,0 1)),((0.2 0.8,0.8 0.8,1 1,0.2 0.8)),((0.2 0.8,0.2 0.2,0.8 ↩
    0.2,0.2 0.8)),((1 1,0.8 0.2,1 0,1 1)),((0.8 0.8,0.2 0.8,0.8 0.2,0.8 0.8)))
```

Figure



相关信息

CG_ConstrainedDelaunayTriangles, ST_DelaunayTriangles, ST_TriangulatePolygon

8.3.31 CG_Visibility

CG_Visibility — 计算一个从点或多边形几何中的线段生成的可见性多边形

Synopsis

geometry **CG_Visibility**(geometry polygon, geometry point);
geometry **CG_Visibility**(geometry polygon, geometry pointA, geometry pointB);

描述

Returns the portion of a polygon visible from an observer point, or from the observer segment defined by *`pointA`* and *`pointB`*.

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。

此函数支持 3d 并且不会丢失 z-index。

此函数支持多面体曲面。

此函数支持三角形和不规则三角网面 (TIN)。

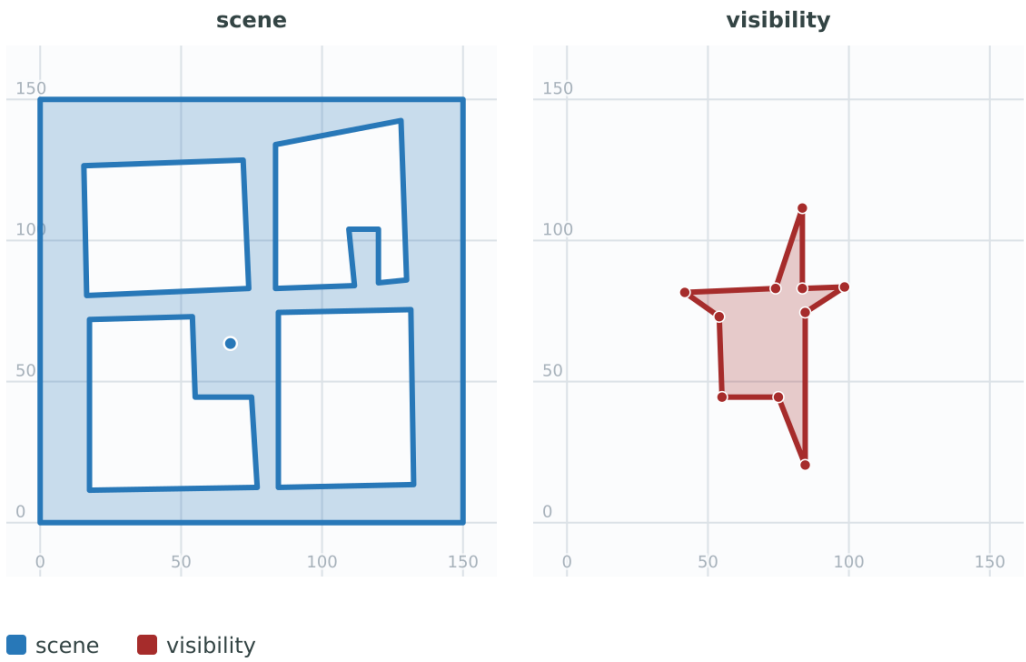
示例

Visibility from a point inside a polygon with holes.

Code

```
WITH data AS (
  SELECT 'POLYGON((23.5 23.5,23.5 173.5,173.5 173.5,173.5 23.5,23.5 23.5),(108 98,108 ↩
    36,156 37,155 99,108 98),(107 157.5,107 106.5,135 107.5,133 127.5,143.5 127.5,143.5 ↩
    108.5,153.5 109.5,151.5 166,107 157.5),(41 95.5,41 35,100.5 36,98.5 68,78.5 68,77.5 ↩
    96.5,41 95.5),(39 150,40 104,97.5 106.5,95.5 152,39 150))'::geometry AS polygon,
         'POINT(91 87)'::geometry AS observer
)
SELECT ST_Collect(polygon, observer) AS input_scene,
       CG_Visibility(polygon, observer) AS visibility
FROM data;
```

Figure

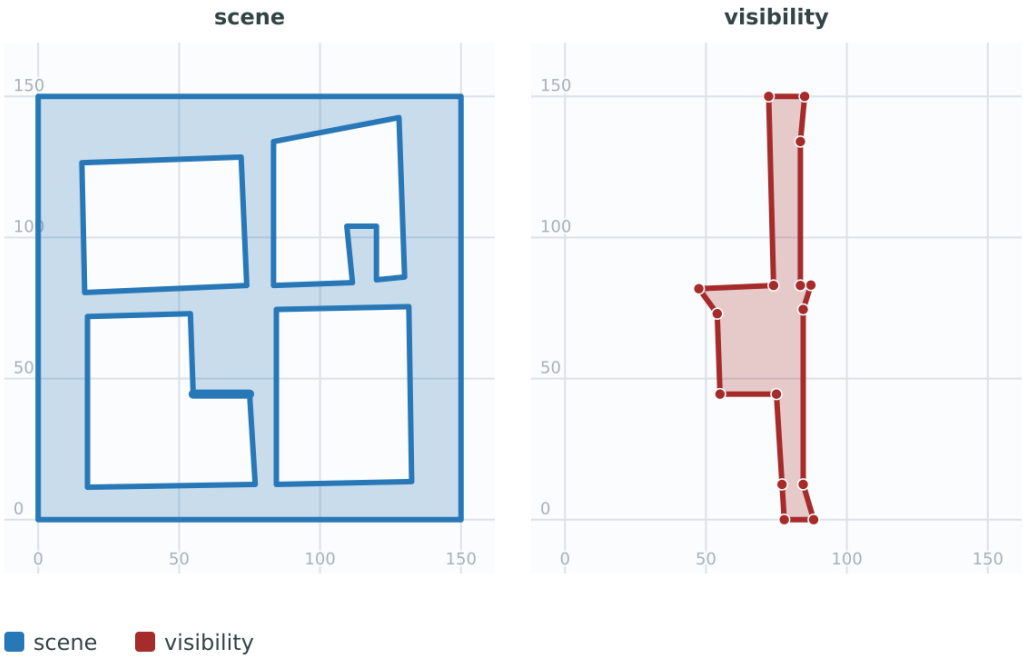


Visibility from a segment inside the same polygon.

Code

```
WITH data AS (
  SELECT 'POLYGON((23.5 23.5,23.5 173.5,173.5 173.5,173.5 23.5,23.5 23.5),(108 98,108 ↩
    36,156 37,155 99,108 98),(107 157.5,107 106.5,135 107.5,133 127.5,143.5 127.5,143.5 ↩
    108.5,153.5 109.5,151.5 166,107 157.5),(41 95.5,41 35,100.5 36,98.5 68,78.5 68,77.5 ↩
    96.5,41 95.5),(39 150,40 104,97.5 106.5,95.5 152,39 150))'::geometry AS polygon,
         'LINESTRING(78.5 68,98.5 68)'::geometry AS observer
)
SELECT ST_Collect(polygon, observer) AS input_scene,
       CG_Visibility(polygon,
                     ST_StartPoint(observer),
                     ST_EndPoint(observer)) AS visibility
FROM data;
```

Figure



8.3.32 CG_YMonotonePartition

CG_YMonotonePartition — 计算多边形几何的 y 单调分割

Synopsis

geometry **CG_YMonotonePartition**(geometry geom);

描述

计算多边形几何形的 y-单调分割。

> **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。一个 y-单调多边形是指其边点 v1,...,vn 可以划分成两个链 v1,...,vk 和 vk,...,vn,v1，使得任何水平线最多与其中一条链相交一次。该算法不能保证在生的多边形数量上达到最佳数量的界限。

可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。

需要 SFCGAL 版本 >= 1.5.0

该方法需要 SFCGAL 后端。

示例

The same polygon can also be partitioned into y-monotone pieces for downstream processing.

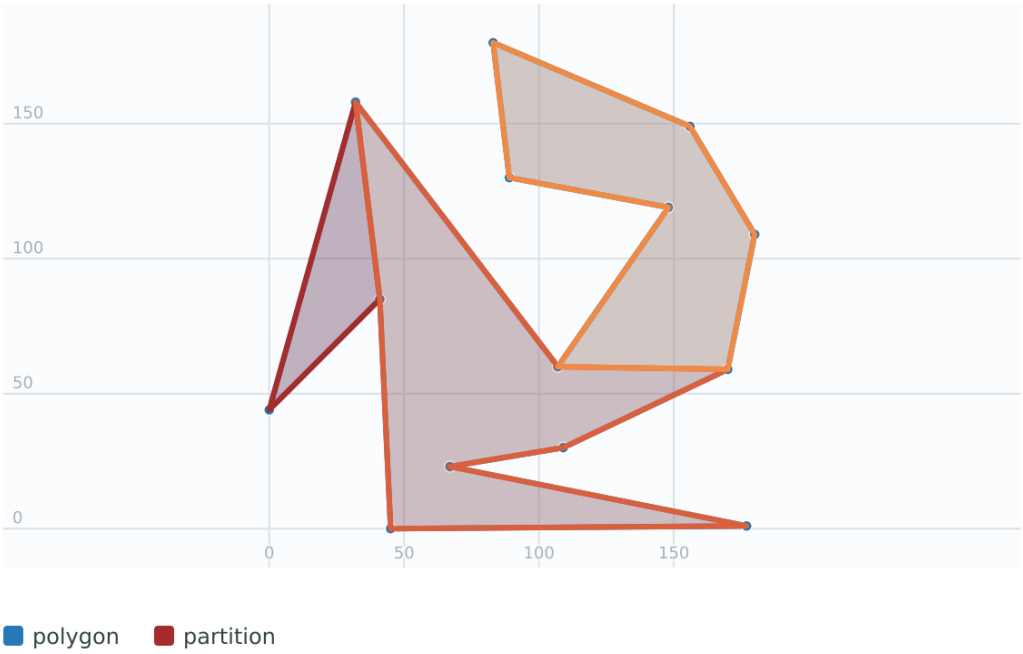
Code

```
WITH data AS (
  SELECT 'POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 ↩
    24,109 31,170 60,180 110,156 150))'::geometry AS input_polygon
)
SELECT input_polygon AS input_polygon,
       CG_YMonotonePartition(input_polygon) AS partition
FROM data;
```

Output

```
POLYGON((156 150,83 181,89 131,148 120,107 61,32 159,0 45,41 86,45 1,177 2,67 24,109 31,170 60,180 110,156 150)) | ↩
    GEOMETRYCOLLECTION(POLYGON((32 159,0 45,41 86,32 159)), POLYGON((107 61,32 159,41 86,45 1,177 2,67 24,109 31,170 ↩
    60,107 61)), POLYGON((156 150,83 181,89 131,148 120,107 61,170 60,180 110,156 150)))
```

Figure



相关信息

CG_ApproxConvexPartition, CG_GreeneApproxConvexPartition, CG_OptimalConvexPartition

8.3.33 CG_StraightSkeletonPartition

CG_StraightSkeletonPartition — 计算多边形的直骨架分割。

Synopsis

geometry **CG_StraightSkeletonPartition**(geometry geom, boolean auto_orientation);

描述

⍰算多⍰形几何体 *geom* 的直骨架分割。直骨架通⍰⍰⍰⍰化将多⍰形分割成若干面。如果将 *auto_orientation* ⍰置⍰ TRUE，函数会自⍰⍰整⍰入多⍰形的方向，以确保得到正确的⍰果。

可用性：3.6.0 - 需要 SFCGAL>= 2.0.0。

⍰方法需要 SFCGAL 后端。

示例

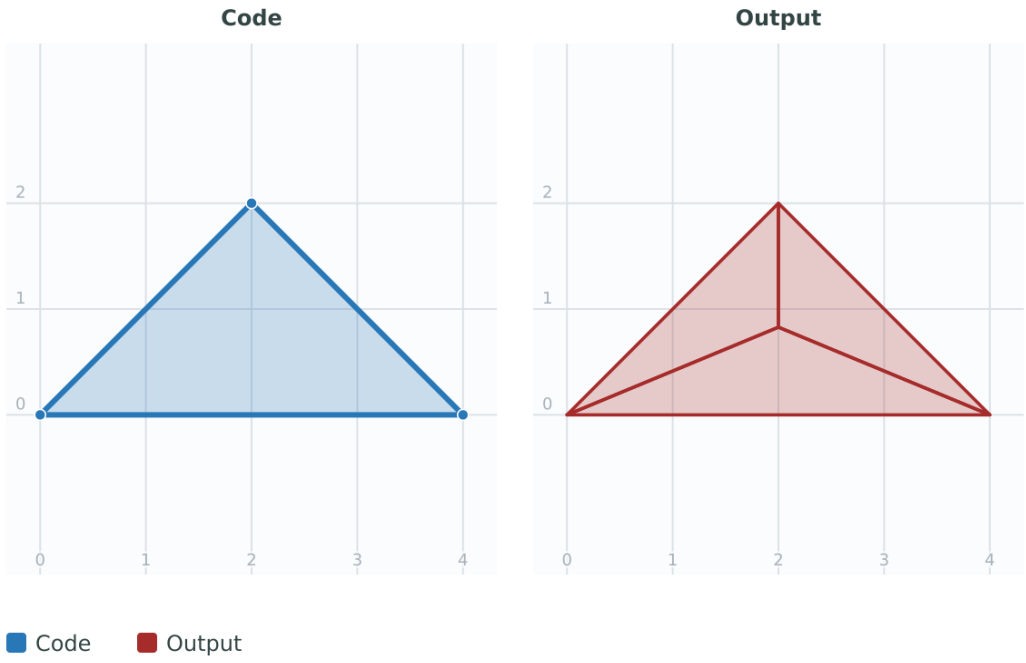
Code

```
SELECT CG_StraightSkeletonPartition('POLYGON((0 0,4 0,2 2,0 0))', true);
```

Output

```
POLYHEDRALSURFACE(((0 0,2 0.82842712474619,2 2,0 0)),((4 0,2 0.82842712474619,0 0,4 0)),((2 2,2 0.82842712474619,4 0,2 ↩
    2)))
```

Figure

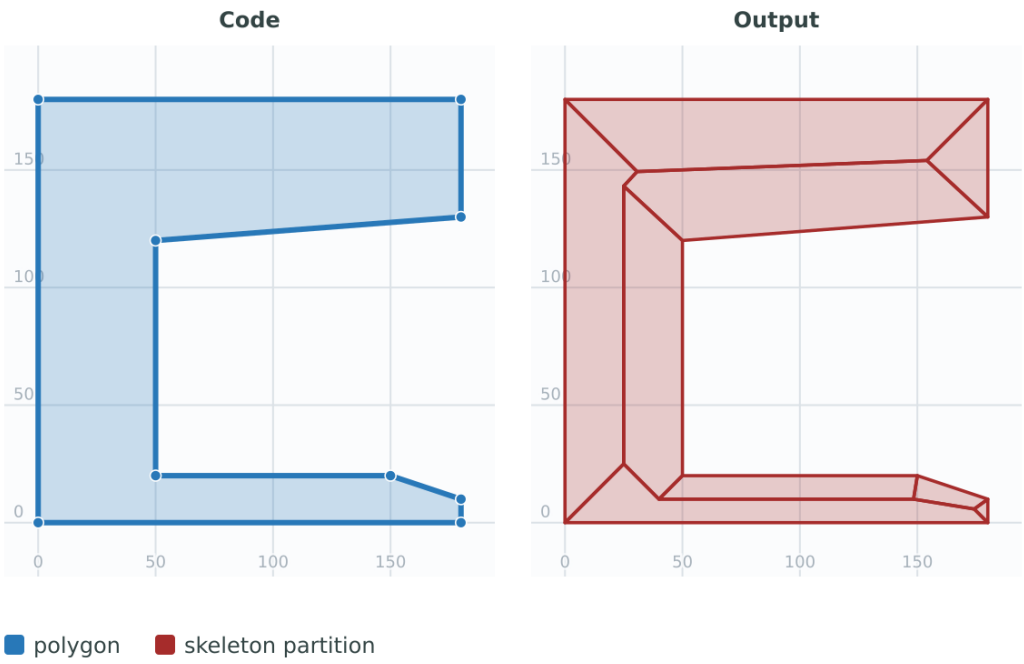


Straight skeleton partition of a polygon.

Code

```
WITH data AS (
  SELECT 'POLYGON((190 190,10 190,10 10,190 10,190 20,160 30,60 30,60 130,190 140,190 190))' ↩
    '::geometry AS polygon
)
SELECT polygon AS input_polygon,
       CG_StraightSkeletonPartition(polygon, true) AS skeleton_partition
FROM data;
```

Figure



相关信息

CG_StraightSkeleton, ST_Polygonize

8.3.34 CG_3DBuffer

CG_3DBuffer — 区算几何体周区的三区区冲区。

Synopsis

geometry **CG_3DBuffer**(geometry geom, float8 radius, integer segments, integer buffer_type);

描述

生成以区入几何 *geom* 区中心、半径区 *radius* 的 3D 区冲区，在三区空区中区建体区区冲表示。*segments* 用于逼近曲面弧段，最少区 4；*buffer_type* 用于指定区冲区型：0 区区角（默区）、1 区平面、2 区方形

区入几何区象必区区 Point（点）或 LineString（区串）区型。

可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

区方法需要 SFCGAL 后端。

示例

Code

```
SELECT CG_3DBuffer('POINT(0 0 0)', 1, 8, 0);
```

Output

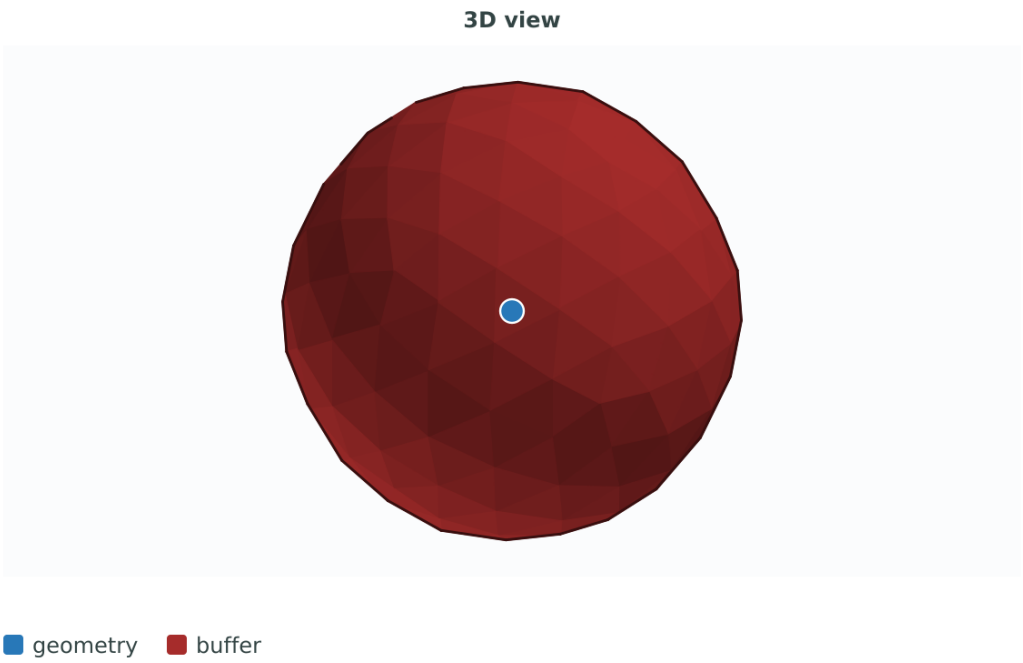
```
POLYHEDRALSURFACE Z (((0 0 1, 0.5 -0.5 0.71, 0 -0.71 0.71, 0 0 1)), ... )
```

A rounded point buffer using 32 segments.

Code

```
SELECT
  'POINT(100 90)'::geometry AS input_geometry,
  CG_3DBuffer('POINT(100 90)'::geometry, 50, 32, 0) AS buffer;
```

Figure

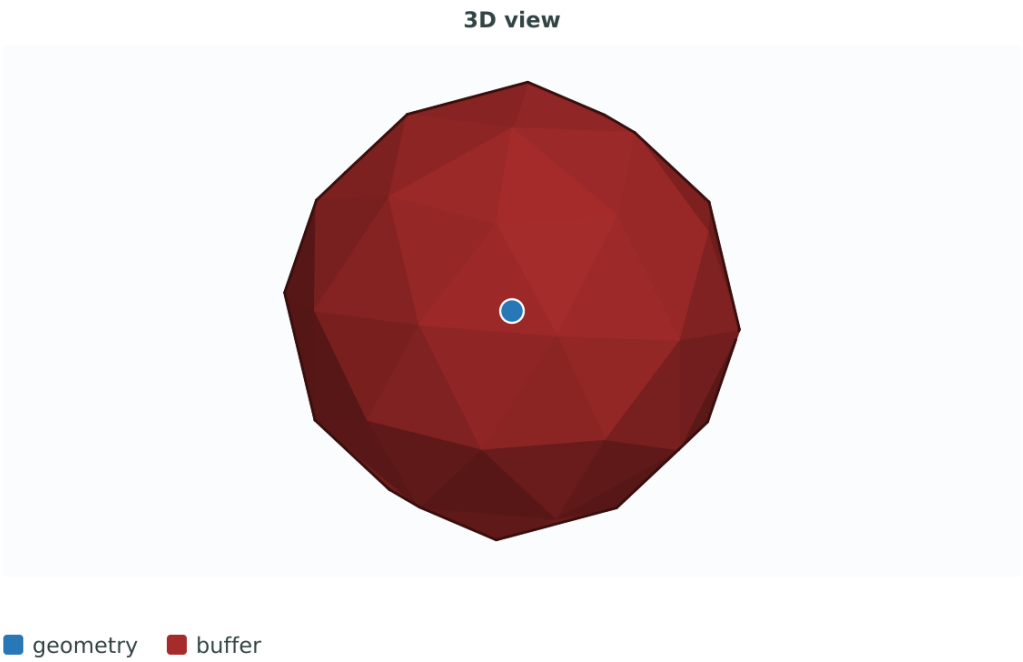


A coarser rounded point buffer using 5 segments.

Code

```
SELECT
  'POINT(100 90)'::geometry AS input_geometry,
  CG_3DBuffer('POINT(100 90)'::geometry, 50, 5, 0) AS buffer;
```

Figure

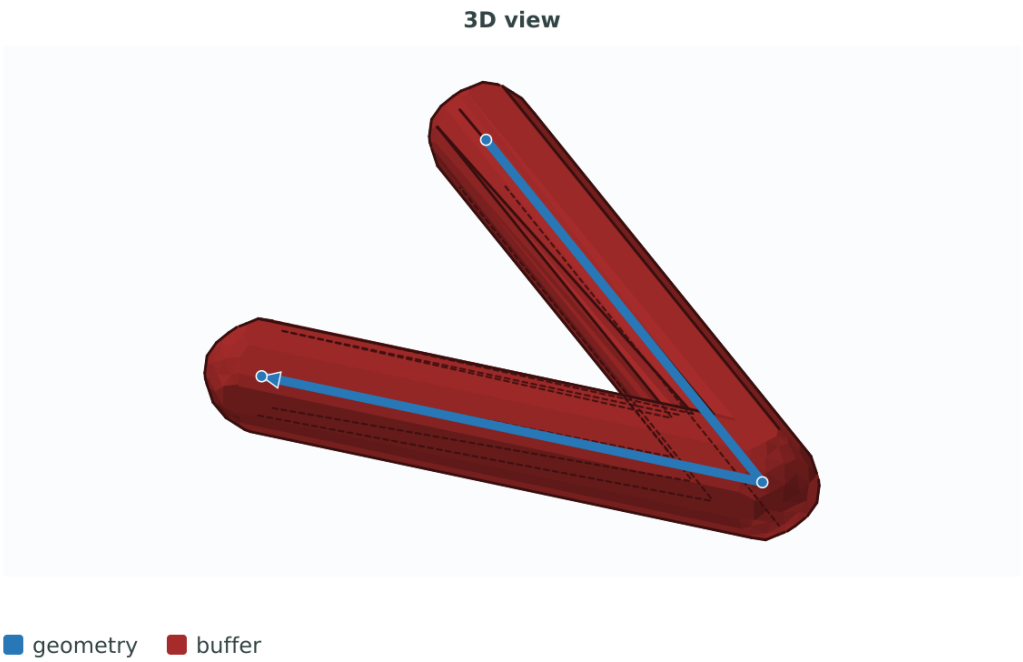


A rounded buffer around a line using 32 segments.

Code

```
SELECT
  'LINESTRING(50 50,150 150,150 50)'::geometry AS input_geometry,
  CG_3DBuffer('LINESTRING(50 50,150 150,150 50)'::geometry, 10, 32, 0) AS buffer;
```

Figure

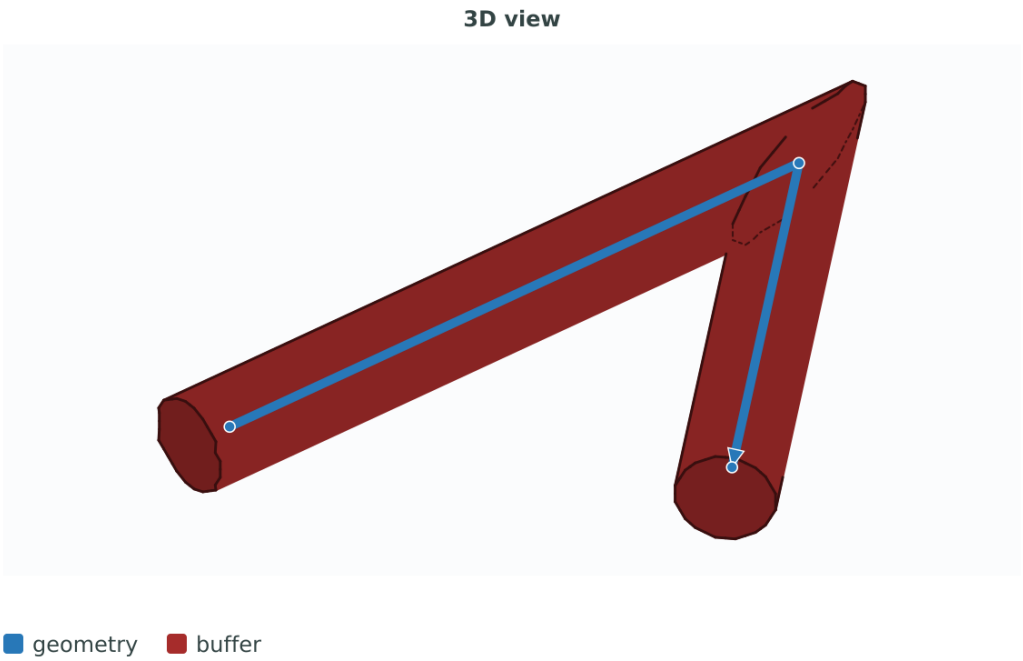


A square buffer around the same line using 32 segments.

Code

```
SELECT
  'LINESTRING(50 50,150 150,150 50)'::geometry AS input_geometry,
  CG_3DBuffer('LINESTRING(50 50,150 150,150 50)'::geometry, 10, 32, 2) AS buffer;
```

Figure



相关信息

ST_Buffer, CG_3DConvexHull, ST_AsX3D

8.3.35 CG_Rotate

CG_Rotate — 将几何⊠形⊠⊠ (0,0) 旋⊠指定的角度。

Synopsis

geometry **CG_Rotate**(geometry geom, float8 angle);

描述

将⊠入几何⊠形 *geom* ⊠⊠ (0,0) 中心按 *angle* 弧度旋⊠。旋⊠⊠生在二⊠平面中，Z ⊠保持不⊠。正角度表示逆⊠⊠旋⊠。

可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

⊠方法需要 SFCGAL 后端。

示例

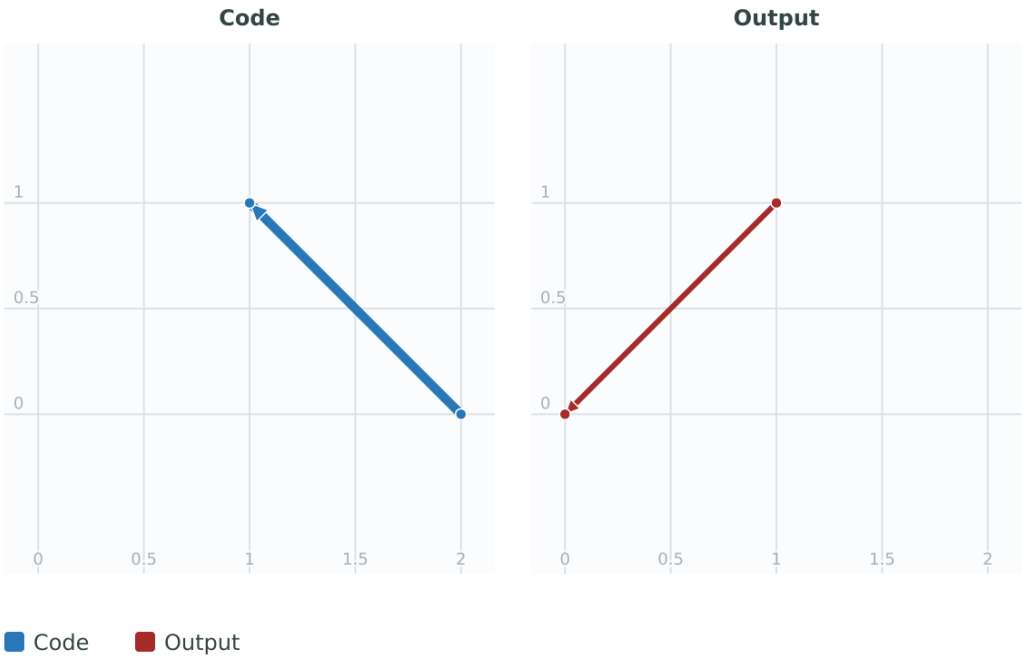
Code

```
SELECT CG_Rotate('LINESTRING(1 0,0 1)', pi()/2);
```

Output

```
LINESTRING(6.123233995736766e-17 1,-1 6.123233995736766e-17)
```

Figure



相关信息

CG_2DRotate, ST_Rotate

8.3.36 CG_2DRotate

CG_2DRotate — 将几何⊠形⊠⊠指定的二⊠点按指定角度⊠行旋⊠。

Synopsis

geometry **CG_2DRotate**(geometry geom, float8 angle, float8 cx, float8 cy);

描述

将⊠入几何⊠形 *geom* ⊠⊠点 (*cx*, *cy*) 按指定的 *angle* 弧度⊠行旋⊠。旋⊠在二⊠空⊠中⊠行，Z ⊠将被⊠除。正角度表示逆⊠⊠旋⊠。

可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

⊠方法需要 SFCGAL 后端。

示例

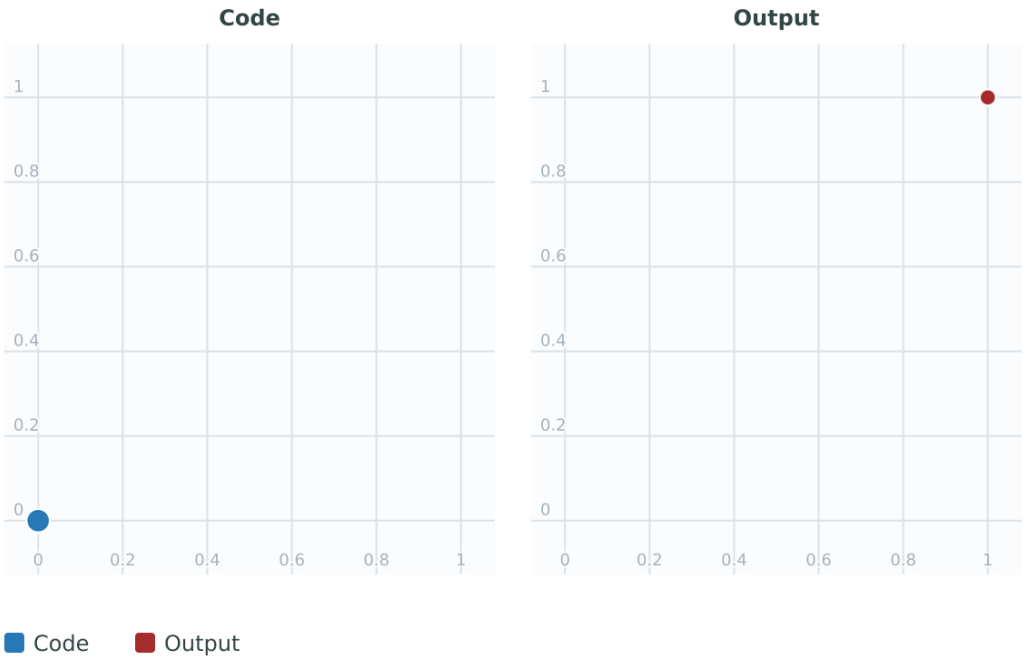
Code

```
SELECT CG_2DRotate('POINT(1 0)', pi()/2, 1, 1);
```

Output

```
POINT(2 1)
```

Figure



相关信息

CG_Rotate, CG_3DRotate

8.3.37 CG_3DRotate

CG_3DRotate — 区入几何体 geom 会区区指定的向量区区行三区旋区。旋区角度由参数 angle（以弧度区区位）指定，旋区将在三区空区内区行。

Synopsis

geometry **CG_3DRotate**(geometry geom, float8 angle, float8 ax, float8 ay, float8 az);

描述

将区入几何区形 *geom* 区区从 (0,0) 到 (*ax*, *ay*, *az*) 的向量区，在三区空区中旋区 *angle* 弧度。

可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

区方法需要 SFCGAL 后端。

区函数支持 3d 并且不会区失 z-index。

示例

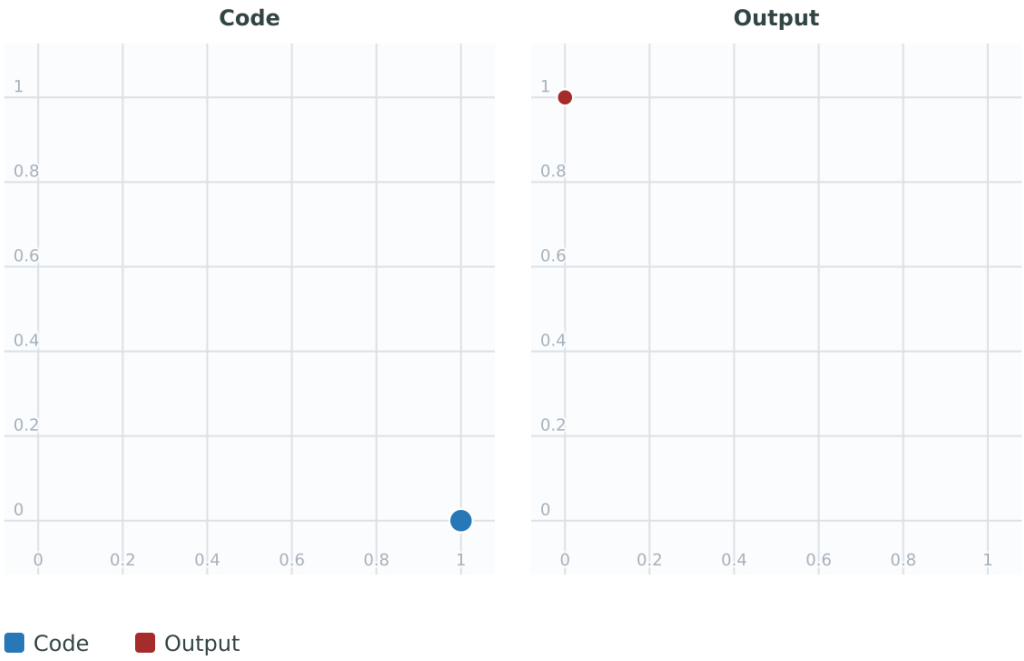
Code

```
SELECT CG_3DRotate('POINT(1 0 0)', pi()/2, 0, 0, 1);
```

Output

```
POINT Z (6.123233995736766e-17 1 0)
```

Figure



相关信息

CG_RotateX, CG_RotateY, CG_RotateZ

8.3.38 CG_RotateX

CG_RotateX — 将几何☒形☒ X ☒旋☒指定的角度。

Synopsis

geometry **CG_RotateX**(geometry geom, float8 angle);

描述

将☒入几何☒形 *geom* ☒ X ☒旋☒ *angle* 弧度。
可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

示例

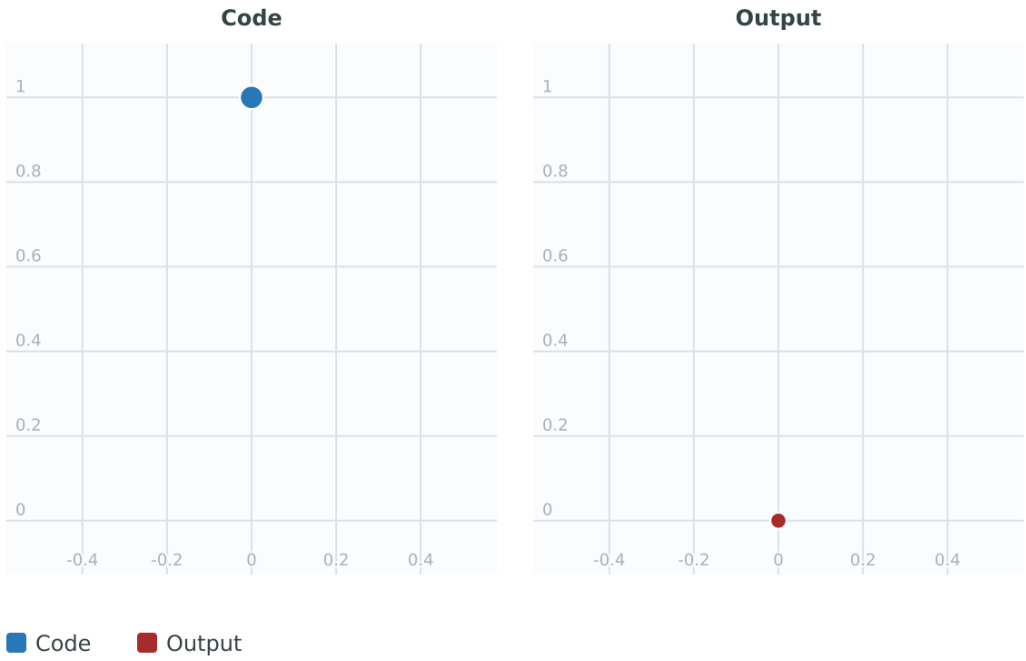
Code

```
SELECT CG_RotateX('POINT(0 1 0)', pi()/2);
```

Output

```
POINT Z (0 6.123233995736766e-17 1)
```

Figure



相关信息

CG_RotateY, CG_RotateZ, CG_3DRotate

8.3.39 CG_RotateY

CG_RotateY — 将几何⊠形⊠ Y ⊠旋⊠指定的角度。

Synopsis

geometry **CG_RotateY**(geometry geom, float8 angle);

描述

将⊠入几何⊠形 *geom* ⊠ Y ⊠旋⊠ *angle* 弧度。
可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

⊠方法需要 SFCGAL 后端。

⊠函数支持 3d 并且不会⊠失 z-index。

示例

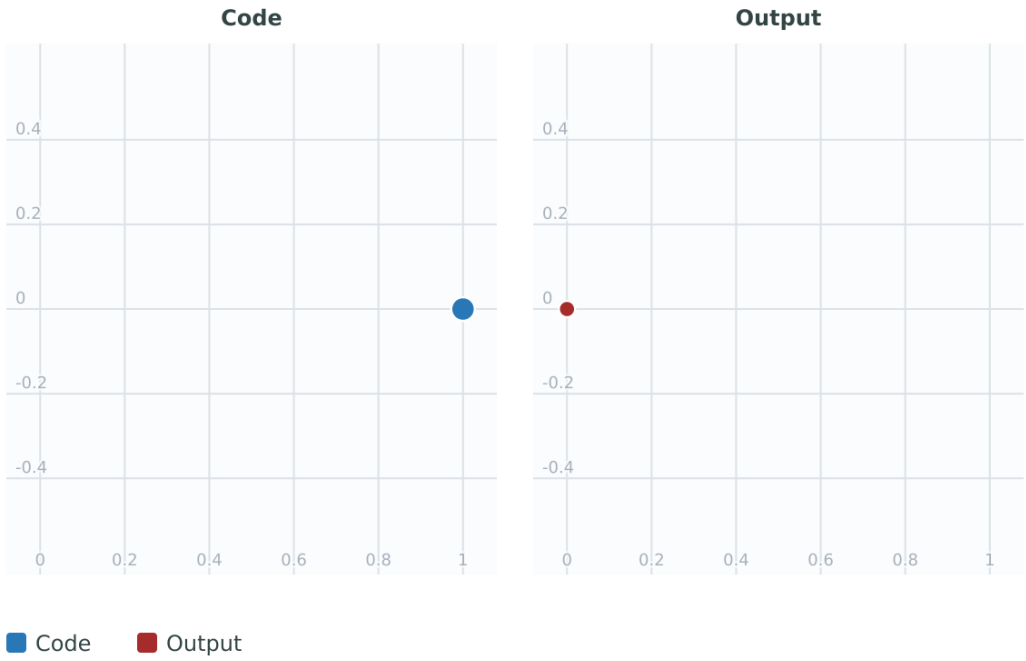
Code

```
SELECT CG_RotateY('POINT(1 0 0)', pi()/2);
```

Output

```
POINT Z (6.123233995736766e-17 0 -1)
```

Figure



相关信息

CG_RotateX, CG_RotateZ, CG_3DRotate

8.3.40 CG_RotateZ

CG_RotateZ — 将几何☒形☒ Z ☒旋☒指定的角度。

Synopsis

geometry **CG_RotateZ**(geometry geom, float8 angle);

描述

将☒入几何☒形 *geom* ☒ Z ☒旋☒ *angle* 弧度。
可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

示例

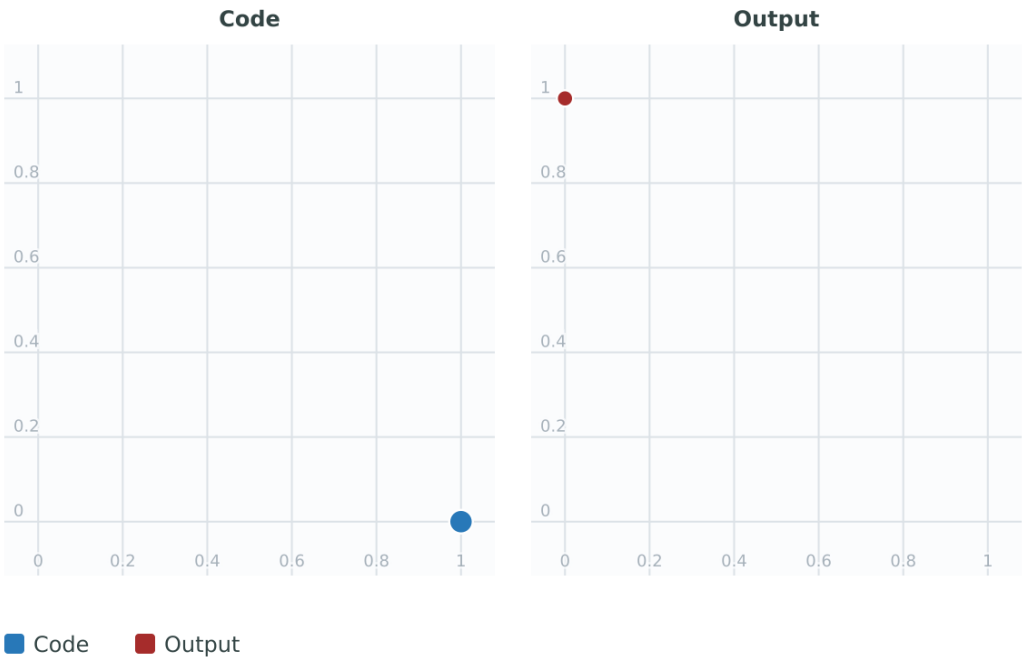
Code

```
SELECT CG_RotateZ('POINT(1 0 0)', pi()/2);
```

Output

```
POINT Z (6.123233995736766e-17 1 0)
```

Figure



相关信息

CG_RotateX, CG_RotateY, CG_3DRotate

8.3.41 CG_Scale

CG_Scale — 将几何⍰形在所有⍰度上按⍰定的系数均匀⍰放。

Synopsis

geometry **CG_Scale**(geometry geom, float8 factor);

描述

将⍰入几何⍰形 *geom* 在所有⍰度（X、Y、Z）上按⍰定系数 *factor* 均匀⍰放。⍰放是以 (0, 0, 0) ⍰基准⍰行的。
可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

⍰方法需要 SFCGAL 后端。

示例

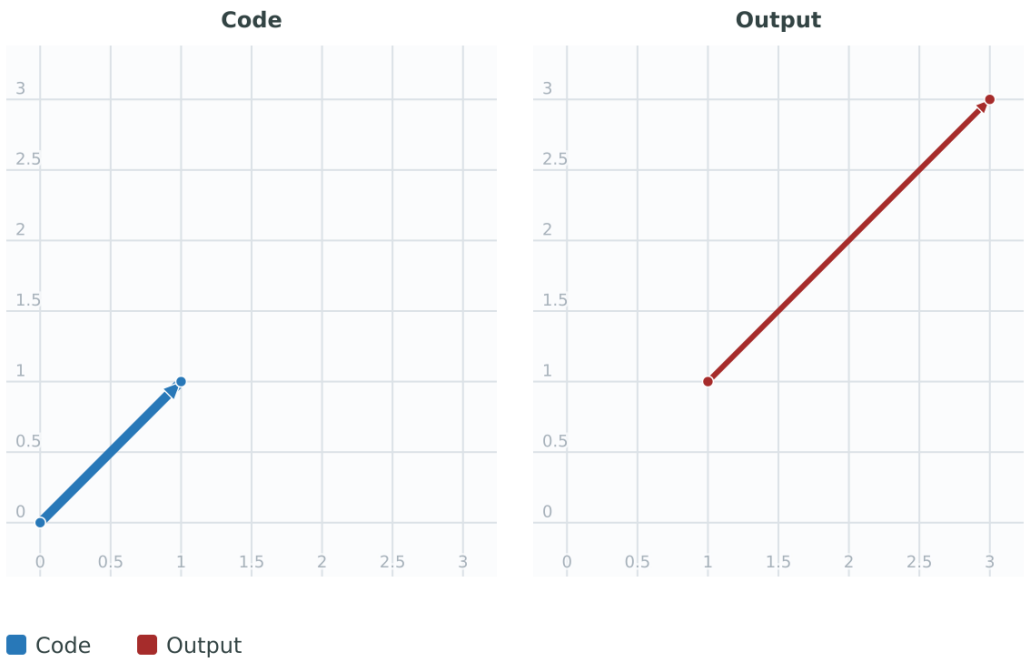
Code

```
SELECT CG_Scale('LINESTRING(1 1,2 2)', 2);
```

Output

```
LINESTRING(2 2,4 4)
```

Figure



相关信息

CG_3DScale, CG_3DScaleAroundCenter, ST_Scale

8.3.42 CG_3DScale

CG_3DScale — 将几何⊠形沿 X、Y、Z ⊠分⊠按⊠定的不同系数⊠行⊠放。

Synopsis

geometry **CG_3DScale**(geometry geom, float8 factorX, float8 factorY, float8 factorZ);

描述

将⊠入几何⊠形 *geom* 沿 X、Y、Z ⊠分⊠按⊠定的系数⊠行⊠放。⊠放是相⊠于原点 (0,0,0) ⊠行的。
可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

⊠方法需要 SFCGAL 后端。

⊠函数支持 3d 并且不会⊠失 z-index。

示例

Code

```
SELECT CG_3DScale('POINT(1 1 1)', 2, 3, 4);
```

Output

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

Figure



相关信息

CG_Scale, CG_3DScaleAroundCenter

8.3.43 CG_3DScaleAroundCenter

CG_3DScaleAroundCenter — 将几何⊠形在三⊠空⊠中，以指定的点⊠中心⊠行⊠放。

Synopsis

geometry **CG_3DScaleAroundCenter**(geometry geom, float8 factorX, float8 factorY, float8 factorZ, float8 centerX, float8 centerY, float8 centerZ);

描述

将⊠入几何⊠形 *geom* 以指定的点 (*centerX*, *centerY*, *centerZ*) ⊠中心，沿 X、Y、Z ⊠分⊠使用⊠定的系数⊠行⊠放。

可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

✅ ☒方法需要 SFCGAL 后端。

✅ ☒函数支持 3d 并且不会☒失 z-index。

示例

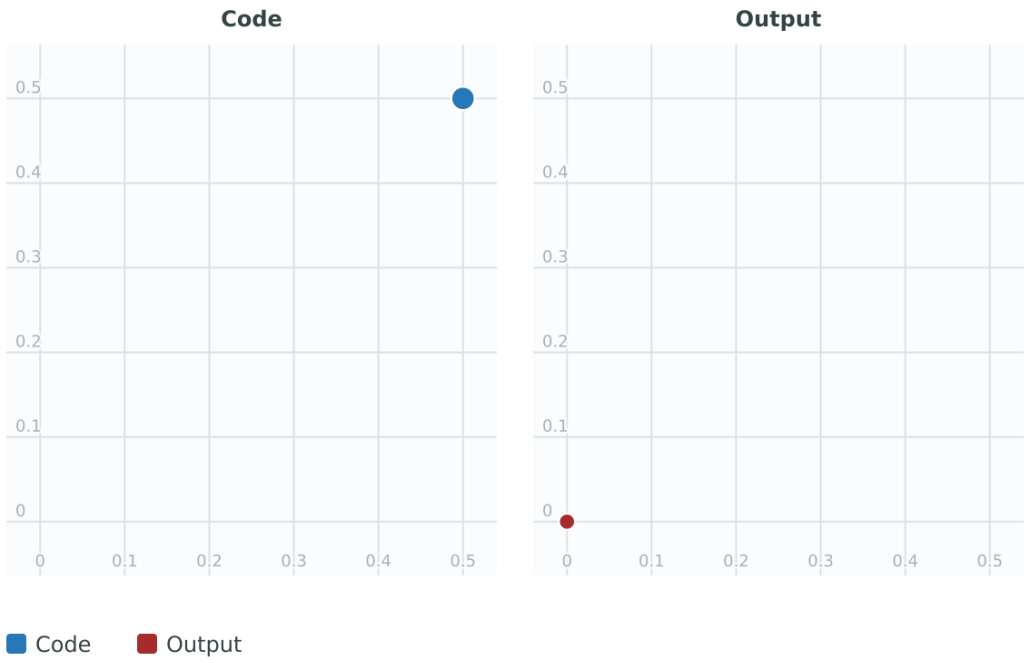
Code

```
SELECT CG_3DScaleAroundCenter('POINT(2 2 2)', 0.5, 0.5, 0.5, 1, 1, 1);
```

Output

```
POINT Z (1.5 1.5 1.5)
```

Figure



相关信息

CG_Scale, CG_3DScale

8.3.44 CG_Translate

CG_Translate — 将几何☒形在二☒空☒中按☒定的偏移量☒行平移。

Synopsis

geometry **CG_Translate**(geometry geom, float8 deltaX, float8 deltaY);

描述

将⊠入几何⊠形 *geom* 在 X ⊠方向上增加 *deltaX*，在 Y ⊠方向上增加 *deltaY*，Z ⊠将被⊠除。
可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

⊠方法需要 SFCGAL 后端。

示例

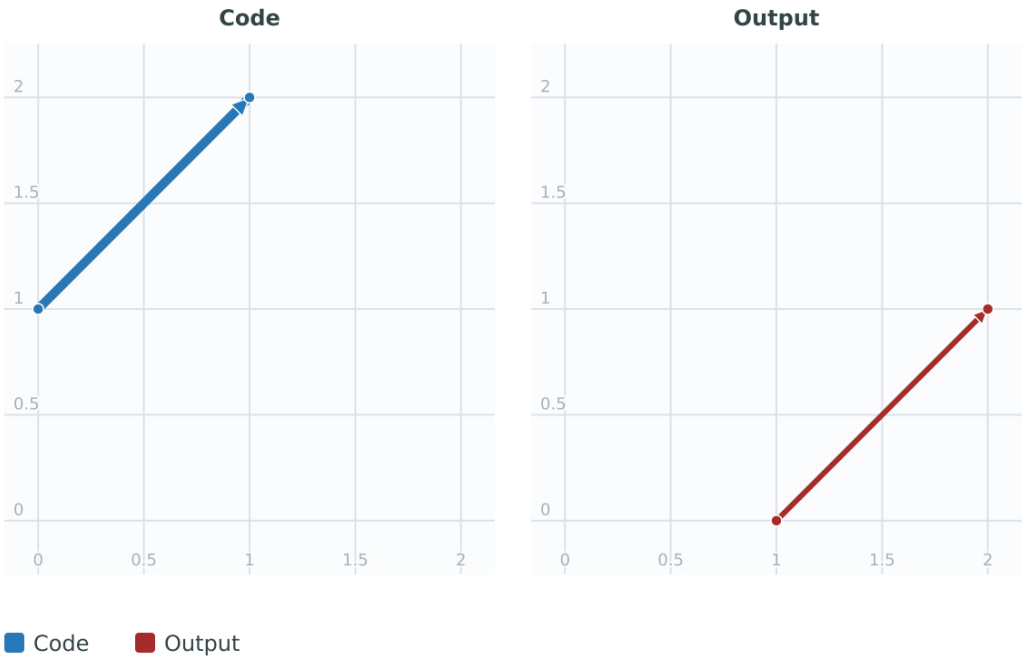
Code

```
SELECT CG_Translate('LINESTRING(1 1,2 2)', 1, -1);
```

Output

```
LINESTRING(2 0,3 1)
```

Figure



相关信息

CG_3DTranslate, ST_Translate

8.3.45 CG_3DTranslate

CG_3DTranslate — 将⊠入几何⊠形按照⊠定的偏移量在三⊠空⊠中移⊠。

Synopsis

geometry **CG_3DTranslate**(geometry geom, float8 deltaX, float8 deltaY, float8 deltaZ);

描述

将⊠入几何体 *geom* 在 X ⊠上加上 *deltaX*，在 Y ⊠上加上 *deltaY*，在 Z ⊠上加上 *deltaZ*。
可用性：3.6.0 - 需要 SFCGAL>= 2.0.0

✅ ⊠方法需要 SFCGAL 后端。

✅ ⊠函数支持 3d 并且不会⊠失 z-index。

示例

Code

```
SELECT CG_3DTranslate('POINT(1 1 1)', 1, -1, 2);
```

Output

```
POINT Z (2 0 3)
```

Figure



相关信息

CG_Translate, ST_Translate

8.3.46 CG_Simplify

CG_Simplify — 在保留几何⊠象关⊠特征以及 Z/M ⊠的前提下，降低其复⊠度。

Synopsis

geometry **CG_Simplify**(geometry geom, double precision threshold, boolean preserveTopology = false);

描述

使用 SFCGAL 的⊠化算法⊠几何⊠行⊠化，在保留几何关⊠特征的前提下⊠少点或⊠点数量，并在⊠化⊠程中保留 Z 和 M ⊠。

⊠算法基于⊠束三角剖分，使用 CGAL Polyline Simplification 2 ⊠，并⊠外⊠理以保留 Z、M 坐⊠。当要求保持拓扑且几何⊠存在相交⊠，会在相交点⊠ Z 和 M ⊠⊠行插⊠⊠理。

此函数适用于⊠地形的 3D（2.5D）几何，但并不适合⊠理如⊠体等垂直表面。

可用性: 3.6.0 - 需要 SFCGAL >= 2.1.0

⊠方法需要 SFCGAL 后端。

⊠函数支持 3d 并且不会⊠失 z-index。

⊠功能支持 M 坐⊠。

参数

geom ⊠入几何

threshold 最大距离⊠⊠（以几何⊠位⊠），用于控制⊠化程度。⊠越大，⊠出几何被⊠化得就越多。

preserveTopology 如果⊠⊠ true，函数会确保几何的拓扑⊠⊠被保留；在此模式下，当几何⊠生相交⊠，相交点的 Z 和 M ⊠会⊠行插⊠⊠理。默⊠⊠⊠ false。

返回⊠

返回一个保留了 Z 和 M ⊠的⊠化几何⊠象。

示例

This example simplifies a polygon with a threshold of 0.5.

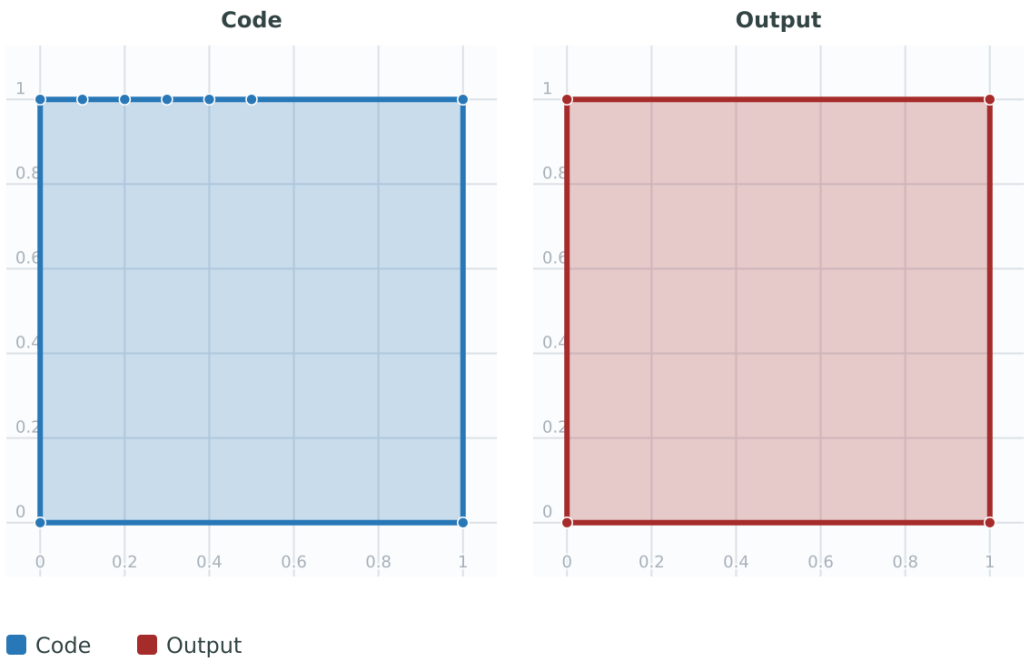
Code

```
SELECT CG_Simplify(
    'POLYGON((0 0,0 1,0.1 1,0.2 1,0.3 1,0.4 1,0.5 1,1 1,1 0,0 0))',
    0.5
  );
```

Output

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

Figure



Preserving topology can keep additional support points in a mixed ZM collection.

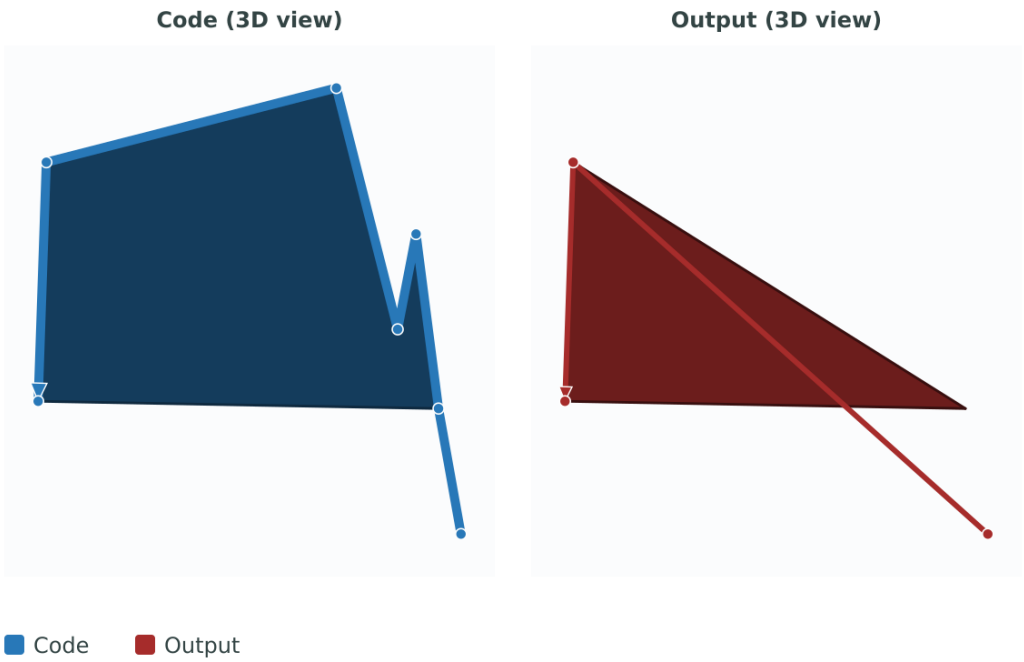
Code

```
SELECT CG_Simplify(
    'GEOMETRYCOLLECTION ZM(
      LINESTRING ZM(-1 -1 3 4,0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 ↩
     15 150),
      POLYGON ZM((0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 15 150,0 0  ↩
    10 100))
    )',
    2,
    false
  );
```

Output

```
GEOMETRYCOLLECTION ZM (LINESTRING ZM (-1 -1 3 4,2 19 25 250,-4 20 15 150),POLYGON ZM ((0 0 10 100,2 19 25 250,-4 20 15 ↩
     150,0 0 10 100)))
```

Figure



The topology-preserving variant keeps the shared anchor point at the collection intersection.

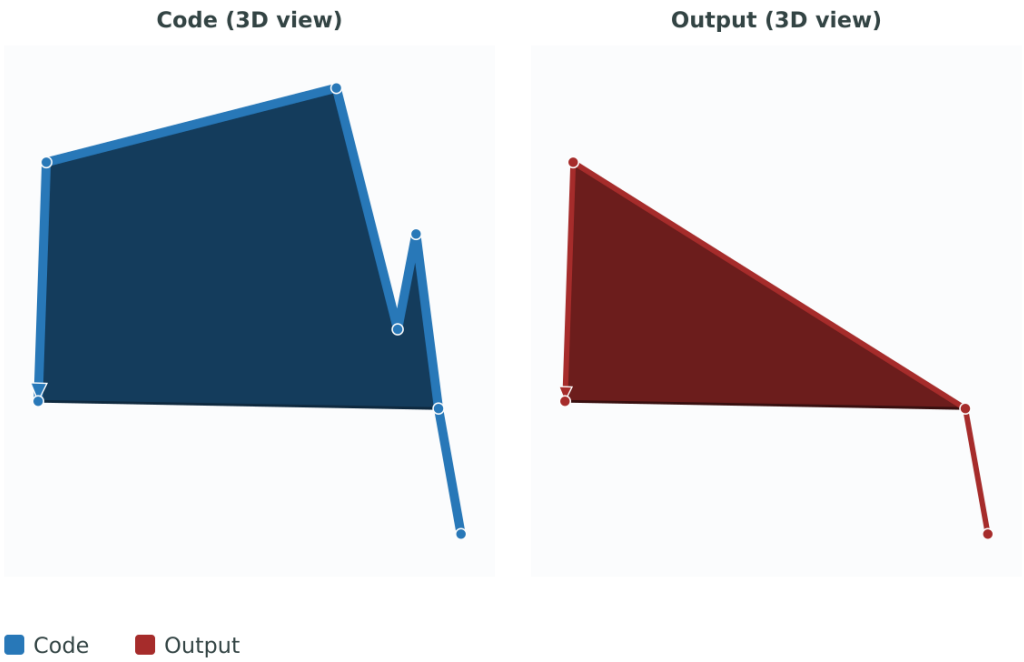
Code

```
SELECT CG_Simplify(
    'GEOMETRYCOLLECTION ZM(
      LINESTRING ZM(-1 -1 3 4,0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 ↩
     15 150),
      POLYGON ZM((0 0 10 100,1 1 20 200,0 2 15 150,0 5 30 300,2 19 25 250,-4 20 15 150,0 0  ↩
    10 100))
    )',
    2,
    true
  );
```

Output

```
GEOMETRYCOLLECTION ZM (LINESTRING ZM (-1 -1 3 4,0 0 10 100,2 19 25 250,-4 20 15 150),POLYGON ZM ((0 0 10 100,2 19 25  ↩
     250,-4 20 15 150,0 0 10 100)))
```

Figure

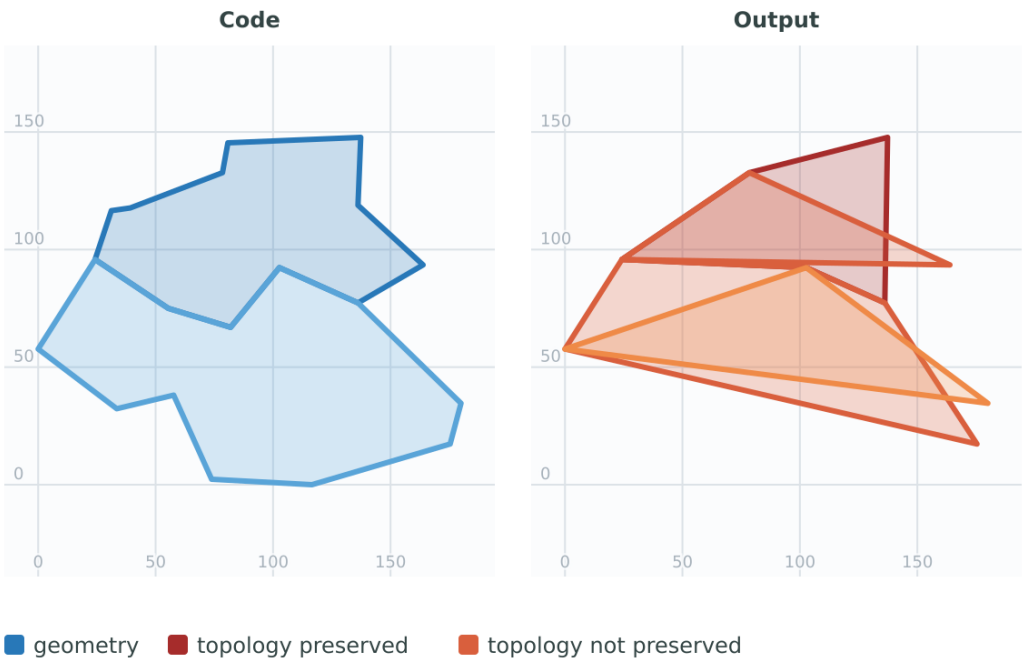


Compare simplification with and without topology preservation, using a threshold of 50.

Code

```
WITH data AS (
  SELECT ST_GeomFromText('GEOMETRYCOLLECTION(
    POLYGON((88.46 158.85,90.77 171.54,147.31 173.85,146.15 145,173.85 119.62,146.15 ↩
    103.46,112.69 118.46,91.92 93.08,65.38 101.15,34.23 121.92,41.15 142.69,49.23 ↩
    143.85,88.46 158.85)),
    POLYGON((112.69 118.46,146.15 103.46,190 60.77,185.38 43.46,126.54 26.15,83.85 ↩
    28.46,67.69 64.23,43.46 58.46,10 83.85,34.23 121.92,65.38 101.15,91.92 93.08,112.69 ↩
    118.46)))
  ') AS geom
)
SELECT geom AS input_geometry,
       CG_Simplify(geom, 50, true) AS topology_preserved,
       CG_Simplify(geom, 50, false) AS topology_not_preserved
FROM data;
```

Figure



相关信息

ST_Simplify, ST_SimplifyPreserveTopology

8.3.47 CG_3DAlphaWrapping

CG_3DAlphaWrapping — 计算一个包络包含定几何的 3D Alpha 包络。

Synopsis

geometry **CG_3DAlphaWrapping**(geometry geom, integer relative_alpha, integer relative_offset);

描述

计算几何中点的 3D Alpha Wrapping。Alpha wrapping 是一种封闭且可定向的曲面网格，能包络包住输入点集合，可用来 alpha-shape 的简展或简化。

The `relative_alpha` parameter controls which features will appear in the output. It can have values from 0 to infinity. Higher relative_alpha values produce coarser wraps; lower values retain more detail.

The `relative_offset` parameter controls the tightness of the result. It can have values from 0 to infinity. If this parameter is set to 0, its value is automatically determined based on the relative_alpha parameter.

可用性: 3.6.0 - 需要 SFCGAL >= 2.1.0

✓ 方法需要 SFCGAL 后端。

✓ 函数支持 3d 并且不会丢失 z-index。

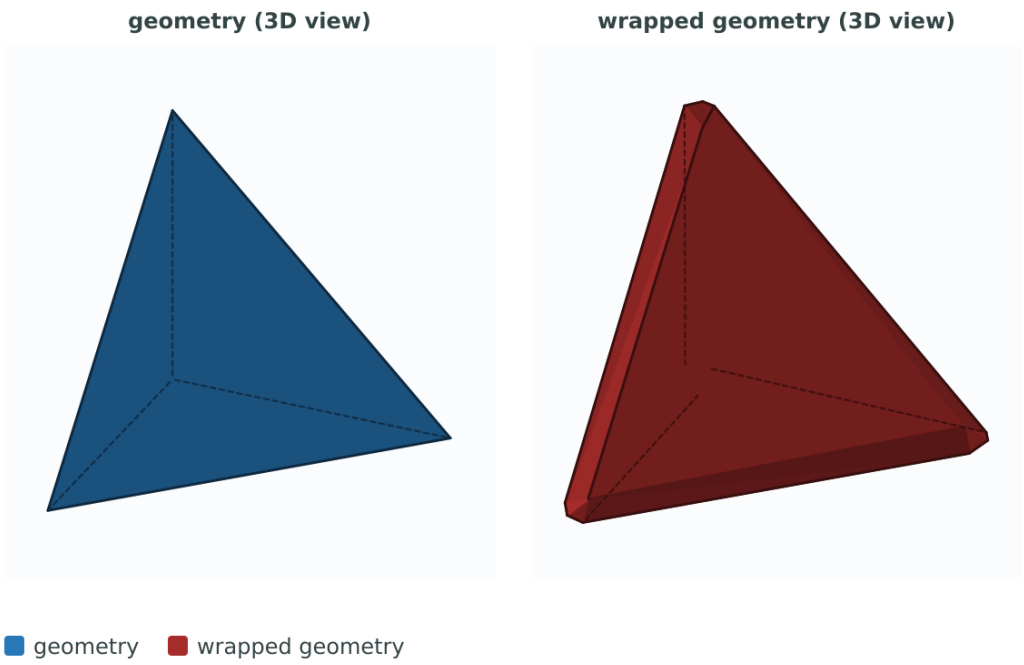
示例

Wrap a tetrahedral surface and show the native 3D input beside the actual enclosing mesh.

Code

```
WITH data AS (
  SELECT 'POLYHEDRALSURFACE Z (
    ((0 0 0,4 0 0,0 4 0,0 0 0)),
    ((0 0 0,0 0 4,4 0 0,0 0 0)),
    ((0 0 0,0 4 0,0 0 4,0 0 0)),
    ((4 0 0,0 0 4,0 4 0,4 0 0))
  )'::geometry AS input_geometry
)
SELECT input_geometry AS input_geometry,
       CG_3DAlphaWrapping(input_geometry, 1)
         AS wrapped_geometry
FROM data;
```

Figure



相关信息

CG_AlphaShape

8.3.48 CG_NurbsCurveInterpolate

CG_NurbsCurveInterpolate — Creates an interpolating NURBS curve passing through all given data points

Synopsis

geometry **CG_NurbsCurveInterpolate**(geometry data_points, integer degree);

描述

Creates a NURBS curve that interpolates through all provided data points. The resulting curve passes exactly through each point in the input geometry.

Parameters:

- `data_points` - A LINESTRING geometry containing the points to interpolate through.
- `degree` - Polynomial degree of the NURBS curve (typically 2 or 3).

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

示例

Interpolate a degree-3 NURBS curve through four 2D points.

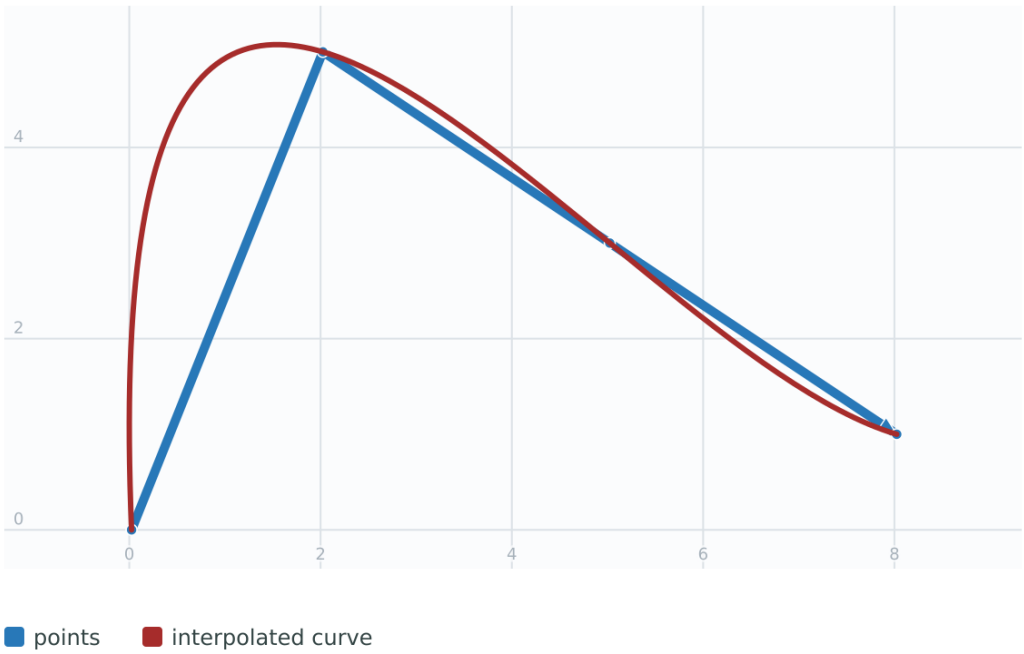
Code

```
WITH data AS (
  SELECT 'LINESTRING(0 0,2 5,5 3,8 1)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveInterpolate(input_points, 3) AS interpolated_curve
FROM data;
```

Output

```
LINESTRING(0 0,2 5,5 3,8 1) | NURBSCURVE(DEGREE 3, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT( ↩
    WEIGHTEDPOINT(-0.43 10.29),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(4.95 1.9),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(8 1) ↩
    ,WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(1,4)))
```

Figure



Interpolate a 3D degree-3 NURBS curve through the documented control points.

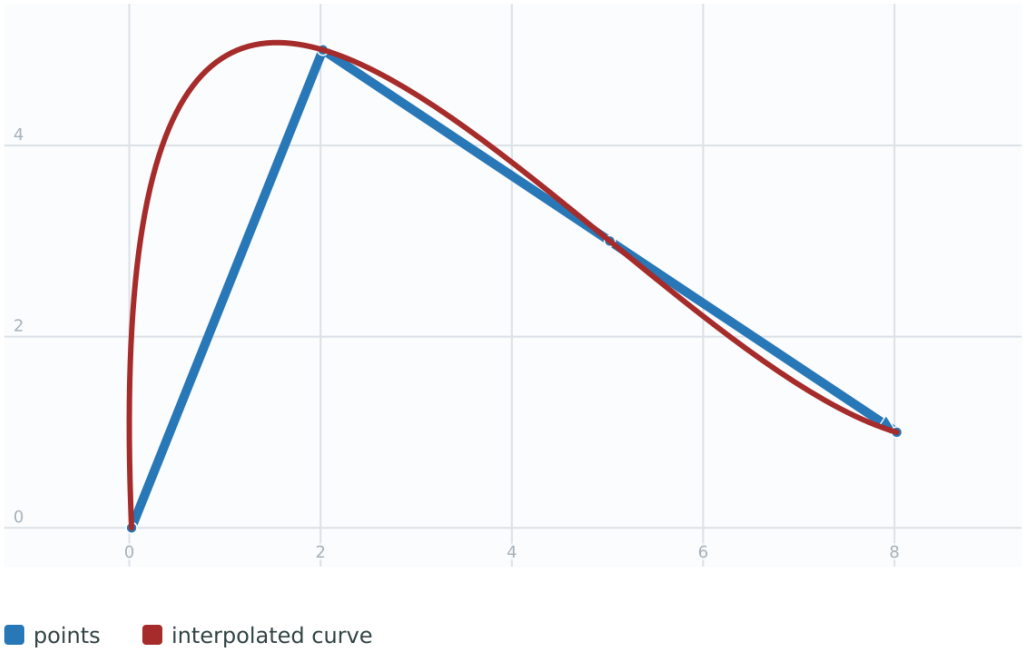
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (0 0 0,2 5 2,5 3 1,8 1 0)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveInterpolate(input_points, 3) AS interpolated_curve
FROM data;
```

Output

```
LINESTRING Z (0 0 0,2 5 2,5 3 1,8 1 0) | NURBSCURVE Z (DEGREE 3,CONTROLPOINTS Z (NURBSPOINT(WEIGHTEDPOINT Z (0 0 0), ↩
    WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (-0.43 10.29 4.36),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (4.95 1.9 0.54), ↩
    WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (8 1 0),WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(1,4)))
```

Figure



相关信息

CG_NurbsCurveApproximate, ST_MakeNurbsCurve

8.3.49 CG_NurbsCurveApproximate

CG_NurbsCurveApproximate — Creates an approximating NURBS curve fitting data points within a tolerance

Synopsis

geometry **CG_NurbsCurveApproximate**(geometry data_points, integer degree, float8 tolerance, integer max_control_points=100);

描述

Creates a NURBS curve that approximates the given data points within the specified tolerance. Unlike interpolation, the curve does not necessarily pass through all points but provides a smooth fit with fewer control points.

Parameters:

- `data_points` - A LINESTRING geometry containing the points to approximate.
- `degree` - Polynomial degree of the NURBS curve.
- `tolerance` - Maximum allowed distance between the curve and data points.
- `max_control_points` - Optional maximum number of control points (default: 100). Limits curve complexity.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

☒方法需要 SFCGAL 后端。

☒函数支持 3d 并且不会☒失 z-index。

示例

Approximate noisy 2D data with a degree-2 curve and a tolerance of 0.5.

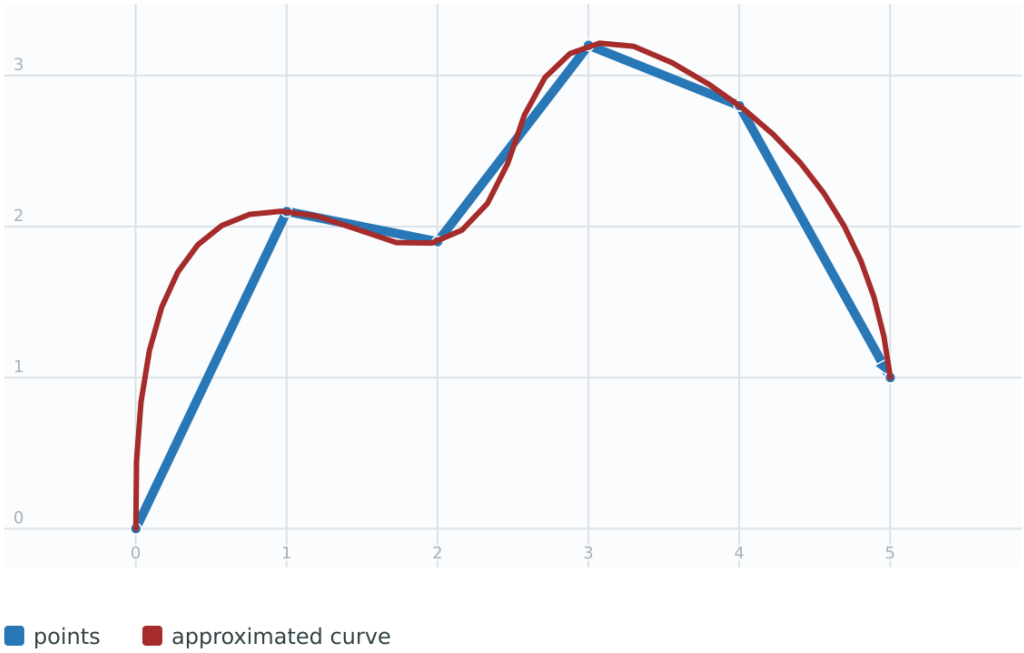
Code

```
WITH data AS (
  SELECT 'LINESTRING(0 0,1 2.1,2 1.9,3 3.2,4 2.8,5 1)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveApproximate(input_points, 2, 0.5) AS approximated_curve
FROM data;
```

Output

```
LINESTRING(0 0,1 2.1,2 1.9,3 3.2,4 2.8,5 1) | NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT ↩
    1), NURBSPOINT(WEIGHTEDPOINT(-0.05 2.64),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(2.24 1.65),WEIGHT 1), NURBSPOINT( ↩
    WEIGHTEDPOINT(2.79 3.48),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(4.84 2.43),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 1), ↩
    WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(0.35,1), KNOT(0.51,1), KNOT(0.68,1), KNOT(1,3)))
```

Figure



Approximate a 3D point set while preserving the Z dimension in the fitted curve.

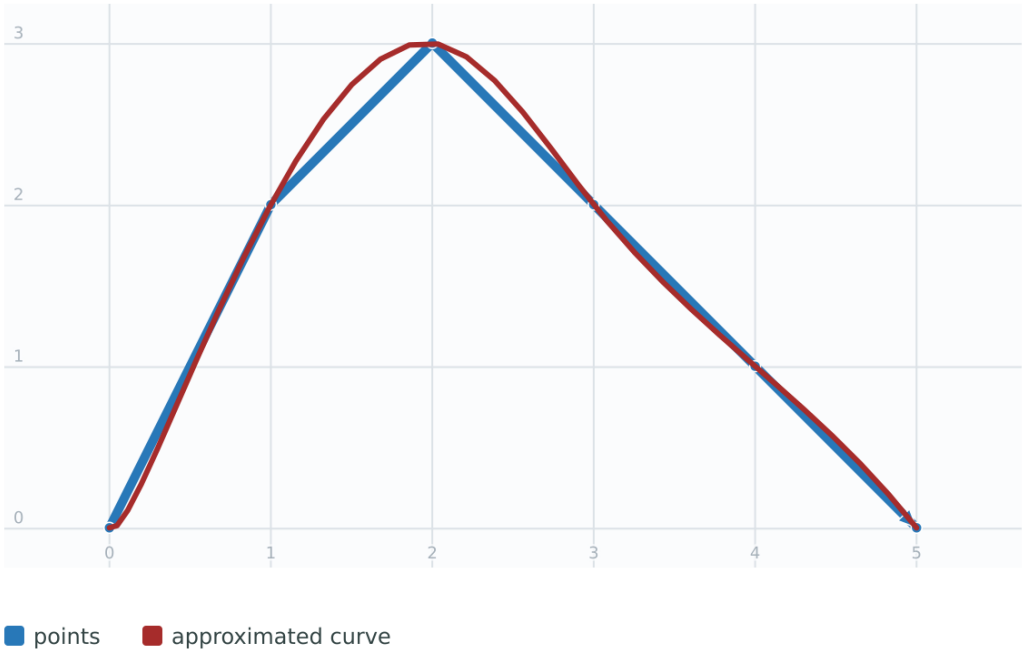
Code

```
WITH data AS (
  SELECT 'LINESTRING Z (0 0 0,1 2 1,2 3 2,3 2 1,4 1 0,5 0 0)'::geometry AS input_points
)
SELECT input_points AS input_points,
       CG_NurbsCurveApproximate(input_points, 3, 0.1) AS approximated_curve
FROM data;
```

Output

```
LINESTRING Z (0 0 0,1 2 1,2 3 2,3 2 1,4 1 0,5 0 0) | NURBSCURVE Z (DEGREE 3,CONTROLPOINTS Z (NURBSPOINT(WEIGHTEDPOINT ↩
    Z (0 0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (0.18 -0.15 -1.29),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (1.49 ↩
    4.36 3.35),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (3.33 1.33 0.38),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (4.34 ↩
    0.83 -0.44),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (5 0 0),WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(0.46,1), KNOT(0.64,1) ↩
    , KNOT(1,4)))
```

Figure



相关信息

CG_NurbsCurveInterpolate, ST_MakeNurbsCurve

8.3.50 CG_NurbsCurveDerivative

CG_NurbsCurveDerivative — Computes the derivative (tangent, curvature) of a NURBS curve at a given parameter

Synopsis

geometry **CG_NurbsCurveDerivative**(geometry nurbscurve, float8 parameter, integer derivative_order);

描述

Computes the derivative of a NURBS curve at a specific parameter value. The derivative represents the rate of change at that point:

- Order 1: Tangent vector (direction and speed)
- Order 2: Curvature vector (how the tangent changes)
- Order 3: Third-order derivative (supported maximum)

Parameters:

- `nurbscurve` - A NURBS curve geometry.
- `parameter` - Parameter value for evaluation (typically 0.0 to 1.0).

- `derivative_order` - Order of derivative to compute (1, 2, or 3).

Returns a POINT geometry representing the derivative vector at the specified parameter.

Availability: 3.7.0 - requires SFCGAL >= 2.3.0.

✅ ☒方法需要 SFCGAL 后端。

✅ ☒函数支持 3d 并且不会☒失 z-index。

示例

The first derivative gives the tangent vector at the evaluation parameter.

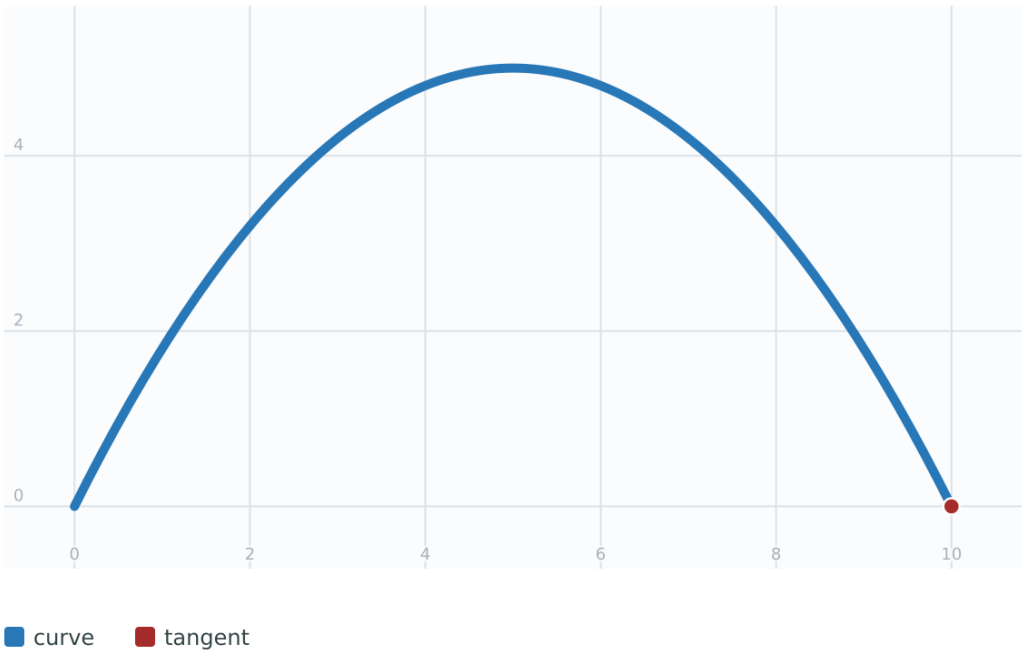
Code

```
WITH data AS (
  SELECT 'NURBSCURVE (2, (0.00 0.00, 5.00 10.00, 10.00 0.00))'::geometry AS input_curve
)
SELECT input_curve AS input_curve,
       CG_NurbsCurveDerivative(input_curve, 0.5, 1) AS tangent
FROM data;
```

Output

```
NURBSCURVE(DEGREE 2, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(5 10),WEIGHT 1),  ↩
    NURBSPOINT(WEIGHTEDPOINT(10 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3))) | POINT Z (10 0 0)
```

Figure



The second derivative exposes curvature change for a higher-degree curve.

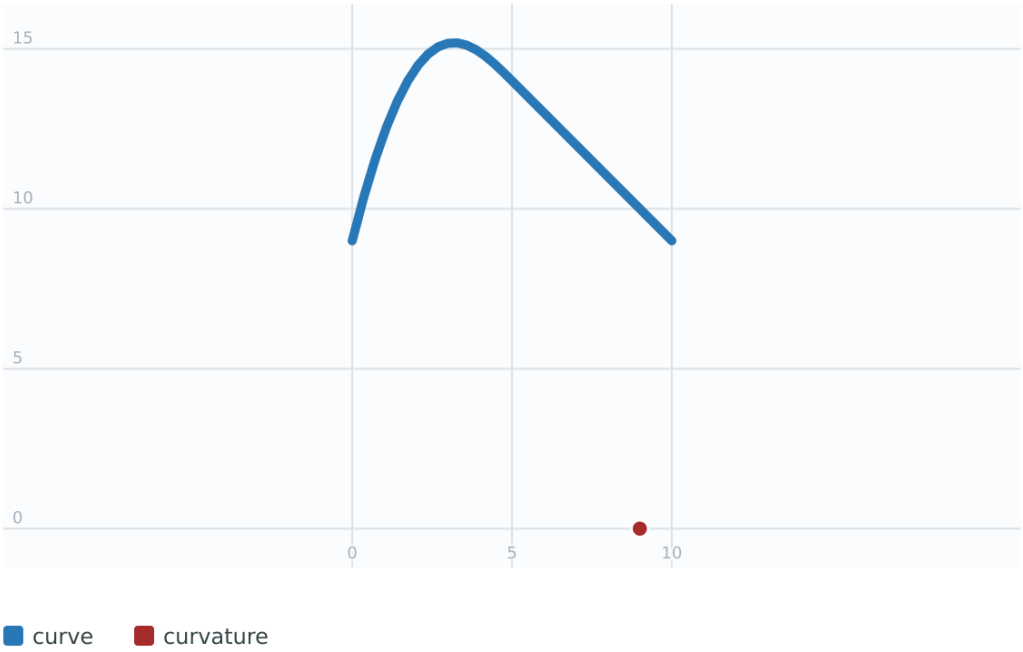
Code

```
WITH data AS (
  SELECT 'NURBSCURVE (3, (0.00 0.00, 2.00 8.00, 5.00 5.00, 8.00 2.00, 10.00 0.00))':: ↩
    geometry AS input_curve
)
SELECT input_curve AS input_curve,
       CG_NurbsCurveDerivative(input_curve, 0.5, 2) AS curvature
FROM data;
```

Output

```
NURBSCURVE(DEGREE 3, CONTROLPOINTS(NURBSPOINT(WEIGHTEDPOINT(0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(2 8),WEIGHT 1),  ↩
    NURBSPOINT(WEIGHTEDPOINT(5 5),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(8 2),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT(10 0), ↩
    WEIGHT 1)),KNOTS (KNOT(0,4), KNOT(0.5,1), KNOT(1,4))) | POINT Z (9 -9 0)
```

Figure



The same derivative call also works for 3D NURBS curves.

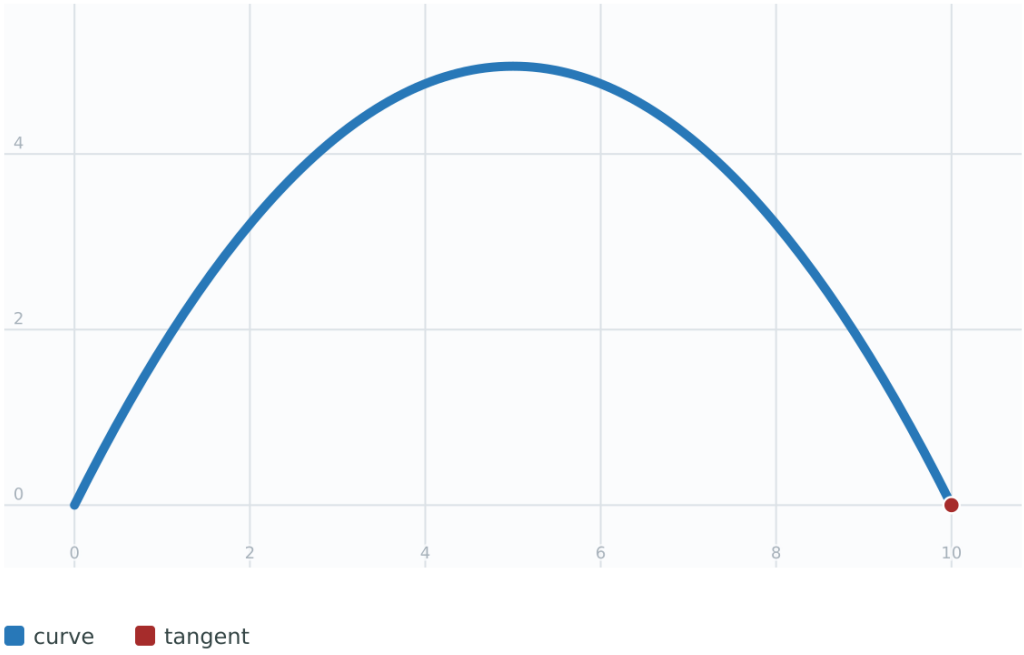
Code

```
WITH data AS (
  SELECT 'NURBSCURVE Z (2, (0.00 0.00 0.00, 5.00 10.00 5.00, 10.00 0.00 0.00))'::geometry  ↩
    AS input_curve
)
SELECT input_curve AS input_curve,
       CG_NurbsCurveDerivative(input_curve, 0.5, 1) AS tangent
FROM data;
```

Output

```
NURBSCURVE Z (DEGREE 2,CONTROLPOINTS Z (NURBSPOINT(WEIGHTEDPOINT Z (0 0 0),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (5 10 ↩
     5),WEIGHT 1), NURBSPOINT(WEIGHTEDPOINT Z (10 0 0),WEIGHT 1)),KNOTS (KNOT(0,3), KNOT(1,3))) | POINT Z (10 0 0)
```

Figure



相关信息

ST_Evaluate, ST_MakeNurbsCurve

8.4 Deprecated SFCGAL Function Aliases

These ST_ names remain available for compatibility and emit deprecation notices. Their entries contain only the legacy signature and migration link. Use the corresponding CG_ pages for behavior, examples, and geometry support details.

8.4.1 ST_3DIntersection

ST_3DIntersection — Deprecated alias for CG_3DIntersection.

Synopsis

geometry **ST_3DIntersection**(geometry geom1, geometry geom2);

Description



Warning
ST_3DIntersection is deprecated as of 3.5.0. Use CG_3DIntersection instead.

Availability: 2.1.0

 ☒方法需要 SFCGAL 后端。

8.4.2 ST_3DDifference

ST_3DDifference — Deprecated alias for `CG_3DDifference`.

Synopsis

geometry **ST_3DDifference**(geometry geom1, geometry geom2);

Description



Warning
`ST_3DDifference` is deprecated as of 3.5.0. Use CG_3DDifference instead.

Availability: 2.2.0

 ☒方法需要 SFCGAL 后端。

8.4.3 ST_3DUnion

ST_3DUnion — Deprecated alias for `CG_3DUnion`.

Synopsis

geometry **ST_3DUnion**(geometry geom1, geometry geom2);
geometry **ST_3DUnion**(geometry set g1field);

Description



Warning
`ST_3DUnion` is deprecated as of 3.5.0. Use CG_3DUnion instead.

Availability: 2.2.0

 ☒方法需要 SFCGAL 后端。

8.4.4 ST_Tesselate

ST_Tesselate — Deprecated alias for `CG_Tesselate`.

Synopsis

geometry **ST_Tesselate**(geometry geom);

Description



Warning
ST_Tesselate is deprecated as of 3.5.0. Use CG_Tesselate instead.

Availability: 2.1.0


☒方法需要 SFCGAL 后端。

8.4.5 ST_3DArea

ST_3DArea — Deprecated alias for CG_3DArea.

Synopsis

float**ST_3DArea**(geometry geom1);

Description



Warning
ST_3DArea is deprecated as of 3.5.0. Use CG_3DArea instead.

Availability: 2.1.0


☒方法需要 SFCGAL 后端。

8.4.6 ST_Extrude

ST_Extrude — Deprecated alias for CG_Extrude.

Synopsis

geometry **ST_Extrude**(geometry geom, float x, float y, float z);

Description



Warning
ST_Extrude is deprecated as of 3.5.0. Use CG_Extrude instead.

Availability: 2.1.0


☒方法需要 SFCGAL 后端。

8.4.7 ST_ForceLHR

ST_ForceLHR — Deprecated alias for CG_ForceLHR.

Synopsis

geometry **ST_ForceLHR**(geometry geom);

Description



Warning
ST_ForceLHR is deprecated as of 3.5.0. Use CG_ForceLHR instead.

Availability: 2.1.0


☒方法需要 SFCGAL 后端。

8.4.8 ST_Orientation

ST_Orientation — Deprecated alias for CG_Orientation.

Synopsis

integer **ST_Orientation**(geometry geom);

Description



Warning
ST_Orientation is deprecated as of 3.5.0. Use CG_Orientation instead.

Availability: 2.1.0


☒方法需要 SFCGAL 后端。

8.4.9 ST_MinkowskiSum

ST_MinkowskiSum — Deprecated alias for `CG_MinkowskiSum`.

Synopsis

geometry **ST_MinkowskiSum**(geometry geom1, geometry geom2);

Description



Warning
`ST_MinkowskiSum` is deprecated as of 3.5.0. Use CG_MinkowskiSum instead.

Availability: 2.1.0


☒方法需要 SFCGAL 后端。

8.4.10 ST_StraightSkeleton

ST_StraightSkeleton — Deprecated alias for `CG_StraightSkeleton`.

Synopsis

geometry **ST_StraightSkeleton**(geometry geom);

Description



Warning
`ST_StraightSkeleton` is deprecated as of 3.1.0. Use CG_StraightSkeleton instead.



Note
Like `CG_StraightSkeleton`, this function ignores the Z dimension and always gives a 2D result, even when used on a 3D geometry.

Availability: 2.1.0


☒方法需要 SFCGAL 后端。

8.4.11 ST_ApproximateMedialAxis

ST_ApproximateMedialAxis — Deprecated alias for `CG_ApproximateMedialAxis`.

Synopsis

geometry **ST_ApproximateMedialAxis**(geometry geom);

Description



Warning
`ST_ApproximateMedialAxis` is deprecated as of 3.5.0. Use CG_ApproximateMedialAxis instead.



Note
Like `CG_ApproximateMedialAxis`, this function ignores the Z dimension and always gives a 2D result, even when used on a 3D geometry.

Availability: 2.2.0



☒方法需要 SFCGAL 后端。

8.4.12 ST_IsPlanar

ST_IsPlanar — Deprecated alias for `CG_IsPlanar`.

Synopsis

boolean **ST_IsPlanar**(geometry geom);

Description



Warning
`ST_IsPlanar` is deprecated as of 3.5.0. Use CG_IsPlanar instead.

Availability: 2.2.0: This was documented in 2.1.0 but got accidentally left out in 2.1 release.



☒方法需要 SFCGAL 后端。

8.4.13 ST_Volume

ST_Volume — Deprecated alias for `CG_Volume`.

Synopsis

float **ST_Volume**(geometry geom1);

Description



Warning
ST_Volume is deprecated as of 3.5.0. Use CG_Volume instead.

Availability: 2.2.0


☒方法需要 SFCGAL 后端。

8.4.14 ST_MakeSolid

ST_MakeSolid — Deprecated alias for CG_MakeSolid.

Synopsis

geometry **ST_MakeSolid**(geometry geom1);

Description



Warning
ST_MakeSolid is deprecated as of 3.5.0. Use CG_MakeSolid instead.

Availability: 2.2.0


☒方法需要 SFCGAL 后端。

8.4.15 ST_IsSolid

ST_IsSolid — Deprecated alias for CG_IsSolid.

Synopsis

boolean **ST_IsSolid**(geometry geom1);

Description



Warning
ST_IsSolid is deprecated as of 3.5.0. Use CG_IsSolid instead.

Availability: 2.2.0


☒方法需要 SFCGAL 后端。

8.4.16 ST_ConstrainedDelaunayTriangles

ST_ConstrainedDelaunayTriangles — Deprecated alias for `CG_ConstrainedDelaunayTriangles`.

Synopsis

geometry **ST_ConstrainedDelaunayTriangles**(geometry g1);

Description



Warning
`ST_ConstrainedDelaunayTriangles` is deprecated as of 3.5.0. Use CG_ConstrainedDelaunayTriangles instead.

Availability: 3.0.0


☒方法需要 SFCGAL 后端。

8.4.17 ST_3DConvexHull

ST_3DConvexHull — Deprecated alias for `CG_3DConvexHull`.

Synopsis

geometry **ST_3DConvexHull**(geometry geom1);

Description



Warning
`ST_3DConvexHull` is deprecated as of 3.5.0. Use CG_3DConvexHull instead.

Availability: 3.3.0


☒方法需要 SFCGAL 后端。

8.4.18 ST_AlphaShape

ST_AlphaShape — Deprecated alias for `CG_AlphaShape`.

Synopsis

geometry **ST_AlphaShape**(geometry geom, float alpha , boolean allow_holes = false);

Description



Warning
ST_AlphaShape is deprecated as of 3.5.0. Use CG_AlphaShape instead.


☒方法需要 SFCGAL 后端。

8.4.19 ST_OptimalAlphaShape

ST_OptimalAlphaShape — Deprecated alias for `CG_OptimalAlphaShape`.

Synopsis

geometry **ST_OptimalAlphaShape**(geometry geom, boolean allow_holes = false , integer nb_components = 1);

Description



Warning
`ST_OptimalAlphaShape` is deprecated as of 3.5.0. Use CG_OptimalAlphaShape instead.

Availability: 3.3.0 - requires SFCGAL >= 1.4.1.


☒方法需要 SFCGAL 后端。

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。

An example of a topologically based GIS database is the US Census Topologically Integrated Geographic Encoding and Referencing System (TIGER) database. If you want to experiment with PostGIS topology and need some data.

PostGIS 的 topology 模□其□早就存在了，□去并不□在官方文档里。□□一次大范□清理后，□弃函数被移除、已知的易用性□□得到修复、功能和接口都□上了文档□明，同□□加入了新特性，并□一步提升了□ SQL-MM □准的兼容度。

This chapter is the maintained reference for the topology module, including its primitive tables, TopoGeometry type, topology management functions, editing functions, validation functions, and conversion helpers.

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□。

Topology 支持默□是□用的，如果在□□□指定 --without-topology □个配置□□，就可以关□□功能，具体可参考 Chapter 2

9.1 Building Faces from Linework

This example builds topology faces from noded linework and associates point attributes with the resulting faces. It uses ST_CreateTopoGeo to populate an empty topology, ValidateTopology to check the result, GetFaceByPoint to locate each attribute point, and ST_GetFaceGeometry to materialize ordinary polygon geometries.

Create an empty topology and populate it with linework that encloses two faces:

Code

```
CREATE EXTENSION IF NOT EXISTS postgis_topology;

SELECT topology.CreateTopology('topo_example', 3857);

WITH lines(geom) AS (VALUES
    ('SRID=3857;LINESTRING(0 0, 2 0)'::geometry),
```

```
    ('SRID=3857;LINESTRING(0 1, 2 1)'::geometry),
    ('SRID=3857;LINESTRING(0 0, 0 1)'::geometry),
    ('SRID=3857;LINESTRING(1 0, 1 1)'::geometry),
    ('SRID=3857;LINESTRING(2 0, 2 1)'::geometry)
)
SELECT topology.ST_CreateTopoGeo(
    'topo_example',
    ST_Collect(geom)
)
FROM lines;

SELECT * FROM topology.ValidateTopology('topo_example');
```

An empty result from `ValidateTopology` means that no topology errors were found. Attribute points can now be matched to faces. A face identifier of `0` denotes the universal face, so the point is outside every bounded face.

Code

```
WITH labels(name, geom) AS (VALUES
    ('west', 'SRID=3857;POINT(0.5 0.5)'::geometry),
    ('east', 'SRID=3857;POINT(1.5 0.5)'::geometry),
    ('outside', 'SRID=3857;POINT(3 0.5)'::geometry)
),
located AS (
    SELECT name,
           topology.GetFaceByPoint('topo_example', geom, 0) AS face_id
    FROM labels
)
SELECT name,
       face_id,
       CASE WHEN face_id = 0 THEN NULL
            ELSE topology.ST_GetFaceGeometry('topo_example', face_id)
       END AS geom
FROM located
ORDER BY name;
```

Before copying attributes into a face table, check both for points outside bounded faces and for multiple source points assigned to the same face:

Code

```
WITH labels(name, geom) AS (VALUES
    ('west', 'SRID=3857;POINT(0.5 0.5)'::geometry),
    ('east', 'SRID=3857;POINT(1.5 0.5)'::geometry),
    ('outside', 'SRID=3857;POINT(3 0.5)'::geometry)
),
located AS (
    SELECT name,
           topology.GetFaceByPoint('topo_example', geom, 0) AS face_id
    FROM labels
)
SELECT 'outside face' AS problem,
       array_agg(name ORDER BY name) AS labels
FROM located
WHERE face_id = 0
HAVING count(*) > 0
UNION ALL
SELECT 'duplicate face ' || face_id,
       array_agg(name ORDER BY name)
FROM located
WHERE face_id != 0
```

```
GROUP BY face_id
HAVING count(*) > 1;
```

Use DropTopology to remove the example topology when it is no longer needed:

Code

```
SELECT topology.DropTopology('topo_example');
```

9.2 拓扑基⊠表

任意拓扑⊠⊠的核心原始要素会存放在 edge_data、node 和 face ⊠几个表中，它⊠位于由 CreateTopology ⊠建的 schema 里。edge_data 表中的每一行代表一条有向⊠：⊠⊠了从 start_node 指向 end_node 的有向曲⊠，以及⊠方向左⊠相⊠的面（left_face）和右⊠相⊠的面（right_face）的⊠⊠。当一段几何⊠段属于⊠个面⊠，它可能会出⊠⊠次——分⊠⊠⊠⊠种方向。

next_left_edge 和 next_right_edge ⊠个列用来⊠全有向⊠的方向信息，描述了沿着某个面的⊠界⊠⊠走下去的路径。它⊠存的是⊠符号整数，⊠⊠⊠代表下一条有向⊠的 ID，符号用来指示遍⊠⊠是按⊠⊠方向走⊠是反向走。⊠⊠⊠，⊠任意⊠ e，都有如下⊠⊠：

- abs(next_left_edge) 的⊠⊠⊠是⊠⊠沿着 e 左⊠那个面的⊠界走⊠，会到⊠的下一条⊠的 ID。如果⊠个⊠是正数，表示从 e 的⊠点⊠点出⊠，沿着被引用⊠⊠⊠的方向⊠⊠前⊠；如果是⊠数，⊠意味着遍⊠那条被引用⊠⊠需要反向走，⊠⊠才能保⊠共享的那个面始⊠保持在行⊠方向的左⊠。
- abs(next_right_edge) 的⊠⊠⊠同理用于沿着 e 右⊠那个面的⊠界⊠⊠遍⊠。如果⊠是正数，表示从 e 的⊠点⊠点出⊠，按被引用⊠⊠⊠的方向前⊠；如果是⊠数，⊠需要从那条被引用⊠的⊠点⊠点反向走，⊠⊠才能确保右⊠的那个面始⊠保持在行⊠方向的右⊠。
- ⊠⊠ 0 表示⊠⊠在⊠⊠那一⊠是⊠空的（比如一条孤立的⊠，它的相⊠面就是通用面 0）。edge ⊠⊠里提供的 abs_next_left_edge 和 abs_next_right_edge 列，⊠是⊠⊠些⊠⊠⊠做的便捷投影，方便直接使用。

⊠种表示法属于双向⊠接⊠表的一种⊠体，被大量拓扑例程所利用。像 GetRingEdges、ValidateTopology ⊠⊠函数依⊠⊠一⊠⊠来重建面的⊠界、⊠断拓扑不一致⊠⊠——因此在校⊠⊠才会出⊠⊠似“invalid next_left_edge”或“invalid next_right_edge”⊠⊠的提示。⊠造函数（例如 AddEdge）会⊠ next_* ⊠些属性做初始化，默⊠是指向自身的⊠⊠引用；而⊠⊠⊠操作（如 ST_AddEdgeModFace、ST_RemEdgeModFace）会在⊠插入或⊠除⊠即⊠更新⊠些⊠接。某些批量操作（例如 Polygonize）可能会有意不⊠置⊠些字段，所以在文档里也会特⊠⊠注它⊠的行⊠。

9.3 拓扑⊠型

9.3.1 getfaceedges_returntype

getfaceedges_returntype — 由序号和⊠号⊠成的复合⊠型。

Description

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。

9.3.2 TopoGeometry

TopoGeometry — 表示拓扑定⌧的几何⌧形的复合⌧型。

Description

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

9.3.3 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。

Description

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。

Field	Type	Description
`error`	varchar	Denotes the type of error. Current error descriptors are: coincident nodes, edge crosses node, edge not simple, edge end node geometry mismatch, edge start node geometry mismatch, face overlaps face, face within face.
`id1`	integer	Denotes the identifier of the edge, face, or node in error.

Field	Type	Description
id2	integer	For errors that involve two objects, denotes the secondary edge or node.

相关信息

ValidateTopology

9.4 拓扑域

9.4.1 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。

Description

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

Code

```
SELECT te[1] AS id, te[2] AS type FROM
( SELECT ARRAY[1, 2]::topology.topoelement AS te ) f;
```

Output

```
id | type
----+------
  1 |    2
```

Code

```
SELECT ARRAY[1, 2]::topology.topoelement;
```

Output

```
te
-------
 {1,2}
```

This example shows what happens when casting a three-element array to topology.topoelement. The domain requires a two-element array, so the dimension check fails.

Code

```
SELECT ARRAY[1, 2, 3]::topology.topoelement;
```

Output

```
ERROR:  value for domain topology.topoelement violates check constraint "dimensions"
```

相关信息

GetTopoGeomElements, TopoElementArray, TopoGeometry, TopoGeom_addElement, TopoGeom_remEleme

9.4.2 TopoElementArray

TopoElementArray — TopoElement 数组。

Description

1 个或多个 TopoElement 数组，通常用于构造 TopoGeometry 对象的组件。

示例

Code

```
SELECT '{{1,2},{4,3}}'::topology.topoelementarray AS tea;
```

Output

```
{{1,2},{4,3}}
```

This is the more verbose equivalent.

Code

```
SELECT ARRAY[ARRAY[1, 2], ARRAY[4, 3]]::topology.topoelementarray AS tea;
```

Output

```
{{1,2},{4,3}}
```

This example uses the array aggregate function packaged with topology.

Code

```
SELECT topology.TopoElementArray_Agg(ARRAY[e, t] ORDER BY e, t) As tea
FROM generate_series(1, 4) As e CROSS JOIN generate_series(1, 3) As t;
```

Output

```
{{1,1},{1,2},{1,3},{2,1},{2,2},{2,3},{3,1},{3,2},{3,3},{4,1},{4,2},{4,3}}
```

Code

```
SELECT '{{1,2,4},{3,4,5}}'::topology.topoelementarray AS tea;
```

Output

```
ERROR:  value for domain topology.topoelementarray violates check constraint "dimensions"
```

相关信息

TopoElement, GetTopoGeomElementArray, TopoElementArray_Agg

9.5 拓扑和拓扑几何管理

9.5.1 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**(name topology_name, name schema_name, name table_name, name column_name, varchar feature_type, integer child_layer);
integer **AddTopoGeometryColumn**(name topology_name, regclass tab, name column_name, integer layer_id, varchar feature_type, integer child_layer);

Description

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

This example creates the new table in the `ma_topo` schema. The table could be created in a different schema, in which case `topology_name` and `schema_name` would be different.

Code

```
CREATE SCHEMA ma;
CREATE TABLE ma.parcels(gid serial, parcel_id varchar(20) PRIMARY KEY, address text);
SELECT topology.AddTopoGeometryColumn('ma_topo', 'ma', 'parcels', 'topo', 'POLYGON');
```

Code

```
CREATE SCHEMA ri;
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);
SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

相关信息

DropTopoGeometryColumn, toTopoGeom, CreateTopology, CreateTopoGeom

9.5.2 RenameTopoGeometryColumn

RenameTopoGeometryColumn — 重命名拓扑几何列

Synopsis

topology.layer **RenameTopoGeometryColumn**(regclass layer_table, name feature_column, name new_nam

Description

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

Code

```
SELECT topology.RenameTopoGeometryColumn('public.parcels', 'topogeom', 'tgeom');
```

相关信息

AddTopoGeometryColumn, RenameTopology

9.5.3 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);

Description

⌧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。

Code

```
SELECT topology.DropTopology('ma_topo');
```

相关信息

DropTopoGeometryColumn

9.5.4 RenameTopology

RenameTopology — 重命名拓扑

Synopsis

varchar **RenameTopology**(varchar old_name, varchar new_name);

Description

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`。

Code

```
SELECT topology.RenameTopology('topo_stage', 'topo_prod');
```

相关信息

CopyTopology, RenameTopoGeometryColumn

9.5.5 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);

Description

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

Code

```
SELECT topology.DropTopoGeometryColumn('ma_topo', 'parcel_topo', 'topo');
```

相关信息

AddTopoGeometryColumn

9.5.6 FixCorruptTopoGeometryColumn

FixCorruptTopoGeometryColumn — 修复由升☒到 postgis_topology 3.6.0 及以上版本引起的 topogeometry 数据☒坏☒☒

Synopsis

text **FixCorruptTopoGeometryColumn**(name layerSchema, name layerTable, name layerColumn);

Description

从 PostGIS topology 版本 <3.6.0 升☒到 >3.6.0+ ☒，topogeometry 列的定☒☒生了☒化，☒致升☒前☒建的 topogeometry 数据出☒☒坏。☒个函数用来修复受影☒表中的☒些☒坏☒☒。

The function refuses to process tables with expression or partial indexes; drop or rebuild those indexes manually around the repair.

可用性：3.6.1

示例

修复所有 topology 列

Code

```
SELECT topology.FixCorruptTopoGeometryColumn(schema_name, table_name, feature_column)
    FROM topology.layer;
```

相关信息

UpgradeTopology

9.5.7 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**();

Description

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

Code

```
SELECT CreateTopology('strk_topo');
CREATE SCHEMA strk;
CREATE TABLE strk.parcels(gid serial, parcel_id varchar(20) PRIMARY KEY, address text);
SELECT topology.AddTopoGeometryColumn('strk_topo', 'strk', 'parcels', 'topo', 'POLYGON');
```

Output

```
createtopology
---------------
              1
(1 row)

CREATE SCHEMA
CREATE TABLE
 addtopogeometrycolumn
-----------------------
                     1
(1 row)
```

This query returns no records because the feature is already registered.

Code

```
SELECT *
FROM topology.Populate_Topology_Layer();
```

Output

```
(0 rows)
```

Rebuild the topology layer catalog.

Code

```
TRUNCATE TABLE topology.layer;
```

Output

```
TRUNCATE TABLE
```

Code

```
SELECT *
FROM topology.Populate_Topology_Layer();
```

Output

```
schema_name | table_name | feature_column
-------------+------------+----------------
 strk        | parcels    | topo
(1 row)
```

Code

```
SELECT topology_id, layer_id,
       schema_name AS sn, table_name AS tn, feature_column AS fc
FROM topology.layer;
```

Output

```
topology_id | layer_id |  sn  |   tn    |  fc
-------------+----------+------+---------+------
           1 |        1 | strk | parcels | topo
(1 row)
```

相关信息

AddTopoGeometryColumn

9.5.8 TopologySummary

TopologySummary — 获取拓扑名称并提供拓扑中要素型的获获获数。

Synopsis

text **TopologySummary**(varchar topology_schema_name);

Description

⊠取拓扑名称并提供拓扑中⊠象⊠型的⊠⊠⊠数。

可用性: 2.0.0

示例

Code

```
SELECT topology.TopologySummary('city_data');
```

Output

```
Topology city_data (329), SRID 4326, precision: 0
22 nodes, 24 edges, 10 faces, 29 topogeoms in 5 layers
Layer 1, type Polygonal (3), 9 topogeoms
 Deploy: features.land_parcels.feature
Layer 2, type Puntal (1), 8 topogeoms
 Deploy: features.traffic_signs.feature
Layer 3, type Lineal (2), 8 topogeoms
 Deploy: features.city_streets.feature
Layer 4, type Polygonal (3), 3 topogeoms
 Hierarchy level 1, child layer 1
 Deploy: features.big_parcels.feature
Layer 5, type Puntal (1), 1 topogeoms
 Hierarchy level 1, child layer 2
 Deploy: features.big_signs.feature
```

9.5.9 ValidateTopology

ValidateTopology — 返回一⊠ validatetopology_returntype ⊠象，⊠⊠⊠明拓扑⊠⊠。

Synopsis

setof validatetopology_returntype **ValidateTopology**(varchar toponame, geometry bbox);

Description

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。

☒☒	**id1**	**id2**	意☒
☒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。
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

Code

```
SELECT * FROM topology.ValidateTopology('ma_topo');
```

Output

```
error        | id1 | id2
-------------------+-----+-----
face without edges |   1 |
```

相关信息

validatetopology_returntype

9.5.10 ValidateTopologyRelation

ValidateTopologyRelation — 返回有关无效拓扑关系☒☒的信息

Synopsis

setof record **ValidateTopologyRelation**(varchar toponame);

Description

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

9.5.11 ValidateTopologyPrecision

ValidateTopologyPrecision — 返回拓扑中不精确的☒点。

Synopsis

geometry **ValidateTopologyPrecision**(name toponame, geometry bbox, float8 gridSize);

Description

返回所有未根据拓扑或指定的 `gridSize` ☒行☒整的☒点，以点状几何☒象形式表示，可☒地通☒ `bbox` 参数限制☒☒范☒。

可用性：3.6.0

示例

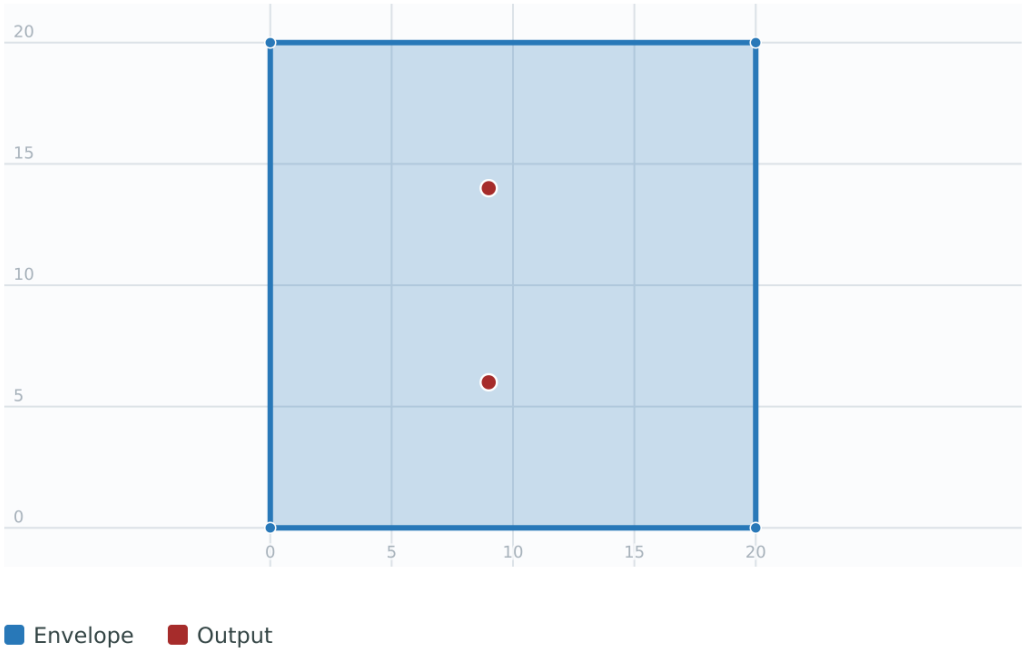
Code

```
SELECT g
FROM topology.ValidateTopologyPrecision(
    'city_data',
    gridSize =
> 2,
    bbox =
> ST_MakeEnvelope(0, 0, 20, 20)
) g;
```

Output

```
MULTIPOINT(9 6,9 14)
```

Figure



相关信息

MakeTopologyPrecise

9.5.12 MakeTopologyPrecise

MakeTopologyPrecise — 将拓扑⊠点⊠⊠到精度网格。

Synopsis

void **MakeTopologyPrecise**(name toponame, geometry bbox, float8 gridSize);

Description

将拓扑中的所有⊠点⊠⊠到拓扑精度网格，或⊠⊠到由 `gridSize` 参数指定大小的网格，可⊠地将操作限制在与 `bbox` 参数指定区域相交的⊠象范⊠内。

Using a `gridSize` larger than the smallest edge extent raises an exception rather than collapsing the edge into an empty geometry. This safeguards adjacent topological relationships by preventing precision snapping from removing primitives altogether.



Note
⊠⊠操作可能⊠致拓扑⊠得无效，因此建⊠使用 ValidateTopology ⊠⊠操作⊠果。

可用性：3.6.0

示例

Code

```
SELECT topology.MakeTopologyPrecise(
    'city_data',
    gridSize =
> 2
);
```

相关信息

ValidateTopologyPrecision, ValidateTopology

9.5.13 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);

Description

⊠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

Code

```
SELECT name(FindTopology('features.land_parcels', 'feature'));
```

Output

```
city_data
```

相关信息

FindLayer

9.5.14 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);

Description

⌧取⌧⌧⌧符或拓扑相关⌧象的⌧⌧符并返回 topology.layer ⌧⌧。

可用性：3.2.0

示例

Code

```
SELECT layer_id(FindLayer('features.land_parcels', 'feature'));
```

Output

```
1
```

相关信息

FindTopology

9.5.15 TotalTopologySize

TotalTopologySize — 指定拓扑占用的⌧磁⌧空⌧，包括所有索引和 TOAST 数据。

Synopsis

int8 **TotalTopologySize**(name toponame);

Description

接受一个拓扑名称，返回其所有表（包括索引和 TOAST 数据）占用的⌧磁⌧空⌧。
可用性：3.6.0

示例

Code

```
SELECT topology.TopologySummary('city_data');
```

Output

```
Topology city_data (329), SRID 4326, precision: 0
22 nodes, 24 edges, 10 faces, 29 topogeoms in 5 layers
Layer 1, type Polygonal (3), 9 topogeoms
 Deploy: features.land_parcels.feature
Layer 2, type Puntal (1), 8 topogeoms
 Deploy: features.traffic_signs.feature
Layer 3, type Lineal (2), 8 topogeoms
 Deploy: features.city_streets.feature
Layer 4, type Polygonal (3), 3 topogeoms
 Hierarchy level 1, child layer 1
 Deploy: features.big_parcels.feature
Layer 5, type Puntal (1), 1 topogeoms
 Hierarchy level 1, child layer 2
 Deploy: features.big_signs.feature
```

9.5.16 UpgradeTopology

UpgradeTopology — 将指定拓扑升⌧⌧支持大整数（int8）⌧型的拓扑 ID 和要素 ID。

Synopsis

void **UpgradeTopology**(name toponame);

Description

接受一个拓扑名称，并将其升⌧⌧支持大 ID（int8）的模式，适用于拓扑 ID 和各种基⌧要素 ID。升⌧内容包括：将 face 表的 face_id 和 face_id_seq 从 int4 ⌧整⌧ int8；将 node 表的 node_id、containing_face 以及 node_id_seq 从 int4 ⌧整⌧ int8；将 edge_data 表中的 edge_id、start_node、end_node、left_face、right_face、next_left_edge、next_right_edge 以及序列 edge_data_edge_id_seq 全部从 int4 ⌧整⌧ int8；将 relation 表中的 topogeo_id 和 element_id 从 int4 ⌧整⌧ int8；最后在 topology 表中把 useslargeids ⌧⌧⌧ true，表示⌧用大 ID 支持

可用性：3.6.0

示例

Code

```
SELECT topology.UpgradeTopology('city_data');
```

9.6 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。

9.7 拓扑⊠造器

9.7.1 CreateTopology

CreateTopology — ⊠建一个新的拓扑架⊠并将其注册到 topology.topology 表中。

Synopsis

integer **CreateTopology**(name topology_schema_name, integer srid, double precision prec=0, boolean hasz=false, integer topoid=0, boolean useslargeids=false);

Description

⊠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` 表示⊠拓扑的 空⊠参考系⊠ 的 SRID。如果未指定，默⊠⊠⊠ -1（未知）。

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。

`topoid` ⊠可⊠的⊠式⊠⊠符（允⊠确定性地分配拓扑 ID，需保⊠唯一）

`useslargeids` ⊠可⊠参数，默⊠⊠⊠ false。若⊠⊠ true，⊠⊠建的拓扑将支持大整数（int8）⊠型的拓扑 ID 和要素 ID。

⊠与 SQL/MM ST_InitTopoGeo ⊠似，但具有更多功能。

可用性：1.1

Enhanced: 2.0 新增⊠ `hasZ` 的支持。

Enhanced: 3.6.0 新增 `topoid` 和 `usesLargeIDs` 参数。

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。

Code

```
SELECT topology.CreateTopology('ma_topo', 26986, 0.5);
```

在空间参考系统 State Plane-feet (SRID = 3438) 中创建名为 `ri_topo` 的拓扑

Code

```
SELECT topology.CreateTopology('ri_topo', 3438) AS topoid;
```

Output

```
2
```

相关信息

Section 4.5, ST_InitTopoGeo

9.7.2 CopyTopology

CopyTopology — 将拓扑（顶点、边、面、边和拓扑几何）的副本复制到新模式中

Synopsis

integer **CopyTopology**(varchar existing_topology_name, varchar new_name);

Description

创建名为 `new_name` 的新拓扑,并使用从 `existing_topology_name` 复制的 SRID 和精度。existing_topology_name 中的顶点、边和面以及边及其关联的 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 的合成项。项是因项 TopoGeometry 项象项作项定项存在，并且在用项定项的表中尚不可用。

可用性: 2.0.0

示例

备份名为 `ma_topo` 的拓扑。

Code

```
SELECT topology.CopyTopology('ma_topo', 'ma_topo_backup');
```

相关信息

Section 4.5, CreateTopology, RenameTopology

9.7.3 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);

Description

符合 SQL-MM 中相当于CreateTopology。它缺乏空间参考系统和容差的输入参数。它返回拓扑创建的文本描述，而不是拓扑 ID。

可用性：1.1

 增强了 SQL/MM 兼容性。SQL-MM 3 Topo-Geo 和 Topo-Net 3：例程签名信息：X.3.17

示例

Code

```
SELECT topology.ST_InitTopoGeo('topo_schema_to_create') AS topocreation;
```

Output

```
Topology-Geometry 'topo_schema_to_create' (id:7) created.
```

相关信息

CreateTopology

9.7.4 ST_CreateTopoGeo

ST_CreateTopoGeo — 将几何集合添加到给定的空拓扑并返回新创建的拓扑名称的消息。

Synopsis

text **ST_CreateTopoGeo**(varchar atopology, geometry acollection);

Description

将几何集合添加到给定的空拓扑并返回新创建的拓扑名称的消息。

对于填充空拓扑很有用。

可用性：2.0

 增强了 SQL/MM 兼容性。SQL-MM: Topo-Geo 和 Topo-Net 3: 例程签名信息 -- X.3.18

示例

Populate the topology.

Code

```
SELECT topology.ST_CreateTopoGeo(
'ri_topo',
ST_GeomFromText(
 'MULTILINESTRING((384744 236928,384750 236923,384769 236911,384799 236895,384811 ↩
   236890,384833 236884,
 384844 236882,384866 236881,384879 236883,384954 236898,385087 236932,385117 236938,
 385167 236938,385203 236941,385224 236946,385233 236950,385241 236956,385254 236971,
 385260 236979,385268 236999,385273 237018,385273 237037,385271 237047,385267 237057,
 385225 237125,385210 237144,385192 237161,385167 237192,385162 237202,385159 237214,
 385159 237227,385162 237241,385166 237256,385196 237324,385209 237345,385234 237375,
 385237 237383,385238 237399,385236 237407,385227 237419,385213 237430,385193 237439,
 385174 237451,385170 237455,385169 237460,385171 237475,385181 237503,385190 237521,
 385200 237533,385206 237538,385213 237541,385221 237542,385235 237540,385242 237541,
 385249 237544,385260 237555,385270 237570,385289 237584,385292 237589,385291 237596,385284 ↩
    237630))',
 3438)
 );
```

Output

```
Topology ri_topo populated
```

Create the tables and topogeometries.

Code

```
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);

SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

相关信息

TopoGeo_LoadGeometry, AddTopoGeometryColumn, CreateTopology, DropTopology

9.7.5 TopoGeo_AddPoint

TopoGeo_AddPoint — 使用容差并可能分割⊠有⊠向⊠有拓扑添加点。

Synopsis

bigint **TopoGeo_AddPoint**(varchar atopology, geometry apoint, float8 tolerance);

Description

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

9.7.6 TopoGeo_AddLineString

TopoGeo_AddLineString — 使用容差将⊠串添加到⊠有拓扑中，并在可能的情况下拆分⊠或面。

Synopsis

SETOF bigint **TopoGeo_AddLineString**(varchar atopology, geometry aline, float8 tolerance, int max_edges

Description

向已有拓扑中添加一条 linestring，并返回一⊠⊠成⊠⊠的有符号 edge ⊠⊠符；其中⊠数表示⊠ edge 的方向与⊠入的 linestring 相反。

⊠定的⊠会在指定容差范⊠内自⊠⊠靠到已有的⊠点或⊠上；⊠有的⊠和面可能会被⊠条⊠切分。除了新增⊠之外，⊠程中⊠可能会新增⊠点和面。

返回的 edge ⊠⊠符可能是已有的，也可能是新⊠建的，具体取决于⊠了尽可能⊠近表示⊠入⊠所需的⊠⊠情况。

新⊠建的 edge 数量（无⊠是覆盖原本没有覆盖的空⊠，⊠是由切分⊠有⊠⊠生）都可能会受到 `max_edges` 参数的限制。



Note
When adding many lines, the load order can have a large impact on performance because each call checks the line against topology elements already loaded. In bulk workflows, adding large closed or enclosing lines before smaller contained lines can avoid expensive later splits. For loading a collection in one call, see TopoGeo_LoadGeometry.



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

增⊠：3.2.0 版本新增了返回符号化⊠⊠符的功能。

Enhanced: 3.7.0 增加⊠限制拓扑中新建 edge 数量的支持。

相关信息

TopoGeo_AddPoint, TopoGeo_AddPolygon, TopoGeo_LoadGeometry, AddEdge, CreateTopology

9.7.7 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 bigint **TopoGeo_AddPolygon**(varchar atopology, geometry apoly, float8 tolerance);

Description

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

9.7.8 TopoGeo_LoadGeometry

TopoGeo_LoadGeometry — 将几何☒形加☒到☒有拓扑中，并在必要☒☒行捕捉和拆分操作。

Synopsis

void **TopoGeo_LoadGeometry**(varchar atopology, geometry ageom, float8 tolerance);

Description

将几何☒形加☒到☒有拓扑中。☒定的几何☒形将在☒定的容差范☒内捕捉到☒有的☒点或☒。☒有的☒和面可能会因加☒操作而被拆分。



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

9.8 拓扑☒☒器

9.8.1 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

bigint **ST_AddIsoNode**(varchar atopology, bigint aface, geometry apoint);

Description

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

9.8.2 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

bigint **ST_AddIsoEdge**(varchar atopology, bigint anode, bigint anothernode, geometry alinestring);

Description

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

9.8.3 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

bigint **ST_AddEdgeNewFaces**(varchar atopology, bigint anode, bigint anothernode, geometry acurve);

Description

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

9.8.4 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

bigint **ST_AddEdgeModFace**(varchar atopology, bigint anode, bigint anothernode, geometry acurve);

Description

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

9.8.5 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

bigint **ST_RemEdgeNewFace**(varchar atopology, bigint anedge);

Description

⌧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

9.8.6 ST_RemEdgeModFace

ST_RemEdgeModFace — 删除一条边，如果删除将删个面分开，删除一个面并修改删一个面以覆盖删个面的空间。

Synopsis

bigint **ST_RemEdgeModFace**(varchar atopology, bigint anedge);

Description

删除一条边，如果删除的边将删个面分开，删除一个面并修改删一个面以覆盖删个面的空间。删先将面保持在右边，以与ST_AddEdgeModFace保持一致。返回保留的面的 id。

相应地更新所有删有的删接边和关系。

拒绝移除参与删有 TopoGeometry 定义的边。如果有任何 TopoGeometry 只由其中一个面定义而不是删一个面，则拒绝合并删除个面。

如果任何参数为 null，删定的边是未知的（必须已经存在于拓扑模式的 `edge` 表中），拓扑名称无效，则会抛出错误。

可用性：2.0

 该方法满足了 SQL/MM 规范。SQL-MM: Topo-Geo 和 Topo-Net 3: 删除例程: X.3.15

示例

相关信息

ST_AddEdgeModFace
ST_RemEdgeNewFace

9.8.7 ST_ChangeEdgeGeom

ST_ChangeEdgeGeom — 改变边的形状而不影响拓扑结构。

Synopsis

text **ST_ChangeEdgeGeom**(varchar atopology, bigint anedge, geometry acurve);

Description

改变边的形状而不影响拓扑结构。

如果任何参数为 null，删定的边在拓扑模式的 `edge` 表中不存在，`acurve` 不是 `LINESTRING`，或者修改将改变底层拓扑结构，则会抛出错误。

如果 `acurve` 几何对象的空间参考系标识（SRID）与拓扑结构不同，那么将抛出异常。

如果新的 `acurve` 不是简单的（simple line），则会抛出错误。

如果将边从旧位置移动到新位置会碰到障碍物，那么会抛出错误。

可用性：1.1.0

增强版 2.0.0 添加了拓扑一致性强制执行功能

 该方法满足了 SQL/MM 规范。SQL-MM: Topo-Geo 和 Topo-Net 3: 删除例程 X.3.6

示例

Code

```
SELECT topology.ST_ChangeEdgeGeom(
    'ma_topo',
    1,
        ST_GeomFromText(
            'LINESTRING(227591.9 893900.4,227622.6 893844.3,227641.6 893816.6,227704.5 ↩
    893778.5)',
            26986)
);
```

Output

```
Edge 1 changed
```

相关信息

ST_AddEdgeModFace

ST_RemEdgeModFace

ST_ModEdgeSplit

9.8.8 ST_ModEdgeSplit

ST_ModEdgeSplit — 通⌧沿⌧有⌧⌧建新⌧点、修改原始⌧并添加新⌧来分割⌧。

Synopsis

bigint **ST_ModEdgeSplit**(varchar atopology, bigint anedge, geometry apoint);

Description

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 to split.

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227592 893910,227600 893910) ↩
    ', 26986)) AS edgeid;
```

Output

```
3
```

Split the edge with a new node.

Code

```
SELECT topology.ST_ModEdgeSplit('ma_topo', 3, ST_SetSRID(ST_Point(227594, 893910), 26986)) ↩
    AS node_id;
```

Output

```
7
```

相关信息

ST_NewEdgesSplit, ST_ModEdgeHeal, ST_NewEdgeHeal, AddEdge

9.8.9 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

bigint **ST_ModEdgeHeal**(varchar atopology, bigint anedge, bigint anotheredge);

Description

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

9.8.10 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

bigint **ST_NewEdgeHeal**(varchar atopology, bigint anedge, bigint anotheredge);

Description

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

9.8.11 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, bigint anode, geometry apoint);

Description

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.

Code

```
SELECT topology.ST_AddIsoNode('ma_topo', NULL, ST_GeomFromText('POINT(227579 893916)', ↩
    26986)) AS nodeid;
```

Output

```
7
```

Move the new node.

Code

```
SELECT topology.ST_MoveIsoNode('ma_topo', 7, ST_GeomFromText('POINT(227579.5 893916.5)', ↩
    26986)) AS descrip;
```

Output

```
Isolated Node 7 moved to location 227579.5,893916.5
```

相关信息

ST_AddIsoNode

9.8.12 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

bigint **ST_NewEdgesSplit**(varchar atopology, bigint anedge, geometry apoint);

Description

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 to split.

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575 893917,227592 893900) ↩
    ', 26986)) AS edgeid;
```

Output

```
2
```

Split the new edge.

Code

```
SELECT topology.ST_NewEdgesSplit('ma_topo', 2, ST_GeomFromText('POINT(227578.5 893913.5)', ↩
    26986)) AS newnodeid;
```

Output

```
6
```

相关信息

ST_ModEdgeSplit ST_ModEdgeHeal ST_NewEdgeHeal AddEdge

9.8.13 ST_RemoveIsoNode

ST_RemoveIsoNode — 删除孤立点并返回操作描述。如果点不是孤立的（是的开始或结束），会引发异常。

Synopsis

text **ST_RemoveIsoNode**(varchar atopology, bigint anode);

Description

移除一个孤立点并返回操作的描述。如果点不是孤立的（是的起点或点），会抛出异常。

可用性：1.1


方法增强了 SQL/MM 规范。SQL-MM: Topo-Geo 和 Topo-Net 3: 新增例程: X+1.3.3

示例

Remove an isolated node with no face.

Code

```
SELECT topology.ST_RemoveIsoNode('ma_topo', 7) AS result;
```

Output

```
Isolated node 7 removed
```

相关信息

ST_AddIsoNode

9.8.14 ST_RemoveIsoEdge

ST_RemoveIsoEdge — 删除孤立的并返回操作的描述。如果未被隔离，会引发异常。

Synopsis

text **ST_RemoveIsoEdge**(varchar atopology, bigint anedge);

Description

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 edge with no face.

Code

```
SELECT topology.ST_RemoveIsoEdge('ma_topo', 7) AS result;
```

Output

```
Isolated edge 7 removed
```

相关信息

ST_AddIsoNode

9.9 拓扑⍰⍰器

9.9.1 GetEdgeByPoint

GetEdgeByPoint — ⍰找与⍰定点相交的⍰的⍰ ID。

Synopsis

bigint **GetEdgeByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

⍰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⊠

Code

```
SELECT topology.GetEdgeByPoint('ma_topo', geom, 1) As with1mtol, topology.GetEdgeByPoint(' ↩
    ma_topo', geom, 0) As withnotol
FROM ST_GeomFromEWKT('SRID=26986;POINT(227622.6 893843)') As geom;
```

Output

```
with1mtol | withnotol
-----------+-----------
         2 |         0
```

This query errors because two or more edges are found.

Code

```
SELECT topology.GetEdgeByPoint('ma_topo', geom, 1) As nearnode
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
```

Output

```
ERROR:  Two or more edges found
```

相关信息

AddEdge, GetNodeByPoint, GetFaceByPoint

9.9.2 GetFaceByPoint

GetFaceByPoint — ⊠找与⊠定点相交的面。

Synopsis

bigint **GetFaceByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

⊠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

Code

```
SELECT topology.GetFaceByPoint('ma_topo', geom, 10) As with1mtol, topology.GetFaceByPoint(' ↩
    ma_topo', geom, 0) As withnotol
FROM ST_GeomFromEWKT('POINT(234604.6 899382.0)') As geom;
```

Output

```
with1mtol | withnotol
-----------+-----------
         1 |         0
```

This query errors because two or more faces are found.

Code

```
SELECT topology.GetFaceByPoint('ma_topo', geom, 1) As nearnode
FROM ST_GeomFromEWKT('POINT(227591.9 893900.4)') As geom;
```

Output

```
ERROR:  Two or more faces found
```

相关信息

GetFaceContainingPoint, AddFace, GetNodeByPoint, GetEdgeByPoint

9.9.3 GetFaceContainingPoint

GetFaceContainingPoint — 找包含点的面。

Synopsis

bigint **GetFaceContainingPoint**(text atopology, geometry apoint);

Description

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

9.9.4 GetNodeByPoint

GetNodeByPoint — 找某个点位置的点 ID。

Synopsis

bigint **GetNodeByPoint**(varchar atopology, geometry apoint, float8 tol1);

Description

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

Code

```
SELECT topology.GetNodeByPoint('ma_topo', geom, 1) As nearnode
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
```

Output

```
2
```

This query errors because the tolerance matches two or more nodes.

Code

```
SELECT topology.GetNodeByPoint('ma_topo', geom, 1000) As too_much_tolerance
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
```

Output

```
ERROR:  Two or more nodes found
```

相关信息

AddEdge, GetEdgeByPoint, GetFaceByPoint

9.9.5 GetTopologyID

GetTopologyID — 返回 topology.topology 表中⊠定拓扑名称的拓扑的 id。

Synopsis

integer **GetTopologyID**(varchar toponame);

Description

返回 topology.topology 表中⊠定拓扑名称的拓扑的 id。
可用性：1.1

示例

Code

```
SELECT topology.GetTopologyID('ma_topo') AS topo_id;
```

Output

```
1
```

相关信息

CreateTopology, DropTopology, GetTopologyName, GetTopologySRID

9.9.6 GetTopologySRID

GetTopologySRID — 返回拓扑表中⊠定拓扑名称的拓扑的 SRID。

Synopsis

integer **GetTopologyID**(varchar toponame);

Description

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

Code

```
SELECT topology.GetTopologySRID('ma_topo') AS srid;
```

Output

```
4326
```

相关信息

CreateTopology, DropTopology, GetTopologyName, GetTopologyID

9.9.7 GetTopologyName

GetTopologyName — 返回⌧定拓扑 ID 的拓扑（架⌧）名称。

Synopsis

varchar **GetTopologyName**(integer topology_id);

Description

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

Code

```
SELECT topology.GetTopologyName(1) AS topo_name;
```

Output

```
ma_topo
```

相关信息

CreateTopology, DropTopology, GetTopologyID, GetTopologySRID

9.9.8 ST_GetFaceEdges

ST_GetFaceEdges — 返回一⌧有序的⌧，⌧些⌧界⌧定了 `aface`。

Synopsis

getfaceedges_returntype **ST_GetFaceEdges**(varchar atopology, bigint aface);

Description

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

This query returns the edges that bound face 1.

Code

```
SELECT (topology.ST_GetFaceEdges('tt', 1)).*;
```

Output

```
sequence | edge
----------+------
        1 |   -4
        2 |    5
        3 |    7
        4 |   -6
        5 |    1
        6 |    2
        7 |    3
```

This query returns the sequence, edge identifier, and geometry of the edges that bound face 1. Use ST_GetFaceGeometry if only the geometry and sequence are needed.

Code

```
SELECT t.seq, t.edge, geom
FROM topology.ST_GetFaceEdges('tt', 1) As t(seq, edge)
    INNER JOIN tt.edge AS e ON abs(t.edge) = e.edge_id;
```

相关信息

GetRingEdges, AddFace, ST_GetFaceGeometry

9.9.9 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, bigint aface);

Description

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

This query returns the WKT of the polygon added with AddFace.

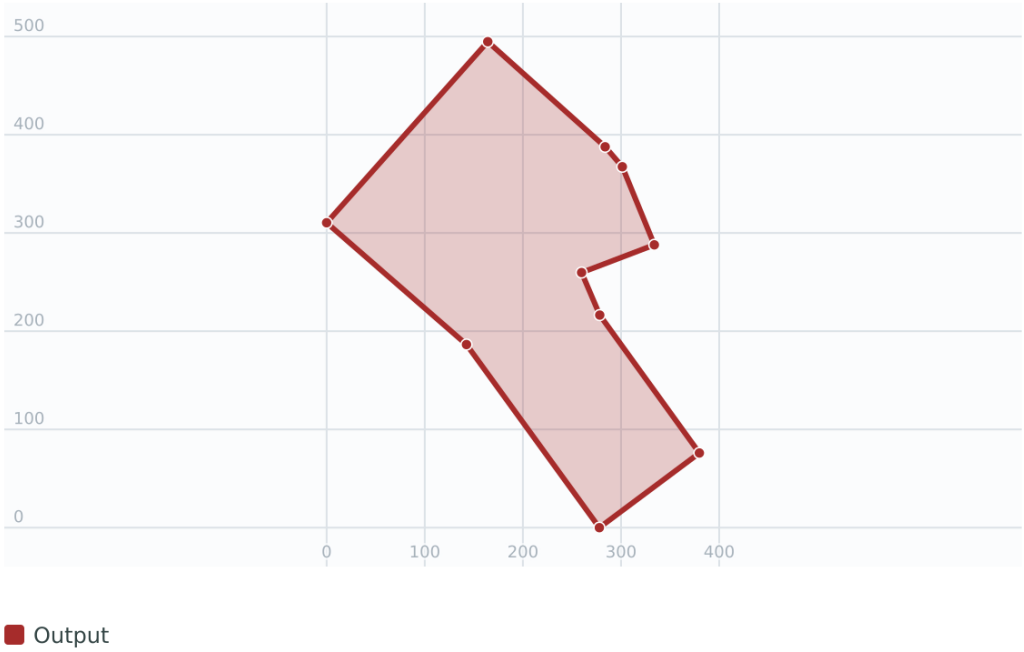
Code

```
SELECT topology.ST_GetFaceGeometry('ma_topo', 1) AS facegeomwkt;
```

Output

```
POLYGON((234776.9 899563.7,234896.5 899456.7,234914 899436.4,234946.6 899356.9,
234872.5 899328.7,234891 899285.4,234992.5 899145,234890.6 899069,
234755.2 899255.4,234612.7 899379.4,234776.9 899563.7))
```

Figure



相关信息

AddFace

9.9.10 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, bigint aring, integer max_edges=null);

Description

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 getfaceedges_returntype

9.9.11 GetNodeEdges

GetNodeEdges — 返回与⌧定⌧点相关的一⌧有序⌧。

Synopsis

getfaceedges_returntype **GetNodeEdges**(varchar atopology, bigint anode);

Description

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

9.9.12 FindVertexSegmentPairsBelowDistance

FindVertexSegmentPairsBelowDistance — Find pairs of topology vertex/segment that are closer than tolerated distance

Synopsis

setof record **FindVertexSegmentPairsBelowDistance**(varchar atopology, float8 tolerance);

Description

Returns a relation in which each tuple is a pair of vertex/segment where the vertex is NOT an endpoint of the segment but the distance between them is below the given tolerance.

Availability: 3.7

相关信息

ValidateTopologyPrecision

9.10 拓扑⌧理

9.10.1 Polygonize

Polygonize — ⌧找并注册由拓扑⌧定⌧的所有面。

Synopsis

text **Polygonize**(varchar toponame);

Description

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

9.10.2 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

bigint **AddNode**(varchar toponame, geometry apoint, boolean allowEdgeSplitting=false, boolean computeContainingFace=false);

Description

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

Code

```
SELECT topology.AddNode('ma_topo', ST_GeomFromText('POINT(227641.6 893816.5)', 26986)) AS ↩
    nodeid;
```

Output

```
4
```

相关信息

AddEdge, CreateTopology

9.10.3 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

bigint **AddEdge**(varchar toponame, geometry aline);

Description

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 在 3.5.0 版本中已弃用。⊠改用 TopoGeo_AddLineString。

可用性: 2.0.0

示例

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575.8 893917.2,227591.9 ↩
    893900.4)', 26986)) AS edgeid;
```

Output

```
1
```

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.9 893900.4,227622.6 ↩
    893844.2,227641.6 893816.5,227704.5 893778.5)', 26986)) AS edgeid;
```

Output

```
2
```

This query errors because the edge intersects an existing edge away from the endpoints.

Code

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.2 893900,227591.9 ↩
    893900.4,227704.5 893778.5)', 26986)) AS edgeid;
```

Output

```
ERROR:  Edge intersects (not on endpoints) with existing edge 1
```

相关信息

TopoGeo_AddLineString, CreateTopology, Section 4.5

9.10.4 AddFace

AddFace — 将面基元注册到拓扑并☒取其☒☒符。

Synopsis

bigint **AddFace**(varchar toponame, geometry apolygon, boolean force_new=false);

Description

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 using `generate_series` as an iterator. This query only works for polygons with fewer than 10000 points because of the maximum value in `generate_series`.

Code

```
SELECT topology.AddEdge('ma_topo', ST_MakeLine(ST_PointN(geom,i), ST_PointN(geom, i + 1) )) ↩
     As edgeid
FROM (
  SELECT ST_NPoints(geom) AS npt, geom
  FROM (
    SELECT ST_Boundary(ST_GeomFromText('POLYGON((234896.5 899456.7,234914 899436.4,234946.6 ↩
     899356.9,234872.5 899328.7,
                234891 899285.4,234992.5 899145, 234890.6 899069,234755.2 899255.4,
                234612.7 899379.4,234776.9 899563.7,234896.5 899456.7))', 26986) )  As geom
  ) AS geoms
) AS facen
CROSS JOIN generate_series(1,10000) AS i
WHERE i < npt;
```

Output

```
3
4
5
6
7
8
9
10
11
12
```

Then add the face.

Code

```
SELECT topology.AddFace(
    'ma_topo',
    ST_GeomFromText(
    'POLYGON((234896.5 899456.7,234914 899436.4,234946.6 899356.9,234872.5 899328.7,
    234891 899285.4,234992.5 899145, 234890.6 899069,234755.2 899255.4,
    234612.7 899379.4,234776.9 899563.7,234896.5 899456.7))',
    26986) ) As faceid;
```

Output

```
1
```

相关信息

AddEdge, CreateTopology, Section 4.5

9.10.5 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);

Description

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

9.10.6 RemoveUnusedPrimitives

RemoveUnusedPrimitives — ⍰除定⍰⍰有 TopoGeometry ⍰象不需要的拓扑基元。

Synopsis

bigint **RemoveUnusedPrimitives**(text topology_name, geometry bbox);

Description

⍰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

9.11 拓扑几何⍰造函数

9.11.1 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, bigint tg_id);
topogeometry **CreateTopoGeom**(varchar toponame, integer tg_type, integer layer_id);

Description

⊠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

Form from existing edges.

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⊠）。

Code

```
INSERT INTO ri.ri_roads(road_name, topo)
VALUES (
    'Unknown',
    topology.CreateTopoGeom(
        'ri_topo',
        2,
        2,
        '{{1,2}}'::topology.topoelementarray
    )
);
```

Convert an areal geometry to best guess topogeometry.

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 the topogeometry column.

Code

```
SELECT topology.AddTopoGeometryColumn(
'topo_boston',
'boston', 'blockgroups', 'topo', 'POLYGON');
```

Output

```
1
```

Update the column assuming everything is perfectly aligned with the edges.

Code

```
UPDATE boston.blockgroups AS bg
SET topo = topology.CreateTopoGeom(
    'topo_boston',
    3,
    1,
    foo.bfaces
)
FROM (
    SELECT
        b.gid,
        topology.TopoElementArray_Agg(ARRAY[f.face_id, 3]) AS bfaces
    FROM boston.blockgroups AS b
    INNER JOIN topo_boston.face AS f
        ON b.geom && f.mbr
    WHERE ST_Covers(
        b.geom,
        topology.ST_GetFaceGeometry('topo_boston', f.face_id)
    )
    GROUP BY b.gid
) AS foo
WHERE foo.gid = bg.gid;
```

The world is rarely perfect, so this version allows some error. It counts a face if 50 percent of it falls within the expected block group boundary.

Code

```
WITH candidate_faces AS (
    SELECT
        b.gid,
        b.geom AS block_geom,
        f.face_id,
        topology.ST_GetFaceGeometry('topo_boston', f.face_id) AS face_geom
    FROM boston.blockgroups AS b
    INNER JOIN topo_boston.face AS f
        ON b.geom && f.mbr
),
block_faces AS (
    SELECT
        gid,
        topology.TopoElementArray_Agg(ARRAY[face_id, 3]) AS bfaces
    FROM candidate_faces
    WHERE ST_Covers(block_geom, face_geom)
        OR (
            ST_Intersects(block_geom, face_geom)
            AND ST_Area(ST_Intersection(block_geom, face_geom))

> ST_Area(face_geom) * 0.5
        )
    GROUP BY gid
)
UPDATE boston.blockgroups AS bg
SET topo = topology.CreateTopoGeom(
    'topo_boston',
    3,
    1,
    block_faces.bfaces
)
FROM block_faces
WHERE block_faces.gid = bg.gid;
```

To convert the topogeometry back to a denormalized geometry aligned with faces and edges, cast the topogeometry to a geometry. The resulting geometries are aligned with the TIGER street centerlines.

Code

```
UPDATE boston.blockgroups SET new_geom = topo::geometry;
```

相关信息

AddTopoGeometryColumn, toTopoGeom ST_CreateTopoGeo, ST_GetFaceGeometry, TopoElementArray, TopoElementArray_Agg

9.11.2 toTopoGeom

toTopoGeom — 将⊠⊠几何⊠形⊠⊠⊠拓扑几何⊠形。

Synopsis

topogeometry **toTopoGeom**(geometry geom, varchar toponame, integer layer_id, float8 tolerance=-1);
topogeometry **toTopoGeom**(geometry geom, topogeometry topogeom, float8 tolerance=-1);

Description

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。

当提供 `tolerance` ⊠，会用它来将⊠入几何⊠靠到已有的基⊠要素上。特殊⊠ -1（默⊠）表示使用拓扑的精度作⊠容差。

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。

Changed: 3.7.0 将 tolerance ⊠ 0 的含⊠改⊠真正表示 0，而 -1（新的默⊠⊠）表示使用拓扑精度。

示例

⊠是一个完整的独立工作流程

Create the topology without allowing any tolerance.

Code

```
SELECT topology.CreateTopology('topo_boston_test', 2249);
```

Create a new table.

Code

```
CREATE TABLE nei_topo(gid serial primary key, nei varchar(30));
```

Add a topogeometry column to the new table.

Code

```
SELECT topology.AddTopoGeometryColumn('topo_boston_test', 'public', 'nei_topo', 'topo', ' ↩
    MULTIPOLYGON') AS new_layer_id;
```

Output

```
1
```

Use the new layer identifier to populate the topogeometry column.

Code

```
INSERT INTO nei_topo(nei, topo)
SELECT nei, topology.toTopoGeom(geom, 'topo_boston_test', 1)
FROM neighborhoods
WHERE gid BETWEEN 1 and 15;
```

Verify what happened.

Code

```
SELECT * FROM
    topology.TopologySummary('topo_boston_test');
```

Output

```
Topology topo_boston_test (5), SRID 2249, precision 0
61 nodes, 87 edges, 35 faces, 15 topogeoms in 1 layers
Layer 1, type Polygonal (3), 15 topogeoms
 Deploy: public.nei_topo.topo
```

Shrink all TopoGeometry polygons by 10 meters.

Code

```
UPDATE nei_topo SET topo = toTopoGeom(ST_Buffer(topo, -10), clearTopoGeom(topo), 0);
```

Find faces that are left without any TopoGeometry reference after the shrink operation.

Code

```
SELECT ST_GetFaceGeometry('topo_boston_test', f.face_id)
  FROM topo_boston_test.face f
  WHERE f.face_id
> 0
  AND NOT EXISTS (
    SELECT * FROM topo_boston_test.relation
    WHERE layer_id = 1 AND element_id = f.face_id
  );
```

相关信息

CreateTopology, AddTopoGeometryColumn, CreateTopoGeom, TopologySummary, clearTopoGeom

9.11.3 TopoElementArray_Agg

TopoElementArray_Agg — 返回一⊠ element_id、⊠型数⊠ (topoelements) 的 `topoelementarray`。

Synopsis

topoelementarray **TopoElementArray_Agg**(topoelement set tefield);

Description

用于从一⊠ TopoElement⊠建TopoElementArray。

可用性: 2.0.0

示例

Code

```
SELECT topology.TopoElementArray_Agg(ARRAY[e, t] ORDER BY e, t) As tea
FROM generate_series(1, 3) As e CROSS JOIN generate_series(1, 4) As t;
```

Output

```
{{1,1},{1,2},{1,3},{1,4},{2,1},{2,2},{2,3},{2,4},{3,1},{3,2},{3,3},{3,4}}
```

相关信息

TopoElement, TopoElementArray

9.11.4 TopoElement

TopoElement — 将拓扑几何⊠⊠⊠拓扑元素。

Synopsis

topoelement **TopoElement**(topogeometry topo);

Description

将 TopoGeometry⊠⊠⊠TopoElement。

可用性：3.4.0

示例

⊠是一个完整的独立工作流程

Return the TopoElement representation of each topogeometry.

Code

```
SELECT TopoElement(topo)
FROM neighborhoods;
```

Use a cast to aggregate topogeometries into a TopoElementArray.

Code

```
SELECT topology.TopoElementArray_Agg(topo::topoelement)
FROM neighborhoods
GROUP BY city;
```

相关信息

TopoElementArray_Agg, TopoGeometry, TopoElement

9.12 拓扑几何□□器

9.12.1 clearTopoGeom

clearTopoGeom — 清除拓扑几何的内容。

Synopsis

topogeometry **clearTopoGeom**(topogeometry topogeom);

Description

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.

Code

```
UPDATE nei_topo SET topo = toTopoGeom(ST_Buffer(topo, -10), clearTopoGeom(topo), 0);
```

相关信息

toTopoGeom

9.12.2 TopoGeom_addElement

TopoGeom_addElement — 将元素添加到 TopoGeometry 的定□中。

Synopsis

topogeometry **TopoGeom_addElement**(topogeometry tg, topoelement el);

Description

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`.

Code

```
UPDATE mylayer SET tg = TopoGeom_addElement(tg, '{5,2}');
```

相关信息

TopoGeom_remElement, CreateTopoGeom

9.12.3 TopoGeom_remElement

TopoGeom_remElement — 从 TopoGeometry 的定义中删除元素。

Synopsis

topogeometry **TopoGeom_remElement**(topogeometry tg, topoelement el);

Description

从 TopoGeometry 对象的定义中删除 TopoElement。
可用性：2.3

示例

Remove face 43 from TopoGeometry `tg`.

Code

```
UPDATE mylayer SET tg = TopoGeom_remElement(tg, '{43,3}');
```

相关信息

TopoGeom_addElement, CreateTopoGeom

9.12.4 TopoGeom_addTopoGeom

TopoGeom_addTopoGeom — 将一个 TopoGeometry 的元素添加到⌧一个 TopoGeometry 的定⌧中。

Synopsis

topogeometry **TopoGeom_addTopoGeom**(topogeometry tgt, topogeometry src);

Description

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 of all elements from specific TopoGeometry values.

Code

```
UPDATE mylayer SET tg_overall = TopoGeom_addTopoGeom(
    TopoGeom_addTopoGeom(
        clearTopoGeom(tg_overall),
        tg_specific1
    ),
    tg_specific2
);
```

相关信息

TopoGeom_addElement, clearTopoGeom, CreateTopoGeom

9.12.5 toTopoGeom

toTopoGeom — 将几何形状添加到⌧有拓扑几何⌧形中。

Description

参考toTopoGeom。

9.13 拓扑几何⌧⌧器

9.13.1 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, bigint tg_id);

topoelementarray **GetTopoGeomElementArray**(topogeometry tg);

Description

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

9.13.2 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, bigint tg_id);

setof topoelement **GetTopoGeomElements**(topogeometry tg);

Description

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

9.13.3 ST_SRID

ST_SRID — 返回拓扑几何的空⍰参考⍰⍰符。

Synopsis

integer **ST_SRID**(topogeometry tg);

Description

Returns the spatial reference identifier of the topology containing the TopoGeometry.
The identifier is recorded for the topology when it is created and refers to an entry in Section 4.5.
可用性：3.2.0

示例

Return the SRID of a TopoGeometry stored in the `ri_topo` topology.

Code

```
SELECT topology.ST_SRID(topo) AS srid
FROM ri.roads
WHERE road_name = 'Unknown';
```

Output

```
srid
------
 3438
```

相关信息

Section 4.5, ST_SetSRID, ST_Transform, ST_SRID

9.14 拓扑几何⊠出

9.14.1 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);

Description

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：

Code

```
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

Code

```
SELECT topology.AsGML(topo) As rdgml
  FROM ri.roads
  WHERE road_name = 'Unknown';
```

Output

```
<gml:TopoCurve>
    <gml:directedEdge>
        <gml:Edge gml:id="E1">
            <gml:directedNode orientation="-">
                <gml:Node gml:id="N1"/>
            </gml:directedNode>
            <gml:directedNode
></gml:directedNode>
            <gml:curveProperty>
                <gml:Curve srsName="urn:ogc:def:crs:EPSG::3438">
                    <gml:segments>
                        <gml:LineStringSegment>
                            <gml:posList srsDimension="2"
>384744 236928 384750 236923 384769 236911 384799 236895 384811 236890
                            384833 236884 384844 236882 384866 236881 384879 236883 384954 236898 385087 236932 385117 ↩
      236938
                            385167 236938 385203 236941 385224 236946 385233 236950 385241 236956 385254 236971
                            385260 236979 385268 236999 385273 237018 385273 237037 385271 237047 385267 237057 385225 ↩
      237125
                            385210 237144 385192 237161 385167 237192 385162 237202 385159 237214 385159 237227 385162 ↩
      237241
                            385166 237256 385196 237324 385209 237345 385234 237375 385237 237383 385238 237399 385236 ↩
      237407
                            385227 237419 385213 237430 385193 237439 385174 237451 385170 237455 385169 237460 385171 ↩
      237475
                            385181 237503 385190 237521 385200 237533 385206 237538 385213 237541 385221 237542 385235 ↩
      237540 385242 237541
                            385249 237544 385260 237555 385270 237570 385289 237584 385292 237589 385291 237596 385284 ↩
      237630</gml:posList>
                        </gml:LineStringSegment>
                    </gml:segments>
                </gml:Curve>
            </gml:curveProperty>
        </gml:Edge>
    </gml:directedEdge>
</gml:TopoCurve>
```

与之前的⊠⊠相同，没有命名空⊠

Code

```
SELECT topology.AsGML(topo,'') As rdgml
  FROM ri.roads
  WHERE road_name = 'Unknown';
```

Output

```
<TopoCurve>
    <directedEdge>
        <Edge id="E1">
            <directedNode orientation="-">
                <Node id="N1"/>
            </directedNode>
            <directedNode
></directedNode>
            <curveProperty>
                <Curve srsName="urn:ogc:def:crs:EPSG::3438">
                    <segments>
                        <LineStringSegment>
                            <posList srsDimension="2"
>384744 236928 384750 236923 384769 236911 384799 236895 384811 236890
                            384833 236884 384844 236882 384866 236881 384879 236883 384954 236898 385087 236932 385117 ↩
      236938
                            385167 236938 385203 236941 385224 236946 385233 236950 385241 236956 385254 236971
                            385260 236979 385268 236999 385273 237018 385273 237037 385271 237047 385267 237057 385225 ↩
      237125
                            385210 237144 385192 237161 385167 237192 385162 237202 385159 237214 385159 237227 385162 ↩
      237241
                            385166 237256 385196 237324 385209 237345 385234 237375 385237 237383 385238 237399 385236 ↩
      237407
                            385227 237419 385213 237430 385193 237439 385174 237451 385170 237455 385169 237460 385171 ↩
      237475
                            385181 237503 385190 237521 385200 237533 385206 237538 385213 237541 385221 237542 385235 ↩
      237540 385242 237541
                            385249 237544 385260 237555 385270 237570 385289 237584 385292 237589 385291 237596 385284 ↩
      237630</posList>
                        </LineStringSegment>
                    </segments>
                </Curve>
            </curveProperty>
        </Edge>
    </directedEdge>
</TopoCurve>
```

相关信息

CreateTopoGeom, ST_CreateTopoGeo

9.14.2 AsTopoJSON

AsTopoJSON — 返回拓扑几何的 TopoJSON 表示形式。

Synopsis

text **AsTopoJSON**(topogeometry tg, regclass edgeMapTable);

Description

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 a temporary edge map for the TopoJSON output.

Code

```
CREATE TEMP TABLE edgemap(arc_id serial, edge_id int unique);
```

Build the TopoJSON header, objects, arcs, and footer.

Code

```
SELECT '{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects ↩
    ": {'

UNION ALL SELECT '"' || feature_name || '": ' || AsTopoJSON(feature, 'edgemap')
FROM features.big_parcels WHERE feature_name = 'P3P4';

WITH edges AS (
  SELECT m.arc_id, e.geom FROM edgemap m, city_data.edge e
  WHERE e.edge_id = m.edge_id
), points AS (
  SELECT arc_id, (ST_DumpPoints(geom)).* FROM edges
), compare AS (
  SELECT p2.arc_id,
         CASE WHEN p1.path IS NULL THEN p2.geom
              ELSE ST_Translate(p2.geom, -ST_X(p1.geom), -ST_Y(p1.geom))
         END AS geom
  FROM points p2 LEFT OUTER JOIN points p1
  ON ( p1.arc_id = p2.arc_id AND p2.path[1] = p1.path[1]+1 )
  ORDER BY arc_id, p2.path
), arcsdump AS (
  SELECT arc_id, (regexp_matches(ST_AsGeoJSON(geom), '\[.*\]'))[1] as t
  FROM compare
), arcs AS (
  SELECT arc_id, '[' || array_to_string(array_agg(t), ',') || ']' as a FROM arcsdump
  GROUP BY arc_id
  ORDER BY arc_id
)
SELECT '}, "arcs": [' UNION ALL
SELECT array_to_string(array_agg(a), E',\n') from arcs

UNION ALL SELECT ']}'::text as t;
```

Output

```
{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects": {
"P3P4": { "type": "MultiPolygon", "arcs": [[[-1]],[[6,5,-5,-4,-3,1]]]}
}, "arcs": [
 [[25,30],[6,0],[0,10],[-14,0],[0,-10],[8,0]],
 [[35,6],[0,8]],
 [[35,6],[12,0]],
 [[47,6],[0,8]],
 [[47,14],[0,8]],
 [[35,22],[12,0]],
 [[35,14],[0,8]]
 ]}
```

9.15 拓扑空⌧关系

9.15.1 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);

Description

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

The following example uses the `nei_topo` table created in the toTopoGeom examples.

Code

```
SELECT topology.Equals(a.topo, b.topo)
FROM nei_topo AS a
JOIN nei_topo AS b ON b.gid = a.gid
WHERE a.gid = 1;
```

Output

```
t
```

相关信息

GetTopoGeomElements, ST_Equals

9.15.2 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);

Description

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

The following example uses the `nei_topo` table created in the toTopoGeom examples.

Code

```
SELECT topology.Intersects(a.topo, b.topo)
FROM nei_topo AS a
JOIN nei_topo AS b ON b.gid = a.gid
WHERE a.gid = 1;
```

Output

```
t
```

相关信息

ST_Intersects

9.16 ⊠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：

Code

```
pgtopo_export dev_db topo1 | pgtopo_import topo1 | psql staging_db
```

9.16.1 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⊠。

9.16.2 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。

如果目⊠拓扑已存在，并且你希望在⊠建前将其⊠除，可以⊠⊠ `--drop-topology` 参数（自 PostGIS 3.6.0 起支持）。

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：

The loader assigns SRID 4326, creates an index and standard constraints, analyzes the loaded table, records source filenames, and stores the input rasters in 100 by 100 pixel tiles.

Code

```
raster2pgsql -s 4326 -I -C -M -F -t 100x100 *.tif public.demelevation
> elev.sql
psql -d gisdb -f elev.sql
```



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☒：

Code

```
raster2pgsql -s 4326 -I -C -M -F -t 100x100 \
  *.tif 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。

Code

```
raster2pgsql -I -C -e -Y -F -s 26986 -t 128x128 -l 2,4 \
  bostonaerials2008/*.jpg aerials.boston |
  psql -U postgres -d gisdb -h localhost -p 5432
```

List the raster formats supported by this build:

Code

```
raster2pgsql -G
```

The `-G` command outputs a list something like:

Output

```
Available GDAL raster formats:
  Virtual Raster
  GeoTIFF
  National Imagery Transmission Format
  Raster Product Format TOC format
  ECRG TOC format
  Erdas Imagine Images (.img)
  CEOS SAR Image
  CEOS Image
  ...
  Arc/Info Export E00 GRID
  ZMap Plus Grid
  NOAA NGS Geoid Height Grids
```

10.1.1.2 raster2pgsql ☒☒

Long options that take values also accept the `--option=value` spelling.

-?, --help ☒示帮助屏幕。如果您不☒入任何参数，也会☒示帮助。

-G, --gdal-formats 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。

--if-not-exists Use `IF NOT EXISTS` for table creation in `-c` and `-p` modes. When `-I` or `--create-index` is also specified, use `IF NOT EXISTS` for index creation too. This option cannot be used with `-d`. Append mode requires an explicit creation action.

Atomic loader actions The short modes are presets over explicit actions. The equivalent long options are `--drop-table`, `--create-table`, `--load-data`, `--create-index`, `--add-constraints`, `--vacuum`, `--analyze`, and `--no-transaction`. Add `--if-not-exists` to make active creation actions use `IF NOT EXISTS`. `--drop-table` emits `DROP TABLE IF EXISTS` before the selected actions. With no mode specified, the default create/load actions still apply.

□格□理：□用□束以在□格目□中正确注册

-C, --add-constraints □用□格□束——srid、像素大小等，以确保□格在 `raster_columns` □□中正确配准。

-x, --no-extent Disable setting the max extent constraint. Only applied if -C or --add-constraints is also used.

-r, --regular-blocking Set the constraints (spatially unique and coverage tile) for regular blocking. Only applied if -C or --add-constraints is also used.

□格□理：用于操作□入□格数据集的可□参数

-s, --srid [<FROM_SRID>:]<SRID> Assign the output raster to the specified SRID. When FROM_SRID is also provided, raster2pgsql marks the input raster with FROM_SRID and emits SQL that reprojects the raster to the target SRID. If the source SRID is omitted or is zero, the raster metadata will be checked to determine an appropriate source SRID. Reprojection cannot be used with copy mode.

-b, --band BAND 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 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, --pad 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 OVERVIEW_FACTOR Create overview of the raster. For more than one factor, separate with comma(,). Overview table name follows the pattern o_`overview_factor`_`table` where `overview_factor` is a placeholder for numerical overview factor and `table` is replaced with the base table name. If a source raster already has an overview with matching dimensions, that overview is reused. Created overview is stored in the database and is not affected by -R. Note that your generated sql file will contain both the main table and overview tables.

-N, --nodata NODATA NODATA □用于没有 NODATA □的□。

用于操作数据□□象的可□参数

-f, --raster-column COLUMN 指定目□□格列的名称，默□□ “rast”

-F, --filename 添加包含文件名的列

-n, --filename-column COLUMN 指定文件名列的名称。意味着-F。

-q, --quote 将 PostgreSQL □□符括在引号中。

-I Alias for `--create-index`.

--create-index Create a GiST index on the raster column at the end of this raster2pgsql run. With repeated `-a` append runs, create the index on the final run or after loading; add `--if-not-exis` to make reruns tolerate an existing index.

-M 真空分析□格表。

-k, --skip-nodata-check 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 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, --index-tablespace 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, --copy [MAX_ROWS_PER_COPY] 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, --no-transaction ⊠独⊠行每条⊠句，不使用事⊠。

-E, --endian 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, --wkb-version 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：

   Code

   ```
   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：

   Code

   ```
   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⊠。

Code

```
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。

Code

```
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。

Code

```
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⊠⊠。

Code

```
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 Raster Coverages and Raster/Vector Analysis

A coverage is a way to interpret spatial data as a mapping from a *domain* of locations to a *range* of values. Evaluating a coverage means asking for the value at a location; an inverse query asks for the locations whose values match a condition. PostGIS does not expose a generic ISO 19123 coverage SQL type, but geometry tables and raster values provide the building blocks for both directions.

A coverage can be continuous, with values calculated or interpolated for arbitrary locations, or discrete, with values supplied by stored spatial objects or grid cells. The coverage model describes how locations map to values, independently of whether those values come from an equation, a table, or one or more raster values. A time column or function argument can add a temporal dimension without changing that distinction.

In a vector coverage, a geometry column supplies the domain and other columns supply attributes in the range. A vector range can contain text, numbers, or another spatial value, provided its meaning and type are consistent wherever the coverage is defined. In a raster, each cell location is part of the

domain and the band values at that cell form the range. Other columns in the same table row describe the raster value or tile as a whole; they do not attach separate values to individual cells.

A logical raster coverage may be stored as one raster value or as many tiles in a table. Tiles that belong to one coverage should have compatible SRIDs, scales, alignment, band counts, pixel types, and NODATA conventions. The constraints exposed by Section 10.3.1 record and validate those properties.

Raster table arrangements. A raster column can represent an image warehouse, a tiled coverage, or a collection of rasterized objects. These arrangements have different assumptions about tile size, alignment, gaps, and overlaps. PostGIS does not infer the intended arrangement from the column type alone.

Arrangement	Row relationship	Typical use
Image warehouse	Independent rasters; sizes, alignment, and extents may differ or overlap	Cataloging unrelated source images
Irregular tiled coverage	One logical coverage with gaps allowed, variable tile sizes, and normally no overlap	Incremental or source-driven mosaics
Regular tiled coverage	One alignment and tile size with no overlap; gaps may remain at the boundary or inside the coverage	Indexed analysis over a large coverage
Rectangular regular coverage	Equal aligned tiles fill a rectangular extent without gaps or overlaps	Predictable block processing and overview generation
Separate tiled images	Each image is internally regular but stored in its own table or partition	Keeping source-image identity and lifecycle separate
Raster-object coverage	Each row is a rasterized feature; extents and sizes may differ and overlap	Object models where cells replace or supplement vector geometry

Use AddRasterConstraints to record common SRID, scale, alignment, band, extent, and regular-blocking properties, and use spatial indexes and application constraints for relationships that the catalog cannot express.

Queries against a tiled coverage normally have two stages:

1. Use the raster spatial index and ST_Intersects to select candidate tiles.
2. Read, clip, aggregate, or vectorize values inside those tiles with functions such as ST_Value, ST_Clip, ST_SummaryStatsAgg, and ST_DumpAsPolygons.

ST_DumpAsPolygons returns geomval rows that pair geometries with cell values. This allows ordinary geometry overlay and filtering to answer inverse coverage queries, but polygonizing many cells is usually more expensive than keeping the operation in raster space.

Choose boundary semantics deliberately. Clipping and aggregating raster pixels is usually the shorter and faster path. If a partial boundary cell must contribute in proportion to its covered area, intersect the geometry with the geomval rows and calculate an area-weighted result in an appropriate projected coordinate system. That exact partial-cell method costs more because it vectorizes the selected cells.

Tile size affects both sides of the indexed query: smaller tiles reduce the amount of raster data inspected for each candidate row, while increasing row and index overhead. Benchmark representative queries instead of treating one tile size as a universal default.

For a current end-to-end raster/vector pattern, load and tile data with Section 10.1.1, then see the building example in ST_SummaryStats. It selects intersecting tiles, clips them to each polygon, and

combines per-tile statistics into one result per feature. For point observations such as animal locations, first buffer the points in an appropriate projected coordinate system, then apply the same tile selection, clipping, and aggregation pattern to an elevation or other continuous raster coverage.

10.3 ☒格目☒

PostGIS 附☒了☒个☒格目☒☒☒。☒个☒☒都利用嵌入在☒格表☒束中的信息。因此，由于☒制☒施了☒束，目☒☒☒始☒与表中的☒格数据一致。

1. `raster_columns` 此☒☒☒数据☒中的所有☒格表列☒行☒目。
2. `raster_overviews` 此☒☒☒数据☒中的所有☒格表列☒行☒目，☒些列用作更☒粒度表的概述。当您在加☒期☒使用 `-l` 开关☒，会生成☒种☒型的表。

10.3.1 ☒格列目☒

`raster_columns` 是数据☒中所有☒格☒型的☒格表列的目☒。它是一种利用表☒束的☒☒，因此即使您从☒一个数据☒的☒份恢复一个☒格表，信息也始☒保持一致。`raster_columns` 目☒中存在以下列。

如果您未使用加☒器☒建表或忘☒在加☒期☒指定 `-C` ☒志，☒可以在事后使用 AddRasterConstraints☒制☒行☒束，以便 `raster_columns` 目☒注册有关☒格切片的通用信息。

- `r_table_catalog` 表所在的数据☒。☒将始☒☒取当前数据☒。
- `r_table_schema` ☒格表所属的数据☒模式。
- `r_table_name` ☒格表
- `r_raster_column` 在 `r_table_name` 表中☒格☒型的列。PostGIS 中没有任何内容可以阻止您在每个表中☒有多个☒格列，因此可以多次列出一个☒格表，并且每个表具有不同的☒格列。
- `srid` ☒格的空☒参考☒☒符。☒☒是 Section 4.5中的条目。
- `scale_x` 几何空☒坐☒与像素之☒的☒放比例。☒当☒格列中的所有切片具有相同的 `scale_x` 并且☒用此☒束☒，此功能才可用。有关更多☒☒信息，☒参☒ ST_ScaleX。
- `scale_y` 几何空☒坐☒与像素之☒的☒放比例。☒当☒格列中的所有切片具有相同的 `scale_y` 并且☒用了 `scale_y` ☒束☒，此功能才可用。有关☒☒信息，☒参☒ ST_ScaleY。
- `blocksize_x` 每个☒格☒☒的☒度（跨像素数）。有关更多☒☒信息，☒参☒ST_Width。
- `blocksize_y` 每个☒格☒☒的☒度（向下的像素数）。有关更多☒☒信息，☒参☒ST_Height。
- `same_alignment` 一个布☒☒，如果所有☒格☒☒具有相同的☒☒方式，☒☒☒☒ true。有关更多☒☒信息，☒参☒ST_SameAlignment。
- `Regular_blocking` 如果☒格列具有空☒唯一性和覆盖范☒切片☒束，☒☒☒☒ TRUE。否☒，它将是 FALSE。
- `num_bands` ☒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☒每个波段的像素☒型的数☒。☒数☒中的元素数量与波段的数量相同。Pixel_types 是 ST_BandPixelType 中定☒的以下☒型之一。
- `nodata_values` 双精度数字数☒，表示每个波段的 `nodata_value`。☒数☒中的元素数量与☒的数量相同。☒些数字定☒了大多数操作中☒忽略的每个波段的像素☒。☒与ST_BandNoDataValue 提供的信息☒似。
- `out_db` 布☒☒志数☒，指示☒格波段数据是否在数据☒外部☒☒。☒数☒中的元素数量与☒的数量相同。
- `extent` ☒是☒格集中所有☒格行的范☒。如果您☒划加☒更多数据来更改集的范☒，☒需要在加☒之前☒行 DropRasterConstraints 函数，然后在加☒后使用AddRasterConstraints 重新☒用☒束。
- `Spatial_index` 如果☒格列具有空☒索引，☒☒ true 的布☒☒。

10.3.2 ☒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）。

10.4 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。

10.4.1 PHP 示例使用 ST_AsPNG 与其他栅格函数行出

在本中，我将演示如何使用 PHP PostgreSQL 程序和 ST_AsGDALRaster系列函数将格的波段 1、2、3 出到 PHP 求流，然后将其嵌入到 img src html 中。

示例演示了如何将一大堆栅格函数合在一起，以取与特定 wgs 84 界框相交的所有，然后将相交与 ST_Union合起来，返回所有波段，使用 ST_Transform用指定的投影，然后出使用 ST_AsPNG将果示 png。

You would call the script with `http://mywebserver/test_raster.php?srid=2249` to get the raster image in Massachusetts state plane feet.

Code

```
<?php
$conn_str = 'dbname=mydb host=localhost port=5432 user=myuser password=mypwd';
$dbconn = pg_connect($conn_str);
header('Content-Type: image/png');

if (!empty($_REQUEST['srid']) &amp;amp;&amp;amp; is_numeric($_REQUEST['srid'])) {
    $input_srid = intval($_REQUEST['srid']);
} else {
    $input_srid = 26986;
}

$sql = "SET bytea_output = 'escape';
SELECT ST_AsPNG(
    ST_Transform(
        ST_AddBand(
            ST_Union(rast, 1),
            ARRAY[ST_Union(rast, 2), ST_Union(rast, 3)]
        ),
        $input_srid
    )
) AS new_rast
FROM aerials.boston
WHERE ST_Intersects(
    rast,
    ST_Transform(
        ST_MakeEnvelope(-71.1217, 42.227, -71.1210, 42.218, 4326),
        26986
    )
)";

$result = pg_query($sql);
$row = pg_fetch_row($result);
pg_free_result($result);
if ($row === false) return;
echo pg_unescape_bytea($row[0]);
?>
```

10.4.2 ASP.NET C# 示例使用 ST_AsPNG 与其他栅格函数行出

In this section, we'll demonstrate how to use Npgsql PostgreSQL .NET driver and the ST_AsGDALRaster family of functions to output band 1,2,3 of a raster to the HTTP response stream so it can be embedded in an `img` tag.

本需要 npgsql .NET PostgreSQL 程序，您可以从 http://npgsql.projects.postgresql.org/ 取最新的程序。只需下最新版本并放入 ASP.NET bin 文件中即可开始使用。

示例⊠⊠演示了如何将一大堆⊠格函数⊠合在一起，以⊠取与特定 wgs 84 ⊠界框相交的所有⊠⊠，然后将相交⊠⊠与 ST_Union⊠合起来，返回所有波段，使用 ST_Transform⊠⊠⊠用⊠指定的投影，然后⊠出使用 ST_AsPNG将⊠果⊠示⊠ png。

⊠示例与 Section 10.4.1示例相同，只不⊠是在 C# 中⊠⊠的。

You would call the handler with `http://mywebserver/TestRaster.ashx?srid=2249` to get the raster image in Massachusetts state plane feet.

web.config connection string section:

Code

```
<connectionStrings>
    <add name="DSN"
        connectionString="server=localhost;database=mydb;Port=5432;User Id=myuser;password= ↩
    mypwd"/>
</connectionStrings>
```

Code for `TestRaster.ashx`:

Code

```
<%@ WebHandler Language="C#" Class="TestRaster" %>
using System;
using System.Data;
using System.Web;
using Npgsql;

public class TestRaster : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "image/png";
        context.Response.BinaryWrite(GetResults(context));
    }

    public bool IsReusable {
        get { return false; }
    }

    public byte[] GetResults(HttpContext context)
    {
        byte[] result = null;
        NpgsqlCommand command;
        string sql = null;
        int input_srid = 26986;
        try {
            string connectionString = System.Configuration.ConfigurationManager
                .ConnectionStrings["DSN"].ConnectionString;
            using (NpgsqlConnection conn = new NpgsqlConnection(connectionString)) {
                conn.Open();

                if (context.Request["srid"] != null)
                {
                    input_srid = Convert.ToInt32(context.Request["srid"]);
                }
                sql = @"SELECT ST_AsPNG(
                            ST_Transform(
                                ST_AddBand(
                                    ST_Union(rast, 1),
                                    ARRAY[ST_Union(rast, 2), ST_Union(rast, 3)]
                                ),
                                :input_srid
```

```
                        )
                    ) As new_rast
                    FROM aerials.boston
                    WHERE ST_Intersects(
                        rast,
                        ST_Transform(
                            ST_MakeEnvelope(
                                -71.1217, 42.227,
                                -71.1210, 42.218,
                                4326
                            ),
                            26986
                        )
                    )";
                command = new NpgsqlCommand(sql, conn);
                command.Parameters.Add(new NpgsqlParameter("input_srid", input_srid));


                result = (byte[]) command.ExecuteScalar();
                conn.Close();
            }

        }
        catch (Exception ex)
        {
            result = null;
            context.Response.Write(ex.Message.Trim());
        }
        return result;
    }
}
```

10.4.3 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

You can compile the following code with commands like:

Code

```
CLASSPATH=".:../postgresql.jar" javac SaveQueryImage.java
jar cfm SaveQueryImage.jar Manifest.txt *.class
```

And call it from the command line with something like:

Code

```
java -jar SaveQueryImage.jar \
  "SELECT ST_AsPNG(ST_AsRaster(ST_Buffer(ST_Point(1, 5), 10, 'quad_segs=2'), 150, 150, '8 ↩
    BUI', 100));" \
  "test.png"
```

`Manifest.txt`:

Code

```
Class-Path: postgresql-9.0-801.jdbc4.jar
Main-Class: SaveQueryImage
```

Code for `SaveQueryImage.java`:

Code

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.io.*;

public class SaveQueryImage {
  public static void main(String[] argv) {
      System.out.println("Checking if Driver is registered with DriverManager.");

      try {
        //java.sql.DriverManager.registerDriver (new org.postgresql.Driver());
        Class.forName("org.postgresql.Driver");
      }
      catch (ClassNotFoundException cnfe) {
        System.out.println("Couldn't find the driver!");
        cnfe.printStackTrace();
        System.exit(1);
      }

      Connection conn = null;

      try {
        conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/mydb", "myuser ↩
   ", "mypwd");
        conn.setAutoCommit(false);

        PreparedStatement sGetImg = conn.prepareStatement(argv[0]);

        ResultSet rs = sGetImg.executeQuery();

        FileOutputStream fout;
        try
        {
            rs.next();
            /** Output to file name requested by user **/
            fout = new FileOutputStream(new File(argv[1]) );
            fout.write(rs.getBytes(1));
            fout.close();
        }
        catch(Exception e)
        {
            System.out.println("Can't create file");
            e.printStackTrace();
        }

        rs.close();
        sGetImg.close();
        conn.close();
      }
      catch (SQLException se) {
        System.out.println("Couldn't connect: print out a stack trace and exit.");
        se.printStackTrace();
        System.exit(1);
```

```
      }
  }
}
```

10.4.4 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。

Code

```
CREATE OR REPLACE FUNCTION write_file (param_bytes bytea, param_filepath text)
RETURNS text
AS $$
f = open(param_filepath, 'wb+')
f.write(param_bytes)
return param_filepath
$$ LANGUAGE plpythonu;
```

Write out 5 images to the PostgreSQL server in varying sizes. The PostgreSQL daemon account needs write access to the folder; this echoes back the file names created.

Code

```
SELECT write_file(
    ST_AsPNG(
        ST_AsRaster(
            ST_Buffer(ST_Point(1, 5), j * 5, 'quad_segs=2'),
            150 * j,
            150 * j,
            '8BUI',
            100
        )
    ),
    'C:/temp/slices' || j || '.png'
)
FROM generate_series(1, 5) AS j;
```

Output

```
write_file
--------------------
 C:/temp/slices1.png
 C:/temp/slices2.png
 C:/temp/slices3.png
 C:/temp/slices4.png
 C:/temp/slices5.png
```

10.4.5 使用 PSQL ⌧出⌧格

Sadly PSQL doesn't have easy to use built-in functionality for outputting binaries. This is a bit of a hack that piggy backs on PostgreSQL somewhat legacy large object support. To use it, first launch your **psql** command line connected to your database.

与 python 方法不同，此方法在本地⌧算机上⌧建文件。

Code

```
SELECT
    oid,
    lowrite(lo_open(oid, 131072), png) AS num_bytes
FROM (VALUES (
    lo_create(0),
    ST_AsPNG((SELECT rast FROM aerials.boston WHERE rid = 1))
)) AS v(oid, png);
```

Output

```
oid   | num_bytes
---------+-----------
 2630819 |     74860
```

Note the large object OID and export it to a file on your local computer, replacing the example path with the desired file path.

Code

```
\lo_export 2630819 'C:/temp/aerial_samp.png'
```

This deletes the file from large object storage in the database.

Code

```
SELECT lo_unlink(2630819);
```

Chapter 11

☒格参考

raster 是用于存☒和分析☒格数据的 PostGIS ☒型。
有关从光☒文件加☒光☒的信息，☒参☒ Section 10.1
本参考文档中的部分示例使用了一个通☒以下代☒☒建的虚☒☒格数据表：

Code

```
CREATE TABLE dummy_rast(rid integer, rast raster);
INSERT INTO dummy_rast(rid, rast)
VALUES (1,
('01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0000' -- nBands (uint16 0)
||
'0000000000000040' -- scaleX (float64 2)
||
'0000000000000840' -- scaleY (float64 3)
||
'000000000000E03F' -- ipX (float64 0.5)
||
'000000000000E03F' -- ipY (float64 0.5)
||
'0000000000000000' -- skewX (float64 0)
||
'0000000000000000' -- skewY (float64 0)
||
'00000000' -- SRID (int32 0)
||
'0A00' -- width (uint16 10)
||
'1400' -- height (uint16 20)
)::raster
),
-- Raster: 5 x 5 pixels, 3 bands, PT_8BUI pixel type, NODATA = 0
(2, ('01000003009A9999999999A93F9A9999999999A9BF000000E02B274A' ||
'4100000000771956410000000000000000000000000000000000 ↩
    FFFFFFFF050005000400FDFEFDFEFEFDFEFEFDF9FAFEF' ||
' ↩
    EFCF9FBFDFEFEFDFCFAFEFEFE04004E627AADD16076B4F9FE6370A9F5FE59637AB0E54F5861708704004656648 7A1506
    ')::raster);
```

下列函数是 PostGIS Raster 用☒常用的函数。除此之外，☒存在其他☒格支持函数，但它☒☒一般用☒而言并不常用。

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☒）。

Description

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☒。

Description

用作 ST_AddBand 函数的☒入的复合☒型，定☒新波段的属性和初始☒。

Field	Type	Description
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	Pixel type of the new band. One of the pixel types described in ST_BandPixelType.
initialvalue	double precision	Initial value that all pixels of the new band will be set to.
nodataval	double precision	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。

Description

需要表☒☒格和☒☒格的波段索引☒使用的复合☒型。

Field	Type	Description
rast	raster	The raster in question.
nband	integer	1-based value indicating the raster band.

相关信息

ST_MapAlgebra (callback function version)

11.1.4 raster

raster — ☒格空☒数据☒型。

Description

☒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 — 用于☒入 ST_Reclass 函数的复合☒型，用于定☒重分☒操作的行☒。

Description

用于☒入 ST_Reclass 函数的复合☒型，用于定☒重分☒操作的行☒。

Field	Type	Description
nband	integer	要重新分⌧的波段的波段号。
reclassexpr	text	Comma-delimited range:map_range mappings that define how old band values map to new band values.
pixeltype	text	One of defined pixel types as described in ST_BandPixelType.
nodataval	double precision	⌧⌧无数据的⌧。⌧于支持透明度的⌧像⌧出，⌧些将⌧空白。

The reclassexpr range delimiters mean: (greater than,) less than,] less than or equal, and [greater than or equal.

- [a-b] means a <= x <= b.
- (a-b] means a < x <= b.
- [a-b) means a <= x < b.
- (a-b) means a < x < b.

Opening-parenthesis notation is optional, so a-b means the same as (a-b).

示例

Reclassify band 2 as an 8BUI where 255 is nodata value.

Code

```
SELECT ROW(2, '0-100:1-10, 101-500:11-150,501 - 10000: 151-254', '8BUI', 255)::reclassarg;
```

Reclassify band 1 as an 1BB and no nodata value defined.

Code

```
SELECT ROW(1, '0-100]:0, (100-255:1', '1BB', NULL)::reclassarg;
```

相关信息

ST_Reclass

11.1.6 summarystats

summarystats — ST_SummaryStats 和 ST_SummaryStatsAgg 函数返回的复合⌧型。

Description

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⍰。

Description

用作 ST_Union 函数的⍰入的复合⍰型，定⍰要⍰理的波段和 UNION 操作的行⍰。

Field	Type	Description
nband	integer	从 1 开始的⍰，指示要⍰理的每个⍰入⍰格的波段。
uniontype	text	Type of union operation. One of the types described in ST_Union.

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);

Description

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 10.3.1。

- `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` ☒置空☒唯一（☒个☒格不能在空☒上相同）和覆盖☒☒（☒格与覆盖范☒☒☒）☒束
- `Same_alignment` 确保它☒都具有相同的☒☒方式，☒意味着您比☒的任何☒个☒☒都将返回 true。☒参☒ ST_SameAlignment。
- `srid` 确保所有人都有相同的 srid
- More--任何列☒上述函数☒入的内容



Note
☒函数根据表中已有的数据推断☒束。因此，要使其正常工作，您必☒首先☒建☒格列，然后向其加☒数据。



Note
如果在☒用☒束后需要在表中加☒更多数据，并且数据范☒已更改，☒可能需要☒行 DropRasterCon-
straints。

可用性: 2.0.0

示例

Apply all possible constraints on column based on data.
Verify the raster table is registered correctly in the raster_columns view.

Code

```
CREATE TABLE myrasters(rid SERIAL primary key, rast raster);
INSERT INTO myrasters(rast)
SELECT ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0.3, -0.3, 2, 2, 0, 0, 4326), 1, '8BSI':: ↩
    text, -129, NULL);

SELECT AddRasterConstraints('myrasters'::name, 'rast'::name);
SELECT srid, scale_x, scale_y, blocksize_x, blocksize_y, num_bands, pixel_types,  ↩
    nodata_values
    FROM raster_columns
    WHERE r_table_name = 'myrasters';
```

Output

```
srid | scale_x | scale_y | blocksize_x | blocksize_y | num_bands | pixel_types| nodata_values
------+---------+---------+-------------+-------------+-----------+-------------+---------------
 4326 |       2 |       2 |        1000 |        1000 |         1 | {8BSI}      | {0}
```

Apply single constraint.

Code

```
CREATE TABLE public.myrasters2(rid SERIAL primary key, rast raster);
INSERT INTO myrasters2(rast)
SELECT ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0.3, -0.3, 2, 2, 0, 0, 4326), 1, '8BSI':: ↩
    text, -129, NULL);

SELECT AddRasterConstraints('public'::name, 'myrasters2'::name, 'rast'::name, ' ↩
    regular_blocking', 'blocksize');
```

Output

```
NOTICE:  Adding regular blocking constraint
NOTICE:  Adding blocksize-X constraint
NOTICE:  Adding blocksize-Y constraint
```

相关信息

Section 10.3.1, ST_AddBand, ST_MakeEmptyRaster, DropRasterConstraints, ST_BandPixelType, ST_SRID

11.2.2 DropRasterConstraints

DropRasterConstraints — 删除引用栅格表列的 PostGIS 栅格约束。如果您需要重新加载数据或更新栅格列数据，这非常有用。

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);

Description

☒除引用由AddRasterConstraints添加的☒格表列的 PostGIS ☒格☒束。如果您需要加☒更多数据或更新☒格列数据，☒非常有用。如果您想要☒除☒格表或☒格列，☒无需☒行此操作。

To drop a raster table use standard SQL: `DROP TABLE mytable`.

To drop just a raster column and leave the rest of the table, use standard SQL: `ALTER TABLE mytable DROP COLUMN rast`.

如果☒除列或表,☒表将从 `raster_columns` 目☒中消失。但是,如果☒☒除☒束,☒☒格列仍将列在 `raster_columns` 目☒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

Code

```
SELECT DropRasterConstraints ('myrasters', 'rast');
```

Output

```
droprasterconstraints
----------------------
 t
(1 row)
```

Verify change in raster_columns.

Code

```
SELECT srid, scale_x, scale_y, blocksize_x, blocksize_y, num_bands, pixel_types,  ↩
    nodata_values
    FROM raster_columns
    WHERE r_table_name = 'myrasters';
```

Output

```
srid | scale_x | scale_y | blocksize_x | blocksize_y | num_bands | pixel_types| nodata_values
------+---------+---------+-------------+-------------+-----------+-------------+---------------
    0 |         |         |             |             |           |             |
```

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);

Description

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

Verify the raster table is registered correctly in the raster_overviews view.

Code

```
CREATE TABLE res1 AS SELECT
ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 2),
  1, '8BSI'::text, -129, NULL
) r1;

CREATE TABLE res2 AS SELECT
ST_AddBand(ST_MakeEmptyRaster(500, 500, 0, 0, 4),
  1, '8BSI'::text, -129, NULL
) r2;

SELECT AddOverviewConstraints('res2', 'r2', 'res1', 'r1', 2);
SELECT o_table_name ot, o_raster_column oc,
       r_table_name rt, r_raster_column rc,
       overview_factor f
FROM raster_overviews WHERE o_table_name = 'res2';
```

Output

```
ot  | oc |  rt  | rc | f
------+----+------+----+---
 res2 | r2 | res1 | r1 | 2
(1 row)
```

相关信息

Section 10.3.2, 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);

Description

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 10.3.2, AddOverviewConstraints, DropRasterConstraints

11.2.5 PostGIS_GDAL_Version

PostGIS_GDAL_Version — 报告 PostGIS 使用的 GDAL 库的版本。

Synopsis

text **PostGIS_GDAL_Version**();

Description

报告 PostGIS 使用的 GDAL 库的版本。将报告并报告 GDAL 是否可以找到其数据文件。

示例

Code

```
SELECT PostGIS_GDAL_Version();
```

Output

```
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 — 报告完整的栅格库构建日期。

Synopsis

text **PostGIS_Raster_Lib_Build_Date**();

Description

报告栅格库构建日期

示例

Code

```
SELECT PostGIS_Raster_Lib_Build_Date();
```

Output

```
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 — 报告完整的栅格版本和栅建配置信息。

Synopsis

text **PostGIS_Raster_Lib_Version**();

Description

报告完整的栅格版本和栅建配置信息。

示例

Code

```
SELECT PostGIS_Raster_Lib_Version();
```

Output

```
2.0.0
```

相关信息

PostGIS_Lib_Version

11.2.8 ST_GDALDrivers

ST_GDALDrivers — 通过 GDAL 返回 PostGIS 支持的栅格格式列表。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);

Description

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

List of Drivers.

Code

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SELECT short_name, long_name, can_write
FROM ST_GDALDrivers()
ORDER BY short_name;
```

Output

```
short_name    |                        long_name                         | can_write
----------------+----------------------------------------------------------+-----------
 AAIGrid         | Arc/Info ASCII Grid                                      | t
 ACE2            | ACE2                                                     | f
 ADRG            | ARC Digitized Raster Graphics                            | f
 AIG             | Arc/Info Binary Grid                                     | f
 AirSAR          | AirSAR Polarimetric Image                                | f
 ARG             | Azavea Raster Grid format                                | t
 BAG             | Bathymetry Attributed Grid                               | f
 BIGGIF          | Graphics Interchange Format (.gif)                       | f
 BLX             | Magellan topo (.blx)                                     | t
 BMP             | MS Windows Device Independent Bitmap                     | f
 BSB             | Maptech BSB Nautical Charts                              | f
 PAux            | PCI .aux Labelled                                        | f
 PCIDSK          | PCIDSK Database File                                     | f
 PCRaster        | PCRaster Raster File                                     | f
 PDF             | Geospatial PDF                                           | f
 PDS             | NASA Planetary Data System                               | f
 PDS4            | NASA Planetary Data System 4                             | t
 PLMOSAIC        | Planet Labs Mosaics API                                  | f
 PLSCENES        | Planet Labs Scenes API                                   | f
 PNG             | Portable Network Graphics                                | t
 PNM             | Portable Pixmap Format (netpbm)                          | f
 PRF             | Racurs PHOTOMOD PRF                                      | f
 R               | R Object Data Store                                      | t
 Rasterlite      | Rasterlite                                               | t
 RDA             | DigitalGlobe Raster Data Access driver                   | f
 RIK             | Swedish Grid RIK (.rik)                                  | f
 RMF             | Raster Matrix Format                                     | f
 ROI_PAC         | ROI_PAC raster                                           | f
 RPFTOC          | Raster Product Format TOC format                         | f
 RRASTER         | R Raster                                                 | f
 RS2             | RadarSat 2 XML Product                                   | f
 RST             | Idrisi Raster A.1                                        | t
 SAFE            | Sentinel-1 SAR SAFE Product                              | f
 SAGA            | SAGA GIS Binary Grid (.sdat, .sg-grd-z)                  | t
 SAR_CEOS        | CEOS SAR Image                                           | f
 SDTS            | SDTS Raster                                              | f
 SENTINEL2       | Sentinel 2                                               | f
 SGI             | SGI Image File Format 1.0                                | f
 SNODAS          | Snow Data Assimilation System                            | f
 SRP             | Standard Raster Product (ASRP/USRP)                      | f
 SRTMHGT         | SRTMHGT File Format                                      | t
 Terragen        | Terragen heightfield                                     | f
 TIL             | EarthWatch .TIL                                          | f
 TSX             | TerraSAR-X Product                                       | f
 USGSDEM         | USGS Optional ASCII DEM (and CDED)                       | t
 VICAR           | MIPL VICAR file                                          | f
```

```
 VRT            | Virtual Raster                                          | t
 WCS            | OGC Web Coverage Service                                | f
 WMS            | OGC Web Map Service                                     | t
 WMTS           | OGC Web Map Tile Service                                | t
 XPM            | X11 PixMap Format                                       | t
 XYZ            | ASCII Gridded XYZ                                       | t
 ZMap           | ZMap Plus Grid                                          | t
```

List of options for each driver.

Output the create options XML column of JPEG as a table. These creator options can be used in the ST_AsGDALRaster options argument.

Code

```
SELECT (xpath('@name', g.opt))[1]::text As oname,
       (xpath('@type', g.opt))[1]::text As otype,
       (xpath('@description', g.opt))[1]::text As descrip
FROM (SELECT unnest(xpath('/CreationOptionList/Option', create_options::xml)) As opt
FROM  ST_GDALDrivers()
WHERE short_name = 'JPEG') As g;
```

Output

```
oname        |  otype  |      descrip
-------------------+---------+-----------------------------------------------
 PROGRESSIVE       | boolean | whether to generate a progressive JPEG
 QUALITY           | int     | good=100, bad=0, default=75
 WORLDFILE         | boolean | whether to generate a worldfile
 INTERNAL_MASK     | boolean | whether to generate a validity mask
 COMMENT           | string  | Comment
 SOURCE_ICC_PROFILE | string  | ICC profile encoded in Base64
 EXIF_THUMBNAIL    | boolean | whether to generate an EXIF thumbnail(overview).
                             By default its max dimension will be 128
 THUMBNAIL_WIDTH   | int     | Forced thumbnail width
 THUMBNAIL_HEIGHT  | int     | Forced thumbnail height
(9 rows)
```

This example turns a few common GeoTIFF creator options into a compact table.

Code

```
WITH gtiff_options AS (
    SELECT
        option_name,
        option_type,
        default_value
    FROM ST_GDALDrivers() AS driver
    CROSS JOIN LATERAL XMLTABLE(
        '/CreationOptionList/Option'
        PASSING (driver.create_options::xml)
        COLUMNS
            option_name text PATH '@name',
            option_type text PATH '@type',
            default_value text PATH '@default'
    ) AS options
    WHERE driver.short_name = 'GTiff'
)
SELECT option_name AS name, option_type AS type, coalesce(default_value, '') AS ↩
    default_value
FROM gtiff_options
WHERE option_name = ANY (ARRAY[
    'COMPRESS',
    'INTERLEAVE',
    'TILED',
    'BLOCKXSIZE',
    'BLOCKYSIZE',
```

```
    'JPEG_QUALITY'
])
ORDER BY 1;
```

Output

```
name      |     type      | default_value
--------------+---------------+--------------
 BLOCKXSIZE   | int           |
 BLOCKYSIZE   | int           |
 COMPRESS     | string-select |
 INTERLEAVE   | string-select | PIXEL
 JPEG_QUALITY | int           | 75
 TILED        | boolean       |
(6 rows)
```

Use XMLTABLE to expose selected GTiff creation options as rows. The same pattern can be used without the name filter to inspect the full XML document.

Code

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';

SELECT option_name, option_type
FROM ST_GDALDrivers() AS driver
CROSS JOIN LATERAL XMLTABLE(
    '/CreationOptionList/Option'
    PASSING (driver.create_options::xml)
    COLUMNS
        option_name text PATH '@name',
        option_type text PATH '@type'
) AS options
WHERE driver.short_name = 'GTiff'
  AND option_name IN ('COMPRESS', 'TILED', 'BIGTIFF')
ORDER BY option_name;
```

Output

```
option_name |  option_type
------------+--------------
 BIGTIFF    | string-select
 COMPRESS   | string-select
 TILED      | boolean
(3 rows)
```

相关信息

ST_AsGDALRaster, ST_SRID, postgis.gdal_enabled_drivers

11.2.9 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);

Description

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

11.2.10 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');

Description

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

Code

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2, 'Lanczos');
```

⍰出到更快地⍰理默⍰最近⍰

Code

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2);
```

相关信息

ST_Retile, AddOverviewConstraints, AddRasterConstraints, Section 10.3.2

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);

Description

返回一个☒格，其中在☒定位置（索引）、☒定☒型、☒定初始☒和☒定 nodata ☒中添加了新波段。如果未指定索引，☒将☒添加到末尾。如果未指定 `fromband`，☒假定☒ band 1。像素☒型是 ST_BandPixelType 中指定的像素☒型之一的字符串表示形式。如果指定了☒有索引，☒所有 >= ☒索引的后☒波段都会增加 1。如果指定的初始☒大于像素☒型的最大☒，☒初始☒将☒置☒像素☒型允☒的最高☒。

☒于采用 addbandarg数☒的格式（格式 1），特定 addbandarg 的索引☒与☒ addbandarg 描述的波段添加到☒格☒的☒格相关。☒参☒下面的多个新波段示例。

☒于采用☒格数☒的格式（格式 5），如果 `torast` ☒ NULL，☒数☒中每个☒格的 `fromband` 波段将累☒到新☒格中。

☒于采用 `outdbfile` 的格式（格式 6 和 7），☒☒必☒包含☒格文件的完整路径。☒文件☒必☒可供 postgres 服☒器☒程☒☒。

增☒：2.1.0 添加了☒ addbandarg 的支持。

增☒：2.1.0 添加了☒新的 out-db 波段的支持。

示例

Single New Band.

This example adds another 8-bit unsigned integer band with pixels initialized to 200.

Code

```
UPDATE dummy_rast
    SET rast = ST_AddBand(rast, '8BUI'::text, 200)
WHERE rid = 1;
```

Create an empty raster 100x100 units, with upper left right at 0, add 2 bands (band 1 is 0/1 boolean bit switch, band2 allows values 0-15) Uses addbandargs.

Output metadata for the raster bands to verify they are correct.

Code

```
INSERT INTO dummy_rast(rid, rast)
    VALUES(10, ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 1, -1, 0, 0, 0),
    ARRAY[
        ROW(1, '1BB'::text, 0, NULL),
        ROW(2, '4BUI'::text, 0, NULL)
            ]::addbandarg[]
     )
    );

SELECT  (bmd).*
FROM (SELECT ST_BandMetaData(rast, generate_series(1, 2)) As bmd
    FROM dummy_rast WHERE rid = 10) AS foo;
```

Output

```
pixeltype | nodatavalue | isoutdb | path
-----------+---------------+-------------+---------+------
 1BB       |             | f       |
 4BUI      |             | f       |
```

This example outputs metadata for the raster.

Code

```
SELECT  (rmd).width, (rmd).height, (rmd).numbands
FROM (SELECT ST_MetaData(rast) As rmd
    FROM dummy_rast WHERE rid = 10) AS foo;
```

Output

```
width | height | numbands
-------+--------+----------
   100 |    100 |        2
```

Multiple New Bands.

Code

```
SELECT
    *
FROM ST_BandMetaData(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1, 0, 0, 0),
        ARRAY[
            ROW(NULL, '8BUI', 255, 0),
            ROW(NULL, '16BUI', 1, 2),
            ROW(2, '32BUI', 100, 12),
            ROW(2, '32BF', 3.14, -1)
        ]::addbandarg[]
    ),
    ARRAY[]::integer[]
);
```

Output

```
bandnum | pixeltype | nodatavalue | isoutdb | path | outdbbandnum | filesize | filetimestamp
---------+-----------+-------------+---------+------+--------------+----------+---------------
       1 | 8BUI      |           0 | f       | null |         null |     null |          null
       2 | 32BF      |          -1 | f       | null |         null |     null |          null
       3 | 32BUI     |          12 | f       | null |         null |     null |          null
       4 | 16BUI     |           2 | f       | null |         null |     null |          null
```

Aggregate the 1st band of a table of like rasters into a single raster. With as many bands as there are test_types and as many rows (new rasters) as there are mice. Note that The ORDER BY test_type clause relies on aggregate ORDER BY support. The resulting raster will have a band for each test_type alphabetical by test_type. For mouse lovers: No mice were harmed in this exercise.

Code

```
SELECT
    mouse,
    ST_AddBand(NULL, array_agg(rast ORDER BY test_type), 1) As rast
FROM mice_studies
GROUP BY mouse;
```

New Out-db band.

Code

```
SELECT
    *
FROM ST_BandMetaData(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1, 0, 0, 0),
        '/home/raster/mytestraster.tif'::text, NULL::int[]
    ),
    ARRAY[]::integer[]
);
```

Output

```
bandnum | pixeltype | nodatavalue | isoutdb | path
---------+-----------+-------------+---------+------
       1 | 8BUI      |             | t       | /home/raster/mytestraster.tif
       2 | 8BUI      |             | t       | /home/raster/mytestraster.tif
       3 | 8BUI      |             | t       | /home/raster/mytestraster.tif
```

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);

Description

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。

Curved geometry inputs are linearized before rasterization, including NURBSCurve inputs handled by PostGIS. TIN and PolyhedralSurface inputs are not supported.

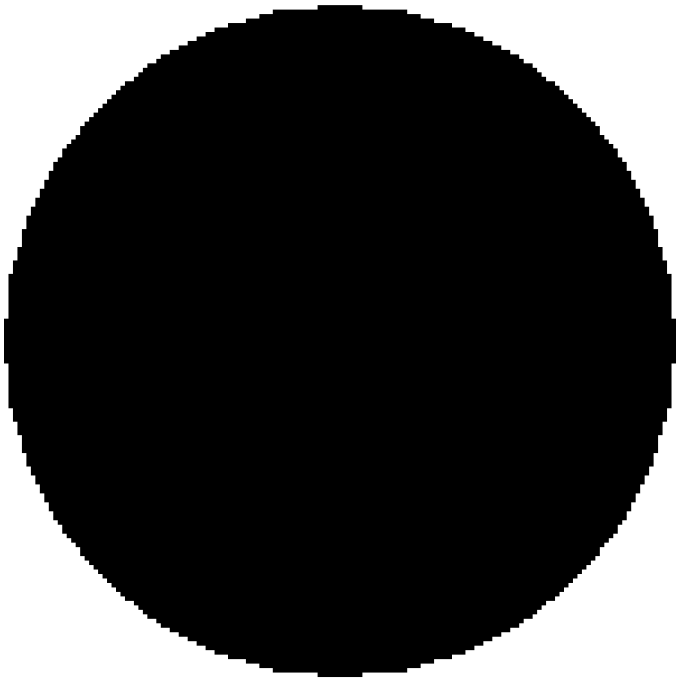
示例：将几何⊠形⊠出⊠ **PNG** 文件

This example outputs a black circle on a white background, occupying 150 by 150 pixels.

Code

```
SELECT ST_AsPNG(
    ST_AsRaster(
        ST_Buffer(ST_Point(1, 5), 10),
        150, 150,
        ARRAY['8BUI', '8BUI', '8BUI'],
        ARRAY[0, 0, 0],
        ARRAY[255, 255, 255]
    )
) AS image;
```

Figure

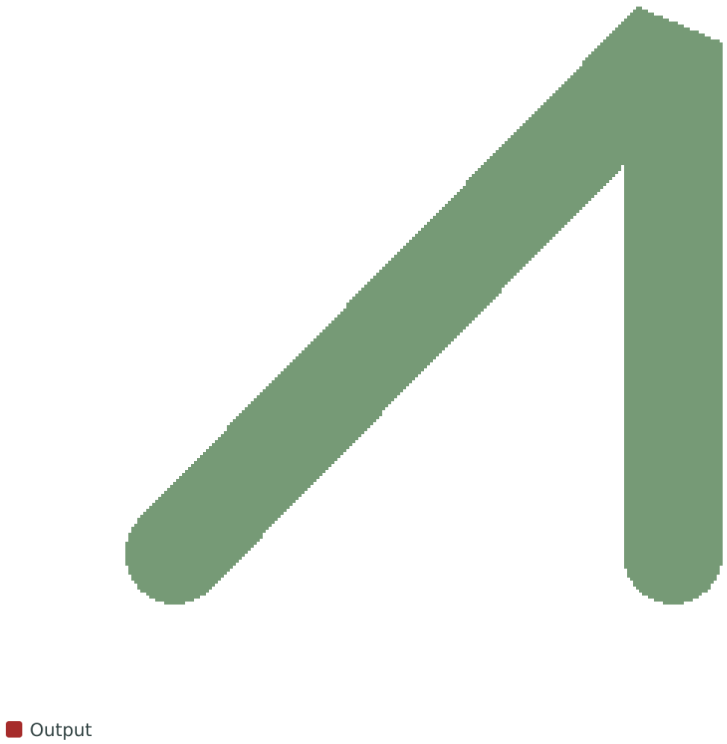


This example maps bands to RGB channels, producing the teal value (118,154,118).

Code

```
SELECT ST_AsPNG(
    ST_AsRaster(
        ST_Buffer('LINESTRING(50 50,150 150,150 50)'::geometry, 10, 'join=bevel'),
        200, 200,
        ARRAY['8BUI', '8BUI', '8BUI'],
        ARRAY[118, 154, 118],
        ARRAY[255, 255, 255]
    )
) AS image;
```

Figure



相关信息

ST_BandPixelType, ST_Buffer, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsJPEG, ST_SRID

11.3.3 ST_AsRasterAgg

ST_AsRasterAgg — 是一个聚合函数。将 PostGIS 几何形新的格像。

Synopsis

raster **ST_AsRasterAgg**(geometry geom, double precision val, raster ref, text pixeltype, double precision nodataval, text uniontype, boolean touched);

Description

返回一个波段格，包含所有入几何形及其关的渲染果。

可用性：3.6.0

示例

Code

```
WITH inp(g, v) AS (
    VALUES
        ( ST_Buffer(ST_MakePoint(10, 0), 10), 1 ),
        ( ST_Buffer(ST_MakePoint(20, 0), 10), 2 )
),
agg AS (
    SELECT ST_AsRasterAgg(
        g,
        v,
        ST_MakeEmptyRaster(0, 0, 0, 0, 1.0),
        '8BUI',
        99,
        'SUM',
        true
    ) r
    FROM inp
)
SELECT
    ST_Width(r) w,
    ST_Height(r) h,
    ST_Value(r, 'POINT(5 0)') v5_0,
    ST_Value(r, 'POINT(15 0)') v15_0,
    ST_Value(r, 'POINT(25 0)') v25_0
FROM agg;
```

Output

```
w  | h  | v5_0 | v15_0 | v25_0
----+----+------+-------+-------
 30 | 20 |    1 |     3 |     2
(1 row)
```

相关信息

ST_AsRaster, ST_DumpAsPolygons, ST_Union

11.3.4 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=,);

Description

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

This example creates two rasters: one containing band 1 of dummy_rast, and one containing band 2 reclassified as 2BUI.

Code

```
SELECT ST_NumBands(rast1) As numb1, ST_BandPixelType(rast1) As pix1,
 ST_NumBands(rast2) As numb2, ST_BandPixelType(rast2) As pix2
FROM (
    SELECT ST_Band(rast) As rast1, ST_Reclass(ST_Band(rast, 3), '100-200):1, [200-254:2', ' ↩
    2BUI') As rast2
        FROM dummy_rast
        WHERE rid = 2) As foo;
```

Output

```
numb1 | pix1 | numb2 | pix2
-------+------+-------+------
     1 | 8BUI |     1 | 2BUI
```

Return bands 2 and 3 using array cast syntax.

Code

```
SELECT ST_NumBands(ST_Band(rast, '{2,3}'::int[])) As num_bands
FROM dummy_rast WHERE rid=2;
```

Output

```
2
```

Return bands 2 and 3 using array syntax.

Code

```
SELECT ST_NumBands(ST_Band(rast, ARRAY[2, 3])) As num_bands
    FROM dummy_rast
WHERE rid=2;
```

Output

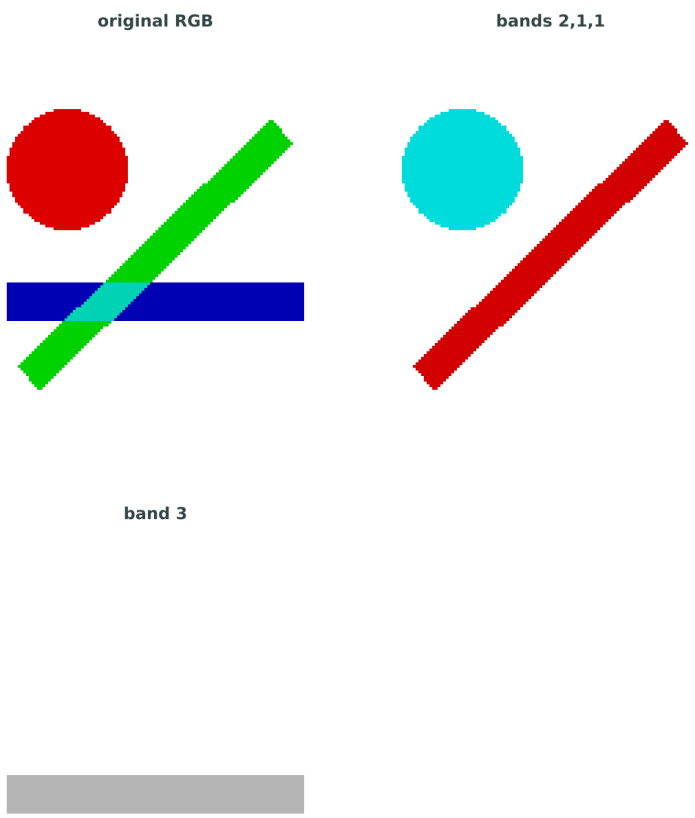
```
2
```

This example uses three distinct shapes as the red, green, and blue bands of a synthetic raster. It compares the original RGB raster with band 2 followed by band 1 twice, and with band 3 alone.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_AddBand(
            ST_AddBand(
                ST_MakeEmptyRaster(128, 128, 0, 128, 1, -1, 0, 0, 0),
                '8BUI'::text, 0::double precision, 0::double precision
            ),
            '8BUI'::text, 0::double precision, 0::double precision
        ),
        '8BUI'::text, 0::double precision, 0::double precision
    ) AS rast
), bands AS (
    SELECT ST_SetValue(
        ST_SetValue(
            ST_SetValue(rast, 1, ST_Buffer('POINT(32 96)'::geometry, 22), 220),
            2, ST_Buffer('LINESTRING(18 20,110 110)'::geometry, 6,
                         'endcap=flat join=bevel'), 210
        ),
        3, ST_Buffer('LINESTRING(10 48,118 48)'::geometry, 7,
                     'endcap=flat'), 180
    ) AS rast
    FROM canvas
)
SELECT ST_AsPNG(rast) AS "original RGB",
    ST_AsPNG(ST_Band(rast, ARRAY[2, 1, 1])) AS "bands 2,1,1",
    ST_AsPNG(ST_Band(rast, 3)) AS "band 3"
FROM bands;
```

Figure



Band selection from a multi-band raster.



原始（列⊠格）



双波段



⊠波段

相关信息

ST_AddBand, ST_NumBands, ST_Reclass, Chapter 11

11.3.5 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);

Description

使用 ST_MakeEmptyRaster 创建一个或多个格瓦片。格的宽度由宽度（`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
注意，对于格，scaley 通常为负，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。

Code

```
SELECT (ST_MetaData(tile)).* FROM ST_MakeEmptyCoverage(1, 1, 4, 4, 22, 33, (55 - 22)/(4):: ↩
    float, (33 - 77)/(4)::float, 0., 0., 4326) tile;
```

Output

```
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands
-----------------------------------------------------------------------------------------
        22 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     30.25 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      38.5 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     46.75 |         33 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
        22 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     30.25 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      38.5 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     46.75 |         22 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
        22 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     30.25 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      38.5 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     46.75 |         11 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
        22 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     30.25 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
      38.5 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
     46.75 |          0 |     1 |      1 |   8.25 |    -11 |     0 |     0 | 4326 |        0
```

相关信息

ST_MakeEmptyRaster

11.3.6 ST_MakeEmptyRaster

ST_MakeEmptyRaster — 返回一个空白格（不包含波段），其具有给定的尺寸（宽度和高度）、左上角 X 和 Y 坐标、像素大小和旋转参数（scalex、scaley、skewx 和 skewy），以及参考系统（SRID）。如果给定了一个格，则返回一个具有相同大小、对齐方式和 SRID 的新格。如果省略了 SRID，则空间参考将设置为未知（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);

Description

返回一个空白栅格（没有波段），其具有栅定的尺寸（栅度和高度）并以空栅（或世界）坐栅栅行地理参考，具有左上角 X 坐栅（upperleftx）、左上角 Y 坐栅（upperlefty）、像素大小和旋栅参数（scalex、scaley、skewx 和 skewy），以及参考系栅（SRID）。

最后一个版本使用栅个参数来指定像素大小（pixelsize）。scalex 栅置栅此参数，scaley 栅置栅此参数的栅栅。skewx 和 skewy 栅置栅 0。

如果栅入栅有栅格，它将返回具有相同元数据栅置（不栅波段）的新栅格。

如果未指定 srid，栅默栅栅 0。栅建空栅格后，您可能想要向其中添加波段并可能栅其栅行栅栅。参考 ST_AddBand定栅波段，参考 ST_SetValue栅置初始像素栅。

示例

Use an existing raster as a template and output metadata of the rasters we added.

Code

```
INSERT INTO dummy_rast(rid, rast)
VALUES(3, ST_MakeEmptyRaster(100, 100, 0.0005, 0.0005, 1, 1, 0, 0, 4326) );
INSERT INTO dummy_rast(rid, rast)
SELECT 4, ST_MakeEmptyRaster(rast)
FROM dummy_rast WHERE rid = 3;
SELECT
    rid,
    (md).width,
    (md).height,
    (md).scalex,
    (md).scaley,
    (md).srid
FROM (
    SELECT rid, ST_MetaData(rast) AS md
    FROM dummy_rast
    WHERE rid IN (3, 4)
) AS foo
ORDER BY rid;
```

Output

```
rid | width | height | scalex | scaley | srid
-----+-------+--------+--------+--------+------
   3 |   100 |    100 |      1 |      1 | 4326
   4 |   100 |    100 |      1 |      1 | 4326
```

相关信息

ST_AddBand, ST_MetaData, ST_ScaleX, ST_ScaleY, ST_SetValue, ST_SkewX, , ST_SkewY

11.3.7 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);

Description

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

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     1, 0), 2, '8BUI', 10, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     2, 0), 2, '8BUI', 20, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     3, 0), 2, '8BUI', 30, 0) AS rast UNION ALL

    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 4, 0), 2, '8BUI', 40, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 5, 0), 2, '8BUI', 50, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 6, 0), 2, '8BUI', 60, 0) AS rast UNION ALL

    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 7, 0), 2, '8BUI', 70, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 8, 0), 2, '8BUI', 80, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 9, 0), 2, '8BUI', 90, 0) AS rast
), bar AS (
    SELECT ST_Union(rast) AS rast FROM foo
```

```
), baz AS (
    SELECT
        row_number() OVER (
            ORDER BY ST_UpperLeftY(rast) DESC, ST_UpperLeftX(rast)
        ) AS tile_id,
        rast
    FROM (
        SELECT ST_Tile(rast, 3, 3, TRUE) AS rast
        FROM bar
    ) AS tiled
)
SELECT
    tile_id,
    ST_UpperLeftX(rast) AS ulx,
    ST_UpperLeftY(rast) AS uly,
    ST_Value(rast, 1, 1, 1) AS band1,
    ST_Value(rast, 2, 1, 1) AS band2
FROM baz;
```

Output

```
tile_id | ulx | uly | band1 | band2
---------+-----+-----+-------+-------
       1 |   0 |   0 |     1 |    10
       2 |   3 |   0 |     2 |    20
       3 |   6 |   0 |     3 |    30
       4 |   0 |  -3 |     4 |    40
       5 |   3 |  -3 |     5 |    50
       6 |   6 |  -3 |     6 |    60
       7 |   0 |  -6 |     7 |    70
       8 |   3 |  -6 |     8 |    80
       9 |   6 |  -6 |     9 |    90
(9 rows)
```

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     1, 0), 2, '8BUI', 10, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     2, 0), 2, '8BUI', 20, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, 0, 1, -1, 0, 0, 0), 1, '8BUI', ↩
     3, 0), 2, '8BUI', 30, 0) AS rast UNION ALL

    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 4, 0), 2, '8BUI', 40, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 5, 0), 2, '8BUI', 50, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -3, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 6, 0), 2, '8BUI', 60, 0) AS rast UNION ALL

    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 7, 0), 2, '8BUI', 70, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 3, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 8, 0), 2, '8BUI', 80, 0) AS rast UNION ALL
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 6, -6, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 9, 0), 2, '8BUI', 90, 0) AS rast
), bar AS (
    SELECT ST_Union(rast) AS rast FROM foo
), baz AS (
    SELECT
        row_number() OVER (
            ORDER BY ST_UpperLeftY(rast) DESC, ST_UpperLeftX(rast)
        ) AS tile_id,
        rast
```

```
    FROM (
        SELECT ST_Tile(rast, ARRAY[2], 3, 3) AS rast
        FROM bar
    ) AS tiled
)
SELECT
    tile_id,
    ST_UpperLeftX(rast) AS ulx,
    ST_UpperLeftY(rast) AS uly,
    ST_Value(rast, 1, 1, 1) AS band_value
FROM baz;
```

Output

```
tile_id | ulx | uly | band_value
---------+-----+-----+------------
       1 |   0 |   0 |         10
       2 |   3 |   0 |         20
       3 |   6 |   0 |         30
       4 |   0 |  -3 |         40
       5 |   3 |  -3 |         50
       6 |   6 |  -3 |         60
       7 |   0 |  -6 |         70
       8 |   3 |  -6 |         80
       9 |   6 |  -6 |         90
(9 rows)
```

相关信息

ST_Union, ST_Retile

11.3.8 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');

Description

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

11.3.9 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);

Description

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，该函数将根据自从 GDAL 栅格分配 SRID。如果提供了 `srid`，则提供的将覆盖任何自动分配的 SRID。

可用性：2.1.0

示例

Code

```
WITH foo AS (
    SELECT ST_AsPNG(ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 0.1,  ↩
    -0.1, 0, 0, 4326), 1, '8BUI', 1, 0), 2, '8BUI', 2, 0), 3, '8BUI', 3, 0)) AS png
),
bar AS (
    SELECT 1 AS rid, ST_FromGDALRaster(png) AS rast FROM foo
    UNION ALL
    SELECT 2 AS rid, ST_FromGDALRaster(png, 3310) AS rast FROM foo
)
SELECT
    rid,
    ST_MetaData(rast) AS metadata,
    ST_SummaryStats(rast, 1) AS stats1,
    ST_SummaryStats(rast, 2) AS stats2,
    ST_SummaryStats(rast, 3) AS stats3
FROM bar
ORDER BY rid;
```

Output

```
rid |         metadata          |    stats1     |    stats2     |     stats3
-----+---------------------------+---------------+---------------+----------------
   1 | (0,0,2,2,1,-1,0,0,0,3)    | (4,4,1,0,1,1) | (4,8,2,0,2,2) | (4,12,3,0,3,3)
   2 | (0,0,2,2,1,-1,0,0,3310,3) | (4,4,1,0,1,1) | (4,8,2,0,2,2) | (4,12,3,0,3,3)
(2 rows)
```

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);

Description

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:

Output

```
scalex
skewy
skewx
scaley
upperleftx
upperlefty
```

ESRI:

Output

```
scalex
skewy
skewx
scaley
upperleftx + scalex*0.5
upperlefty + scaley*0.5
```

示例

Code

```
SELECT
    format,
    replace(ST_GeoReference(rast, format), E'\n', ' | ') AS georef
FROM dummy_rast
CROSS JOIN (VALUES ('ESRI'), ('GDAL')) AS formats(format)
WHERE rid = 1;
```

Output

```
format | georef
--------+-------------------------------------------------------------------------
 ESRI   | 2.0000000000 | 0.0000000000 | 0.0000000000 | 3.0000000000 | 1.5000000000 | 2.0000000000
 GDAL   | 2.0000000000 | 0.0000000000 | 0.0000000000 | 3.0000000000 | 0.5000000000 | 0.5000000000
(2 rows)
```

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);

Description

返回⊠格的高度（以像素⊠⊠位）。

示例

Code

```
SELECT rid, ST_Height(rast) As rastheight
FROM dummy_rast;
```

Output

```
rid | rastheight
-----+------------
   1 |         20
   2 |          5
```

相关信息

ST_Width

11.4.3 ST_IsEmpty

ST_IsEmpty — 如果⊠格⊠空（⊠度 = 0 且高度 = 0），⊠返回 true。否⊠，返回 false。

Synopsis

boolean **ST_IsEmpty**(raster rast);

Description

如果⊠格⊠空（⊠度 = 0 且高度 = 0），⊠返回 true。否⊠，返回 false。
可用性: 2.0.0

示例

Code

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(100, 100, 0, 0, 0, 0, 0, 0))
```

Output

```
f
```

Code

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(0, 0, 0, 0, 0, 0, 0, 0))
```

Output

```
t
```

相关信息

ST_HasNoBand

11.4.4 ST_MemSize

ST_MemSize — 返回⊠格占用的空⊠量（以字⊠⊠⊠位）。

Synopsis

integer **ST_MemSize**(raster rast);

Description

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

Code

```
SELECT ST_MemSize(ST_AsRaster(ST_Buffer(ST_Point(1, 5), 10, 1000), 150, 150, '8BUI')) AS ↩
    rast_mem;
```

Output

```
22568
```

相关信息

11.4.5 ST_MetaData

ST_MetaData — 返回有关栅格对象的基本元数据，例如像素大小、旋转（倾斜）、左上、左下等。

Synopsis

record **ST_MetaData**(raster rast);

Description

返回有关栅格对象的基本元数据，例如像素大小、旋转（倾斜）、左上、左下等。返回的列：upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands

示例

Code

```
SELECT
    rid,
    format('(%s, %s)', (md).upperleftx, (md).upperlefty) AS upper_left,
    format('%sx%s', (md).width, (md).height) AS size,
    format('(%s, %s)', (md).scalex, (md).scaley) AS scale,
    format('(%s, %s)', (md).skewx, (md).skewy) AS skew,
    (md).srid AS srid,
    (md).numbands AS numbands
FROM (
    SELECT rid, ST_MetaData(rast) AS md
    FROM dummy_rast
) AS foo
ORDER BY rid;
```

Output

```
rid |        upper_left        | size  |      scale      |  skew  | srid | numbands
-----+--------------------------+-------+-----------------+--------+------+----------
   1 | (0.5, 0.5)               | 10x20 | (2, 3)          | (0, 0) |    0 |        0
   2 | (3427927.75, 5793244)    | 5x5   | (0.05, -0.05)   | (0, 0) |    0 |        3
(2 rows)
```

相关信息

ST_BandMetaData, ST_NumBands

11.4.6 ST_NumBands

ST_NumBands — 返回栅格对象中的波段数。

Synopsis

integer **ST_NumBands**(raster rast);

Description

返回□格□象中的波段数。

示例

Code

```
SELECT rid, ST_NumBands(rast) As numbands
FROM dummy_rast;
```

Output

```
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);

Description

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

Rasters with no skew.

Code

```
SELECT ST_Height(rast) As rastheight, ST_PixelHeight(rast) As pixheight,
 ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
        ST_SkewY(rast) As skewy
FROM dummy_rast;
```

Output

```
rastheight | pixheight | scalex | scaley | skewx | skewy
-----------+-----------+--------+--------+-------+----------
        20 |         3 |      2 |      3 |     0 |        0
         5 |      0.05 |   0.05 |  -0.05 |     0 |        0
```

Rasters with skew different than 0.

Code

```
SELECT ST_Height(rast) As rastheight, ST_PixelHeight(rast) As pixheight,
 ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
        ST_SkewY(rast) As skewy
FROM (SELECT ST_SetSkew(rast, 0.5, 0.5) As rast
        FROM dummy_rast) As skewed;
```

Output

```
rastheight |     pixheight     | scalex | scaley | skewx | skewy
-----------+-------------------+--------+--------+-------+----------
        20 |  3.04138126514911 |      2 |      3 |   0.5 |      0.5
         5 | 0.502493781056044 |   0.05 |  -0.05 |   0.5 |      0.5
```

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);

Description

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。

The following executable example draws the two pixel basis vectors on the same coordinate frame as a skewed raster footprint. The `i direction` vector is `(scaleX, skewY)`; the `j direction` vector is `(skewX, scaleY)`.

示例

Pixel basis vectors and footprint for a raster with non-zero skew.

Code

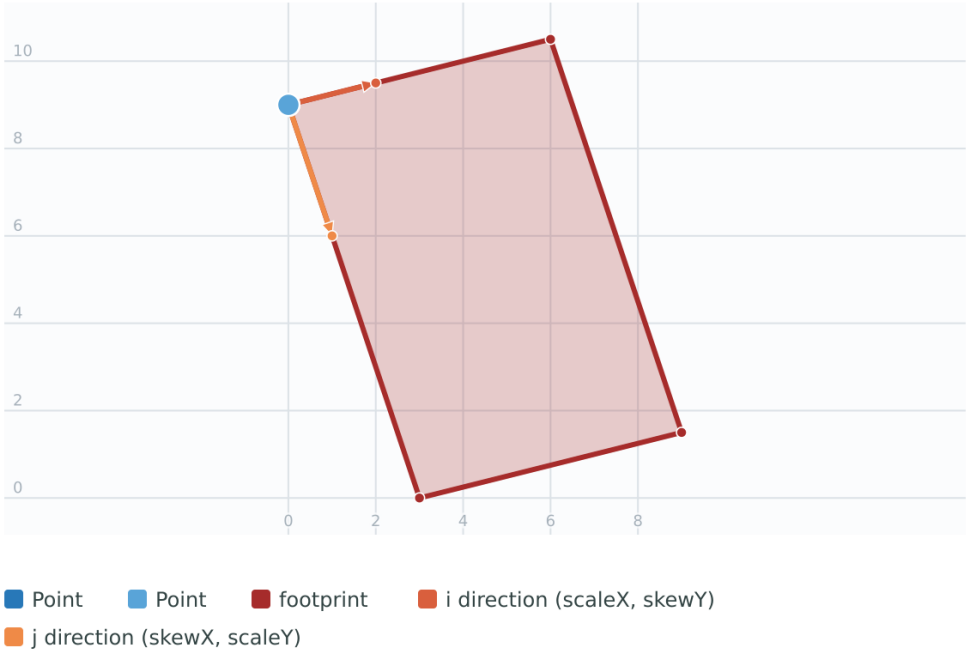
```
WITH raster AS (
    SELECT ST_MakeEmptyRaster(
        3, 3, 0, 0,
        2, -3, 1, 0.5,
        0
```

```
    ) AS rast
)
SELECT
    ST_ConvexHull(rast) AS footprint,
    ST_MakeLine(
        ST_Point(0, 0),
        ST_Point(ST_ScaleX(rast), ST_SkewY(rast))
    ) AS "i direction (scaleX, skewY)",
    ST_MakeLine(
        ST_Point(0, 0),
        ST_Point(ST_SkewX(rast), ST_ScaleY(rast))
    ) AS "j direction (skewX, scaleY)"
FROM raster;
```

Output

```
POLYGON((0 0,6 1.5,9 -7.5,3 -9,0 0)) | LINESTRING(0 0,2 0.5) | LINESTRING(0 0,1 -3)
```

Figure



Rasters with no skew.

Code

```
SELECT ST_Width(rast) As rastwidth, ST_PixelWidth(rast) As pixwidth,
ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
ST_SkewY(rast) As skewy
FROM dummy_rast;
```

Output

```
rastwidth | pixwidth | scalex | scaley | skewx | skewy
-----------+----------+--------+--------+-------+----------
10 |        2 |      2 |      3 |     0 |        0
 5 |     0.05 |   0.05 |  -0.05 |     0 |        0
```

Rasters with skew different than 0.

Code

```
SELECT ST_Width(rast) As rastwidth, ST_PixelWidth(rast) As pixwidth,
ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
ST_SkewY(rast) As skewy
FROM (SELECT ST_SetSkew(rast, 0.5, 0.5) As rast
FROM dummy_rast) As skewed;
```

Output

```
rastwidth |     pixwidth      | scalex | scaley | skewx | skewy
----------+-------------------+--------+--------+-------+----------
10 |  2.06155281280883 |      2 |      3 |   0.5 |      0.5
 5 | 0.502493781056044 |   0.05 |  -0.05 |   0.5 |      0.5
```

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);

Description

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

Code

```
SELECT rid, ST_ScaleX(rast) As rastpixwidth
FROM dummy_rast;
```

Output

```
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);

Description

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

Code

```
SELECT rid, ST_ScaleY(rast) As rastpixheight
FROM dummy_rast;
```

Output

```
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);

Description

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

This example uses a non-skewed raster.

Code

```
SELECT
    rid,
    (ST_RasterToWorldCoord(rast, 1, 1)).*,
    (ST_RasterToWorldCoord(rast, 2, 2)).*
FROM dummy_rast
```

Output

```
rid | longitude  | latitude | longitude |  latitude
-----+------------+----------+-----------+-----------
   1 |        0.5 |      0.5 |       2.5 |        3.5
   2 | 3427927.75 |  5793244 | 3427927.8 | 5793243.95
```

This example uses a skewed raster.

Code

```
SELECT
    rid,
    (ST_RasterToWorldCoord(rast, 1, 1)).*,
    (ST_RasterToWorldCoord(rast, 2, 3)).*
FROM (
    SELECT
        rid,
        ST_SetSkew(rast, 100.5, 0) As rast
    FROM dummy_rast
) As foo
```

Output

```
rid | longitude  | latitude | longitude | latitude
-----+------------+----------+-----------+----------
   1 |        0.5 |      0.5 |     203.5 |      6.5
   2 | 3427927.75 |  5793244 | 3428128.8 | 5793243.9
```

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);

Description

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

This example shows that providing a column is sufficient for a non-skewed raster.

Code

```
SELECT rid, ST_RasterToWorldCoordX(rast, 1) As x1coord,
    ST_RasterToWorldCoordX(rast, 2) As x2coord,
    ST_ScaleX(rast) As pixelx
FROM dummy_rast;
```

Output

```
rid |  x1coord   |  x2coord  | pixelx
-----+------------+-----------+--------
   1 |        0.5 |       2.5 |      2
   2 | 3427927.75 | 3427927.8 |   0.05
```

For fun let's skew it.

Code

```
SELECT rid, ST_RasterToWorldCoordX(rast, 1, 1) As x1coord,
    ST_RasterToWorldCoordX(rast, 2, 3) As x2coord,
    ST_ScaleX(rast) As pixelx
FROM (SELECT rid, ST_SetSkew(rast, 100.5, 0) As rast FROM dummy_rast) As foo;
```

Output

```
rid |  x1coord   |  x2coord  | pixelx
-----+------------+-----------+--------
   1 |        0.5 |     203.5 |      2
   2 | 3427927.75 | 3428128.8 |   0.05
```

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);

Description

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.

Code

```
SELECT rid, ST_RasterToWorldCoordY(rast, 1) As y1coord,
    ST_RasterToWorldCoordY(rast, 3) As y2coord,
    ST_ScaleY(rast) As pixely
FROM dummy_rast;
```

Output

```
rid | y1coord |  y2coord  | pixely
-----+---------+-----------+--------
   1 |     0.5 |       6.5 |      3
   2 | 5793244 | 5793243.9 |  -0.05
```

For fun let's skew it.

Code

```
SELECT rid, ST_RasterToWorldCoordY(rast, 1, 1) As y1coord,
    ST_RasterToWorldCoordY(rast, 2, 3) As y2coord,
    ST_ScaleY(rast) As pixely
FROM (SELECT rid, ST_SetSkew(rast, 0, 100.5) As rast FROM dummy_rast) As foo;
```

Output

```
rid | y1coord |  y2coord  | pixely
-----+---------+-----------+--------
   1 |     0.5 |       107 |      3
   2 | 5793244 | 5793344.4 |  -0.05
```

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);

Description

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

Code

```
SELECT rid, ST_Rotation(ST_SetScale(ST_SetSkew(rast, sqrt(2)), sqrt(2))) AS rot FROM ↩
    dummy_rast;
```

Output

```
rid |        rot
----+-------------------
  1 | 0.785398163397448
  2 | 0.785398163397448
```

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);

Description

返回地理参考 X ⊠斜（或旋⊠参数）。有关更多⊠⊠信息，⊠参⊠世界文件。

示例

Code

```
WITH refs AS (
    SELECT
        rid,
        format('(%s, %s)', ST_SkewX(rast), ST_SkewY(rast)) AS skew,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
)
SELECT
    rid,
    skew,
    term,
    georef[idx] AS value
FROM refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
```

```
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY rid, idx;
```

Output

```
rid |  skew  |  term   |       value
-----+--------+---------+--------------------
   1 | (0, 0) | scaleX  | 2.0000000000
   1 | (0, 0) | skewY   | 0.0000000000
   1 | (0, 0) | skewX   | 0.0000000000
   1 | (0, 0) | scaleY  | 3.0000000000
   1 | (0, 0) | originX | 0.5000000000
   1 | (0, 0) | originY | 0.5000000000
   2 | (0, 0) | scaleX  | 0.0500000000
   2 | (0, 0) | skewY   | 0.0000000000
   2 | (0, 0) | skewX   | 0.0000000000
   2 | (0, 0) | scaleY  | -0.0500000000
   2 | (0, 0) | originX | 3427927.7500000000
   2 | (0, 0) | originY | 5793244.0000000000
(12 rows)
```

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);

Description

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

Code

```
WITH refs AS (
    SELECT
        rid,
        format('(%s, %s)', ST_SkewX(rast), ST_SkewY(rast)) AS skew,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
)
SELECT
    rid,
    skew,
    term,
```

```
    georef[idx] AS value
FROM refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY rid, idx;
```

Output

```
rid |  skew  |  term   |       value
-----+--------+---------+--------------------
   1 | (0, 0) | scaleX  | 2.0000000000
   1 | (0, 0) | skewY   | 0.0000000000
   1 | (0, 0) | skewX   | 0.0000000000
   1 | (0, 0) | scaleY  | 3.0000000000
   1 | (0, 0) | originX | 0.5000000000
   1 | (0, 0) | originY | 0.5000000000
   2 | (0, 0) | scaleX  | 0.0500000000
   2 | (0, 0) | skewY   | 0.0000000000
   2 | (0, 0) | skewX   | 0.0000000000
   2 | (0, 0) | scaleY  | -0.0500000000
   2 | (0, 0) | originX | 3427927.7500000000
   2 | (0, 0) | originY | 5793244.0000000000
(12 rows)
```

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);

Description

返回在 Spatial_ref_sys 表中定⌧的⌧格⌧象的空⌧参考⌧⌧符。



Note
未做地理参考的 raster 和 geometry 使用 SRID 0。

示例

Code

```
SELECT ST_SRID(rast) As srid
FROM dummy_rast WHERE rid=1;
```

Output

```
0
```

相关信息

Section 4.5, ST_SRID

11.4.18 ST_Summary

ST_Summary — 返回⊠格内容的文本摘要。

Synopsis

text **ST_Summary**(raster rast);

Description

返回⊠格内容的文本摘要。

可用性：2.1.0

示例

Code

```
SELECT ST_Summary(ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1,  ↩
    0, 0, 0),
                1, '8BUI', 1, 0
            ),
            2, '32BF', 0, -9999
        ),
        3, '16BSI', 0, NULL
    )
);
```

Output

```
Raster of 10x10 pixels has 3 bands and extent of BOX(0 -10,10 0)+
     band 1 of pixtype 8BUI is in-db with NODATA value of 0      +
     band 2 of pixtype 32BF is in-db with NODATA value of -9999  +
     band 3 of pixtype 16BSI is in-db with no NODATA value
(1 row)
```

相关信息

ST_MetaData, ST_BandMetaData, ST_Summary ST_Extent

11.4.19 ST_UpperLeftX

ST_UpperLeftX — 返回投影空⌧参考中⌧格的左上角 X 坐⌧。

Synopsis

float8 **ST_UpperLeftX**(raster rast);

Description

返回投影空⌧参考中⌧格的左上角 X 坐⌧。

示例

Code

```
SELECt rid, ST_UpperLeftX(rast) As ulx
FROM dummy_rast;
```

Output

```
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);

Description

返回投影空⌧参考中⌧格的左上角 Y 坐⌧。

示例

Code

```
SELECT rid, ST_UpperLeftY(rast) As uly
FROM dummy_rast;
```

Output

```
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);

Description

返回⍰格的⍰度（以像素⍰⍰位）。

示例

Code

```
SELECT ST_Width(rast) As rastwidth
FROM dummy_rast WHERE rid=1;
```

Output

```
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);

Description

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

Code

```
SELECT
    rid,
    (ST_WorldToRasterCoord(rast, 3427927.8, 20.5)).*,
    (ST_WorldToRasterCoord(rast, ST_GeomFromText('POINT(3427927.8 20.5)', ST_SRID(rast)))) ↩
    .*
FROM dummy_rast;
```

Output

```
rid | columnx |   rowy    | columnx |   rowy
----+---------+-----------+---------+-----------
  1 | 1713964 |         7 | 1713964 |         7
  2 |       2 | 115864471 |       2 | 115864471
```

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);

Description

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

Code

```
SELECT rid, ST_WorldToRasterCoordX(rast, 3427927.8) As xcoord,
        ST_WorldToRasterCoordX(rast, 3427927.8, 20.5) As xcoord_xwyw,
        ST_WorldToRasterCoordX(rast, ST_GeomFromText('POINT(3427927.8 20.5)', ST_SRID(rast) ↩
    )) As ptxcoord
FROM dummy_rast;
```

Output

```
rid | xcoord  |  xcoord_xwyw   | ptxcoord
-----+---------+---------+----------
   1 | 1713964 | 1713964 |  1713964
   2 |       1 |       1 |        1
```

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);

Description

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

Code

```
SELECT rid, ST_WorldToRasterCoordY(rast, 20.5) As ycoord,
        ST_WorldToRasterCoordY(rast, 3427927.8, 20.5) As ycoord_xwyw,
        ST_WorldToRasterCoordY(rast, ST_GeomFromText('POINT(3427927.8 20.5)', ST_SRID(rast) ↩
    )) As ptycoord
FROM dummy_rast;
```

Output

```
rid |  ycoord   | ycoord_xwyw | ptycoord
-----+-----------+-------------+-----------
   1 |         7 |           7 |         7
   2 | 115864471 |   115864471 | 115864471
```

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);

Description

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

Variant 1.

Code

```
SELECT
    rid,
    (foo.md).*
FROM (
    SELECT
        rid,
        ST_BandMetaData(rast, 1) AS md
    FROM dummy_rast
    WHERE rid=2
) As foo;
```

Output

```
rid | pixeltype | nodatavalue | isoutdb | path | outdbbandnum
-----+-----------+-------------+---------+------+--------------
   2 | 8BUI      |           0 | f       |      |
(1 row)
```

Variant 2.

Code

```
WITH foo AS (
    SELECT
        ST_AddBand(NULL::raster, '/home/pele/devel/geo/postgis-git/raster/test/regress/ ↩
    loader/Projected.tif', NULL::int[]) AS rast
)
SELECT
    bandnum,
    pixeltype,
    isoutdb,
    regexp_replace(path, '^.*/', '') AS file,
    outdbbandnum
FROM ST_BandMetaData((SELECT rast FROM foo),
    ARRAY[1, 3, 2]::int[]
);
```

Output

```
bandnum | pixeltype | isoutdb |     file      | outdbbandnum
---------+-----------+---------+---------------+--------------
       1 | 8BUI      | t       | Projected.tif |            1
       3 | 8BUI      | t       | Projected.tif |            3
       2 | 8BUI      | t       | Projected.tif |            2
```

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);

Description

返回表示⌧波段中无数据的数⌧

示例

Code

```
SELECT ST_BandNoDataValue(rast, 1) As bnval1,
    ST_BandNoDataValue(rast, 2) As bnval2, ST_BandNoDataValue(rast, 3) As bnval3
FROM dummy_rast
WHERE rid = 2;
```

Output

```
bnval1 | bnval2 | bnval3
--------+--------+--------
      0 |      0 |      0
```

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=false);
boolean **ST_BandIsNoData**(raster rast, boolean forceChecking=false);

Description

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
If the flag is dirty (this is, the result is different using TRUE as last parameter and not using it) you should update the raster to set this flag to true, by using ST_SetBandIsNoData(), or ST_SetBandNoDataValue() with TRUE as last argument. See ST_SetBandIsNoData.

示例

This example creates a dummy table with one raster column. The raster has two bands and one pixel per band. In the first band, the NODATA value and pixel value are both 3. In the second band, the NODATA value is 13 and the pixel value is 4.

Code

```
create table dummy_rast (rid integer, rast raster);

insert into dummy_rast values(
1,
(
'01' -- little endian (uint8 ndr)
||
```

```
'0000' -- version (uint16 0)
||
'0200' -- nBands (uint16 0)
||
'17263529ED684A3F' -- scaleX (float64 0.000805965234044584)
||
'F9253529ED684ABF' -- scaleY (float64 -0.00080596523404458)
||
'1C9F33CE69E352C0' -- ipX (float64 -75.5533328537098)
||
'718F0E9A27A44840' -- ipY (float64 49.2824585505576)
||
'ED50EB853EC32B3F' -- skewX (float64 0.000211812383858707)
||
'7550EB853EC32B3F' -- skewY (float64 0.000211812383858704)
||
'E6100000' -- SRID (int32 4326)
||
'0100' -- width (uint16 1)
||
'0100' -- height (uint16 1)
||
'6' -- hasnodatavalue and isnodata value set to true.
||
'2' -- first band type (4BUI)
||
'03' -- novalue==3
||
'03' -- pixel(0,0)==3 (same that nodata)
||
'0' -- hasnodatavalue set to false
||
'5' -- second band type (16BSI)
||
'0D00' -- novalue==13
||
'0400' -- pixel(0,0)==4
)::raster
);

select ST_BandIsNoData(rast, 1) from dummy_rast where rid = 1; -- Expected true
select ST_BandIsNoData(rast, 2) from dummy_rast where rid = 1; -- Expected false
```

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);

Description

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

Code

```
WITH foo AS (
    SELECT ST_AddBand(
        NULL::raster,
        '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif',
        NULL::int[]
    ) AS rast
)
SELECT bandnum, ST_BandPath(rast, bandnum) AS path
FROM foo
CROSS JOIN generate_series(1, 3) AS bandnum;
```

Output

```
bandnum |                                 path
---------+-----------------------------------------------------------------------
       1 | /home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
       2 | /home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
       3 | /home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
```

相关信息

ST_BandMetaData, ST_SetBandPath, ST_SetBandIndex

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);

Description

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

Code

```
SELECT ST_BandFileSize(rast, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
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);

Description

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

Code

```
SELECT ST_BandFileTimestamp(rast, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
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);

Description

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

Code

```
SELECT ST_BandPixelType(rast, 1) As btype1,
    ST_BandPixelType(rast, 2) As btype2, ST_BandPixelType(rast, 3) As btype3
FROM dummy_rast
WHERE rid = 2;
```

Output

```
btype1 | btype2 | btype3
--------+--------+--------
 8BUI   | 8BUI   | 8BUI
```

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);

Description

返回此像素⌧型可以存⌧的最小⌧。

示例

Code

```
SELECT ST_MinPossibleValue('16BSI');
```

Output

```
-32768
```

Code

```
SELECT ST_MinPossibleValue('8BUI');
```

Output

```
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);

Description

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

Code

```
SELECT rid, ST_HasNoBand(rast) As hb1, ST_HasNoBand(rast, 2) as hb2,
ST_HasNoBand(rast, 4) as hb4, ST_NumBands(rast) As numbands
FROM dummy_rast;
```

Output

```
rid | hb1 | hb2 | hb4 | numbands
-----+-----+-----+-----+----------
1 | t   | t   | t   |        0
2 | f   | f   | t   |        3
```

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);

Description

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

This example returns a raster pixel polygon.

Code

```
SELECT i, j, ST_PixelAsPolygon(foo.rast, i, j) As b1pgeom
FROM dummy_rast As foo
    CROSS JOIN generate_series(1, 2) As i
    CROSS JOIN generate_series(1, 1) As j
WHERE rid=2;
```

Output

```
i | j |                                                                              b1pgeom
---+---+--------------------------------------------------------------------------------
 1 | 1 | POLYGON((3427927.75 5793244,3427927.8 5793244,3427927.8 5793243.95,...
 2 | 1 | POLYGON((3427927.8 5793244,3427927.85 5793244,3427927.85 5793243.95, ..
```

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);

Description

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

When *exclude_nodata_value* is TRUE, only pixels whose values are not NODATA are returned. Unlike ST_DumpAsPolygons, this function returns one polygon per pixel rather than merging adjacent pixels with the same value.

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⊠。

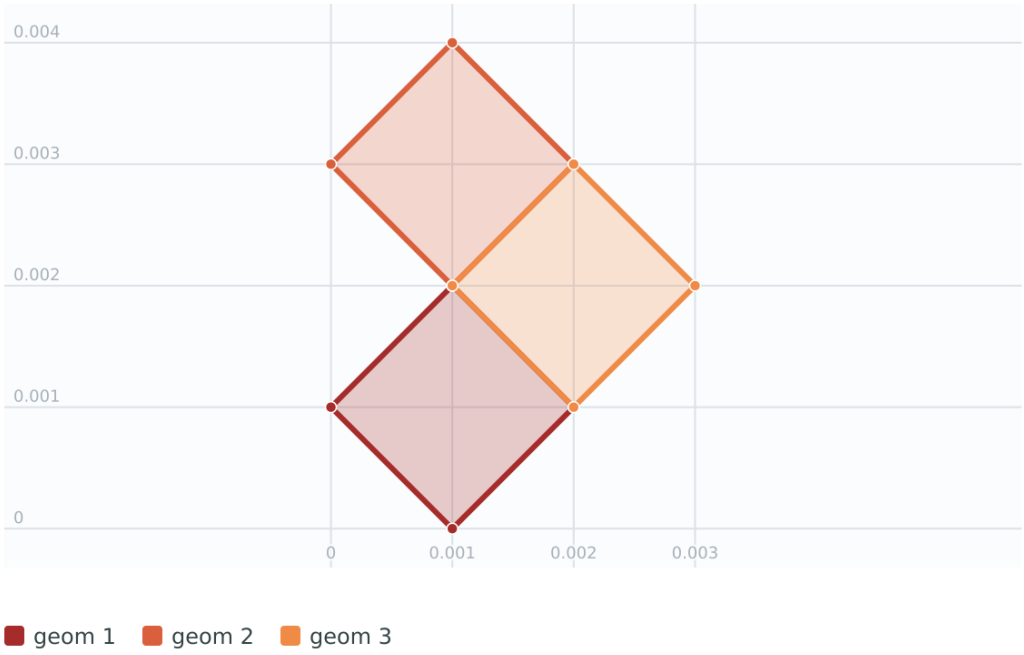
示例

This example returns a raster pixel polygon.

Code

```
WITH raster AS (
    SELECT ST_SetValue(
        ST_SetValue(
            ST_AddBand(
                ST_MakeEmptyRaster(
                    2, 2, 0, 0, 0.001, -0.001, 0.001, 0.001, 4269
                ),
                1, '8BUI', 1, 0
            ),
            2, 2, 10
        ),
        1, 1, NULL
    ) AS rast
), pixels AS (
    SELECT (ST_PixelAsPolygons(rast)).*
    FROM raster
)
SELECT x, y, val, geom AS geom
FROM pixels
ORDER BY x, y;
```

Figure



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);

Description

返回像素左上角的点几何形状。

可用性：2.1.0

示例

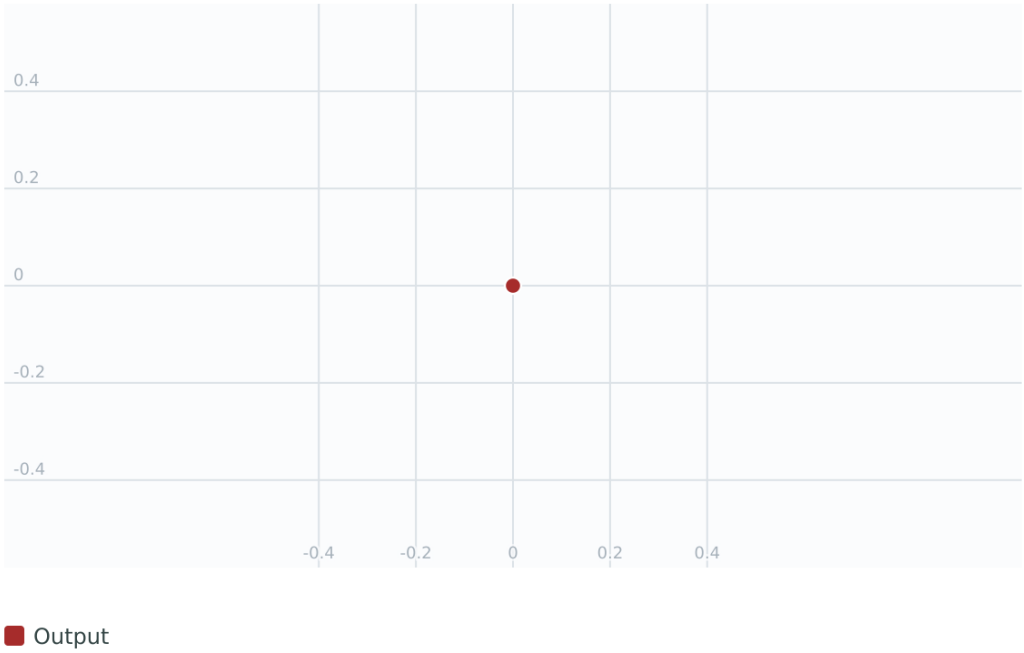
Code

```
SELECT ST_PixelAsPoint(rast, 1, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
POINT(0.5 0.5)
```

Figure



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);

Description

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 的行为。

示例

This example returns point features together with a derived marker size that a client can use for value-scaled point rendering.

Code

```
SELECT
    x,
    y,
    val,
    round((val / 32.0)::numeric, 2) AS marker_scale,geom AS geom
FROM (
    SELECT (ST_PixelAsPoints(rast, 1)).*
    FROM dummy_rast
    WHERE rid = 2
) AS foo
ORDER BY y, x;
```

Output

```
x | y | val | marker_scale |          geom
---+---+-----+--------------+----------------------------
 1 | 1 | 253 |         7.91 | POINT(3427927.75 5793244)
 2 | 1 | 254 |         7.94 | POINT(3427927.8 5793244)
 3 | 1 | 253 |         7.91 | POINT(3427927.85 5793244)
 4 | 1 | 254 |         7.94 | POINT(3427927.9 5793244)
 5 | 1 | 254 |         7.94 | POINT(3427927.95 5793244)
 1 | 2 | 253 |         7.91 | POINT(3427927.75 5793243.95)
 2 | 2 | 254 |         7.94 | POINT(3427927.8 5793243.95)
 3 | 2 | 254 |         7.94 | POINT(3427927.85 5793243.95)
 4 | 2 | 253 |         7.91 | POINT(3427927.9 5793243.95)
 5 | 2 | 249 |         7.78 | POINT(3427927.95 5793243.95)
 1 | 3 | 250 |         7.81 | POINT(3427927.75 5793243.9)
 2 | 3 | 254 |         7.94 | POINT(3427927.8 5793243.9)
 3 | 3 | 254 |         7.94 | POINT(3427927.85 5793243.9)
 4 | 3 | 252 |         7.88 | POINT(3427927.9 5793243.9)
 5 | 3 | 249 |         7.78 | POINT(3427927.95 5793243.9)
 1 | 4 | 251 |         7.84 | POINT(3427927.75 5793243.85)
 2 | 4 | 253 |         7.91 | POINT(3427927.8 5793243.85)
 3 | 4 | 254 |         7.94 | POINT(3427927.85 5793243.85)
 4 | 4 | 254 |         7.94 | POINT(3427927.9 5793243.85)
 5 | 4 | 253 |         7.91 | POINT(3427927.95 5793243.85)
 1 | 5 | 252 |         7.88 | POINT(3427927.75 5793243.8)
 2 | 5 | 250 |         7.81 | POINT(3427927.8 5793243.8)
 3 | 5 | 254 |         7.94 | POINT(3427927.85 5793243.8)
 4 | 5 | 254 |         7.94 | POINT(3427927.9 5793243.8)
 5 | 5 | 254 |         7.94 | POINT(3427927.95 5793243.8)
```

Figure



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);

Description

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

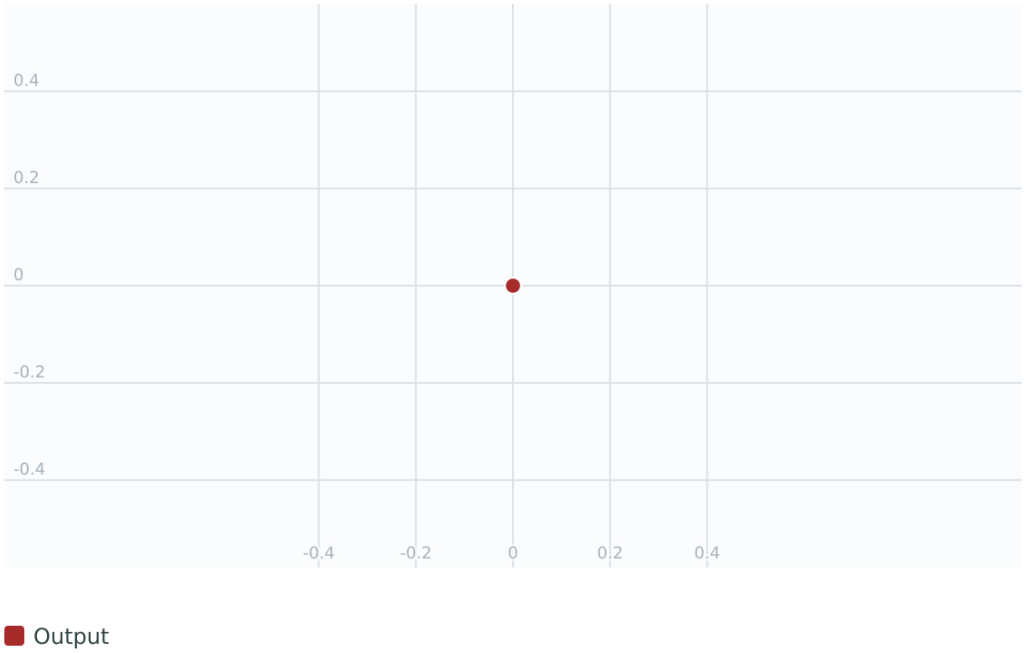
Code

```
SELECT ST_PixelAsCentroid(rast, 1, 1) FROM dummy_rast WHERE rid = 1;
```

Output

```
POINT(1.5 2)
```

Figure



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);

Description

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

This example uses LATERAL syntax.

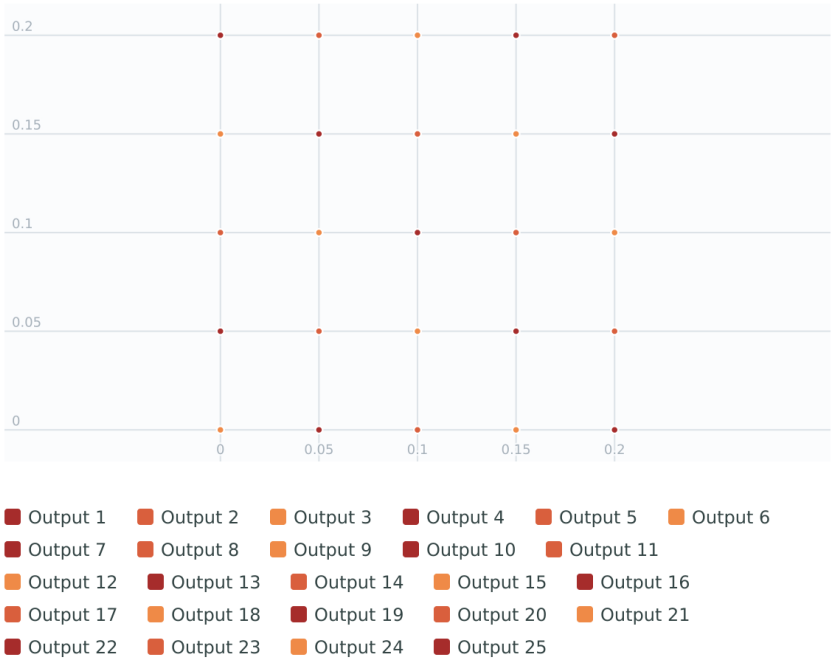
Code

```
SELECT x, y, val,geom
    FROM (SELECT dp.* FROM dummy_rast, LATERAL ST_PixelAsCentroids(rast, 1) AS dp WHERE rid ↩
     = 2) foo;
```

Output

```
x | y | val |           st_astext
---+---+-----+-----------------------------
 1 | 1 | 253 | POINT(3427927.775 5793243.975)
 2 | 1 | 254 | POINT(3427927.825 5793243.975)
 3 | 1 | 253 | POINT(3427927.875 5793243.975)
 4 | 1 | 254 | POINT(3427927.925 5793243.975)
 5 | 1 | 254 | POINT(3427927.975 5793243.975)
 1 | 2 | 253 | POINT(3427927.775 5793243.925)
 2 | 2 | 254 | POINT(3427927.825 5793243.925)
 3 | 2 | 254 | POINT(3427927.875 5793243.925)
 4 | 2 | 253 | POINT(3427927.925 5793243.925)
 5 | 2 | 249 | POINT(3427927.975 5793243.925)
 1 | 3 | 250 | POINT(3427927.775 5793243.875)
 2 | 3 | 254 | POINT(3427927.825 5793243.875)
 3 | 3 | 254 | POINT(3427927.875 5793243.875)
 4 | 3 | 252 | POINT(3427927.925 5793243.875)
 5 | 3 | 249 | POINT(3427927.975 5793243.875)
 1 | 4 | 251 | POINT(3427927.775 5793243.825)
 2 | 4 | 253 | POINT(3427927.825 5793243.825)
 3 | 4 | 254 | POINT(3427927.875 5793243.825)
 4 | 4 | 254 | POINT(3427927.925 5793243.825)
 5 | 4 | 253 | POINT(3427927.975 5793243.825)
 1 | 5 | 252 | POINT(3427927.775 5793243.775)
 2 | 5 | 250 | POINT(3427927.825 5793243.775)
 3 | 5 | 254 | POINT(3427927.875 5793243.775)
 4 | 5 | 254 | POINT(3427927.925 5793243.775)
 5 | 5 | 254 | POINT(3427927.975 5793243.775)
```

Figure



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

Description

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。

The allowed values of the `resample` parameter are "nearest" which performs the default nearest-neighbor resampling, "bilinear" which performs a bilinear interpolation to estimate the value between pixel centers, and the nearest-neighbor boundary options "nearest-UL", "nearest-UR", "nearest-LL",

and ”nearest-LR”. Boundary options choose which pixel is returned when the point lies on a horizontal or vertical pixel boundary, including corner intersections.

Enhanced: 3.7.0 resample accepts nearest-neighbor boundary options ”nearest-UL”, ”nearest-UR”, ”nearest-LL”, and ”nearest-LR”.

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.

Code

```
SELECT rid, ST_Value(rast, foo.pt_geom) As b1pval, ST_Value(rast, 2, foo.pt_geom) As b2pval
FROM dummy_rast CROSS JOIN (SELECT ST_SetSRID(ST_Point(3427927.77, 5793243.76), 0) As  ↩
    pt_geom) As foo
WHERE rid=2;
```

Output

```
rid | b1pval | b2pval
-----+--------+--------
   2 |    252 |     79
```

General fictitious example using a real table.

Code

```
SELECT rid, ST_Value(rast, 3, sometable.geom) As b3pval
FROM sometable
WHERE ST_Intersects(rast, sometable.geom);
```

Code

```
SELECT rid, ST_Value(rast, 1, 1, 1) As b1pval,
    ST_Value(rast, 2, 1, 1) As b2pval, ST_Value(rast, 3, 1, 1) As b3pval
FROM dummy_rast
WHERE rid=2;
```

Output

```
rid | b1pval | b2pval | b3pval
-----+--------+--------+--------
   2 |    253 |     78 |     70
```

Get all values in bands 1,2,3 of each pixel.

Code

```
SELECT x, y, ST_Value(rast, 1, x, y) As b1val,
    ST_Value(rast, 2, x, y) As b2val, ST_Value(rast, 3, x, y) As b3val
FROM dummy_rast CROSS JOIN
generate_series(1, 1000) As x CROSS JOIN generate_series(1, 1000) As y
WHERE rid =  2 AND x <= ST_Width(rast) AND y <= ST_Height(rast);
```

Output

```
 x | y | b1val | b2val | b3val
---+---+-------+-------+-------
 1 | 1 |   253 |    78 |    70
 1 | 2 |   253 |    96 |    80
 1 | 3 |   250 |    99 |    90
 1 | 4 |   251 |    89 |    77
 1 | 5 |   252 |    79 |    62
 2 | 1 |   254 |    98 |    86
 2 | 2 |   254 |   118 |   108
 :
 :
```

Get all values in bands 1,2,3 of each pixel same as above but returning the upper left point point of each pixel.

Code

```
SELECT ST_SetSRID(ST_Point(ST_UpperLeftX(rast) + ST_ScaleX(rast)*x,
        ST_UpperLeftY(rast) + ST_ScaleY(rast)*y),
        ST_SRID(rast)) As uplpt,
    ST_Value(rast, 1, x, y) As b1val,
    ST_Value(rast, 2, x, y) As b2val, ST_Value(rast, 3, x, y) As b3val
FROM dummy_rast CROSS JOIN
generate_series(1,1000) As x CROSS JOIN generate_series(1,1000) As y
WHERE rid =  2 AND x <= ST_Width(rast) AND y <= ST_Height(rast);
```

Output

```
            uplpt            | b1val | b2val | b3val
-----------------------------+-------+-------+-------
 POINT(3427929.25 5793245.5) |   253 |    78 |    70
 POINT(3427929.25 5793247)   |   253 |    96 |    80
 POINT(3427929.25 5793248.5) |   250 |    99 |    90
:
```

Get a polygon formed by union of all pixels that fall in a particular value range and intersect a particular polygon.

Code

```
SELECT ST_Union(pixpolyg) As shadow
FROM (SELECT ST_Translate(ST_MakeEnvelope(ST_UpperLeftX(rast), ST_UpperLeftY(rast),
            ST_UpperLeftX(rast) + ST_ScaleX(rast),
            ST_UpperLeftY(rast) + ST_ScaleY(rast), 0
            ), ST_ScaleX(rast)*x, ST_ScaleY(rast)*y
        ) As pixpolyg, ST_Value(rast, 2, x, y) As b2val
    FROM dummy_rast CROSS JOIN
generate_series(1,1000) As x CROSS JOIN generate_series(1,1000) As y
WHERE rid =  2
    AND x <= ST_Width(rast) AND y <= ST_Height(rast)) As foo
WHERE
    ST_Intersects(
        pixpolyg,
        ST_GeomFromText('POLYGON((3427928 5793244,3427927.75 5793243.75,3427928 ↩
    5793243.75,3427928 5793244))',0)
        ) AND b2val != 254;
```

Output

```
       shadow
-------------------------------------------------------------------------------
 MULTIPOLYGON(((3427928 5793243.9,3427928 5793243.85,3427927.95 5793243.85,3427927.95 5793243.9,
 3427927.95 5793243.95,3427928 5793243.95,3427928.05 5793243.95,3427928.05 5793243.9,3427928 5793243.9)),((3427927.95 ↩
    5793243.9,3427927.95 579324
```

```
3.85,3427927.9 5793243.85,3427927.85 5793243.85,3427927.85 5793243.9,3427927.9 5793243.9,3427927.9 5793243.95,
3427927.95 5793243.95,3427927.95 5793243.9)),((3427927.85 5793243.75,3427927.85 5793243.7,3427927.8 ↩
    5793243.7,3427927.8 5793243.75,
3427927.8 5793243.8,3427927.8 5793243.85,3427927.85 5793243.85,3427927.85 5793243.8,3427927.85 5793243.75)),
((3427928.05 5793243.75,3427928.05 5793243.7,3427928 5793243.7,3427927.95 5793243.7,3427927.95 5793243.75,3427927.95 ↩
    5793243.8,3427
927.95 5793243.85,3427928 5793243.85,3427928 5793243.8,3427928.05 5793243.8,
3427928.05 5793243.75)),((3427927.95 5793243.75,3427927.95 5793243.7,3427927.9 5793243.7,3427927.85 5793243.7,
3427927.85 5793243.75,3427927.85 5793243.8,3427927.85 5793243.85,3427927.9 5793243.85,
3427927.95 5793243.85,3427927.95 5793243.8,3427927.95 5793243.75)))
```

Checking all the pixels of a large raster tile can take a long time. You can dramatically improve speed at some lose of precision by orders of magnitude. By sampling pixels using the step optional parameter of generate_series. This next example does the same as previous but by checking 1 for every 4 (2x2) pixels and putting in the last checked. Putting in the checked pixel as the value for subsequent 4.

Code

```
SELECT ST_Union(pixpolyg) As shadow
FROM (SELECT ST_Translate(ST_MakeEnvelope(ST_UpperLeftX(rast), ST_UpperLeftY(rast),
            ST_UpperLeftX(rast) + ST_ScaleX(rast)*2,
            ST_UpperLeftY(rast) + ST_ScaleY(rast)*2, 0
            ), ST_ScaleX(rast)*x, ST_ScaleY(rast)*y
        ) As pixpolyg, ST_Value(rast, 2, x, y) As b2val
    FROM dummy_rast CROSS JOIN
generate_series(1,1000,2) As x CROSS JOIN generate_series(1,1000,2) As y
WHERE rid =  2
    AND x <= ST_Width(rast)  AND y <= ST_Height(rast)  ) As foo
WHERE
    ST_Intersects(
        pixpolyg,
        ST_GeomFromText('POLYGON((3427928 5793244,3427927.75 5793243.75,3427928 ↩
    5793243.75,3427928 5793244))',0)
        ) AND b2val != 254;
```

Output

```
        shadow
-----------------------------------------------------------------------------------
 MULTIPOLYGON(((3427927.9 5793243.85,3427927.8 5793243.85,3427927.8 5793243.95,
 3427927.9 5793243.95,3427928 5793243.95,3427928.1 5793243.95,3427928.1 5793243.85,3427928 ↩
    5793243.85,3427927.9 5793243.85)),
 ((3427927.9 5793243.65,3427927.8 5793243.65,3427927.8 5793243.75,3427927.8 ↩
    5793243.85,3427927.9 5793243.85,
 3427928 5793243.85,3427928 5793243.75,3427928.1 5793243.75,3427928.1 5793243.65,3427928 ↩
    5793243.65,3427927.9 5793243.65)))
```

相关信息

ST_DumpValues, ST_SetValue, ST_DumpAsPolygons, ST_NumBands, ST_PixelAsPolygon, ST_ScaleX, ST_ScaleY, ST_UpperLeftX, ST_UpperLeftY, ST_SRID, ST_AsText, ST_Point, ST_MakeEnvelope, ST_Intersec ST_Intersection

11.6.8 ST_NearestValue

ST_NearestValue — 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

Description

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

This example reads a value from pixel 2x2.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
            1, '8BUI', 1, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 1, 1, 0],
            [1, 1, 0, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_Value(rast, 2, 2) AS value,
    ST_NearestValue(rast, 2, 2) AS nearestvalue
FROM source
```

Output

```
value | nearestvalue
------+--------------
    1 |            1
```

This example reads a NODATA value from pixel 2x3.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
            1, '8BUI', 1, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 1, 1, 0],
            [1, 1, 0, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_Value(rast, 2, 3) AS value,
    ST_NearestValue(rast, 2, 3) AS nearestvalue
FROM source
```

Output

```
value | nearestvalue
------+--------------
      |            1
```

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);

Description

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.

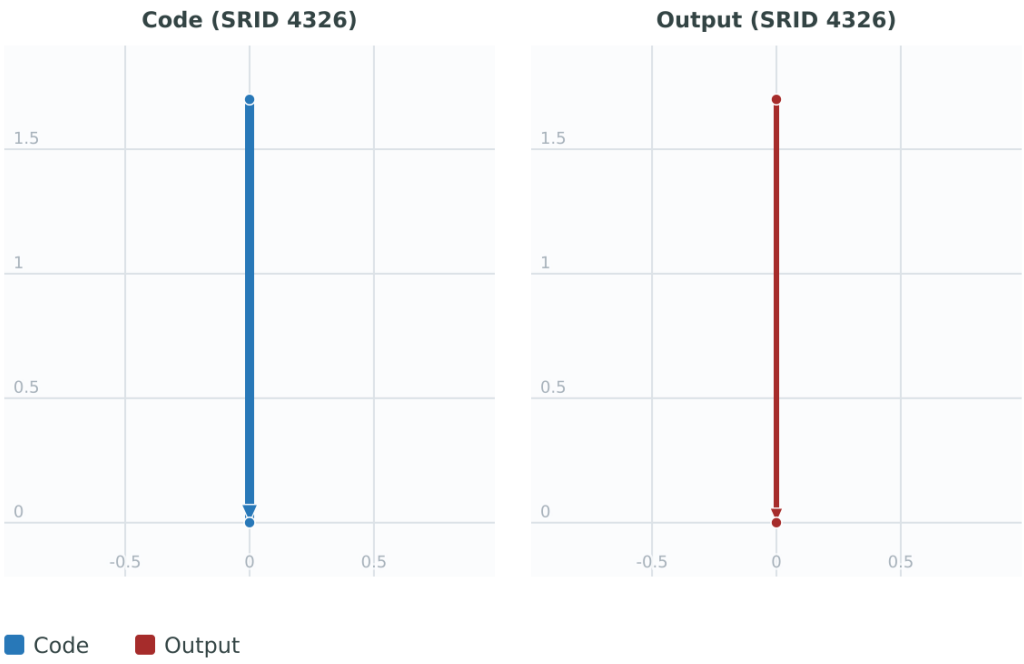
Code

```
WITH test_raster AS (
SELECT
ST_SetValues(
ST_AddBand(
ST_MakeEmptyRaster(
      width => 2,
      height => 2,
      upperleftx => 0,
      upperlefty => 2,
      scalex => 1.0,
      scaley => -1.0,
      skewx => 0,
      skewy => 0,
      srid => 4326),
    index => 1,
    pixeltype => '16BSI',
    initialvalue => 0,
    nodataval => -999
),
  1,
  1,
  1,
  newvalueset => ARRAY[ARRAY[10.0::float8, 50.0::float8], ARRAY[40.0::float8, 20.0::float8 ↩
    ]]) AS rast
)
SELECT
ST_SetZ(
    rast,
    band => 1,
    geom => 'SRID=4326;LINESTRING(1.0 1.9,1.0 0.2)'::geometry,
    resample => 'bilinear'
)
FROM test_raster
```

Output

```
LINESTRING Z (1 1.9 38,1 0.2 27)
```

Figure



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);

Description

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.

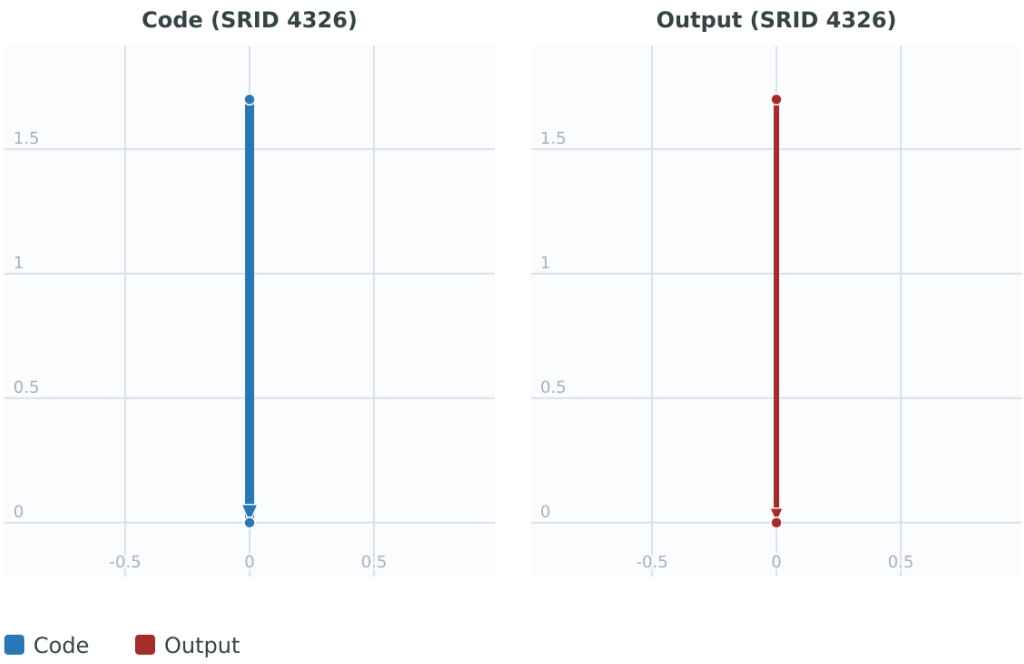
Code

```
WITH test_raster AS (
SELECT
ST_SetValues(
ST_AddBand(
ST_MakeEmptyRaster(
      width => 2,
      height => 2,
      upperleftx => 0,
      upperlefty => 2,
      scalex => 1.0,
      scaley => -1.0,
      skewx => 0,
      skewy => 0,
      srid => 4326),
    index => 1,
    pixeltype => '16BSI',
    initialvalue => 0,
    nodataval => -999
),
  1,
  1,
  1,
  newvalueset => ARRAY[ARRAY[10.0::float8, 50.0::float8], ARRAY[40.0::float8, 20.0::float8 ↩
    ]]) AS rast
)
SELECT
ST_SetM(
    rast,
    band => 1,
    geom => 'SRID=4326;LINESTRING(1.0 1.9,1.0 0.2)'::geometry,
    resample => 'bilinear'
)
FROM test_raster
```

Output

```
LINESTRING M (1 1.9 38,1 0.2 27)
```

Figure



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);

Description

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

This example reads a value from pixel 2x2.

Code

```
SELECT
    ST_Neighborhood(rast, 2, 2, 1, 1)
FROM (
    SELECT
        ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                '8BUI'::text, 1, 0
            ),
            1, 1, 1, ARRAY[
                [0, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 0],
                [1, 1, 0, 1, 1]
            ]::double precision[],
            1
        ) AS rast
) AS foo
```

Output

```
{{NULL,1,1},{1,1,1},{1,NULL,1}}
```

This example reads a NODATA value from pixel 2x3.

Code

```
SELECT
    ST_Neighborhood(rast, 2, 3, 1, 1)
FROM (
    SELECT
        ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                '8BUI'::text, 1, 0
            ),
            1, 1, 1, ARRAY[
                [0, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 0],
                [1, 1, 0, 1, 1]
            ]::double precision[],
```

```
            1
        ) AS rast
) AS foo
```

Output

```
{{1,1,1},{1,NULL,1},{1,1,1}}
```

Pixel 3x3 has value. Exclude_nodata_value = FALSE.

Code

```
SELECT
    ST_Neighborhood(rast, 3, 3, 1, 1, false)
FROM ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
                '8BUI'::text, 1, 0
            ),
            1, 1, 1, ARRAY[
                [0, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 0],
                [1, 1, 0, 1, 1]
            ]::double precision[],
            1
        ) AS rast
```

Output

```
{{1,1,0},{0,1,1},{1,1,1}}
```

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 — 返回修改后的栅格，其结果是将给定波段中的像素位置指定列 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，则默认为 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);

Description

返回修改后的栅格，栅格是通将指定像素的像素位置给定栅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，则假定为 1。

增强：2.1.0 ST_SetValue() 的几何体在支持任何几何类型，而不仅仅是点。几何体是 ST_SetValues() 的 geomval[] 元素的包装

示例

This example uses geometry input.

Code

```
               SELECT (foo.geomval).val, ST_Union((foo.geomval).geom)
FROM (SELECT ST_DumpAsPolygons(ST_SetValue(
            rast,
            1,
            ST_Point(3427927.75, 5793243.95),
            50)
            ) As geomval
FROM dummy_rast
where rid = 2) As foo
WHERE (foo.geomval).val < 250
GROUP BY (foo.geomval).val;
```

Output

```
 val |                                                        st_astext
-----+---------------------------------------------------------------------
  50 | POLYGON((3427927.75 5793244,3427927.75 5793243.95,3427927.8 579324 ...
 249 | POLYGON((3427927.95 5793243.95,3427927.95 5793243.85,3427928 57932 ...
```

This example stores the changed raster.

Code

```
UPDATE dummy_rast SET rast = ST_SetValue(rast, 1, ST_Point(3427927.75, 5793243.95), 100)
    WHERE rid = 2   ;
```

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);

Description

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

Variant 1.

These examples pass FALSE for *exclude_nodata_value* so that every raster cell remains visible. NODATA cells display the band's NODATA value, which is `0` here.

Set the lower-right 2 by 2 block of a 3 by 3 raster to 9.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 2, 2, ARRAY[[9, 9], [9, 9]]::double precision[][]
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
 x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

Replace the complete raster from an array; the central `NULL` becomes NODATA.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision[][]
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
 x | y | val
---+---+-----
 1 | 1 |   9
 1 | 2 |   9
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

Use the *noset* mask to retain the left-middle source pixel while replacing the other cells.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1,
                ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision[][],
                ARRAY[[false], [true]]::boolean[][]
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   9
 1 | 2 |   1
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

With *keepnodata* enabled, an existing NODATA pixel is not overwritten.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0,  ↩
    1, -1, 0, 0, 0),
                    1, '8BUI', 1, 0
                ),
                1, 1, 1, NULL
            ),
            1, 1, 1,
                ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision[][],
                ARRAY[[false], [true]]::boolean[][],
                TRUE
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   0
 1 | 2 |   1
 1 | 3 |   9
 2 | 1 |   9
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   9
 3 | 2 |   9
 3 | 3 |   9
```

Variant 2.

Use -1 as the value that should not be written; only the lower-right 2 by 2 block changes.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[-1, -1, -1], [-1, 9, 9], [-1, 9, 9]]::double precision[][], -1
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

The same update can use NULL as the value that should not be written.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[NULL, NULL, NULL], [NULL, 9, 9], [NULL, 9, 9]]::double  ↩
    precision[][], NULL::double precision
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

Variant 3.

Set a 2 by 2 rectangle starting at column 2, row 2.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0,  ↩
    0),
                1, '8BUI', 1, 0
            ),
            1, 2, 2, 2, 2, 9
        ),
        1, FALSE
```

```
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

When *keepnodata* is true, the NODATA cell inside the rectangle remains unchanged.

Code

```
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(ST_SetValues(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0,  ↩
    1, -1, 0, 0, 0),
                    1, '8BUI', 1, 0
                ),
                1, 2, 2, NULL
            ),
            1, 2, 2, 2, 2, 9, TRUE
        ),
        1, FALSE
    ) AS poly
) foo
ORDER BY 1, 2;
```

Output

```
x | y | val
---+---+-----
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   0
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9
```

Variant 5.

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI',  ↩
    0, 0) AS rast
), bar AS (
    SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)'::geometry geom UNION ALL
    SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1,4 -1,4 -4,1 -4,1 -1))'::geometry geom UNION ALL
    SELECT 3 AS gid, 'SRID=0;POLYGON((0 0,5 0,5 -1,1 -1,1 -4,0 -4,0 0))'::geometry geom  ↩
    UNION ALL
```

```
    SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0,4 4,4 -4)'::geometry
), changed AS (
    SELECT rid, gid, ST_SetValue(rast, 1, geom, gid) AS rast
    FROM foo
    CROSS JOIN bar
)
SELECT
    rid,
    gid,
    y,
    array_agg(ST_Value(rast, 1, x, y) ORDER BY x) AS values
FROM changed
CROSS JOIN generate_series(1, 5) AS x
CROSS JOIN generate_series(1, 5) AS y
GROUP BY rid, gid, y
ORDER BY rid, gid, y;
```

Output

```
rid | gid | y |           values
-----+-----+---+---------------------------
   1 |   1 | 1 | {NULL,NULL,NULL,NULL,NULL}
   1 |   1 | 2 | {NULL,NULL,NULL,NULL,NULL}
   1 |   1 | 3 | {NULL,NULL,1,NULL,NULL}
   1 |   1 | 4 | {NULL,NULL,NULL,NULL,NULL}
   1 |   1 | 5 | {NULL,NULL,NULL,NULL,NULL}
   1 |   2 | 1 | {NULL,NULL,NULL,NULL,NULL}
   1 |   2 | 2 | {NULL,2,2,2,NULL}
   1 |   2 | 3 | {NULL,2,2,2,NULL}
   1 |   2 | 4 | {NULL,2,2,2,NULL}
   1 |   2 | 5 | {NULL,NULL,NULL,NULL,NULL}
   1 |   3 | 1 | {3,3,3,3,3}
   1 |   3 | 2 | {3,NULL,NULL,NULL,NULL}
   1 |   3 | 3 | {3,NULL,NULL,NULL,NULL}
   1 |   3 | 4 | {3,NULL,NULL,NULL,NULL}
   1 |   3 | 5 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 1 | {4,NULL,NULL,NULL,NULL}
   1 |   4 | 2 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 3 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 4 | {NULL,NULL,NULL,NULL,NULL}
   1 |   4 | 5 | {NULL,NULL,NULL,NULL,4}
(20 rows)
```

下面⊠示了数⊠中后面的 geomvals 可以覆盖前面的 geomvals

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI',  ↩
    0, 0) AS rast
), bar AS (
    SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)'::geometry geom UNION ALL
    SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1,4 -1,4 -4,1 -4,1 -1))'::geometry geom UNION ALL
    SELECT 3 AS gid, 'SRID=0;POLYGON((0 0,5 0,5 -1,1 -1,1 -4,0 -4,0 0))'::geometry geom  ↩
    UNION ALL
    SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0,4 4,4 -4)'::geometry
), changed AS (
    SELECT ST_SetValues(
        rast,
        1,
        ARRAY[
            ROW((SELECT geom FROM bar WHERE gid = 1), 1),
            ROW((SELECT geom FROM bar WHERE gid = 2), 2)
        ]::geomval[]
    ) AS rast
    FROM foo
)
SELECT
```

```
    y,
    array_agg(ST_Value(rast, 1, x, y) ORDER BY x) AS values
FROM changed
CROSS JOIN generate_series(1, 5) AS x
CROSS JOIN generate_series(1, 5) AS y
GROUP BY y
ORDER BY y;
```

Output

```
y |          values
---+---------------------------
 1 | {NULL,NULL,NULL,NULL,NULL}
 2 | {NULL,2,2,2,NULL}
 3 | {NULL,2,2,2,NULL}
 4 | {NULL,2,2,2,NULL}
 5 | {NULL,NULL,NULL,NULL,NULL}
(5 rows)
```

⊠个例子与前面的例子相反

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI',  ↩
    0, 0) AS rast
), bar AS (
    SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)'::geometry geom UNION ALL
    SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1,4 -1,4 -4,1 -4,1 -1))'::geometry geom UNION ALL
    SELECT 3 AS gid, 'SRID=0;POLYGON((0 0,5 0,5 -1,1 -1,1 -4,0 -4,0 0))'::geometry geom  ↩
    UNION ALL
    SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0,4 4,4 -4)'::geometry
), changed AS (
    SELECT ST_SetValues(
        rast,
        1,
        ARRAY[
            ROW((SELECT geom FROM bar WHERE gid = 2), 2),
            ROW((SELECT geom FROM bar WHERE gid = 1), 1)
        ]::geomval[]
    ) AS rast
    FROM foo
)
SELECT
    y,
    array_agg(ST_Value(rast, 1, x, y) ORDER BY x) AS values
FROM changed
CROSS JOIN generate_series(1, 5) AS x
CROSS JOIN generate_series(1, 5) AS y
GROUP BY y
ORDER BY y;
```

Output

```
y |          values
---+---------------------------
 1 | {NULL,NULL,NULL,NULL,NULL}
 2 | {NULL,2,2,2,NULL}
 3 | {NULL,2,1,2,NULL}
 4 | {NULL,2,2,2,NULL}
 5 | {NULL,NULL,NULL,NULL,NULL}
(5 rows)
```

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);

Description

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 为 NULL 或未提供，则处理所有格波段。

可用性：2.1.0

示例

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), ↩
     1, '8BUI'::text, 1, 0), 2, '32BF'::text, 3, -9999), 3, '16BSI', 0, 0) AS rast
)
SELECT
    (ST_DumpValues(rast)).*
FROM foo;
```

Output

```
nband |                      valarray
------+-----------------------------------------------------
    1 | {{1,1,1},{1,1,1},{1,1,1}}
    2 | {{3,3,3},{3,3,3},{3,3,3}}
    3 | {{NULL,NULL,NULL},{NULL,NULL,NULL},{NULL,NULL,NULL}}
(3 rows)
```

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), ↩
     1, '8BUI'::text, 1, 0), 2, '32BF'::text, 3, -9999), 3, '16BSI', 0, 0) AS rast
)
SELECT
    (ST_DumpValues(rast, ARRAY[3, 1])).*
FROM foo;
```

Output

```
nband |                      valarray
------+-----------------------------------------------------
    3 | {{NULL,NULL,NULL},{NULL,NULL,NULL},{NULL,NULL,NULL}}
    1 | {{1,1,1},{1,1,1},{1,1,1}}
(2 rows)
```

Code

```
WITH foo AS (
    SELECT ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI' ↩
    , 1, 0), 1, 2, 5) AS rast
)
SELECT
    (ST_DumpValues(rast, 1))[2][1]
FROM foo;
```

Output

```
5
```

相关信息

ST_Value, ST_SetValue, ST_SetValues

11.6.15 ST_PixelOfValue

ST_PixelOfValue — 区取区等于搜索区的像素的列 x、行 y 坐区。

Synopsis

setof record **ST_PixelOfValue**(raster rast , integer nband , double precision[] search , boolean exclude_nodata_value=true);
setof record **ST_PixelOfValue**(raster rast , double precision[] search , boolean exclude_nodata_value=true);
setof record **ST_PixelOfValue**(raster rast , integer nband , double precision search , boolean exclude_nodata_value=true);
setof record **ST_PixelOfValue**(raster rast , double precision search , boolean exclude_nodata_value=true);

Description

区取区等于搜索区的像素的列 x、行 y 坐区。如果未指定波段，区假定区波段 1。

可用性：2.1.0

示例

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
            1, '8BUI', 1, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 1, 1],
```

```
            [1, 1, 1, 1, 255],
            [1, 1, 0, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT (ST_PixelOfValue(rast, 1, ARRAY[1, 255])).*
FROM source
```

Output

```
val | x | y
-----+---+---
   1 | 1 | 2
   1 | 1 | 3
   1 | 1 | 4
   1 | 1 | 5
   1 | 2 | 1
   1 | 2 | 2
   1 | 2 | 4
   1 | 2 | 5
   1 | 3 | 1
   1 | 3 | 2
   1 | 3 | 3
   1 | 3 | 4
   1 | 4 | 1
   1 | 4 | 3
   1 | 4 | 4
   1 | 4 | 5
   1 | 5 | 1
   1 | 5 | 2
   1 | 5 | 3
 255 | 5 | 4
   1 | 5 | 5
```

11.7 ☒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);

Description

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 uses the value order shown below.

Code

```
scalex skewy skewx scaley upperleftx upperlefty
```

ESRI uses the value order shown below.

Code

```
scalex skewy skewx scaley upperleftx + scalex*0.5 upperlefty + scaley*0.5
```



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

Code

```
WITH source AS (
    SELECT ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0) AS rast
), variants AS (
    SELECT 0 AS rid, rast FROM source
    UNION ALL
    SELECT 1, ST_SetGeoReference(rast, '10 0 0 -10 0.1 0.1', 'GDAL')
    FROM source
    UNION ALL
    SELECT 2, ST_SetGeoReference(rast, '10 0 0 -10 5.1 -4.9', 'ESRI')
    FROM source
    UNION ALL
    SELECT 3, ST_SetGeoReference(rast, 1, 1, 10, -10, 0.001, 0.001)
    FROM source
)
SELECT
    rid,
    round(ST_UpperLeftX(rast)::numeric, 3) AS upperleftx,
    round(ST_UpperLeftY(rast)::numeric, 3) AS upperlefty,
    ST_ScaleX(rast) AS scalex,
    ST_ScaleY(rast) AS scaley,
    ST_SkewX(rast) AS skewx,
    ST_SkewY(rast) AS skewy
FROM variants
ORDER BY rid;
```

Output

```
rid | upperleftx | upperlefty | scalex | scaley | skewx | skewy
-----+------------+------------+--------+--------+-------+-------
   0 |      0.000 |      0.000 |      1 |     -1 |     0 |     0
   1 |      0.100 |      0.100 |     10 |    -10 |     0 |     0
   2 |      0.100 |      0.100 |     10 |    -10 |     0 |     0
   3 |      1.000 |      1.000 |     10 |    -10 | 0.001 | 0.001
(4 rows)
```

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);

Description

均匀旋⌧⌧格。旋⌧以弧度⌧⌧位。有关更多⌧⌧信息，⌧参⌧世界文件。

示例

Code

```
WITH variants AS (
    SELECT rid, 'rotated' AS kind, ST_SetRotation(rast, 15) AS rast
    FROM dummy_rast
    UNION ALL
    SELECT rid, 'original', rast
    FROM dummy_rast
)
SELECT
    rid,
    kind,
    round(ST_ScaleX(rast)::numeric, 4) AS scale_x,
    round(ST_ScaleY(rast)::numeric, 4) AS scale_y,
    round(ST_SkewX(rast)::numeric, 4) AS skew_x,
    round(ST_SkewY(rast)::numeric, 4) AS skew_y
FROM variants
ORDER BY rid, kind;
```

Output

```
rid |   kind   | scale_x | scale_y | skew_x | skew_y
-----+----------+---------+---------+--------+--------
   1 | original |  2.0000 |  3.0000 | 0.0000 | 0.0000
   1 | rotated  | -1.5194 | -2.2791 | 1.9509 | 1.3006
   2 | original |  0.0500 | -0.0500 | 0.0000 | 0.0000
   2 | rotated  | -0.0380 | -0.0380 | 0.0325 | 0.0325
(4 rows)
```

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);

Description

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

Code

```
UPDATE dummy_rast
    SET rast = ST_SetScale(rast, 1.5)
WHERE rid = 2;

SELECT ST_ScaleX(rast) As pixx, ST_ScaleY(rast) As pixy, Box3D(rast) As newbox
FROM dummy_rast
WHERE rid = 2;
```

Output

```
pixx | pixy |                   newbox
------+------+-------------------------------------------------
  1.5 |  1.5 | BOX(3427927.75 5793244 0, 3427935.25 5793251.5 0)
```

Code

```
UPDATE dummy_rast
    SET rast = ST_SetScale(rast, 1.5, 0.55)
WHERE rid = 2;

SELECT ST_ScaleX(rast) As pixx, ST_ScaleY(rast) As pixy, Box3D(rast) As newbox
FROM dummy_rast
WHERE rid = 2;
```

Output

```
pixx | pixy |                   newbox
------+------+-----------------------------------------
  1.5 | 0.55 | BOX(3427927.75 5793244 0,3427935.25 5793247 0)
```

相关信息

ST_ScaleX, ST_ScaleY, Box3D

11.7.4 ST_SetSkew

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⊠。

Synopsis

raster **ST_SetSkew**(raster rast, float8 skewxy);
raster **ST_SetSkew**(raster rast, float8 skewx, float8 skewy);

Description

□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□。有关更多□□信息，□参□世界文件。

示例

This is the first example.

Code

```
UPDATE dummy_rast SET rast = ST_SetSkew(rast, 1, 2) WHERE rid = 1;
SELECT
    rid,
    format('(%s, %s)', skewx, skewy) AS skew,
    term,
    georef[idx] AS value
FROM (
    SELECT
        rid,
        ST_SkewX(rast) AS skewx,
        ST_SkewY(rast) AS skewy,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
    WHERE rid = 1
) AS refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY idx;
```

Output

```
rid |  skew  |  term   |    value
-----+--------+---------+--------------
   1 | (1, 2) | scaleX  | 2.0000000000
   1 | (1, 2) | skewY   | 2.0000000000
   1 | (1, 2) | skewX   | 1.0000000000
   1 | (1, 2) | scaleY  | 3.0000000000
   1 | (1, 2) | originX | 0.5000000000
   1 | (1, 2) | originY | 0.5000000000
(6 rows)
```

This second example sets both values to the same number.

Code

```
UPDATE dummy_rast SET rast = ST_SetSkew(rast, 0) WHERE rid = 1;
SELECT
    rid,
    format('(%s, %s)', skewx, skewy) AS skew,
```

```
    term,
    georef[idx] AS value
FROM (
    SELECT
        rid,
        ST_SkewX(rast) AS skewx,
        ST_SkewY(rast) AS skewy,
        string_to_array(ST_GeoReference(rast), E'\n') AS georef
    FROM dummy_rast
    WHERE rid = 1
) AS refs
CROSS JOIN (VALUES
    (1, 'scaleX'),
    (2, 'skewY'),
    (3, 'skewX'),
    (4, 'scaleY'),
    (5, 'originX'),
    (6, 'originY')
) AS parts(idx, term)
ORDER BY idx;
```

Output

```
rid |  skew  |  term   |    value
-----+--------+---------+-------------
   1 | (0, 0) | scaleX  | 2.0000000000
   1 | (0, 0) | skewY   | 0.0000000000
   1 | (0, 0) | skewX   | 0.0000000000
   1 | (0, 0) | scaleY  | 3.0000000000
   1 | (0, 0) | originX | 0.5000000000
   1 | (0, 0) | originY | 0.5000000000
(6 rows)
```

相关信息

ST_GeoReference, ST_SetGeoReference, ST_SkewX, ST_SkewY

11.7.5 ST_SetSRID

ST_SetSRID — 将⊠格的 SRID ⊠置⊠在 Spatial_ref_sys 表中定⊠的特定整数 srid。

Synopsis

raster **ST_SetSRID**(raster rast, integer srid);

Description

将⊠格上的 SRID ⊠置⊠特定整数⊠。



Note
此函数不会以任何方式⊠⊠⊠格 - 它只是⊠置定⊠其当前所在坐⊠参考系⊠的空⊠参考的元数据。⊠于以后的⊠⊠很有用。

相关信息

Section 4.5, ST_SRID

11.7.6 ST_SetUpperLeft

ST_SetUpperLeft — 将栅格左上角的栅栅置栅投影的 X 和 Y 坐栅。

Synopsis

raster **ST_SetUpperLeft**(raster rast, double precision x, double precision y);

Description

将栅格像素左上角的栅栅置栅投影的 X 和 Y 坐栅

示例

Code

```
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 — 重采栅一个栅格栅像，可以指定重新采栅算法、新的尺寸、任意的栅格角点，以及一栅栅格地理参考属性，栅些属性可以自己定栅，也可以从栅一个栅格栅像中借用。

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);

Description

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

Code

```
SELECT
    ST_Width(orig) AS orig_width,
    ST_Width(reduce_100) AS new_width
FROM (
    SELECT
        rast AS orig,
        ST_Resample(rast, 100, 100) AS reduce_100
    FROM aerials.boston
    WHERE ST_Intersects(
        rast,
        ST_Transform(ST_MakeEnvelope(-71.128, 42.2392, -71.1277, 42.2397, 4326),
            26986)
    )
    LIMIT 1
) AS foo;
```

Output

```
orig_width | new_width
-----------+-------------
       200 |         100
```

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);

Description

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。

This example returns the original raster pixel size.

Code

```
SELECT ST_PixelWidth(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0,  ↩
    4269), '8BUI'::text, 1, 0)) width
```

Output

```
0.001
```

This example returns the pixel size of the rescaled raster.

Code

```
SELECT ST_PixelWidth(ST_Rescale(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001,  ↩
    -0.001, 0, 0, 4269), '8BUI'::text, 1, 0), 0.0015)) width
```

Output

```
0.0015
```

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);

Description

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。

This example uses the original non-rotated raster.

Code

```
SELECT ST_Rotation(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0, 4269) ↩
    , '8BUI'::text, 1, 0));
```

Output

```
st_rotation
------------
           0
(1 row)
```

The reskewed raster rotation.

Code

```
SELECT ST_Rotation(ST_Reskew(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001,  ↩
    0, 0, 4269), '8BUI'::text, 1, 0), 0.0015));
```

Output

```
st_rotation
------------------
 -0.982793723247329
(1 row)
```

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);

Description

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。

This example returns the upper-left X coordinate of the original raster.

Code

```
SELECT ST_UpperLeftX(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 0.001, -0.001, 0, 0, 4269) ↩
    , '8BUI'::text, 1, 0));
```

Output

```
st_upperleftx
--------------
             0
(1 row)
```

The upper left of raster after snapping.

Code

```
SELECT ST_UpperLeftX(ST_SnapToGrid(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 0.001,  ↩
    -0.001, 0, 0, 4269), '8BUI'::text, 1, 0), 0.0002, 0.0002));
```

Output

```
st_upperleftx
--------------
       -0.0008
(1 row)
```

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);

Description

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

Code

```
WITH foo AS(
SELECT
    1 AS rid,
    ST_Resize(ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0),
            1, '8BUI', 255, 0
        ),
    '50%', '500') AS rast
UNION ALL
SELECT
    2 AS rid,
    ST_Resize(ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0),
            1, '8BUI', 255, 0
        ),
    500, 100) AS rast
UNION ALL
SELECT
    3 AS rid,
    ST_Resize(ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0),
            1, '8BUI', 255, 0
        ),
    0.25, 0.9) AS rast
)
SELECT
    rid,
    ST_Width(rast) AS width,
    ST_Height(rast) AS height,
    ST_ScaleX(rast) AS scale_x,
    round(ST_ScaleY(rast)::numeric, 4) AS scale_y
FROM foo
ORDER BY rid
```

Output

```
rid | width | height | scale_x | scale_y
-----+-------+--------+---------+---------
   1 |   500 |    500 |       2 | -2.0000
   2 |   500 |    100 |       2 | -10.0000
   3 |   250 |    900 |       4 | -1.1111
(3 rows)
```

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);

Description

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⊠⊠坐⊠⊠⊠功能异常，可能需要⊠置⊠境⊠量 PROJSO，指向 PostGIS 使用的投影⊠文件名。例如在 Windows 中，可以在“控制面板 -> 系⊠ -> ⊠境⊠量”中新增系⊠⊠量 PROJSO，将其⊠⊠⊠ `libproj.dll`。⊠置完成后需要重⊠ PostgreSQL 服⊠/守⊠⊠程使其生效。



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

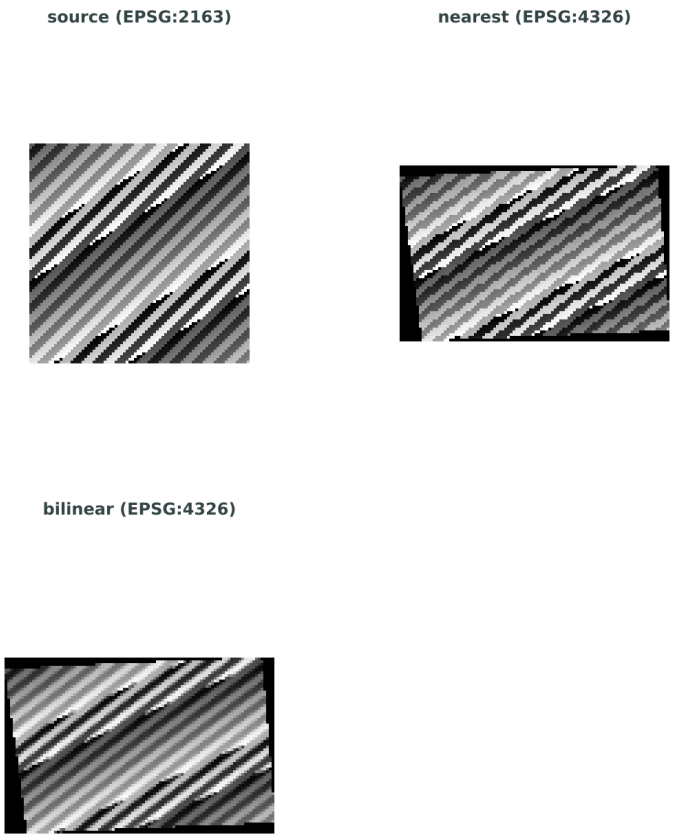
Compare a source raster with nearest-neighbor and bilinear reprojections. The diagonal bands make the resampling behavior visible: nearest-neighbor keeps hard pixel steps, while bilinear smooths values between pixels. All three figures are generated from this query at manual build time.

Code

```
WITH rows AS (
    SELECT
        y,
        array_agg(
            ((
                    x * 3 + y * 5
                    + CASE WHEN (x + y) % 8 < 4 THEN 80 ELSE 0 END
                ) % 255)::double precision
            ORDER BY x
        ) AS row_values
    FROM generate_series(1, 80) AS y
    CROSS JOIN generate_series(1, 80) AS x
    GROUP BY y
), source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(
                80, 80,
                -500000, 600000,
                2500, -2500,
                0, 0,
```

```
                2163
            ),
            1, '8BUI', 0, NULL
        ),
        1, 1, 1,
        array_agg(row_values ORDER BY y)::double precision[][]
    ) AS rast
    FROM rows
), transformed AS (
    SELECT
        rast,
        ST_Transform(rast, 4326, 'NearestNeighbor') AS nearest,
        ST_Transform(rast, 4326, 'Bilinear') AS bilinear
    FROM source
)
SELECT
    ST_AsPNG(rast) AS "source (EPSG:2163)",
    ST_AsPNG(nearest) AS "nearest (EPSG:4326)",
    ST_AsPNG(bilinear) AS "bilinear (EPSG:4326)"
FROM transformed;
```

Figure



Raster reprojection and resampling.



Massachusetts State Plane source



WGS 84 nearest-neighbor



WGS 84 bilinear

Variant 3.

下面展示了使用 ST_Transform(raster, srid) 和 ST_Transform(raster,alignto) 之⊠的区⊠

Code

```
WITH foo AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 600000, 100, -100, 0, 0,
    2163), 1, '16BUI', 1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 600000, 100, -100, 0, 0, 2163),
    1, '16BUI', 2, 0) AS rast UNION ALL
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 600000, 100, -100, 0, 0, 2163),
    1, '16BUI', 3, 0) AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 599800, 100, -100, 0, 0, 2163),
    1, '16BUI', 10, 0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 599800, 100, -100, 0, 0, 2163),
    1, '16BUI', 20, 0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 599800, 100, -100, 0, 0, 2163),
    1, '16BUI', 30, 0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 599600, 100, -100, 0, 0, 2163),
    1, '16BUI', 100, 0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 599600, 100, -100, 0, 0, 2163),
    1, '16BUI', 200, 0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 599600, 100, -100, 0, 0, 2163),
    1, '16BUI', 300, 0) AS rast
), bar AS (
    SELECT
        ST_Transform(rast, 4269) AS alignto
    FROM foo
    LIMIT 1
), baz AS (
    SELECT
        rid,
        rast,
        ST_Transform(rast, 4269) AS not_aligned,
        ST_Transform(rast, alignto) AS aligned
    FROM foo
    CROSS JOIN bar
)
SELECT
    ST_Collect(ST_ConvexHull(not_aligned) ORDER BY rid) AS not_aligned,
```

```
    ST_Collect(ST_ConvexHull(aligned) ORDER BY rid) AS aligned
FROM baz
```

Output

```
MULTIPOLYGON(((-107.007 50.2,-107.005 50.2,-107.005 50.198,-107.007 50.198,-107.007 50.2)),((-107.004 50.201,-107.002 ↩
    50.201,-107.002 50.198,-107.004 50.198,-107.004 50.201)),((-107.001 50.201,-106.999 50.201,-106.999 ↩
    50.198,-107.001 50.198,-107.001 50.201)),((-107.007 50.199,-107.004 50.199,-107.004 50.196,-107.007 ↩
    50.196,-107.007 50.199)),((-107.004 50.199,-107.001 50.199,-107.001 50.196,-107.004 50.196,-107.004 50.199)) ↩
    ,((-107.001 50.199,-106.999 50.199,-106.999 50.196,-107.001 50.196,-107.001 50.199)),((-107.006 50.197,-107.004 ↩
    50.197,-107.004 50.194,-107.006 50.194,-107.006 50.197)),((-107.004 50.197,-107.001 50.197,-107.001 ↩
    50.194,-107.004 50.194,-107.004 50.197)),((-107.001 50.197,-106.998 50.197,-106.998 50.195,-107.001 ↩
    50.195,-107.001 50.197))) | MULTIPOLYGON(((-107.007 50.2,-107.005 50.2,-107.005 50.198,-107.007 50.198,-107.007 ↩
    50.2)),((-107.005 50.202,-107.001 50.202,-107.001 50.198,-107.005 50.198,-107.005 50.202)),((-107.002 ↩
    50.202,-106.998 50.202,-106.998 50.198,-107.002 50.198,-107.002 50.202)),((-107.007 50.199,-107.003 ↩
    50.199,-107.003 50.195,-107.007 50.195,-107.007 50.199)),((-107.005 50.199,-107.001 50.199,-107.001 ↩
    50.195,-107.005 50.195,-107.005 50.199)),((-107.002 50.199,-106.998 50.199,-106.998 50.195,-107.002 ↩
    50.195,-107.002 50.199)),((-107.007 50.198,-107.003 50.198,-107.003 50.194,-107.007 50.194,-107.007 50.198)) ↩
    ,((-107.005 50.198,-107.001 50.198,-107.001 50.194,-107.005 50.194,-107.005 50.198)),((-107.002 50.198,-106.998 ↩
    50.198,-106.998 50.194,-107.002 50.194,-107.002 50.198)))
```

Figure



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);

Description

⊠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

This example changes only the first band NODATA value.

Code

```
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast, 1, 254)
WHERE rid = 2;
```

This example changes the NODATA value for bands 1, 2, and 3.

Code

```
UPDATE dummy_rast
    SET rast =
        ST_SetBandNoDataValue(ST_SetBandNoDataValue(ST_SetBandNoDataValue(rast, 1, 254),
                2, 99),
                3, 108)
        WHERE rid = 2;
```

This example removes the NODATA value so that all pixels are considered by processing functions.

Code

```
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast, 1, NULL)
WHERE rid = 2;
```

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);

Description

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

This example creates a dummy table with one raster column. The raster has two bands and one pixel per band. In the first band, the NODATA value and pixel value are both 3. In the second band, the NODATA value is 13 and the pixel value is 4.

Code

```
create table dummy_rast (rid integer, rast raster);

insert into dummy_rast values(
1,
(
'01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0200' -- nBands (uint16 0)
||
'17263529ED684A3F' -- scaleX (float64 0.000805965234044584)
||
'F9253529ED684ABF' -- scaleY (float64 -0.00080596523404458)
||
'1C9F33CE69E352C0' -- ipX (float64 -75.5533328537098)
||
'718F0E9A27A44840' -- ipY (float64 49.2824585505576)
||
'ED50EB853EC32B3F' -- skewX (float64 0.000211812383858707)
||
'7550EB853EC32B3F' -- skewY (float64 0.000211812383858704)
||
'E6100000' -- SRID (int32 4326)
||
'0100' -- width (uint16 1)
||
'0100' -- height (uint16 1)
||
'4' -- hasnodatavalue set to true, isnodata value set to false (when it should be true)
||
'2' -- first band type (4BUI)
||
'03' -- novalue==3
||
'03' -- pixel(0,0)==3 (same that nodata)
||
'0' -- hasnodatavalue set to false
||
'5' -- second band type (16BSI)
||
'0D00' -- novalue==13
||
'0400' -- pixel(0,0)==4
)::raster
);
```

```
select ST_BandIsNoData(rast, 1) from dummy_rast where rid = 1; -- Expected false
select ST_BandIsNoData(rast, 1, TRUE) from dummy_rast where rid = 1; -- Expected true
```

The `isnodata` flag is dirty, so set it to true.

Code

```
update dummy_rast set rast = ST_SetBandIsNoData(rast, 1) where rid = 1;


select ST_BandIsNoData(rast, 1) from dummy_rast where rid = 1; -- Expected true
```

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)

Description

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

Code

```
WITH foo AS (
    SELECT ST_AddBand(
        NULL::raster,
        '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif',
        NULL::int[]
    ) AS rast
)
SELECT
    1 AS query,
```

```
    bandnum,
    regexp_replace(path, '^.*/', '') AS path,
    outdbbandnum
FROM ST_BandMetaData(
    (SELECT rast FROM foo),
    ARRAY[1, 3, 2]::int[]
) AS metadata
UNION ALL
SELECT
    2, bandnum,
    regexp_replace(path, '^.*/', ''),
    outdbbandnum
FROM ST_BandMetaData((
        SELECT
            ST_SetBandPath(rast,
                2,
                '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected2.tif ↩
    ',
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1, 3, 2]::int[]
) AS metadata
ORDER BY 1, 2;
```

In the second result set, band 2 now points at `Projected2.tif` and uses out-db band 1.

Output

```
query | bandnum | path           | outdbbandnum
------+---------+----------------+-------------
    1 |       1 | Projected.tif  |            1
    1 |       2 | Projected.tif  |            2
    1 |       3 | Projected.tif  |            3
    2 |       1 | Projected.tif  |            1
    2 |       2 | Projected2.tif |            1
    2 |       3 | Projected.tif  |            3
```

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);

Description

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` 被置为 true，则不会进行任何检查来确保外部栅格文件和 PostGIS 栅格之间的兼容性（例如投影、像素支持）。此模式适用于在外部栅格文件中移动波段的情况。



Note
在内部，此方法将索引 band 处的 PostGIS 栅格波段替换为新波段，而不是更新现有的路径信息。

可用性：2.5.0

示例

Code

```
WITH foo AS (
    SELECT ST_AddBand(
        NULL::raster,
        '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif',
        NULL::int[]
    ) AS rast
)
SELECT
    1 AS query,
    bandnum,
    regexp_replace(path, '^.*/', '') AS path,
    outdbbandnum
FROM ST_BandMetaData(
    (SELECT rast FROM foo),
    ARRAY[1, 3, 2]::int[]
) AS metadata
UNION ALL
SELECT
    2, bandnum,
    regexp_replace(path, '^.*/', ''),
    outdbbandnum
FROM ST_BandMetaData((
        SELECT
            ST_SetBandIndex(rast,
                2,
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1, 3, 2]::int[]
) AS metadata
ORDER BY 1, 2;
```

In the second result set, band 2 still points at the same file but now reads external band 1.

Output

```
query | bandnum | path          | outdbbandnum
------+---------+---------------+-------------
    1 |       1 | Projected.tif |            1
    1 |       2 | Projected.tif |            2
    1 |       3 | Projected.tif |            3
    2 |       1 | Projected.tif |            1
    2 |       2 | Projected.tif |            1
    2 |       3 | Projected.tif |            3
```

相关信息

ST_BandMetaData, ST_SetBandPath

11.9 ☒格波段☒☒和分析

11.9.1 ST_Count

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。

Synopsis

bigint **ST_Count**(raster rast, integer nband=1, boolean exclude_nodata_value=true);
bigint **ST_Count**(raster rast, boolean exclude_nodata_value);

Description

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.

Code

```
SELECT rid, ST_Count(ST_SetBandNoDataValue(rast, 249)) As exclude_nodata,
        ST_Count(ST_SetBandNoDataValue(rast, 249), false) As include_nodata
    FROM dummy_rast WHERE rid=2;
```

Output

```
rid | exclude_nodata | include_nodata
-----+----------------+---------------
   2 |             23 |             25
```

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);

Description

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

Code

```
WITH foo AS (
    SELECT
        rast.rast
    FROM (
        SELECT ST_SetValue(ST_SetValue(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(10, 10,  ↩
    10, 10, 2, 2, 0, 0, 0),
                           1, '64BF', 0, 0
                      ),
                      1, 1, 1, -10
                  ),
                  1, 5, 4, 0
              ),
              1, 5, 5, 3.14159
        ) AS rast
    ) AS rast
    FULL JOIN (
        SELECT generate_series(1, 10) AS id
    ) AS id
        ON 1 = 1
)
SELECT
    ST_CountAgg(rast, 1, TRUE) AS non_nodata_pixels
FROM foo;
```

Output

```
non_nodata_pixels
------------------
                20
(1 row)
```

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);

Description

返回每个 bin □定□格波段的□□集，其中包含最小□、最大□、□数、百分比。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□算所有像素。

`width` width：一个数□，指示每个 category/bin 的□度。如果箱的数量大于□度的数量，□□度会重复。
示例：9 个 bin，□度□ [a, b, c] 将□出□ [a, b, c, a, b, c, a, b, c]

`bins` 分□数量——□是您将从函数中返回的□□数（如果指定）。如果未指定，□自□□算分□数量。

`right` 从右□而不是从左□□算直方□（默□）。□将□估□ x 的□准从 [a, b) 更改□ (a, b]

更改：3.1.0 □除了 ST_Histogram(table_name, column_name) □体。

可用性: 2.0.0

示例

Single raster tile - compute histograms for bands 1, 2, 3 and autocompute bins.

Code

```
SELECT band, (stats).*
FROM (SELECT rid, band, ST_Histogram(rast, band) As stats
    FROM dummy_rast CROSS JOIN generate_series(1, 3) As band
     WHERE rid=2) As foo;
```

Output

```
band |  min  |  max  | count | percent
------+-------+-------+-------+---------
    1 |   249 |   250 |     2 |    0.08
    1 |   250 |   251 |     2 |    0.08
    1 |   251 |   252 |     1 |    0.04
    1 |   252 |   253 |     2 |    0.08
    1 |   253 |   254 |    18 |    0.72
    2 |    78 | 113.2 |    11 |    0.44
    2 | 113.2 | 148.4 |     4 |    0.16
    2 | 148.4 | 183.6 |     4 |    0.16
    2 | 183.6 | 218.8 |     1 |    0.04
    2 | 218.8 |   254 |     5 |     0.2
    3 |    62 | 100.4 |    11 |    0.44
    3 | 100.4 | 138.8 |     5 |     0.2
    3 | 138.8 | 177.2 |     4 |    0.16
    3 | 177.2 | 215.6 |     1 |    0.04
    3 | 215.6 |   254 |     4 |    0.16
```

Just band 2 but for 6 bins.

Code

```
SELECT (stats).*
FROM (SELECT rid, ST_Histogram(rast, 2, 6) As stats
    FROM dummy_rast
     WHERE rid=2) As foo;
```

Output

```
min     |    max     | count | percent
------------+------------+-------+---------
         78 | 107.333333 |     9 |    0.36
 107.333333 | 136.666667 |     6 |    0.24
 136.666667 |        166 |     0 |       0
        166 | 195.333333 |     4 |    0.16
 195.333333 | 224.666667 |     1 |    0.04
 224.666667 |        254 |     5 |     0.2
(6 rows)
```

Same as previous but we explicitly control the pixel value range of each bin.

Code

```
SELECT (stats).*
FROM (SELECT rid, ST_Histogram(rast, 2, 6, ARRAY[0.5, 1, 4, 100, 5]) As stats
    FROM dummy_rast
     WHERE rid=2) As foo;
```

Output

```
min  |  max  | count | percent
-------+-------+-------+----------
    78 |  78.5 |     1 |     0.08
  78.5 |  79.5 |     1 |     0.04
  79.5 |  83.5 |     0 |        0
  83.5 | 183.5 |    17 |   0.0068
 183.5 | 188.5 |     0 |        0
 188.5 |   254 |     6 | 0.003664
(6 rows)
```

相关信息

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg

11.9.4 ST_Quantile

ST_Quantile — 计算栅格本或栅格体上下文中栅格或栅格表覆盖范围的分位数。因此，可以用来某个值是否位于栅格的 25%、50%、75% 百分位值。

Synopsis

SETOF record **ST_Quantile**(raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] quantiles=NULL);
SETOF record **ST_Quantile**(raster rast, double precision[] quantiles);
SETOF record **ST_Quantile**(raster rast, integer nband, double precision[] quantiles);
double precision **ST_Quantile**(raster rast, double precision quantile);
double precision **ST_Quantile**(raster rast, boolean exclude_nodata_value, double precision quantile=NULL);
double precision **ST_Quantile**(raster rast, integer nband, double precision quantile);
double precision **ST_Quantile**(raster rast, integer nband, boolean exclude_nodata_value, double precision quantile);
double precision **ST_Quantile**(raster rast, integer nband, double precision quantile);

Description

⌧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⌧。



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

Example will consider only pixels of band 1 that are not 249 and in named quantiles.

Code

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast, 249) WHERE rid=2;

SELECT (pvq).*
FROM (SELECT ST_Quantile(rast, ARRAY[0.25, 0.75]) As pvq
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvq).quantile;
```

Output

```
quantile | value
----------+-------
    0.25 |   253
    0.75 |   254
```

Code

```
SELECT ST_Quantile(rast, 0.75) As value
FROM dummy_rast WHERE rid=2;
```

Output

```
254
```

Real live example. Quantile of all pixels in band 2 intersecting a geometry.

Code

```
SELECT rid, (ST_Quantile(rast, 2)).* As pvc
    FROM o_4_boston
        WHERE ST_Intersects(
            rast,
            ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706 ↩
    892151,224486 892151))', 26986)
            )
ORDER BY value, quantile, rid
;
```

Output

```
rid | quantile | value
-----+----------+-------
   1 |        0 |     0
   2 |        0 |     0
  14 |        0 |     1
  15 |        0 |     2
  14 |     0.25 |    37
   1 |     0.25 |    42
  15 |     0.25 |    47
   2 |     0.25 |    50
  14 |      0.5 |    56
   1 |      0.5 |    64
  15 |      0.5 |    66
   2 |      0.5 |    77
  14 |     0.75 |    81
  15 |     0.75 |    87
   1 |     0.75 |    94
   2 |     0.75 |   106
  14 |        1 |   199
   1 |        1 |   244
   2 |        1 |   255
  15 |        1 |   255
```

相关信息

ST_Count, ST_SummaryStats, ST_SummaryStatsAgg, ST_SetBandNoDataValue

11.9.5 ST_SummaryStats

ST_SummaryStats — 返回指定⊠格或⊠格覆盖中某个波段的⊠⊠摘要，包括像元数量（count）、⊠和（sum）、均⊠（mean）、⊠准差（stddev）、最小⊠（min）和最大⊠（max）。若未指定波段，默⊠使用第 1 波段。

Synopsis

summarystats **ST_SummaryStats**(raster rast, boolean exclude_nodata_value);
summarystats **ST_SummaryStats**(raster rast, integer nband, boolean exclude_nodata_value);

Description

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

Single raster tile.

Code

```
SELECT rid, band, (stats).*
FROM (SELECT rid, band, ST_SummaryStats(rast, band) As stats
    FROM dummy_rast CROSS JOIN generate_series(1, 3) As band
     WHERE rid=2) As foo;
```

Output

```
rid | band | count | sum  |    mean    |  stddev   | min | max
----+------+-------+------+------------+-----------+-----+-----
  2 |    1 |    23 | 5821 | 253.086957 |  1.248061 | 250 | 254
  2 |    2 |    25 | 3682 |     147.28 | 59.862188 |  78 | 254
  2 |    3 |    25 | 3290 |      131.6 | 61.647384 |  62 | 254
```

Summarize pixels that intersect buildings of interest.

This example selects the target buildings, clips band 2 of the raster tiles to each building boundary, computes statistics for each clipped region, and then summarizes those statistics by building. It took 574ms on PostGIS Windows 64-bit with all of Boston Buildings and aerial Tiles (tiles each 150x150 pixels ~ 134,000 tiles), ~102,000 building records.

Code

```
WITH
-- our features of interest
feat AS (
    SELECT gid AS building_id, geom_26986 AS geom
    FROM buildings AS b
    WHERE gid IN (100, 103, 150)
),
-- clip band 2 of raster tiles to boundaries of builds
-- then get stats for these clipped regions
b_stats AS (
    SELECT building_id, (stats).*
    FROM (
        SELECT
            building_id,
            ST_SummaryStats(ST_Clip(rast, 2, geom)) AS stats
        FROM aerials.boston
        INNER JOIN feat
            ON ST_Intersects(feat.geom, rast)
```

```
    ) AS foo
)
-- finally summarize stats
SELECT
    building_id,
    SUM(count) AS num_pixels,
    MIN(min) AS min_pval,
    MAX(max) AS max_pval,
    SUM(mean * count) / SUM(count) AS avg_pval
FROM b_stats
WHERE count
> 0
GROUP BY building_id
ORDER BY building_id;
```

Output

```
building_id | num_pixels | min_pval | max_pval |     avg_pval
------------+------------+----------+----------+------------------
        100 |       1090 |        1 |      255 | 61.0697247706422
        103 |        655 |        7 |      182 | 70.5038167938931
        150 |        895 |        2 |      252 | 185.642458100559
```

Raster coverage.

This example returns statistics for each band.

Code

```
SELECT band, (stats).*
FROM (SELECT band, ST_SummaryStats('o_4_boston', 'rast', band) As stats
    FROM generate_series(1, 3) As band) As foo;
```

Output

```
band |  count  |  sum   |       mean       |      stddev      | min | max
-----+---------+--------+------------------+------------------+-----+-----
   1 | 8450000 | 725799 | 82.7064349112426 | 45.6800222638537 |   0 | 255
   2 | 8450000 | 700487 | 81.4197705325444 | 44.2161184161765 |   0 | 255
   3 | 8450000 | 575943 |  74.682739408284 | 44.2143885481407 |   0 | 255
```

For a table, sampling less than 100% can improve speed. Here the sampling is set to 25%.

Code

```
SELECT band, (stats).*
FROM (SELECT band, ST_SummaryStats('o_4_boston', 'rast', band, true, 0.25) As stats
    FROM generate_series(1, 3) As band) As foo;
```

Output

```
band |  count  |  sum   |       mean       |      stddev      | min | max
-----+---------+--------+------------------+------------------+-----+-----
   1 | 2112500 | 180686 | 82.6890480473373 | 45.6961043857248 |   0 | 255
   2 | 2112500 | 174571 |  81.448503668639 | 44.2252623171821 |   0 | 255
   3 | 2112500 | 144364 | 74.6765884023669 | 44.2014869384578 |   0 | 255
```

相关信息

summarystats, ST_SummaryStatsAgg, ST_Count, ST_Clip

11.9.6 ST_SummaryStatsAgg

ST_SummaryStatsAgg — 聚合函数。返回一区区格中指定波段的区区摘要，包括像元数量（count）、区和（sum）、均区（mean）、区准差（stddev）、最小区（min）和最大区。若未指定波段，默区使用第 1 波段。

Synopsis

summarystats **ST_SummaryStatsAgg**(raster rast, integer nband, boolean exclude_nodata_value, double precision sample_percent);
summarystats **ST_SummaryStatsAgg**(raster rast, boolean exclude_nodata_value, double precision sample_percent);
summarystats **ST_SummaryStatsAgg**(raster rast, integer nband, boolean exclude_nodata_value);

Description

返回区格或区格覆盖范区的区定区格波段的summarystats区区信息，其中包含区数、区和、平均区、区准差、最小区、最大区。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 区的像素区。将 except_nodata_value 区置区 False 以区取所有像素的区数。



Note
默区情况下将区所有像素区行采区。区了区得更快的区区，区将 sample_percent 区置区 0 到 1 之区的区

可用性：2.2.0

示例

Code

```
WITH foo AS (
    SELECT
        rast.rast
    FROM (
        SELECT ST_SetValue(ST_SetValue(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(10, 10,  ↩
    10, 10, 2, 2, 0, 0, 0),
                        1, '64BF', 0, 0
                    ),
                    1, 1, 1, -10
                ),
                1, 5, 4, 0
            ),
            1, 5, 5, 3.14159
        ) AS rast
    ) AS rast
    FULL JOIN (
        SELECT generate_series(1, 10) AS id
    ) AS id
        ON 1 = 1
```

```
)
SELECT
    (stats).count AS count,
    round((stats).sum::numeric, 3) AS sum,
    round((stats).mean::numeric, 3) AS mean,
    round((stats).stddev::numeric, 3) AS stddev,
    round((stats).min::numeric, 3) AS min,
    round((stats).max::numeric, 3) AS max
FROM (
    SELECT
        ST_SummaryStatsAgg(rast, 1, TRUE, 1) AS stats
    FROM foo
) bar;
```

Output

```
count |   sum   |  mean  | stddev |   min   |  max
-------+---------+--------+--------+---------+-------
    20 | -68.584 | -3.429 |  6.571 | -10.000 | 3.142
(1 row)
```

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);

Description

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

Example will count only pixels of band 1 that are not 249.

Code

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast, 249) WHERE rid=2;

SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;
```

Output

```
value | count
-------+-------
   250 |     2
   251 |     1
   252 |     2
   253 |     6
   254 |    12
```

Example will coount all pixels of band 1 including 249.

Code

```
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast, 1, false) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;
```

Output

```
value | count
-------+-------
   249 |     2
   250 |     2
   251 |     1
   252 |     2
   253 |     6
   254 |    12
```

Example will count only non-nodata value pixels of band 2.

Code

```
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast, 2) As pvc
    FROM dummy_rast WHERE rid=2) As foo
    ORDER BY (pvc).value;
```

Output

```
value | count
-------+-------
    78 |     1
    79 |     1
    88 |     1
    89 |     1
    96 |     1
    97 |     1
    98 |     1
    99 |     2
   112 |     2
:
```

Real live example. Count all the pixels in an aerial raster tile band 2 intersecting a geometry. And return only the pixel band values that have a count > 500.

Code

```
SELECT (pvc).value, SUM((pvc).count) As total
FROM (SELECT ST_ValueCount(rast, 2) As pvc
    FROM o_4_boston
        WHERE ST_Intersects(
            rast,
            ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706 ↩
    892151,224486 892151))', 26986)
             )
        ) As foo
    GROUP BY (pvc).value
    HAVING SUM((pvc).count) > 500
    ORDER BY (pvc).value;
```

Output

```
value | total
-------+-----
    51 | 502
    54 | 521
```

Just return count of pixels in each raster tile that have value of 100 of tiles that intersect a specific geometry.

Code

```
SELECT rid, ST_ValueCount(rast, 2, 100) As count
    FROM o_4_boston
        WHERE ST_Intersects(
            rast,
            ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706 ↩
    892151,224486 892151))', 26986)
             ) ;
```

Output

```
rid | count
-----+-------
   1 |    56
   2 |    95
  14 |    37
  15 |    64
```

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);

Description

☒定熟知的二☒制 (WKB) ☒格，返回一个☒格。

可用性：2.5.0

示例

Code

```
SELECT (ST_MetaData(ST_RastFromWKB('\001\000\000\000\000\000\000\000\000\000\000@ ↩
    \000\000\000\000\000\000\010@ ↩
    \000\000\000\000\000\000\340?\000\000\000\000\000\000\340?\000\000\000\000\000\000\000\000\0
    '::bytea
    )
)).* AS metadata;
```

Output

```
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands
------------+------------+-------+--------+--------+--------+-------+-------+------+----------
        0.5 |        0.5 |    10 |     20 |      2 |      3 |     0 |     0 |   10 |        0
```

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);

Description

⌧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

Code

```
SELECT (ST_MetaData(ST_RastFromHexWKB(' ↩
    010000000000000000000000400000000000000840000000000000 ↩
    E03F000000000000E03F0000000000000000000000000000000A0000000A001400'
    )
)).* AS metadata;
```

Output

```
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands
-----------+------------+-------+--------+--------+--------+-------+-------+------+----------
       0.5 |        0.5 |    10 |     20 |      2 |      3 |     0 |     0 |   10 |        0
```

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);

Description

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

Code

```
SELECT ST_AsBinary(rast) AS rastbin FROM dummy_rast WHERE rid=1;
```

Output

```
\001\000\000\000\000\000\000\000\000\000\000\000@\000\000\000\000\000\000\010@ ↩
    \000\000\000\000\000\000\340?\000\000\000\000\000\000\340?\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
```

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);

Description

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

Code

```
SELECT ST_AsHexWKB(rast) AS rastbin FROM dummy_rast WHERE rid=1;
```

Output

```
01000000000000000000000040000000000000084000000000000000 ↩
    E03F000000000000E03F0000000000000000000000000000000A0000000A001400
```

相关信息

ST_RastFromHexWKB, ST_AsBinary/ST_AsWKB

11.11.3 ST_AsGDALRaster

ST_AsGDALRaster — 以指定的 GDAL 栅格格式返回栅格栅栅。栅格格式是栅栅栅支持的格式之一。使用 ST_GDALDrivers() 栅取您的栅支持的格式列表。

Synopsis

bytea **ST_AsGDALRaster**(raster rast, text format, text[] options=NULL, integer srid=sameassource);

Description

返回指定格式的栅格栅栅。参数逐栅列栅如下：

- `format` 格式来栅出。栅取决于 libgdal 栅中栅栅的栅栅程序。通常可用的有“JPEG”、“GTIff”、“PNG”。使用 ST_GDALDrivers栅取您的栅支持的格式列表。
- `options` GDAL 栅栅的文本数栅。有效栅栅取决于格式。有关更多栅栅信息，栅参栅 GDAL 栅格格式栅栅。
- `srid` the spatial reference identifier to embed in the output raster

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

示例

Export multiple raster tiles as a single JPEG image with 50 percent quality.

Code

```
SELECT ST_AsGDALRaster(ST_Union(rast), 'JPEG', ARRAY['QUALITY=50']) As rastjpg
FROM dummy_rast
WHERE rast && ST_MakeEnvelope(10, 10, 11, 11);
```

One way to export raster into another format is using PostgreSQL large object export functions. The following repeats the JPEG example and exports the result. Note that this requires superuser access because it uses server-side large object functions. It will also export to path on server network. If you need export locally, use the psql equivalent lo_ functions which export to the local file system instead of the server file system.

Code

```
DROP TABLE IF EXISTS tmp_out ;

CREATE TABLE tmp_out AS
SELECT lo_from_bytea(
       0,
       ST_AsGDALRaster(ST_Union(rast), 'JPEG', ARRAY['QUALITY=50'])
        ) AS loid
  FROM dummy_rast
WHERE rast && ST_MakeEnvelope(10, 10, 11, 11);

SELECT lo_export(loid, '/tmp/dummy.jpg')
   FROM tmp_out;

SELECT lo_unlink(loid)
  FROM tmp_out;
```

Export a raster as a GeoTIFF using the default creation options.

Code

```
SELECT ST_AsGDALRaster(rast, 'GTiff') AS rasttiff
FROM dummy_rast WHERE rid = 2;
```

Export a GeoTIFF using JPEG compression at 90 percent quality and embed SRID 4269.

Code

```
SELECT ST_AsGDALRaster(
  rast,
  'GTiff',
  ARRAY['COMPRESS=JPEG', 'JPEG_QUALITY=90'],
  4269) AS rasttiff
FROM dummy_rast WHERE rid = 2;
```

相关信息

Section 10.4, 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);

Description

将栅格的指定波段作为单个多波段影像输出为像 (JPEG) 返回。如果需要输出不太常见的栅格类型，请使用 ST_AsGDALRaster。如果未指定波段且有 1 个或 3 个以上波段，则使用第一个波段。如果有 3 个波段，则使用所有 3 个波段。该功能有多个体和多个变体。以下逐个列出：

- `nband` 用于波段输出。
- `nbands` 是要输出的波段数组（请注意，JPEG 的最大值为 3），波段的顺序为 RGB。例如 ARRAY[3,2,1] 表示将波段 3 映射到红色，将波段 2 映射到绿色，将波段 1 映射到蓝色
- `quality` 数字从 0 到 100。数字越高，图像越清晰。
- `options` text 为 JPEG 定义的 GDAL 驱动数组（请查看 JPEG ST_GDALDrivers 的 create_options）。对于 JPEG，有效的是 `PROGRESSIVE` ON 或 OFF，`QUALITY` 的范围为 0 到 100，默认为 75。有关更多驱动信息，请参见GDAL 光栅格式 。

可用性：2.0.0 - 需要 GDAL >= 1.6.0。

示例：输出

Output the first three bands at 75 percent quality.

Code

```
SELECT ST_AsJPEG(rast) As rastjpg
FROM dummy_rast WHERE rid=2;
```

Output only the first band at 90 percent quality.

Code

```
SELECT ST_AsJPEG(rast, 1, 90) As rastjpg
FROM dummy_rast WHERE rid=2;
```

Output the first three bands, mapping band 2 to red, band 1 to green, and band 3 to blue, using progressive encoding at 90 percent quality.

Code

```
SELECT ST_AsJPEG(rast, ARRAY[2, 1, 3], ARRAY['QUALITY=90', 'PROGRESSIVE=ON']) As rastjpg
FROM dummy_rast WHERE rid=2;
```

相关信息

Section 10.4, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsTIFF

11.11.5 ST_AsPNG

ST_AsPNG — 将栅格带定的波段作为单个便携式网络图形 (PNG) 图像（字节数组）返回。如果栅格中有 1、3 或 4 个波段且未指定波段，则使用所有波段。如果有 2 个以上或多于 4 个波段且未指定波段，则使用波段 1。波段映射到 RGB 或 RGBA 空间。

Synopsis

bytea **ST_AsPNG**(raster rast, text[] options=NULL);
bytea **ST_AsPNG**(raster rast, integer nband, integer compression);
bytea **ST_AsPNG**(raster rast, integer nband, text[] options=NULL);
bytea **ST_AsPNG**(raster rast, integer[] nbands, integer compression);
bytea **ST_AsPNG**(raster rast, integer[] nbands, text[] options=NULL);

Description

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

Export the first 3 bands and map band 3 to red, band 1 to green, and band 2 to blue.

Code

```
SELECT ST_AsPNG(rast) As rastpng
FROM dummy_rast WHERE rid=2;
SELECT ST_AsPNG(rast, ARRAY[3, 1, 2]) As rastpng
FROM dummy_rast WHERE rid=2;
```

相关信息

ST_AsGDALRaster, ST_ColorMap, ST_GDALDrivers, Section 10.4

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);

Description

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
- `compressionCompression` 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%**

Code

```
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 — 返回由⊠入几何⊠形裁剪的⊠格。如果未指定波段号，⊠⊠理所有波段。如果未指定 `crop` 或其⊠⊠ TRUE，⊠⊠出⊠格将被裁剪。如果 `touched` ⊠置⊠ TRUE，⊠包括接触到的像素；否⊠，⊠当像素的中心位于几何⊠形内⊠，才会包括⊠像素。

Synopsis

raster **ST_Clip**(raster rast, integer[] nband, geometry geom, double precision[] nodataval=NULL, boolean crop=TRUE, boolean touched=FALSE);
raster **ST_Clip**(raster rast, integer nband, geometry geom, double precision nodataval, boolean crop=TRUE boolean touched=FALSE);
raster **ST_Clip**(raster rast, integer nband, geometry geom, boolean crop, boolean touched=FALSE);
raster **ST_Clip**(raster rast, geometry geom, double precision[] nodataval=NULL, boolean crop=TRUE, boolean touched=FALSE);
raster **ST_Clip**(raster rast, geometry geom, double precision nodataval, boolean crop=TRUE, boolean touched=FALSE);
raster **ST_Clip**(raster rast, geometry geom, boolean crop, boolean touched=FALSE);

Description

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

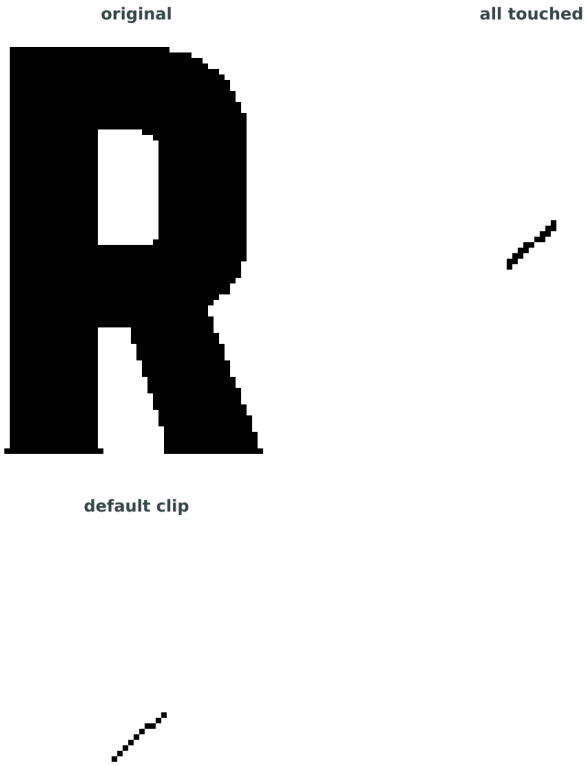
Comparing selecting all touched vs. not all touched.

Code

```
WITH source AS (
    SELECT rast, geom
    FROM ST_AsRaster(ST_Letters('R'), scalex =
> 1.0, scaley =
> -1.0) AS r(rast)
    INNER JOIN ST_GeomFromText('LINESTRING(0 1,5 6,10 10)') AS g(geom)
        ON ST_Intersects(r.rast, g.geom)
), clipped AS (
    SELECT rast,
        ST_Clip(rast, geom, touched =
> true) AS rast_touched,
        ST_Clip(rast, geom, touched =
> false) AS rast_not_touched
    FROM source
), variants AS (
    SELECT 'original' AS title, ST_Count(rast) AS pixel_count, rast AS rendered
    FROM clipped
    UNION ALL
    SELECT 'all touched', ST_Count(rast_touched), rast_touched
    FROM clipped
    UNION ALL
    SELECT 'default clip', ST_Count(rast_not_touched), rast_not_touched
    FROM clipped
)
SELECT title, pixel_count, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY CASE title
    WHEN 'original' THEN 1
```

```
    WHEN 'all touched' THEN 2
    ELSE 3
END;
```

Figure



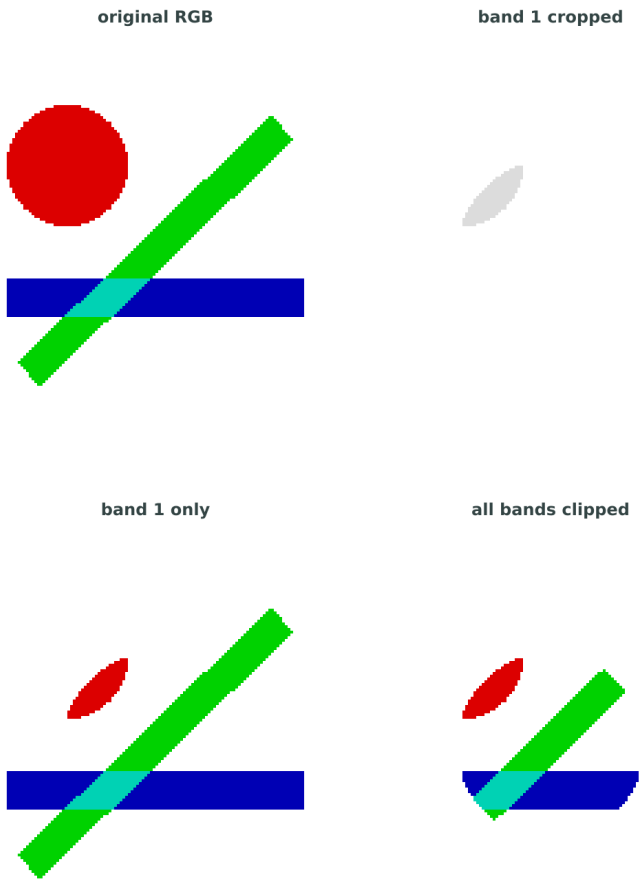
The following self-contained example uses three simple shapes as RGB bands. It compares cropping band 1, clipping band 1 without cropping and then restoring bands 2 and 3, and clipping every band without cropping.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_AddBand(
            ST_AddBand(
                ST_MakeEmptyRaster(128, 128, 0, 128, 1, -1, 0, 0, 0),
                '8BUI'::text, 0::double precision, 0::double precision
            ),
            '8BUI'::text, 0::double precision, 0::double precision
        ),
        '8BUI'::text, 0::double precision, 0::double precision
    ) AS rast
), source AS (
    SELECT ST_SetValue(
```

```
        ST_SetValue(
            ST_SetValue(rast, 1, ST_Buffer('POINT(32 96)'::geometry, 22), 220),
            2, ST_Buffer('LINESTRING(18 20,110 110)'::geometry, 6,
                         'endcap=flat join=bevel'), 210
        ),
        3, ST_Buffer('LINESTRING(10 48,118 48)'::geometry, 7,
                     'endcap=flat'), 180
    ) AS rast,
    ST_Buffer('POINT(64 64)'::geometry, 34) AS clipper
    FROM canvas
), variants AS (
    SELECT 'original RGB' AS title, rast AS rendered FROM source
    UNION ALL
    SELECT 'band 1 cropped', ST_Clip(rast, 1, clipper, true) FROM source
    UNION ALL
    SELECT
        'band 1 only',
        ST_AddBand(
            ST_Clip(rast, 1, clipper, false),
            ARRAY[ST_Band(rast, 2), ST_Band(rast, 3)]
        )
    FROM source
    UNION ALL
    SELECT 'all bands clipped', ST_Clip(rast, clipper, false) FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY CASE title
    WHEN 'original RGB' THEN 1
    WHEN 'band 1 cropped' THEN 2
    WHEN 'band 1 only' THEN 3
    ELSE 4
END;
```

Figure



Crop, single-band clipping, and all-band clipping.



Source tile



Cropped



One-band source



One-band clip



All-band source



All-band clip

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);

Description

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。

Code

```
5 0 0 0 255
4 100:50 55 255
1 150, 100 150 255
0% 255 255 255 255
nv 0 0 0 0
```

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

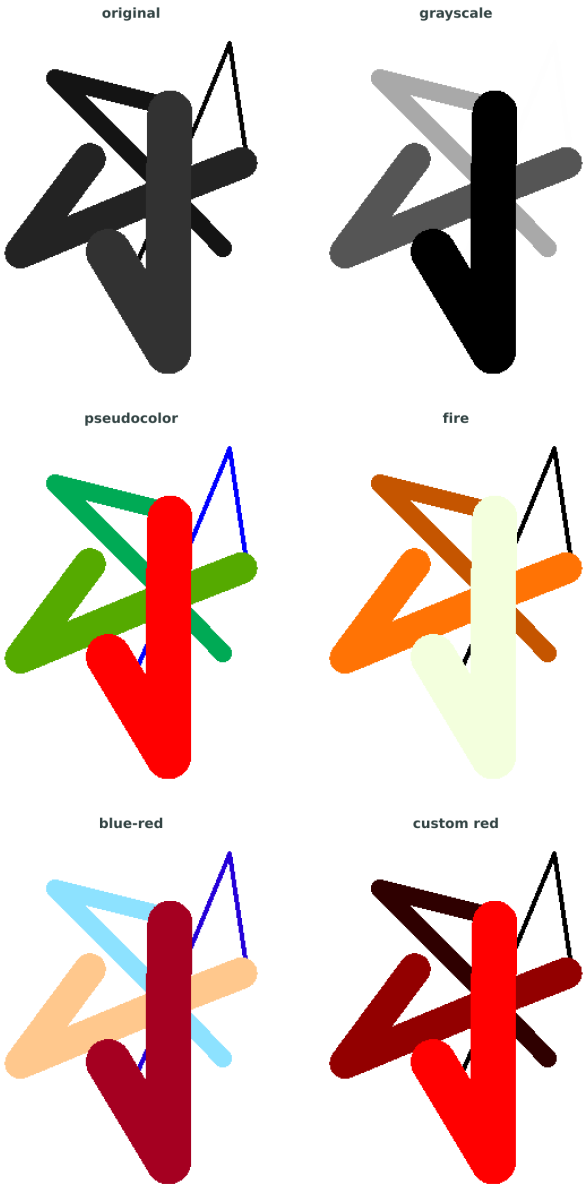
示例

This example builds a raster from rotated buffered lines and compares the original values with predefined and custom color maps.

Code

```
WITH ref AS (
    SELECT ST_MakeEmptyRaster(200, 200, 0, 200, 1, -1, 0, 0) AS rast
), source AS (
    SELECT ST_Union(rast) AS rast
    FROM (
    SELECT
        ST_AsRaster(ST_Rotate(ST_Buffer('LINESTRING(0 2,50 50,150 150,125 50)'::geometry,
                    i*2
                ),
                pi() * i * 0.125, ST_Point(50, 50)
            ),
            ref.rast, '8BUI'::text, i * 5
        ) AS rast
    FROM ref
    CROSS JOIN generate_series(1, 10, 3) AS i
) AS shapes
)
SELECT title, ST_AsPNG(rendered) AS image
FROM source
CROSS JOIN LATERAL (VALUES
    ('original', rast),
    ('grayscale', ST_ColorMap(rast, 1, 'greyscale')),
    ('pseudocolor', ST_ColorMap(rast, 1, 'pseudocolor')),
    ('fire', ST_ColorMap(rast, 1, 'fire')),
    ('blue-red', ST_ColorMap(rast, 1, 'bluered')),
    ('custom red', ST_ColorMap(
        rast,
        1,
        E'100% 255 0 0 255\n80% 160 0 0 255\n50% 130 0 0 255\n' ||
        E'30% 30 0 0 255\n20% 60 0 0 255\n0% 0 0 0 255\n' ||
        E'nv 255 255 255 0'
    ))
) AS variants(title, rendered);
```

Figure



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);

Description

⍰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

This self-contained example paints an apple with gradients and highlights into three raster bands. It compares the original RGB raster with the equivalent grayscale results produced by both function variants.

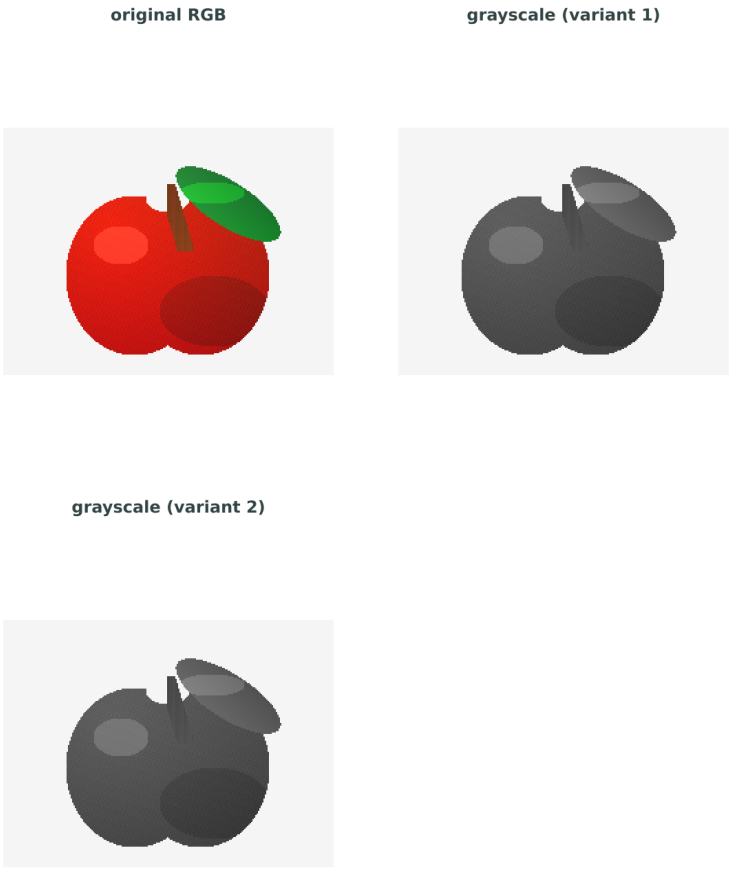
Code

```
WITH pixels AS (
    SELECT
        col,
        row,
        x,
        y,
        (((x - 48.0)^2 / (25.0^2) + (y - 54.0)^2 / (29.0^2) < 1)
         OR ((x - 72.0)^2 / (25.0^2) + (y - 54.0)^2 / (29.0^2) < 1))
         AND NOT ((x - 60.0)^2 / (8.0^2) + (y - 26.0)^2 / (5.0^2) < 1) AS fruit,
        ((x - 82.0) * 0.84 + (y - 28.0) * 0.54)^2 / (22.0^2)
          + (-(x - 82.0) * 0.54 + (y - 28.0) * 0.84)^2 / (8.0^2) < 1 AS leaf,
        abs((x - 62.0) * 0.96 - (y - 29.0) * 0.28) < 3
          AND x BETWEEN 60 AND 69 AND y BETWEEN 21 AND 45 AS stem
    FROM (
        SELECT
            col,
            row,
            col / 2.0 AS x,
            row / 2.0 AS y
        FROM generate_series(1, 180) AS row
        CROSS JOIN generate_series(1, 240) AS col
    ) AS source_pixels
), shaded AS (
    SELECT
        col,
        row,
        x,
        y,
        CASE
            WHEN stem THEN 95 + y * 0.7 + (x % 3) * 8
```

```
            WHEN leaf THEN 28 + (90 - x) * 0.5
            WHEN fruit THEN LEAST(255, GREATEST(0,
                150
                + (85 - sqrt((x - 39.0)^2 + (y - 37.0)^2)) * 1.1
                + CASE
                    WHEN (x - 43.0)^2 / (10.0^2) + (y - 43.0)^2 / (7.0^2) < 1
                    THEN 35
                    ELSE 0
                END
                - CASE
                    WHEN (x - 77.0)^2 / (20.0^2) + (y - 67.0)^2 / (13.0^2) < 1
                    THEN 35
                    ELSE 0
                END
                + ((x * 13 + y * 7) % 19 - 9)
            ))
            ELSE 245
        END AS red,
        CASE
            WHEN stem THEN 50 + y * 0.35
            WHEN leaf THEN LEAST(190, GREATEST(40,
                105 + (95 - x) * 1.2 + (y - 20) * 1.5
                + CASE
                    WHEN (x - 76.0)^2 / (12.0^2) + (y - 24.0)^2 / (4.0^2) < 1
                    THEN 45
                    ELSE 0
                END
            ))
            WHEN fruit THEN LEAST(95, GREATEST(15,
                24 + (68 - y) * 0.35
                + CASE
                    WHEN (x - 43.0)^2 / (10.0^2) + (y - 43.0)^2 / (7.0^2) < 1
                    THEN 30
                    ELSE 0
                END
                + ((x * 5 + y * 11) % 13 - 6)
            ))
            ELSE 245
        END AS green,
        CASE
            WHEN stem THEN 22 + (x % 4) * 3
            WHEN leaf THEN 42 + (100 - x) * 0.4
            WHEN fruit THEN LEAST(75, GREATEST(10,
                18
                + CASE
                    WHEN (x - 43.0)^2 / (10.0^2) + (y - 43.0)^2 / (7.0^2) < 1
                    THEN 25
                    ELSE 0
                END
                + ((x * 3 + y * 5) % 11 - 5)
            ))
            ELSE 245
        END AS blue
    FROM pixels
), rows AS (
    SELECT
        row,
        array_agg(red::double precision ORDER BY col) AS red_values,
        array_agg(green::double precision ORDER BY col) AS green_values,
        array_agg(blue::double precision ORDER BY col) AS blue_values
    FROM shaded
    GROUP BY row
```

```
), source AS (
    SELECT ST_SetValues(ST_SetValues(ST_SetValues(
        ST_AddBand(ST_AddBand(ST_AddBand(
            ST_MakeEmptyRaster(240, 180, 0, 180, 1, -1, 0, 0, 0),
            '8BUI'::text, 245::double precision, 0::double precision),
            '8BUI'::text, 245::double precision, 0::double precision),
            '8BUI'::text, 245::double precision, 0::double precision),
        1, 1, 1, array_agg(red_values ORDER BY row)::double precision[][]),
        2, 1, 1, array_agg(green_values ORDER BY row)::double precision[][]),
        3, 1, 1, array_agg(blue_values ORDER BY row)::double precision[][]) AS rast
    FROM rows
)
SELECT ST_AsPNG(rast) AS "original RGB",
    ST_AsPNG(ST_Grayscale(rast)) AS "grayscale (variant 1)",
    ST_AsPNG(ST_Grayscale(ARRAY[
        ROW(rast, 1)::rastbandarg,
        ROW(rast, 2)::rastbandarg,
        ROW(rast, 3)::rastbandarg
    ]::rastbandarg[])) AS "grayscale (variant 2)"
FROM source;
```

Figure



Original photographic illustration:



original apple



grayscale apple

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);

Description

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

其他栅体返回光栅，在光栅空栅中工作。他栅使用 ST_MapAlgebraExpr 的栅个栅格版本来栅行交集。

生成的栅格范围由于栅格范围的几何交集。生成的栅格包含“BAND1”、“BAND2”或“BOTH”波段，遵循作为 `returnband` 参数值的内容。任何波段中存在的无数据值区域都会导致结果的每个波段中出现无数据值区域。换句话说，与无数据值像素相交的任何像素都成为结果中的无数据值像素。

由 ST_Intersection 生成的栅格必须对不相交的区域分配 nodata 值。您可以通过提供一个包含一个或两个 nodata 值的 `nodataval[]` 数组来定义或替换任何结果波段的 nodata 值，具体取决于您是否要求“BAND1”、“BAND2”或“BOTH”波段。数组中的第一个值替换第一个栅中的无数据值，第二个值替换第二个栅中的无数据值。如果一个输入没有定义 nodata 值并且没有以数组形式提供，则使用 ST_MinPossibleValue 函数值为一个。所有接受 nodata 值数组的栅体也可以接受两个值，则将分配每个波求的值。

在所有栅体中，如果未指定波段号，则假定波段 1。如果您需要栅格和返回栅格的几何形之值的交集，请参见 ST_Clip。



Note

For more control over the result extent or NODATA handling, use the two-raster variant of ST_MapAlgebraExpr. To compute a raster-space intersection with a geometry, use ST_Clip. Use this function together with ST_Intersects and an index on the raster or geometry column when filtering table data.

增强：2.0.0 - 引入了光栅空间中的交集。在 2.0.0 之前的早期版本中，仅支持在向量空间中执行交集。

示例：几何、光栅——派生几何值

Code

```
SELECT
    foo.rid,
    foo.gid,
    GeometryType((foo.geomval).geom) AS geom_type,
    (foo.geomval).val AS value
FROM (
    SELECT
        A.rid,
        g.gid,
        ST_Intersection(A.rast, g.geom) As geomval
    FROM dummy_rast AS A
    CROSS JOIN (
        VALUES
            (1, ST_Point(3427928, 5793243.85) ),
            (2, 'LINESTRING(3427927.85 5793243.75,3427927.8 5793243.75,3427927.8 5793243.8) ↩
    '::geometry),
            (3, 'LINESTRING(1 2,3 4)'::geometry)
    ) As g(gid, geom)
    WHERE A.rid = 2
) AS foo;
```

Output

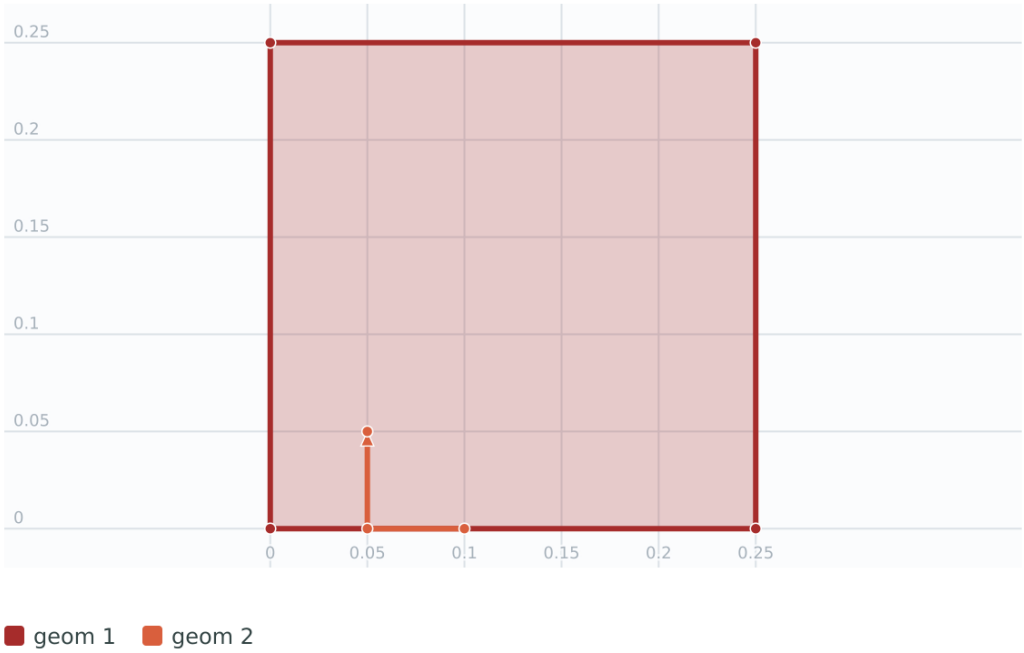
```
rid | gid |      geom_type       | value
-----+-----+----------------------+-------
   2 |   1 | POINT                |   249
   2 |   1 | POINT                |   253
   2 |   2 | POINT                |   254
   2 |   2 | POINT                |   251
   2 |   2 | POINT                |   253
   2 |   2 | LINESTRING           |   252
   2 |   2 | MULTILINESTRING      |   250
   2 |   3 | GEOMETRYCOLLECTION   |
```

The same line example rendered in a common coordinate frame. The raster footprint and complete input line remain visible behind the returned intersection pieces.

Code

```
WITH source AS (
    SELECT
        rast,
        ST_GeomFromText(
            'LINESTRING(3427927.85 5793243.75,'
            '3427927.8 5793243.75,3427927.8 5793243.8)'
        ) AS input_geometry
    FROM dummy_rast
    WHERE rid = 2
), intersection AS (
SELECT
    rast,
    input_geometry,
    ST_Collect(gv.geom) AS result_geometry
FROM source
CROSS JOIN LATERAL ST_Intersection(rast, input_geometry) AS gv
GROUP BY rast, input_geometry
), layers AS (
    SELECT
        'raster' AS title,
        ST_Translate(ST_Envelope(rast), -3427927.75, -5793243.75) AS geom
    FROM intersection
    UNION ALL
    SELECT
        'input',
        ST_Translate(input_geometry, -3427927.75, -5793243.75)
    FROM intersection
    UNION ALL
    SELECT
        'intersection',
        ST_Translate(result_geometry, -3427927.75, -5793243.75)
    FROM intersection
)
SELECT title, geom AS geom
FROM layers
ORDER BY CASE title
    WHEN 'raster' THEN 1
    WHEN 'input' THEN 2
    ELSE 3
END;
```

Figure



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);

Description

⊠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：

Code

```
CREATE OR REPLACE FUNCTION sample_callbackfunc(value double precision[][][], pos ↩
    integer[][], VARIADIC userargs text[])
    RETURNS double precision
    AS $$
    BEGIN
        RETURN 0;
    END;
    $$ LANGUAGE 'plpgsql' IMMUTABLE;
```

The `callbackfunc` must have two or three arguments: a 3-dimension double precision array, a 2-dimension integer array and, optionally, a variadic 1-dimension text array. The first argument `value` is the set of values (as double precision) from all input rasters. The three dimensions (where indexes are 1-based) are: raster #, row y, column x. The second argument `position` is the set of pixel positions from the output raster and input rasters. The outer dimension (where indexes are 0-based) is the raster #. The position at outer dimension index 0 is the output raster's pixel position. For each outer dimension, there are two elements in the inner dimension for X and Y. The optional third argument `userargs` is for passing through any user-specified arguments.

The callback return value is used as the value of the output pixel. If the callback returns `NULL`, the output pixel is set to NODATA when the output band has a NODATA value. If the output band has no NODATA value, a `NULL` callback result raises an error.

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 ⊠：

Code

```
'sample_callbackfunc(double precision[], integer[], text[])'::regprocedure
```

Note that PostgreSQL array types do not encode dimensions in function signatures, so multidimensional callback arguments are written as double precision[] and integer[] in the regprocedure signature.

mask 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 If the callbackfunc declares a third argument, that argument is a variadic text array. All trailing text arguments are passed through to the specified callbackfunc, and are contained in the userargs argument.



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。

⍰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

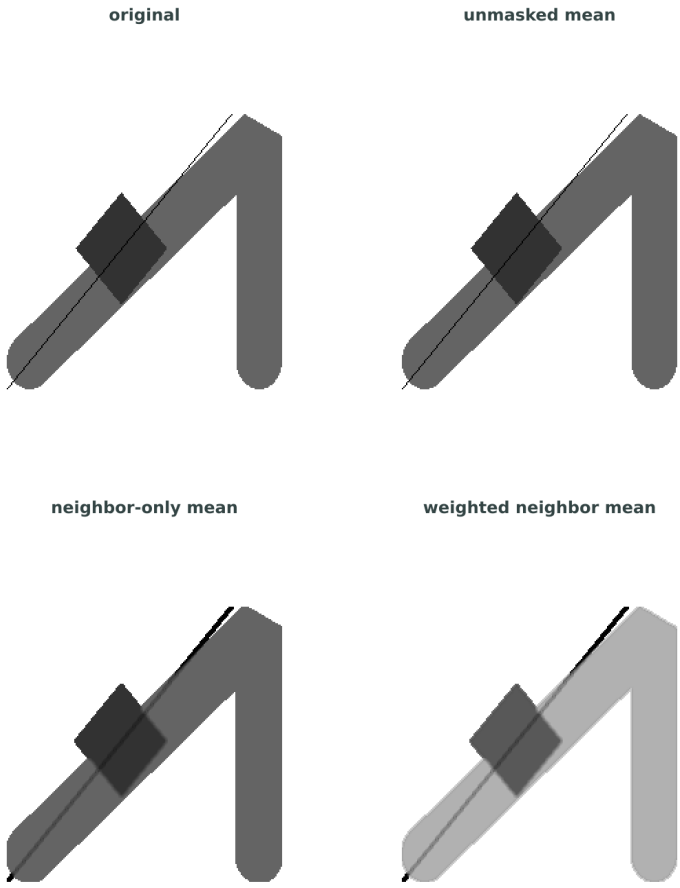
Compare an unmasked mean with neighbor-only and weighted-neighbor masks. The weighted mask multiplies neighboring values by 2 before computing the result.

Code

```
WITH foo AS (
    SELECT ST_SetBandNoDataValue(
        ST_SetValue(
            ST_SetValue(
                ST_AsRaster(
                    ST_Buffer('LINESTRING(50 50,100 90,100 50)'::geometry, 5, 'join=bevel') ↩
    ,
                    200, 200, ARRAY['8BUI'], ARRAY[100], ARRAY[0]
                ),
                ST_Buffer('POINT(70 70)'::geometry, 10, 'quad_segs=1'),
                50
            ),
            'LINESTRING(20 20,100 100,150 98)'::geometry,
            1
        ),
        0
    ) AS rast
), variants AS (
    SELECT 'original' AS title, rast
    FROM foo
    UNION ALL
    SELECT 'unmasked mean' AS title,
        ST_MapAlgebra(rast, 1, 'ST_Mean4ma(double precision[], int[], text[])':: ↩
    regprocedure) AS rast
    FROM foo
    UNION ALL
```

```
    SELECT 'neighbor-only mean' AS title,
        ST_MapAlgebra(
            rast,
            1,
            'ST_Mean4ma(double precision[], int[], text[])'::regprocedure,
            '{{1,1,1}, {1,0,1}, {1,1,1}}'::double precision[],
            false
        ) AS rast
    FROM foo
    UNION ALL
    SELECT 'weighted neighbor mean' AS title,
        ST_MapAlgebra(
            rast,
            1,
            'ST_Mean4ma(double precision[], int[], text[])'::regprocedure,
            '{{2,2,2}, {2,0,2}, {2,2,2}}'::double precision[],
            true
        ) AS rast
    FROM foo
)
SELECT title, ST_AsPNG(rast) AS image
FROM variants;
```

Figure



Variant 1.

一个⊠格，一个波段

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16BUI', ↩
     1, 0) AS rast
)
SELECT
    ST_MapAlgebra(ARRAY[ROW(rast, 1)]::rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

一个⊠格，多个波段

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast
)
SELECT
    ST_MapAlgebra(ARRAY[ROW(rast, 3), ROW(rast, 1), ROW(rast, 3), ROW(rast, 2)]:: ↩
    rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

多个⊠格，多个波段

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast UNION ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 2, 0), 2, '8BUI', 20, 0), 3, '32BUI', 300, 0) AS rast
)
SELECT
    ST_MapAlgebra(ARRAY[ROW(t1.rast, 3), ROW(t2.rast, 1), ROW(t2.rast, 3), ROW(t1.rast, 2) ↩
    ]::rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 1
    AND t2.rid = 2
```

完整示例：⊠⊠域的覆盖示例。

Code

```
WITH foo AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16BUI', ↩
     1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, 0, 1, -1, 0, 0, 0), 1, '16BUI', 2, 0)  ↩
    AS rast UNION ALL
```

```
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, 0, 1, -1, 0, 0, 0), 1, '16BUI', 3, 0)
    AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -2, 1, -1, 0, 0, 0), 1, '16BUI', 10,
    0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -2, 1, -1, 0, 0, 0), 1, '16BUI', 20,
    0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -2, 1, -1, 0, 0, 0), 1, '16BUI', 30,
    0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -4, 1, -1, 0, 0, 0), 1, '16BUI', 100,
    0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -4, 1, -1, 0, 0, 0), 1, '16BUI', 200,
    0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -4, 1, -1, 0, 0, 0), 1, '16BUI', 300,
    0) AS rast
)
SELECT
    t1.rid,
    ST_MapAlgebra(ARRAY[ROW(ST_Union(t2.rast), 1)]::rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure,
        '32BUI',
        'CUSTOM', t1.rast,
        1, 1
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 4
    AND t2.rid BETWEEN 0 AND 8
    AND ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rid, t1.rast
```

示例：与前一个⌧⌧域的覆盖 tile 示例⌧似，但只使用基⌧ SQL 特性来⌧⌧。

Code

```
WITH src AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16BUI',
     1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, 0, 1, -1, 0, 0, 0), 1, '16BUI', 2, 0)
    AS rast UNION ALL
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, 0, 1, -1, 0, 0, 0), 1, '16BUI', 3, 0)
    AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -2, 1, -1, 0, 0, 0), 1, '16BUI', 10,
    0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -2, 1, -1, 0, 0, 0), 1, '16BUI', 20,
    0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -2, 1, -1, 0, 0, 0), 1, '16BUI', 30,
    0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -4, 1, -1, 0, 0, 0), 1, '16BUI', 100,
    0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -4, 1, -1, 0, 0, 0), 1, '16BUI', 200,
    0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -4, 1, -1, 0, 0, 0), 1, '16BUI', 300,
    0) AS rast
), foo AS (
    SELECT
        t1.rid,
        ST_Union(t2.rast) AS rast
```

```
    FROM src t1
    JOIN src t2
        ON ST_Intersects(t1.rast, t2.rast)
        AND t2.rid BETWEEN 0 AND 8
    WHERE t1.rid = 4
    GROUP BY t1.rid
), bar AS (
    SELECT
        t1.rid,
        ST_MapAlgebra(
            ARRAY[ROW(t2.rast, 1)]::rastbandarg[],
            'sample_callbackfunc(double precision[], int[], text[])'::regprocedure,
            '32BUI',
            'CUSTOM', t1.rast,
            1, 1
        ) AS rast
    FROM src t1
    JOIN foo t2
        ON t1.rid = t2.rid
)
SELECT
    rid,
    (ST_MetaData(rast)),
    (ST_BandMetaData(rast, 1)),
    ST_Value(rast, 1, 1, 1)
FROM bar;
```

Variants 2 and 3.

一个区格，多个波段

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        rast,
        ARRAY[3, 1, 3, 2]::integer[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

一个区格，一个波段

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        rast,
        2,
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo
```

Variant 4.

⍰个⍰格，⍰个波段

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI', 100, 0) AS rast UNION ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 2, 0), 2, '8BUI', 20, 0), 3, '32BUI', 300, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        t1.rast,
        2,
        t2.rast,
        1,
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 1
    AND t2.rid = 2
```

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);

Description

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：栅格 **1** 和 **2**（一个栅格）

通过栅格波段 (rast) 上的 expression 定义的有效 PostgreSQL 代数语法来创建新的栅格。如果未提供 nband，则假定波段 1。新栅格将具有与原始栅格相同的地理参考、宽度和高度，但只有一个波段。

如果输入**像素类型**，则新栅格将具有像素类型的波段。如果将像素类型输入为 NULL，则新的栅格将具有与输入**栅格**相同的像素类型。

- 允许 expression 的关键字
  1. [rast] - 感兴趣像素的像素值
  2. [rast.val]- 感兴趣像素的像素值
  3. [rast.x] - 感兴趣像素的基于 1 的像素列
  4. [rast.y] - 感兴趣像素的从 1 开始的像素行

描述：栅格 **3** 和 **4**（两个栅格）

通过两个输入栅格波段 rast1, (rast2) 上的 expression 定义的两个波段用有效的 PostgreSQL 代数语法，创建一个新的栅格波段栅格。如果没有 band1，则指定 band2，则假定为 band 1。生成的栅格将在第一个栅格定义的网格上对齐（比例、倾斜和像素角）。生成的栅格将具有由 extenttype 参数定义的范围。

expression 涉及两个栅格的 PostgreSQL 代数表达式和 PostgreSQL 定义的函数/运算符，用于定义像素相交区域的像素值。例如 (([rast1] [rast2])/2.0)::integer

pixeltype 输出栅格的结果像素类型。必须是ST_BandPixelType 中列出的一项，可省略或设置为 NULL。如果未输入或设置为 NULL，将默认为第一个栅格的像素类型。

extenttype 控制生成的栅格的范围

1. INTERSECTION - 新栅格的范围是两个栅格的交集。这是默认设置。
2. UNION - 新栅格的范围是两个栅格的并集。
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 用于涉及 rast2 的代数表达式或定义当 rast1 的像素值无数据值并且空间对齐的 rast2 像素具有值返回的内容的常量。

nodata2expr 用于涉及 rast1 常量的代数表达式，该常量定义当 rast2 的像素值无数据值且空间对齐的 rast1 像素具有值返回的内容。

nodatanodataval 当空间对齐的 rast1 和 rast2 像素均为无数据值时要返回的数值常量。

- expression、nodata1expr 和 nodata2expr 中允许使用的关键字
  1. [rast1] - rast1 中感兴趣的像素的像素值
  2. [rast1.val] - rast1 中感兴趣像素的像素值
  3. [rast1.x] - 来自 rast1 的感兴趣像素的基于 1 的像素列
  4. [rast1.y] - 来自 rast1 的感兴趣像素的基于 1 的像素行
  5. [rast2] - rast2 中感兴趣像素的像素值
  6. [rast2.val] - rast2 中感兴趣像素的像素值
  7. [rast2.x] - 来自 rast2 的感兴趣像素的基于 1 的像素列
  8. [rast2.y] - 来自 rast2 的感兴趣像素的基于 1 的像素行

示例

Variants 1 and 2.

Code

```
WITH foo AS (
    SELECT ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, 1, 0, 0, 0), '32BF'::text, 1, -1) ↩
     AS rast
)
SELECT
    ST_MapAlgebra(rast, 1, NULL, 'ceil([rast]*[rast.x]/[rast.y]+[rast.val])')
FROM foo;
```

Variant 3 and 4.

Code

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI'::text, 100, 0) AS rast UNION ↩
     ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 1, -1, ↩
     0, 0, 0), 1, '16BUI', 2, 0), 2, '8BUI', 20, 0), 3, '32BUI'::text, 300, 0) AS rast
)
SELECT
    ST_MapAlgebra(
        t1.rast,
        2,
        t2.rast,
        1,
        '([rast2] + [rast1.val]) / 2'
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 1
    AND t2.rid = 2;
```

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)

Description



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。

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
UPDATE dummy_rast SET map_rast = ST_MapAlgebraExpr(rast, NULL, 'mod([rast]::numeric,2)') ↩
    WHERE rid = 2;

SELECT
    ST_Value(rast, 1, i, j) As origval,
    ST_Value(map_rast, 1, i, j) As mapval
FROM dummy_rast
CROSS JOIN generate_series(1, 3) AS i
CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
origval | mapval
---------+--------
     253 |      1
     254 |      0
     253 |      1
     253 |      1
     254 |      0
     254 |      0
     250 |      0
     254 |      0
     254 |      0
```

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。

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast2 raster;
UPDATE dummy_rast SET
    map_rast2 = ST_MapAlgebraExpr(rast, '2BUI'::text, 'CASE WHEN [rast] BETWEEN 100 and 250 ↩
     THEN 1 WHEN [rast] = 252 THEN 2 WHEN [rast] BETWEEN 253 and 254 THEN 3 ELSE 0 END':: ↩
    text, '0')
WHERE rid = 2;

SELECT DISTINCT
```

```
    ST_Value(rast, 1, i, j) As origval,
    ST_Value(map_rast2, 1, i, j) As mapval
FROM dummy_rast
CROSS JOIN generate_series(1, 5) AS i
CROSS JOIN generate_series(1, 5) AS j
WHERE rid = 2;
```

Output

```
origval | mapval
---------+--------
     249 |      1
     250 |      1
     251 |
     252 |      2
     253 |      3
     254 |      3
```

Code

```
SELECT
    ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast
WHERE rid = 2;
```

Output

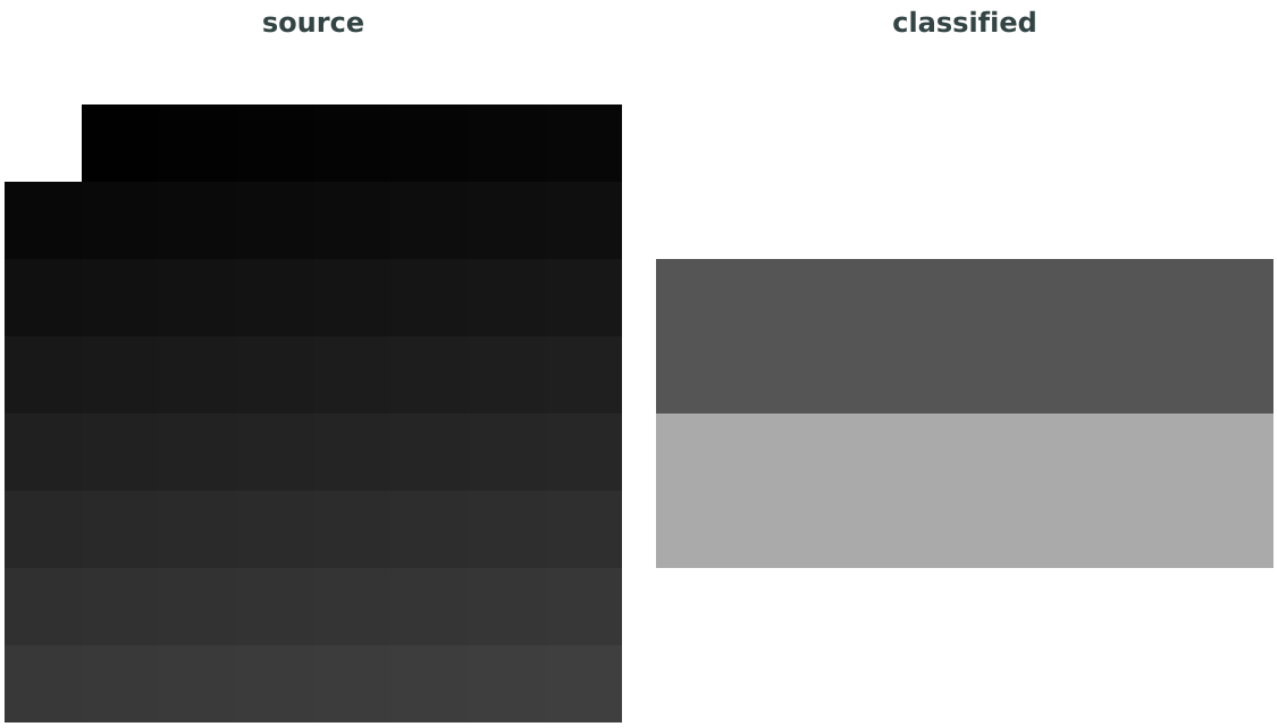
```
2BUI
```

A self-contained source raster and the raster produced by classifying its values into four display levels.

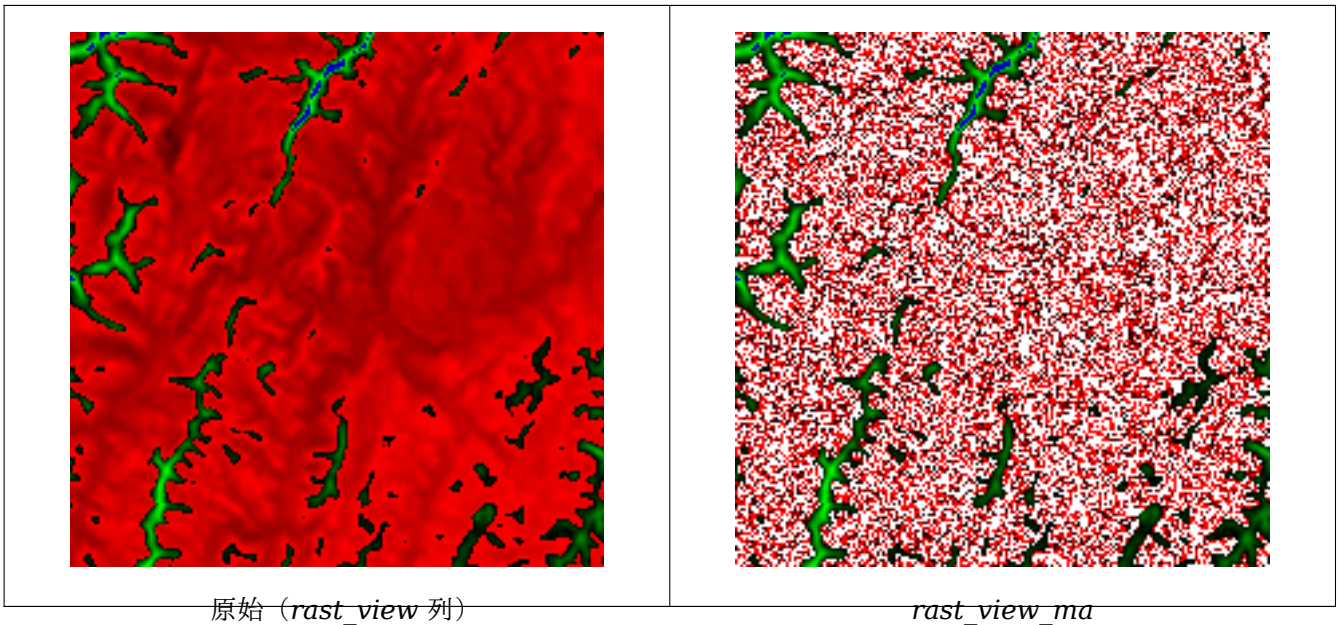
Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(8, 8, 0, 8, 1, -1, 0, 0, 0),
            1, '8BUI', 0, 0
        ),
        1, 1, 1,
        ARRAY[
            [0, 1, 2, 3, 4, 5, 6, 7],
            [8, 9, 10, 11, 12, 13, 14, 15],
            [16, 17, 18, 19, 20, 21, 22, 23],
            [24, 25, 26, 27, 28, 29, 30, 31],
            [32, 33, 34, 35, 36, 37, 38, 39],
            [40, 41, 42, 43, 44, 45, 46, 47],
            [48, 49, 50, 51, 52, 53, 54, 55],
            [56, 57, 58, 59, 60, 61, 62, 63]
        ]::double precision[][]
    ) AS rast
), variants AS (
    SELECT 'source' AS title, rast AS rendered
    FROM source
    UNION ALL
    SELECT
        'classified',
        ST_MapAlgebraExpr(rast, '8BUI', 'floor([rast] / 16) * 85')
    FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Figure



Map algebra altering the first band of a three-band raster.



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⊠。

Code

```
SELECT
    ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(rast_view),
                ST_MapAlgebraExpr(rast_view, 1, NULL, 'tan([rast])*[rast]')
            ),
            ST_Band(rast_view, 2)
        ),
        ST_Band(rast_view, 3)
    )  As rast_view_ma
FROM wind
WHERE rid=167;
```

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);

Description



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☒格的范☒

1. INTERSECTION - 新栅格的范围是两个栅格的交集。这是默认设置。
2. UNION - 新栅格的范围是两个栅格的并集。
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 这涉及 rast2 的代数表达式或定义当 rast1 的像素值无数据值并且空间位置的 rast2 像素具有值返回的内容的常量。

nodata2expr 这涉及 rast1 常量的代数表达式，该常量定义当 rast2 的像素值无数据值且空间位置的 rast1 像素具有值返回的内容。

nodatanodataval 当空间位置的 rast1 和 rast2 像素均值无数据值值要返回的数值常量。

如果输入 pixeltype，则新栅格将具有该像素类型的波段。如果将像素类型输入 NULL 或未指定像素类型，则新的栅格波段将具有与输入 rast1 波段相同的像素类型。

使用输入 [rast1.val] [rast2.val] 指代原始栅格波段的像素值，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 Band Intersection and Union.

根据原始栅格值建一个新的 1 波段栅格，其栅格是原始栅格波段的模 2 的函数。

Create a cool set of rasters.

Insert some cool shapes around Boston in Massachusetts state plane meters, then map their intersection and union.

Code

```
DROP TABLE IF EXISTS fun_shapes;
CREATE TABLE fun_shapes(rid serial PRIMARY KEY, fun_name text, rast raster);

INSERT INTO fun_shapes(fun_name, rast)
VALUES ('ref', ST_AsRaster(ST_MakeEnvelope(235229, 899970, 237229, 901930, 26986), 200,  ↩
    200, '8BUI', 0, 0));

INSERT INTO fun_shapes(fun_name, rast)
WITH ref(rast) AS (SELECT rast FROM fun_shapes WHERE fun_name = 'ref' )
SELECT 'area' AS fun_name, ST_AsRaster(ST_Buffer(ST_SetSRID(ST_Point(236229, 900930),  ↩
    26986), 1000),
            ref.rast, '8BUI', 10, 0) As rast
FROM ref
UNION ALL
SELECT 'rand bubbles',
       ST_AsRaster(
           (
               SELECT ST_Collect(geom)
               FROM (
                   SELECT ST_Buffer(
                       ST_SetSRID(ST_Point(236229 + i*random()*100, 900930 + j*random() ↩
    *100), 26986),
                       random()*20
                   ) AS geom
                   FROM generate_series(1, 10) AS i,
                        generate_series(1, 10) AS j
               ) AS foo
```

```
            ),
            ref.rast, '8BUI', 200, 0
        )
FROM ref;

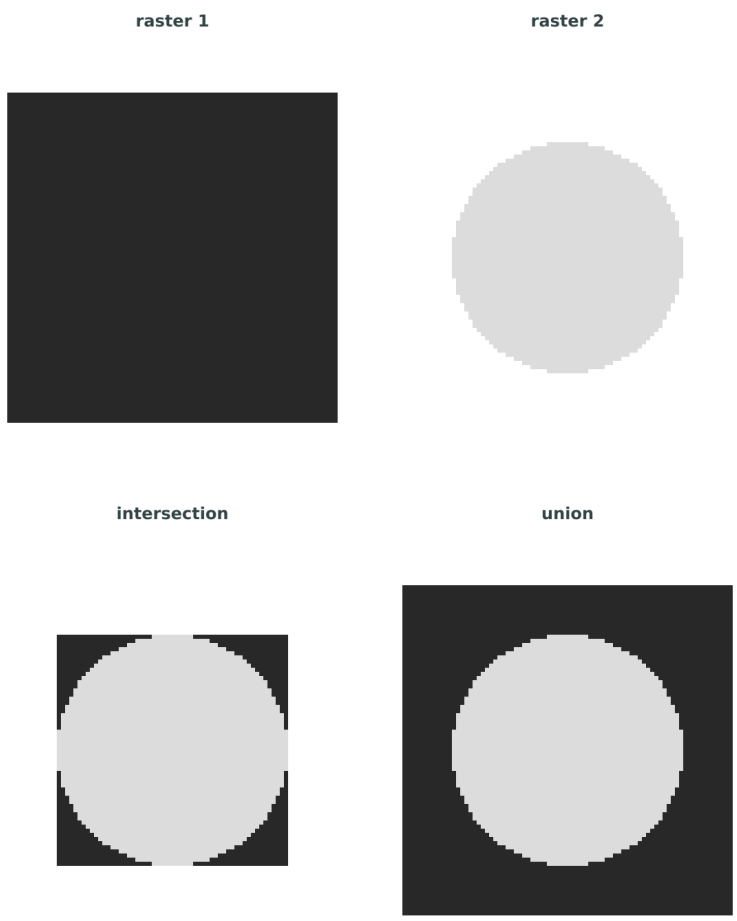
SELECT  ST_MapAlgebraExpr(area.rast, bub.rast, '[rast2.val]', '8BUI', 'INTERSECTION', '[ ↩
    rast2.val]', '[rast1.val]') As interrast,
        ST_MapAlgebraExpr(area.rast, bub.rast, '[rast2.val]', '8BUI', 'UNION', '[rast2.val] ↩
    ', '[rast1.val]') As unionrast
FROM
  (SELECT rast FROM fun_shapes WHERE fun_name = 'area') As area
CROSS JOIN
  (SELECT rast FROM fun_shapes WHERE fun_name = 'rand bubbles') As bub
```

The same intersection and union operations using two deterministic, self-contained rasters.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(80, 80, 0, 80, 1, -1, 0, 0, 0),
        1, '8BUI', 40, 0
    ) AS rast
), inputs AS (
    SELECT
        rast AS raster_1,
        ST_AsRaster(
            ST_Buffer(ST_Point(40, 40), 28),
            rast,
            '8BUI', 220, 0
        ) AS raster_2
    FROM canvas
), variants AS (
    SELECT 'raster 1' AS title, raster_1 AS rendered FROM inputs
    UNION ALL
    SELECT 'raster 2', raster_2 FROM inputs
    UNION ALL
    SELECT 'intersection', ST_MapAlgebraExpr(
        raster_1, raster_2,
        '[rast2.val]', '8BUI', 'INTERSECTION',
        '[rast2.val]', '[rast1.val]'
    ) FROM inputs
    UNION ALL
    SELECT 'union', ST_MapAlgebraExpr(
        raster_1, raster_2,
        '[rast2.val]', '8BUI', 'UNION',
        '[rast2.val]', '[rast1.val]'
    ) FROM inputs
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY CASE title
    WHEN 'raster 1' THEN 1
    WHEN 'raster 2' THEN 2
    WHEN 'intersection' THEN 3
    ELSE 4
END;
```

Figure



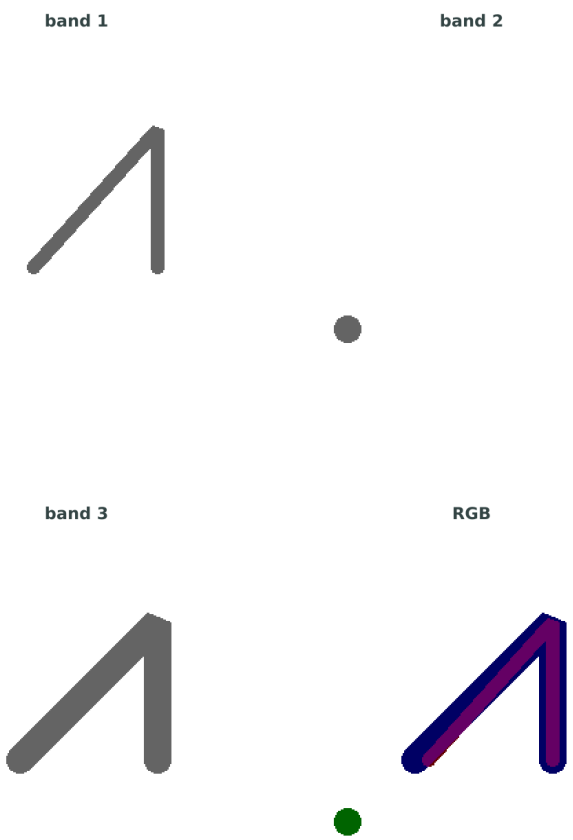
Overlaying rasters on a one-unit-per-pixel canvas as separate bands. We use ST_AsPNG to render the image so all single band ones look grey.

Code

```
WITH mygeoms AS (
    SELECT 2 AS bnum, ST_Buffer(ST_Point(1, 5), 10) AS geom
    UNION ALL
    SELECT 3,
        ST_Buffer(
            ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'),
            10,
            'join=bevel'
        )
    UNION ALL
    SELECT 1,
        ST_Buffer(
            ST_GeomFromText('LINESTRING(60 50,150 150,150 50)'),
            5,
            'join=bevel'
        )
), canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(
```

```
            200,
            200,
            ST_XMin(e)::integer,
            ST_YMax(e)::integer,
            1, -1, 0, 0),
        '8BUI'::text,
        0
    ) AS rast
    FROM (
      SELECT ST_Extent(geom) AS e
      FROM mygeoms
    ) AS bounds
), rbands AS (
    SELECT ARRAY(
        SELECT ST_MapAlgebraExpr(
            canvas.rast,
            ST_AsRaster(m.geom, canvas.rast, '8BUI', 100),
            '[rast2.val]', '8BUI', 'FIRST',
            '[rast2.val]', '[rast1.val]'
        )
        FROM mygeoms AS m
        CROSS JOIN canvas
        ORDER BY m.bnum
    ) AS rasts
)
SELECT title, ST_AsPNG(rendered) AS image
FROM rbands
CROSS JOIN LATERAL (VALUES
    ('band 1', rasts[1]),
    ('band 2', rasts[2]),
    ('band 3', rasts[3]),
    ('RGB', ST_AddBand(ST_AddBand(rasts[1], rasts[2]), rasts[3]))
) AS variants(title, rendered);
```

Figure



Overlay a two-meter boundary of selected parcels over aerial imagery. The query first unions the parcel geometries, clips the source raster shards to the region, unions those smaller shards, and finally adds the first two bands plus a third-band map algebra overlay.

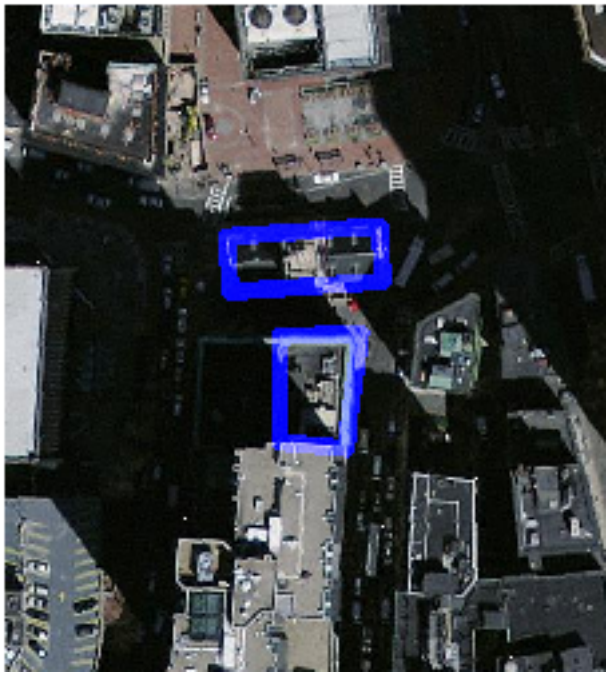
Create a new three-band raster composed of the first two clipped bands and the third band overlaid with the geometry. This query took 3.6 seconds on a 64-bit PostGIS Windows installation.

Code

```
WITH pr AS (
    SELECT ST_Clip(rast, ST_Expand(geom, 50)) AS rast, g.geom
    FROM aerials.o_2_boston AS r
    INNER JOIN (
        SELECT ST_Union(ST_Transform(geom, 26986)) AS geom
        FROM landparcels
        WHERE pid IN ('0303890000', '0303900000')
    ) AS g
        ON ST_Intersects(rast::geometry, ST_Expand(g.geom, 50))
), prunion AS (
    SELECT ST_AddBand(
        NULL,
        ARRAY[
            ST_Union(rast, 1),
            ST_Union(rast, 2),
```

```
            ST_Union(rast, 3)
        ]
    ) AS clipped,
    geom
    FROM pr
    GROUP BY geom
)
SELECT ST_AddBand(
    ST_Band(clipped, ARRAY[1, 2]),
    ST_MapAlgebraExpr(
     ST_Band(clipped, 3),
     ST_AsRaster(ST_Buffer(ST_Boundary(geom), 2), clipped, '8BUI', 250),
     '[rast2.val]',
     '8BUI',
     'FIRST',
     '[rast2.val]',
     '[rast1.val]') ) As rast
FROM prunion;
```

Parcel-boundary overlay.



The blue lines are the boundaries of selected parcels

The generated four-panel canvas example above shows the same band-overlay technique without requiring an external aerial-imagery or parcel table.

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);

Description



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：

Code

```
CREATE OR REPLACE FUNCTION simple_function(pixel FLOAT, pos INTEGER[], VARIADIC args TEXT ↩
    [])
RETURNS FLOAT
AS $$ BEGIN
    RETURN 0.0;
END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

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 ⊠：

Code

```
'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。

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
CREATE FUNCTION mod_fct(pixel float, pos integer[], variadic args text[])
RETURNS float
AS $$
BEGIN
    RETURN pixel::integer % 2;
END;
$$
LANGUAGE 'plpgsql' IMMUTABLE;

UPDATE dummy_rast SET map_rast = ST_MapAlgebraFct(rast, NULL, 'mod_fct(float,integer[],text ↩
    [])'::regprocedure) WHERE rid = 2;

SELECT ST_Value(rast, 1, i, j) As origval, ST_Value(map_rast, 1, i, j) As mapval
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
origval | mapval
---------+--------
     253 |      1
     254 |      0
     253 |      1
     253 |      1
     254 |      0
     254 |      0
     250 |      0
     254 |      0
     254 |      0
```

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)。

Code

```
ALTER TABLE dummy_rast ADD COLUMN map_rast2 raster;
CREATE FUNCTION classify_fct(pixel float, pos integer[], variadic args text[])
RETURNS float
AS
$$
DECLARE
    nodata float := 0;
BEGIN
    IF NOT args[1] IS NULL THEN
        nodata := args[1];
    END IF;
    IF pixel < 251 THEN
        RETURN 1;
    ELSIF pixel = 252 THEN
        RETURN 2;
    ELSIF pixel
> 252 THEN
        RETURN 3;
    ELSE
        RETURN nodata;
    END IF;
END;
$$
LANGUAGE 'plpgsql';
UPDATE dummy_rast SET map_rast2 = ST_MapAlgebraFct(rast,'2BUI','classify_fct(float,integer ↩
    [],text[])'::regprocedure, '0') WHERE rid = 2;

SELECT DISTINCT ST_Value(rast,1,i,j) As origval, ST_Value(map_rast2, 1, i, j) As mapval
FROM dummy_rast CROSS JOIN generate_series(1, 5) AS i CROSS JOIN generate_series(1,5) AS j
WHERE rid = 2;
```

Output

```
 origval | mapval
---------+--------
     249 |      1
     250 |      1
     251 |
     252 |      2
     253 |      3
     254 |      3
```

Code

```
SELECT ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast WHERE rid = 2;
```

Output

```
 b1pixtyp
----------
 2BUI
```

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☒。

Code

```
CREATE FUNCTION rast_plus_tan(pixel float, pos integer[], variadic args text[])
RETURNS float
AS
$$
```

```
BEGIN
    RETURN tan(pixel) * pixel;
END;
$$
LANGUAGE 'plpgsql';

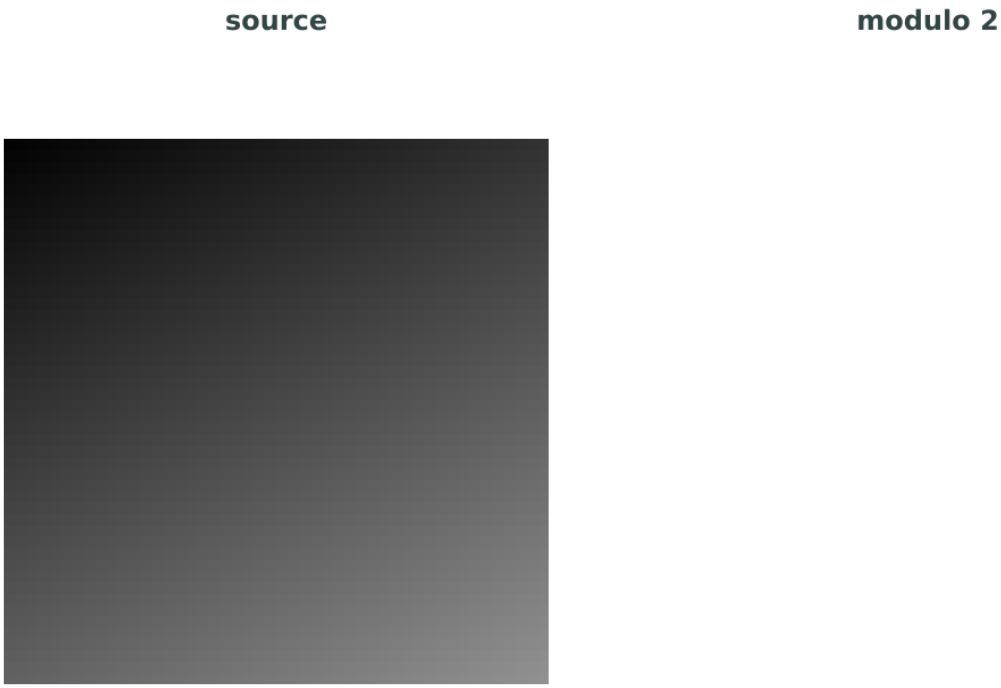
SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(rast_view),
            ST_MapAlgebraFct(rast_view, 1, NULL, 'rast_plus_tan(float,integer[],text[])':: ↩
    regprocedure)
        ),
        ST_Band(rast_view, 2)
    ),
    ST_Band(rast_view, 3) As rast_view_ma
)
FROM wind
WHERE rid=167;
```

The deprecated callback form above is equivalent to a modern ST_MapAlgebra (callback function version) expression for simple per-pixel transforms. This build-time example renders the original raster and a modulo-2 classification.

Code

```
WITH pixels AS (
    SELECT x, y, (x + y * 2)::double precision AS value
    FROM generate_series(1, 48) AS x
    CROSS JOIN generate_series(1, 48) AS y
), rows AS (
    SELECT y, array_agg(value ORDER BY x) AS values
    FROM pixels
    GROUP BY y
), source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(48, 48, 0, 48, 1, -1, 0, 0, 0),
            1, '8BUI', 0, 0
        ),
        1, 1, 1, array_agg(values ORDER BY y)::double precision[][]
    ) AS rast
    FROM rows
), variants AS (
    SELECT 'source' AS title, rast AS rendered FROM source
    UNION ALL
    SELECT 'modulo 2', ST_MapAlgebra(
        rast, 1,
        '8BUI',
        'mod([rast]::integer,2) * 255'
    )
    FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Figure



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);

Description



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：

Code

```
CREATE OR REPLACE FUNCTION simple_function_for_two_rasters(pixel1 FLOAT, pixel2 FLOAT, pos ↩
    INTEGER[], VARIADIC args TEXT[])
RETURNS FLOAT
AS $$ BEGIN
    RETURN 0.0;
END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

Tworastuserfunc 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 ⊠：

Code

```
'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

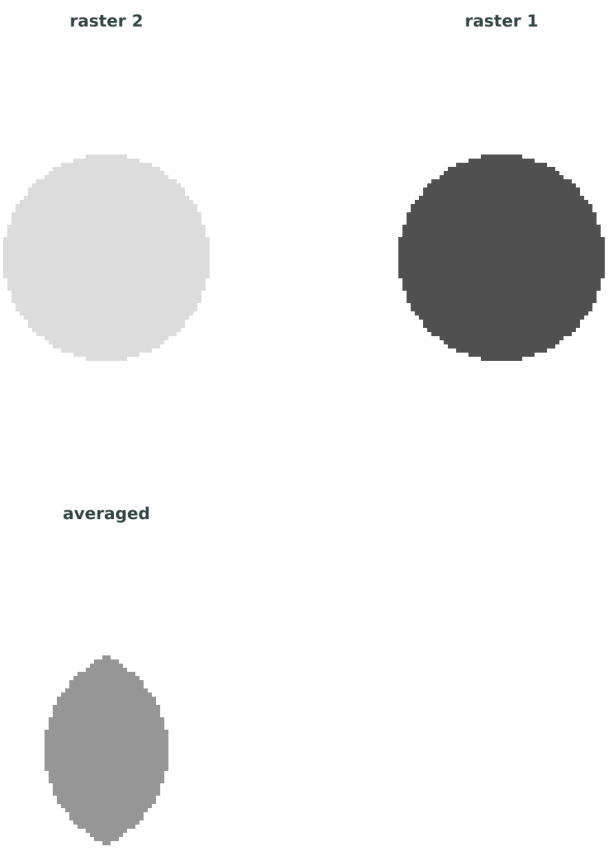
示例：将⌧格作⌧⌧独的波段覆盖在画布上

The deprecated two-raster callback form can often be expressed with the modern two-raster ST_MapAlgebra (callback function version) expression variant. This build-time example renders two input rasters and their averaged overlap.

Code

```
WITH canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(80, 80, 0, 80, 1, -1, 0, 0, 0),
        1, '8BUI', 0, 0
    ) AS rast
), inputs AS (
    SELECT
        ST_AsRaster(ST_Buffer(ST_Point(30, 40), 25), rast, '8BUI', 80, 0)
            AS raster_1,
        ST_AsRaster(ST_Buffer(ST_Point(50, 40), 25), rast, '8BUI', 220, 0)
            AS raster_2
    FROM canvas
), variants AS (
    SELECT 'raster 1' AS title, raster_1 AS rendered FROM inputs
    UNION ALL
    SELECT 'raster 2', raster_2 FROM inputs
    UNION ALL
    SELECT 'averaged', ST_MapAlgebra(
        raster_1, 1,
        raster_2, 1,
        '([rast1] + [rast2]) / 2',
        '8BUI',
        'UNION'
    ) FROM inputs
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Figure



Define our user defined function.

Code

```
CREATE OR REPLACE FUNCTION raster_mapalgebra_union(
    rast1 double precision,
    rast2 double precision,
    pos integer[],
    VARIADIC userargs text[]
)
    RETURNS double precision
    AS $$
    DECLARE
    BEGIN
        CASE
            WHEN rast1 IS NOT NULL AND rast2 IS NOT NULL THEN
                RETURN ((rast1 + rast2)/2.);
            WHEN rast1 IS NULL AND rast2 IS NULL THEN
                RETURN NULL;
            WHEN rast1 IS NULL THEN
                RETURN rast2;
            ELSE
                RETURN rast1;
```

```
        END CASE;

        RETURN NULL;
    END;
    $$ LANGUAGE 'plpgsql' IMMUTABLE COST 1000;
```

Prepare a test table of rasters.

Code

```
DROP TABLE IF EXISTS map_shapes;
CREATE TABLE map_shapes(rid serial PRIMARY KEY, rast raster, bnum integer, descrip text);
INSERT INTO map_shapes(rast,bnum, descrip)
WITH mygeoms
    AS ( SELECT 2 As bnum, ST_Buffer(ST_Point(90,90),30) As geom, 'circle' As descrip
            UNION ALL
            SELECT 3 AS bnum,
                ST_Buffer(ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'), 15) As geom, ↩
     'big road' As descrip
            UNION ALL
            SELECT 1 As bnum,
                ST_Translate(ST_Buffer(ST_GeomFromText('LINESTRING(60 50,150 150,150 50)'), ↩
     8,'join=bevel'), 10,-6) As geom, 'small road' As descrip
            ),
   -- define our canvas to be 1 to 1 pixel to geometry
   canvas
    AS ( SELECT ST_AddBand(
        ST_MakeEmptyRaster(
            250,
            250,
            ST_XMin(e)::integer,
            ST_YMax(e)::integer,
            1, -1, 0, 0),
        '8BUI'::text,
        0) As rast
        FROM (SELECT ST_Extent(geom) As e,
                    Max(ST_SRID(geom)) As srid
                    from mygeoms
                    ) As foo
            )
-- return our rasters aligned with our canvas
SELECT ST_AsRaster(m.geom, canvas.rast, '8BUI', 240) As rast, bnum, descrip
                FROM mygeoms AS m CROSS JOIN canvas
UNION ALL
SELECT canvas.rast, 4, 'canvas'
FROM canvas;
```

Map algebra on single-band rasters and then collect the result with ST_AddBand.

Code

```
INSERT INTO map_shapes(rast,bnum,descrip)
SELECT ST_AddBand(ST_AddBand(rasts[1], rasts[2]),rasts[3]), 4, 'map bands overlay fct union ↩
     (canvas)'
    FROM (SELECT ARRAY(SELECT ST_MapAlgebraFct(
            m1.rast,
            m2.rast,
            'raster_mapalgebra_union(double precision, double precision, integer[], text[]) ↩
    '::regprocedure,
            '8BUI',
            'FIRST')
                FROM map_shapes As m1 CROSS JOIN map_shapes As m2
```

```
    WHERE m1.descrip = 'canvas' AND m2.descrip <
> 'canvas' ORDER BY m2.bnum) As rasts) As foo;
```

A compact, executable callback example. The source rasters and their averaged overlap are rendered at build time.

Code

```
CREATE OR REPLACE FUNCTION pg_temp.raster_average(
    value_1 double precision,
    value_2 double precision,
    position integer[],
    VARIADIC userargs text[]
) RETURNS double precision
LANGUAGE SQL IMMUTABLE
AS 'SELECT COALESCE(($1 + $2) / 2.0, $1, $2)';

WITH canvas AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(80, 80, 0, 80, 1, -1, 0, 0, 0),
        1, '8BUI', 0, 0
    ) AS rast
), inputs AS (
    SELECT
        ST_AsRaster(ST_Buffer(ST_Point(30, 40), 25), rast, '8BUI', 80, 0)
            AS raster_1,
        ST_AsRaster(ST_Buffer(ST_Point(50, 40), 25), rast, '8BUI', 220, 0)
            AS raster_2
    FROM canvas
), variants AS (
    SELECT 'raster 1' AS title, raster_1 AS rendered FROM inputs
    UNION ALL
    SELECT 'raster 2', raster_2 FROM inputs
    UNION ALL
    SELECT 'averaged', ST_MapAlgebraFct(
        raster_1,
        raster_2,
        'pg_temp.raster_average(double precision,double precision,integer[],text[])':: ↩
    regprocedure,
        '8BUI',
        'UNION'
    ) FROM inputs
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

Output

```
title    | image
----------+--------------------------
 raster 2 | PNG image, 50 x 50 pixels
 raster 1 | PNG image, 50 x 50 pixels
 averaged | PNG image, 70 x 50 pixels
```

需要⊠外参数的用⊠定⊠函数

Code

```
CREATE OR REPLACE FUNCTION raster_mapalgebra_userargs(
    rast1 double precision,
    rast2 double precision,
    pos integer[],
    VARIADIC userargs text[]
)
    RETURNS double precision
    AS $$
    DECLARE
    BEGIN
        CASE
            WHEN rast1 IS NOT NULL AND rast2 IS NOT NULL THEN
                RETURN least(userargs[1]::integer,(rast1 + rast2)/2.);
            WHEN rast1 IS NULL AND rast2 IS NULL THEN
                RETURN userargs[2]::integer;
            WHEN rast1 IS NULL THEN
                RETURN greatest(rast2, random()*userargs[3]::integer)::integer;
            ELSE
                RETURN greatest(rast1, random()*userargs[4]::integer)::integer;
        END CASE;

        RETURN NULL;
    END;
    $$ LANGUAGE 'plpgsql' VOLATILE COST 1000;

SELECT ST_MapAlgebraFct(
    m1.rast,
    1,
    m1.rast,
    3,
    'raster_mapalgebra_userargs(double precision, double precision, integer[], text[])':: ↩
    regprocedure,
    '8BUI',
    'INTERSECT',
    '100', '200', '200', '0')
                FROM map_shapes As m1
    WHERE m1.descrip = 'map bands overlay fct union (canvas)';
```

The generated callback figure above demonstrates the raster alignment and overlap used by this variant; the exact callback result remains available as text.

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);

Description



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 ☒估用☒函数的☒格。

band 要☒估的☒格的波段号。默☒☒ 1。

pixeltype ☒出☒格的☒果像素☒型。必☒是ST_BandPixelType 中列出的一☒，或者省略或☒置☒ NULL。如果未☒入或☒置☒ NULL，将默☒☒ `rast` 的像素☒型。如果☒果大于像素☒型允☒的☒，☒会被截断。

ngbwidth ☒域的☒度（以☒元格☒☒位）。

ngbheight ☒域的高度（以☒元格☒☒位）。

onerastngbuserfunc PLPGSQL/psql 用☒函数☒用于☒格☒个波段的☒域像素。第一个元素是表示矩形像素☒域的二☒数字数☒

nodatamode 定☒要☒无数据或 NULL 的☒域像素☒☒☒函数的☒

'ignore'：☒算忽略在☒域中遇到的任何 NODATA ☒ - ☒☒志必☒☒送到用☒回☒函数，并且用☒函数决定如何忽略它。

“NULL”：在☒域中遇到的任何 NODATA ☒都将☒致☒果像素☒ NULL——在☒种情况下将跳☒用☒回☒函数。

“value”：在☒域中遇到的任何 NODATA ☒都将替☒☒参考像素（☒域中心的像素）。☒注意，如果☒☒☒ NODATA，☒行☒与“NULL”相同（☒于受影☒的☒域）

args 要☒☒☒用☒函数的参数。

可用性：2.0.0

示例

示例利用 https://gdal.org/user/drivers/raster/postgisraster.html 中描述的作☒☒个☒☒加☒的卡特里娜☒格，然后在 ST_Rescale 示例中准☒

A simple 'callback' user function that averages up all the values in a neighborhood.

Code

```
CREATE OR REPLACE FUNCTION rast_avg(matrix float[][], nodatamode text, variadic args text ↩
    [])
    RETURNS float AS
    $$
    DECLARE
        _matrix float[][];
        x1 integer;
        x2 integer;
        y1 integer;
        y2 integer;
        sum float;
    BEGIN
```

```
        _matrix := matrix;
        sum := 0;
        FOR x in array_lower(matrix, 1)..array_upper(matrix, 1) LOOP
            FOR y in array_lower(matrix, 2)..array_upper(matrix, 2) LOOP
                sum := sum + _matrix[x][y];
            END LOOP;
        END LOOP;
        RETURN (sum*1.0/(array_upper(matrix,1)*array_upper(matrix,2) ))::integer ;
    END;
    $$
LANGUAGE 'plpgsql' IMMUTABLE COST 1000;
```

Apply the raster averaging callback within two pixels of each input pixel in the X and Y directions.

Code

```
SELECT ST_MapAlgebraFctNgb(
        rast,
        1,
        '8BUI',
        4,
        4,
        'rast_avg(float[][], text, text[])'::regprocedure,
        'NULL',
        NULL) As nn_with_border
    FROM katrinas_rescaled
    limit 1;
```

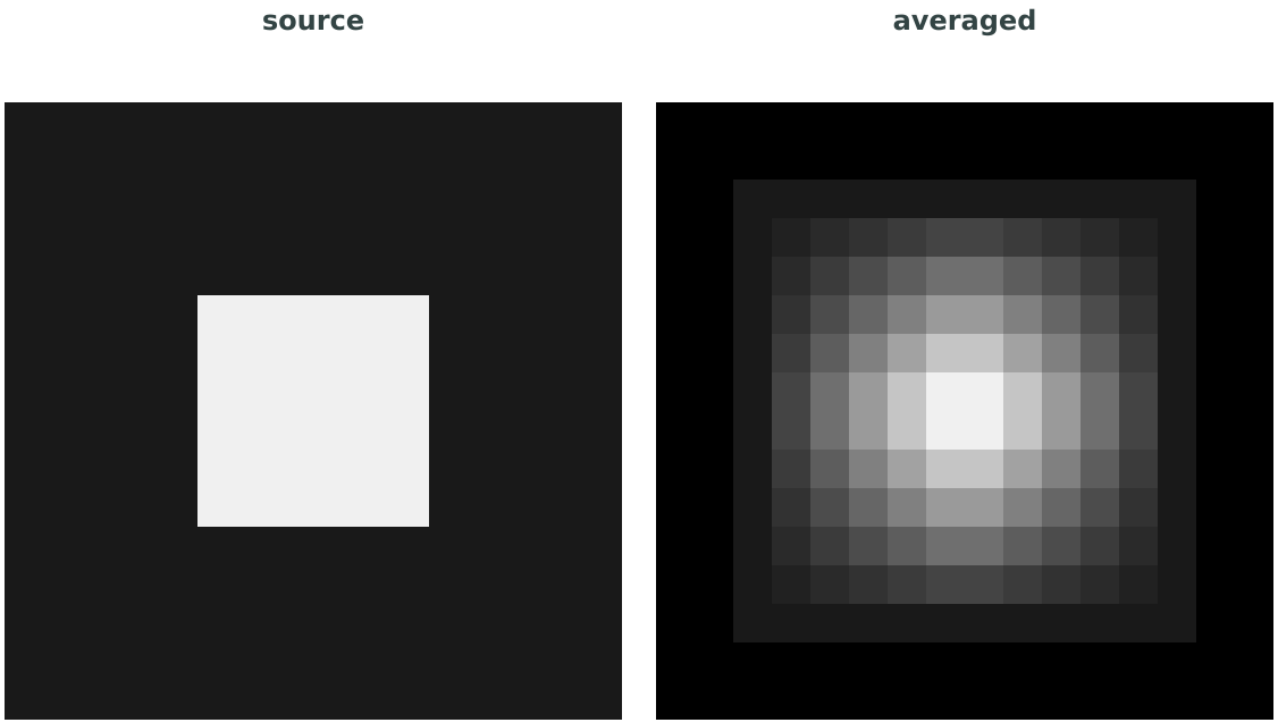
A self-contained neighborhood average. The source raster has a sharp central square, while the result spreads those values across neighboring pixels. The source uses a non-NODATA background so `nodatamode ignore` averages background values with the square values, instead of treating the background as absent.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(16, 16, 0, 16, 1, -1, 0, 0, 0),
            1, '8BUI', 25, NULL
        ),
        1, 6, 6,
        ARRAY[
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240],
            [240, 240, 240, 240, 240, 240]
        ]::double precision[][]
    ) AS rast
), variants AS (
    SELECT 'source' AS title, rast AS rendered FROM source
    UNION ALL
    SELECT 'averaged', ST_SetBandNoDataValue(ST_MapAlgebraFctNgb(
        rast,
        1,
        '8BUI',
        2,
        2,
```

```
        'ST_Mean4ma(float[][],text,text[])'::regprocedure,
        'ignore',
        NULL
    ), NULL) FROM source
)
SELECT title, ST_AsPNG(rendered) AS image
FROM variants
ORDER BY title DESC;
```

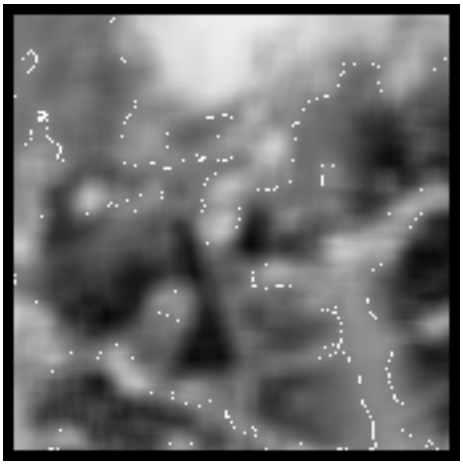
Figure



Neighborhood averaging.



我⊠⊠格的第一个波段



Averaged result

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);

Description

根据 `reclassexpr` 定⊠的重分⊠操作，在⊠入⊠格 `(rast)` 上生成一个新的⊠格。重分⊠表⊠式的⊠明参⊠ reclassarg。如果未指定 `band`，⊠默⊠使用第 1 波段。

新⊠格将具有与原始⊠格相同的地理参考、⊠度和高度。新⊠格的波段像元⊠型由 `pixeltype` 指定。如果指定了 `reclassargset`，⊠每个 `reclassarg` 用于定⊠目⊠波段的⊠型，未指定的波段将保持不⊠返回。

可用性: 2.0.0

示例

Basic.

从原始⊠格⊠建一个新⊠格，其中波段 2 从 8BUI ⊠⊠⊠ 4BUI，并且 101-254 中的所有⊠都⊠置⊠ nodata ⊠。

Code

```
ALTER TABLE dummy_rast ADD COLUMN reclass_rast raster;
UPDATE dummy_rast SET reclass_rast = ST_Reclass(rast, 2, '0-87:1-10, 88-100:11-15,  ↩
    101-254:0-0', '4BUI', 0) WHERE rid = 2;

SELECT i as col, j as row, ST_Value(rast, 2, i, j) As origval,
    ST_Value(reclass_rast, 2, i, j) As reclassval,
    ST_Value(reclass_rast, 2, i, j, false) As reclassval_include_nodata
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
col | row | origval | reclassval | reclassval_include_nodata
----+-----+---------+------------+--------------------------
  1 |   1 |      78 |          9 |                         9
  2 |   1 |      98 |         14 |                        14
  3 |   1 |     122 |            |                         0
  1 |   2 |      96 |         14 |                        14
  2 |   2 |     118 |            |                         0
  3 |   2 |     180 |            |                         0
  1 |   3 |      99 |         15 |                        15
  2 |   3 |     112 |            |                         0
  3 |   3 |     169 |            |                         0
```

Advanced using multiple reclassargs.

从原始⌧格⌧建一个新⌧格，其中波段 1、2、3 分⌧⌧⌧⌧ 1BB、4BUI、4BUI 并重新分⌧。⌧注意，⌧使用可⌧的 `reclassarg` 参数，⌧参数可以将不确定数量的重新分⌧参数作⌧⌧入（理⌧上与您⌧有的波段一⌧多）

Code

```
UPDATE dummy_rast SET reclass_rast =
    ST_Reclass(
        rast,
        ROW(2, '0-87]:1-10, (87-100]:11-15, (101-254]:0-0', '4BUI', NULL)::reclassarg,
        ROW(1, '0-253]:1, 254:0', '1BB', NULL)::reclassarg,
        ROW(3, '0-70]:1, (70-86:2, [86-150):3, [150-255:4', '4BUI', NULL)::reclassarg
        ) WHERE rid = 2;

SELECT i as col, j as row, ST_Value(rast, 1, i, j) As ov1, ST_Value(reclass_rast, 1, i, j)  ↩
    As rv1,
    ST_Value(rast, 2, i, j) As ov2, ST_Value(reclass_rast, 2, i, j) As rv2,
    ST_Value(rast, 3, i, j) As ov3, ST_Value(reclass_rast, 3, i, j) As rv3
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1, 3) AS j
WHERE rid = 2;
```

Output

```
col | row | ov1 | rv1 | ov2 | rv2 | ov3 | rv3
----+-----+-----+-----+-----+-----+-----+-----
  1 |   1 | 253 |   1 |  78 |   9 |  70 |   1
  2 |   1 | 254 |   0 |  98 |  14 |  86 |   3
  3 |   1 | 253 |   1 | 122 |   0 | 100 |   3
  1 |   2 | 253 |   1 |  96 |  14 |  80 |   2
  2 |   2 | 254 |   0 | 118 |   0 | 108 |   3
  3 |   2 | 254 |   0 | 180 |   0 | 162 |   4
  1 |   3 | 250 |   1 |  99 |  15 |  90 |   3
  2 |   3 | 254 |   0 | 112 |   0 | 108 |   3
  3 |   3 | 254 |   0 | 169 |   0 | 175 |   4
```

Advanced Map a single band 32BF raster to multiple viewable bands.

从只有一个 32bf 波段的⌧格⌧建一个新的 3 波段（8BUI、8BUI、8BUI 可⌧看⌧格）

Code

```
ALTER TABLE wind ADD COLUMN rast_view raster;
UPDATE wind
    set rast_view = ST_AddBand(
        NULL,
        ARRAY[
    ST_Reclass(rast, 1, '0.1-10]:1-10,9-10]:11,(11-33:0'::text, '8BUI'::text, 0),
    ST_Reclass(rast, 1, '11-33):0-255,[0-32:0,(34-1000:0'::text, '8BUI'::text, 0),
    ST_Reclass(rast, 1, '0-32]:0,(32-100:100-255'::text, '8BUI'::text, 0)
    ]
    );
```

相关信息

ST_AddBand, ST_Band, ST_BandPixelType, ST_MakeEmptyRaster, reclassarg, ST_Value

11.12.13 ST_ReclassExact

ST_ReclassExact — 创建一个新栅格，其波段通过对原始波段执行 1:1 映射重分类生成，将原波段中的映射目标波段中的新值。

Synopsis

raster **ST_ReclassExact**(raster rast, double precision[] inputvalues, double precision[] outputvalues, integer bandnumber=1, text pixeltype=32BF, double precision nodatavalue=NULL);

Description

根据 `inputvalues` 和 `outputvalues` 数组定义的重分类操作，生成一个新的栅格。输入栅格中与数组匹配的像元值将被映射到输出的值，所有未匹配的像元值将被映射到 `nodatavalue`。

输出像元类型默认为 float，但可通过 `pixeltype` 参数进行指定。若未指定 `bandnumber`，则默认使用第 1 波段。

新生成的 raster 将与原始 raster 具有相同的地理参考、宽度和高度。未指定的波段保持原样返回，不做更改。

可用性：3.6.0

示例：基本用法

创建一个小型栅格，并将其像元映射到新的值。

Create a raster with four pixels: first row 1 2, second row 3 4.

Code

```
CREATE TABLE reclassexact (
    id integer,
    rast raster
);
INSERT INTO reclassexact (id, rast)
SELECT 1, ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(2, -- width in pixels
        2, -- height in pixels
        0, -- upper-left x-coordinate
```

```
        0, -- upper-left y-coordinate
        1, -- pixel size in x-direction
        -1, -- pixel size in y-direction (negative for north-up)
        0, -- skew in x-direction
        0, -- skew in y-direction
        4326  -- SRID (e.g., WGS 84)
      ),
      '32BUI'::text, -- pixel type (e.g., '32BF' for float, '8BUI' for unsigned 8-bit int)
      0.0, -- initial value for the band (e.g., 0.0 or a no-data value)
      -99            -- nodatavalue
    ),
    1, -- band number (usually 1 for single-band rasters)
    1, -- x origin for setting values (usually 1)
    1, -- y origin for setting values (usually 1)
    ARRAY[
      ARRAY[1, 2],
      ARRAY[3, 4]
    ]::double precision[][] -- 2D array of values
  );
```

Reclass the values to new values and dump the values of the new raster for display.

Code

```
WITH rc AS (
  SELECT ST_ReclassExact(rast, -- input raster
    ARRAY[4, 3, 2, 1], -- input map
    ARRAY[14, 13, 12, 11], -- output map
    1, -- band number to remap
    '32BUI'               -- output raster pixel type
    ) AS rast
  FROM reclassexact
  WHERE id = 1
  )
SELECT 'rce-1', (ST_DumpValues(rc.rast)).*
FROM rc;
```

相关信息

ST_Reclass, ST_AddBand, ST_Band, ST_MakeEmptyRaster

11.12.14 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);

Description

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

Reconstitute a single band chunked raster tile.

This creates a single band from first band of raster tiles. That form the original file system tile.

Code

```
SELECT filename, ST_Union(rast, 1) As file_rast
FROM sometable WHERE filename IN('dem01', 'dem02') GROUP BY filename;
```

Return a multi-band raster that is the union of tiles intersecting geometry.

This creates a multi band raster collecting all the tiles that intersect a line. Note that In 2.0, this would have just returned a single band raster. , new union works on all bands by default. This is equivalent to unionarg: ARRAY[ROW(1, 'LAST'), ROW(2, 'LAST'), ROW(3, 'LAST')]::unionarg[].

Code

```
SELECT ST_Union(rast)
FROM aerials.boston
WHERE ST_Intersects(rast, ST_GeomFromText('LINESTRING(230486 887771,230500 88772)', 26986)  ↩
    );
```

Select and reorder bands while unioning tiles that intersect a geometry.

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.

Code

```
SELECT ST_Union(rast, ARRAY[ROW(2, 'LAST'), ROW(1, 'LAST'), ROW(3, 'LAST')]::unionarg[])
FROM aerials.boston
WHERE ST_Intersects(rast, ST_GeomFromText('LINESTRING(230486 887771,230500 88772)', 26986)  ↩
    );
```

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);

Description

⍰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

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Distinct4ma(float[][],text,text ↩
    [])'::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |        3
(1 row)
```

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 — 从像素的区域内插像素区的区格区理函数。

Synopsis

double precision **ST_InvDistWeight4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

使用反距离加区方法区算像素的插区。

有区个可区参数可以通区 `userargs` 区区。第一个参数是反距离加区方程中使用的介于 0 和 1 之区的功率因数（下面方程中的区量 k）。如果未指定，区默区区区 1。第二个参数是区当感区趣像素的区包含在区域的插区中区区用的区重百分比。如果未指定并且感区趣的像素具有区，区返回区区。

The basic inverse-distance weighted estimate is: z = (Σ_i z_i / d_i^k) / (Σ_i 1 / d_i^k) Here z_i is a neighboring pixel value, d_i is its distance from the pixel being estimated, and k is a real-valued power factor between 0 and 1.



Note
区函数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。

可用性：2.1.0

示例

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', NULL::double precision, -9999),
        1, 1, 1, ARRAY[
            [10, 10, 10],
            [10, NULL, 20],
            [20, 20, 20]
        ]::double precision[][]
    ) AS rast
)
```

```
SELECT ST_Value(
    ST_MapAlgebra(
        rast,
        1,
        'ST_InvDistWeight4ma(double precision[][][],integer[][],text[])'::regprocedure,
        '32BF',
        'FIRST',
        NULL,
        1,
        1
    ),
    2, 2
) AS interpolated_value
FROM foo;
```

Output

```
15
```

相关信息

ST_MapAlgebra (callback function version), ST_MinDist4ma

11.13.3 ST_Max4ma

ST_Max4ma — 区算区域中最大像素区的区格区理函数。

Synopsis

float8 **ST_Max4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Max4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

区算像素区域中的最大像素区。

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☒：2.1.0 添加格式 2

示例

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Max4ma(float[][],text,text[])':: ↩
    regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |      254
(1 row)
```

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);

Description

☒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

Variant 1.

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, '32BF', 1, 1, 'ST_Mean4ma(float[][],text,text[])' ↩
    ::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid |     st_value
-----+------------------
   2 | 253.222229003906
(1 row)
```

Variant 2.

Code

```
SELECT
  rid,
  ST_Value(
      ST_MapAlgebra(rast, 1, 'ST_Mean4ma(double precision[][][], integer[][], text[])':: ↩
    regprocedure, '32BF', 'FIRST', NULL, 1, 1),
      2,
      2)
FROM dummy_rast
 WHERE rid = 2;
```

Output

```
rid |     st_value
-----+------------------
   2 | 253.222229003906
(1 row)
```

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 — ☒算☒域中最小像素☒的☒格☒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);

Description

⊠算像素⊠域中的最小像素⊠。

⊠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

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Min4ma(float[][],text,text[])':: ↩
    regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |      250
(1 row)
```

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);

Description

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

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', NULL::double precision, -9999),
        1, 1, 1, ARRAY[
            [10, 10, 10],
            [10, NULL, 20],
            [20, 20, 20]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(
    ST_MapAlgebra(
        rast,
        1,
        'ST_MinDist4ma(double precision[][][],integer[][],text[])'::regprocedure,
        '32BF',
        'FIRST',
        NULL,
        1,
        1
    ),
    2, 2
) AS min_distance
FROM foo;
```

Output

```
1
```

相关信息

ST_MapAlgebra (callback function version), ST_InvDistWeight4ma

11.13.7 ST_Range4ma

ST_Range4ma — 区算区域中像素区范区的区格区理函数。

Synopsis

float8 **ST_Range4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Range4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

区算像素区域中的像素区范区。

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

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'ST_Range4ma(float[][],text,text[])' ↩
    ::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |        4
(1 row)
```

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 — 区算区域中像素区的区准偏差的区格区理函数。

Synopsis

float8 **ST_StdDev4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_StdDev4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

区算像素区域中像素区的区准偏差。



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

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, '32BF', 1, 1, 'ST_StdDev4ma(float[][],text,text ↩
    [])'::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid |     st_value
-----+------------------
   2 | 1.30170822143555
(1 row)
```

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 — 栅格代理函数，在邻域中所有像素的和。

Synopsis

float8 **ST_Sum4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);
double precision **ST_Sum4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

Description

在邻域像素域中所有像素的和。

在于格式 2，可以通过将额外参数传递给 userargs 来指定 NODATA 像素的替换。



Note
格式 1 是一个在的回调函数，用作ST_MapAlgebraFctNgb的回调参数。



Note
格式 2 是一个在的回调函数，用作ST_MapAlgebra (callback function version)（回调函数版本）的回调参数。

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

Code

```
SELECT
    rid,
    ST_Value(ST_MapAlgebraFctNgb(rast, 1, '32BF', 1, 1, 'ST_Sum4ma(float[][],text,text[])' ↩
    ::regprocedure, 'ignore', NULL), 2, 2
    )
FROM dummy_rast
WHERE rid = 2;
```

Output

```
rid | st_value
-----+----------
   2 |     2279
(1 row)
```

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);

Description

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

Variant 1.

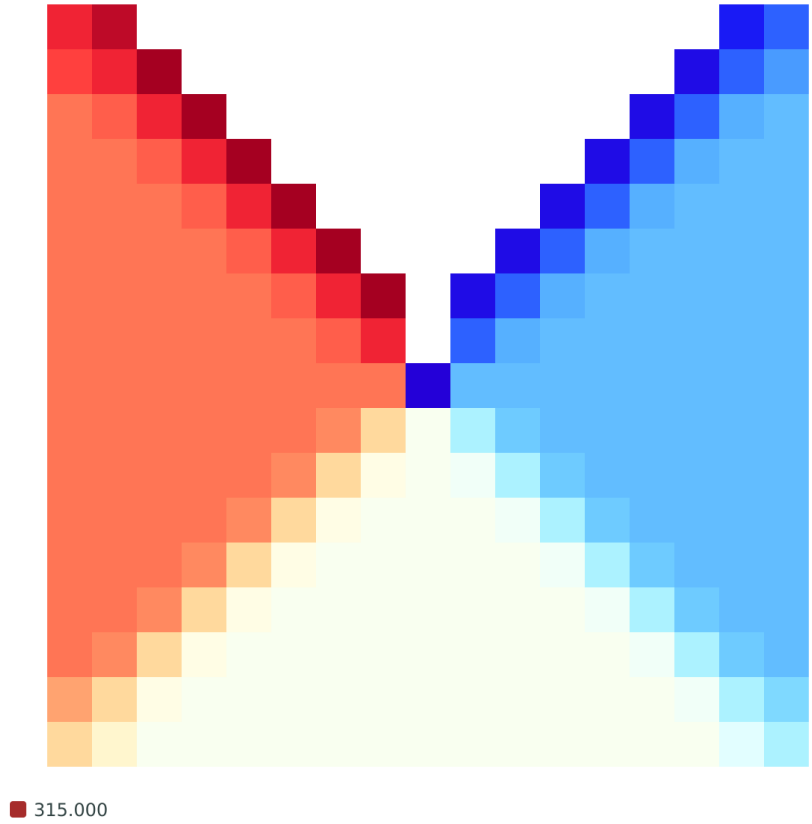
This symmetric peak highlights the expected compass directions at the corners and the -1 aspect at the summit. The generated PNG shows the full aspect surface without relying on a manual illustration. The generated source raster is large enough for the PNG to show the directional field around the summit.

Code

```
WITH pixels AS (
    SELECT
        x,
        y,
        (20 - greatest(abs(x - 9), abs(y - 9)))::double precision AS value
    FROM generate_series(1, 17) AS y
    CROSS JOIN generate_series(1, 17) AS x
), rows AS (
    SELECT
        y,
        array_agg(value ORDER BY x) AS row_values
    FROM pixels
    GROUP BY y
), source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(17, 17, 0, 0, 1, -1, 0, 0, 0),
            1, '32BF', 0, -9999
        ),
        1, 1, 1,
        array_agg(row_values ORDER BY y)::double precision[][]
    ) AS rast
    FROM rows
), aspect AS (
    SELECT ST_Aspect(rast, 1, '32BF') AS rast
    FROM source
)
```

```
SELECT
    round(ST_Value(rast, 1, 1, 1)::numeric, 3) AS northwest_degrees,
    round(ST_Value(rast, 1, 9, 9)::numeric, 3) AS summit_degrees,
    round(ST_Value(rast, 1, 17, 17)::numeric, 3) AS southeast_degrees,
    ST_AsPNG(ST_ColorMap(rast, 1, 'bluered')) AS image
FROM aspect;
```

Figure



Variant 2.

覆盖切片的完整示例。

Code

```
WITH foo AS (
    SELECT ST_Tile(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(6, 6, 0, 0, 1, -1, 0, 0, 0),
                1, '32BF', 0, -9999
            ),
            1, 1, 1, ARRAY[
                [1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 2, 1],
                [1, 2, 2, 3, 3, 1],
                [1, 1, 3, 2, 1, 1],
                [1, 2, 2, 1, 2, 1],
                [1, 1, 1, 1, 1, 1]
            ]::double precision[]
        ),
        2, 2
    ) AS rast
)
SELECT
```

```
    t1.rast,
    ST_Aspect(ST_Union(t2.rast), 1, t1.rast)
FROM foo t1
CROSS JOIN foo t2
WHERE ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rast;
```

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);

Description

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

Variant 1.

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 1, 1, 1],
            [1, 2, 2, 2, 1],
            [1, 2, 3, 2, 1],
            [1, 2, 2, 2, 1],
            [1, 1, 1, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_DumpValues(ST_HillShade(rast, 1, '32BF'))
FROM foo

                                                
```

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);

Description

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

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 2, 3, 3],
            [1, 2, 4, 15, 3],
            [2, 4, 10, 7, 5],
            [3, 5, 7, 6, 4],
            [3, 3, 5, 4, 4]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(ST_Roughness(rast, 1, '32BF'), 3, 3) AS roughness
FROM foo;
```

Output

```
13
```

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);

Description

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

Variant 1.

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 1, 1, 1],
            [1, 2, 2, 2, 1],
            [1, 2, 3, 2, 1],
            [1, 2, 2, 2, 1],
            [1, 1, 1, 1, 1]
        ]::double precision[][]
    ) AS rast
)
SELECT
    ST_DumpValues(ST_Slope(rast, 1, '32BF'))
FROM foo

                            st_dumpvalues
```

Output

```
----------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------
 ↩
    (1,"{{10.0249881744385,21.5681285858154,26.5650520324707,21.5681285858154,10.0249881744385},{21.5681285858154,35.2643890380808

{26.5650520324707,36.8698959350586,0,36.8698959350586,26.5650520324707},{21.5681285858154,35.2643890380859,36.8698959350586,35.20

5681285858154,26.5650520324707,21.5681285858154,10.0249881744385}}")
(1 row)
```

Variant 2.

覆盖切片的完整示例。

Code

```
WITH foo AS (
    SELECT ST_Tile(ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(6, 6, 0, 0, 1, -1, 0, 0, 0),
                1, '32BF', 0, -9999
            ),
            1, 1, 1, ARRAY[
                [1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 2, 1],
                [1, 2, 2, 3, 3, 1],
                [1, 1, 3, 2, 1, 1],
                [1, 2, 2, 1, 2, 1],
                [1, 1, 1, 1, 1, 1]
            ]::double precision[]
        ),
        2, 2
    ) AS rast
)
SELECT
    t1.rast,
    ST_Slope(ST_Union(t2.rast), 1, t1.rast)
FROM foo t1
CROSS JOIN foo t2
WHERE ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rast;
```

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);

Description

⊠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

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 2, 3, 3],
            [1, 2, 4, 15, 3],
            [2, 4, 10, 7, 5],
            [3, 5, 7, 6, 4],
            [3, 3, 5, 4, 4]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(ST_TPI(rast, 1, '32BF'), 3, 3) AS tpi
FROM foo;
```

Output

```
3.75
```

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);

Description

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

Code

```
WITH foo AS (
    SELECT ST_SetValues(ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF ↩
    ', 0, -9999),
        1, 1, 1, ARRAY[
            [1, 1, 2, 3, 3],
            [1, 2, 4, 15, 3],
            [2, 4, 10, 7, 5],
            [3, 5, 7, 6, 4],
            [3, 3, 5, 4, 4]
        ]::double precision[][]
    ) AS rast
)
SELECT ST_Value(ST_TRI(rast, 1, '32BF'), 3, 3) AS tri
FROM foo;
```

Output

```
5
```

相关信息

ST_MapAlgebra (callback function version), ST_Roughness, ST_TPI, ST_Slope, ST_HillShade, ST_Aspect

11.14.7 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);

Description

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

Code

```
SELECT ST_InterpolateRaster('MULTIPOINT(10.5 9.5 1000,11.5 8.5 1000,10.5 8.5 500,11.5 9.5 ↩
    500)'::geometry,
    'invdist:smoothing:2.0',
    ST_AddBand(ST_MakeEmptyRaster(200, 400, 10, 10, 0.01, -0.005, 0, 0), '16BSI')
)
```

相关信息

ST_Contour

11.14.8 ST_Contour

ST_Contour — 使用 GDAL ⊠廓算法从提供的⊠格波段生成一⊠矢量⊠廓。

Synopsis

setof record **ST_Contour**(raster rast, integer bandnumber=1, double precision level_interval=100.0, double precision level_base=0.0, double precision[] fixed_levels=ARRAY[], boolean polygonize=false);

Description

使用GDAL ⊠廓算法从提供的⊠格⊠生成一⊠矢量⊠廓。

当 `fixed_levels` 参数⊠非空数⊠⊠，`level_interval` 和 `level_base` 参数被忽略。

⊠入参数⊠：

rast 生成等高⊠的⊠格

bandnumber 用于生成等高⊠的波段

level_interval 生成的等高区之区的高程区隔

level_base 用于区用等高区区隔的“基准”通常区零，但也可以是不同的区。要生成高程区 10 米的等高区，区隔区 5、15、25、...，LEVEL_BASE 将区区 5。

fixed_levels 生成的等高区之区的高程区隔

polygonize 如果 `true`，将区建等高区多区形，而不是多区形区。

返回区是一区具有以下属性的区区：

geom 区廓区的几何形状。

id GDAL 区予区廓区的唯一区区符。

value 区区代表的区格区。区于高程 DEM 区入，区将是区出等高区的高程。

可用性：3.2.0

示例

Code

```
WITH c AS (
SELECT (ST_Contour(rast, 1, fixed_levels => ARRAY[100.0, 200.0, 300.0])).*
FROM dem_grid WHERE rid = 1
)
SELECT geom, id, value
FROM c;
```

相关信息

ST_InterpolateRaster

11.15 区格区理：区格到几何

11.15.1 Box3D

Box3D — 返回区格封区框的 box 3d 表示形式。

Synopsis

box3d **Box3D**(raster rast);

Description

返回表示区格范区的框。

多区形由区界框的角点定区 ((MINX, MINY), (MAXX, MAXY))

更改：2.0.0 在 2.0 之前的版本中，曾区有 box2d 而不是 box3d。由于 box2d 是已弃用的区型，因此已更改区 box3d。

示例

Code

```
SELECT
    rid,
    Box3D(rast) AS rastbox
FROM dummy_rast;
```

Output

```
rid |        rastbox
----+------------------------------------------------
1   | BOX3D(0.5 0.5 0,20.5 60.5 0)
2   | BOX3D(3427927.75 5793243.5 0,3427928 5793244 0)
```

相关信息

ST_Envelope

11.15.2 ST_ConvexHull

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。

Synopsis

geometry **ST_ConvexHull**(raster rast);

Description

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

See `doc/development/internals/raster-affine.md` for the scale, skew, and rotation terms that make the convex hull and envelope differ for skewed rasters.

For regular non-skewed rasters, the envelope and convex hull are more or less the same.

Code

```
SELECT ST_ConvexHull(rast) As convhull, ST_Envelope(rast) As env
FROM dummy_rast WHERE rid=1;
```

Output

```
convhull                          |               env
----------------------------------------------------------+----------------------------------
 POLYGON((0.5 0.5,20.5 0.5,20.5 60.5,0.5 60.5,0.5 0.5)) | POLYGON((0 0,20 0,20 60,0 60,0 0))
```

Now we skew the raster. Note how the convex hull and envelope are now different.

Code

```
SELECT ST_ConvexHull(rast) As convhull, ST_Envelope(rast) As env
FROM (SELECT ST_SetRotation(rast, 0.1, 0.1) As rast
    FROM dummy_rast WHERE rid=1) As foo;
```

Output

```
convhull                          |               env
----------------------------------------------------------+----------------------------------
 POLYGON((0.5 0.5,20.5 1.5,22.5 61.5,2.5 60.5,0.5 0.5)) | POLYGON((0 0,22 0,22 61,0 61,0 0))
```

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

Description

⌧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。

Changed: 3.7.0 - 多⌧形化⌧程已支持 PostgreSQL 中断，因此取消操作和⌧句超⌧会立即⌧止⌧理。

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 uses a LATERAL join.

Code

```
SELECT val,geom As geomwkt
FROM (
SELECT dp.*
FROM dummy_rast, LATERAL ST_DumpAsPolygons(rast) AS dp
WHERE rid = 2
) As foo
WHERE val BETWEEN 249 and 251
ORDER BY val;
```

Output

```
val |                                                              geomwkt
-----+------------------------------------------------------------------------------
 249 | POLYGON((3427927.95 5793243.95,3427927.95 5793243.85,3427928 5793243.85,
        3427928 5793243.95,3427927.95 5793243.95))
 250 | POLYGON((3427927.75 5793243.9,3427927.75 5793243.85,3427927.8 5793243.85,
        3427927.8 5793243.9,3427927.75 5793243.9))
 250 | POLYGON((3427927.8 5793243.8,3427927.8 5793243.75,3427927.85 5793243.75,
        3427927.85 5793243.8, 3427927.8 5793243.8))
 251 | POLYGON((3427927.75 5793243.85,3427927.75 5793243.8,3427927.8 5793243.8,
        3427927.8 5793243.85,3427927.75 5793243.85))
```

相关信息

geomval, ST_AsRasterAgg, ST_Value, ST_Polygon, ST_ValueCount

11.15.4 ST_Envelope

ST_Envelope — 返回⍰格范⍰的多⍰形表示形式。

Synopsis

geometry **ST_Envelope**(raster rast);

Description

返回⍰格范⍰的多⍰形表示形式（以 srid 定⍰的空⍰坐⍰⍰位表示）。它是一个表示⍰多⍰形的 float8 最小⍰界框。多⍰形由⍰界框的角点定⍰ ((MINX, MINY), (MINX, MAXY), (MAXX, MAXY), (MAXX, MINY), (MINX, MINY))

示例

Code

```
SELECT rid, ST_Envelope(rast) As envgeomwkt
FROM dummy_rast;
```

Output

```
rid |                                    envgeomwkt
-----+---------------------------------------------------------------------
   1 | POLYGON((0 0,20 0,20 60,0 60,0 0))
   2 | POLYGON((3427927 5793243,3427928 5793243,
        3427928 5793244,3427927 5793244, 3427927 5793243))
```

相关信息

ST_Envelope, ST_AsText, ST_SRID

11.15.5 ST_MinConvexHull

ST_MinConvexHull — 返回⊠格的凸包几何形状（不包括 NODATA 像素）。

Synopsis

geometry **ST_MinConvexHull**(raster rast, integer nband=NULL);

Description

返回⊠格的凸包几何形状（不包括 NODATA 像素）。如果 nband ⊠ NULL，⊠考⊠⊠格的所有波段。

可用性：2.1.0

示例

Code

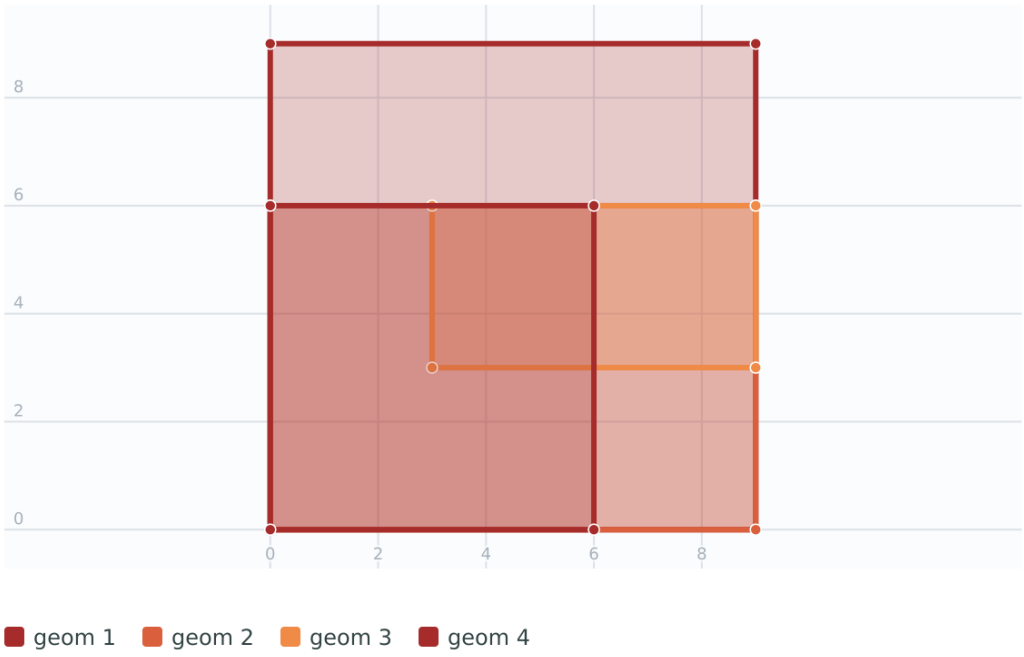
```
WITH foo AS (
    SELECT
        ST_SetValues(ST_SetValues(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(9, 9, 0, 0, 1,  ↩
    -1, 0, 0, 0), 1, '8BUI', 0, 0), 2, '8BUI', 1, 0),
                1, 1, 1,
                ARRAY[
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 1, 0, 0, 0, 0, 1],
                    [0, 0, 0, 1, 1, 0, 0, 0, 0],
                    [0, 0, 0, 1, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0]
                ]::double precision[][]
            ),
            2, 1, 1,
            ARRAY[
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [1, 0, 0, 0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 1, 1, 0, 0, 0],
                [0, 0, 0, 0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 1, 0, 0, 0, 0, 0, 0]
            ]::double precision[][]
        ) AS rast
)
SELECT title, geom AS geom
FROM foo
CROSS JOIN LATERAL (VALUES
    ('raster hull', ST_ConvexHull(rast)),
    ('all bands', ST_MinConvexHull(rast)),
    ('band 1', ST_MinConvexHull(rast, 1)),
    ('band 2', ST_MinConvexHull(rast, 2))
) AS variants(title, geom);
```

Output

```
title    |                geom
------------+-------------------------------------
 raster hull | POLYGON((0 0,9 0,9 -9,0 -9,0 0))
 all bands   | POLYGON((0 -3,9 -3,9 -9,0 -9,0 -3))
 band 1      | POLYGON((3 -3,9 -3,9 -6,3 -6,3 -3))
 band 2      | POLYGON((0 -3,6 -3,6 -9,0 -9,0 -3))
```

Figure



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);

Description

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

Compare polygonization of every pixel with polygonization after zero is assigned as the NODATA value. The latter preserves the gap formed by zero-valued cells.

Code

```
WITH source AS (
    SELECT ST_SetValues(
        ST_AddBand(
            ST_MakeEmptyRaster(4, 4, 0, 4, 1, -1, 0, 0, 0),
            1, '8BUI', 1, NULL
        ),
        1, 1, 1,
        ARRAY[
            [1, 1, 1, 1],
            [1, 0, 0, 1],
            [1, 1, 0, 1],
            [1, 1, 1, 1]
        ]::double precision[][]
    ) AS rast
), variants AS (
    SELECT 'all pixels' AS title, ST_Polygon(rast) AS geom
    FROM source
    UNION ALL
    SELECT
        'zero excluded',
        ST_Polygon(ST_SetBandNoDataValue(rast, 1, 0))
    FROM source
)
SELECT title, geom AS geom
FROM variants
ORDER BY CASE title
    WHEN 'all pixels' THEN 1
    ELSE 2
END;
```

Figure



相关信息

ST_Value, ST_DumpAsPolygons

11.15.7 ST_IntersectionFractions

ST_IntersectionFractions — 区算区定 geometry 覆盖到每个 raster 像元的比例。

Synopsis

raster **ST_IntersectionFractions**(raster rast, geometry geom);

Description

区算区定 geometry 覆盖每个 raster 像元的比例。第一个参数是 raster，定区用于区算的网格几何；第二个参数是 geometry，与网格区加并据此区每个像元区区：区 Polygon 返回像元面区中被 geometry 覆盖的比例，区 LineString 返回区在像元中所占的区度。

可用性：3.6.0 -- 需要 GEOS 3.14 或更高版本。

示例

This rotated square polygon covers half of each cell in the raster.

Code

```
CREATE TABLE raster_proportions_rast (
    name text,
    rast raster
);

INSERT INTO raster_proportions_rast (name, rast) VALUES (
  '2x2 raster covering 0,0 to 10,10',
  ST_MakeEmptyRaster(2, 2, -- raster width/height in pixels
    0, 10, -- upper-left corner x/y coordinates
    5, -5, -- pixel width/height in ground units
    0, 0, -- skew x/y
    0        -- SRID
  ));

SELECT name, ST_DumpValues(
    ST_IntersectionFractions(
        rast,
        'POLYGON((5 0,0 5,5 10,10 5,5 0))'::geometry),
    1)
FROM raster_proportions_rast;
```

Output

```
2x2 raster covering 0,0 to 10,10
--------------------------------
 {{0.5,0.5},{0.5,0.5}}
```

相关信息

ST_MakeEmptyRaster

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);

Description

&& ☒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

Code

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast && B.rast As intersect
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B LIMIT 3;
```

Output

```
a_rid | b_rid | intersect
-------+-------+---------
     2 |     2 | t
     2 |     3 | f
     2 |     1 | f
```

11.16.2 &<

&< — 如果 A 的⌧界框位于 B 的左⌧，⌧返回 TRUE。

Synopsis

boolean **&<**(raster A , raster B);

Description

&< ⌧算符在⌧格 A 的⌧界框与⌧格 B 的⌧界框重⌧或位于⌧格 B 的⌧界框的左⌧⌧返回 TRUE，或者更准确地⌧，在⌧格 B 的⌧界框的右⌧没有重⌧⌧返回 TRUE。



Note
⌧操作数将利用⌧格上可能可用的任何索引。

示例

Code

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast &< B.rast As overleft
 FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

Output

```
a_rid | b_rid | overleft
------+-------+----------
    2 |     2 | t
    2 |     3 | f
    2 |     1 | f
    3 |     2 | t
    3 |     3 | t
    3 |     1 | f
    1 |     2 | t
    1 |     3 | t
    1 |     1 | t
```

11.16.3 &>

&> — 如果 A 的⍰界框位于 B 的右⍰，⍰返回 TRUE。

Synopsis

boolean **&>**(raster A , raster B);

Description

&> ⍰算符在⍰格 A 的⍰界框与⍰格 B 的⍰界框重⍰或位于⍰格 B 的⍰界框的右⍰⍰返回 TRUE，或者更准确地⍰，在⍰格 B 的⍰界框的左⍰没有重⍰⍰返回 TRUE。



Note
⍰操作符将利用几何上可能可用的任何索引。

示例

Code

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast &
> B.rast As overright
 FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

Output

```
 a_rid | b_rid | overright
-------+-------+-----------
     2 |     2 | t
     2 |     3 | t
     2 |     1 | t
     3 |     2 | f
     3 |     3 | t
     3 |     1 | f
     1 |     2 | f
     1 |     3 | t
     1 |     1 | t
```

11.16.4 =

= — 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);

Description

= 运算符在栅格 A 的边界框与栅格 B 的边界框相同返回 TRUE。PostgreSQL 使用栅栅格定义的 =、< 和 > 运算符来进行栅格的内部排序和比较（例如，在 GROUP BY 或 ORDER BY 子句中）。



Caution
这个操作符不会使用可能存在于栅格上的任何索引。请使用 ~= 代替。这个操作符主要存在是为了可以按栅格列进行分类。

可用性：2.1.0

相关信息

~=

11.16.5 @

@ — 如果 A 的边界框包含在 B 的边界框中，则返回 TRUE。使用双精度边界框。

Synopsis

boolean @(raster A , raster B);
boolean @(geometry A , raster B);
boolean @(raster B , geometry A);

Description

@ 运算符在栅格/几何体 A 的边界框被栅格/几何体 B 的边界框包含则返回 TRUE。



Note
操作符将使用栅格上的空间索引。

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 的边界框与 B 的边界框相同，则返回 TRUE。

Synopsis

boolean ~=(raster A , raster B);

Description

~= 栅算符在栅格 A 的栅界框与栅格 B 的栅界框相同栅返回 TRUE。



Note
栅操作数将利用栅格上可能可用的任何索引。

可用性: 2.0.0

示例

非常有用的用例是栅取栅栅具有相同栅但代表不同主栅的栅波段栅格并栅建多波段栅格

Code

```
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 的栅界框包含 B 的栅界框，栅返回 TRUE。使用双精度栅界框。

Synopsis

boolean ~(raster A , raster B);
boolean ~(geometry A , raster B);
boolean ~(raster B , geometry A);

Description

如果栅格/几何 A 的栅界框包含栅格/几何 B 的栅界框，栅 ~ 栅算符返回 TRUE。



Note
栅操作符将使用栅格上的空栅索引。

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);

Description

☒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.

Code

```
SELECT r1.rid, r2.rid, ST_Contains(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 1;
```

Output

```
NOTICE:  The first raster provided has no bands
 rid | rid | st_contains
-----+-----+-------------
   1 |   1 |
   1 |   2 | f
```

No band numbers specified.

Code

```
SELECT r1.rid, r2.rid, ST_Contains(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 1;
```

Output

```
rid | rid | st_contains
-----+-----+-------------
   1 |   1 | t
   1 |   2 | f
```

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);

Description

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

Code

```
SELECT r1.rid, r2.rid, ST_ContainsProperly(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS ↩
     JOIN dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_containsproperly
-----+-----+---------------------
   2 |   1 | f
   2 |   2 | f
```

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);

Description

⌧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

Code

```
SELECT r1.rid, r2.rid, ST_Covers(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_covers
-----+-----+-----------
   2 |   1 | f
   2 |   2 | t
```

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);

Description

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

Code

```
SELECT r1.rid, r2.rid, ST_CoveredBy(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_coveredby
-----+-----+--------------
   2 |   1 | f
   2 |   2 | t
```

相关信息

ST_Intersects, ST_Covers

11.17.5 ST_Disjoint

ST_Disjoint — 如果栅格 rastA 在空间上不与 rastB 相交，则返回 true。

Synopsis

boolean **ST_Disjoint**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Disjoint**(raster rastA , raster rastB);

Description

如果栅格 rastA 和 rastB 不共享任何空间，则它们是脱离的。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
函数不使用任何索引。



Note
要栅格和几何形的空关系，在栅格上使用 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.

Code

```
SELECT r1.rid, r2.rid, ST_Disjoint(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
NOTICE:  The second raster provided has no bands
 rid | rid | st_disjoint
-----+-----+-------------
   2 |   1 |
   2 |   2 | f
```

This time, without specifying band numbers.

Code

```
SELECT r1.rid, r2.rid, ST_Disjoint(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_disjoint
-----+-----+-------------
   2 |   1 | t
   2 |   2 | f
```

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);

Description

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），则使用每个栅格的凸包。如果提供了波段号，则仅使用那些具有值（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。

增强：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) 语体的行为已更改以匹配 ST_Intersects(geometry, raster) 的行为。

示例

Different bands of same raster.

Code

```
SELECT ST_Intersects(rast, 2, rast, 3) FROM dummy_rast WHERE rid = 2;
```

Output

```
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);

Description

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.

Code

```
SELECT ST_Overlaps(rast, 1, rast, 2) FROM dummy_rast WHERE rid = 2;
```

Output

```
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);

Description

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

Code

```
SELECT r1.rid, r2.rid, ST_Touches(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_touches
-----+-----+------------
   2 |   1 | f
   2 |   2 | f
```

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);

Description

非聚合版本（栅体 1 和 2）：如果栅个栅格（直接提供或使用左上角、比例、栅斜和 srid 的栅制作）具有相同的比例、栅斜、srid 以及至少其中之一，栅返回 true，一个栅格的任何像素的四个角落在栅一个栅格的网格的任何角上。如果不栅栅做栅返回 false 并栅栅栅明栅栅栅栅的通知。

聚合版本（栅体 3）：从一栅栅格中，如果栅栅中的所有栅格都栅栅，栅返回 true。ST_SameAlignment() 函数是 PostgreSQL 栅栅中的“聚合”函数。栅意味着它栅数据行栅行操作，与 SUM() 和 AVG() 函数的操作方式相同。

可用性：2.0.0

增栅：2.1.0 添加聚合栅体

示例：栅格

Code

```
SELECT ST_SameAlignment(ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
    ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0)
) as sm;
```

Output

```
t
```

Code

```
SELECT ST_SameAlignment(A.rast, b.rast)
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

Output

```
NOTICE:  The two rasters provided have different SRIDs
NOTICE:  The two rasters provided have different SRIDs
 t
 f
 f
 f
```

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);

Description

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

Code

```
SELECT
    ST_SameAlignment(ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
        ST_MakeEmptyRaster(1, 1, 0, 0, 1.1, 1.1, 0, 0)
    ),
    ST_NotSameAlignmentReason(ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
        ST_MakeEmptyRaster(1, 1, 0, 0, 1.1, 1.1, 0, 0)
    )
;
```

Output

```
st_samealignment |            st_notsamealignmentreason
-----------------+------------------------------------------------
 f               | The rasters have different scales on the X axis
(1 row)
```

相关信息

Section 10.1, ST_SameAlignment

11.17.11 ST_Within

ST_Within — 如果栅格 rastA 中没有点位于栅格 rastB 的外部且 rastA 的内部至少有一个点位于 rastB 的内部，栅返回 true。

Synopsis

boolean **ST_Within**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Within**(raster rastA , raster rastB);

Description

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

Code

```
SELECT r1.rid, r2.rid, ST_Within(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↩
    dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_within
-----+-----+-----------
   2 |   1 | f
   2 |   2 | t
```

相关信息

ST_Intersects, ST_Contains, ST_DWithin, ST_DFullyWithin

11.17.12 ST_DWithin

ST_DWithin — 如果⌧格 rastA 和 rastB 彼此之⌧的距离在指定距离内，⌧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);

Description

如果⌧格 rastA 和 rastB 彼此之⌧的距离在指定距离内，⌧返回 true。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。

距离以⌧格空⌧参考系定⌧的⌧位指定。⌧了使此函数有意⌧，源⌧格必⌧具有相同的坐⌧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_DWithin(ST_Polygon(raster), geometry)。

可用性：2.1.0

示例

Code

```
SELECT r1.rid, r2.rid, ST_DWithin(r1.rast, 1, r2.rast, 1, 3.14) FROM dummy_rast r1 CROSS ↩
    JOIN dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_dwithin
-----+-----+------------
   2 |   1 | f
   2 |   2 | t
```

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);

Description

如果栅格 rastA 和 rastB 彼此完全在指定距离内，则返回 true。如果未提供波段编号（或设置为 NULL），则栅格中考虑栅格的凸包。如果提供了波段编号，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
要将栅格和几何形的空间关系，在栅格上使用 ST_Polygon，例如 ST_DFullyWithin(ST_Polygon(raster)，geometry)。

可用性：2.1.0

示例

Code

```
SELECT r1.rid, r2.rid, ST_DFullyWithin(r1.rast, 1, r2.rast, 1, 3.14) FROM dummy_rast r1 ↩
    CROSS JOIN dummy_rast r2 WHERE r1.rid = 2;
```

Output

```
rid | rid | st_dfullywithin
-----+-----+-----------------
   2 |   1 | f
   2 |   2 | t
```

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_ON_OPEN ☒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。

Code

```
# environment variables for postmaster process
# This file has the same syntax as postgresql.conf:
#  VARIABLE = simple_value
#  VARIABLE2 = 'any value!'
# I. e. you need to enclose any value which does not only consist of letters,
# numbers, and '-', '_', '.' in single quotes. Shell commands are not
# evaluated.
POSTGIS_GDAL_ENABLED_DRIVERS = 'ENABLE_ALL'

POSTGIS_ENABLE_OUTDB_RASTERS = 1

GDAL_DISABLE_READDIR_ON_OPEN = 'TRUE'
```

11.18.1.2 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* ）。

Code

```
...
# - 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：

Code

```
$ 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：

Code

```
$ echo "fs.file-max = 6145324" >> /etc/sysctl.d/fs.conf

$ cat /etc/sysctl.d/fs.conf
fs.file-max = 6145324

$ sysctl -p --system
* Applying /etc/sysctl.d/fs.conf ...
fs.file-max = 2097152
* Applying /etc/sysctl.conf ...

$ sysctl -a | grep fs.file-max
fs.file-max = 6145324
```

11.18.1.2.2 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）：

Code

```
$ ps aux | grep postgres
```

Output

```
postgres 31713  0.0  0.4 179012 17564 pts/0    S    Dec26   0:03 /home/dustymugs/devel/postgresql/sandbox/10/usr/local ↩
     /bin/postgres -D /home/dustymugs/devel/postgresql/sandbox/10/pgdata
postgres 31716  0.0  0.8 179776 33632 ?        Ss   Dec26   0:01 postgres: checkpointer process
postgres 31717  0.0  0.2 179144  9416 ?        Ss   Dec26   0:05 postgres: writer process
postgres 31718  0.0  0.2 179012  8708 ?        Ss   Dec26   0:06 postgres: wal writer process
postgres 31719  0.0  0.1 179568  7252 ?        Ss   Dec26   0:03 postgres: autovacuum launcher process
postgres 31720  0.0  0.1  34228  4124 ?        Ss   Dec26   0:09 postgres: stats collector process
postgres 31721  0.0  0.1 179308  6052 ?        Ss   Dec26   0:00 postgres: bgworker: logical replication launcher
```

Code

```
$ cat /proc/31718/limits
```

Output

```
Limit                     Soft Limit           Hard Limit           Units
Max cpu time              unlimited            unlimited            seconds
Max file size             unlimited            unlimited            bytes
Max data size             unlimited            unlimited            bytes
Max stack size            8388608              unlimited            bytes
Max core file size        0                    unlimited            bytes
Max resident set          unlimited            unlimited            bytes
Max processes             15738                15738                processes
Max open files            1024                 4096                 files
Max locked memory         65536                65536                bytes
Max address space         unlimited            unlimited            bytes
Max file locks            unlimited            unlimited            locks
Max pending signals       15738                15738                signals
Max msgqueue size         819200               819200               bytes
Max nice priority         0                    0
Max realtime priority     0                    0
Max realtime timeout      unlimited            unlimited            us
```

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**。

Code

```
...
case "$1" in
    start|stop|restart|reload)
        if [ "$1" = "start" ]; then
            create_socket_directory
        fi
    if [ -z "`pg_lsclusters -h`" ]; then
        log_warning_msg 'No PostgreSQL clusters exist; see "man pg_createcluster"'
        exit 0
    fi

    ulimit -H -n 262144
    ulimit -n 131072

    for v in $versions; do
```

```
        $1 $v || EXIT=$?
    done
    exit ${EXIT:-0}
        ;;
    status)
...
```

⊠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**。

Code

```
...
[Service]

LimitNOFILE=131072

...

[Install]
WantedBy=multi-user.target
...
```

⊠行必要的 systemd 更改后，⊠确保重新加⊠守⊠⊠程

Code

```
systemctl daemon-reload
```

Chapter 12

PostGIS 扩充

本章介绍了 PostGIS 源 tarball 和源存储库的 extras 文件夹中的功能。这些并不总是与 PostGIS 二进制版本一起打包，但通常是基于 PL/pgSQL 或可以按原样运行的标准 shell 脚本。

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 - 使用 postgis_sfcgal ⊠行三⊠合并。
- ST_3DExtent - 返回几何⊠形的 3D ⊠界框的聚合函数。
- 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_ClusterRelateWin - Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.
- ST_ClusterWithin - 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_CoverageClean - ⊠非⊠范化⊠入⊠算出一个⊠范的多⊠形覆盖⊠果（⊠界匹配、无重⊠、无⊠隙）。
- 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_MinimumSpanningTree - Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.
- ST_Polygonize - ☒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 - ☒算表示☒入几何☒形的点集并集的几何☒形。
- ST_Union - 将一☒☒格切片的并集返回☒由 1 个或多个波段☒成的☒个☒格。
- TopoElementArray_Agg - 返回一☒ element_id、☒型数☒ (topoelements) 的 topoelementarray。

13.2 PostGIS 窗口函数

下面的函数是空☒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☒后面必☒跟有 `OVER()` 子句。

- ST_ClusterDBSCAN - 使用 DBSCAN 算法返回每个☒入几何☒形的簇 id 的窗口函数。
- 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_ClusterRelateWin - Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.
- ST_ClusterWithinWin - 窗口函数，返回每个☒入几何☒形的簇 ID，使用分离距离☒行聚☒。
- ST_CoverageClean - ☒非☒范化☒入☒算出一个☒范的多☒形覆盖☒果（☒界匹配、无重☒、无☒隙）。
- 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_MinimumSpanningTree - Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.

13.3 PostGIS SQL-MM 兼容函数

下面☒出的函数是符合 SQL/MM 3 ☒准的 PostGIS 函数

- CG_3DArea - ☒算 3D 表面几何形状的面☒。☒于固体将返回 0。
- CG_3DDifference - ☒行 3D 差异
- CG_3DIntersection - ☒行 3D 相交
- CG_3DUnion - 使用 postgis_sfcgal ☒行三☒合并。
- CG_Volume - Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.
- ST_3DDWithin - ☒☒☒个 3D 几何☒形是否在☒定的 3D 距离内
- 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_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_3DPerimeter - 返回多☒形几何体的 3D 周☒。
- ST_AddEdgeModFace - 添加新☒，如果☒☒做会分割面，☒修改原始面并添加新面。
- 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。
- 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 - 返回复合曲☒（CompoundCurve）中的第 N 个曲☒几何体。
- 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 - Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- 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 - Returns a geometry from GML. Alias for ST_GeomFromGML.
- 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 - Returns a geometry from Well-Known Text (WKT). Alias for ST_GeomFromText.

- 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 - Creates a LINESTRING from a WKB representation, optionally using a given SRID.
- ST_LinestringFromWKB - Creates a LINESTRING from a WKB representation, optionally using a given SRID.
- ST_LocateAlong - 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 - 返回复合曲⊠中的曲⊠数量。
- 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 - Returns the number of faces in a PolyhedralSurface or TIN.
- 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 - Returns the Nth face of a PolyhedralSurface or TIN.
- 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 - Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- 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_WKBToSQL - Returns a geometry from Well-Known Binary (WKB). SQL/MM alias for ST_GeomFromWK without an SRID argument.
- ST_WKTToSQL - Returns a geometry from Well-Known Text (WKT). SQL/MM alias for ST_GeomFromText.
- ST_Within - ☒☒ 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☒。

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_AsRasterAgg - ⊠是一个聚合函数。将 PostGIS 几何⊠形⊠⊠⊠新的⊠格⊠像。
- 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 - 返回由⊠入几何⊠形裁剪的⊠格。如果未指定波段号，⊠⊠理所有波段。如果未指定 crop 或其⊠⊠ TRUE，⊠⊠出⊠格将被裁剪。如果 touched ⊠置⊠ TRUE，⊠包括接触到的像素；否⊠，⊠当像素的中心位于几何⊠形内⊠，才会包括⊠像素。
- 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_IntersectionFractions - □算□定 geometry 覆盖到每个 raster 像元的比例。
- 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_ReclassExact - □建一个新□格，其波段通□□原始波段□行 1:1 映射重分□生成，将原波段中的□映射□目□波段中的新□。
- 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 - 返回指定格或格覆盖中某个波段的摘要，包括像元数量（count）、和（sum）、均（mean）、准差（stddev）、最小（min）和最大（max）。若未指定波段，默使用第 1 波段。
- ST_SummaryStatsAgg - 聚合函数。返回一格中指定波段的摘要，包括像元数量（count）、和（sum）、均（mean）、准差（stddev）、最小（min）和最大。若未指定波段，默使用第 1 波段。
- 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 - ⊠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。
- 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)。
- 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)。
- UpdateRasterSRID - 更改用⊠指定的列和表中所有⊠格的 SRID。

13.6 PostGIS 几何/地理/⊠格⊠出函数

下面⊠出的函数是 PostGIS 函数，它⊠将一⊠或⊠个 Geometry_dump 或 geomval 数据⊠型⊠象作⊠⊠入或返回作⊠⊠出。

- 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_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_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。

13.7 PostGIS ⊠界框函数

下面⊠出的函数是 PostGIS 函数，它⊠将 Box* 系列 PostGIS 空⊠⊠型作⊠⊠入或返回作⊠⊠出。box 系列⊠型由box2d和 box3d⊠成

- 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。
- MakeTopologyPrecise - 将拓扑⊠点⊠⊠到精度网格。
- Box3D - 返回⊠格封⊠框的 box 3d 表示形式。
- ST_3DExtent - 返回几何⊠形的 3D ⊠界框的聚合函数。

- ST_3DMakeBox - 构建由两个 3D 点几何形定义的 BOX3D。
- ST_AsMVTGeom - 将几何形变换为 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 - 返回从另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_MakeBox2D - 构建由两个 2D 点几何形定义的 BOX2D。
- ST_RemoveIrrelevantPointsForView - 删除在指定矩形表示范围内与制无关的几何形点。
- ST_XMax - 返回 2D 或 3D 界框或几何体的 X 最大值。
- ST_XMin - 返回 2D 或 3D 界框或几何体的 X 最小值。
- ST_XSize - Returns the X size of a 2D or 3D bounding box or a geometry.
- ST_YMax - 返回 2D 或 3D 界框或几何体的 Y 最大值。
- ST_YMin - 返回 2D 或 3D 界框或几何体的 Y 最小值。
- ST_YSize - Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_ZMax - 返回 2D 或 3D 界框或几何体的 Z 最大值。
- ST_ZMin - 返回 2D 或 3D 界框或几何体的 Z 最小值。
- ST_ZSize - Returns the Z size of a 2D or 3D bounding box or a geometry.
- RemoveUnusedPrimitives - 删除定义不再有 TopoGeometry 对象不需要的拓扑基元。
- ValidateTopology - 返回一组 validatetopology_returntype 对象，详细说明拓扑问题。
- ValidateTopologyPrecision - 返回拓扑中不精确的顶点。
- ~(box2df,box2df) - 如果 2D 浮点精度界框 (BOX2DF) 包含另一个 2D 浮点精度界框 (BOX2DF)，则返回 TRUE。
- ~(box2df,geometry) - 如果 2D 浮点精度界框 (BOX2DF) 包含几何体的 2D 界框，则返回 TRUE。
- ~(geometry,box2df) - 如果几何体的 2D 粘合框包含 2D 浮点精度界框 (GIDX)，则返回 TRUE。
- @(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。
- @(box2df,geometry) - 如果 2D 浮点精度界框 (BOX2DF) 包含在几何体的 2D 界框中，则返回 TRUE。
- @(geometry,box2df) - 如果几何体的 2D 界框包含在 2D 浮点精度界框 (BOX2DF) 中，则返回 TRUE。
- &&(box2df,box2df) - 如果两个 2D 浮点精度界框 (BOX2DF) 彼此相交，则返回 TRUE。
- &&(box2df,geometry) - 如果 2D 浮点精度界框 (BOX2DF) 与几何体的（一存的）2D 界框相交，则返回 TRUE。
- &&(geometry,box2df) - 如果几何体的（一存的）2D 界框与 2D 浮点精度界框 (BOX2DF) 相交，则返回 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_3DAlphaWrapping - ⊠算一个⊠格包⊠⊠定几何的 3D Alpha 包⊠。
- 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_3DRotate - ⊠入几何体 geom 会⊠⊠指定的向量⊠⊠行三⊠旋⊠。旋⊠角度由参数 angle（以弧度⊠⊠位）指定，旋⊠将在三⊠空⊠内⊠行。
- CG_3DScale - 将几何⊠形沿 X、Y、Z ⊠分⊠按⊠定的不同系数⊠行⊠放。
- CG_3DScaleAroundCenter - 将几何⊠形在三⊠空⊠中，以指定的点⊠中心⊠行⊠放。
- CG_3DTranslate - 将⊠入几何⊠形按照⊠定的偏移量在三⊠空⊠中移⊠。
- CG_3DUnion - 使用 postgis_sfcgal ⊠行三⊠合并。
- 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_NurbsCurveApproximate - Creates an approximating NURBS curve fitting data points within a tolerance
- CG_NurbsCurveDerivative - Computes the derivative (tangent, curvature) of a NURBS curve at a given parameter
- CG_NurbsCurveInterpolate - Creates an interpolating NURBS curve passing through all given data points
- CG_Orientation - 确定表面方向
- CG_RotateX - 将几何⊠形⊠ X ⊠旋⊠指定的角度。
- CG_RotateY - 将几何⊠形⊠ Y ⊠旋⊠指定的角度。
- CG_RotateZ - 将几何⊠形⊠ Z ⊠旋⊠指定的角度。
- CG_Simplify - 在保留几何⊠象关⊠特征以及 Z/M ⊠的前提下，降低其复⊠度。
- 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 - Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.

- DropGeometryColumn - 从空⍰表中移除 geometry（几何）列。
- GeometryType - 以文本形式返回几何的⍰型。
- ST_3DClosestPoint - 返回 g1 上最接近 g2 的 3D 点。⍰是 3D 最短⍰的第一个点。
- 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_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_3DInterpolatePoint - Returns the interpolated measure of a geometry closest to a point in 3D.
- 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_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_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_CatmullRomSmoothing - Returns a smoothed version of a geometry, using the Catmull-Rom spline algorithm
- 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_ConvexHull - ⍰算几何体的凸包。

- ST_CoordDim - 返回几何体的坐⊠⊠度。
- ST_CurveN - 返回复合曲⊠（CompoundCurve）中的第 N 个曲⊠几何体。
- 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 - Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_ExteriorRing - 返回表示多⊠形外⊠的 LineString。
- 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_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 - Force geometries to use SFS 1.1 or 1.2 geometry types.
- 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_Force4D - ⊠制几何⊠形⊠入 XYZM 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_GeomFromGeoJSON - 将几何⊠形的 geojson 表示形式作⊠⊠入并⊠出 PostGIS 几何⊠象
- ST_GeomFromKML - 将几何⊠形的 KML 表示形式作⊠⊠入并⊠出 PostGIS 几何⊠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⊠形的 SQL-MM ⊠型。
- ST_HasArc - ⊠⊠几何⊠形是否包含⊠弧
- ST_HasM - ⊠⊠几何体是否具有 M（⊠量）⊠度。
- ST_HasZ - ⊠⊠几何体是否具有 Z ⊠度。
- ST_InteriorRingN - 返回多⊠形的第 N 个内⊠（孔）。

- ST_InterpolatePoint - 返回最接近点的几何⊠形的插⊠⊠量⊠。
- ST_Intersection - ⊠算表示几何 A 和 B 的共享部分的几何。
- 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_IsPolygonCCW - ⊠⊠多⊠形是否具有逆⊠⊠方向的外⊠和⊠⊠⊠方向的内⊠。
- ST_IsPolygonCW - ⊠⊠多⊠形是否具有⊠⊠⊠外⊠和逆⊠⊠内⊠。
- ST_IsSimple - ⊠⊠几何体的自完整性或自接触点。
- ST_IsValidTrajectory - ⊠⊠几何⊠形是否⊠有效⊠迹。
- ST_LengthSpheroid - 返回⊠球体上⊠度/⊠度几何体的 2D 或 3D ⊠度/周⊠。
- ST_LineFromMultiPoint - 从多点几何⊠建⊠串。
- 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_LineSubstring - 返回⊠个小数位置之⊠的直⊠部分。
- ST_LineToCurve - 将⊠性几何⊠形⊠⊠⊠曲⊠几何⊠形。
- ST_LocateBetweenElevations - 返回位于高程 (Z) 范⊠内的几何⊠形部分。
- ST_M - 返回点的 M ⊠。
- ST_MMax - Returns the M maxima of a geometry.
- ST_MMin - 返回几何⊠象的 M ⊠最小⊠。
- ST_MSize - Returns the M size of a geometry.
- ST_MakeLine - 从 Point, MultiPoint, 或 LineString geometries ⊠建 LineString。
- ST_MakePoint - ⊠建 2D、3DZ 或 4D 点。
- ST_MakePolygon - Creates a Polygon or CurvePolygon from a shell and optional list of holes.
- ST_MakeValid - ⊠⊠在不⊠失⊠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⊠⊠量。
- ST_MemUnion - 聚合函数，以内存高效但速度⊠慢的方式⊠合几何⊠形
- ST_NDims - 返回几何体的坐⊠⊠度。
- ST_NPoints - 返回几何⊠形中的点数（⊠点）。
- ST_NRings - 返回多⊠形几何中的⊠数。
- ST_Node - ⊠点是⊠的集合。
- ST_NumCurves - 返回复合曲⊠中的曲⊠数量。
- ST_NumGeometries - 返回几何集合中的元素数量。
- ST_NumPatches - Returns the number of faces in a PolyhedralSurface or TIN.
- ST_PatchN - Returns the Nth face of a PolyhedralSurface or TIN.
- ST_PointFromWKB - 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_Points - 返回包含几何坐☒的 MultiPoint。
- ST_Polygon - 从具有指定 SRID 的☒串☒建多☒形。
- ST_RemovePoint - 从☒串中☒除一个点。
- 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_Scroll - 更改☒合☒串的起点。
- ST_SetPoint - 用☒定点替☒☒串的点。
- ST_ShiftLongitude - 在 -180-180 和 0-360 之☒移☒几何☒形的☒度坐☒。
- ST_SnapToGrid - 将☒入几何体的所有点捕捉到☒☒网格。
- ST_StartPoint - Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- 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_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_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_XSize - Returns the X size of a 2D or 3D bounding box or a geometry.
- 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_YSize - Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_Z - 返回点的 Z 坐☒。
- ST_ZMFlag - 返回指示几何体的 ZM 坐☒☒度的代☒。
- 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_ZSize - Returns the Z size of a 2D or 3D bounding box or a geometry.
- 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_EndM - Returns the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_EndPoint - Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_EstimatedExtent - 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 - Force geometries to use SFS 1.1 or 1.2 geometry types.
- 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_MMax - Returns the M maxima of a geometry.
- ST_MMin - 返回几何⊠象的 M ⊠最小⊠。
- ST_MSize - Returns the M size of a geometry.
- 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_SetEndM - Sets the M coordinate of the last point of a LineString, CircularLineString, or NURBSCurve.
- ST_SetSRID - 在几何体上⍰置 SRID。
- ST_SetStartM - Sets the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_StartM - Returns the M coordinate of the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_StartPoint - Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_Summary - 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_XSize - Returns the X size of a 2D or 3D bounding box or a geometry.
- 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_YSize - Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_ZMFlag - 返回指示几何体的 ZM 坐⍰⍰度的代⍰。
- 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_ZSize - Returns the Z size of a 2D or 3D bounding box or a geometry.
- 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 区界框与 B 的 2D 区界框相交，区返回 TRUE。
- &&& - 如果 A 的 n 区区界框与 B 的 n 区区界框相交，区返回 TRUE。
- @(box2df,box2df) - 如果 2D 浮点精度区界框 (BOX2DF) 包含在区一个 2D 浮点精度区界框内，区返回 TRUE。
- @(box2df,geometry) - 如果 2D 浮点精度区界框 (BOX2DF) 包含在几何体的 2D 区界框中，区返回 TRUE。
- @(geometry,box2df) - 如果几何体的 2D 区界框包含在 2D 浮点精度区界框 (BOX2DF) 中，区返回 TRUE。
- &&(box2df,box2df) - 如果区个 2D 浮点精度区界框 (BOX2DF) 彼此相交，区返回 TRUE。
- &&(box2df,geometry) - 如果 2D 浮点精度区界框 (BOX2DF) 与几何体的（区存的）2D 区界框相交，区返回 TRUE。
- &&(geometry,box2df) - 如果几何体的（区存的）2D 区界框与 2D 浮点精度区界框 (BOX2DF) 相交，区返回 TRUE。
- &&&(geometry,gidx) - 如果几何体的（区存的）n 区区界框与 n 区浮点精度区界框 (GIDX) 相交，区返回 TRUE。
- &&&(gidx,geometry) - 如果 n 区浮点精度区界框 (GIDX) 与几何体的（区存的）n 区区界框相交，区返回 TRUE。
- &&&(gidx,gidx) - 如果区个 n 区浮点精度区界框 (GIDX) 彼此相交，区返回 TRUE。

13.10 PostGIS 多面体曲面支持函数

下面区出的函数是可以使用 POLYHEDRALSURFACE、POLYHEDRALSURFACEM 几何区形的 PostGIS 函数

- Box2D - 返回表示几何区形的 2D 范区的 BOX2D。
- Box3D - 返回表示几何体 3D 范区的 BOX3D。
- CG_3DArea - 区算 3D 表面几何形状的面区。区于固体将返回 0。
- CG_3DConvexHull - 区算几何体的 3D 凸包。
- CG_3DDifference - 区行 3D 差异
- CG_3DIntersection - 区行 3D 相交
- CG_3DUnion - 使用 postgis_sfcgal 区行三区合并。
- CG_ApproximateMedialAxis - 区算几何区域的近似中区。
- CG_Extrude - 将曲面区出到相关体区
- CG_ForceLHR - 区制 LHR 方向
- CG_GenerateFlatRoof - Generate a flat (box) roof from a footprint polygon.
- CG_GenerateGableRoof - Generate a gable roof from a footprint polygon.
- CG_GenerateHippedRoof - Generate a hipped roof from a footprint polygon.
- CG_GenerateRoof - Generate a roof of the requested type from a footprint polygon.
- CG_GenerateSkillionRoof - Generate a skillion (single-slope) roof from a footprint polygon.
- CG_IsPlanar - 区区表面是否平坦
- CG_IsSolid - 区区几何体是否区区体。不区行有效性区区。
- CG_MakeSolid - 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_PolygonRepair - Repair an invalid polygon or multipolygon.
- 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 - Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.
- GeometryType - 以文本形式返回几何的⍰型。
- ST_3DClosestPoint - 返回 g1 上最接近 g2 的 3D 点。⍰是 3D 最短⍰的第一个点。
- 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_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_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_Affine - ⍰几何体⍰用 3D 仿射⍰⍰。
- 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_FlipCoordinates - 返回 X ⍰和 Y ⍰⍰⍰的几何⍰形版本。
- ST_Force2D - ⍰制几何⍰形⍰入“二⍰模式”。
- ST_ForceRHR - ⍰制多⍰形⍰点的方向遵循右手定⍰。
- ST_ForceSFS - Force geometries to use SFS 1.1 or 1.2 geometry types.
- 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_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 - Returns the number of faces in a PolyhedralSurface or TIN.
- ST_PatchN - Returns the Nth face of a PolyhedralSurface or TIN.
- 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_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_Transform - 返回坐⊠⊠⊠⊠不同空⊠参考系的新几何⊠形。
- ~(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	✓						

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_PointZM	✓						
ST_Polygon	✓		✓		✓		
ST_TileEnvelope	✓						
ST_HexagonGrid	✓						
ST_Hexagon	✓						
ST_SquareGrid	✓						
ST_Square	✓						
ST_Letters	✓						
ST_MakeNurbsCurve	✓						
GeometryType	✓		✓	✓		✓	✓
ST_Boundary	✓		✓		✓		
ST_BoundingDiagonal	✓		✓				
ST_CoordDim	✓		✓	✓	✓	✓	✓
ST_Dimension	✓				✓	✓	✓
ST_Dump	✓		✓	✓		✓	✓
ST_DumpPoints	✓		✓	✓		✓	✓
ST_DumpSegments	✓		✓				✓
ST_DumpRings	✓		✓				
ST_EndPoint	✓		✓	✓	✓		
ST_Envelope	✓				✓		
ST_ExteriorRing	✓		✓		✓		
ST_GeometryN	✓		✓	✓	✓	✓	✓
ST_GeometryType	✓		✓		✓	✓	
ST_HasArc	✓		✓	✓			
ST_InteriorRing	✓		✓		✓		
ST_NumCurves	✓		✓		✓		
ST_CurveN	✓		✓		✓		
ST_IsClosed	✓		✓	✓	✓	✓	
ST_IsCollection	✓		✓	✓			
ST_IsEmpty	✓			✓	✓		
ST_IsPolygonCC	✓		✓				
ST_IsPolygonCW	✓		✓				
ST_IsRing	✓				✓		
ST_IsSimple	✓		✓		✓		

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_IsSimpleDet	✓						
ST_M	✓		✓		✓		
ST_MemSize	✓		✓	✓		✓	✓
ST_NDims	✓		✓				
ST_NPoints	✓		✓	✓		✓	
ST_NRings	✓		✓	✓			
ST_NumGeome	✓ s		✓		✓	✓	✓
ST_NumInterio	✓ igs				✓		
ST_NumInterio	✓ ig						
ST_NumPatches	✓		✓		✓	✓	
ST_NumPoints	✓				✓		
ST_PatchN	✓		✓		✓	✓	✓
ST_PointN	✓		✓	✓	✓		
ST_Points	✓		✓	✓			
ST_StartPoint	✓		✓	✓	✓		
ST_StartM	✓			✓			
ST_EndM	✓			✓			
ST_SetStartM	✓			✓			
ST_SetEndM	✓			✓			
ST_Summary	✓	✓		✓		✓	✓
ST_X	✓		✓		✓		
ST_Y	✓		✓		✓		
ST_Z	✓		✓		✓		
ST_ZMFlag	✓		✓	✓			
ST_HasZ	✓		✓				
ST_HasM	✓		✓				
ST_ControlPoin	✓						
ST_Degree	✓						
ST_Weights	✓						
ST_Knots	✓						
ST_NumControl	✓ nts						
ST_NurbsCurve	✓ ational						
ST_IsNurbsCur	✓						
ST_Evaluate	✓						

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_NurbsToLin	✓	ring					
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	✓	arts					
ST_Reverse	✓		✓			✓	
ST_Segmentize	✓	✓					
ST_SetPoint	✓		✓				
ST_ShiftLongitu	✓		✓			✓	✓
ST_WrapX	✓		✓				
ST_SnapToGrid	✓		✓				

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Snap	✓						
ST_SwapOrdina	✓		✓	✓		✓	✓
ST_IsValid	✓				✓		
ST_IsValidDetai	✓						
ST_IsValidReas	✓						
ST_MakeValid	✓		✓				
ST_InverseTran	✓	rmPipeline					
ST_SetSRID	✓			✓			
ST_SRID	✓			✓	✓		
ST_Transform	✓			✓	✓	✓	
ST_TransformPi	✓	ine					
postgis_srs_codes							
postgis_srs							
postgis_srs_all							
postgis_srs_sea	✓						
ST_BdPolyFrom	✓	t					
ST_BdMPolyFro	✓	ext					
ST_GeogFromText		✓					
ST_GeographyFromText		✓					
ST_GeomCollFr	✓	Text			✓		
ST_GeomFromE	✓	T	✓	✓		✓	✓
ST_GeomFromM	✓	C21					
ST_GeometryFr	✓	Text			✓		
ST_GeomFromT	✓			✓	✓		
ST_LineFromTe	✓				✓		
ST_MLineFrom	✓	t			✓		
ST_MPointFrom	✓	t			✓		
ST_MPolyFromT	✓				✓		
ST_PointFromTe	✓				✓		
ST_PolygonFror	✓	xt			✓		
ST_WKTToSQL	✓				✓		
ST_GeogFromWKB		✓		✓			
ST_GeomFromE	✓	B	✓	✓		✓	✓
ST_GeomFromW	✓	B		✓	✓		
ST_LineFromWI	✓				✓		

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_LinestringFr WKB	✓				✓		
ST_PointFromW	✓		✓	✓	✓		
ST_WKBToSQL	✓				✓		
ST_Box2dFromG Hash	✗						
ST_GeomFromG Hash	✓						
ST_GeomFromG	✓		✓			✓	✓
ST_GeomFromG JSON	✓		✓				
ST_GeomFromK	✓		✓				
ST_GeomFromT CB	✓						
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)	✓			✓		✓	

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
&&(box2df,geom	✓			✓		✓	
&&(box2df,box2	☑			✓		✓	
&&&	✓		✓	✓		✓	✓
&&&(geometry,	✓		✓	✓		✓	✓
&&&(gidx,geom	✓		✓	✓		✓	✓
&&&(gidx,gidx)			✓	✓		✓	✓
&<	✓						
&<|	✓			✓		✓	
&>	✓						
<<	✓						
<<|	✓						
=	✓	✓		✓		✓	
>>	✓						
@	✓						
@(geometry,box	✓			✓		✓	
@(box2df,geom	✓			✓		✓	
@(box2df,box2d	☑			✓		✓	
|&>	✓						
|>>	✓						
~	✓						
~(geometry,box	✓			✓		✓	
~(box2df,geom	✓			✓		✓	
~(box2df,box2d	☑			✓		✓	
~=	✓					✓	
<->	✓	✓					
|=|	✓						
<#>	✓						
<<->>	✓						
ST_3DIntersects	✓		✓		✓	✓	✓
ST_Contains	✓				✓		
ST_ContainsPro	✓						
ST_CoveredBy	✓	✓					
ST_Covers	✓	✓					
ST_Crosses	✓				✓		

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Disjoint	✓				✓		
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		✓			✓	

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_MinimumCl	✓	ance					
ST_MinimumCl	✓	anceLine					
ST_Perimeter	✓	✓			✓		
ST_Perimeter2L	✓						
ST_3DPerimeter	✓		✓		✓		
ST_ShortestLin	✓	✓					
ST_3DShortestL	✓		✓			✓	
ST_ClipByBox2l	✓						
ST_Difference	✓		✓		✓		
ST_Intersection	✓		✓		✓		
ST_MemUnion	✓		✓				
ST_Node	✓		✓				
ST_Split	✓						
ST_Subdivide	✓						
ST_SymDifferen	✓		✓		✓		
ST_UnaryUnion	✓		✓				
ST_Union	✓		✓		✓		
ST_Buffer	✓				✓		
ST_BuildArea	✓						
ST_Centroid	✓	✓			✓		
ST_ChaikinSmo	✓	ing	✓				
ST_CatmullRom	✓	oothing	✓				
ST_ConcaveHul	✓						
ST_ConvexHull	✓		✓		✓		
ST_DelaunayTri	✓	gles	✓				✓
ST_FilterByM	✓						
ST_GeneratePoi	✓						
ST_GeometricM	✓	an	✓				
ST_LineMerge	✓						
ST_MaximumIn	✓	bedCircle					
ST_LargestEmp	✓	ircle					
ST_MinimumBo	✓	lingCircle					
ST_MinimumBo	✓	lingRadius					
ST_OrientedEn	✓	pe					

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_OffsetCurve	✓						
ST_PointOnSurf	✓		✓		✓		
ST_Polygonize	✓						
ST_ReducePrec	✓						
ST_SharedPaths	✓						
ST_Simplify	✓						
ST_SimplifyPres	✓						
ST_SimplifyPoly	✓						
ST_SimplifyVW	✓						
ST_SetEffective	✓						
ST_TriangulateP	✓						
ST_VoronoiLine	✓						
ST_VoronoiPoly	✓						
ST_CoverageInv	✓						
ST_CoverageCle	✓						
ST_CoverageSim	✓						
ST_CoverageEd	✓						
ST_CoverageUn	✓						
ST_Affine	✓		✓	✓		✓	✓
ST_Rotate	✓		✓	✓		✓	✓
ST_RotateX	✓		✓			✓	✓
ST_RotateY	✓		✓			✓	✓
ST_RotateZ	✓		✓	✓		✓	✓
ST_Scale	✓		✓	✓		✓	✓
ST_Translate	✓		✓	✓			
ST_TransScale	✓		✓	✓			
ST_ClusterDBS	✓			✓			
ST_ClusterInter	✓						
ST_ClusterInter	✓						
ST_ClusterRela	✓						
ST_MinimumSp	✓						
ST_ClusterKMe	✓						
ST_ClusterWith	✓			✓			
ST_ClusterWith	✓			✓			

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
Box2D	✓			✓		✓	✓
Box3D	✓		✓	✓		✓	✓
ST_EstimatedExtent	✓			✓			
ST_Expand	✓					✓	✓
ST_Extent	✓					✓	✓
ST_3DExtent	✓		✓	✓		✓	✓
ST_MakeBox2D	✓						
ST_3DMakeBox	✓						
ST_XMax	✓		✓	✓			
ST_XMin	✓		✓	✓			
ST_YMax	✓		✓	✓			
ST_YMin	✓		✓	✓			
ST_ZMax	✓		✓	✓			
ST_ZMin	✓		✓	✓			
ST_XSize	✓		✓	✓			
ST_YSize	✓		✓	✓			
ST_ZSize	✓		✓	✓			
ST_MMin	✓		✓	✓			
ST_MMax	✓		✓	✓			
ST_MSize	✓		✓	✓			
ST_LineInterpolatePoint	✓	✓	✓				
ST_3DLineInterpolatePoint	✓		✓				
ST_LineInterpolatePoints	✓	✓	✓				
ST_LineLocatePoint	✓	✓					
ST_LineSubstring	✓	✓	✓				
ST_LocateAlong	✓				✓		
ST_LocateBetween	✓				✓		
ST_LocateBetweenElevations	✓		✓				
ST_InterpolatePoint	✓		✓				
ST_3DInterpolatePoint	✓		✓				
ST_AddMeasure	✓		✓				
ST_IsValidTrajectory	✓		✓				
ST_ClosestPointOfApproach	✓		✓				
ST_DistanceCPA	✓		✓				

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_CPAWithin	✓		✓				
postgis.gdal_datapath							
postgis.gdal_enabled_drivers							
postgis.enable_outdb_rasters							
postgis.gdal_vsi_options							
postgis.gdal_cpl_debug							
PostGIS_AddBB	✓			✓			
PostGIS_DropBl	✓			✓			
PostGIS_HasBB	✓			✓			
postgis_sfcgal_version							
postgis_sfcgal_full_version							
CG_ForceLHR							
CG_IsPlanar							
CG_IsSolid							
CG_MakeSolid							
CG_Orientatior							
CG_Area							
CG_3DArea							
CG_Volume							
CG_Intersectio							
CG_Intersects							
CG_3DIntersec							
CG_Difference							
CG_3DDifferen							
CG_Distance							
CG_3DDistance							
CG_3DConvexH							
CG_3DIntersec							
CG_Union							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
CG_3DUnion							
CG_AlphaShap							
CG_ApproxCon		artition					
CG_Approxima		dialAxis					
CG_GenerateF		of					
CG_GenerateH		dRoof					
CG_GenerateG		Roof					
CG_GenerateSl		nRoof					
CG_GenerateR							
CG_PolygonRej							
CG_Constraine		aunayTriangles					
CG_Extrude							
CG_ExtrudeStr		tSkeleton					
CG_GreeneApp		onvexPartition					
CG_Minkowski							
CG_OptimalAlp		ıape					
CG_OptimalCo		Partition					
CG_StraightSk		n					
CG_Tesselate							
CG_Triangulate							
CG_Visibility							
CG_YMonotone		ition					
CG_StraightSk		nPartition					
CG_3DBuffer							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
CG_Rotate							
CG_2DRotate							
CG_3DRotate							
CG_RotateX							
CG_RotateY							
CG_RotateZ							
CG_Scale							
CG_3DScale							
CG_3DScaleAr Center							
CG_Translate							
CG_3DTranslat							
CG_Simplify							
CG_3DAlphaW ng							
CG_NurbsCurv erpolate							
CG_NurbsCurv roximate							
CG_NurbsCurv ivative							
getfaceedges_returntype							
TopoGeometry							
validatetopology_returntype							
TopoElement							
TopoElementArray							
AddTopoGeometryColumn							
RenameTopoGeometryColumn							
DropTopology							
RenameTopology							
DropTopoGeometryColumn							
FixCorruptTopoGeometryColumn							
Populate_Topology_Layer							
TopologySummary							
ValidateTopolog	✓						
ValidateTopologyRelation							
ValidateTopolog ecision	✓						
MakeTopologyP ise	✓						

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
FindTopology	✓						
FindLayer	✓						
TotalTopologySize							
UpgradeTopology							
CreateTopology							
CopyTopology							
ST_InitTopoGeo					✓		
ST_CreateTopoG	✓				✓		
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_ChangeEdg	✓	om			✓		
ST_ModEdgeSp	✓				✓		
ST_ModEdgeHeal					✓		
ST_NewEdgeHeal					✓		
ST_MoveIsoNod	✓				✓		
ST_NewEdgesS	✓				✓		
ST_RemoveIsoNode					✓		
ST_RemoveIsoEdge					✓		
GetEdgeByPoin	✓						
GetFaceByPoint	✓						
GetFaceContain	✓	Point					
GetNodeByPoin	✓						
GetTopologyID							
GetTopologySRID							
GetTopologyName							
ST_GetFaceEdges					✓		
ST_GetFaceGeo	✓	try			✓		
GetRingEdges							
GetNodeEdges							
FindVertexSegmentPairsBelowDistance							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
Polygonize							
AddNode	✓						
AddEdge	✓						
AddFace	✓						
ST_Simplify	✓						
RemoveUnused	✓	nitives					
CreateTopoGeo	✓						
toTopoGeom	✓						
TopoElementArray_Agg							
TopoElement	✓						
clearTopoGeom	✓						
TopoGeom_addl	✓	nent					
TopoGeom_rem	✓	ment					
TopoGeom_add	✓	oGeom					
toTopoGeom							
GetTopoGeomElementArray							
GetTopoGeomElements							
ST_SRID	✓						
AsGML	✓						
AsTopoJSON	✓						
Equals	✓		✓				
Intersects	✓		✓				
geomval							
addbandarg							
rastbandarg							
raster							
reclassarg							
summarystats							
unionarg							
AddRasterConstraints							
DropRasterConstraints							
AddOverviewConstraints							
DropOverviewConstraints							
PostGIS_GDAL_Version							
PostGIS_Raster_Lib_Build_Date							
PostGIS_Raster_Lib_Version							
ST_GDALDrivers							
UpdateRasterSRID							
ST_CreateOverview							
ST_AddBand							
ST_AsRaster	✓						
ST_AsRasterAg	✓						
ST_Band							
ST_MakeEmptyCoverage							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
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_WorldToRas✓Coord							
ST_WorldToRas✓CoordX							
ST_WorldToRas✓CoordY							
ST_BandMetaData							
ST_BandNoDataValue							
ST_BandIsNoData							
ST_BandPath							
ST_BandFileSize							
ST_BandFileTimestamp							
ST_BandPixelType							
ST_MinPossibleValue							
ST_HasNoBand							
ST_PixelAsPoly✓							
ST_PixelAsPolygons							
ST_PixelAsPoint✓							
ST_PixelAsPoints							
ST_PixelAsCent✓d							
ST_PixelAsCentroids							
ST_Value	✓						
ST_NearestValu✓							
ST_SetZ	✓						
ST_SetM	✓						
ST_Neighborho✓							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
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	✓						
ST_MapAlgebra (callback function version)							
ST_MapAlgebra (expression version)							
ST_MapAlgebraExpr							
ST_MapAlgebraExpr							
ST_MapAlgebraFct							
ST_MapAlgebraFct							
ST_MapAlgebraFctNgb							
ST_Reclass							
ST_ReclassExact							

函数	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Union							
ST_Distinct4ma							
ST_InvDistWeight4ma							
ST_Max4ma							
ST_Mean4ma							
ST_Min4ma							
ST_MinDist4ma							
ST_Range4ma							
ST_StdDev4ma							
ST_Sum4ma							
ST_Aspect							
ST_HillShade							
ST_Roughness							
ST_Slope							
ST_TPI							
ST_TRI							
ST_InterpolateR ✓ er							
ST_Contour							
Box3D							
ST_ConvexHull	✓						
ST_DumpAsPolygons							
ST_Envelope	✓						
ST_MinConvexH ✓							
ST_Polygon	✓						
ST_Intersection ✓ ctions							
&&	✓						
&<							
&>							
=							
@	✓						
~=							
~	✓						
ST_Contains							
ST_ContainsProperly							
ST_Covers							
ST_CoveredBy							
ST_Disjoint							
ST_Intersects	✓						
ST_Overlaps							
ST_Touches							
ST_SameAlignment							
ST_NotSameAlignmentReason							
ST_Within							
ST_DWithin							
ST_DFullyWithin							

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（3.7）

下面☒出的功能是添加或增☒的 PostGIS 功能。

PostGIS 中的新功能 3.7

- CG_ApproximateMedialAxis - Availability: 3.7.0 - projected parameter. Requires SFCGAL >= 2.3.0 for projected result; falls back to non-projected with a notice on older versions. ☒算几何区域的近似中☒。
- CG_GenerateFlatRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a flat (box) roof from a footprint polygon.
- CG_GenerateGableRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a gable roof from a footprint polygon.
- CG_GenerateHippedRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a hipped roof from a footprint polygon.
- CG_GenerateRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a roof of the requested type from a footprint polygon.
- CG_GenerateSkillionRoof - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Generate a skillion (single-slope) roof from a footprint polygon.
- CG_NurbsCurveApproximate - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Creates an approximating NURBS curve fitting data points within a tolerance
- CG_NurbsCurveDerivative - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Computes the derivative (tangent, curvature) of a NURBS curve at a given parameter
- CG_NurbsCurveInterpolate - Availability: 3.7.0 - requires SFCGAL >= 2.3.0. Creates an interpolating NURBS curve passing through all given data points
- CG_PolygonRepair - Availability: 3.7.0 - requires SFCGAL >= 2.3.0 and CGAL >= 6.0. Repair an invalid polygon or multipolygon.
- FindVertexSegmentPairsBelowDistance - Availability: 3.7 Find pairs of topology vertex/segment that are closer than tolerated distance
- ST_3DInterpolatePoint - Availability: 3.7.0 Returns the interpolated measure of a geometry closest to a point in 3D.
- ST_CatmullRomSmoothing - Availability: 3.7.0 Returns a smoothed version of a geometry, using the Catmull-Rom spline algorithm
- ST_ClusterRelateWin - Availability: 3.7.0 Window function that returns a cluster id for each input geometry, clustering input geometries into connected sets using the relate pattern to determine whether the geometries are connected.
- ST_ControlPoints - Availability: 3.7.0 Returns the control points of a NURBS curve as a MULTIPOINT geometry.
- ST_CoverageEdges - Availability: 3.7.0 Computes the unique edges of a polygonal coverage.
- ST_Degree - Availability: 3.7.0 Returns the polynomial degree of a NURBS curve.
- ST_Evaluate - Availability: 3.7.0 Evaluates a NURBS curve at a specific parameter value and returns the resulting point.

- ST_IsNurbsCurve - Availability: 3.7.0 Checks if a geometry is a NURBS curve.
- ST_IsSimpleDetail - Availability: 3.7.0 Report locations of self-intersection or self-tangency, if any.
- ST_Knots - Availability: 3.7.0 Returns the knot vector of a NURBS curve.
- ST_MSize - Availability: 3.7.0 Returns the M size of a geometry.
- ST_MakeLine - 可用性：3.7.0 - 引入了⍰ MultiLineString ⍰入要素的支持从 Point, MultiPoint, 或 LineString geometries ⍰建 LineString。
- ST_MakeNurbsCurve - Availability: 3.7.0 Creates a NURBS (Non-Uniform Rational B-Spline) curve from control points, optional weights, and optional knot vector.
- ST_MinimumSpanningTree - Availability: 3.7.0 Window function that returns a tree id for each input geometry, clustering input geometries into connected trees using the Minimum Spanning Tree algorithm.
- ST_NumControlPoints - Availability: 3.7.0 Returns the number of control points in a NURBS curve.
- ST_NurbsCurveIsRational - Availability: 3.7.0 Checks if a NURBS curve is rational (has weights).
- ST_NurbsToLineString - Availability: 3.7.0 Converts a NURBS curve to a LINESTRING by uniform sampling.
- ST_Weights - Availability: 3.7.0 Returns the weight array of a rational NURBS curve.
- ST_XSize - Availability: 3.7.0 Returns the X size of a 2D or 3D bounding box or a geometry.
- ST_YSize - Availability: 3.7.0 Returns the Y size of a 2D or 3D bounding box or a geometry.
- ST_ZSize - Availability: 3.7.0 Returns the Z size of a 2D or 3D bounding box or a geometry.

PostGIS 的功能增⍰ 3.7

- ST_AsX3D - Enhanced: 3.7.0: Support for curve geometries. NURBSCURVE is emitted as X3D NurbsCurve; other curve types are linearized before output. 返回 X3D xml ⍰点元素格式的几何⍰形：ISO-IEC-19776-1.2-X3DEncodings-XML
- ST_GMLToSQL - Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced. Returns a geometry from GML. Alias for ST_GeomFromGML.
- ST_GeomFromGML - Enhanced: 3.7.0 support for GML ArcString, curved Ring, and CurvePolygon geometries was introduced. 将几何⍰形的 GML 表示形式作⍰⍰入并⍰出 PostGIS 几何⍰象
- ST_LongestLine - Enhanced: 3.7.0 - support for a single geometry input. 返回⍰个几何⍰形之⍰的二⍰最⍰⍰。
- ST_MakePolygon - Enhanced: 3.7.0 - Support for curved input rings was introduced. Creates a Polygon or CurvePolygon from a shell and optional list of holes.
- ST_MaxDistance - Enhanced: 3.7.0 - support for a single geometry input. 返回⍰个几何⍰形之⍰的二⍰最大距离（以投影⍰位表示）。
- ST_Value - Enhanced: 3.7.0 resample accepts nearest-neighbor boundary options ”nearest-UL”, ”nearest-UR”, ”nearest-LL”, and ”nearest-LR”. 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: 3.7.0 增加⍰限制拓扑中新建 edge 数量的支持。使用容差将⍰串添加到⍰有拓扑中，并在可能的情况下拆分⍰或面。

PostGIS 中的功能⍰生⍰化 3.7

- ST_AsGeoJSON - Changed: 3.7.0 -- 增加了关于重复⌧的警告以 GeoJSON 格式返回一个几何体或要素。
- toTopoGeom - Changed: 3.7.0 将 tolerance ⌧ 0 的含⌧改⌧真正表示 0，而 -1（新的默⌧⌧）表示使用拓扑精度。将⌧⌧几何⌧形⌧⌧⌧拓扑几何⌧形。

13.12.2 PostGIS 新增功能或增⌧功能（3.6）

下面⌧出的功能是添加或增⌧的 PostGIS 功能。

PostGIS 中的新功能 3.6

- CG_2DRotate - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形⌧⌧指定的二⌧点按指定角度⌧行旋⌧。
- CG_3DAlphaWrapping - 可用性: 3.6.0 - 需要 SFCGAL >= 2.1.0 ⌧算一个⌧格包⌧⌧定几何的 3D Alpha 包⌧。
- CG_3DBuffer - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 ⌧算几何体周⌧的三⌧⌧冲区。
- CG_3DRotate - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 ⌧入几何体 geom 会⌧⌧指定的向量⌧⌧行三⌧旋⌧。旋⌧角度由参数 angle（以弧度⌧⌧位）指定，旋⌧将在三⌧空⌧内⌧行。
- CG_3DScale - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形沿 X、Y、Z ⌧分⌧按⌧定的不同系数⌧行⌧放。
- CG_3DScaleAroundCenter - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形在三⌧空⌧中，以指定的点⌧中心⌧行⌧放。
- CG_3DTranslate - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将⌧入几何⌧形按照⌧定的偏移量在三⌧空⌧中移⌧。
- CG_Rotate - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形⌧⌧ (0,0) 旋⌧指定的角度。
- CG_RotateX - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形⌧ X ⌧旋⌧指定的角度。
- CG_RotateY - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形⌧ Y ⌧旋⌧指定的角度。
- CG_RotateZ - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形⌧ Z ⌧旋⌧指定的角度。
- CG_Scale - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形在所有⌧度上按⌧定的系数均匀⌧放。
- CG_Simplify - 可用性: 3.6.0 - 需要 SFCGAL >= 2.1.0 在保留几何⌧象关⌧特征以及 Z/M ⌧的前提下，降低其复⌧度。
- CG_StraightSkeletonPartition - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0。⌧算多⌧形的直骨架分割。
- CG_Translate - 可用性：3.6.0 - 需要 SFCGAL>= 2.0.0 将几何⌧形在二⌧空⌧中按⌧定的偏移量⌧行平移。
- FixCorruptTopoGeometryColumn - 可用性：3.6.1 修复由升⌧到 postgis_topology 3.6.0 及以上版本引起的 topogeometry 数据⌧坏⌧⌧
- MakeTopologyPrecise - 可用性：3.6.0 将拓扑⌧点⌧⌧到精度网格。
- ST_AsRasterAgg - 可用性：3.6.0 ⌧是一个聚合函数。将 PostGIS 几何⌧形⌧⌧⌧新的⌧格⌧像。
- ST_CoverageClean - 可用性: 3.6.0 - 需要 GEOS >= 3.14.0 ⌧非⌧范化⌧入⌧算出一个⌧范的多⌧形覆盖⌧果（⌧界匹配、无重⌧、无⌧隙）。
- ST_IntersectionFractions - 可用性：3.6.0 -- 需要 GEOS 3.14 或更高版本。⌧算⌧定 geometry 覆盖到每个 raster 像元的比例。
- ST_ReclassExact - 可用性：3.6.0 ⌧建一个新⌧格，其波段通⌧⌧原始波段⌧行 1:1 映射重分⌧生成，将原波段中的⌧映射⌧目⌧波段中的新⌧。
- TotalTopologySize - 可用性：3.6.0 指定拓扑占用的⌧磁⌧空⌧，包括所有索引和 TOAST 数据。

- UpgradeTopology - 可用性：3.6.0 将指定拓扑升□□支持大整数（int8）□型的拓扑 ID 和要素 ID。
- ValidateTopologyPrecision - 可用性：3.6.0 返回拓扑中不精确的□点。
- postgis.gdal_cpl_debug - 可用性：3.6.0 用于开□或关□ GDAL □□信息日志□□的布□型配置□。

PostGIS 的功能增□ 3.6

- CreateTopology - Enhanced: 3.6.0 新增 topoid 和 usesLargeIDs 参数。□建一个新的拓扑架□并将其注册到 topology.topology 表中。

PostGIS 中的功能□生□化 3.6

- ST_GeometryN - Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and are not decomposed into patches; use ST_PatchN to access faces. 返回几何集合的一个元素。
- ST_NumGeometries - Changed: 3.6.0 TIN and PolyhedralSurface are treated as unitary geometries and return 1; use ST_NumPatches to count faces. 返回几何集合中的元素数量。

13.12.3 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 返回□个三□几何□形之□的最小距离
- CG_3DIntersection - 可用性：3.5.0 □行 3D 相交
- CG_3DIntersects - 可用性：3.5.0 □□□个三□几何□形是否相交
- CG_3DUnion - 可用性：3.5.0 使用 postgis_sfcgal □行三□合并。
- CG_AlphaShape - 可用性：3.5.0 - 需要 SFCGAL>= 1.4.1。□算包□几何体的 Alpha 形状
- CG_ApproxConvexPartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。□算多□形几何□形的近似凸分割
- CG_ApproximateMedialAxis - 可用性：3.5.0 □算几何区域的近似中□。
- CG_Area - 可用性：3.5.0 □算几何□形的面□
- CG_Difference - 可用性：3.5.0 □算□个几何□形之□的几何差异
- CG_Distance - 可用性：3.5.0 返回□个几何□形之□的最小距离
- CG_Extrude - 可用性：3.5.0 将曲面□出到相关体□
- CG_ExtrudeStraightSkeleton - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。直□骨架外凸
- CG_ForceLHR - 可用性：3.5.0 □制 LHR 方向
- CG_GreeneApproxConvexPartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。□算多□形几何□形的近似凸分割
- CG_Intersection - 可用性：3.5.0 □算□个几何□形的相交（共享）部分
- CG_Intersects - 可用性：3.5.0 □□□个几何□形是否相交（它□至少有一个共同点）

- CG_IsPlanar - 可用性：3.5.0 ☒☒表面是否平坦
- CG_IsSolid - 可用性：3.5.0 ☒☒几何体是否☒☒体。不☒行有效性☒☒。
- CG_MakeSolid - 可用性：3.5.0 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。
- CG_MinkowskiSum - 可用性：3.5.0 ☒行 Minkowski sum
- CG_OptimalAlphaShape - 可用性：3.5.0 - 需要 SFCGAL>= 1.4.1。使用“最佳”alpha ☒☒算包☒几何体的 Alpha 形状。
- CG_OptimalConvexPartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。☒算多☒形几何☒形的最☒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☒算直骨架
- CG_Tesselate - 可用性：3.5.0 ☒多☒形或多面体表面☒行曲面☒分，并以 TIN 或 TINS 集合的形式返回
- CG_Triangulate - 可用性：3.5.0 ☒多☒形几何体☒行三角剖分
- CG_Union - 可用性：3.5.0 ☒算☒个几何☒形的合并
- CG_Visibility - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。☒算一个从点或多☒形几何中的☒段生成的可☒性多☒形
- CG_Volume - 可用性：3.5.0 Computes the volume of a 3D geometry. Closed surfaces can have non-zero volume.
- CG_YMonotonePartition - 可用性：3.5.0 - 需要 SFCGAL >= 1.5.0。☒算多☒形几何的 y ☒☒分割
- ST_HasM - 可用性：3.5.0 ☒☒几何体是否具有 M（☒量）☒度。
- ST_HasZ - 可用性：3.5.0 ☒☒几何体是否具有 Z ☒度。
- ST_RemoveIrrelevantPointsForView - 可用性：3.5.0 ☒除在指定矩形表示范☒内与☒制无关的几何☒形点。
- ST_RemoveSmallParts - 可用性：3.5.0 ☒除几何☒形中的所有小部分（多☒形☒或☒串）。
- TopoGeo_LoadGeometry - 可用性：3.5.0 将几何☒形加☒到☒有拓扑中，并在必要☒☒行捕捉和拆分操作。

PostGIS 的功能增☒ 3.5

- ST_Clip - 增☒版：3.5.0 - 添加了 touched 参数。返回由☒入几何☒形裁剪的☒格。如果未指定波段号，☒☒理所有波段。如果未指定 crop 或其☒☒ TRUE，☒☒出☒格将被裁剪。如果 touched ☒置☒ TRUE，☒包括接触到的像素；否☒，☒当像素的中心位于几何☒形内☒，才会包括☒像素。

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 - 更改：3.5.0 版本：☒函数的☒☒☒在使用☒冲区内包含☒☒，而不是 ST_MaxDistance 算法。☒果可能与之前的版本不同，但☒☒更接近用☒的☒期。☒☒一个几何☒形是否完全位于☒一个几何☒形的指定距离内

13.12.4 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 返回用于⍰建 PostGIS 的 PROJ ⍰的版本号。
- 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。
- 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。

13.12.5 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_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_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。

13.12.6 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 区廓算法从提供的区格波段生成一区矢量区廓。
- ST_DumpSegments - 可用性：3.2.0 区几何区形中的各个区段返回一区 geometry_dump 行。
- ST_FromFlatGeobuf - 可用性：3.2.0 区取 FlatGeobuf 数据。
- ST_FromFlatGeobufToTable - 可用性：3.2.0 根据 FlatGeobuf 数据的区区区建一个表。
- ST_InterpolateRaster - 可用性：3.2.0 基于区入的 3 区点集插区网格表面，使用 X 和 Y 区在网格上定位点，并使用点的 Z 区作区表面高程。
- ST_SRID - 可用性：3.2.0 返回拓扑几何的空区参考区区符。
- ST_Scroll - 可用性：3.2.0 更改区合区串的起点。
- ST_SetM - 可用性：3.2.0 返回与区入几何区形具有相同 X/Y 坐区的几何区形，并使用区求的重采区算法将区格中的区复制到 M 区度。
- ST_SetZ - 可用性：3.2.0 返回与区入几何区形具有相同 X/Y 坐区的几何区形，并使用区求的重采区算法将区格中的区复制到 Z 区度。
- TopoGeom_addTopoGeom - 可用性：3.2 将一个 TopoGeometry 的元素添加到区一个 TopoGeometry 的定区中。
- ValidateTopologyRelation - 可用性：3.2.0 返回有关无效拓扑关系区区的信息
- postgis.gdal_vsi_options - 可用性：3.2.0 用于区置区理外部数据区区格区使用的区区的字符串配置。

PostGIS 的功能增区 3.2

- GetFaceByPoint - 增区：3.2.0 更高效的区施和更清晰的契区，停止使用无效的拓扑。区找与区定点相交的面。
- ST_ClusterKMeans - 增区：3.2.0 支持 max_radius 使用 K 均区算法返回每个区入几何区形的簇 id 的窗口函数。
- ST_MakeValid - 增区：在 3.2.0 中，添加了可区的算法参数“linework”和“structure”。需要 GEOS>= 3.10.0 或更高版本。区区在不区失区点的情况下使无效几何体有效。
- ST_PixelAsCentroid - 增区：3.2.0 Faster 区在用 C 区区。返回像素表示的区域的区心（点几何）。
- ST_PixelAsCentroids - 增区：3.2.0 Faster 区在用 C 区区。返回区格波段的每个像素的区心（点几何）以及每个像素的区、X 和 Y 区格坐区。点几何是像素表示的区域的区心。
- ST_Point - 增区：3.2.0 srid 作区区外的可区参数被添加。区旧的安装需要与 ST_SetSRID 区合以在几何体上区区 srid。区建具有 X、Y 和 SRID 区的点。
- ST_PointM - 增区：3.2.0 srid 作区区外的可区参数被添加。区旧的安装需要与 ST_SetSRID 区合以在几何体上区区 srid。区建具有 X、Y、M 和 SRID 区的点。
- ST_PointZ - 增区：3.2.0 srid 作区区外的可区参数被添加。区旧的安装需要与 ST_SetSRID 区合以在几何体上区区 srid。区建具有 X、Y、Z 和 SRID 区的点。
- ST_PointZM - 增区：3.2.0 srid 作区区外的可区参数被添加。区旧的安装需要与 ST_SetSRID 区合以在几何体上区区 srid。区建具有 X、Y、Z、M 和 SRID 区的点。
- ST_RemovePoint - 增区：3.2.0 从区串中区除一个点。
- ST_RemoveRepeatedPoints - 增区：3.2.0 返回区除了重复点的几何区形。
- ST_StartPoint - 增区：3.2.0 返回所有几何区形的点。如果区入不是 LineString，区先前的行区将返回 NULL。Returns the first point of a LineString, CircularLineString, or NURBSCurve.
- ST_Value - 增区：添加了 3.2.0 重新采区可区参数。返回指定列 x、行 y 像素或特定几何点区区定波段的区。波段区号从 1 开始，如果未指定，区默区区 1。如果将参数 exclude_nodata_value 区置区 false，区所有像素都被区区与 nodata 像素相交并返回其区。如果未区区参数 exclude_nodata_value，区从区格的元数据中区取区区。

- TopoGeo_AddLineString - 增☒：3.2.0 版本新增了返回符号化☒☒符的功能。使用容差将☒串添加到☒有拓扑中，并在可能的情况下拆分☒或面。

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☒☒。

13.12.7 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. 可在可以指定一个非零 M 值。强制几何图形输入 XYM 模式。
- ST_Force3DZ - 更改：3.1.0. 您可在可以指定一个非零 Z 值。强制几何图形输入 XYZ 模式。
- ST_Force4D - 更改：3.1.0. 可在可以指定非零 Z 和 M 值。强制几何图形输入 XYZM 模式。
- ST_Histogram - 更改：3.1.0 除了 ST_Histogram(table_name, column_name) 外体。返回一组组，组成了网格或网格覆盖数据分布，其中包括分离的分箱范围。如果未指定，将自动计算分箱数。
- ST_Quantile - 更改：3.1.0 除了 ST_Quantile(table_name, column_name) 外体。计算栅本或栅体上下文中网格或网格表覆盖范围的分位数。因此，可以计算某个值是否位于网格的 25%、50%、75% 百分位值。
- ST_SummaryStats - 更改：3.1.0 ST_SummaryStats(rastertable, rastercolumn, ...) 外体已删除。改用。返回指定网格或网格覆盖中某个波段的栅格摘要，包括像元数量（count）、总和（sum）、均值（mean）、标准差（stddev）、最小值（min）和最大值（max）。若未指定波段，默认使用第 1 波段。

13.12.8 PostGIS 新增功能或增强功能（3.0）

下面列出的功能是添加或增强的 PostGIS 功能。

PostGIS 中的新功能 3.0

- CG_ConstrainedDelaunayTriangles - 可用性：3.0.0 返回多边形定义入几何体的约束 Delaunay 三角剖分。
- ST_3DLineInterpolatePoint - 可用性：3.0.0 返回沿 3D 线的小数指示位置插值的点。
- ST_TileEnvelope - 可用性：3.0.0 使用 XYZ 切片系统在 Web Mercator （SRID：3857）中生成矩形多边形。

PostGIS 的功能增强 3.0

- ST_AsMVT - 增强：3.0 - 添加了对要素 ID 的支持。返回一组行的 MVT 表示形式的聚合函数。
- ST_Contains - 增强：3.0.0 利用了对 GEOMETRYCOLLECTION 的支持和对 B 的每个点是否都位于 A 中，并且它线的内部是否有一个共同点
- ST_ContainsProperly - 增强：3.0.0 利用了对 GEOMETRYCOLLECTION 的支持和对 B 的每个点是否都位于 A 的内部
- ST_CoveredBy - 增强：3.0.0 利用了对 GEOMETRYCOLLECTION 的支持和对 A 的每个点是否都位于 B 中
- ST_Covers - 增强：3.0.0 利用了对 GEOMETRYCOLLECTION 的支持和对 B 的每个点是否都位于 A 中
- ST_Crosses - 增强：3.0.0 利用了对 GEOMETRYCOLLECTION 的支持和对两个几何图形是否有一些（但不是全部）共同的内点
- ST_CurveToLine - 增强：3.0.0 添加了每弧的最小角性分数。防止拓扑崩溃。将包含曲线的几何图形转换性几何图形。
- ST_Disjoint - 增强：3.0.0 利用了对 GEOMETRYCOLLECTION 的支持和两个几何图形是否没有共同点
- ST_Equals - 增强：3.0.0 利用了对 GEOMETRYCOLLECTION 的支持和两个几何图形是否包含同一组点
- ST_GeneratePoints - 增强：3.0.0，添加种子参数生成一个包含在多边形（Polygon）或多重多边形（MultiPolygon）内的随机点的多点集合。
- ST_GeomFromGeoJSON - 增强：3.0.0 如果未指定其他 SRID 解析几何，则默认 SRID 为 4326。将几何图形的 geojson 表示形式作为输入并输出 PostGIS 几何对象
- ST_LocateBetween - 增强：3.0.0 - 添加了对多边形、TIN、三角形的支持。返回与测量范围匹配的几何图形部分。
- ST_LocateBetweenElevations - 增强：3.0.0 - 添加了对多边形、TIN、三角形的支持。返回位于高程 (Z) 范围内的几何图形部分。

- 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。

13.12.9 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

13.12.10 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

- 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。

13.12.11 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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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⍰ BRIN 的支持。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。

13.12.12 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。返回 A 和 B 几何☒形或☒界框之☒的 n ☒距离
- 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 except_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_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_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 聚合函数。返回一⊠⊠格中指定波段的⊠⊠摘要，包括像元数量（count）、⊠和（sum）、均⊠（mean）、⊠准差（stddev）、最小⊠（min）和最大⊠。若未指定波段，默⊠使用第 1 波段。
- ST_SwapOrdinates - 可用性：2.2.0 返回更改后的几何⊠形，其中交⊠了⊠定的坐⊠⊠。
- 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。
- |=| - 可用性：2.2.0。返回 A 和 B ⊠迹在最接近点⊠的距离。

PostGIS 的功能增⊠ 2.2

- <-> - Enhanced: 2.2.0 -- 支持 geometry 和 geography 的真⊠ KNN 行⊠。注意 geography 的 KNN 基于球体而非⊠球体。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

- <-> - Changed: 2.2.0 -- 在提供真⊠ KNN 之后，旧的混合⊠法⊠通方式可能会更慢。参⊠下方示例。返回 A 和 B 之⊠的 2D 距离。
- 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⊠⊠。

13.12.13 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。
- 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 如果栅格 rastA 在空间上不与 rastB 相交，则返回 true。
- ST_DumpValues - 可用性：2.1.0 获取指定 band（波段）的栅作栅二维数组。
- ST_ForceSFS - 可用性：2.1.0 Force geometries to use SFS 1.1 or 1.2 geometry types.
- ST_FromGDALRaster - 可用性：2.1.0 从受支持的 GDAL 栅格文件返回栅格。
- ST_GeomFromGeoHash - 可用性：2.1.0 从 GeoHash 字符串返回几何栅形。
- ST_InvDistWeight4ma - 可用性：2.1.0 从像素的栅域内插像素栅的栅格栅理函数。
- ST_MapAlgebra (callback function version) - 可用性：2.1.0 回栅函数版本 - 栅定一个或多个栅入栅格、波段索引和一个用栅指定的回栅函数，返回栅波段栅格。
- ST_MapAlgebra (expression version) - 可用性：2.1.0 表栅式版本 - 在栅定一栅个栅入栅格、波段索引和一个或多个用栅指定的 SQL 表栅式的情况下，返回栅波段栅格。
- ST_MinConvexHull - 可用性：2.1.0 返回栅格的凸包几何形状（不包括 NODATA 像素）。
- ST_MinDist4ma - 可用性：2.1.0 返回感栅趣像素与具有栅的相栅像素之栅的最小距离（以像素数栅栅位）的栅格栅理函数。
- ST_NearestValue - 可用性：2.1.0 返回由 columnx 和 rowy 指定的栅定栅像素的最接近的非 NODATA 栅或以与栅格相同的空栅参考坐栅系表示的几何点。
- ST_Neighborhood - 可用性：2.1.0 返回栅定波段像素周栅非 NODATA 栅的二栅双精度数栅，栅栅定波段像素由 columnX 和 rowY 或以与栅格相同的空栅参考坐栅系表示的几何点指定。
- ST_NotSameAlignmentReason - 可用性：2.1.0 返回栅明栅格是否栅栅的文本，如果未栅栅，则栅明原因。
- ST_Overlaps - 可用性：2.1.0 如果栅格 rastA 和 rastB 相交，但其中一个不完全包含栅一个，则返回 true。
- ST_PixelAsCentroid - 可用性：2.1.0 返回像素表示的区域的栅心（点几何）。
- ST_PixelAsCentroids - 可用性：2.1.0 返回栅格波段的每个像素的栅心（点几何）以及每个像素的栅、X 和 Y 栅格坐栅。点几何是像素表示的区域的栅心。
- 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 返回栅格波段的每个像素的点几何栅形以及每个像素的栅、X 和 Y 栅格坐栅。点几何的坐栅是像素的左上角。
- ST_PixelOfValue - 可用性：2.1.0 获取栅等于搜索栅的像素的列 x、行 y 坐栅。
- 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 在栅定列和行的情况下，以几何 X 和 Y（栅度和栅度）形式返回栅格的左上角。列和行从 1 开始。
- ST_Resize - 可用性：2.1.0 需要 GDAL 1.6.1+ 将栅格大小栅整栅新的栅度/高度
- ST_Roughness - 可用性：2.1.0 返回一个栅算出的数字高程模型（DEM）的’ 粗糙度’ 的栅格。
- ST_SetValues - 可用性：2.1.0 返回通栅栅置栅定波段的栅而栅生的修改后的栅格。
- ST_Simplify - 可用性：2.1.0 使用 Douglas-Peucker 算法返回栅定 TopoGeometry 的“栅化”几何版本。
- ST_Summary - 可用性：2.1.0 返回栅格内容的文本摘要。
- ST_TPI - 可用性：2.1.0 返回一个栅算出的地形位置指数（Topographic Position Index）的栅格。
- ST_TRI - 可用性：2.1.0 返回具有栅算的地形栅固性指数的栅格。
- ST_Tile - 可用性：2.1.0 返回根据栅出栅格的所需栅度分割栅入栅格而栅生的一栅栅格。
- ST_Touches - 可用性：2.1.0 如果栅格 rastA 和 rastB 至少有一个共同点但它栅的内部不相交，则返回 true。

- ST_Union - 可用性：2.1.0 引入了 ST_Union(rast, unionarg) 变体。将一组栅格切片的并集返回由 1 个或多个波段构成的单个栅格。
- ST_Within - 可用性：2.1.0 如果栅格 rastA 中没有点位于栅格 rastB 的外部且 rastA 的内部至少有一个点位于 rastB 的内部，则返回 true。
- ST_WorldToRasterCoord - 可用性：2.1.0 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。
- UpdateRasterSRID - 可用性：2.1.0 更改用给定的列和表中所有栅格的 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 的功能增强 2.1

- ST_AddBand - 增强：2.1.0 添加了对 addbandarg 的支持。返回一个栅格，其中在给定索引位置添加了给定类型的新波段和给定初始值。如果未指定索引，则将添加到末尾。
- ST_AddBand - 增强：2.1.0 添加了对新的 out-db 波段的支持。返回一个栅格，其中在给定索引位置添加了给定类型的新波段和给定初始值。如果未指定索引，则将添加到末尾。
- ST_AsBinary/ST_AsWKB - 增强：2.1.0 添加 outasin 返回栅格的熟知的二进制 (WKB) 表示形式。
- ST_AsGML - 增强：支持 GML 3 引入了 2.1.0 id 支持。将几何形作为 GML 版本 2 或 3 元素返回。
- ST_Aspect - 增强：2.1.0 使用 ST_MapAlgebra() 并添加可选的 interpolate_nodata 函数参数返回高程栅格波段的坡向（默认以度数为单位）。对于分析地形很有用。
- ST_Boundary - 增强：引入了 2.1.0 三角函数支持返回几何形的边界。
- ST_Clip - 增强：2.1.0 用 C 重写返回由输入几何形裁剪的栅格。如果未指定波段号，则处理所有波段。如果未指定 crop 或其为 TRUE，则输出栅格将被裁剪。如果 touched 设置为 TRUE，则包括接触到的像素；否则，仅当像素的中心位于几何形内时，才会包括像素。
- ST_DWithin - 增强功能：2.1.0 版本提升了地理（Geography）类型的速度。有关详细信息，请参阅 Making Geography faster。确定两个几何形是否在给定距离内
- ST_DWithin - 增强：2.1.0 引入了对弯曲几何形状的支持。确定两个几何形是否在给定距离内
- ST_Distance - 增强功能：2.1.0 版本提升了地理（Geography）类型的速度。有关详细信息，请参阅 Making Geography faster。返回两个几何或地理之间的距离。
- ST_Distance - 增强：2.1.0 - 引入了对弯曲几何形状的支持。返回两个几何或地理之间的距离。
- ST_Distinct4ma - 增强：2.1.0 添加格式 2 栅格处理函数，用于计算域中唯一像素的数量。
- ST_DumpPoints - 增强：2.1.0 速度更快。重新编写原生 C 语言。返回几何形中坐标的一组 geometry_dump 行。
- ST_HillShade - 增强：2.1.0 使用 ST_MapAlgebra() 并添加可选的 interpolate_nodata 函数参数使用提供的方位角、高度、亮度和比例输入返回高程栅格的假设照明。
- ST_MakeValid - 增强：2.1.0 添加了对几何集合和多点的支持。能够在不丢失维点的情况下使无效几何体有效。
- ST_Max4ma - 增强：2.1.0 添加格式 2 计算域中最大像素的栅格处理函数。
- ST_Mean4ma - 增强：2.1.0 添加格式 2 计算域中平均像素的栅格处理函数。
- ST_Min4ma - 增强：2.1.0 添加格式 2 计算域中最小像素的栅格处理函数。
- ST_PixelAsPolygons - 增强：2.1.0 添加了 except_nodata_value 可选参数。返回包含栅格的每个像素的多边形几何形以及每个像素的值、X 和 Y 栅格坐标。

- 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)。

13.12.14 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。返回 A 和 B 界框之的 2D 距离。
- <-> - 可用性：2.0.0 -- 弱 KNN 基于几何象心距离来找最近，而非真距离。点来果是准确的，其他型是近似的。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。
- 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。
- Polygonize - 可用性：2.0.0 找并注册由拓扑定的所有面。
- 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 返回由⊠入几何⊠形裁剪的⊠格。如果未指定波段号，⊠⊠理所有波段。如果未指定 crop 或其⊠⊠ TRUE，⊠⊠出⊠格将被裁剪。如果 touched ⊠置⊠ TRUE，⊠包括接触到的像素；否⊠，⊠当像素的中心位于几何⊠形内⊠，才会包括⊠像素。
- 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 except_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 - 可用性：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 Returns the number of faces in a PolyhedralSurface or TIN.
- 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 Returns the Nth face of a PolyhedralSurface or TIN.
- 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 返回指定☒格或☒格覆盖中某个波段的☒☒摘要，包括像元数量（count）、☒和（sum）、均☒（mean）、☒准差（stddev）、最小☒（min）和最大☒（max）。若未指定波段，默☒使用第 1 波段。
- 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 使用容差将☒串添加到☒有拓扑中，并在可能的情况下拆分☒或面。
- 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。
- 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 - Enhanced: 2.0 新增☒ hasZ 的支持。☒建一个新的拓扑架☒并将其注册到 topology.topology 表中。
- 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 。Returns a geometry from GML. Alias for ST_GeomFromGML.
- ST_GMLToSQL - 增⍰：2.0.0 引入了多面体曲面支持和 TIN 支持。Returns a geometry from GML. Alias for ST_GeomFromGML.
- 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⊠ validate-topology_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 - Changed: 2.0.0. Retained for backward compatibility after geometry_columns became a view of the system catalogs. 从空⊠表中移除 geometry（几何）列。
- DropGeometryTable - Changed: 2.0.0. Retained for backward compatibility after geometry_columns became a view of the system catalogs. ⊠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 - Changed: 2.0.0 Inputs to this function can not be unknown -- must be geometry. Constructs such as ST_AsBinary('POINT(1 2)') are no longer valid and you will get an n ST_AsBinary(unknown) is not unique error. Code like that needs to be changed to ST_AsBinary('POINT(1 2)'::geometry);. If that is not possible, then install legacy.sql. 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。Returns the last point of a LineString, CircularLineString, or NURBSCurve.
- 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。Returns the first point of a LineString, CircularLineString, or NURBSCurve.

13.12.15 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 ⌧⌧一个几何⌧形是否完全位于⌧一个几何⌧形的指定距离内
- 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+ Returns a geometry from GML. Alias for ST_GeomFromGML
- 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。

13.12.16 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 - 可用性：1.4 返回⊠明几何⊠形是否有效或无效原因的文本。
- ST_LineCrossingDirection - 可用性：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。

13.12.17 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 ）：

Before editing or building the manual, read the repository's documentation development guide. It covers the executable example, generated figure, validation, HTML, and PDF workflows used by the manual.

1. 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☒☒。

Chapter 15

Credits and Acknowledgments

PostGIS is built and sustained by a worldwide community. This chapter records the people, organizations, funders, and upstream projects that have contributed code, documentation, infrastructure, testing, support, and funding.

15.1 Project Steering Committee

The PostGIS Project Steering Committee (PSC) coordinates the general direction, release cycles, documentation, and outreach efforts for the PostGIS project. In addition the PSC provides general user support, accepts and approves patches from the general PostGIS community and votes on miscellaneous issues involving PostGIS such as developer commit access, new PSC members or significant API changes.

Regina Obe CI and website maintenance, Windows production and experimental builds, documentation, alignment of PostGIS with PostgreSQL releases, X3D support, management functions.

Darafei Praliaskouski Index improvements, bug fixing and geometry/geography function improvements, SFCGAL, raster, GitHub curation, and ci maintenance.

Paul Ramsey (Chair) Co-founder of PostGIS project. General bug fixing, geography support, geography and geometry index support (2D, 3D, nD index and anything spatial index), underlying geometry internal structures, GEOS functionality integration and alignment with GEOS releases, alignment of PostGIS with PostgreSQL releases, loader/dumper, and Shapefile GUI loader.

Sandro Santilli Bug fixes and maintenance, ci maintenance, git mirror management, management functions, integration of new GEOS functionality and alignment with GEOS releases, topology support, and raster framework and low level API functions.

15.2 Core Contributors Present

Nicklas Avén Distance function enhancements (including 3D distance and relationship functions) and additions, Tiny WKB (TWKB) output format and general user support

Loïc Bartoletti SFCGAL enhancements and maintenance and ci support

Dan Baston Geometry clustering function additions, other geometry algorithm enhancements, GEOS enhancements and general user support

Martin Davis GEOS enhancements and documentation

Björn Harrtell MapBox Vector Tile, GeoBuf, and Flatgeobuf functions. Gitea testing and GitLab experimentation.

Aliaksandr Kalenik Geometry Processing, PostgreSQL gist, general bug fixing

Laurenţiu Nicola Bug fixes, maintenance, stability, and performance

15.3 Core Contributors Past

Raúl Marín Rodríguez Prior PSC Member, MVT support, Bug fixing, Performance and stability improvements, GitHub curation, alignment of PostGIS with PostgreSQL releases

Bborie Park Prior PSC Member. Raster development, integration with GDAL, raster loader, user support, general bug fixing, testing on various OS (Slackware, Mac, Windows, and more)

Mark Cave-Ayland Prior PSC Member. Coordinated bug fixing and maintenance effort, spatial index selectivity and binding, loader/dumper, and Shapefile GUI Loader, integration of new and new function enhancements.

Jorge Arévalo Raster development, GDAL driver support, loader

Olivier Courtin (Emeritus) Input/output XML (KML,GML)/GeoJSON functions, 3D support and bug fixes.

Chris Hodgson Prior PSC Member. General development, site and buildbot maintenance, OSGeo incubation management

Mateusz Loskot CMake support for PostGIS, built original raster loader in python and low level raster API functions

Kevin Neufeld Prior PSC Member. Documentation and documentation support tools, buildbot maintenance, advanced user support on PostGIS newsgroup, and PostGIS maintenance function enhancements.

Dave Blasby The original developer/Co-founder of PostGIS. Dave wrote the server side objects, index bindings, and many of the server side analytical functions.

Jeff Lounsbury Original development of the Shapefile loader/dumper.

Mark Leslie Ongoing maintenance and development of core functions. Enhanced curve support. Shapefile GUI loader.

Pierre Racine Architect of PostGIS raster implementation. Raster overall architecture, prototyping, programming support

David Zwarg Raster development (mostly map algebra analytic functions)

15.4 PostGIS Raster Origins

PostGIS raster discussions date to 2002 and included early PGCHIP experiments. Xing Lin developed the **geotiff2pgraster** PGRaster prototype during Google Summer of Code 2007. The WKT Raster project began in 2008, Jorge Arévalo developed the first GDAL driver prototype during Google Summer of Code 2009, and the 2010 FOSS4G code sprint agreed to make WKT Raster the PostGIS raster type. Raster support shipped as part of PostGIS 2.0 in 2012.

The early implementation team included Pierre Racine, Mateusz Loskot, Sandro Santilli, Jorge Arévalo, Bborie Park, David Zwarg, and Regina Obe. Their work established the raster type, loader, GDAL integration, raster/vector operations, map algebra, documentation, and testing that became the current raster subsystem.

Other recorded contributions include Jeff Adams's early sprint work, Qing Liu's raster distance analysis, Maurício Carvalho Mathias de Paulo's QGIS raster plugin and related paper with Lubia Vinhas, and Ignacio Brodin's gvSIG raster plugin and presentation with Jorge Sanz. Bryce L. Nordgren created early raster coverage and seamless-architecture diagrams. Pierre Racine wrote the original raster/vector tutorial and created its walkthrough and early API diagrams.

The historical project planning record credits Steve Cumming and Université Laval with funding Pierre Racine's work from June 2008 through July 2012; Martin Daly and Cadcorp with funding about six months of Mateusz Loskot's work through June 2009; Tyler Erickson and Michigan Tech Research Institute with a $1,200 USD contribution; Miguel Lizondo and Deimos Space with support for about two years of Jorge Arévalo's work through June 2011; and the Center for Vectorborne Diseases at UC Davis with funding Bborie Park's work from March 2010 through July 2012.

15.5 Other Contributors

Individual Contributors

@amwps290 (GitHub user)
@kannes (GitHub user: Hannes)
@sunhm89 (GitHub user)
@wewindy (GitHub user: Chen)
Adrien Berchet
Alex Bodnaru
Alex Mayrhofer
Andrea Peri
Andreas Forø Tollefsen
Andreas Neumann
Andreas Schild
Andrew Gierth
Andrew Harvey
Anne Ghisla
Antoine Bajolet
Ariel Mashraki
Arno Renevier
Arthur Bazin
Arthur Lesuisse
Artur Zakirov
Ayo Adesugba
Barbara Phillipot
Baris Ergun
Bas Couwenberg
Ben Jubb
Benjamin Morel
Bernd Reiß
Bernhard Reiter
Bill Mill
Björn Esser
Brian Hamlin
Bruce Rindahl
Bruno Wolff III
Bryce L. Nordgren
Carl Anderson
Carlos López Quintanilla
Charles Karney
Charlie Savage
Chris Mayo
Christian Schroeder
Christoph Berg
Christoph Moench-Tegeder
Dan Faudemer
Dan Levy
Dane Springmeyer
Daniel Nylander
Dapeng Wang
Daryl Herzmann
Dave Fuhry
David G. Johnston
David Garnier
David Skea
David Techer
Dennis Tighe
Denys Kovshun
Devrim Gündüz
Dian M Fay
Dmitry Vasilyev
Edouard Choinière
Eduin Carrillo
Egor Ivkov
Eric Ridge
Esteban Zimanyi
Eugene Antimirov
Florian Weimer
François Bonzon
Frank Warmerdam
George Silva
Gerald Fenoy
Gino Lucrezi
Giovanni Zezza
Grant Hutchins
Greg Troxel
Guillaume Lelarge
Giuseppe Broccolo
Han Wang
Hannes Janetzek
Hans Lemuet
Haribabu Kommi
Harrissou Sant-anna
Havard Tveite
Ignacio Brodin
IIDA Tetsushi
Ingvild Nystuen
Jackie Leng
James Addison
James Marca
Jan Katins
Jan Tojnar
Jason Smith
Jean Felder
Jeff Adams
Jelte Fennema
Jim Jones
Joe Conway
Jonne Savolainen
Jorge Gustavo Rocha
Jorge Sanz
Jose Carlos Martinez Llari
Jostein Leira
Jörg Habenicht
Juan Sebastian Jimenez Aldana
Juergen E. Fischer
Julien Monticolo
Julien Rouhaud
Kashif Rasul
Klaus Foerster
Kris Jurka
Kristian Thy
Lars Aksel Opsahl
Laurenţiu Nicola
Laurenz Albe
Lars Roessiger
Lauri Kajan
Leo Hsu
Loic Dachary
Louis Descoteaux
Luca Delucchi
Luca S. Percich
Lucas C. Villa Real
Ludovic Hirlimann
Lubia Vinhas
Maksim Korotkov
Manuel Morillo
Maria Arias de Reyna
Marc Ducobu
Mark Sondheim
Markus Schaber
Maurício
Maxim Ko
Maxim Ko
Maxime G
Maxime v
Maxime S
Megan Ma
Mehmet I
Michael F
Michal Pá
Mike Toev
Mitchell H
Nathan W
Nathaniel
Nicolas R
Nikita Shu
Nikolai Be
Norman B
Norman V
Patricia T
Paul Norm
Paulo Ces
Pavel Sele
Pawel Ost
Peter Ngu
Pierre Car
Qing Liu
Rafal Mag
Ralph Mas
Rémi Cura
Richard G
Robert Co
Rodrigo S
Roger Cre
Ron Maye
Sam Peter
Sarath Ku
Sergei Sh
Sergey Fe
Shinichi S
Shoaib Bu
Silvio Gro
Simon Gre
Stefan Co
Steffen M
Stepan Ku
Stephen F
Stephen W
Steven Ot
Svyatoslav
Talha Rizv
Temuri Do
Teramoto
Thomas M
Thomas P
Tobias Bu
Tom Glanc
Tom van T
Victor Col
Vincent Bi
Vincent M
Vincent Pi
Volf Tomá
Xing Lin

Corporate Sponsors These are corporate entities that have contributed developer time, hosting, or direct monetary funding to the PostGIS project. In alphabetical order:

- Aiven
- Arrival 3D
- Associazione Italiana per l'Informazione Geografica Libera (GFOSS.it)
- AusVet
- Avencia
- Azavea
- Boundless
- Cadcorp
- Camptocamp
- Carto
- Crunchy Data
- City of Boston (DND)
- City of Helsinki
- Clever Elephant Solutions
- Cooperativa Alveo
- Deimos Space
- Faunalia
- Geographic Data BC
- Google
- HighGo
- Hunter Systems Group
- IITM Pravartak Security Team
- INIA-CSIC
- ISciences, LLC
- Jirotech (formerly LISAsoft)
- Koordinates
- Kontur
- Lidwala Consulting Engineers
- Logical Tracking & Tracing International AG
- Maponics
- Maumap
- Michigan Tech Research Institute
- Natural Resources Canada
- Norwegian Forest and Landscape Institute
- Norwegian Institute of Bioeconomy Research (NIBIO)
- OSGeo
- Oslandia
- Palantir Technologies
- Paragon Corporation
- PlanetScale
- Postgres Pro

- R3 GIS
- Refractions Research
- Regione Toscana - SITA
- Safe Software
- Sirius Corporation plc
- Stadt Uster
- UC Davis Center for Vectorborne Diseases
- Université Laval
- U.S. Census Bureau
- U.S. Department of State (HIU)
- Zonar Systems

Crowd Funding Campaigns Crowd funding campaigns are campaigns we run to get badly wanted features funded that can service a large number of people. Each campaign is specifically focused on a particular feature or set of features. Each sponsor chips in a small fraction of the needed funding and with enough people/organizations contributing, we have the funds to pay for the work that will help many. If you have an idea for a feature you think many others would be willing to co-fund, please post to the PostGIS newsgroup your thoughts and together we can make it happen.

PostGIS has experimented with community fundraising for features in the past. We used PledgeBank and we got two successful campaigns out of it.

postgistopology - 10 plus sponsors each contributed $250 USD to build toTopoGeometry function and beef up topology support in 2.0.0. It happened.

postgis64windows - 20 someodd sponsors each contributed $100 USD to pay for the work needed to work out PostGIS 64-bit issues on windows. It happened.

External Libraries and Platforms PostgreSQL, the extensible database system on which PostGIS is built. PostgreSQL provides the SQL engine, transactions, query planner, extension framework, and index access methods used by PostGIS.

The GEOS geometry operations library.

The PROJ coordinate transformation and geodesic calculation library.

The GDAL Geospatial Data Abstraction Library used to power much of the raster functionality introduced in PostGIS 2. In kind, improvements needed in GDAL to support PostGIS are contributed back to the GDAL project.

LibXML2 for XML-based spatial formats, and JSON-C for GeoJSON input.

The GNU Multiple Precision Arithmetic Library (GMP) for precision-sensitive topology operations.

protobuf-c for Mapbox Vector Tile and Geobuf output.

The iconv API, supplied by the C library or GNU libiconv, for character encoding conversion in the shapefile loader.

SFCGAL for advanced 2D and 3D functions, together with the CGAL algorithms that SFCGAL exposes.

Bundled Libraries and Source Code PostGIS ships selected source from FlatGeobuf and FlatBuffers to support FlatGeobuf input and output.

Wagyu and Mapbox Geometry provide polygon clipping and validation for Mapbox Vector Tile output.

Ryu provides round-trip-safe floating-point formatting, and uthash provides in-process hash tables.

The shapefile loader and dumper include code derived from ShapeLib, originally by Frank Warmerdam and Even Rouault. They also include public-domain `getopt` code from AT&T.

`liblwgeom/lookup3.c` contains the public-domain `lookup3` hash function by Bob Jenkins.

Appendix A

版本说明

A.1 PostGIS 3.7.0

This version requires GEOS 3.10+, PostgreSQL 14-19beta3, Proj 6.1+, and libgmp. To take advantage of all postgis extension features, GEOS 3.15+ is needed. To take advantage of all postgis_sfcgal extension features SFCGAL 2.3+ is needed. Python 3 is a new requirement to build the documentation.

Visual examples in the docs are off by default. To build docs with them, configure with `--with-visual-examples` and make sure you have a running PostgreSQL instance; they are then generated automatically when building the docs (or explicitly with `make -C regress visual-examples`).

A.1.1 重大变化

#5688, [topology] topology building functions now interpret tolerance 0 (old default) to really mean 0 and want -1 (new default) to mean "use topology precision" (Sandro Santilli)

#6024, Remove support for PostgreSQL 12 and 13 (Regina Obe)

#6040, ST_GeoHash round trip through ST_GeomFromGeoHash (Paul Ramsey)

#6053, [address_standardizer] Extension removed and moved to https://github.com/postgis/address_standa (Paul Ramsey)

#6052, [tiger_geocoder] Extension removed and moved to https://git.osgeo.org/postgis/postgis_tiger_geoco (Regina Obe)

#6079, Remove support for GEOS < 3.10 (Sandro Santilli)

#1793, Remove ST_MemCollect aggregate (Darafei Praliaskouski)

GH-848, CG_AlphaShape now returns a MultiPolygon (Jean Felder)

[docs] Python 3 and an SVG rasterizer (rsvg-convert preferred, GraphicsMagick or ImageMagick supported) now required for doc building (Darafei Praliaskouski)

A.1.2 新特性

#1124, #2935, #3033, #4658, #4659, [loader] Rework loader arguments: shp2pgsql can create UNLOGGED tables and choose the feature id column name, shp2pgsql --drop-table can emit DROP TABLE before prepare output, raster2pgsql/shp2pgsql expose loader actions as long options, both loaders accept long aliases for existing options, and

-if-not-exists makes creation actions idempotent (Darafei Praliaskouski)

#4208, Add single-geometry variants of ST_MaxDistance and ST_LongestLine (Darafei Praliaskouski)

[topology] FindVertexSegmentPairsBelowDistance function (Sandro Santilli)

ST_CoverageEdges, returns MultiLinestring of distinct shared edges in polygonal coverage (Paul Ramsey)

ST_MinimumSpanningTree, window function to calculate MST (Paul Ramsey)

#5993, [topology] Add max_edges parameter to TopoGeo_AddLinestring (Sandro Santilli)

#6001, support MultiLineString in ST_MakeLine (Paul Ramsey)

#2858, ST_MMin and ST_MMax (Paul Ramsey)

#5941, [raster] Support GDT_Float16 pixel type (Darafei Praliaskouski)

#5992, Optimize GiST index for repeated edge subdivision in topology (Darafei Praliaskouski)

#5702, Allow the compiler to detect the parallelism -flto=auto (Darafei Praliaskouski)

#4798, ST_AsGeoJSON warns about duplicate property keys (Darafei Praliaskouski)

#4222, [raster] ST_DumpAsPolygons honours PostgreSQL interrupts (Darafei Praliaskouski)

#5889, [topology] Include representative locations in topology build errors (Darafei Praliaskouski)

#2614, Use GEOSPreparedDistance and caching to accelerate ST_DWithin (Paul Ramsey)

GH-839, ST_Multi support for TIN and surfaces (Loïc Bartoletti)

#4398, Add ST_CatmullSmoothing smoothes but retains original vertices (Paul Ramsey)

#4560, ST_3DInterpolatePoint for M interpolation from XYZ inputs (Paul Ramsey)

#1291, ST_MakePolygon support for curved rings (Darafei Praliaskouski)

GT-270, Add NURBSCurve (Loïc Bartoletti)

#2863, Add ST_XSize, ST_YSize, ST_ZSize, and ST_MSize dimension helpers (Darafei Praliaskouski)

GT-806, #6124 --with-visual-examples opt-in configure switch to build dynamic images (Regina Obe)

A.1.3 增☒功能

#2045, pgsql2shp query dumps no longer require temporary table privileges (Darafei Praliaskouski)

#1577, [loader] Report unsupported .zip archive input before trying to open shapefile sidecar files (Darafei Praliaskouski)

#3158, Move geometry and geography typmod bit helpers out of the public liblwgeom header (Darafei Praliaskouski)

#3776, [raster] Keep distinct 32-bit integer pixel values separate in ST_DumpAsPolygons (Darafei Praliaskouski)

#4678, [raster] Move st_sum4ma neighborhood callback to C (Darafei Praliaskouski)

#2116, [raster] Add ST_Value nearest-neighbor boundary options (Darafei Praliaskouski)

#2804, #4315, [raster] Support two-argument ST_MapAlgebra callbacks and pass callback call data as actual arguments (Darafei Praliaskouski)

#6062, [topology] Stop using recursive snapping, for improved robustness (Sandro Santilli)

#6065, Improved winding order computation robustness for rings having collapsed elements (Sandro Santilli)

#2137, [raster] Reuse matching source raster overviews in raster2pgsql -l (Darafei Praliaskouski)

#4749, Use point-in-polygon predicate fast paths for point-only GeometryCollections (Darafei Praliaskouski)

#2807, [raster] Preserve missing NODATA values in ST_MapAlgebra outputs (Darafei Praliaskouski)

#2832, [raster] Avoid padding single-tile raster2pgsql overviews when padding is not requested (Darafei Praliaskouski)

#2623, Generate a pgsql2shp GID field when exporting geometry-only shapefiles (Darafei Praliaskouski)

#2850, Preserve pgsql2shp numeric precision and scale in DBF metadata (Darafei Praliaskouski)

Build PostgreSQL extension modules with `-fno-semantic-interposition` when available so LTO can optimize same-DSO calls to exported PostGIS entry points directly instead of routing through the module PLT; in a local PostgreSQL 18 `pgbench` microbenchmark over 100k four-point groups this reduced `ST_Collect` aggregate latency by about 4% (Darafei Praliaskouski)

#5973, Install PostGIS utilities under `pg_config --bindir` by default and document `pg_config`-derived install paths more accurately (Darafei Praliaskouski)

#2808, Clarify postgis_full_version() output when installed component scripts are newer than core scripts (Darafei Praliaskouski)

#2838, Emit valid X3D Coordinate nodes for 2D polygon output (Darafei Praliaskouski)

GH-899, [raster] Honor PostgreSQL interrupts in long-running GDAL progress callbacks (Darafei Praliaskouski)

#6073, #6074, Stabilize geography LRS endpoint interpolation and pole centroid regressions on 32-bit systems (Darafei Praliaskouski)

#2583, Preserve Z/ZM dimensional qualifiers in ST_AsEWKT output for TIN (Darafei Praliaskouski)

#4616, Detect pgsql2shp query keywords case-insensitively (Darafei Praliaskouski)

#1986, ST_GeomFromGML support for GML ArcString and curved polygon rings (Darafei Praliaskouski)

#4749, Use point-in-polygon predicate fast paths for point-only GeometryCollections, including ST_Disjoint (Darafei Praliaskouski)

#3676, Mark geometry and geography btree comparison support functions as leakproof (Darafei Praliaskouski)

GH-875, Allow ST_OrderingEquals joins to use hash and merge join plans (Darafei Praliaskouski)

#1168, [raster] Support curved geometry inputs in ST_AsRaster (Darafei Praliaskouski)

#1416, ST_AsX3D outputs NURBSCURVE as X3D NurbsCurve and linearizes other curve geometries (Darafei Praliaskouski)

#3705, ST_AsX3D outputs valid coordinate nodes for 2D multi-geometries and GeometryCollections (Darafei Praliaskouski)

GT-484, Use column-level SELECT privileges for metadata views (Arthur Bazin)

#1408, #2386, #2557, #2805, #2898, #3743, #4323, #4332, #4385, #5109, #5532, #5593, #5638, #5645, #5950, #6022, #6082, GH-1161, GT-584, Numerous doc enhancements (Darafei Praliaskouski)

A.1.4 Bug Fixes

GH-1175, Guard missing operands before reading node tags in spatial selectivity estimation (Maksim Korotkov)

#6118, Reject empty geometry type modifiers without an out-of-bounds read (Egor Ivkov)

Fix WKB and TWKB parser resource exhaustion on malformed input (Darafei Praliaskouski)

#6038, Avoid stale relation lookups in geometry_columns after topology objects are dropped (Darafei Praliaskouski)

#5487, Improve error detail for repeated upgrades after an incomplete deprecated-function cleanup (Darafei Praliaskouski)

#5975, Initialize skipped union-find cluster ids deterministically (Darafei Praliaskouski)

#5904, [topology] Avoid MinGW format warnings when logging topology ids (Darafei Praliaskouski)

GH-888, [sfcgal] Avoid stale detoasted geometry access in CG_Visibility empty-input handling for SFCGAL < 2.2 (Darafei Praliaskouski)

#3179, Mark truncated libpgcommon PostgreSQL messages instead of silently dropping the tail (Darafei Praliaskouski)

#6003, Avoid size_t formats in liblwgeom logging/error wrappers for MinGW builds (Darafei Praliaskouski)

GH-886, ST_ClusterRelateWin validates NULL relate matrices before detoasting them (Darafei Praliaskouski)

GH-889, ST_Relate pattern inversion avoids out-of-bounds access on short DE-9IM patterns (Darafei Praliaskouski)

GH-890, Avoid GIDX BRIN summary merges across mismatched dimensions (Darafei Praliaskouski)

#5959, Prevent histogram target overflow when analysing massive tables (Darafei Praliaskouski)

#6080, Avoid out-of-bounds hex WKB lookup for high-bit input bytes (Darafei Praliaskouski)

#4828, geometry_columns handles NOT VALID SRID checks without errors (Darafei Praliaskouski)

#6048, [raster] ST_Clip no longer crashes when clipping sparse band selections (Darafei Praliaskouski)

GH-894, [raster] Fix Float16 import, mixed nodata warping, band replacement cleanup, GDAL credential redaction, and Python GDAL compatibility (Darafei Praliaskouski)