

**PostGIS 2.5.0 Manual**

---

# Contents

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Comitê Diretor do Projeto	1
1.2	Contribuidores Núcleo Atuais	1
1.3	Contribuidores Núcleo Passado	2
1.4	Outros Contribuidores	2
1.5	Mais informações	3
<b>2</b>	<b>Instalação do PostGIS</b>	<b>4</b>
2.1	Versão Reduzida	4
2.2	Instalando pacotes requeridos	5
2.3	Obtendo o Fonte	7
2.4	Compilando e instalando da fonte: detalhado	7
2.4.1	Configuração	8
2.4.2	Construindo	9
2.4.3	Construindo extensões PostGIS e implantando-as	9
2.4.4	Testando	12
2.4.5	Instalação	21
2.5	Criando uma base de dados espacial usando EXTENSÕES	21
2.6	Create a spatially-enabled database without using extensions	22
2.7	Instalando e usando o padronizador de endereço	23
2.7.1	Instalando Regex::Montar	23
2.8	Instalando, Atualizando o Tiger Geocoder e carregando dados	23
2.8.1	Tiger Geocoder ativando seu banco de dados PostGIS: Usando Extensão	24
2.8.1.1	Convertendo uma Instalação Tiger Geocoder Regular para Modelo de Extensão	26
2.8.2	Tiger Geocoder Ativando seu banco de dados PostGIS: Sem Utilizar Extensões	26
2.8.3	Usando Padronizador de Endereço com Tiger Geocoder	27
2.8.4	Carregando Dados Tiger	27
2.8.5	Atualizando sua Instalação Tiger Geocoder	28
2.9	Criar um banco de dados espacialmente ativado de um template	28
2.10	Atualizando	29

2.10.1	Atualização flexível . . . . .	29
2.10.1.1	Atualização Soft Pre 9.1+ ou sem extensões . . . . .	29
2.10.1.2	Atualização flexível 9.1+ usando extensões . . . . .	29
2.10.2	Atualização rígida . . . . .	30
2.11	Problemas comuns durante a instalação . . . . .	31
2.12	Carregador/Dumper . . . . .	32
<b>3</b>	<b>Perguntas frequentes PostGIS</b>	<b>33</b>
<b>4</b>	<b>Usando o PostGIS: Gerenciamento de dados e consultas</b>	<b>38</b>
4.1	Objetos GIS . . . . .	38
4.1.1	OpenGIS WKB e WKT . . . . .	38
4.1.2	PostGIS EWKB, EWKT e Formas Canônicas . . . . .	39
4.1.3	SQL-MM Part 3 . . . . .	40
4.2	Tipo de geografia POstGIS . . . . .	40
4.2.1	Geografia Básica . . . . .	41
4.2.2	Quando usar o tipo de dados Geografia sobre os dados Geometria . . . . .	43
4.2.3	FAQ de Geografia Avançada . . . . .	43
4.3	Usando os Padrões OpenGIS . . . . .	44
4.3.1	The SPATIAL_REF_SYS Table and Spatial Reference Systems . . . . .	44
4.3.2	A GEOMETRY_COLUMNS VIEW . . . . .	45
4.3.3	Criando uma Tabela Espacial . . . . .	46
4.3.4	Registrando manualmente as colunas geométricas em geometry_columns . . . . .	46
4.3.5	Assegurando a confirmação de geometrias OpenGIS . . . . .	48
4.3.6	Dimensionalidade estendida 9 Modelo de Interseção (DE-9IM) . . . . .	52
4.3.6.1	Teoria . . . . .	54
4.4	Carregando dados GIS (Vector) . . . . .	58
4.4.1	Carregando Dados Usando SQL . . . . .	58
4.4.2	shp2pgsql: Using the ESRI Shapefile Loader . . . . .	58
4.5	Recuperando dados GIS . . . . .	60
4.5.1	Usando SQL para recuperar dados . . . . .	60
4.5.2	Usando o Dumper . . . . .	61
4.6	Construindo índices . . . . .	61
4.6.1	Índices GiST . . . . .	62
4.6.2	BRIN Indexes . . . . .	62
4.6.3	SP-GiST Indexes . . . . .	63
4.6.4	Usando Índices . . . . .	64
4.7	Consultas Complexas . . . . .	64
4.7.1	Tirando vantagem dos índices . . . . .	65
4.7.2	Exemplos de SQL espacial . . . . .	65

---

<b>5</b>	<b>Gerência de dados raster, pesquisas e aplicações</b>	<b>68</b>
5.1	Carregando e criando dados matriciais . . . . .	68
5.1.1	Usando o raster2pgsql para carregar dados matricias . . . . .	68
5.1.2	Criando rasters utilizando as funções rasters do PostGIS . . . . .	72
5.2	Catálogos Raster . . . . .	72
5.2.1	Catálogo de Colunas Raster . . . . .	73
5.2.2	Panoramas Raster . . . . .	74
5.3	Construindo Aplicações Personalizadas com o PostGIS Raster . . . . .	74
5.3.1	PHP Exemplo Outputting usando ST_AsPNG em consenso co outras funções raster . . . . .	75
5.3.2	ASP.NET C# Exemplo gerado usando ST_AsPNG em consenso com outras funções raster . . . . .	75
5.3.3	O app console Java que gera a consulta raster como arquivo de imagem . . . . .	77
5.3.4	Use PLPython para excluir imagens via SQL . . . . .	78
5.3.5	Rasters de saída com PSQL . . . . .	79
<b>6</b>	<b>Usando a Geometria do PostGIS: Criando aplicativos</b>	<b>80</b>
6.1	Usando o MapServer . . . . .	80
6.1.1	Uso Básico . . . . .	80
6.1.2	Perguntas Frequentes . . . . .	81
6.1.3	Uso Avançado . . . . .	82
6.1.4	Exemplos . . . . .	83
6.2	Clientes Java (JDBC) . . . . .	84
6.3	Clientes C (libpq) . . . . .	86
6.3.1	Cursors de Texto . . . . .	86
6.3.2	Cursors Binários . . . . .	86
<b>7</b>	<b>Dicas de desempenho</b>	<b>87</b>
7.1	Pequenas tabelas de grandes geometrias . . . . .	87
7.1.1	Descrição do problema . . . . .	87
7.1.2	Soluções . . . . .	87
7.2	CLUSTERizando índices geométricos . . . . .	88
7.3	Evitando conversão de dimensões . . . . .	88
7.4	Tunando sua configuração . . . . .	89
7.4.1	Início . . . . .	89
7.4.2	Runtime . . . . .	89

---

<b>8</b>	<b>Referência do PostGIS</b>	<b>91</b>
8.1	PostgreSQL PostGIS Geometry/Geography/Box Types	91
8.1.1	box2d	91
8.1.2	box3d	91
8.1.3	geometry	92
8.1.4	geometry_dump	92
8.1.5	geografia	92
8.2	Grandes Variáveis Unificadas Personalizadas do PostGIS (GUCs)	93
8.2.1	postgis.backend	93
8.2.2	postgis.gdal_datapath	93
8.2.3	postgis.gdal_enabled_drivers	94
8.2.4	postgis.enable_outdb_rasters	96
8.3	Funções de Gestão	96
8.3.1	AddGeometryColumn	96
8.3.2	DropGeometryColumn	98
8.3.3	DropGeometryTable	99
8.3.4	PostGIS_Extensions_Upgrade	100
8.3.5	PostGIS_Full_Version	101
8.3.6	PostGIS_GEOS_Version	101
8.3.7	PostGIS_Liblwgeom_Version	102
8.3.8	PostGIS_LibXML_Version	102
8.3.9	PostGIS_Lib_Build_Date	103
8.3.10	PostGIS_Lib_Version	103
8.3.11	PostGIS_PROJ_Version	104
8.3.12	PostGIS_Scripts_Build_Date	104
8.3.13	PostGIS_Scripts_Installed	105
8.3.14	PostGIS_Scripts_Released	106
8.3.15	PostGIS_Version	106
8.3.16	Populate_Geometry_Columns	107
8.3.17	UpdateGeometrySRID	108
8.4	Construtores de geometria	109
8.4.1	ST_BdPolyFromText	109
8.4.2	ST_BdMPolyFromText	110
8.4.3	ST_Box2dFromGeoHash	111
8.4.4	ST_GeogFromText	111
8.4.5	ST_GeographyFromText	112
8.4.6	ST_GeogFromWKB	112
8.4.7	ST_GeomFromTWKB	113
8.4.8	ST_GeomCollFromText	114

8.4.9	ST_GeomFromEWKB	114
8.4.10	ST_GeomFromEWKT	116
8.4.11	ST_GeometryFromText	117
8.4.12	ST_GeomFromGeoHash	118
8.4.13	ST_GeomFromGML	118
8.4.14	ST_GeomFromGeoJSON	121
8.4.15	ST_GeomFromKML	122
8.4.16	ST_GMLToSQL	123
8.4.17	ST_GeomFromText	123
8.4.18	ST_GeomFromWKB	125
8.4.19	ST_LineFromEncodedPolyline	126
8.4.20	ST_LineFromMultiPoint	127
8.4.21	ST_LineFromText	127
8.4.22	ST_LineFromWKB	128
8.4.23	ST_LinestringFromWKB	129
8.4.24	ST_MakeBox2D	130
8.4.25	ST_3DMakeBox	130
8.4.26	ST_MakeLine	131
8.4.27	ST_MakeEnvelope	133
8.4.28	ST_MakePolygon	133
8.4.29	ST_MakePoint	135
8.4.30	ST_MakePointM	136
8.4.31	ST_MLineFromText	137
8.4.32	ST_MPointFromText	138
8.4.33	ST_MPolyFromText	138
8.4.34	ST_Point	139
8.4.35	ST_PointFromGeoHash	140
8.4.36	ST_PointFromText	141
8.4.37	ST_PointFromWKB	142
8.4.38	ST_Polygon	143
8.4.39	ST_PolygonFromText	143
8.4.40	ST_WKBToSQL	144
8.4.41	ST_WKTToSQL	145
8.5	Acessors de Geometria	145
8.5.1	Tipo de geometria	145
8.5.2	ST_Boundary	146
8.5.3	ST_CoordDim	149
8.5.4	ST_Dimension	150
8.5.5	ST_EndPoint	150

---

8.5.6	ST_Envelope	151
8.5.7	ST_BoundingDiagonal	153
8.5.8	ST_ExteriorRing	154
8.5.9	ST_GeometryN	155
8.5.10	ST_GeometryType	157
8.5.11	ST_InteriorRingN	158
8.5.12	ST_IsPolygonCCW	159
8.5.13	ST_IsPolygonCW	160
8.5.14	ST_IsClosed	161
8.5.15	ST_IsCollection	162
8.5.16	ST_IsEmpty	164
8.5.17	ST_IsRing	165
8.5.18	ST_IsSimple	166
8.5.19	ST_IsValid	167
8.5.20	ST_IsValidReason	167
8.5.21	ST_IsValidDetail	169
8.5.22	ST_M	170
8.5.23	ST_NDims	171
8.5.24	ST_NPoints	171
8.5.25	ST_NRings	172
8.5.26	ST_NumGeometries	172
8.5.27	ST_NumInteriorRings	173
8.5.28	ST_NumInteriorRing	174
8.5.29	ST_NumPatches	174
8.5.30	ST_NumPoints	175
8.5.31	ST_PatchN	176
8.5.32	ST_PointN	177
8.5.33	ST_Points	178
8.5.34	ST_SRID	179
8.5.35	ST_StartPoint	180
8.5.36	ST_Summary	181
8.5.37	ST_X	182
8.5.38	ST_XMax	183
8.5.39	ST_XMin	184
8.5.40	ST_Y	185
8.5.41	ST_YMax	186
8.5.42	ST_YMin	187
8.5.43	ST_Z	188
8.5.44	ST_ZMax	189

---

8.5.45	ST_Zmflag	190
8.5.46	ST_ZMin	190
8.6	Editores de geometria	191
8.6.1	ST_AddPoint	191
8.6.2	ST_Affine	192
8.6.3	ST_Force2D	194
8.6.4	ST_Force3D	195
8.6.5	ST_Force3DZ	196
8.6.6	ST_Force3DM	197
8.6.7	ST_Force4D	197
8.6.8	ST_ForcePolygonCCW	198
8.6.9	ST_ForceCollection	199
8.6.10	ST_ForcePolygonCW	200
8.6.11	ST_ForceSFS	200
8.6.12	ST_ForceRHR	201
8.6.13	ST_ForceCurve	201
8.6.14	ST_LineMerge	202
8.6.15	ST_CollectionExtract	203
8.6.16	ST_CollectionHomogenize	204
8.6.17	ST_Multi	205
8.6.18	ST_Normalize	206
8.6.19	ST_QuantizeCoordinates	206
8.6.20	ST_RemovePoint	208
8.6.21	ST_Reverse	209
8.6.22	ST_Rotate	210
8.6.23	ST_RotateX	211
8.6.24	ST_RotateY	211
8.6.25	ST_RotateZ	212
8.6.26	ST_Scale	213
8.6.27	ST_Segmentize	215
8.6.28	ST_SetPoint	216
8.6.29	ST_SetSRID	217
8.6.30	ST_SnapToGrid	218
8.6.31	ST_Snap	219
8.6.32	ST_Transform	222
8.6.33	ST_Translate	225
8.6.34	ST_TransScale	226
8.7	Saídas de geometria	227
8.7.1	ST_AsBinary	227

---



8.7.2	ST_AsEncodedPolyline	229
8.7.3	ST_AsEWKB	230
8.7.4	ST_AsEWKT	231
8.7.5	ST_AsGeoJSON	232
8.7.6	ST_AsGML	233
8.7.7	ST_AsHEXEWKB	236
8.7.8	ST_AsKML	237
8.7.9	ST_AsLatLonText	238
8.7.10	ST_AsSVG	240
8.7.11	ST_AsText	240
8.7.12	ST_AsTWKB	242
8.7.13	ST_AsX3D	243
8.7.14	ST_GeoHash	246
8.7.15	ST_AsGeobuf	247
8.7.16	ST_AsMVTGeom	247
8.7.17	ST_AsMVT	248
8.8	Operadores	249
8.8.1	&&	249
8.8.2	&&(geometry,box2df)	250
8.8.3	&&(box2df,geometry)	251
8.8.4	&&(box2df,box2df)	251
8.8.5	&&&	252
8.8.6	&&&(geometry,gidx)	253
8.8.7	&&&(gidx,geometry)	254
8.8.8	&&&(gidx,gidx)	255
8.8.9	&<	256
8.8.10	&<	257
8.8.11	&>	258
8.8.12	<<	258
8.8.13	<<	259
8.8.14	=	260
8.8.15	>>	261
8.8.16	@	262
8.8.17	@(geometry,box2df)	263
8.8.18	@(box2df,geometry)	264
8.8.19	@(box2df,box2df)	264
8.8.20	&>	265
8.8.21	>>	266
8.8.22	~	267

8.8.23	~(geometry,box2df)	267
8.8.24	~(box2df,geometry)	268
8.8.25	~(box2df,box2df)	269
8.8.26	~=	270
8.8.27	<->	271
8.8.28	=	273
8.8.29	<#>	274
8.8.30	<<->>	275
8.8.31	<<#>>	275
8.9	Relações espaciais e medidas	276
8.9.1	ST_3DClosestPoint	276
8.9.2	ST_3DDistance	277
8.9.3	ST_3DDWithin	278
8.9.4	ST_3DDFullyWithin	279
8.9.5	ST_3DIntersects	280
8.9.6	ST_3DLongestLine	281
8.9.7	ST_3DMaxDistance	282
8.9.8	ST_3DShortestLine	283
8.9.9	ST_Area	285
8.9.10	ST_Azimuth	286
8.9.11	ST_Angle	288
8.9.12	ST_Centroid	288
8.9.13	ST_ClosestPoint	290
8.9.14	ST_ClusterDBSCAN	292
8.9.15	ST_ClusterIntersecting	294
8.9.16	ST_ClusterKMeans	294
8.9.17	ST_ClusterWithin	295
8.9.18	ST_Contains	296
8.9.19	ST_ContainsProperly	299
8.9.20	ST_Covers	300
8.9.21	ST_CoveredBy	302
8.9.22	ST_Crosses	303
8.9.23	ST_LineCrossingDirection	305
8.9.24	ST_Disjoint	307
8.9.25	ST_Distance	308
8.9.26	ST_MinimumClearance	310
8.9.27	ST_MinimumClearanceLine	311
8.9.28	ST_HausdorffDistance	312
8.9.29	ST_FrechetDistance	313

---

8.9.30	ST_MaxDistance	314
8.9.31	ST_DistanceSphere	315
8.9.32	ST_DistanceSpheroid	315
8.9.33	ST_DFullyWithin	316
8.9.34	ST_DWithin	317
8.9.35	ST_Equals	318
8.9.36	ST_GeometricMedian	319
8.9.37	ST_HasArc	321
8.9.38	ST_Intersects	321
8.9.39	ST_Length	323
8.9.40	ST_Length2D	324
8.9.41	ST_3DLength	324
8.9.42	ST_LengthSpheroid	325
8.9.43	ST_Length2D_Spheroid	326
8.9.44	ST_LongestLine	327
8.9.45	ST_OrderingEquals	329
8.9.46	ST_Overlaps	330
8.9.47	ST_Perimeter	332
8.9.48	ST_Perimeter2D	334
8.9.49	ST_3DPerímetro	334
8.9.50	ST_PointOnSurface	335
8.9.51	ST_Project	336
8.9.52	ST_Relate	337
8.9.53	ST_RelateMatch	338
8.9.54	ST_ShortestLine	339
8.9.55	ST_Touches	340
8.9.56	ST_Within	342
8.10	SFCGAL Funções	344
8.10.1	postgis_sfcgal_version	344
8.10.2	ST_Extrude	344
8.10.3	ST_StraightSkeleton	346
8.10.4	ST_ApproximateMedialAxis	347
8.10.5	ST_IsPlanar	348
8.10.6	ST_Orientation	348
8.10.7	ST_ForceLHR	348
8.10.8	ST_MinkowskiSum	349
8.10.9	ST_3DIntersection	351
8.10.10	ST_3DDifference	353
8.10.11	ST_3DUnion	354

---

8.10.12 ST_3DArea . . . . .	355
8.10.13 ST_Tesselate . . . . .	356
8.10.14 ST_Volume . . . . .	358
8.10.15 ST_MakeSolid . . . . .	359
8.10.16 ST_IsSolid . . . . .	359
8.11 Processamento de Geometria . . . . .	360
8.11.1 ST_Buffer . . . . .	360
8.11.2 ST_BuildArea . . . . .	365
8.11.3 ST_ClipByBox2D . . . . .	366
8.11.4 ST_Collect . . . . .	367
8.11.5 ST_ConcaveHull . . . . .	369
8.11.6 ST_ConvexHull . . . . .	373
8.11.7 ST_CurveToLine . . . . .	374
8.11.8 ST_DelaunayTriangles . . . . .	377
8.11.9 ST_Difference . . . . .	382
8.11.10 ST_Dump . . . . .	383
8.11.11 ST_DumpPoints . . . . .	385
8.11.12 ST_DumpRings . . . . .	389
8.11.13 ST_FlipCoordinates . . . . .	390
8.11.14 ST_GeneratePoints . . . . .	391
8.11.15 ST_Intersection . . . . .	392
8.11.16 ST_LineToCurve . . . . .	394
8.11.17 ST_MakeValid . . . . .	396
8.11.18 ST_MemUnion . . . . .	396
8.11.19 ST_MinimumBoundingCircle . . . . .	397
8.11.20 ST_MinimumBoundingRadius . . . . .	398
8.11.21 ST_OrientedEnvelope . . . . .	399
8.11.22 ST_Polygonize . . . . .	400
8.11.23 ST_Node . . . . .	401
8.11.24 ST_OffsetCurve . . . . .	402
8.11.25 ST_RemoveRepeatedPoints . . . . .	405
8.11.26 ST_SharedPaths . . . . .	406
8.11.27 ST_ShiftLongitude . . . . .	408
8.11.28 ST_WrapX . . . . .	409
8.11.29 ST_Simplify . . . . .	409
8.11.30 ST_SimplifyPreserveTopology . . . . .	410
8.11.31 ST_SimplifyVW . . . . .	411
8.11.32 ST_ChaikinSmoothing . . . . .	412
8.11.33 ST_FilterByM . . . . .	413

---

8.11.34	ST_SetEffectiveArea	414
8.11.35	ST_Split	415
8.11.36	ST_SymDifference	417
8.11.37	ST_Subdivide	419
8.11.38	ST_SwapOrdinates	421
8.11.39	ST_Union	422
8.11.40	ST_UnaryUnion	424
8.11.41	ST_Voronoi	425
8.11.42	ST_Voronoi	426
8.12	Referência linear	429
8.12.1	ST_LineInterpolatePoint	429
8.12.2	ST_LineInterpolatePoints	431
8.12.3	ST_LineLocatePoint	432
8.12.4	ST_LineSubstring	433
8.12.5	ST_LocateAlong	435
8.12.6	ST_LocateBetween	436
8.12.7	ST_LocateBetweenElevations	437
8.12.8	ST_InterpolatePoint	438
8.12.9	ST_AddMeasure	438
8.13	Suporte Temporal	439
8.13.1	ST_IsValidTrajectory	439
8.13.2	ST_ClosestPointOfApproach	440
8.13.3	ST_DistanceCPA	441
8.13.4	ST_CPAWithin	441
8.14	Suporte de longas transações	442
8.14.1	AddAuth	442
8.14.2	CheckAuth	443
8.14.3	DesativarLongasTransações	444
8.14.4	AtivarLongasTransações	444
8.14.5	LockRow	445
8.14.6	UnlockRows	446
8.15	Funções Variadas	446
8.15.1	ST_Accum	446
8.15.2	Caixa2D	447
8.15.3	Caixa3D	448
8.15.4	ST_EstimatedExtent	449
8.15.5	ST_Expand	449
8.15.6	ST_Extent	451
8.15.7	ST_3DExtent	452

---

8.15.8	Find_SRID	454
8.15.9	ST_MemSize	454
8.15.10	ST_PointInsideCircle	455
8.16	Funções excepcionais	456
8.16.1	PostGIS_AddBBox	456
8.16.2	PostGIS_DropBBox	457
8.16.3	PostGIS_HasBBox	458
<b>9</b>	<b>Referência Raster</b>	<b>459</b>
9.1	Tipos de suporte de dados raster	460
9.1.1	geomval	460
9.1.2	addbandarg	460
9.1.3	rastbandarg	460
9.1.4	raster	461
9.1.5	reclassarg	461
9.1.6	summarystats	462
9.1.7	unionarg	462
9.2	Gerenciamento Raster	463
9.2.1	AddRasterConstraints	463
9.2.2	DropRasterConstraints	465
9.2.3	AddOverviewConstraints	466
9.2.4	DropOverviewConstraints	466
9.2.5	PostGIS_GDAL_Version	467
9.2.6	PostGIS_Raster_Lib_Build_Date	467
9.2.7	PostGIS_Raster_Lib_Version	468
9.2.8	ST_GDALDrivers	468
9.2.9	UpdateRasterSRID	473
9.2.10	ST_CreateOverview	473
9.3	Construtores Raster	474
9.3.1	ST_AddBand	474
9.3.2	ST_AsRaster	477
9.3.3	ST_Band	479
9.3.4	ST_MakeEmptyCoverage	480
9.3.5	ST_MakeEmptyRaster	482
9.3.6	ST_Tile	483
9.3.7	ST_Retile	485
9.3.8	ST_FromGDALRaster	485
9.4	Assessores Raster	486
9.4.1	ST_GeoReference	486

---

9.4.2	ST_Height	487
9.4.3	ST_IsEmpty	488
9.4.4	ST_MemSize	488
9.4.5	ST_MetaData	489
9.4.6	ST_NumBands	489
9.4.7	ST_PixelHeight	490
9.4.8	ST_PixelWidth	491
9.4.9	ST_ScaleX	492
9.4.10	ST_ScaleY	493
9.4.11	ST_RasterToWorldCoord	493
9.4.12	ST_RasterToWorldCoordX	494
9.4.13	ST_RasterToWorldCoordY	495
9.4.14	ST_Rotation	496
9.4.15	ST_SkewX	497
9.4.16	ST_SkewY	497
9.4.17	ST_SRID	498
9.4.18	ST_Summary	499
9.4.19	ST_UpperLeftX	499
9.4.20	ST_UpperLeftY	500
9.4.21	ST_Width	500
9.4.22	ST_WorldToRasterCoord	501
9.4.23	ST_WorldToRasterCoordX	502
9.4.24	ST_WorldToRasterCoordY	502
9.5	Assessores de banda raster	503
9.5.1	ST_BandMetaData	503
9.5.2	ST_BandNoDataValue	504
9.5.3	ST_BandIsNoData	505
9.5.4	ST_BandPath	506
9.5.5	ST_BandFileSize	507
9.5.6	ST_BandFileTimestamp	507
9.5.7	ST_BandPixelType	508
9.5.8	ST_HasNoBand	509
9.6	Assessores e Setters de Pixel Raster	509
9.6.1	ST_PixelAsPolygon	509
9.6.2	ST_PixelAsPolygons	510
9.6.3	ST_PixelAsPoint	511
9.6.4	ST_PixelAsPoints	512
9.6.5	ST_PixelAsCentroid	513
9.6.6	ST_PixelAsCentroids	513

---

9.6.7	ST_Value	514
9.6.8	ST_NearestValue	517
9.6.9	ST_Neighborhood	519
9.6.10	ST_SetValue	521
9.6.11	ST_SetValues	522
9.6.12	ST_DumpValues	530
9.6.13	ST_PixelOfValue	531
9.7	Editores Raster	533
9.7.1	ST_SetGeoReference	533
9.7.2	ST_SetRotation	534
9.7.3	ST_SetScale	535
9.7.4	ST_SetSkew	536
9.7.5	ST_SetSRID	536
9.7.6	ST_SetUpperLeft	537
9.7.7	ST_Resample	537
9.7.8	ST_Rescale	539
9.7.9	ST_Reskew	540
9.7.10	ST_SnapToGrid	541
9.7.11	ST_Resize	542
9.7.12	ST_Transform	543
9.8	Editores de Banda Raster	546
9.8.1	ST_SetBandNoDataValue	546
9.8.2	ST_SetBandIsNoData	547
9.8.3	ST_SetBandPath	548
9.8.4	ST_SetBandIndex	549
9.9	Análises e Estatísticas de Banda Raster	551
9.9.1	ST_Count	551
9.9.2	ST_CountAgg	552
9.9.3	ST_Histogram	553
9.9.4	ST_Quantile	555
9.9.5	ST_SummaryStats	556
9.9.6	ST_SummaryStatsAgg	559
9.9.7	ST_ValueCount	560
9.10	Raster Inputs	562
9.10.1	ST_RastFromWKB	562
9.10.2	ST_RastFromHexWKB	563
9.11	Raster Outputs	564
9.11.1	ST_AsBinary/ST_AsWKB	564
9.11.2	ST_AsHexWKB	564

---



9.11.3	ST_AsGDALRaster	565
9.11.4	ST_AsJPEG	566
9.11.5	ST_AsPNG	567
9.11.6	ST_AsTIFF	568
9.12	Processamento Raster	569
9.12.1	Mapa Algébrico	569
9.12.1.1	ST_Clip	569
9.12.1.2	ST_ColorMap	572
9.12.1.3	ST_Grayscale	575
9.12.1.4	ST_Intersection	577
9.12.1.5	Funções retorno de mapa algébrico embutido	579
9.12.1.6	ST_MapAlgebraExpr	585
9.12.1.7	ST_MapAlgebraExpr	587
9.12.1.8	ST_MapAlgebraExpr	589
9.12.1.9	ST_MapAlgebraFct	594
9.12.1.10	ST_MapAlgebraFct	598
9.12.1.11	ST_MapAlgebraFctNgb	602
9.12.1.12	ST_Reclass	604
9.12.1.13	ST_Union	605
9.12.2	Funções retorno de mapa algébrico embutido	607
9.12.2.1	ST_Distinct4ma	607
9.12.2.2	ST_InvDistWeight4ma	608
9.12.2.3	ST_Max4ma	608
9.12.2.4	ST_Mean4ma	609
9.12.2.5	ST_Min4ma	611
9.12.2.6	ST_MinDist4ma	612
9.12.2.7	ST_Range4ma	612
9.12.2.8	ST_StdDev4ma	613
9.12.2.9	ST_Sum4ma	614
9.12.3	DEM (Elevação)	615
9.12.3.1	ST_Aspect	615
9.12.3.2	ST_HillShade	617
9.12.3.3	ST_Roughness	619
9.12.3.4	ST_Slope	619
9.12.3.5	ST_TPI	621
9.12.3.6	ST_TRI	622
9.12.4	Raster para Geometria	622
9.12.4.1	Caixa3D	622
9.12.4.2	ST_ConvexHull	623

9.12.4.3	ST_DumpAsPolygons . . . . .	624
9.12.4.4	ST_Envelope . . . . .	625
9.12.4.5	ST_MinConvexHull . . . . .	626
9.12.4.6	ST_Polygon . . . . .	627
9.13	Operadores Raster . . . . .	628
9.13.1	&& . . . . .	628
9.13.2	&< . . . . .	629
9.13.3	&> . . . . .	629
9.13.4	= . . . . .	630
9.13.5	@ . . . . .	631
9.13.6	~= . . . . .	631
9.13.7	~ . . . . .	632
9.14	Relações raster e raster de banda espacial . . . . .	632
9.14.1	ST_Contains . . . . .	632
9.14.2	ST_ContainsProperly . . . . .	633
9.14.3	ST_Covers . . . . .	634
9.14.4	ST_CoveredBy . . . . .	635
9.14.5	ST_Disjoint . . . . .	636
9.14.6	ST_Intersects . . . . .	637
9.14.7	ST_Overlaps . . . . .	638
9.14.8	ST_Touches . . . . .	638
9.14.9	ST_SameAlignment . . . . .	639
9.14.10	ST_NotSameAlignmentReason . . . . .	640
9.14.11	ST_Within . . . . .	641
9.14.12	ST_DWithin . . . . .	642
9.14.13	ST_DFullyWithin . . . . .	643
9.15	Raster Tips . . . . .	644
9.15.1	Out-DB Rasters . . . . .	644
9.15.1.1	Directory containing many files . . . . .	644
9.15.1.2	Maximum Number of Open Files . . . . .	644
9.15.1.2.1	Maximum number of open files for the entire system . . . . .	645
9.15.1.2.2	Maximum number of open files per process . . . . .	645

## 10 Perguntas frequentes PostGIS Raster

648

<b>11 Topologia</b>	<b>652</b>
11.1 Tipos de topologia	652
11.1.1 getfaceedges_returntype	652
11.1.2 TopoGeometry	653
11.1.3 validateTopology_returntype	653
11.2 Domínios de Topologia	654
11.2.1 TopoElement	654
11.2.2 TopoElementArray	654
11.3 Gerenciamento de Topologia e TopoGeometria	655
11.3.1 AddTopoGeometryColumn	655
11.3.2 DropTopology	656
11.3.3 DropTopoGeometryColumn	657
11.3.4 Populate_Topology_Layer	657
11.3.5 TopologySummary	658
11.3.6 ValidateTopology	659
11.4 Construtores de topologia	660
11.4.1 Cria topologia	660
11.4.2 CopyTopology	661
11.4.3 ST_InitTopoGeo	661
11.4.4 ST_CreateTopoGeo	662
11.4.5 TopoGeo_AddPoint	663
11.4.6 TopoGeo_AddLineString	663
11.4.7 TopoGeo_AddPolygon	664
11.5 Editores de Topologia	664
11.5.1 ST_AddIsoNode	664
11.5.2 ST_AddIsoEdge	665
11.5.3 ST_AddEdgeNewFaces	665
11.5.4 ST_AddEdgeModFace	666
11.5.5 ST_RemEdgeNewFace	666
11.5.6 ST_RemEdgeModFace	667
11.5.7 ST_ChangeEdgeGeom	668
11.5.8 ST_ModEdgeSplit	668
11.5.9 ST_ModEdgeHeal	669
11.5.10 ST_NewEdgeHeal	670
11.5.11 ST_MoveIsoNode	670
11.5.12 ST_NewEdgesSplit	671
11.5.13 ST_RemoveIsoNode	672
11.5.14 ST_RemoveIsoEdge	672
11.6 Assessores de Topologia	673

---

11.6.1	GetEdgeByPoint	673
11.6.2	GetFaceByPoint	674
11.6.3	GetNodeByPoint	674
11.6.4	GetTopologyID	675
11.6.5	GetTopologySRID	676
11.6.6	GetTopologyName	676
11.6.7	ST_GetFaceEdges	677
11.6.8	ST_GetFaceGeometry	678
11.6.9	GetRingEdges	678
11.6.10	GetNodeEdges	679
11.7	Processamento de Topologia	679
11.7.1	Polygonize	679
11.7.2	AddNode	680
11.7.3	AddEdge	681
11.7.4	AddFace	682
11.7.5	ST_Simplify	683
11.8	Construtores de TopoGeometria	684
11.8.1	CreateTopoGeom	684
11.8.2	toTopoGeom	685
11.8.3	TopoElementArray_Agg	687
11.9	Editores de TopoGeometria	687
11.9.1	clearTopoGeom	687
11.9.2	TopoGeom_addElement	688
11.9.3	TopoGeom_remElement	688
11.9.4	toTopoGeom	689
11.10	Assessores de TopoGeometria	689
11.10.1	GetTopoGeomElementArray	689
11.10.2	GetTopoGeomElements	690
11.11	TopoGeometry Outputs	690
11.11.1	AsGML	690
11.11.2	AsTopoJSON	692
11.12	Relações de Topologia Espacial	694
11.12.1	Equivalentes	694
11.12.2	Intercepta	695

---

<b>12</b>	<b>Padronizador de endereço</b>	<b>696</b>
12.1	Como o analisador sintático funciona . . . . .	696
12.2	Tipos de padronizador de endereço . . . . .	696
12.2.1	stdaddr . . . . .	696
12.3	Mesas de padronizador de endereço . . . . .	697
12.3.1	mesa de regras . . . . .	697
12.3.2	lex table . . . . .	700
12.3.3	gaz table . . . . .	701
12.4	Funções do padronizador de endereços . . . . .	701
12.4.1	parse_address . . . . .	701
12.4.2	standardize_address . . . . .	702
<b>13</b>	<b>PostGIS Extras</b>	<b>705</b>
13.1	Tiger Geocoder . . . . .	705
13.1.1	Drop_Indexes_Generate_Script . . . . .	705
13.1.2	Drop_Nation_Tables_Generate_Script . . . . .	706
13.1.3	Drop_State_Tables_Generate_Script . . . . .	707
13.1.4	Geocode . . . . .	708
13.1.5	Geocode_Intersection . . . . .	710
13.1.6	Get_Geocode_Setting . . . . .	711
13.1.7	Get_Tract . . . . .	712
13.1.8	Install_Missing_Indexes . . . . .	713
13.1.9	Loader_Generate_Census_Script . . . . .	713
13.1.10	Loader_Generate_Script . . . . .	715
13.1.11	Loader_Generate_Nation_Script . . . . .	717
13.1.12	Missing_Indexes_Generate_Script . . . . .	718
13.1.13	Normalize_Address . . . . .	719
13.1.14	Pagc_Normalize_Address . . . . .	720
13.1.15	Pprint_Addy . . . . .	722
13.1.16	Reverse_Geocode . . . . .	723
13.1.17	Topology_Load_Tiger . . . . .	725
13.1.18	Set_Geocode_Setting . . . . .	727
<b>14</b>	<b>PostGIS Special Functions Index</b>	<b>728</b>
14.1	PostGIS Aggregate Functions . . . . .	728
14.2	PostGIS Window Functions . . . . .	728
14.3	PostGIS SQL-MM Compliant Functions . . . . .	729
14.4	PostGIS Geography Support Functions . . . . .	734
14.5	PostGIS Raster Support Functions . . . . .	735

14.6 PostGIS Geometry / Geography / Raster Dump Functions . . . . .	740
14.7 PostGIS Box Functions . . . . .	741
14.8 PostGIS Functions that support 3D . . . . .	742
14.9 PostGIS Curved Geometry Support Functions . . . . .	747
14.10 PostGIS Polyhedral Surface Support Functions . . . . .	750
14.11 PostGIS Function Support Matrix . . . . .	753
14.12 New, Enhanced or changed PostGIS Functions . . . . .	762
14.12.1 PostGIS Functions new or enhanced in 2.5 . . . . .	762
14.12.2 PostGIS Functions new or enhanced in 2.4 . . . . .	763
14.12.3 PostGIS Functions new or enhanced in 2.3 . . . . .	765
14.12.4 PostGIS Functions new or enhanced in 2.2 . . . . .	766
14.12.5 PostGIS functions breaking changes in 2.2 . . . . .	766
14.12.6 PostGIS Functions new or enhanced in 2.1 . . . . .	766
14.12.7 PostGIS Functions new, behavior changed, or enhanced in 2.0 . . . . .	767
14.12.8 PostGIS Functions changed behavior in 2.0 . . . . .	767
14.12.9 PostGIS Functions new, behavior changed, or enhanced in 1.5 . . . . .	768
14.12.10 PostGIS Functions new, behavior changed, or enhanced in 1.4 . . . . .	768
14.12.11 PostGIS Functions new in 1.3 . . . . .	768
<b>15 Reporting Problems</b>	<b>769</b>
15.1 Reporting Software Bugs . . . . .	769
15.2 Reporting Documentation Issues . . . . .	769
<b>A Apêndice</b>	<b>770</b>
A.1 Release 2.5.0 . . . . .	770
A.1.1 Novos Recursos . . . . .	770
A.1.2 Breaking Changes . . . . .	771
A.2 Release 2.4.4 . . . . .	771
A.2.1 Correção de Erros . . . . .	772
A.2.2 Melhorias . . . . .	772
A.3 Release 2.4.3 . . . . .	772
A.3.1 Bug Fixes and Enhancements . . . . .	772
A.4 Release 2.4.2 . . . . .	772
A.4.1 Bug Fixes and Enhancements . . . . .	773
A.5 Release 2.4.1 . . . . .	773
A.5.1 Bug Fixes and Enhancements . . . . .	773
A.6 Release 2.4.0 . . . . .	773
A.6.1 Novos Recursos . . . . .	773
A.6.2 Enhancements and Fixes . . . . .	774

---

A.6.3 Breaking Changes . . . . .	774
A.7 Release 2.3.3 . . . . .	775
A.7.1 Bug Fixes and Enhancements . . . . .	775
A.8 Release 2.3.2 . . . . .	775
A.8.1 Bug Fixes and Enhancements . . . . .	775
A.9 Release 2.3.1 . . . . .	775
A.9.1 Bug Fixes and Enhancements . . . . .	776
A.10 Release 2.3.0 . . . . .	776
A.10.1 Importante / Mudanças Críticas . . . . .	776
A.10.2 Novos Recursos . . . . .	776
A.10.3 Correção de Erros . . . . .	777
A.10.4 Melhorias de Desempenho . . . . .	777
A.11 Release 2.2.2 . . . . .	777
A.11.1 Novos Recursos . . . . .	777
A.12 Release 2.2.1 . . . . .	778
A.12.1 Novos Recursos . . . . .	778
A.13 Versão 2.2.0 . . . . .	778
A.13.1 Novos Recursos . . . . .	779
A.13.2 Melhorias . . . . .	780
A.14 Versão 2.1.8 . . . . .	780
A.14.1 Correção de Erros . . . . .	781
A.15 Versão 2.1.7 . . . . .	781
A.15.1 Correção de Erros . . . . .	781
A.16 Versão 2.1.6 . . . . .	781
A.16.1 Melhorias . . . . .	781
A.16.2 Correção de Erros . . . . .	781
A.17 Versão 2.1.5 . . . . .	782
A.17.1 Melhorias . . . . .	782
A.17.2 Correção de Erros . . . . .	782
A.18 Versão 2.1.4 . . . . .	782
A.18.1 Melhorias . . . . .	782
A.18.2 Correção de Erros . . . . .	783
A.19 Versão 2.1.3 . . . . .	783
A.19.1 Mudanças importantes . . . . .	783
A.19.2 Correção de Erros . . . . .	784
A.20 Versão 2.1.2 . . . . .	784
A.20.1 Correção de Erros . . . . .	784
A.20.2 Melhorias . . . . .	784
A.21 Versão 2.1.1 . . . . .	785

---

A.21.1 Mudanças importantes . . . . .	785
A.21.2 Correção de Erros . . . . .	785
A.21.3 Melhorias . . . . .	785
A.22 Versão 2.1.0 . . . . .	785
A.22.1 Importante / Mudanças Críticas . . . . .	785
A.22.2 Novos Recursos . . . . .	786
A.22.3 Melhorias . . . . .	787
A.22.4 Correções . . . . .	789
A.22.5 Known Issues . . . . .	790
A.23 Versão 2.0.5 . . . . .	790
A.23.1 Correção de Erros . . . . .	790
A.23.2 Mudanças importantes . . . . .	790
A.24 Versão 2.0.4 . . . . .	790
A.24.1 Correção de Erros . . . . .	790
A.24.2 Melhorias . . . . .	791
A.24.3 Known Issues . . . . .	791
A.25 Versão 2.0.3 . . . . .	791
A.25.1 Correção de Erros . . . . .	792
A.25.2 Melhorias . . . . .	792
A.26 Versão 2.0.2 . . . . .	792
A.26.1 Correção de Erros . . . . .	792
A.26.2 Melhorias . . . . .	793
A.27 Versão 2.0.1 . . . . .	793
A.27.1 Correção de Erros . . . . .	794
A.27.2 Melhorias . . . . .	795
A.28 Versão 2.0.0 . . . . .	795
A.28.1 Verificadores - Nossos heróis não aclamados . . . . .	795
A.28.2 Importante / Mudanças Críticas . . . . .	795
A.28.3 Novos Recursos . . . . .	796
A.28.4 Melhorias . . . . .	796
A.28.5 Correção de Erros . . . . .	797
A.28.6 Créditos específicos deste lançamento . . . . .	797
A.29 Versão 1.5.4 . . . . .	797
A.29.1 Correção de Erros . . . . .	797
A.30 Versão 1.5.3 . . . . .	798
A.30.1 Correção de Erros . . . . .	798
A.31 Versão 1.5.2 . . . . .	798
A.31.1 Correção de Erros . . . . .	798
A.32 Versão 1.5.1 . . . . .	799

---



A.32.1 Correção de Erros . . . . .	799
A.33 Versão 1.5.0 . . . . .	799
A.33.1 API Stability . . . . .	800
A.33.2 Compatibilidade . . . . .	800
A.33.3 Novos Recursos . . . . .	800
A.33.4 Melhorias . . . . .	801
A.33.5 Correção de Erros . . . . .	801
A.34 Versão 1.4.0 . . . . .	801
A.34.1 API Stability . . . . .	801
A.34.2 Compatibilidade . . . . .	801
A.34.3 Novos Recursos . . . . .	801
A.34.4 Melhorias . . . . .	802
A.34.5 Correção de Erros . . . . .	802
A.35 Versão 1.3.6 . . . . .	802
A.36 Versão 1.3.5 . . . . .	802
A.37 Versão 1.3.4 . . . . .	803
A.38 Versão 1.3.3 . . . . .	803
A.39 Versão 1.3.2 . . . . .	803
A.40 Versão 1.3.1 . . . . .	803
A.41 Versão 1.3.0 . . . . .	803
A.41.1 Funcionalidade Adicionada . . . . .	803
A.41.2 Melhorias de Desempenho . . . . .	803
A.41.3 Outras Mudanças . . . . .	804
A.42 Versão 1.2.1 . . . . .	804
A.42.1 Mudanças . . . . .	804
A.43 Versão 1.2.0 . . . . .	804
A.43.1 Mudanças . . . . .	804
A.44 Versão 1.1.6 . . . . .	804
A.44.1 Atualizando . . . . .	804
A.44.2 Correção de Erros . . . . .	805
A.44.3 Outras mudanças . . . . .	805
A.45 Versão 1.1.5 . . . . .	805
A.45.1 Atualizando . . . . .	805
A.45.2 Correção de Erros . . . . .	805
A.45.3 Novos Recursos . . . . .	805
A.46 Versão 1.1.4 . . . . .	805
A.46.1 Atualizando . . . . .	806
A.46.2 Correção de Erros . . . . .	806
A.46.3 Mudanças para o Java . . . . .	806

---

A.47	Versão 1.1.3	806
A.47.1	Atualizando	806
A.47.2	Bug fixes / correctness	806
A.47.3	New functionalities	807
A.47.4	Mudanças para o JDBC	807
A.47.5	Outras mudanças	807
A.48	Versão 1.1.2	807
A.48.1	Atualizando	807
A.48.2	Correção de Erros	807
A.48.3	New functionalities	808
A.48.4	Outras mudanças	808
A.49	Versão 1.1.1	808
A.49.1	Atualizando	808
A.49.2	Correção de Erros	808
A.49.3	New functionalities	808
A.50	Versão 1.1.0	809
A.50.1	Credits	809
A.50.2	Atualizando	809
A.50.3	Novas funções	809
A.50.4	Correção de Erros	810
A.50.5	Function semantic changes	810
A.50.6	Performance improvements	810
A.50.7	JDBC2 works	810
A.50.8	Outras coisas novas	810
A.50.9	Outras mudanças	810
A.51	Versão 1.0.6	811
A.51.1	Atualizando	811
A.51.2	Correção de Erros	811
A.51.3	Melhorias	811
A.52	Versão 1.0.5	811
A.52.1	Atualizando	811
A.52.2	Library changes	812
A.52.3	Loader changes	812
A.52.4	Outras mudanças	812
A.53	Versão 1.0.4	812
A.53.1	Atualizando	812
A.53.2	Correção de Erros	812
A.53.3	Melhorias	813
A.54	Versão 1.0.3	813

---

A.54.1	Atualizando . . . . .	813
A.54.2	Correção de Erros . . . . .	813
A.54.3	Melhorias . . . . .	813
A.55	Versão 1.0.2 . . . . .	813
A.55.1	Atualizando . . . . .	814
A.55.2	Correção de Erros . . . . .	814
A.55.3	Melhorias . . . . .	814
A.56	Versão 1.0.1 . . . . .	814
A.56.1	Atualizando . . . . .	814
A.56.2	Library changes . . . . .	814
A.56.3	Other changes/additions . . . . .	814
A.57	Versão 1.0.0 . . . . .	815
A.57.1	Atualizando . . . . .	815
A.57.2	Library changes . . . . .	815
A.57.3	Other changes/additions . . . . .	815
A.58	Versão 1.0.0RC6 . . . . .	815
A.58.1	Atualizando . . . . .	815
A.58.2	Library changes . . . . .	815
A.58.3	Scripts changes . . . . .	815
A.58.4	Outras mudanças . . . . .	816
A.59	Release 1.0.0RC5 . . . . .	816
A.59.1	Atualizando . . . . .	816
A.59.2	Library changes . . . . .	816
A.59.3	Outras mudanças . . . . .	816
A.60	Versão 1.0.0RC4 . . . . .	816
A.60.1	Atualizando . . . . .	816
A.60.2	Library changes . . . . .	816
A.60.3	Scripts changes . . . . .	817
A.60.4	Outras mudanças . . . . .	817
A.61	Versão 1.0.0RC3 . . . . .	817
A.61.1	Atualizando . . . . .	817
A.61.2	Library changes . . . . .	817
A.61.3	Scripts changes . . . . .	817
A.61.4	Mudanças para o JDBC . . . . .	818
A.61.5	Outras mudanças . . . . .	818
A.62	Versão 1.0.0RC2 . . . . .	818
A.62.1	Atualizando . . . . .	818
A.62.2	Library changes . . . . .	818
A.62.3	Scripts changes . . . . .	818
A.62.4	Outras mudanças . . . . .	819
A.63	Versão 1.0.0RC1 . . . . .	819
A.63.1	Atualizando . . . . .	819
A.63.2	Mudanças . . . . .	819

---

## Abstract

PostGIS é uma extensão para o sistema de banco de dados objeto-relacional PostgreSQL que permite que objetos SIG (Sistema de Informação Geográfica) sejam armazenados em banco de dados. O PostGIS inclui suporte a índices espaciais baseado em GiST R-Tree, e funções para análise e processamento de objetos SIG.



Este é o manual para a versão 2.5.0



Este trabalho está licenciado sobre a [Creative Commons Attribution-Share Alike 3.0 License](https://creativecommons.org/licenses/by-sa/3.0/). Sinta-se livre para utilizar este material como quiser, mas pedimos que você atribua o crédito ao projeto PostGIS e sempre que possível cite o link <http://www.postgis.org>.

# Chapter 1

## Introdução

PostGIS is a spatial extender for the PostgreSQL relational database that was created by Refrations Research Inc, as a spatial database technology research project. Refrations is a GIS and database consulting company in Victoria, British Columbia, Canada, specializing in data integration and custom software development.

PostGIS is now a project of the OSGeo Foundation and is developed and funded by many FOSS4G Developers as well as corporations all over the world that gain great benefit from its functionality and versatility.

The PostGIS project development group plans on supporting and enhancing PostGIS to better support a range of important GIS functionality in the areas of OpenGIS and SQL/MM spatial standards, advanced topological constructs (coverages, surfaces, networks), data source for desktop user interface tools for viewing and editing GIS data, and web-based access tools.

### 1.1 Comitê Diretor do Projeto

O Comitê Diretor do Projeto PostGIS (PSC - Project Steering Comitee, em inglês) é responsável pela direção geral, ciclos de lançamento, documentação e os esforços para o projeto. Além disso, o comitê dá suporte ao usuário comum, aceita e aprova novas melhorias da comunidade e vota em questões diversas envolvendo o PostGIS, como por exemplo, uma permissão de commit direta, novos membros do comitê e mudanças significativas da API (Application Programming Interface).

**Mark Cave-Ayland** Coordinates bug fixing and maintenance effort, spatial index selectivity and binding, loader/dumper, and Shapefile GUI Loader, integration of new and new function enhancements.

**Regina Obe** Buildbot Maintenance, windows production and experimental builds, Documentation, alignment of PostGIS with PostgreSQL releases, general user support on PostGIS newsgroup, X3D support, Tiger Geocoder Support, management functions, and smoke testing new functionality or major code changes.

**Bborie Park** O desenvolvimento raster, integração com GDAL, carregador raster, suporte ao usuário, correção de bugs em geral, testes em diversos sistemas operacionais (Slackware, Mac, Windows, e outros)

**Paul Ramsey (Presidente)** 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, PointCloud (in development), 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, git mirrors management, integration of new GEOS functionality and alignment with GEOS releases, Topology support, and Raster framework and low level api functions.

### 1.2 Contribuidores Núcleo Atuais

**Jorge Arévalo** Desenvolvimento Raster, suporte do driver GDAL e importador.

---

**Nicklas Avén** Melhorias em funções de distância (incluindo suporte a distância 3D e funções de relacionamento), Tiny WKB (TWKB) (em desenvolvimento) e suporte ao usuário geral.

**Dan Baston** Geometry clustering function additions, other geometry algorithm enhancements, GEOS enhancements and general user support

**Olivier Courtin** Funções para entrada e saída de XML (KML, GML)/GeoJSON, suporte a 3D e correção de bugs.

**Björn Harrtell** MapBox Vector Tile and GeoBuf functions. Gogs testing and GitLab experimentation.

**Mateusz Loskot** Suporte CMake para o PostGIS, criou o carregador raster original em Python e funções de baixo nível da API raster

**Raúl Marín Rodríguez** Bug fixing

**Darafei Praliaskouski** Index improvements, bug fixing and geometry/geography function improvements, GitHub curator, and Travis bot maintenance.

**Pierre Racine** Arquitetura Raster, prototipação e suporte ao desenvolvimento.

### 1.3 Contribuidores Núcleo Passado

**Chris Hodgson** Antigo membro do comitê. Desenvolvimento em geral, manutenção do website e buildbot, gerente da incubação na OSGeo.

**Kevin Neufeld** Ex PSC. Documentação e suporte a ferramentas de documentação, suporte e manutenção do buildbot, suporte avançado de usuários em listas de discussão e melhorias em funções do PostGIS

**Dave Blasby** Desenvolvedor original e co-fundador do PostGIS. Dave escreveu os objetos do servidor, chamadas de índices e muitas das funcionalidades analíticas presentes no servidor.

**Jeff Lounsbury** Desenvolvedor original do importador/exportador de shapefiles. Atual representante do Dono do Projeto.

**Mark Leslie** Manutenção e desenvolvimento de funções do núcleo. Melhorias para o suporte a curvas e no importador GUI.

**David Zwarg** Desenvolvimento raster (funções analíticas de álgebra de mapas)

### 1.4 Outros Contribuidores

**Contribuidores Individuais** Em ordem alfabética: Alex Bodnar, Alex Mayrhofer, Andrea Peri, Andreas Forø Tollefsen, Andreas Neumann, Anne Ghisla, Barbara Phillipot, Ben Jubb, Bernhard Reiter, Brian Hamlin, Bruce Rindahl, Bruno Wolff III, Bryce L. Nordgren, Carl Anderson, Charlie Savage, Dane Springmeyer, David Skea, David Techer, Eduin Carrillo, Even Rouault, Frank Warmerdam, George Silva, Gerald Fenoy, Gino Lucrezi, Guillaume Lelarge, IIDA Tetsushi, Ingvid Nystuen, Jason Smith, Jeff Adams, Jose Carlos Martinez Llari, Julien Rouhaud, Kashif Rasul, Klaus Foerster, Kris Jurka, Leo Hsu, Loic Dachary, Luca S. Percich, Maria Arias de Reyna, Mark Sondheim, Markus Schaber, Maxime Guillaud, Maxime van Noppen, Michael Fuhr, Mike Toews, Nathan Wagner, Nathaniel Clay, Nikita Shulga, Norman Vine, Rafal Magda, Ralph Mason, Rémi Cura, Richard Greenwood, Silvio Grosso, Steffen Macke, Stephen Frost, Tom van Tilburg, Vincent Mora, Vincent Picavet

**Patrocinadores corporativos** Estas são entidades corporativas que contribuíram com horas home, hospedagem ou suporte monetário direto ao projeto PostGIS

Em ordem alfabética: Arrival 3D, Associazione Italiana per l'Informazione Geografica Libera (GFOSS.it), AusVet, Avenacia, Azavea, Cadcorp, CampToCamp, CartoDB, City of Boston (DND), Clever Elephant Solutions, Cooperativa Alveo, Deimos Space, Faunalia, Geographic Data BC, Hunter Systems Group, Lidwala Consulting Engineers, LisaSoft, Logical Tracking & Tracing International AG, Maponics, Michigan Tech Research Institute, Natural Resources Canada, Norwegian Forest and Landscape Institute, Boundless (former OpenGeo), OSGeo, Oslandia, Palantir Technologies, Paragon Corporation, R3 GIS, Refractions Research, Regione Toscana - SITA, Safe Software, Sirius Corporation plc, Stadt Uster, UC Davis Center for Vectorborne Diseases, University of Laval, U.S Department of State (HIU), Zonar Systems

**Campanhas de financiamento coletivo** Crowd funding campaigns - Campanhas de financiamento de multidões são campanhas que executamos para obter os recursos que queremos para financiar um projeto por um grande número de pessoas. Cada campanha é especificamente focada em um recurso ou conjunto de recursos específicos. Cada patrocinador utiliza uma fração pequena do financiamento necessário e com o número suficiente de pessoas / organizações contribuindo, temos os fundos para pagar o trabalho que vai ajudar muitos. Se você tiver uma idéia de um recurso que você acha que muitos outros estariam dispostos a co-financiar, por favor, postar para o [newsgroup PostGIS](#) suas ideias, e juntos podemos fazer isso acontecer.

A versão 2.0.0 foi a primeira em que testamos esta estratégia. Utilizamos o [PledgeBank](#) e conseguimos realizar duas campanhas bem sucedidas.

**postgistopology** - 10 patrocinadores, cada um contribuiu com USD \$250,00 para a construção da função toTopoGeometry e melhorias gerais no suporte a topologia da versão 2.0.0. Aconteceu!

**postgis64windows** - 20 patrocinadores, contribuíram com \$100 USD cada, para pagar para a compilação do PostGIS no Windows 64bits. Aconteceu. Agora temos uma versão 64-bits do PostGIS 2.0.1 disponível na PostgreSQL Stack Builder.

**Bibliotecas importantes** A [GEOS](#), biblioteca geométrica e o trabalho em algoritmos de Martin Davis, manutenção atual de Mateusz Loskot, Sandro Santilli (strk), Paul Ramsey e outros.

A [GDAL](#), Geospatial Data Abstraction Library (Biblioteca de abstração de dados geoespaciais), por Frank Wamerdam e outros é utilizada para rodar muitas das funcionalidades raster introduzidas na versão 2.0.0. Em tempo, as melhorias necessárias na GDAL para suportar o PostGIS tem sido contribuídas de volta para o projeto.

A biblioteca [Proj4](#) e o trabalho de Gerald Evenden e Frank Wamerdam em sua criação e manutenção.

Por último, mas não menos importante, o [PostgreSQL DBMS](#), o gigante sobre qual o PostGIS se apóia. Muito da velocidade e flexibilidade do PostGIS não seria possível sem a extensibilidade, um grande analisador de consultas, índice GIST e uma variedade de funcionalidades SQL dadas pelos PostgreSQL.

## 1.5 Mais informações

- As últimas novidades, versões, documentação estão disponíveis no website do PostGIS, <http://postgis.net>.
- Mais informações sobre a biblioteca GEOS estão disponíveis em <http://trac.osgeo.org/geos/>.
- Mais informações sobre a biblioteca Proj4 estão disponíveis em <http://trac.osgeo.org/proj/>.
- Mais informações sobre o PostgreSQL estão disponíveis em <http://www.postgresql.org>.
- Mais informações sobre o índice GiST estão disponíveis no site de desenvolvimento do PostgreSQL em <http://www.sai.msu.su/~megeera/postgres/gist/>.
- Mais informações sobre o MapServer estão disponíveis em <http://mapserver.org>
- As especificações "[Simple Features for Specification for SQL](#)" estão disponíveis no site da OGC: <http://www.opengeospatial.org/>

## Chapter 2

# Instalação do PostGIS

Este capítulo detalha os passos necessários para instalar o PostGIS.

### 2.1 Versão Reduzida

Para compilar, assumindo que você tem todas as dependências em seu caminho de busca (search path):

```
tar xvfz postgis-2.5.0.tar.gz
cd postgis-2.5.0
./configure
make
make install
```

Assim que o PostGIS esteja instalado, ele precisa ser habilitado em cada banco de dados que você deseje utilizá-lo.

**Note**

O suporte a raster é opcional, mas é instalado por padrão. Para instalar utilizando o modelo PostgreSQL 9.1 ou maior, o suporte a raster é requerido. Utilizar a extensão é preferido e mais amigável. Para habilitar espacialmente seu banco de dados:

```
psql -d yourdatabase -c "CREATE EXTENSION postgis;"
psql -d yourdatabase -c "CREATE EXTENSION postgis_topology;"
-- if you built with sfcgal support --
psql -d yourdatabase -c "CREATE EXTENSION postgis_sfcgal;"

-- if you want to install tiger geocoder --
psql -d yourdatabase -c "CREATE EXTENSION fuzzystmatch"
psql -d yourdatabase -c "CREATE EXTENSION postgis_tiger_geocoder;"

-- if you installed with pcre
-- you should have address standardizer extension as well
psql -d yourdatabase -c "CREATE EXTENSION address_standardizer;"
```

Para maiores detalhes sobre pesquisa de extensões instaladas e disponíveis e atualizações, veja: [Section 2.4.3](#)

Para os que decidiram não instalar o suporte a raster ou preferem a versão antiga, aqui estão as instruções para você:

Todos os arquivos .sql depois de instalados estão disponíveis na pasta share/contrib/postgis-2.4 de sua instalação do PostgreSQL



```
createdb yourdatabase
createlang plpgsql yourdatabase
psql -d yourdatabase -f postgis.sql
psql -d yourdatabase -f postgis_comments.sql
psql -d yourdatabase -f spatial_ref_sys.sql
psql -d yourdatabase -f topology.sql
psql -d yourdatabase -f topology_comments.sql

-- only if you compiled with raster (GDAL)
psql -d yourdatabase -f rtpostgis.sql
psql -d yourdatabase -f raster_comments.sql

--if you built with sfcgal support --
psql -d yourdatabase -f sfcgal.sql
psql -d yourdatabase -f sfcgal_comments.sql
```

O restante deste capítulo entra em detalhes em cada uma das etapas de instalação acima.

Assim como no PostGIS 2.1.3, out-of-db raters e todos os drivers rasters são desativados por padrão. para reativá-los você precisa configurar as variáveis de ambiente a seguir `POSTGIS_GDAL_ENABLED_DRIVERS` e `POSTGIS_ENABLE_OUTDB_RASTERS` no ambiente do servidor. Para PostGIS 2.2, você pode usar a cross-platform mais próxima da configuração correspondente [Section 8.2](#).

Se quiser ativar o raster offline:

```
POSTGIS_ENABLE_OUTDB_RASTERS=1
```

Qualquer outra configuração ou nenhuma configuração vai desabilitar o banco de dados rasters.

Se quiser ativar os drivers GDAL disponíveis na sua instalação GDAL, configure esta variável de ambiente como segue:

```
POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL
```

Se você quiser ativar drivers específicos, configure sua variável de ambiente como segue:

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

**Note**

Se você está no windows , não cite a lista de driver

---

Setting environment variables varies depending on OS. For PostgreSQL installed on Ubuntu or Debian via `apt-postgresql`, the preferred way is to edit `/etc/postgresql/10/main/environment` where 9.3 refers to version of PostgreSQL and main refers to the cluster.

Se você está usando o Windows, você pode configurar via variáveis de sistema as quais, para o Windows 7, você pode conseguir clicando com botão direito em Computador->Propriedades Avançadas Configurações de Sistema ou navegando no explorador para Painel de Controle\Todos os Itens do Painel de Controle\ Sistema. Depois clicando em *Configurações Avançadas de Sistema* ->*Avançado*->*Variáveis de Ambiente* e adicionando novas variáveis no sistema.

Depois de configurar suas variáveis de ambiente, você precisará reiniciar seu serviço PostgreSQL para as mudanças acontecerem.

## 2.2 Instalando pacotes requeridos

PostGIS tem os seguintes requisitos para a construção e uso:

**Necessário**

- PostgreSQL 9.4 ou superior. A instalação completa do PostgreSQL (incluindo cabeçalhos de servidor) é necessária. PostgreSQL está disponível a partir do <http://www.postgresql.org> .  
Para uma matriz completa de suporte do PostgreSQL / PostGIS e do PostGIS/GEOS veja em <http://trac.osgeo.org/postgis/wiki/UsersWikiPostgreSQLPostGIS>
- Compilador GNU C ( `gcc`). Alguns outros compiladores ANSI C podem ser utilizados para compilar o PostGIS, mas nós encontramos menos problemas ao compilar com `gcc`.
- GNU Make (`gmake` ou `make`). Para varios sistemas, GNU `make` é a versão padrão do `make`. Verifique a versão invocando `make -v`. Outras versões do `make` pode não processar o PostGIS `Makefile` corretamente.
- Proj4 reprojection library, version 4.9.0 or greater. Proj4 4.9 or above is needed to take advantage of improved geodetic. The Proj4 library is used to provide coordinate reprojection support within PostGIS. Proj4 is available for download from <http://trac.osgeo.org/proj/> .
- GEOS geometry library, version 3.5 or greater, but GEOS 3.7+ is recommended to take full advantage of all the new functions and features. You should have at least GEOS 3.5, without which you will be missing some major enhancements such as `ST_ClipByBox2D` and `ST_Subdivide`. GEOS is available for download from <http://trac.osgeo.org/geos/> and 3.5+ is backward-compatible with older versions so fairly safe to upgrade.
- LibXML2, versão 2.5.x ou superior. LibXML2 é atualmente utilizado em algumas funções de importação (`ST_GeomFromGML` and `ST_GeomFromKML`). LibXML2 está disponível para baixar em <http://xmlsoft.org/downloads.html>.
- JSON-C, versão 0.9 ou maior. JSON-C é atualmente utilizado para importar GeoJSON através da função `ST_GeomFromGeoJson`. JSON-C está disponível para download em <https://github.com/json-c/json-c/releases/>.
- GDAL, versão 1.8 ou maior (1.9 ou maior é fortemente recomendado, já que algumas coisas não funcionam bem ou se comportam de maneira diferente das versões mais antigas). Isto é pré-requisito para suporte a raster e para instalar o PostGIS via `CREATE EXTENSION postgis`, e é muito recomendado para todos rodando o PostgreSQL 9.1 ou maior. <http://trac.osgeo.org/gdal/wiki/DownloadSource>.
- Este parâmetro está atualmente sem funcionalidade, já que o pacote somente irá instalar na localização do PostgreSQL. Visite <http://trac.osgeo.org/postgis/ticket/635> para acompanhar este bug.

## Opcional

- GDAL (pseudo opcional) somente se você não quer o suporte o raster e não se importa pela instalação através do comando `CREATE EXTENSION postgis`. Lembre-se que outras extensões podem requerer o PostGIS como uma extensão, que irá impedi-lo de instalá-las. É altamente recomendado que você compile o PostGIS com suporte a raster.  
Certifique-se também de ativar os dispositivos que deseja usar como está descrito em Section 2.1.
- GTK (requer GTK+2.0, 2.8+) para compilar o `shp2pgsql-gui` para formar o carregador de arquivo. <http://www.gtk.org/> .
- SFCGAL, versão 1.1 (ou superior) poderia ser usada para fornecer funções de análise 2D e 3D adicionais para o PostGIS cf Section 8.10. Também permite usar o SFCGAL além do GEOS para algumas funções 2D fornecidas por ambos (como `ST_Intersection` ou `ST_Area`, por enquanto). A variável de configuração `postgis.backend` do PostgreSQL permite ao usuário controlar qual backend pretende usar caso o SFCGAL estiver instalado (GEOS é o padrão). Nota: SFCGAL 1.2 depende pelo menos do CGAL 4.3 e Boost 1.54 (cf: <http://oslandia.github.io/SFCGAL/installation.html>) <https://github.com/Oslandia/SFCGAL>.
- Com a intenção de construir o Chapter 12 você também irá precisar do PCRE <http://www.pcre.org> (que normalmente já está instalado nos sistemas nix). `Regex::Assemble` o pacote perl CPAN só é necessário se quiser reconstruir os dados encoded em `parseaddress-stcities.h`. Chapter 12 vai automaticamente ser construída se ele detectar uma biblioteca PCRE, ou você passa em um válido `--with-pcre-dir=/path/to/pcre` durante a configuração.
- To enable `ST_AsMVT` `protobuf-c` library (for usage) and the `protoc-c` compiler (for building) are required. Also, `pkg-config` is required to verify the correct minimum version of `protobuf-c`. See [protobuf-c](#).
- CUnit (CUnit). Isto é necessário para o teste de regressão. <http://cunit.sourceforge.net/>

- DocBook (`xsltproc`) é necessário para a construção da documentação. Docbook esta disponível em <http://www.docbook.org/>.
- DBLatex (`dblatex`) é necessário para a construção da documentação em formato PDF. DBLatex está disponível em <http://dblatex.sourceforge.net/>.
- ImageMagick (`convert`) é necessário para gerar as imagens usadas na documentação. ImageMagick está disponível em <http://www.imagemagick.org/>.

## 2.3 Obtendo o Fonte

Obtenha o fonte do PostGIS através da seção de downloads do website <http://download.osgeo.org/postgis/source/postgis-2.5.0.tar.gz>

```
wget http://download.osgeo.org/postgis/source/postgis-2.5.0.tar.gz
tar -xvzf postgis-2.5.0.tar.gz
```

Isto irá criar um diretório chamado `postgis-2.5.0` no diretório de trabalho atual.

Outra alternativa, é o checkout da fonte do `svn` repository <http://svn.osgeo.org/postgis/trunk/>.

```
svn checkout http://svn.osgeo.org/postgis/trunk/ postgis-2.5.0
```

Mude para o recém criado `postgis-2.5.0` diretório para continuar a instalação.

## 2.4 Compilando e instalando da fonte: detalhado

### Note

Muitos sistemas operacionais agora incluem pacotes pré-compilados para PostgreSQL / PostGIS. Em muitos casos, a compilação só é necessário se você quiser as versões ponta ou você é um mantenedor do pacote.

Esta seção inclui instruções gerais de compilação, se você está compilando para Windows etc ou outro sistema operacional, você pode encontrar ajuda mais detalhada adicional no [PostGIS User contributed compile guides](#) e [PostGIS Dev Wiki](#).

Pacotes pré-instalados para vários SO estão listados no [PostGIS Pre-built Packages](#)

Se você é um usuário windows, você pode obter builds estáveis via Stackbuilder [PostGIS Windows download site](#) Também [builds experimentais para windows](#) são builds lançadas geralmente uma ou duas vezes por semana ou sempre que algo emocionante acontece. Você pode usá-los para experimentar os lançamentos em progresso de PostGIS

O módulo PostGIS é uma extensão para o servidor PostgreSQL. Além disso, PostGIS 2.5.0 *requer* acesso completo ao PostgreSQL para compilação. Isso pode ser feito no PostgreSQL 9.4 ou superior. Versões anteriores *não* são compatíveis.

Refere-se ao guia de instalação do PostgreSQL se você ainda não tiver instalado o PostgreSQL <http://www.postgresql.org>.

### Note

Para funcionalidade da GEOS, quando você instalar o PostgreSQL você pode ter que linkar explicitamente o PostgreSQL contra a biblioteca padrão C++:

```
LDFLAGS=-lstdc++ ./configure [SUAS OPÇÕES AQUI]
```

Isto é uma forma de contornar as exceções falso-positivas da interação do C++ com ferramentas de desenvolvimento mais antigas. Se você experimentar problemas estranho (backend fechando de forma inesperada ou coisas similares), tente este truque. Isto irá requerir que você compile o PostgreSQL do zero, claro.

Os passos a seguir demonstram a configuração e compilação dos fontes do PostGIS. Eles são escritos para usuários de Linux e não funcionarão em Windows ou Mac.

## 2.4.1 Configuração

Como a maior parte das instalações Linux, o primeiro passo é gerar o Makefile que será utilizado para construção do código fonte. Isto é feito utilizando o script shell

### **./configure**

Sem parâmetros adicionais, este comando tentará automaticamente localizar os componentes necessários e bibliotecas para construção do fonte do PostGIS em seu sistema. Embora esta é a forma comum de uso do **./configure**, o script aceita diversos parâmetros para aqueles que tem as bibliotecas e programas necessários em localizações do sistema operacional que não são padrão.

A lista a seguir mostra apenas os parâmetros comumente utilizados. Para uma lista completa, utilize os parâmetros **--help** ou **--help=short**.

**--prefix=PREFIX** Esta é a localização onde as bibliotecas do PostGIS e scripts SQL serão instalados. Por padrão, esta localização é a mesma detectada pela instalação do PostgreSQL.



#### **Caution**

Este parâmetro está atualmente sem funcionalidade, já que o pacote somente irá instalar na localização do PostgreSQL. Visite <http://trac.osgeo.org/postgis/ticket/635> para acompanhar este bug.

- with-pgconfig=FILE** O PostgreSQL oferece um utilitário chamado **pg\_config** para habilitar extensões como o PostGIS a localizar a instalação do PostgreSQL. Use o parâmetro (**--with-pgconfig=/path/to/pg\_config** para especificar manualmente uma instalação específica do PostgreSQL que será usada pelo PostGIS.
- with-gdalconfig=FILE** GDAL, uma biblioteca requerida, provê funcionalidades necessárias para o suporte a raster. Use o comando **gdal-config** para localizar o diretório de instalação da GDAL. Use este parâmetro (**--with-gdalconfig=/path/to/gdal-config**) para manualmente especificar uma instalação em particular da GDAL que o PostGIS irá utilizar.
- with-geosconfig=FILE** GEOS é uma biblioteca requerida, dá um utilitário chamado **geos-config** para localizar o diretório de instalação da GEOS. Use este parâmetro (**--with-geosconfig=/path/to/geos-config**) para especificar manualmente uma instalação da GEOS que o PostGIS irá utilizar.
- with-xml2config=FILE** LibXML é a biblioteca exigida para fazer os processos GeomFormKML/GML. É encontrada normalmente se você tem o libxml instalado, mas se não tiver ou quiser uma versão específica usada, você precisará apontar o PostGIS para um `xml2-config` específico para ativar as instalações de software para localizar a lista de instalação do LibXML. Use esse parâmetro (**--with-xml2config=/path/to/xml2-config**) para especificar manualmente uma instalação do LibXML que o PostGIS irá construir contra.
- with-projdir=DIR** A Proj4 é uma biblioteca pra reprojeção de coordenadas, na qual o PostGIS depende. Use este parâmetro (**--with-projdir=/path/to/projdir** para especificar manualmente uma instalação do Proj4 que o PostGIS irá utilizar para compilação.
- with-libiconv=DIR** Diretório onde o iconv esta instalado.
- with-jsondir=DIR** **JSON-C** é uma biblioteca MIT-licensed JSON exigida pelo suporte PostGIS `ST_GeomFromJSON`. Use esse parâmetro (**--with-jsondir=/path/to/jsondir**) para especificar manualmente uma instalação do JSON-C que o PostGIS irá construir contra.
- with-pcredir=DIR** **PCRE** é uma biblioteca BSD-licensed Perl Compatible Regular Expression requerida pela extensão `address_standardizer`. Use esse parâmetro (**--with-pcredir=/path/to/pcredir**) para especificar manualmente uma instalação do PCRE que o PostGIS irá construir contra.
- with-gui** Compile a interface de usuário para importação de dados (requer GTK+2.0). Isto irá criar a ferramenta de interface gráfica `shp2pgsql-gui` para o utilitário `shp2pgsql`.
- with-raster** Compilar com o suporte raster. Isto irá compilar a biblioteca `rtpostgis-2.5.0` e o arquivo `rtpostgis.sql`. Isto não é requerido e na versão final de lançamento o plano é compilar o suporte raster por default.

- without-topology** Disable topology support. There is no corresponding library as all logic needed for topology is in postgis-2.5.0 library.
- with-gettext=no** Por padrão o PostGIS vai tentar detectar o suporte gettext e compilar com ele, porém se você tiver problemas incompatíveis que causem dano de carregamento, você pode o desabilitar com esse comando. Referir-se ao ticket <http://trac.osgeo.org/postgis/ticket/748> para um exemplo de problema resolvido configurando com este. NOTA: que você não está perdendo muito desligando isso. É usado para ajuda internacional para o carregador GUI que ainda não está documentado e permanece experimental.
- with-sfcgal=PATH** Por padrão PostGIS não tentará instalar com suporte sfcgal sem esta mudança. PATH é um argumento opcional que permite especificar um PATH alternativo para sfcgal-config.

---

**Note**

Se conseguiu o PostGIS do SVN [depósito](#), o primeiro passo é fazer funcionar o script

`./autogen.sh`

Este script gera a **configurar** script que na volta é usada para personalizar a instalação do PostGIS.

Se em vez de conseguir o PostGIS como tarball, rodando `./autogen.sh` não é necessariamente como **configurar** já foi gerado.

---

## 2.4.2 Construindo

Uma vez que o Makefile tenha sido gerado, compilar o PostGIS é simples como rodar o comando

**make**

A última linha da saída deve ser "PostGIS was built successfully. Ready to install.."

As of PostGIS v1.4.0, all the functions have comments generated from the documentation. If you wish to install these comments into your spatial databases later, run the command which requires docbook. The `postgis_comments.sql` and other package comments files `raster_comments.sql`, `topology_comments.sql` are also packaged in the tar.gz distribution in the doc folder so no need to make comments if installing from the tar ball. Comments are also included as part of the CREATE EXTENSION install.

**fazer comentários**

Apresentado ao PostGIS 2.0. Isto gera html cheat sheets adequadas para referências rápidas ou para handouts dos estudantes. Exige xsltproc para construir e vai gerar 4 arquivos no folder do documento `topology_cheatsheet.html`, `tiger_geocoder_cheatsheet.html`, `raster_cheatsheet.html`, `postgis_cheatsheet.html`

Você pode baixar alguns pre-construídos disponíveis em html e pdf de [PostGIS / PostgreSQL Study Guides](#)

**faça anotações**

## 2.4.3 Construindo extensões PostGIS e implantado-as

As extensões do PostGIS são contruídas e instaladas automaticamente se você estiver usando PostgreSQL 9.1 ou superior.

Se você está compilando do repositório, você precisa de compilar a função de descrições primeiro. Estas são compiladas se você possui o docbook instalado. Você pode também construir manualmente com o comando:

**fazer comentários**

Construir a documentação não é necessário se você está construindo de uma versão de lançamento no formato tar ball, já que estas são empacotadas pré-construídas com o tar ball.

Se você está construindo o PostGIS contra o PostgreSQL 9.1, as extensão devem ser automaticamente construídas como parte do processo de make. Você pode, contudo, se necessário, construir das pastas de extensões ou copiar os arquivos se você precisar dos mesmos em um servidor diferente.

---

```

cd extensions
cd postgis
make clean
make
make install
cd ..
cd postgis_topology
make clean
make
make install
cd ..
cd postgis_sfcgal
make clean
make
make install

cd ..
cd address_standardizer
make clean
make
make install
make installcheck

cd ..
cd postgis_tiger_geocoder
make clean
make
make install
make installcheck

```

Os arquivos de extensões sempre serão os mesmos para a mesma versão do PostGIS, independente do Sistema Operacional, então é fácil copiar os arquivos de extensão de um sistema operacional para outro, desde que você tenha os binários do PostGIS instalados em seus servidores.

Se você deseja instalar as extensões manualmente em um servidor diferente, do seu servidor de desenvolvimento, você precisará copiar os seguintes arquivos da pasta de extensões para a pasta `PostgreSQL /share/extension` da sua instalação do PostgreSQL, bem como os binários necessários para o PostGIS, se você não os tem ainda no servidor de destino.

- Existe arquivos de controle que denotam informações como a versão da extensão a ser instalada, caso não seja especificada. `postgis.control`, `postgis_topology.control`.
- Todos os arquivos na pasta `/sql` de cada extensão. Note que estes precisam ser copiados para a raiz da pasta `share/extension` do PostgreSQL `extensions/postgis/sql/*.sql`, `extensions/postgis_topology/sql/*.sql`

Quando você finalizar este processo, você deverá ver `postgis`, `postgis_topology` como extensões disponíveis no PgAdmin -> Extensões.

Se você está utilizando `psql`, pode verificar quais extensões estão instaladas executando essa query:

```

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

```

name	default_version	installed_version
address_standardizer	2.5.0	2.5.0
address_standardizer_data_us	2.5.0	2.5.0
postgis	2.5.0	2.5.0
postgis_sfcgal	2.5.0	
postgis_tiger_geocoder	2.5.0	2.5.0
postgis_topology	2.5.0	

(6 rows)

Se você tem a extensão instalada no banco de dados de seu interesse, você a verá mencionada na coluna `installed_version`. Se você não receber nenhum registro de volta, significa que você não tem extensões do PostGIS instaladas no servidor. PgAdmin III 1.14+ também irá lhe dar esta informação na seção `extensions` do navegador de banco de dados e até permitirá o upgrade ou a desinstalação utilizando o clique com o botão direito.

Se você tem extensões disponíveis, pode instalar a extensão `postgis` no seu database escolhido usando a interface da extensão `pgAdmin` ou rodando esses comandos `sql`:

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

No `psql` você pode ver quais versões foram instaladas e qual esquema eles estão instalando.

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

```
List of installed extensions
-[ RECORD 1 ]-----
Name          | postgis
Version       | 2.5.0
Schema        | public
Description   | PostGIS geometry, geography, and raster spat..
-[ RECORD 2 ]-----
Name          | postgis_tiger_geocoder
Version       | 2.5.0
Schema        | tiger
Description   | PostGIS tiger geocoder and reverse geocoder
-[ RECORD 3 ]-----
Name          | postgis_topology
Version       | 2.5.0
Schema        | topology
Description   | PostGIS topology spatial types and functions
```

### Warning



Extensões `table spatial_ref_sys`, `layer`, `topology` não podem ter o backup feito explicitamente. Só podem ser feito o backup quando a respectiva `postgis` ou `postgis_topology` extensão estiver com o backup feito, o que só acontece quando você faz backup de todo o database. Assim como PostGIS 2.0.1, somente os registros `srid` não compactados com o PostGIS tem o backup quando há o backup do database, então não faça mudanças nos `srids` que nós compactamos e espere que suas mudanças estejam lá. Coloque em um em um bilhete se encontrar algum problema. As estruturas de extensões `table` nunca têm o backup feito desde que elas são criadas com `CREATE EXTENSION` e supostas a serem as mesmas para uma dada versão de uma extensão. Estes comportamentos são construídos na extensão atual do PostgreSQL, portanto não há nada que possamos fazer a respeito.

Se você instalou 2.5.0 sem usar nosso sistema de extensão maravilhoso, você pode mudar para uma extensão baseada em primeiro atualizando para a última micro versão rodando as scripts atualizadas: `postgis_upgrade_22_minor.sql`, `raster_upgrade_22_minor.sql`, `topology_upgrade_22_minor.sql`.

Se você instalou `postgis` sem o auxílio `raster`, você vai precisar instalar o auxílio `raster` primeiro (usando o completo `rtpostgis.sql`

Então você poderá rodar os comandos abaixo para compactar as funções nas suas respectivas extensões.

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

## 2.4.4 Testando

Se desejar testar o PostGIS, rode

### make check

O comando acima irá rodar através de várias verificações e testes de regressão usando a biblioteca gerada contra o database do PostgreSQL atual.



#### Note

Se você configurou o PostGIS usando o não padronizado PostgreSQL, GEOS, ou Proj4 localizações, talvez você precise adicionar a biblioteca de localizações deles à LD\_LIBRARY\_PATH variável de ambiente.



#### Caution

Atualmente, o **faz verificação** confia nas variáveis de ambiente PATH e PGPORT quando vai fazer as verificações - ele *não* usa a versão PostgreSQL que talvez tenha sido especificada utilizando o parâmetro de configuração **--with-pgconfig**. Portanto, certifique-se que para modificar seu PATH para ser compatível com a instalação do PostgreSQL detectada durante a configuração ou esteja preparado para iminentes aborrecimentos.

Se tiver êxito, a saída do teste deve ser parecida com a subsequente:

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

```
Suite: computational_geometry
Test: test_lw_segment_side ...passed
Test: test_lw_segment_intersects ...passed
Test: test_lwline_crossing_short_lines ...passed
Test: test_lwline_crossing_long_lines ...passed
Test: test_lwline_crossing_bugs ...passed
Test: test_lwpoint_set_ordinate ...passed
Test: test_lwpoint_get_ordinate ...passed
Test: test_point_interpolate ...passed
Test: test_lwline_clip ...passed
Test: test_lwline_clip_big ...passed
Test: test_lwmline_clip ...passed
Test: test_geohash_point ...passed
Test: test_geohash_precision ...passed
Test: test_geohash ...passed
Test: test_geohash_point_as_int ...passed
Test: test_isclosed ...passed
Test: test_lwgeom_simplify ...passed
Suite: buildarea
Test: buildarea1 ...passed
Test: buildarea2 ...passed
Test: buildarea3 ...passed
Test: buildarea4 ...passed
Test: buildarea4b ...passed
Test: buildarea5 ...passed
```



```
Test: buildarea6 ...passed
Test: buildarea7 ...passed
Suite: geometry_clean
Test: test_lwgeom_make_valid ...passed
Suite: clip_by_rectangle
Test: test_lwgeom_clip_by_rect ...passed
Suite: force_sfs
Test: test_sfs_11 ...passed
Test: test_sfs_12 ...passed
Test: test_sqlmm ...passed
Suite: geodetic
Test: test_sphere_direction ...passed
Test: test_sphere_project ...passed
Test: test_lwgeom_area_sphere ...passed
Test: test_signum ...passed
Test: test_gbox_from_spherical_coordinates ...passed
Test: test_serialized_get_gbox_geocentric ...passed
Test: test_clairaut ...passed
Test: test_edge_intersection ...passed
Test: test_edge_intersects ...passed
Test: test_edge_distance_to_point ...passed
Test: test_edge_distance_to_edge ...passed
Test: test_lwgeom_distance_sphere ...passed
Test: test_lwgeom_check_geodetic ...passed
Test: test_gserialized_from_lwgeom ...passed
Test: test_spheroid_distance ...passed
Test: test_spheroid_area ...passed
Test: test_lwpoly_covers_point2d ...passed
Test: test_gbox_utils ...passed
Test: test_vector_angle ...passed
Test: test_vector_rotate ...passed
Test: test_lwgeom_segmentize_sphere ...passed
Test: test_ptarray_contains_point_sphere ...passed
Test: test_ptarray_contains_point_sphere_iowa ...passed
Suite: GEOS
Test: test_geos_noop ...passed
Test: test_geos_subdivide ...passed
Test: test_geos_linemerge ...passed
Suite: Clustering
Test: basic_test ...passed
Test: nonsequential_test ...passed
Test: basic_distance_test ...passed
Test: single_input_test ...passed
Test: empty_inputs_test ...passed
Suite: Clustering Union-Find
Test: test_unionfind_create ...passed
Test: test_unionfind_union ...passed
Test: test_unionfind_ordered_by_cluster ...passed
Suite: homogenize
Test: test_coll_point ...passed
Test: test_coll_line ...passed
Test: test_coll_poly ...passed
Test: test_coll_coll ...passed
Test: test_geom ...passed
Test: test_coll_curve ...passed
Suite: encoded_polyline_input
Test: in_encoded_polyline_test_geoms ...passed
Test: in_encoded_polyline_test_precision ...passed
Suite: geojson_input
Test: in_geojson_test_srid ...passed
Test: in_geojson_test_bbox ...passed
Test: in_geojson_test_geoms ...passed
```

```
Suite: twkb_input
  Test: test_twkb_in_point ...passed
  Test: test_twkb_in_linestring ...passed
  Test: test_twkb_in_polygon ...passed
  Test: test_twkb_in_multipoint ...passed
  Test: test_twkb_in_multilinestring ...passed
  Test: test_twkb_in_multipolygon ...passed
  Test: test_twkb_in_collection ...passed
  Test: test_twkb_in_precision ...passed
Suite: serialization/deserialization
  Test: test_typmod_macros ...passed
  Test: test_flags_macros ...passed
  Test: test_serialized_srid ...passed
  Test: test_gserialized_from_lwgeom_size ...passed
  Test: test_gbox_serialized_size ...passed
  Test: test_lwgeom_from_gserialized ...passed
  Test: test_lwgeom_count_vertices ...passed
  Test: test_on_gser_lwgeom_count_vertices ...passed
  Test: test_geometry_type_from_string ...passed
  Test: test_lwcollection_extract ...passed
  Test: test_lwgeom_free ...passed
  Test: test_lwgeom_flip_coordinates ...passed
  Test: test_f2d ...passed
  Test: test_lwgeom_clone ...passed
  Test: test_lwgeom_force_clockwise ...passed
  Test: test_lwgeom_calculate_gbox ...passed
  Test: test_lwgeom_is_empty ...passed
  Test: test_lwgeom_same ...passed
  Test: test_lwline_from_lwmpoint ...passed
  Test: test_lwgeom_as_curve ...passed
  Test: test_lwgeom_scale ...passed
  Test: test_gserialized_is_empty ...passed
  Test: test_gbox_same_2d ...passed
Suite: measures
  Test: test_mindistance2d_tolerance ...passed
  Test: test_rect_tree_contains_point ...passed
  Test: test_rect_tree_intersects_tree ...passed
  Test: test_lwgeom_segmentize2d ...passed
  Test: test_lwgeom_locate_along ...passed
  Test: test_lw_dist2d_pt_arc ...passed
  Test: test_lw_dist2d_seg_arc ...passed
  Test: test_lw_dist2d_arc_arc ...passed
  Test: test_lw_arc_length ...passed
  Test: test_lw_dist2d_pt_ptarrayarc ...passed
  Test: test_lw_dist2d_ptarray_ptarrayarc ...passed
  Test: test_lwgeom_tcpa ...passed
  Test: test_lwgeom_is_trajectory ...passed
Suite: effectivearea
  Test: do_test_lwgeom_effectivearea_lines ...passed
  Test: do_test_lwgeom_effectivearea_polys ...passed
Suite: miscellaneous
  Test: test_misc_force_2d ...passed
  Test: test_misc_simplify ...passed
  Test: test_misc_count_vertices ...passed
  Test: test_misc_area ...passed
  Test: test_misc_wkb ...passed
  Test: test_grid ...passed
Suite: nodding
  Test: test_lwgeom_node ...passed
Suite: encoded_polyline_output
  Test: out_encoded_polyline_test_geoms ...passed
  Test: out_encoded_polyline_test_srid ...passed
```

```
Test: out_encoded_polyline_test_precision ...passed
Suite: geojson_output
Test: out_geojson_test_precision ...passed
Test: out_geojson_test_dims ...passed
Test: out_geojson_test_srid ...passed
Test: out_geojson_test_bbox ...passed
Test: out_geojson_test_geoms ...passed
Suite: gml_output
Test: out_gml_test_precision ...passed
Test: out_gml_test_srid ...passed
Test: out_gml_test_dims ...passed
Test: out_gml_test_geodetic ...passed
Test: out_gml_test_geoms ...passed
Test: out_gml_test_geoms_prefix ...passed
Test: out_gml_test_geoms_nodims ...passed
Test: out_gml2_extent ...passed
Test: out_gml3_extent ...passed
Suite: kml_output
Test: out_kml_test_precision ...passed
Test: out_kml_test_dims ...passed
Test: out_kml_test_geoms ...passed
Test: out_kml_test_prefix ...passed
Suite: svg_output
Test: out_svg_test_precision ...passed
Test: out_svg_test_dims ...passed
Test: out_svg_test_relative ...passed
Test: out_svg_test_geoms ...passed
Test: out_svg_test_srid ...passed
Suite: x3d_output
Test: out_x3d3_test_precision ...passed
Test: out_x3d3_test_geoms ...passed
Test: out_x3d3_test_option ...passed
Suite: ptarray
Test: test_ptarray_append_point ...passed
Test: test_ptarray_append_ptarray ...passed
Test: test_ptarray_locate_point ...passed
Test: test_ptarray_isccw ...passed
Test: test_ptarray_signed_area ...passed
Test: test_ptarray_unstroke ...passed
Test: test_ptarray_insert_point ...passed
Test: test_ptarray_contains_point ...passed
Test: test_ptarrayarc_contains_point ...passed
Test: test_ptarray_scale ...passed
Suite: printing
Test: test_lwprint_default_format ...passed
Test: test_lwprint_format_orders ...passed
Test: test_lwprint_optional_format ...passed
Test: test_lwprint_oddball_formats ...passed
Test: test_lwprint_bad_formats ...passed
Suite: SFCGAL
Test: test_sfcgal_noop ...passed
Suite: split
Test: test_lwline_split_by_point_to ...passed
Test: test_lwgeom_split ...passed
Suite: stringbuffer
Test: test_stringbuffer_append ...passed
Test: test_stringbuffer_aprintf ...passed
Suite: surface
Test: triangle_parse ...passed
Test: tin_parse ...passed
Test: polyhedralsurface_parse ...passed
Test: surface_dimension ...passed
```

```
Suite: Internal Spatial Trees
  Test: test_tree_circ_create ...passed
  Test: test_tree_circ_pip ...passed
  Test: test_tree_circ_pip2 ...passed
  Test: test_tree_circ_distance ...passed
  Test: test_tree_circ_distance_threshold ...passed
Suite: triangulate
  Test: test_lwgeom_delaunay_triangulation ...passed
Suite: twkb_output
  Test: test_twkb_out_point ...passed
  Test: test_twkb_out_linestring ...passed
  Test: test_twkb_out_polygon ...passed
  Test: test_twkb_out_multipoint ...passed
  Test: test_twkb_out_multilinestring ...passed
  Test: test_twkb_out_multipolygon ...passed
  Test: test_twkb_out_collection ...passed
  Test: test_twkb_out_idlist ...passed
Suite: varint
  Test: test_zigzag ...passed
  Test: test_varint ...passed
  Test: test_varint_roundtrip ...passed
Suite: wkb_input
  Test: test_wkb_in_point ...passed
  Test: test_wkb_in_linestring ...passed
  Test: test_wkb_in_polygon ...passed
  Test: test_wkb_in_multipoint ...passed
  Test: test_wkb_in_multilinestring ...passed
  Test: test_wkb_in_multipolygon ...passed
  Test: test_wkb_in_collection ...passed
  Test: test_wkb_in_circularstring ...passed
  Test: test_wkb_in_compoundcurve ...passed
  Test: test_wkb_in_curvpolygon ...passed
  Test: test_wkb_in_multicurve ...passed
  Test: test_wkb_in_multisurface ...passed
  Test: test_wkb_in_malformed ...passed
Suite: wkb_output
  Test: test_wkb_out_point ...passed
  Test: test_wkb_out_linestring ...passed
  Test: test_wkb_out_polygon ...passed
  Test: test_wkb_out_multipoint ...passed
  Test: test_wkb_out_multilinestring ...passed
  Test: test_wkb_out_multipolygon ...passed
  Test: test_wkb_out_collection ...passed
  Test: test_wkb_out_circularstring ...passed
  Test: test_wkb_out_compoundcurve ...passed
  Test: test_wkb_out_curvpolygon ...passed
  Test: test_wkb_out_multicurve ...passed
  Test: test_wkb_out_multisurface ...passed
  Test: test_wkb_out_polyhedralsurface ...passed
Suite: wkt_input
  Test: test_wkt_in_point ...passed
  Test: test_wkt_in_linestring ...passed
  Test: test_wkt_in_polygon ...passed
  Test: test_wkt_in_multipoint ...passed
  Test: test_wkt_in_multilinestring ...passed
  Test: test_wkt_in_multipolygon ...passed
  Test: test_wkt_in_collection ...passed
  Test: test_wkt_in_circularstring ...passed
  Test: test_wkt_in_compoundcurve ...passed
  Test: test_wkt_in_curvpolygon ...passed
  Test: test_wkt_in_multicurve ...passed
  Test: test_wkt_in_multisurface ...passed
```

```

Test: test_wkt_in_tin ...passed
Test: test_wkt_in_polyhedralsurface ...passed
Test: test_wkt_in_errlocation ...passed
Suite: wkt_output
Test: test_wkt_out_point ...passed
Test: test_wkt_out_linestring ...passed
Test: test_wkt_out_polygon ...passed
Test: test_wkt_out_multipoint ...passed
Test: test_wkt_out_multilinestring ...passed
Test: test_wkt_out_multipolygon ...passed
Test: test_wkt_out_collection ...passed
Test: test_wkt_out_circularstring ...passed
Test: test_wkt_out_compoundcurve ...passed
Test: test_wkt_out_curvpolygon ...passed
Test: test_wkt_out_multicurve ...passed
Test: test_wkt_out_multisurface ...passed

Run Summary:
  Type      Total      Ran Passed Failed Inactive
  suites      38       38   n/a     0       0
  tests     251     251   251     0       0
  asserts  2468   2468  2468     0     n/a

Elapsed time = 0.298 seconds

Creating database 'postgis_reg'
Loading PostGIS into 'postgis_reg'
  /projects/postgis/branches/2.2/regress/00-regress-install/share/contrib/postgis/postgis. ←
  sql
  /projects/postgis/branches/2.2/regress/00-regress-install/share/contrib/postgis/ ←
  postgis_comments.sql
Loading SFCGAL into 'postgis_reg'
  /projects/postgis/branches/2.2/regress/00-regress-install/share/contrib/postgis/sfcgal. ←
  sql
  /projects/postgis/branches/2.2/regress/00-regress-install/share/contrib/postgis/ ←
  sfcgal_comments.sql
PostgreSQL 9.4.4, compiled by Visual C++ build 1800, 32-bit
Postgis 2.2.0dev - r13980 - 2015-08-23 06:13:07
scripts 2.2.0dev r13980
GEOS: 3.5.0-CAPI-1.9.0 r4088
PROJ: Rel. 4.9.1, 04 March 2015
SFCGAL: 1.1.0

Running tests

loader/Point ..... ok
loader/PointM ..... ok
loader/PointZ ..... ok
loader/MultiPoint ..... ok
loader/MultiPointM ..... ok
loader/MultiPointZ ..... ok
loader/Arc ..... ok
loader/ArcM ..... ok
loader/ArcZ ..... ok
loader/Polygon ..... ok
loader/PolygonM ..... ok
loader/PolygonZ ..... ok
loader/TSTPolygon ..... ok
loader/TSTIPolygon ..... ok
loader/TSTIPolygon ..... ok
loader/PointWithSchema ..... ok
loader/NoTransPoint ..... ok
loader/NotReallyMultiPoint ..... ok

```

```
loader/MultiToSinglePoint ..... ok
loader/ReprojectPts ..... ok
loader/ReprojectPtsGeog ..... ok
loader/Latin1 .... ok
loader/Latin1-implicit .... ok
loader/mfile .... ok
dumper/literalsrid ..... ok
dumper/realtable ..... ok
affine .. ok
bestsrid .. ok
binary .. ok
boundary .. ok
cluster .. ok
concave_hull .. ok
ctors .. ok
dump .. ok
dumppoints .. ok
empty .. ok
forcecurve .. ok
geography .. ok
in_geohash .. ok
in_gml .. ok
in_kml .. ok
in_encodedpolyline .. ok
iscollection .. ok
legacy .. ok
long_xact .. ok
lwgeom_regress .. ok
measures .. ok
operators .. ok
out_geometry .. ok
out_geography .. ok
polygonize .. ok
polyhedralsurface .. ok
postgis_type_name .. ok
regress .. ok
regress_bdpoly .. ok
regress_index .. ok
regress_index_nulls .. ok
regress_management .. ok
regress_selectivity .. ok
regress_lrs .. ok
regress_ogc .. ok
regress_ogc_cover .. ok
regress_ogc_prep .. ok
regress_proj .. ok
relate .. ok
remove_repeated_points .. ok
removepoint .. ok
setpoint .. ok
simplify .. ok
simplifyvw .. ok
size .. ok
snaptogrid .. ok
split .. ok
sql-mm-serialize .. ok
sql-mm-circularstring .. ok
sql-mm-compoundcurve .. ok
sql-mm-curvepoly .. ok
sql-mm-general .. ok
sql-mm-multicurve .. ok
sql-mm-multisurface .. ok
```

```

swapordinates .. ok
summary .. ok
temporal .. ok
tickets .. ok
twkb .. ok
typmod .. ok
wkb .. ok
wkt .. ok
wmsservers .. ok
knn .. ok
hausdorff .. ok
regress_buffer_params .. ok
offsetcurve .. ok
relatemark .. ok
isvaliddetail .. ok
sharedpaths .. ok
snap .. ok
node .. ok
unaryunion .. ok
clean .. ok
relate_bnr .. ok
delaunaytriangles .. ok
clipbybox2d .. ok
subdivide .. ok
in_geojson .. ok
regress_sfcgal .. ok
sfcgal/empty .. ok
sfcgal/geography .. ok
sfcgal/legacy .. ok
sfcgal/measures .. ok
sfcgal/regress_ogc_prep .. ok
sfcgal/regress_ogc .. ok
sfcgal/regress .. ok
sfcgal/tickets .. ok
sfcgal/concave_hull .. ok
sfcgal/wmsservers .. ok
sfcgal/approximatemedialaxis .. ok
uninstall . /projects/postgis/branches/2.2/regress/00-regress-install/share/contrib/ ↔
  postgres/uninstall_sfcgal.sql
  /projects/postgis/branches/2.2/regress/00-regress-install/share/contrib/postgis/ ↔
  uninstall_postgis.sql
. ok (4336)

```

Run tests: 118

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/

```

Suite: Shapefile Loader File shp2pgsql Test

Test: test\_ShpgLoaderCreate() ...passed

Test: test\_ShpgLoaderDestroy() ...passed

Suite: Shapefile Loader File postgres2shp Test

Test: test\_ShpgDumperCreate() ...passed

Test: test\_ShpgDumperDestroy() ...passed

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
```

As extensões `postgis_tiger_geocoder` and `address_standardizer` , atualmente só suportam o modelo PostgreSQL `installcheck`. Para testá-los use abaixo. Nota: fazer a instalação não é necessária se você já instalou na raiz do folder code do PostGIS.

Para `address_standardizer`:

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

Saída deve parecer com:

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

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

Para o geocoder tiger, certifique-se que você tem extensões `portgis` e `fuzzystratch` disponíveis no seu PostgreSQL. Os testes `address_standardizer` também irão desprezar se seus `postgis` construídos com `address_standardizer` suportar:

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

saída deve parecer com:

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

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



## 2.4.5 Instalação

Para instalar o PostGIS, digite

### **make install**

Isso irá copiar a instalação dos arquivos do PostGIS para suas subdireções específicas pelo **--prefix** parâmetro de configuração. Particularmente:

- Os binários do carregador e do dumper estão instalados no `[prefix]/bin`.
- Os arquivos SQL, como `postgis.sql`, estão instalados em `[prefix]/share/contrib`.
- As bibliotecas do PostGIS estão instaladas em `[prefix]/lib`.

Se anteriormente você rodou o comando **make comments** para gerar o arquivo `postgis_comments.sql`, `raster_comments.sql`, instale o arquivo sql para executar

### **make comments-install**



#### **Note**

`postgis_comments.sql`, `raster_comments.sql`, `topology_comments.sql` foi separado da construção típica e instalações alvo desde que como isso, veio a dependência extra do **xsltproc**.

## 2.5 Criando uma base de dados espacial usando EXTENSÕES

Se está usando o PostgreSQL 9.1+ e compilou e instalou as extensões/módulos `postgis`, você pode criar um banco de dados espacial de uma nova forma.

### **createdb [seubancodedados]**

A extensão central `postgis` instala a geometria PostGIS, geografia, raster, `spatial_ref_sys` e todas as funções e comentários com um simples comando:

```
CRIAR EXTENSÃO postgis;
```

```
psql -d [SEU_BANCO_DE_DADOS] -c "CREATE EXTENSION postgis;"
```

Topologia é compactada como uma extensão separada e instalável com o comando:

```
psql -d [SEU_BANCO_DE_DADOS] -c "CREATE EXTENSION postgis_topology;"
```

Se planeja restaurar um backup antigo das primeiras versões nesse novo db, faça funcionar:

```
psql -d [yourdatabase] -f legacy.sql
```



#### **Note**

If you need legacy functions, you'll need to reinstall the `legacy.sql` script whenever you upgrade the minor version of PostGIS. E.g. if you upgraded from 2.4.3 to 2.5.0, then you need to reinstall the `legacy.sql` packaged with 2.5.0. This is because some of the functions make reference to the library and the library is named with the minor in it.

Você pode usar `uninstall_legacy.sql` para livrar-se das funções menosprezadas depois de terminar de restaurar e limpar.

## 2.6 Create a spatially-enabled database without using extensions



### Note

This is generally only needed if you built PostGIS without raster support. Since raster functions are part of the `postgis` extension, extension support is not enabled if PostGIS is built without raster.

O primeiro passo para criar um banco de dados do PostGIS é criar um simples banco de dados PostgreSQL.

**createdb [seubancodedados]**

A maioria das funções do PostGIS são escritas na linguagem processual PL/pgSQL. Sendo assim, o próximo passo para criar o banco de dados do PostGIS é ativar a linguagem PL/pgSQL no seu novo banco de dados. Isso é atingido pelo comando abaixo comando. Para PostgreSQL 8.4+, normalmente ele já está instalado

**createlang plpgsql [SEU\_BANCO\_DE\_DADOS]**

Agora, desloque o objeto e as definições de função do PostGIS para dentro do seu banco de dados carregando o `postgis.sql` arquivo das definições (localizado em `[prefix]/share/contrib` como especificado durante o passo de configuração).

**psql -d [SEU\_BANCO\_DE\_DADOS] -f postgis.sql**

Para uma configuração completa do sistema coordenado de definição dos identificadores do EPSG, você também pode carregar o arquivo de definições `spatial_ref_sys.sql` e popularizar a table `spatial_ref_sys`. Isso irá permitir que você desempenhe operações `ST_Transform()` nas geometrias.

**psql -d [SEU\_BANCO\_DE\_DADOS] -f spatial\_ref\_sys.sql**

Se deseja adicionar comentários às funções do PostGIS, o último passo é carregar o `postgis_comments.sql` no seu banco de dados espacial. Os comentários podem ser visualizados digitando `\dd [function_name]` de uma **psql** terminal window.

**psql -d [SEU\_BANCO\_DE\_DADOS] -f postgis\_comments.sql**

Instalar suporte a raster

**psql -d [yourdatabase] -f rtpostgis.sql**

Instale os comentários do suporte raster. Isso irá fornecer uma ajuda rápida para cada função raster usando `psql` ou PgAdmin ou qualquer outra ferramenta PostgreSQL que pode mostrar função comentários.

**psql -d [SEU\_BANCO\_DE\_DADOS] -f raster\_comments.sql**

Instalar suporte a topologia

**psql -d [yourdatabase] -f topology/topology.sql**

Instale os comentários do suporte de topologia. Isso irá fornecer uma ajuda rápida para cada função/tipo de topologia usando `psql` ou PgAdmin ou qualquer outra ferramenta PostgreSQL que pode mostrar função comentários

**psql -d [yourdatabase] -f topology/topology\_comments.sql**

Se planeja restaurar um backup antigo das primeiras versões nesse novo db, faça funcionar:

**psql -d [yourdatabase] -f legacy.sql**



### Note

Existe uma alternativa `legacy_minimal.sql` que você pode utilizar que irá instalar barebones necessários para recuperar tables e trabalhar com aplicativos como MapServer e GeoServer. Se você tem vistas que utilizam distância/comprimento etc, você precisará do `legacy.sql` completo

Você pode usar `uninstall_legacy.sql` para livrar-se das funções menosprezadas depois de terminar de restaurar e limpar.

## 2.7 Instalando e usando o padronizador de endereço

A extensão `address_standardizer` era usada para ser um pacote separado que requeria download separado. Do PostGIS 2.2 em diante é compactado. Para mais informações sobre o `address_standardize`, o que ele faz e como fazer sua configuração, referir-se Chapter 12.

O padronizador pode ser usado em conjunção com a extensão `tiger_geocoder` PostGIS compactada como uma reposição para a `Normalize_Address` discutida. Para usar como reposição Section 2.8.3. Você também pode utilizar como um building block para seu próprio geocoder ou como padronizar seu endereço para uma comparação de endereços mais fácil.

O padronizador de endereço confia no PCRE que já está instalado na maioria dos sistemas Nix, mas você pode baixar a última versão em: <http://www.pcre.org>. Se durante Section 2.4.1, o PCRE é encontrado, então a extensão do padronizador de endereço será automaticamente construída. Se você tem um pcre personalizado que queira usar, passe a configurar `--with-pcre-dir=/path/to/pcre` onde `/path/to/pcre` é a pasta root para o seu pcre incluso e lista lib.

Para usuários do Windows, o pacote PostGIS 2.1+ já está compactado com o `address_standardizer`, então não precisa compilar podendo seguir direto para o passo `CREATE EXTENSION`.

Uma vez que instalou, você pode conectar no seu banco de dados e rodar o SQL:

```
CRIAR EXTENSÃO address_standardizer;
```

O teste seguinte não requer `tables rules`, `gaz` ou `lex`.

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

Saída deve ser

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

### 2.7.1 Instalando Regexp::Montar

Perl `Regexp:Assemble` não é mais necessário para a compilação extensão `address_standardizer` desde que os arquivos que ele gera são parte da fonte três. Entretanto se precisar editar o `usps-st-city-orig.txt` ou `usps-st-city-orig.txt` `usps-st-city-adds.tx`, você precisa reconstruir `parseaddress-stcities.h` que exige `Regexp:Assemble`.

```
cpan Regexp::Montar
```

ou se estiver no Ubuntu / Debian talvez você precise fazer

```
sudo perl -MCPAN -e "install Regexp::Assemble"
```

## 2.8 Instalando, Atualizando o Tiger Geocoder e carregando dados

Extras like `Tiger geocoder` may not be packaged in your PostGIS distribution. If you are missing the `tiger geocoder` extension or want a newer version than what your install comes with, then use the `share/extension/postgis_tiger_geocoder.*` files from the packages in [Windows Unreleased Versions](#) section for your version of PostgreSQL. Although these packages are for windows, the `postgis_tiger_geocoder` extension files will work on any OS since the extension is an SQL/plpgsql only extension.

## 2.8.1 Tiger Geocoder ativando seu banco de dados PostGIS: Usando Extensão

Se está usando PostgreSQL 9.1+ e PostGIS 2.1+, você pode obter vantagem do novo modelo de extensão para instalar o tiger geocoder. Para fazer:

1. Primeiramente obtenha binários para PostGIS 2.1+ ou compile e instale como de costume. Isso deverá instalar os arquivos de extensão necessários bem como para o geocoder.
2. Conecte ao seu banco de dados via psql ou pgAdmin ou qualquer outra ferramenta e execute os comandos SQL seguintes. Note que se você está instalando em um banco de dados que já possui o postgis, você não precisa fazer o primeiro passo. Se você já tem a extensão `fuzzystrmatch` instalada, não é preciso fazer o segundo passo também.

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

Se você já tem a extensão `postgis_tiger_geocoder` instalada e só quer atualizar para a última versão, execute:

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

Se você fez entradas personalizadas ou alterações nos `tiger.loader_platform` e `tiger.loader_variables`, talvez você precisará atualizar estes.

3. Para confirmar que sua instalação está funcionando corretamente, execute esse sql no seu banco de dados:

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

Qual deve sair

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

4. Criar um novo registro na table `tiger.loader_platform` com os paths dos seus executáveis e servidor.

Então, por exemplo, para criar um perfil chamado `debbie` que segue a convenção `sh`, você deveria fazer:

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

E então, edite os paths na coluna `declare_sect` para aqueles que servem ao pg, unzip, shp2pgsql, psql, etc da Debbie.

Se você não editou essa table `loader_platform`, ela só irá conter casos comuns de localizações de itens e você terá que editar a script gerada depois que ela for gerada.

5. As of PostGIS 2.4.1 the Zip code-5 digit tabulation area `zcta5` load step was revised to load current `zcta5` data and is part of the **Loader\_Generate\_Nation\_Script** when enabled. It is turned off by default because it takes quite a bit of time to load (20 to 60 minutes), takes up quite a bit of disk space, and is not used that often.

To enable it, do the following:

```
UPDATE tiger.loader_lookuptables SET load = true WHERE table_name = 'zcta510';
```

If present the [Geocode](#) function can use it if a boundary filter is added to limit to just zips in that boundary. The [Reverse\\_Geocode](#) function uses it if the returned address is missing a zip, which often happens with highway reverse geocoding.

6. Criar uma pasta chamada `gisdata` na raiz do servidor ou do seu computador local, se você tem uma rede de conexão rápida com o servidor. Essa pasta está onde os arquivos tiger serão baixados e processados. Se não estiver satisfeito em ter a pasta na raiz do servidor ou, simplesmente, quiser alterar para uma outra pasta para representação, edite o campo `staging_fold` na table `tiger.loader_variables`.
7. Criar uma pasta chamada `temp` na pasta `gisdata` ou onde designar a `staging_fold`. Esta será a pasta onde o carregador extrai os dados tiger baixados.
8. Then run the [Loader\\_Generate\\_Nation\\_Script](#) SQL function make sure to use the name of your custom profile and copy the script to a `.sh` or `.bat` file. So for example to build the nation load:

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

9. Run the generated nation load commandline scripts.

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

10. After you are done running the nation script, you should have three tables in your `tiger_data` schema and they should be filled with data. Confirm you do by doing the following queries from `psql` or `pgAdmin`

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

```
count
-----
  3233
(1 row)
```

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

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

11. By default the tables corresponding to `bg`, `tract`, `tabblock` are not loaded. These tables are not used by the geocoder but are used by folks for population statistics. If you wish to load them as part of your state loads, run the following statement to enable them.

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

Alternatively you can load just these tables after loading state data using the [Loader\\_Generate\\_Census\\_Script](#)

12. For each state you want to load data for, generate a state script [Loader\\_Generate\\_Script](#).



#### Warning

DO NOT Generate the state script until you have already loaded the nation data, because the state script utilizes county list loaded by nation script.

---

13.
 

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

#### 14. Executar as scripts commandlines geradas

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

#### 15. Depois que terminar de carregar todos os dados ou estiver parado em um ponto, é bom analisar todas as tiger tables para atualizar as estatísticas (incluindo as estatísticas herdadas)

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

### 2.8.1.1 Convertendo uma Instalação Tiger Geocoder Regular para Modelo de Extensão

Se você instalou o tiger geocoder sem utilizar a extensão modelo, você pode converter para a extensão modelo como segue:

1. Siga as instruções em Section 2.8.5 para a atualização sem extensão modelo.
2. Conecte ao seu banco de dados com psql ou pgAdmin e execute o seguinte comando:

```
CREATE EXTENSION postgis_tiger_geocoder FROM unpackaged;
```

### 2.8.2 Tiger Geocoder Ativando seu banco de dados PostGIS: Sem Utilizar Extensões

Primeiro instale PostGIS usando as instruções prévias.

Se você não tem uma pasta extras, faça o download <http://download.osgeo.org/postgis/source/postgis-2.5.0.tar.gz>

```
tar xvfz postgis-2.5.0.tar.gz
```

```
cd postgis-2.5.0/extras/tiger_geocoder
```

Edite o `tiger_loader_2015.sql` (ou o último arquivo carregador que encontrar, a menos que queira carregar um ano diferente) para os paths dos seus executáveis, servidor etc ou alternativamente você pode atualizar a table `loader_platform` uma vez instalada. Se não quiser editar esse arquivo ou a table `loader_platform`, só irá conter os casos comuns de localização de itens e você terá que editar a script gerada depois de executar as funções SQL `Loader_Generate_Nation_Script` e `Loader_Generate_Script`.

Se estiver instalando o Tiger geocoder pela primeira vez, edite a script `create_geocode.bat` se você está no Windows ou a `create_geocode.sh` se estiver no Linux/Unix/Mac OSX com suas configurações específicas do PostgreSQL e execute a script correspondente da commandline.

Certifique-se que agora você tem um esquema `tiger` no seu banco de dados e que ele é parte do seu bando de dados `search_path`. Se ele não for, adicione-o com um comando algo ao longo da linha de:

```
ALTER DATABASE geocoder SET search_path=public, tiger;
```

A funcionalidade do normalizador de endereço trabalha relativamente sem nenhum dado, exceto por endereços complicados. Executar este teste e verificar as coisas se parecem com isso:

```
SELECT pprint_addy(normalize_address('202 East Fremont Street, Las Vegas, Nevada 89101')) ←
  As pretty_address;
pretty_address
-----
202 E Fremont St, Las Vegas, NV 89101
```

### 2.8.3 Usando Padronizador de Endereço com Tiger Geocoder

Uma das maiores queixas das pessoas é a função normalizador de endereços `Normalize_Address` que normaliza um endereço para preparação antes da geocoding. O normalizador está longe da perfeição e tentar corrigir suas imperfeições demanda um grande número de recursos. Como tal nós integramos com outro projeto que tem um mecanismo de padronizador de endereços muito melhor. Para usar esse novo `address_standardizer`, você compila a extensão como está descrito em Section 2.7 e instala como uma extensão no seu banco de dados.

Uma vez que você instala essa extensão no mesmo banco de dados que instalou `postgis_tiger_geocoder`, então o `Page_Normalize_Address` pode ser usado ao invés do `Normalize_Address`. Essa extensão é avessa ao tiger, logo pode ser usada com outras fontes de dados como: endereços internacionais. A extensão tiger geocoder vem compactada com suas próprias versões personalizadas de `mesa de regras` (`tiger.pagc_rules`), `gaz table` (`tiger.pagc_gaz`), e `lex table` (`tiger.pagc_lex`). Essas você pode adicionar e atualizar para melhorar sua experiência com o padronizador de acordo com suas necessidades.

### 2.8.4 Carregando Dados Tiger

As instruções para carregar dados estão disponíveis de uma maneira mais detalhada em `extras/tiger_geocoder/tiger_2011/README`. Isso só inclui os passos gerais.

O carregador processa dados de downloads do site de censo para os respectivos arquivos de nação, solicitações de estados, extrai os arquivos e carrega cada estado para seu grupo separado de state tables. Cada state table herda das tables definidas no esquema tiger sendo suficiente apenas para pesquisar aquelas tables para acessar todos os dados e derrubar um conjunto de state tables a qualquer momento usando o `Drop_State_Tables_Generate_Script` se quiser recarregar um estado ou não precisa de um estado mais.

Para ser capaz de carregar dados, você vai precisar das seguintes ferramentas:

- Uma ferramenta para descompactar os arquivos compactados do site de censo.  
Para Unix como sistemas: executável `unzip` que é instalado, normalmente, na maioria dos Unix como plataformas.  
Para Windows, 7-zip é uma ferramenta comprimir/descomprimir grátis que você pode baixar no <http://www.7-zip.org/>
- `shp2pgsql` commandline que é instalada por padrão quando você instala o PostGIS.
- `wget` que é uma ferramenta grabber da internet, instalada na maioria dos sistemas Unix/Linux.  
Se você está no Windows, você pode obter binários pre compilados do <http://gnuwin32.sourceforge.net/packages/wget.htm>

Se estiver atualizando do tiger\_2010, você precisará gerar e executar `Drop_Nation_Tables_Generate_Script`. Antes de carregar qualquer dado de estado, você precisa carregar os dados da nação que será feito com `Loader_Generate_Nation_Script`. O qual irá gerar uma script de carregamento para você. `Loader_Generate_Nation_Script` é um passo que deve ser dado para atualizar (do 2010) e para novas instalações.

Para carregar dados do estado referir-se a `Loader_Generate_Script` para gerar uma script de dados de carregamento para sua plataforma para os estados que deseja. Note que você pode instalar estes gradativamente. Você não precisa carregar todos os estados de uma só vez. Pode carregá-los à medida que for precisando deles.

Depois que os estados desejados forem carregados, certifique-se de executar o:

```
SELECT install_missing_indexes();
```

como está descrito em `Install_Missing_Indexes`.

Para testar que está tudo funcionando normalmente, tente executar um geocode em um endereço no seu estado, usando `Geocode`

## 2.8.5 Atualizando sua Instalação Tiger Geocoder

Se você tem o Tiger Geocoder compactado com 2.0+ já instalado, você pode atualizar as funções de qualquer lugar, mesmo de uma tar ball provisória, se existem correções que você mal precisa. Isso só irá funcionar para as funções não instaladas do Tiger geocoder.

Se você não tem uma pasta extras, faça o download <http://download.osgeo.org/postgis/source/postgis-2.5.0.tar.gz>

```
tar xvfz postgis-2.5.0.tar.gz
```

```
cd postgis-2.5.0/extras/tiger_geocoder/tiger_2011
```

Localize a script `upgrade_geocoder.bat` se você está no Windows ou a script `upgrade_geocoder.sh` se você está no Linux/Unix/Mac OSX. Edite o arquivo para ter suas credenciais no banco de dados do postgis.

Se estiver atualizando de 2010 ou 2011, certifique-se de desmarcar a script de carregamento para ter a última script para carregar os dados de 2012.

Então, execute a script correspondente da commandline.

Em seguida, derrube todas as nation tables e carregue as novas. Gere uma drop script com essa declaração SQL, como está detalhado em [Drop\\_Nation\\_Tables\\_Generate\\_Script](#)

```
SELECT drop_nation_tables_generate_script();
```

Execute as declarações geradas drop SQL.

Gere uma script que carrega uma nação com SELECIONAR como está detalhado em [Loader\\_Generate\\_Nation\\_Script](#)

### Para windows

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

### Para unix/linux

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

Para instruções de como executar a script gerada use: Section [2.8.4](#). Isso só precisa ser feito uma vez.



#### Note

Você pode ter uma mistura de state tables de 2010/2011 e pode atualizar cada estado separadamente. Antes de atualizar um estado para 2011, você precisa, primeiramente, derrubar as tables de 2010 para aquele estado, usando: [Drop\\_State\\_Tables\\_Generate\\_Script](#).

## 2.9 Criar um banco de dados espacialmente ativado de um template

Algumas distribuições compactadas do PostGIS (em particular os instaladores Win32 para PostGIS >= 1.1.5), carregam as funções do PostGIS dentro de um template de banco de dados chamado `template_postgis`. Se o banco de dados `template_postgis` existe na sua instalação PostgreSQL, é possível para os usuários e/ou aplicações para criar bancos de dados espacialmente ativados utilizando somente um comando. Note que em ambos os casos, o usuário do banco de dados deve ter presumido o privilégio de criar novos bancos de dados.

Do shell:

```
# createdb -T template_postgis my_spatial_db
```

De SQL:

```
postgres=# CREATE DATABASE my_spatial_db TEMPLATE=template_postgis
```



## 2.10 Atualizando

Atualizar bancos de dados espaciais existentes pode ser complicado como ele requiere reposição ou introdução das novas definições de objeto do PostGIS.

Infelizmente, nem todas as definições podem ser facilmente repostas em um banco de dados vivo, então, algumas vezes a melhor opção é abandonar/recarregar o processo.

O PostGIS fornece um procedimento de ATUALIZAÇÃO SOFT para liberações menores e um procedimento de ATUALIZAÇÃO HARD para liberações maiores.

Antes de tentar atualizar o PostGIS, é sempre bom fazer o backup dos seus dados. Se você usa a `-Fc` flag para `pg_dump` você sempre vai poder restaurar a descarga de dados com uma ATUALIZAÇÃO HARD.

### 2.10.1 Atualização flexível

Se você instalou seu banco de dados utilizando extensões, você vai precisar atualizar utilizando o modelo extensão também. Se você instalou usando o a antiga script sql, então você deveria atualizar usando a mesma. Por favor, referir-se ao apropriado.

#### 2.10.1.1 Atualização Soft Pre 9.1+ ou sem extensões

Essa seção aplica-se somente àqueles que instalaram o PostGIS sem usar extensões. Se você tiver extensões e tentar atualizar com essa abordagem, você irá receber mensagens como:

```
não pode excluir ... porque a extensão do postgis depende disso.
```

Depois de compilar e instalar (fazer instalação), você deve encontrar um `postgis_upgrade.sql` e `rtpostgis_upgrade.sql` nas pastas de instalação. Por exemplo, `/usr/share/postgresql/9.3/contrib/postgis_upgrade.sql`. Instale o `postgis_upgrade.sql`. Se você tem a funcionalidade raster instalada, também precisará instalar o `/usr/share/postgresql/9.3/contrib/postgis_upgrade.sql`. Se está mudando do PostGIS 1.\* para o PostGIS 2.\* ou do anterior PostGIS 2.\* para r7409, você deve fazer uma ATUALIZAÇÃO HARD.

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

O mesmo procedimento aplica-se para as extensões de raster e topologia, com arquivos de atualização chamados `rtpostgis_upgrade*.sql` e `topology_upgrade*.sql`, respectivamente. Se você precisar deles:

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

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



#### Note

Se não conseguir encontrar o `postgis_upgrade*.sql` específico para atualizar sua versão, você está usando uma versão muito cedo para uma atualização soft e precisa de uma ATUALIZAÇÃO HARD.

A função `PostGIS_Full_Version` deve informar sobre a necessidade de executar esse tipo de atualização usando a mensagem "procs precisa de atualização".

#### 2.10.1.2 Atualização flexível 9.1+ usando extensões

Se você, originalmente, instalou o PostGIS com extensão, você precisará atualizar usando extensões também. Fazer uma atualização menor com extensões é razoavelmente indolor.

```
ALTER EXTENSION postgis UPDATE TO "2.5.0";
ALTER EXTENSION postgis_topology UPDATE TO "2.5.0";
```

Se você obtiver um erro note algo como:

```
Nenhum path de migração definido por ... para 2.5.0
```

Então você precisará fazer o backup do seu banco de dados, crie um novo como está descrito em [Section 2.5](#) e restaure seu backup no topo desse novo banco de dados.

Se você receber um aviso como:

```
Versão "2.5.0" da extensão "postgis" já está instalada
```

Então tudo já está atualizado e você pode ignorar seguramente. **A MENOS** que você esteja tentando atualizar de uma versão SVN para a próxima (a qual não recebe um número de versão novo); nesse caso você pode incluir "próximo" à string da versão e da próxima vez vai precisar derrubar o "próximo" sufixo novamente:

```
ALTER EXTENSION postgis UPDATE TO "2.5.0next";
ALTER EXTENSION postgis_topology UPDATE TO "2.5.0next";
```



#### Note

Se você instalou o PostGIS sem uma versão específica, você pode pular a reinstalação da extensão do postgis antes de restaurar, já que o backup só tem `CREATE EXTENSION postgis` e assim, escolha a versão mais nova durante a restauração.

## 2.10.2 Atualização rígida

Através da ATUALIZAÇÃO HARD nós queremos dizer descarga de dados/re-carregamento completo dos bancos de dados ativados do postgis. Você precisa dessa atualização quando a memória interna do objeto do PostGIS mudar ou quando a ATUALIZAÇÃO SOFT não for possível. O apêndice [Release Notes](#) relata que para cada versão você precisa de uma descarga de dados/re-carregamento (ATUALIZAÇÃO HARD) para atualizar.

O processo de descarga de dados/re-carregamento é auxiliado pela script `postgis_restore.pl` que cuida de pular todas as definições de descarga de dados que pertencem ao POstGIS (incluindo as antigas), permitindo que você restaure seus dados e esquemas em um banco de dados com o PostGIS instalado sem duplicar os erros de símbolo ou trazer objetos menosprezados adiante.

Instruções suplementares para usuários do Windows estão disponíveis em [Windows Hard upgrade](#).

O procedimento é:

1. Criar uma descarga de dados com "formato-personalizado" do banco de dados que você quer atualizar (chamaremos de `olddb`), inclua blobs binários (-b) e saídas verbosas (-v). O usuário pode ser o dono do banco de dados, não precisa ser `postgres` super conta.

```
pg_dump -h localhost -p 5432 -U postgres -Fc -b -v -f "/diretório/banco_antigo.backup" ↔
banco_antigo
```

2. Faça uma instalação nova do PostGIS em um banco de dados novo -- chamaremos esse banco de dados de `newdb`. Por favor use [Section 2.6](#) e [Section 2.5](#) para instruções de como fazer.

As entradas `spatial_ref_sys` encontradas na sua descarga de dados serão restauradas, mas elas não substituirão as que existem no `spatial_ref_sys`. Isso é para garantir que as correções no conjunto oficial serão precisamente propagadas para banco de dados restaurados. Se, por alguma razão, você quiser suas próprias substituições de entradas padrão, simplesmente não carregue o arquivo `spatial_ref_sys.sql` quando estiver criando o novo banco de dados.

Se seu banco de dados é muito antigo ou você sabe que esteve usando funções menosprezadas nas suas views e funções, provavelmente precise carregar `legacy.sql` para todas as suas funções e views etc. para retornar propriamente. Só faça isso se for necessário. Considere atualizar suas views e funções depois, se possível. As funções menosprezadas podem ser removidas posteriormente carregando `uninstall_legacy.sql`.

3. Restaure seu backup no seu novo banco de dados `newdb` usando `postgis_restore.pl`. Erros inesperados, se existirem, serão impressos na stream erro padrão através do `psql`. Mantenha-se conectado com eles.

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

Erros retornarão nos seguintes casos:

1. Algumas das suas views ou funções utilizam objetos menosprezados do PostGIS. Com o propósito de consertar isso, você talvez tente carregar a script anterior `legacy.sql` para restaurar ou você tenha que restaurar para uma versão do PostGIS que contenha esses objetos e tenha que tentar uma migração novamente depois de fazer a portabilidade do seu código. Se o `legacy.sql` funciona para você, não se esqueça de consertar seu código para parar de usar funções menosprezadas e derrube elas carregando: `uninstall_legacy.sql`.

2. Alguns relatos personalizados da `spatial_ref_sys` no arquivo de descarga de dados, têm o valor SRID inválido. os valores válidos são maiores que 0 e menores que 999000. Valores na extensão 999000.999999 são reservados para uso interno, enquanto valores > 999999 não podem ser usados de forma alguma. Todos os seus relatos personalizados com SRIDs inválidas serão retidos, com esses > 999999 movidos para a extensão reservada, mas a table `spatial_ref_sys` perderia a restrição verificada de defesa para aquela invariante para segurar e possivelmente sua chave primária (quando múltiplos SRIDS inválidos se convertem para o mesmo valor SRID reservado).

A fim de consertar isso, você deveria copiar sua SRS personalizada para uma SRID com valor válido (talvez na extensão 910000..910999), converter todas as suas tables para a srid nova (veja [UpdateGeometrySRID](#)), deletar a entrada inválida da `spatial_ref_sys` e reconstruir os check(s):

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

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

## 2.11 Problemas comuns durante a instalação

Existem várias coisas para averiguar quando a instalação ou atualização não saem como o esperado.

1. Certifique-se que instalou o PostgreSQL 9.4 ou mais novo e que você está compilando contra a mesma versão da fonte PostgreSQL assim como a versão do PostgreSQL que está sendo executada. Confusões podem acontecer quando sua distribuição (Linux) já instalou o PostgreSQL, ou você instalou o PostgreSQL antes e se esqueceu disso. PostGIS só irá funcionar com o PostgreSQL 9.4 ou mais novo, e mensagens estranhas e inesperadas de erro aparecerão se você usar uma versão mais antiga. Para verificar a versão PostgreSQL que está sendo executada, conecte ao banco de dados usando `psql` e faça essa consulta:

```
SELECT version();
```

Se você está usando uma distribuição baseada em RPM, você pode confirmar a existência de pacotes pre instalados utilizando o comando `rpm` como segue: `rpm -qa | grep postgresql`

2. Se sua atualização falhar, certifique-se que você está restaurando em um banco de dados que já possui o PostGIS instalado.

```
SELECT postgis_full_version();
```

Também certifique que a configuração detectou a localização e versão corretas do PostgreSQL, da biblioteca do Proj4 e da biblioteca do GEOS.

1. A saída da configuração foi usada para gerar o arquivo `postgis_config.h`. Verifique que as variáveis `POSTGIS_PGSQL_VERSION`, `POSTGIS_PROJ_VERSION` e `POSTGIS_GEOS_VERSION` foram configuradas corretamente.

## 2.12 Carregador/Dumper

Os dados do carregador e dumper são construídos e instalados automaticamente como parte do PostGIS. Para construir e instalar eles manualmente:

```
# cd postgis-2.5.0/loader
# make
# make install
```

O carregador é chamado `shp2pgsql` e converte arquivos ESRI Shape em SQL para carregar em PostGIS/PostgreSQL. O dumper é chamado `pgsql2shp` e converte as tables (ou pesquisas) PostGIS em arquivos ESRI Shape. Para mais documentação prolixo, veja a ajuda online e as páginas do manual.

## Chapter 3

# Perguntas frequentes PostGIS

1. *Onde posso encontrar tutoriais, guias e oficinas sobre como trabalhar com o PostGIS*

OpenGeo tem uma oficina passo a passo [Introdução ao PostGIS](#). Inclui dados para testes e uma introdução sobre como trabalhar com a OpenGeo Suite. Possivelmente o melhor tutorial sobre PostGIS. BostonGIS também tem [Um guia para quase idiotas para começar com PostGIS](#). Este é mais focado em usuários Windows.

2. *Minhas aplicações e ferramentas desktop funcionavam com PostGIS 1.5, mas não funcionam com o PostGIS 2.0. Como corrigir isto?*

Muitas funcionalidades obsoletas foram removidas da base de código do PostGIS na versão 2.0. Isto afetou aplicações e em especial, ferramentas de terceiros, como Geoserver, MapServer, QuantumGIS e OpenJump, para citar alguns casos. Existem algumas maneiras de resolver isto. Para aplicações de terceiros, você pode tentar atualizar para as versões mais atuais, muitas das quais corrigiram estes problemas. Para seu próprio código, você pode alterá-lo para não utilizar as funções removidas. A maior parte destas funções não possuem o prefixo ST\_ de ST\_Union, ST\_Length, etc. Como um último recurso, você pode instalar todo o script legado `legacy.sql` ou apenas as porções de `legacy.sql` que você precisa. O arquivo `legacy.sql` está localizado na mesma pasta em que o arquivo `postgis.sql`. Você pode instalar este arquivo após a instalação do `postgis.sql` e `spatial_ref_sys.sql`, para ter de volta todas 200 funções que foram removidas.

3. *Quando eu carregado dados do OpenStreetMap com o osm2pgsql, estou recebendo o erro: ERROR: operator class "gist\_geometry\_ops" does not exist for access method "gist" Error occurred. Isto funcionava perfeitamente no PostGIS 1.5.*

No PostGIS 2, o operador de geometria padrão `gist_geometry_ops` foi alterado para `gist_geometry_ops_2d` e o `gist_geometry_ops` foi completamente removido. Isto foi feito pois o PostGIS 2 também introduziu índices espaciais n-dimensionais para suporte a 3D e o nome antigo não era uma boa escolha. Algumas aplicações mais antigas, criavam tabelas e índices, explicitamente escolhendo o nome do operador. Isto era desnecessário se você quisesse o índice padrão 2D. Altere a criação do índice de:ERRADO:

```
CREATE INDEX idx_my_table_geom ON my_table USING gist(geom gist_geometry_ops);
```

**CORRETO:**

```
CREATE INDEX idx_my_table_geom ON my_table USING gist(geom);
```

O único caso onde você precisará especificar a classe do operador é se você quer um índice espacial 3D:

```
CREATE INDEX idx_my_super3d_geom ON my_super3d USING gist(geom gist_geometry_ops_nd);
```

Se você infelizmente está preso com código compilado que você não pode alterar e usa o operador `gist_geometry_ops`, você pode criar a classe antiga usando `legacy_gist.sql` empacotado na instalação do PostGIS 2.0.2 ou maior. Contudo, se você utilizar esta "correção", é recomendado que você, após a criação do índice, delete-o, e recrie-o sem a classe do operador. Isto vai te salvar no futuro quando precisar realizar o upgrade.

4. *Estou rodando PostgreSQL 9.0 e não mais posso ler/visualizar geometrias no OpenJump, Safe FME e outras ferramentas?*

No PostgreSQL 9.0+, o encoding padrão para dados bytea foi alterado para hex e outros drivers JDBC ainda assumem o formato escapado. Isto afetou aplicações Java utilizando drivers antigos JDBC ou aplicações .NET que usam drivers antigos npgsql e esperam o comportamento antigo de ST\_AsBinary. Existem duas formas de resolver este problema. Você pode atualizar seu driver JDBC para a última versão do PostgreSQL 9.0, que você obter em <http://jdbc.postgresql.org/download.html>. Se você está utilizando uma aplicação .NET, você pode usar o driver Npgsql 2.0.11 ou melhor, que você fazer o download em [http://pgfoundry.org/frs/?group\\_id=1000140](http://pgfoundry.org/frs/?group_id=1000140) e descrito em [Francisco Figueiredo's Npgsql 2.0.11 released blog entry](#). Se realizar um upgrade dos drivers PostgreSQL não é uma opção, então você pode setar o valor padrão para o comportamento antigo, usando a seguinte mudança:

```
ALTER DATABASE mypostgisdb SET bytea_output='escape';
```

5. *Tentei usar o PgAdmin para visualizar minhas colunas geométricas e ela está em branco?*

PgAdmin não mostra nada para grandes colunas. A melhor maneira de garantir que você tem dados em suas colunas geométricas são

```
-- isso deve retornar a zero registros se todos seus campos geométricos estão ←
preenchidos
SELECT somefield FROM mytable WHERE geom IS NULL;
```

```
-- Para ter uma ideia do tamanho de sua geometria, faça a query que vai lhe dizer o ←
numero de pontos em suas colunas
SELECT MAX(ST_NPoints(geom)) FROM tabela;
```

6. *Quais tipos de objetos geométricos posso armazenar?*

Você pode armazenar pontos, linhas, polígonos, multipontos, multilinhas, multipolígonos e coleções geométricas. No PostGIS 2.0 e além você também pode armazenar TINS e superfícies poliédricas no tipo básico de geometria. Esses dados estão especificados no Open GIS Well Known Text Format (com extensões Z,M e ZM). Há três tipos de dados que são suportados atualmente. A geometria OGC padrão que utiliza um sistema de coordenadas planas para medidas, o tipo de dados de geografia que utiliza o sistema geodésico de coordenadas, com cálculos na esfera ou esferoide. O mais novo membro da família PostGIS tipo espacial é o raster para armazenar e analisar os dados do raster. existe um FAQ próprio para o raster. Disponível em [Chapter 10](#) e [Chapter 9](#) para mais detalhes.

7. *Estou confuso. Qual tipo de dados devo utilizar, geometria ou geografia?*

Resposta curta: geografia é um novo tipo de dados que suporta medições de distância de longo alcance, mas a maioria dos cálculos sobre ele estão atualmente mais lento do que são na geometria. Se você usar geografia -- você não precisará aprender muito sobre sistemas de coordenadas planar. Geografia é geralmente melhor se tudo o que importa é a medição de distâncias e comprimentos e você tem dados de todo o mundo. Tipo de dado geometria é um tipo mais antigo de dados que possui mais funções que o suportam, goza de maior apoio de ferramentas de terceiros, e as operações nele contidas são geralmente mais rápidas -- às vezes até 10 vezes mais rápido para geometrias maiores. Geometria é melhor se você se sentir confortável com sistemas de referência espacial ou se estiver lidando com dados localizados onde todos seus dados se encaixam em um único **Sistema Espacial de Referência (SRID)**, ou você precisará fazer muito processamento espacial. Nota: É bastante fácil de fazer conversões pontuais entre dois tipos para ganhar os benefícios de cada. Consulte [Section 14.11](#) para ver o que é e não é suportado. Resposta longa: Se refere a nossa mais longa discursão no [Section 4.2.2](#) e [tipo de função matriz](#).

8. *Tenho questões mais aprofundadas sobre geography, tais como quão grande a geografia de uma região pode ser adicionada em uma coluna geography e continua razoavelmente responsiva. Existem limitações tais como pólos, tudo no campo deve caber em um hemisfério (como SQL Server 2008 tem), velocidade etc?*

Suas perguntas são profundas e complexas para serem respondidas de forma adequada nesta seção. Por favor, refira-se ao documento [Section 4.2.3](#).

9. *Como insiro um objeto GIS dentro do banco de dados?*

Primeiramente, você precisa criar uma tabela com uma coluna do tipo "geometry" ou "geography" para armazenar seus dados GIS. O armazenamento do dado do tipo "geography" é um pouco diferente do tipo "geometry". Consulte o documento [Section 4.2.1](#) para detalhes. Para geometrias: conecte seu banco de dados com `psql` e tente o seguinte SQL:

```
CREATE TABLE gtest (id serial primary key, name varchar(20), geom geometry(LINESTRING)) ←
;
```

Se a definição da coluna geométrica falhar, você provavelmente não carregou as funções do PostGIS em seu banco de dados ou está usando uma versão pré-2.0. Veja: [Section 2.4](#). Após a criação da tabela, você pode inserir uma geometria através de um comando SQL INSERT. O objeto GIS em si é formatado utilizando o formato OGC WKT ("well known text"):

```
INSERT INTO gtest (ID, NAME, GEOM)
VALUES (
  1,
  'First Geometry',
  ST_GeomFromText('LINESTRING(2 3,4 5,6 5,7 8)')
);
```

Para maiores informações sobre outros objetos GIS, veja a [referência de objetos](#). Para visualizar seus dados GIS na tabela:

```
SELECT id, name, ST_AsText(geom) AS geom FROM gtest;
```

O valor de retorno deve se parecer com algum assim:

```
id | name           | geom
----+-----+-----
  1 | First Geometry | LINESTRING(2 3,4 5,6 5,7 8)
(1 row)
```

#### 10. Como construo uma pesquisa geoespacial?

Da mesma forma como você constrói qualquer pesquisa no banco de dados, com SQL em uma combinação de valores de retorno, funções e testes de álgebra booleana. Para pesquisas geoespaciais, existem duas questões que são importantes de se ter em mente durante sua construção: existe um índice geoespacial que você pode utilizar; e, você está realizando cálculos computacionalmente caros em um número grande de geometrias. Em geral, você vai querer utilizar os operadores de interseção (&&) que testa se os retângulos envolventes de feições se interseccionam. A razão que torna o operador && útil é que existe um índice geoespacial para acelerar a resolução do teste, que será utilizado pelo operador. Isto pode tornar as pesquisas muito mais rápidas. Você ainda pode usar funções geoespaciais, como ST\_Distance(), ST\_Intersects(), ST\_Contains() e ST\_Within(), entre outras, para filtrar ainda mais os resultados de sua pesquisa. A maior parte das pesquisas geoespaciais incluem ambos testes: um teste indexado e um teste de função geoespacial. O teste indexado serve para limitar o número de tuplas para as que *podem* ter a condição de interesse. As funções geoespaciais servem para testar a condição exatamente como esperado.

```
SELECT id, the_geom
FROM thetable
WHERE
  ST_Contains(the_geom, 'POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))');
```

#### 11. Como acelero pesquisas geoespaciais em grandes tabelas?

Pesquisas rápidas em tabelas grandes é a *raison d'être* de banco de dados geoespaciais (bem como suporte a transações), então ter um bom índice é importante. Para criar um índice espacial em uma tabela com uma coluna `geometry`, utilize a função "CREATE INDEX" que segue abaixo:

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

A opção "USING GIST" diz ao servidor que utilize um índice GIST (Generalized Search Tree).



#### Note

Índices GIST são assumidamente lossy. Índices lossy usam um objeto de busca (em casos espaciais, uma bounding box) para construir o índice.

Você também deveria garantir que o plano de acesso do PostgreSQL tem informações suficientes sobre seu índice para fazer decisões racionais sobre quando utilizá-lo. Para fazer isso, você deve atualizar as estatísticas nas suas tabelas geométricas. Para o PostgreSQL 8.0.x ou posterior, apenas execute o comando `VACUUM ANALYZE`. Para o PostgreSQL 7.4.x ou anterior, execute o comando `SELECT UPDATE_GEOMETRY_STATS()`.

12. *Por que o índice do PostgreSQL R-Tree não é suportado?*

Versões mais recentes do PostGIS no PostgreSQL usam índice R-Tree. Por outro lado, PostgreSQL R-Trees foi totalmente descontinuado desde a versão 0.6, e índice espacial é provido com um esquema R-Tree-over-GIST. Nossos testes tem mostrado que a velocidade de busca para o índice nativo R-Tree e o GIST são comparáveis. O nativo R-Tree tem duas limitações que dele indesejável para utilizar com funcionalidades GIS (note que essa limitação é devido a atual implementação do PostgreSQL, não do conceito R-Tree em geral):

- Índices R-Tree no PostgreSQL não podem manipular funcionalidades que são maiores que 8K em tamanho. Já os índices GIST podem, utilizando o truque "lossy" de substituição da bounding pela própria funcionalidade.
- Índices R-Tree no PostgreSQL não são "null safe", então construir um índice em uma coluna geometry que contenha null, falhará.

13. *Porque eu devo utilizar a função `AddGeometryColumn()` e todas as outras coisas do OpenGIS?*

Se você não quer utilizar o suporte à funções OpenGIS, você não precisa. Simplesmente crie tabelas como nas versões antigas, definindo as colunas geometry no comando CREATE. Todas suas geometrias terão SRIDs de -1, e as tabelas de metadados do OpenGIS *não* serão preenchidas apropriadamente. No entanto, causará falha na maioria das aplicações baseadas no PostGIS, e é geralmente aconselhado que utilize `AddGeometryColumn()` para criar tabelas geometry. MapServer é uma aplicação que faz uso de `geometry_columns` metadado. Especificamente, MapServer pode user SRID da coluna geometry para fazer reprojeção on-the-fly das funcionalidades dentro de uma correta projeção no mapa.

14. *Qual é a melhor maneira de achar todos os objetos dentre um radius e outro objeto?*

Para usar a base de dados mais eficientemente, é melhor fazer radius queries que combine com teste radius com teste bounding box: o teste bounding box usa índice espacial, dando rapidez no acesso para subconjunto de dados que o teste radius é aplicada. A função `ST_DWithin(geometry, geometry, distance)` é uma forma conveniente de realizar uma busca distante no índice. Ele trabalha criando retângulo de busca suficiente para encobrir todo o raio, depois realiza uma busca exata da distância no subconjunto de resultados do índice. Por exemplo, para encontrar todos os objetos com 100 metros de `POINT(1000 1000)` a query a seguir trabalharia corretamente:

```
SELECT * FROM geotable
WHERE ST_DWithin(geocolumn, 'POINT(1000 1000)', 100.0);
```

15. *Como posso fazer uma reprojeção de coordenadas como parte de uma query?*

Para realizar uma reprojeção, ambos as coordenadas fonte e destino devem estar definidas na tabela `SPATIAL_REF_SYS`, e a geometria reprojeta deve já ter um SRID setado para ela. Uma vez isso feito, a reprojeção é tão simples quanto referenciar um SRID de destino desejado. The below projects a geometry to NAD 83 long lat. Abaixo apenas trabalhará se o srid do `the_geom` não for -1 (não indefinido spatial ref)

```
SELECT ST_Transform(the_geom, 4269) FROM geotable;
```

16. *Faço uma `ST_AsEWKT` e `ST_AsText` na minha maior geometria e isso me retorna em branco. O que acontece?*

Vocês está provavelmente utilizando PgAdmin ou outra ferramenta que não retorna grandes textos. Se sua geometria é muito grande, aparecerá vazio nessas ferramentas. Use PSQL se você realmente precisa ver isso ou retornar em WKT.

Para verificar o de geometrias que realmente estão vazias:

```
SELECT count(gid) FROM geotable WHERE the_geom IS NULL;
```

17. *Quando eu faço um `ST_Intersects`, tenho o retorno que minhas duas geometrias não intersectam quando EU SEI QUE SIM. O que acontece?*

Isso geralmente acontece em dois casos comuns. Sua geometria é inválida -- verifique `ST_IsValid` ou or você está assumindo que elas intesectam porque `ST_AsText` trucou os números e você tem muitos decimais que não estão sendo exibidos a você.



18. *Estou liberando software que usa PostGIS, o que significa que meu software foi licenciado utilizado a GPL como PostGIS? Terei que publicar todo o meu código se utilizar o PostGIS?*

Almost certainly not. As an example, consider Oracle database running on Linux. Linux is GPL, Oracle is not: does Oracle running on Linux have to be distributed using the GPL? No. Similarly your software can use a PostgreSQL/PostGIS database as much as it wants and be under any license you like. The only exception would be if you made changes to the PostGIS source code, and *distributed your changed version* of PostGIS. In that case you would have to share the code of your changed PostGIS (but not the code of applications running on top of it). Even in this limited case, you would still only have to distribute source code to people you distributed binaries to. The GPL does not require that you *publish* your source code, only that you share it with people you give binaries to. The above remains true even if you use PostGIS in conjunction with the optional CGAL-enabled functions. Portions of CGAL are GPL, but so is all of PostGIS already: using CGAL does not make PostGIS any more GPL than it was to start with.

## Chapter 4

# Usando o PostGIS: Gerenciamento de dados e consultas

### 4.1 Objetos GIS

The GIS objects supported by PostGIS are a superset of the "Simple Features" defined by the OpenGIS Consortium (OGC). PostGIS supports all the objects and functions specified in the OGC "Simple Features for SQL" specification.

PostGIS extends the standard with support for 3DZ, 3DM and 4D coordinates.

#### 4.1.1 OpenGIS WKB e WKT

A especificação OpenGIS define dois caminhos padrão de expressar objetos espaciais: o Well-Known Text (WKT) e o Well-Known Binary (WKB). Ambos incluem informação sobre o tipo do objeto e as coordenadas que os formam.

A seguir, exemplos das representações de textos (WKT) dos objetos espaciais das características:

- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- 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))
- 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))

A especificação OpenGIS também requer que o formato do armazenamento interno dos objetos espaciais incluam um identificador de sistema de referência espacial (SRID). O SRID é fundamental na criação de objetos espaciais para a inserção no banco de dados.

Entrada/Saída destes formatos estão disponíveis usando as seguintes interfaces:

```
bytea WKB = ST_AsBinary(geometry);
text WKT = ST_AsText(geometry);
geometry = ST_GeomFromWKB(bytea WKB, SRID);
geometry = ST_GeometryFromText(text WKT, SRID);
```

Por exemplo, uma declaração inserida válida para criar e inserir um objeto espacial OGC seria:

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



### 4.1.3 SQL-MM Part 3

E especificação SQL Multimedia Applications Spatial estende os aspectos simples para SQL spec definindo um número de curvas circularmente interpoladas.

The SQL-MM definitions include 3DM, 3DZ and 4D coordinates, but do not allow the embedding of SRID information.

The Well-Known Text extensions are not yet fully supported. Examples of some simple curved geometries are shown below:

- CIRCULARSTRING(0 0, 1 1, 1 0)

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

The CIRCULARSTRING is the basic curve type, similar to a LINESTRING in the linear world. A single segment required three points, the start and end points (first and third) and any other point on the arc. The exception to this is for a closed circle, where the start and end points are the same. In this case the second point **MUST** be the center of the arc, ie the opposite side of the circle. To chain arcs together, the last point of the previous arc becomes the first point of the next arc, just like in LINESTRING. This means that a valid circular string must have an odd number of points greater than 1.

- COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0 1))

Uma curva composta é uma curva única e contínua que tem segmentos curvados (circulares) e lineares. Isto significa que, além de ter componentes bem formados, o ponto final de cada componente (exceto o último) deve ser coincidente com o ponto inicial do componente seguinte.

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

Exemplo de curva composta em um polígono curvo: CURVEPOLYGON(COMPOUNDCURVE(CIRCULARSTRING(0 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) )

Um POLÍGONOCURVO é como um polígono, com um anel externo e zero ou mais anéis internos. A diferença é que um anel pode obter a forma de uma string circular, linear ou composta.

Assim como o PostGIS 1.4, o PostGIS suporta curvas compostas em um polígono curvo.

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

A MULTICURVA é uma coleção de curvas, que podem incluir strings lineares, circulares e compostas.

- 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)))

Esta é uma coleção de superfícies, que podem ser polígonos (lineares) ou polígonos curvos.



#### Note

Todos as comparações de pontos flutuantes dentro da implementação SQL-MM são representadas com uma tolerância específica, atualmente 1E-8.

## 4.2 Tipo de geografia PPostGIS

O tipo de geografia fornece suporte natural para características representadas nas coordenadas "geográficas" (às vezes chamadas de coordenadas "geodéticas", ou "lat/lon", ou 'lon/lat'). As coordenadas geográficas são coordenadas esféricas expressadas em unidades angulares (graus).

A base para a geometria PostGIS é um plano. O menor caminho entre dois pontos no plano é uma linha. Isso quer dizer que cálculos em geometrias (áreas, distâncias, comprimentos, interseções etc) podem ser feitos usando matemática cartesiana e vetores de linhas.

The basis for the PostGIS geographic type is a sphere. The shortest path between two points on the sphere is a great circle arc. That means that calculations on geographies (areas, distances, lengths, intersections, etc) must be calculated on the sphere, using more complicated mathematics. For more accurate measurements, the calculations must take the actual spheroidal shape of the world into account.

Devido à matemática fundamental ser muito mais complicada, existem poucas funções definidas pela geografia em vez da geometria. Ao longo do tempo, à medida que os algoritmos forem adicionados, as capacidades da geografia serão expandidas.

It uses a data type called `geography`. None of the GEOS functions support the `geography` type. As a workaround one can convert back and forth between geometry and geography types.

Prior to PostGIS 2.2, the geography type only supported WGS 84 long lat (SRID:4326). For PostGIS 2.2 and above, any long/lat based spatial reference system defined in the `spatial_ref_sys` table can be used. You can even add your own custom spheroidal spatial reference system as described in [geography type is not limited to earth](#).

Regardless which spatial reference system you use, the units returned by the measurement (`ST_Distance`, `ST_Length`, `ST_Perimeter`, `ST_Area`) and for input of `ST_DWithin` are in meters.

The geography type uses the PostgreSQL typmod definition format so that a table with a geography field can be added in a single step. All the standard OGC formats except for curves are supported.

### 4.2.1 Geografia Básica

The geography type does not support curves, TINS, or POLYHEDRALSURFACES, but other geometry types are supported. Standard geometry type data will autocast to geography if it is of SRID 4326. You can also use the EWKT and EWKB conventions to insert data.

- POINT: Creating a table with 2D point geography when srid is not specified defaults to 4326 WGS 84 long lat:

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

POINT: Creating a table with 2D point geography in NAD83 longlat:

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

Creating a table with z coordinate point and explicitly specifying srid

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

- LINESTRING

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

- POLYGON

```
--polygon NAD 1927 long lat
CREATE TABLE lgeognad27(gid serial PRIMARY KEY, geog geography(POLYGON,4267) );
```

- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

The geography fields get registered in the `geography_columns` system view.

Agora, veja a view "geography\_columns" e note que sua tabela está listada.

You can create a new table with a GEOGRAPHY column using the CREATE TABLE syntax.

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

Note that the location column has type GEOGRAPHY and that geography type supports two optional modifiers: a type modifier that restricts the kind of shapes and dimensions allowed in the column; an SRID modifier that restricts the coordinate reference identifier to a particular number.

Os valores permitidos para o modificador de tipo são: PONTO, LINESTRING, POLÍGONO, MULTIPONTO, MULTILINESTRING, MULTIPOLÍGONO. O modificador também suporta restrições de dimensionalidade através de sufixos: Z, M, e ZM. Então, por exemplo, um modificador de 'LINESTRINGM' só permitiria line strings com três dimensões, e trataria a terceira dimensão como uma medida. Da mesma forma, 'PONTOZM' esperaria dados de quatro dimensões.

If you do not specify an SRID, the SRID will default to 4326 WGS 84 long/lat will be used, and all calculations will proceed using WGS84.

Uma vez que tenha criado sua tabela, você pode vê-la na tabela GEOGRAPHY\_COLUMNS:

```
-- See the contents of the metadata view
SELECT * FROM geography_columns;
```

Você pode inserir dados na tabela da mesma forma que iria se estivesse usando uma coluna GEOMETRIA:

```
-- Add some data into the test table
INSERT INTO global_points (name, location) VALUES ('Town', 'SRID=4326;POINT(-110 30)');
INSERT INTO global_points (name, location) VALUES ('Forest', 'SRID=4326;POINT(-109 29)');
INSERT INTO global_points (name, location) VALUES ('London', 'SRID=4326;POINT(0 49)');
```

Criar um índice funciona da mesma forma que uma GEOMETRIA. O PostGIS irá notar que o tipo de coluna é GEOGRAFIA e criará um índice baseado em esfera apropriado em vez do de costume usado para GEOMETRIA.

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

As funções de consulta e medida usam unidades em metros. Então, os parâmetros de distância deveriam ser esperados em metros (ou metros quadrados para áreas).

```
-- Show a distance query and note, London is outside the 1000km tolerance
SELECT name FROM global_points WHERE ST_DWithin(location, 'SRID=4326;POINT(-110 29)::geography, 1000000);
```

You can see the power of GEOGRAPHY in action by calculating how close a plane flying from Seattle to London (LINESTRING(-122.33 47.606, 0.0 51.5)) comes to Reykjavik (POINT(-21.96 64.15)).

```
-- Distance calculation using GEOGRAPHY (122.2km)
SELECT ST_Distance('LINESTRING(-122.33 47.606, 0.0 51.5)::geography, 'POINT(-21.96 64.15)::geography);
```

```
-- Distance calculation using GEOMETRY (13.3 "degrees")
SELECT ST_Distance('LINESTRING(-122.33 47.606, 0.0 51.5)::geometry, 'POINT(-21.96 64.15)::geometry);
```

Testing different lon/lat projects. Any long lat spatial reference system listed in spatial\_ref\_sys table is allowed.

```
-- NAD 83 lon/lat
SELECT 'SRID=4269;POINT(-123 34)::geography;
        geography
-----
0101000020AD10000000000000000000C05EC000000000000004140
(1 row)
```

```
-- NAD27 lon/lat
SELECT 'SRID=4267;POINT(-123 34)::geography;
        geography
-----
0101000020AB10000000000000000000C05EC000000000000004140
(1 row)
```

```
-- NAD83 UTM zone meters, yields error since its a meter based projection
SELECT 'SRID=26910;POINT(-123 34) '::geography;

ERROR:  Only lon/lat coordinate systems are supported in geography.
LINE 1: SELECT 'SRID=26910;POINT(-123 34) '::geography;
```

O tipo GEOGRAFIA calcula a verdadeira menor distância sobre a esfera entre Reykjavik e o grande caminho de voo circular entre Seattle e Londres.

**Great Circle mapper** A GEOMETRIA calcula a distância cartesiana insignificante entre Reykjavik e o caminho direto de Seattle para Londres marcado em um mapa. As unidades nominais do resultado podem ser chamadas de "graus", mas o resultado não corresponde a nenhuma diferença angular verdadeira entre os pontos, então, chamá-las de "graus" é incoerente.

## 4.2.2 Quando usar o tipo de dados Geografia sobre os dados Geometria

The geography type allows you to store data in longitude/latitude coordinates, but at a cost: there are fewer functions defined on GEOGRAPHY than there are on GEOMETRY; those functions that are defined take more CPU time to execute.

O tipo que você escolheu deveria ser condicionado da área de trabalho esperada da aplicação que você está construindo. Seus dados irão abranger o globo ou uma grande área continental, ou é local para um estado, condado ou município?

- Se seus dados estiverem contidos em uma pequena área, talvez perceba que escolher uma projeção apropriada e usar GEOMETRIA é a melhor solução, em termos de desempenho e funcionalidades disponíveis.
- Se seus dados são globais ou cobrem uma região continental, você pode perceber que GEOGRAFIA permite que você construa uma sistema sem ter que se preocupar com detalhes de projeção. Você armazena seus dados em longitude/latitude, e usa as funções que foram definidas em GEOGRAFIA.
- Se você não entende de projeções, não quer aprender sobre elas e está preparado para aceitar as limitações em funcionalidade disponíveis em GEOGRAFIA, então pode ser mais fácil se usar GEOGRAFIA em vez de GEOMETRIA. Simplesmente carregue seus dados como longitude/latitude e comece a partir daqui.

Recorra a Section [14.11](#) para uma comparação entre o que é suportado pela Geografia vs. Geometria. Para uma breve lista e descrição das funções da Geografia, recorra a Section [14.4](#)

## 4.2.3 FAQ de Geografia Avançada

### 1. *Você calcula na esfera ou esferoide?*

Por padrão, todos os cálculos de distância e área são feitos no esferoide. Você irá encontrar que os resultados dos cálculos nas áreas locais combinam com os resultados locais planares em boas projeções locais. Em grandes áreas, os cálculos esferoidais são mais precisos que os feitos em um plano projetado. Todas as funções de geografia têm a opção de usar um cálculo esférico, configurando um parâmetro booleano final para 'FALSO'. isto irá acelerar os cálculos, particularmente para casos onde as geometrias são bem simples.

### 2. *E a linha de data e os pólos?*

Nenhum cálculo possui a compreensão de linha de data ou polos, as coordenadas são esféricas (longitude/latitude), então uma forma que cruza a linha de data não é, de um ponto de cálculo de view, diferente de nenhuma outra forma.

### 3. *Qual é o maior arco que pode ser processado?*

Nós usamos grandes arcos círculos como a "linha de interpolação" entre dois pontos. Isso significa que quaisquer dois pontos estão de fato juntaram-se de duas maneiras, depende qual direção você vá no grande círculo. Todo o nosso código assume que os pontos estão juntos pelo \*menor\* dos dois caminhos ao longo do grande círculo. Como consequência, formas que têm arcos de mais de 180 graus não serão modeladas corretamente.

#### 4. Por que é tão lento para calcular a área da Europa / Rússia / insira uma grande região geográfica aqui ?

Porque o polígono é muito grande! Grandes áreas são ruins por duas razões: seus limites são grandes, logo o índice tende a puxar o traço, não importa qual consulta você execute; o número de vértices é enorme, e testes (distância, contenção) têm que atravessar a lista de vértices pelo menos uma vez e algumas vezes, N vezes (com N sendo o número de vértices em outra característica candidata). As with GEOMETRY, we recommend that when you have very large polygons, but are doing queries in small areas, you "denormalize" your geometric data into smaller chunks so that the index can effectively subquery parts of the object and so queries don't have to pull out the whole object every time. Please consult [ST\\_Subdivide](#) function documentation. Just because you \*can\* store all of Europe in one polygon doesn't mean you \*should\*.

## 4.3 Usando os Padrões OpenGIS

O OpenGIS "Especificação de Características Simples para SQL" define tipos padrão de objetos GIS, as funções requeridas para manipulá-los e um conjunto de tabelas de metadados. Querendo certificar-se de que os metadados permaneçam consistentes, operações como criar e remover uma coluna espacial são carregadas para fora dos procedimentos especiais definido pelo OpenGIS.

Existem duas tabelas de metadados OpenGIS: `SPATIAL_REF_SYS` e `GEOMETRY_COLUMNS`. A tabela `SPATIAL_REF_SYS` detém as IDs numéricas e descrições textuais de sistemas coordenados usadas no banco de dados espacial.

### 4.3.1 The `SPATIAL_REF_SYS` Table and Spatial Reference Systems

A tabela `spatial_ref_sys` está incluída no POstGIS e uma tabela do banco de dados OGC dependente que lista mais de 3000 [spatial reference systems](#) conhecidas e detalhes necessários para transformar/reprojetar entre eles.

Although the PostGIS `spatial_ref_sys` table contains over 3000 of the more commonly used spatial reference system definitions that can be handled by the proj library, it does not contain all known to man and you can define your own custom projection if you are familiar with proj4 constructs. Keep in mind that most spatial reference systems are regional and have no meaning when used outside of the bounds they were intended for.

Uma ótima fonte para encontrar sistemas de referência espacial não definidos na configuração central é <http://spatialreference.org/>

Alguns dos sistemas de referência espacial mais comumente usados são: [4326 - WGS 84 Long Lat](#), [4269 - NAD 83 Long Lat](#), [3395 - WGS 84 World Mercator](#), [2163 - US National Atlas Equal Area](#), Spatial reference systems para cada NAD 83, WGS 84 UTM zona - zonas UTM são as mais ideais para medição, mas só cobrem 6-graus regiões.

Vários estados dos EUA no sistema de referência espacial (em metros ou pés) - normalmente um ou 2 existem por estado. A maioria dos que estão em metros estão no centro, mas muitos dos que estão em pés ou foram criados por ESRI precisarão de [spatialreference.org](#).

Para maiores detalhes na determinação de qual zona UTM usar para sua área de interesse, veja [utmzone PostGIS plpgsql helper function](#).

A tabela `SPATIAL_REF_SYS` de definição está como segue:

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

As colunas `SPATIAL_REF_SYS` estão como segue:

**SRID** Um valor inteiro que só identifica o Sistema de Referência Espacial (SRS) dentro do banco de dados.

**AUTH\_NAME** O nome do corpo padrão ou corpos padrões que estão sendo citados por este sistema de referência. Por exemplo, "EPSG" seria um `AUTH_NAME` válido.



**AUTH\_SRID** A ID do sistema de referência espacial como definido pela autoridade citada no AUTH\_NAME. No caso do EPSG, isto é onde o código da projeção EPSG estaria.

**SRTEXT** A representação bem conhecida de texto do sistema de referência espacial. Um exemplo de uma representação WKT SRS é:

```
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]
]
```

Para uma listagem de códigos de projeção EPSG e suas correspondentes representações WKT, veja <http://www.opengeospatial.org>. Para uma discussão geral de WKT, veja o OpenGIS "Coordinate Transformation Services Implementation Specification" em <http://www.opengeospatial.org/standards>. Para maiores informações no European Petroleum Survey Group (EPSG) e no banco de dados deles de sistemas de referência espacial, veja <http://www.epsg.org>.

**PROJ4TEXT** O PostGIS usa a biblioteca Proj4 para fornecer capacidades de transformação de coordenada. A coluna PROJ4TEXT contém a string de definição da coordenada Proj4 para um SRID específico. Por exemplo:

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

Para maiores informações a respeito, veja o website do Proj4 <http://trac.osgeo.org/proj/>. O arquivo `spatial_ref_sys.sql` contém as definições SRTEXT e PROJ4TEXT para todas as projeções EPSG.

### 4.3.2 A GEOMETRY\_COLUMNS VIEW

GEOMETRY\_COLUMNS is a view reading from database system catalogs. Its structure is as follows:

```
\d geometry_columns
```

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

The column meanings are:

**F\_TABLE\_CATALOG, F\_TABLE\_SCHEMA, F\_TABLE\_NAME** O nome completo da tabela de característica que contém a coluna geométrica. Note que os termos "catálogo" e "esquema" são Oracle. Não existe um análogo do "catálogo" PostgreSQL, logo a coluna é deixada em branco -- para "esquema" o nome do esquema PostgreSQL é usado (`public` é o padrão).

**F\_GEOMETRY\_COLUMN** O nome da coluna geométrica na tabela característica.

**COORD\_DIMENSION** A dimensão espacial (2, 3 ou 4 dimensões) da coluna.

**SRID** A ID do sistema de referência espacial usada pela coordenada nesta coluna. É uma referência de chave estrangeira para SPATIAL\_REF\_SYS.

**TYPE** O tipo do objeto espacial. Para restringir a coluna espacial a um tipo só, use um dos: PONTO, LINESTRING, POLÍGONO, MULTIPONTO, MULTILINESTRING, MULTIPOLÍGONO, GEOMETRYCOLLECTION ou versões correspondentes XYM PONTOM, LINESTRINGM, POLÍGONOM, MULTIPOINTM, MULTILINESTRINGM, MULTIPOLÍGONOM, GEOMETRYCOLLECTIONM. Para coleções heterogêneas (do tipo mistas), você pode usar "GEOMETRIA" como o tipo.



**Note**

Este atributo (provavelmente) não é parte da especificação OpenGIS, mas é solicitada para assegurar a homogeneidade do tipo.

### 4.3.3 Criando uma Tabela Espacial

Criando uma tabela com dados espaciais, pode ser feito em um passo. Como mostra no exemplo seguinte, o qual cria tabelas de ruas com uma coluna de geometria linestring 2D em WGS84 long lat

```
CREATE TABLE ROADS (ID serial, ROAD_NAME text, geom geometry(LINESTRING,4326) );
```

Podemos adicionar colunas usando o comando padrão ALTERAR TABELA como fazemos no próximo exemplo onde adicionamos uma linestring 3-D.

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

### 4.3.4 Registrando manualmente as colunas geométricas em geometry\_columns

Two of the cases where you may need this are the case of SQL Views and bulk inserts. For bulk insert case, you can correct the registration in the geometry\_columns table by constraining the column or doing an alter table. For views, you could expose using a CAST operation. Note, if your column is typmod based, the creation process would register it correctly, so no need to do anything. Also views that have no spatial function applied to the geometry will register the same as the underlying table geometry column.

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

-- For it to register correctly
-- You need to cast the geometry
--
DROP VIEW public.vwmytablemercator;
CREATE VIEW public.vwmytablemercator AS
    SELECT gid, ST_Transform(geom, 3395)::geometry(Geometry, 3395) As geom, f_name
    FROM public.mytable;

-- If you know the geometry type for sure is a 2D POLYGON then you could do
DROP VIEW public.vwmytablemercator;
CREATE VIEW public.vwmytablemercator AS
    SELECT gid, ST_Transform(geom,3395)::geometry(Polygon, 3395) As geom, f_name
    FROM public.mytable;
```

```
-- Lets say you created a derivative table by doing a bulk insert
SELECT poi.gid, poi.geom, citybounds.city_name
INTO myschema.my_special_pois
FROM poi INNER JOIN citybounds ON ST_Intersects(citybounds.geom, poi.geom);

-- Create 2D index on new table
CREATE INDEX idx_myschema_myspecialpois_geom_gist
  ON myschema.my_special_pois USING gist(geom);

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

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

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

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.

```
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);
```

### Se executarmos em psql

```
\d pois_ny;
```

Observamos que elas são definidas de maneira diferente -- uma é typmod, outra é restrição

```
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)
```

Nas `geometry_columns`, elas registram corretamente

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

f_table_name	f_geometry_column	srid	type
pois_ny	geom	4326	POINT
pois_ny	geom_2160	2160	POINT

Entretanto -- se se quiséssemos criar uma view como essa

```
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';
```

A coluna baseada em typmod registra corretamente, mas a baseada em restrições não.

f_table_name	f_geometry_column	srid	type
vw_pois_ny_parks	geom	4326	POINT
vw_pois_ny_parks	geom_2160	0	GEOMETRY

Isto pode modificar as versões futuras do PostGIS, mas por enquanto para forçar a restrição baseada em coluna view registrar corretamente, precisamos fazer isto:

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

f_table_name	f_geometry_column	srid	type
vw_pois_ny_parks	geom	4326	POINT
vw_pois_ny_parks	geom_2160	2160	POINT

### 4.3.5 Assegurando a confirmação de geometrias OpenGIS

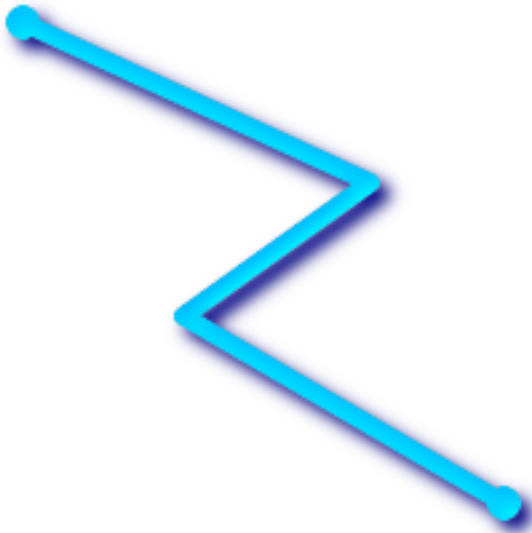
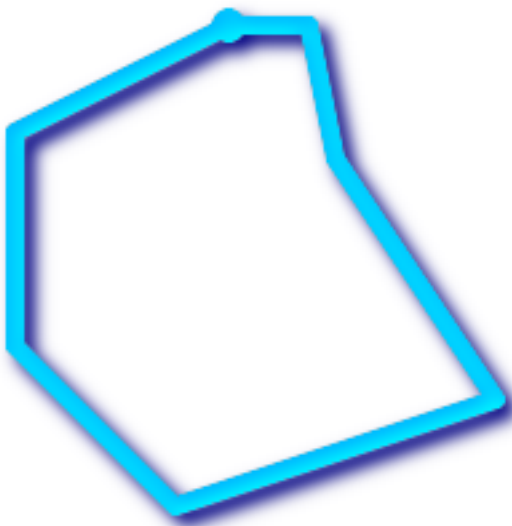
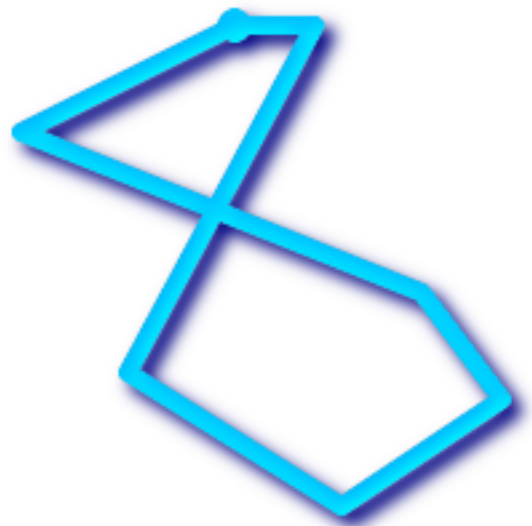
O PostGIS é condescendente com as Open Geospatial Consortium's (OGC) OpenGIS Specifications. Dessa forma, vários métodos PostGIS requerem, ou mais precisamente, presumem que as geometrias que são operadas são simples e válidas. Por exemplo, não faz sentido calcular a área de um polígono que tem um buraco definido fora do polígono, ou construir um polígono de uma linha delimitadora não simples.

De acordo com as especificações OGC, uma geometria *simple* é aquela que não possui pontos geométricos anômalos, como: auto interseção ou auto tangenciação e refere-se primeiramente a geometrias 0 ou 1-diemensional (ex.: [MULTI]POINT, [MULTI]LINESTRING). Por outro lado, a validade da geometria refere-se primeiramente a geometrias 2-dimensionais (ex.: [MULTI]POLYGON) e define o conjunto de afirmações que caracterizam um polígono válido. A descrição de cada classe de geometria inclui condições específicas que detalham mais a simplicidade e validade geométricas.

Um POINT é herdado *simple* como um objeto geométrico 0-dimensional.

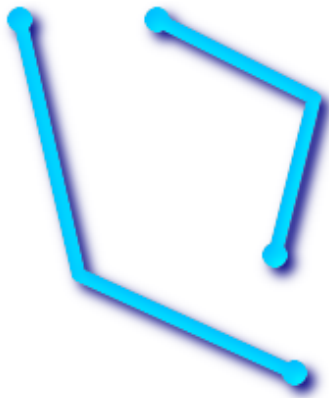
MULTIPOINTS são *simple* se nenhuma de duas coordenadas (POINTS) forem iguais (tenham o valor de coordenadas idêntico).

Uma LINESTRING é *simple* se não passa pelo mesmo POINT duas vezes (exceto para ponto finais, em cada caso são referidos como um anel linear e considerados fechados).

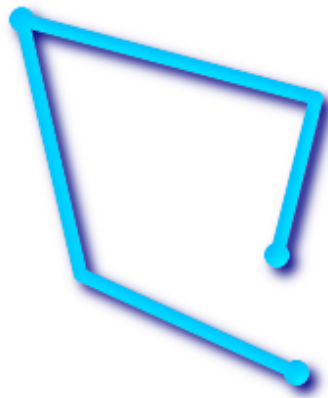
**(a)****(b)****(c)****(d)**

**(a)** e **(c)** são simples `LINESTRINGS`, **(b)** e **(d)** não são.

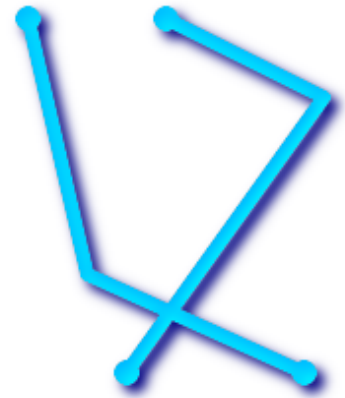
Uma `MULTILINESTRING` é *simple* somente se todos seus elementos forem simples e a única interseção entre qualquer um dos dois elementos ocorre em `POINTS` que estão nos limites dos dois elementos.



(e)



(f)



(g)

(e) e (f) são simples MULTILINESTRINGs, (g) não são.

Por definição, um POLYGON é sempre *simple*. Ele é *valid* se nenhum dos dois anéis (feitos de um exterior e um interior) no limite cruzar. O limite de um POLYGON pode intersectar em um POINT mas só como uma tangente (ex.: não em uma linha). Um POLYGON pode não ter linhas cortadas ou extremidades e os anéis interiores devem estar inteiramente contidos dentro dos anéis exteriores.



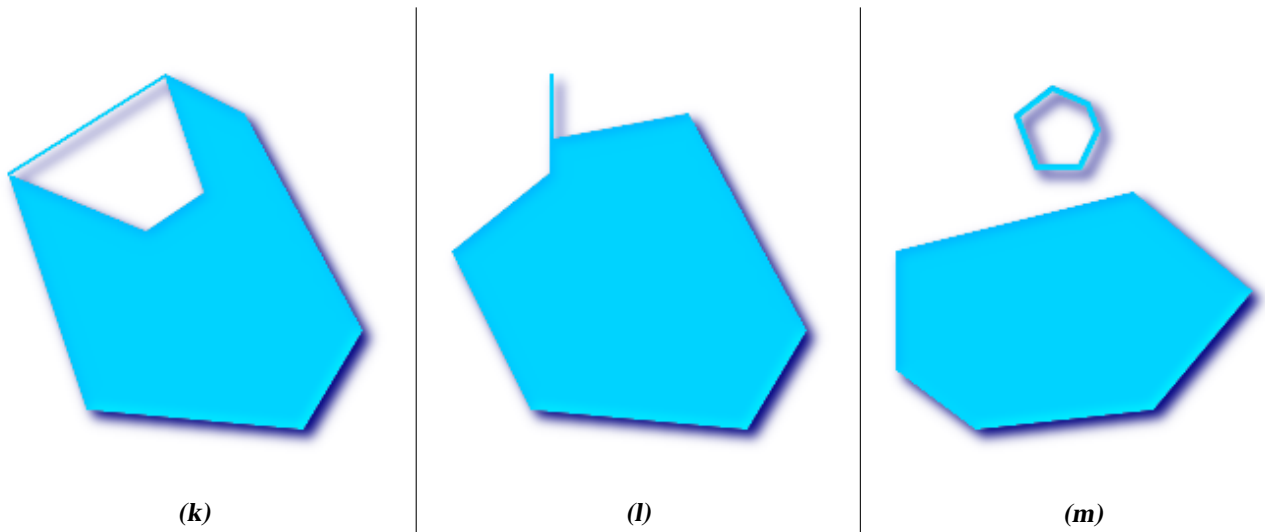
(h)



(i)

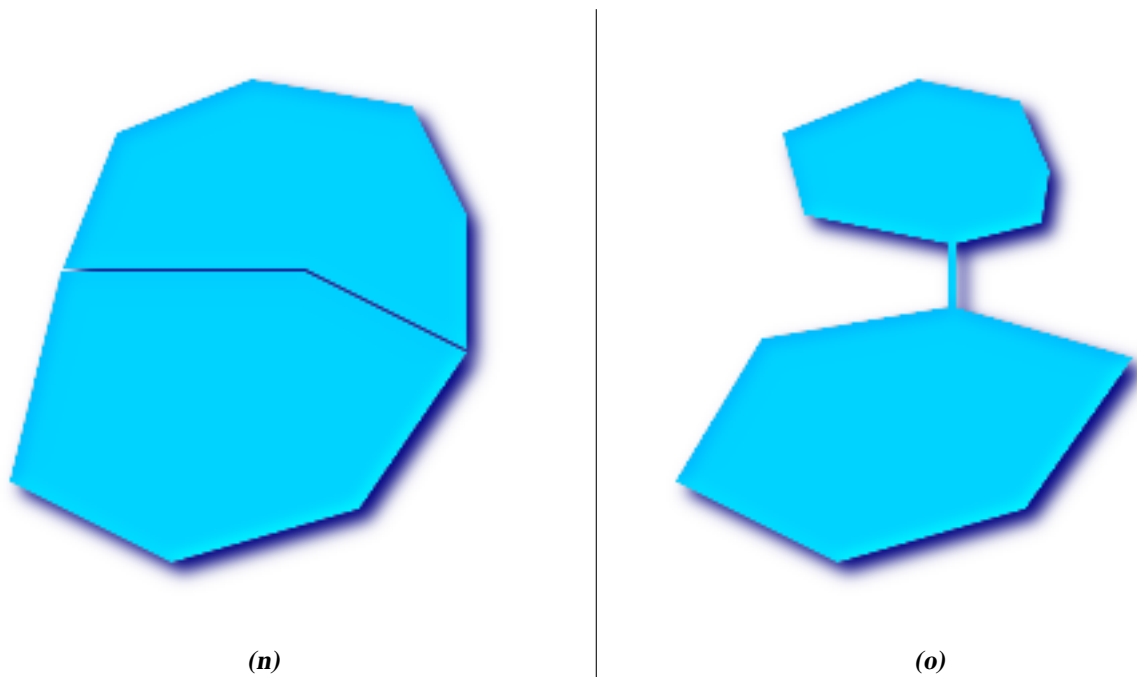


(j)



(h) e (i) são válidas POLYGONS, (j-m) não podem ser representados como POLYGONS, mas (j) e (m) podem ser representados como uma MULTIPOLYGON válida.

Um MULTIPOLYGON é *valid* somente se todos seus elementos forem válidos e os interiores de nenhum dos dois elementos intersectarem. Os limites de quaisquer dois elementos podem se tocar, mas somente em um número finito de POINTS.



(n) e (o) não são válidos MULTIPOLYGONS. (p), entretanto, é válido.

A maioria das funções implementadas pela biblioteca GEOS confiam na suposição de que suas geometrias são válidas como especificados pela OpenGIS Simple Feature Specification. Para verificar a simplicidade ou validade de geometrias, você pode usar a [ST\\_IsSimple\(\)](#) e [ST\\_IsValid\(\)](#)

```
-- Typically, it doesn't make sense to check
-- for validity on linear features since it will always return TRUE.
-- But in this example, PostGIS extends the definition of the OGC IsValid
```

```
-- by returning false if a LineString has less than 2 *distinct* vertices.
gisdb=# SELECT
  ST_IsValid('LINESTRING(0 0, 1 1)'),
  ST_IsValid('LINESTRING(0 0, 0 0, 0 0)');

 st_isvalid | st_isvalid
-----+-----
      t      |          f
```

Por padrão, o PostGIS não aplica essa verificação de validade na geometria de entrada, porque testar a validade requer muito tempo da CPU para geometrias complexas, especialmente polígonos. Se você não confia nas fontes dos seus dados, você pode executar uma verificação nas suas tabelas adicionando uma restrição de verificação:

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

If you encounter any strange error messages such as "GEOS Intersection() threw an error!" when calling PostGIS functions with valid input geometries, you likely found an error in either PostGIS or one of the libraries it uses, and you should contact the PostGIS developers. The same is true if a PostGIS function returns an invalid geometry for valid input.



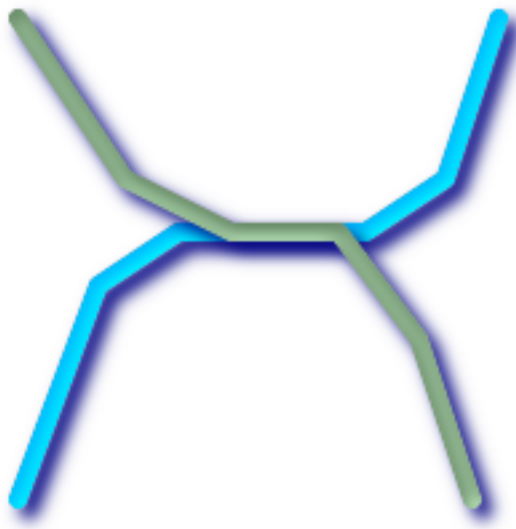
#### Note

Estritamente condescendentes geometrias OGC não podem ter valores Z ou M. A função `ST_IsValid()` não irá considerar geometrias com dimensões maiores inválidas! Invocações de `AddGeometryColumn()` irão adicionar uma restrição verificando dimensões de geometrias, então é suficiente especificar 2 lá.

### 4.3.6 Dimensionalidade estendida 9 Modelo de Interseção (DE-9IM)

It is sometimes the case that the typical spatial predicates (`ST_Intersects`, `ST_Contains`, `ST_Crosses`, `ST_Touches`, ...) are insufficient in and of themselves to adequately provide that desired spatial filter.

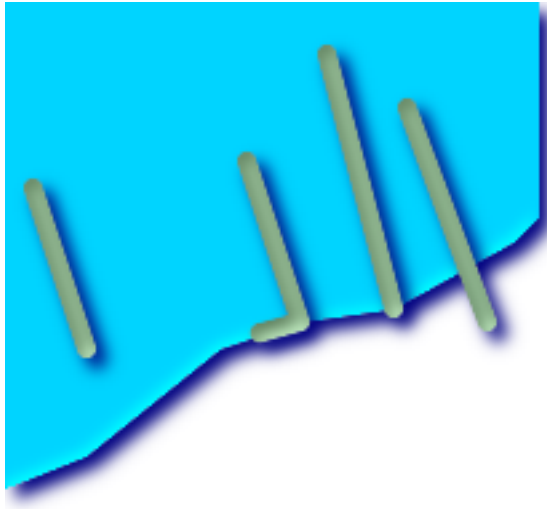




Considere um dataset linear representando uma rede de ruas. Pode ser a tarefa de um analista GIS identificar todos os segmentos de ruas que cruzam outras, não em um ponto, mas em uma linha, talvez invalidando alguma regra de negócio. Neste caso, a `ST_Crosses` não fornece adequadamente o filtro espacial necessário já que, para características lineares, ela retorna `true` só onde elas cruzam em um ponto.

Uma solução de dois passos pode ser primeiro representar a verdadeira interseção (`ST_Intersection`) de pares de segmentos de ruas que se intersectam espacialmente (`ST_Intersects`), e comparar as `ST_GeometryType` das intersecções com a 'LINESTRING' (lidando propriamente com casos que retornam `GEOMETRYCOLLECTIONS` de `[MULTI]POINTS`, `[MULTI]LINESTRINGs`, etc.).

Uma solução mais elegante/rápida pode ser de fato desejável.



Um segundo exemplo [teórico] pode ser aquele de um analista GIS tentando localizar todos os píeres ou docas que intersectam as margens de um lago em um linha e onde somente um fim do píer está em terra. Em outras palavras, onde um píer está, mas não completamente, dentro de um lago, e onde os pontos finais do píer estão completamente dentro e nas margens do lago. O analista pode precisar usar uma combinação de predicados espaciais para isolar o as características procuradas:

- `ST_Contains(lake, wharf) = TRUE`
- `ST_ContainsProperly(lake, wharf) = FALSE`
- `ST_GeometryType(ST_Intersection(wharf, lake)) = 'LINESTRING'`
- `ST_NumGeometries(ST_Multi(ST_Intersection(ST_Boundary(wharf), ST_Boundary(lake)))) = 1`  
... (desnecessário dizer, isto poderia se tornar um pouco complicado)

Então entre a Dimensionalidade estendida 9 Modelo de Interseção ou DE-9IM para abreviação.

#### 4.3.6.1 Teoria

De acordo com [OpenGIS Simple Features Implementation Specification for SQL](#), "a abordagem básica para comparar duas geometrias é fazer testes par-wise das interseções entre os interiores, limites e exteriores das duas geometrias e classificar a relação entre as duas geometrias baseada nas entradas e na matriz de "interseção" resultante."

##### Boundary

O limite de uma geometria é o conjunto de geometrias da próxima menor dimensão. Para POINTs, os quais têm uma dimensão 0, o limite é o conjunto vazio. O limite de uma LINESTRING são os dois pontos finais. Para POLYGONs, o limite é a linework que faz os anéis interiores e exteriores.

##### Interior

O interior de uma geometria são aqueles pontos deixados quando um limite é removido. Para POLYGONs, o interior é o próprio POLYGON. O interior de uma LINESTRING são o conjunto de pontos reais entre os pontos finais. Para POLYGONs, o interior é a superfície areal dentro do polígono.

##### Exterior

O exterior de uma geometria é o universo, uma superfícies areal, não no interior ou limite da geometria.

Dada a geometria  $a$ , onde o  $I(a)$ ,  $B(a)$ , and  $E(a)$  são *Interior*, *Boundary*, e *Exterior* de  $a$ , a representação matemática da matriz é:





























	<b>Interior</b>	<b>Boundary</b>	<b>Exterior</b>
<b>Interior</b>	$dim( I(a) \cap I(b) )$	$dim( I(a) \cap B(b) )$	$dim( I(a) \cap E(b) )$
<b>Boundary</b>	$dim( B(a) \cap I(b) )$	$dim( B(a) \cap B(b) )$	$dim( B(a) \cap E(b) )$
<b>Exterior</b>	$dim( E(a) \cap I(b) )$	$dim( E(a) \cap B(b) )$	$dim( E(a) \cap E(b) )$

Onde  $dim(a)$  é a dimensão de  $a$  como especificado em **ST\_Dimension** mas tem o domínio de  $\{0, 1, 2, T, F, *\}$

- 0 => point
- 1 => line
- 2 => area
- T =>  $\{0, 1, 2\}$
- F => empty set
- \* => don't care

Visualmente, para dois polígonos que se sobrepõem, isto se parece com:



	<table border="1"> <thead> <tr> <th></th> <th>Interior</th> <th>Boundary</th> <th>Exterior</th> </tr> </thead> <tbody> <tr> <th>Interior</th> <td>   <math>dim(...) = 2</math> </td> <td>   <math>dim(...) = 1</math> </td> <td>   <math>dim(...) = 2</math> </td> </tr> <tr> <th>Boundary</th> <td>   <math>dim(...) = 1</math> </td> <td>   <math>dim(...) = 0</math> </td> <td>   <math>dim(...) = 1</math> </td> </tr> <tr> <th>Exterior</th> <td>   <math>dim(...) = 2</math> </td> <td>   <math>dim(...) = 1</math> </td> <td>   <math>dim(...) = 2</math> </td> </tr> </tbody> </table>		Interior	Boundary	Exterior	Interior	 $dim(...) = 2$	 $dim(...) = 1$	 $dim(...) = 2$	Boundary	 $dim(...) = 1$	 $dim(...) = 0$	 $dim(...) = 1$	Exterior	 $dim(...) = 2$	 $dim(...) = 1$	 $dim(...) = 2$
		Interior	Boundary	Exterior													
	Interior	 $dim(...) = 2$	 $dim(...) = 1$	 $dim(...) = 2$													
Boundary	 $dim(...) = 1$	 $dim(...) = 0$	 $dim(...) = 1$														
Exterior	 $dim(...) = 2$	 $dim(...) = 1$	 $dim(...) = 2$														

Leia da esquerda para a direita e de cima para baixo, a matriz dimensional é representada, '212101212'.

Uma matriz conta que iria portanto representar nosso primeiro exemplo de duas linhas que intersectam em uma linha, seria: '1\*1\*\*\*1\*\*'

```
-- Identify road segments that cross on a line
SELECT a.id
FROM roads a, roads b
WHERE a.id != b.id
AND a.geom && b.geom
AND ST_Relate(a.geom, b.geom, '1*1***1**');
```

Uma matriz conta que representa o segundo exemplo de cais parcialmente na margem do lago seria '102101FF2'

```
-- Identify wharfs partly on a lake's shoreline
SELECT a.lake_id, b.wharf_id
FROM lakes a, wharfs b
```

```
WHERE a.geom && b.geom
AND ST_Relate(a.geom, b.geom, '102101FF2');
```

Para maiores informações ou leituras, veja:

- [OpenGIS Simple Features Implementation Specification for SQL](#) (version 1.1, section 2.1.13.2)
- [Dimensionalidade Estendida Modelo de Interseção-Nove \(DE-9IM\)](#)
- [FerramentasGeo: Teoria de Ponto e a Matriz DE-9IM](#)
- *Encyclopedia of GIS* By Hui Xiong

## 4.4 Carregando dados GIS (Vector)

Uma vez que tenha criado uma tabela espacial, você está pronto para atualizar os dados GIS no banco de dados. No momento, existe duas formas de colocar os dados no banco de dados PostGIS/PostgreSQL: usando as declarações SQL ou usando o shape file loader/dumper.

### 4.4.1 Carregando Dados Usando SQL

Se você puder converter seus dados para uma representação de texto, então usar SQL formatado pode ser mais fácil de colocar seus dados no PostGIS. Como com o Oracle e outros banco de dados SQL, dados só podem ser carregados em volume canalizando um grande arquivo de texto cheio de declarações SQL "INSERT" dentro do monitor SQL.

Um arquivo de atualização de dados (roads.sql por exemplo) deve se parecer com:

```
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;
```

O arquivo de dados pode ser canalizado para PostgreSQL facilmente usando o "psql" SQL monitor terminal:

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

### 4.4.2 shp2pgsql: Using the ESRI Shapefile Loader

O carregador de dados shp2pgsql converte ESRI Shape files em SQL adequado para inserção dentro de um banco de dados PostGIS/PostgreSQL, seja em formato de geometria ou geografia. O carregador possui vários modos de operação distinguidos pelas linhas de bandeiras de comando:

Juntamente com o comando carregador shp2pgsql, existe uma interface shp2pgsql-gui gráfica com a maioria das opções como o carregador, mas pode ser mais fácil de usar para um carregamento único non-scripted ou se você é novo no PostGIS. Pode ser configurado como um plugin do PgAdminIII.

(claldp) Essas são opções mutualmente exclusivas:

- c Cria uma tabela nova e popula do shapefile. *Este é o modo padrão.*
  - a Anexa dados do shapefile dentro do banco de dados da tabela. Note que para usar esta opção para carregar vários arquivos, eles devem ter os mesmos atributos e tipos de dados.
  - d Derruba a tabela do banco de dados, criando uma nova tabela com os dados do shapefile.
  - p Produz somente a criação da tabela do código SQL, sem adicionar nenhum dado de fato. Isto pode ser usado se você precisar separar completamente a tabela de criação e os passos de carregamento de dados.
- ? Exibir tela de ajuda.
- D** Use o formato PostgreSQL "dump" para os dados de saída. Pode ser combinado com -a, -c e -d. É muito mais rápido para carregar que o formato padrão "insert" SQL. Use isto para dados muito grandes.
- s** [**<FROM\_SRID>**;**>**;**<SRID>**] Cria e popula as tabelas de geometria com o SRID específico. Especifica, opcionalmente, que o shapefile de entrada usa o FROM\_SRID dado, caso em que as geometrias serão reprojatadas para o SRID alvo. FROM\_SRID não pode ser especificado com -D.
- k** Mantém identificadores (coluna, esquema e atributos). Note que os atributos no shapefile estão todos em CAIXAALTA.
- i** Coage todos os inteiros para 32-bit integers padrão, não cria 64-bit bigints, mesmo se a assinatura DBF parecer justificar ele.
- I** Cria um índice GiST na coluna geométrica.
- m** -m a\_file\_name Especifica um arquivo contendo um conjunto de mapas de nomes (longos) de colunas para nomes de colunas DBF com 10 caracteres. O conteúdo deste arquivo é uma ou mais linhas de dois nomes separados por um espaço branco e seguindo ou liderando espaço. Por exemplo:
- ```
COLUMNNAME DBFFIELD1
AVERYLONGCOLUMNNAME DBFFIELD2
```
- S** Gera geometrias simples em vez de MULTI geometrias. Só irá ter sucesso se todas as geometrias forem de fato únicas (ex.: um MULTIPOLÍGONO com uma única shell, ou um MULTIPONTO com um único vértice).
- t** **<dimensionality>** Força a geometria de saída a ter dimensionalidade especificada. Use as strings seguintes para indicar a dimensionalidade: 2D, 3DZ, 3DM, 4D.
- Se a entrada tiver poucas dimensões especificadas, a saída terá essas dimensões cheias com zeros. Se a entrada tiver mais dimensões especificadas, as que indesejadas serão tiradas.
- w** Gera o formato WKT em vez do WKB. Note que isto pode introduzir impulsos de coordenadas para perda de precisão.
- e** Execute cada declaração por si mesma, sem usar uma transação. Isto permite carregar a maioria dos dados bons quando existem geometrias ruins que geram erros. Note que não pode ser usado com a bandeira -D como o formato "dump" sempre usa a transação.
- W** **<encoding>** Especifica codificação dos dados de entrada (arquivo dbf). Quando usado, todos os atributos do dbf são convertidos da codificação especificada para UTF8. A saída SQL resultante conterá um comando SET CLIENT\_ENCODING to UTF8, então o backend será capaz de reconverter do UTF8 para qualquer codificação que o banco de dados estiver configurado para usar internamente.
- N** **<policy>** Políticas para lidar com geometrias NULAS (insert\*,skip,abort)
- n** -n Só importa arquivo DBF. Se seus dados não possuem shapefile correspondente, ele irá trocar automaticamente para este modo e carregar só o dbf. Então, só é necessário configurar esta bandeira se você tiver um shapefile completo, e se quiser os dados atributos e nenhuma geometria.
- G** Use geografia em vez de geometria (requer dados long/lat) em WGS84 long lat (SRID=4326)
- T** **<tablespace>** Especifica o espaço para a nova tabela. Os índices continuarão usando espaço padrão a menos que o parâmetro -X também seja usado. A documentação PostgreSQL tem uma boa descrição quando usa espaços personalizados.
- X** **<tablespace>** Especifica o espaço para os novos índices da tabela. Isto se aplica ao primeiro índice chave, e o índice GIST espacial, se -I também for usado.

Uma seção exemplo usando o carregador para criar um arquivo de entrada e atualizando ele pode parecer com:

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

Uma conversão e um upload podem ser feitos em apenas um passo usando encadeamento UNIX:

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

## 4.5 Recuperando dados GIS

Os dados podem ser extraídos do banco de dados usando o SQL ou o Shape file loader/dumper. Na seção do SQL discutiremos alguns dos operadores disponíveis para comparações e consultas em tabelas espaciais.

### 4.5.1 Usando SQL para recuperar dados

O mais simples significa extrair os dados do banco de dados, é usar uma consulta SQL para reduzir o número de RELATOS e COLUNAS retornados e abandonar as colunas resultantes dentro de um arquivo de texto analisável:

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

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

Entretanto, às vezes algum tipo de restrição será necessária para cortar o número de campos retornados. No caso de restrições baseadas em atributos, só use a mesma sintaxe SQL como normal com uma tabela não espacial. No caso de restrições espaciais, os operadores seguintes são úteis/disponíveis:

**ST\_Intersects** This function tells whether two geometries share any space.

= Isto testa se duas geometrias são geometricamente iguais. Por exemplo, se 'POLYGON((0 0,1 1,1 0,0 0))' é o mesmo que 'POLYGON((0 0,1 1,1 0,0 0))' (é).

Note: before PostGIS 2.4 this compared only boxes of geometries.

Next, you can use these operators in queries. Note that when specifying geometries and boxes on the SQL command line, you must explicitly turn the string representations into geometries function. The 312 is a fictitious spatial reference system that matches our data. So, for example:

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

A consulta acima retornaria um único relato da tabela "ROADS\_GEOM" na qual a geometria era igual ao valor.

To check whether some of the roads passes in the area defined by a polygon:

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



The most common spatial query will probably be a "frame-based" query, used by client software, like data browsers and web mappers, to grab a "map frame" worth of data for display.

Usando o operador "&&", você pode especificar uma CAIXA3D como uma característica de comparação ou uma GEOMETRIA. Entretanto, quando você especifica uma GEOMETRIA, a caixa delimitadora dela será usada para a comparação.

Using a "BOX3D" object for the frame, such a query looks like this:

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

Observe o uso do SRID 312, para especificar a projeção do envelope.

## 4.5.2 Usando o Dumper

A tabela dumper `pgsql2shp` conecta diretamente ao banco de dados e converte uma tabela (possivelmente definida por uma consulta) em um shapefile. A sintaxe básica é:

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

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

As opções da commandline são:

- f <filename>** Atribui a saída a um filename específico.
- h <host>** O hospedeiro do banco de dados para se conectar.
- p <port>** A porta para conectar no hospedeiro do banco de dados.
- P <password>** A senha para usar quando conectar ao banco de dados.
- u <user>** O nome de usuário para usar quando conectado ao banco de dados.
- g <geometry column>** No caso de tabelas com várias colunas geométricas, a coluna para usar quando atribuindo o shapefile.
- b** Use um cursor binário. Isto tornará a operação mais rápida, mas não funcionará se qualquer atributo NÃO-geométrico na tabela necessitar de um cast para o texto.
- r** Modo cru. Não derruba o campo `gid`, ou escapa o nome das colunas.
- m filename** Remapeia os identificadores para nomes com dez caracteres. O conteúdo do arquivo é linhas de dois símbolos separados por um único espaço branco e nenhum espaço seguindo ou à frente: `VERYLONGSYMBOL SHORTONE ANOTHERVERYLONGSYMBOL SHORTER` etc.

## 4.6 Construindo Índices

Indexes are what make using a spatial database for large data sets possible. Without indexing, any search for a feature would require a "sequential scan" of every record in the database. Indexing speeds up searching by organizing the data into a search tree which can be quickly traversed to find a particular record. PostgreSQL supports three kinds of indexes by default: B-Tree indexes, SP-GiST and GiST indexes.

- B-Trees are used for data which can be sorted along one axis; for example, numbers, letters, dates. Spatial data can be sorted along a space-filling curve, Z-order curve or Hilbert curve. This representation however does not allow speeding up common operations.
- GiST (Generalized Search Trees) dissolvem dados em "coisas de um lado", "coisas que sobrepõem", "coisas que estão dentro" e pode ser usado em vários tipos de dados, incluindo dados GIS. O PostGIS usa o índice R-Tree implementado no topo do GiST para classificar dados GIS.

### 4.6.1 Índices GiST

GiST significa "Árvores de Pesquisa Generalizada" e é uma forma genérica de classificar. Além disso, ele é usado para acelerar pesquisas em todos os tipos de estruturas de dados irregulares (arranjos inteiros, dados espectrais etc) que não são agradáveis à classificação normal B-Tree.

Uma vez que uma tabela de dados GIS excede pouco mais de mil filas, você irá querer construir um índice para acelerar pesquisas espaciais dos dados (a menos que suas pesquisas sejam baseadas em atributos, você vai querer construir um índice normal nos campos de atributo).

A sintaxe para construir um índice GiST em uma coluna "geométrica" é a seguinte:

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

The above syntax will always build a 2D-index. To get the an n-dimensional index for the geometry type, you can create one using this syntax:

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

Building a spatial index is a computationally intensive exercise. It also blocks write access to your table for the time it creates, so on a production system you may want to do in a slower CONCURRENTLY-aware way:

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

After building an index, it is sometimes helpful to force PostgreSQL to collect table statistics, which are used to optimize query plans:

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

### 4.6.2 BRIN Indexes

BRIN stands for "Block Range Index" and is a generic form of indexing that has been introduced in PostgreSQL 9.5. BRIN is a lossy kind of index, and its main usage is to provide a compromise for both read and write performance. Its primary goal is to handle very large tables for which some of the columns have some natural correlation with their physical location within the table. In addition to GIS indexing, BRIN is used to speed up searches on various kinds of regular or irregular data structures (integer, arrays etc).

Once a GIS data table exceeds a few thousand rows, you will want to build an index to speed up spatial searches of the data (unless all your searches are based on attributes, in which case you'll want to build a normal index on the attribute fields). GiST indexes are really performant as long as their size doesn't exceed the amount of RAM available for the database, and as long as you can afford the storage size, and the penalty in write workload. Otherwise, BRIN index can be considered as an alternative.

The idea of a BRIN index is to store only the bounding box englobing all the geometries contained in all the rows in a set of table blocks, called a range. Obviously, this indexing method will only be efficient if the data is physically ordered in a way where the resulting bounding boxes for block ranges will be mutually exclusive. The resulting index will be really small, but will be less efficient than a GiST index in many cases.

Building a BRIN index is way less intensive than building a GiST index. It's quite common to build a BRIN index in more than ten time less than a GiST index would have required. As a BRIN index only store one bounding box for one to many table blocks, it's pretty common to consume up to a thousand time less disk space for this kind of indexes.

You can choose the number of blocks to summarize in a range. If you decrease this number, the index will be bigger but will probably help to get better performance.

The syntax for building a BRIN index on a "geometry" column is as follows:

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

The above syntax will always build a 2D-index. To get a 3D-dimensional index, you can create one using this syntax

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

You can also get a 4D-dimensional index using the 4D operator class

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

These above syntaxes will use the default number or block in a range, which is 128. To specify the number of blocks you want to summarise in a range, you can create one using this syntax

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

Also, keep in mind that a BRIN index will only store one index value for a large number of rows. If your table stores geometries with a mixed number of dimensions, it's likely that the resulting index will have poor performance. You can avoid this drop of performance by choosing the operator class with the least number of dimensions of the stored geometries

Also the "geography" datatype is supported for BRIN indexing. The syntax for building a BRIN index on a "geography" column is as follows:

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

The above syntax will always build a 2D-index for geospatial objects on the spheroid.

Currently, just the "inclusion support" is considered here, meaning that just &&, ~ and @ operators can be used for the 2D cases (both for "geometry" and for "geography"), and just the &&& operator can be used for the 3D geometries. There is no support for kNN searches at the moment.

### 4.6.3 SP-GiST Indexes

SP-GiST stands for "Space-Partitioned Generalized Search Tree" and is a generic form of indexing that supports partitioned search trees, such as quad-trees, k-d trees, and radix trees (tries). The common feature of these data structures is that they repeatedly divide the search space into partitions that need not be of equal size. In addition to GIS indexing, SP-GiST is used to speed up searches on many kinds of data, such as phone routing, ip routing, substring search, etc.

As it is the case for GiST indexes, SP-GiST indexes are lossy, in the sense that they store the bounding box englobing the spatial objects. SP-GiST indexes can be considered as an alternative to GiST indexes. The performance tests reveal that SP-GiST indexes are especially beneficial when there are many overlapping objects, that is, with so-called "spaghetti data".

Once a GIS data table exceeds a few thousand rows, an SP-GiST index may be used to speed up spatial searches of the data. The syntax for building an SP-GiST index on a "geometry" column is as follows:

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

The above syntax will build a 2-dimensional index. A 3-dimensional index for the geometry type can be created using the 3D operator class:

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

Building a spatial index is a computationally intensive operation. It also blocks write access to your table for the time it creates, so on a production system you may want to do in in a slower CONCURRENTLY-aware way:

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

After building an index, it is sometimes helpful to force PostgreSQL to collect table statistics, which are used to optimize query plans:

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

An SP-GiST index can accelerate queries involving the following operators:

- `<<`, `&<`, `&>`, `>>`, `<<|`, `&<|`, `|&>`, `|>>`, `&&`, `@>`, `<@`, and `~=`, for 2-dimensional indexes,
- `&/&`, `~==`, `@>>`, and `<<@`, for 3-dimensional indexes.

There is no support for kNN searches at the moment.

#### 4.6.4 Usando Índices

Ordinarily, indexes invisibly speed up data access: once the index is built, the query planner transparently decides when to use index information to speed up a query plan. Unfortunately, the PostgreSQL query planner sometimes does not optimize the use of GiST indexes well, so sometimes searches which should use a spatial index instead may perform a sequential scan of the whole table.

Se você achar que seus índices não estão sendo usados (ou seus atributos) há algumas coisas que pode fazer:

- Firstly, read query plan and check your query actually tries to compute the thing you need. A runaway JOIN condition, either forgotten or to the wrong table, can unexpectedly bring you all of your table multiple times. To get query plan, add EXPLAIN keyword in front of your query.
- Second, make sure statistics are gathered about the number and distributions of values in a table, to provide the query planner with better information to make decisions around index usage. **VACUUM ANALYZE** will compute both.  
You should regularly vacuum your databases anyways - many PostgreSQL DBAs have **VACUUM** run as an off-peak cron job on a regular basis.
- If vacuuming does not help, you can temporarily force the planner to use the index information by using the **set enable\_seqscan to off;** command. This way you can check whether planner is at all capable to generate an index accelerated query plan for your query. You should only use this command only for debug: generally speaking, the planner knows better than you do about when to use indexes. Once you have run your query, do not forget to set `ENABLE_SEQSCAN` back on, so that other queries will utilize the planner as normal.
- If **set enable\_seqscan to off;** helps your query to run, your Postgres is likely not tuned for your hardware. If you find the planner wrong about the cost of sequential vs index scans try reducing the value of `random_page_cost` in `postgresql.conf` or using **set random\_page\_cost to 1.1;**. Default value for the parameter is 4, try setting it to 1 (on SSD) or 2 (on fast magnetic disks). Decreasing the value makes the planner more inclined of using Index scans.
- If **set enable\_seqscan to off;** does not help your query, it may happen you use a construction Postgres is not yet able to untangle. A subquery with inline select is one example - you need to rewrite it to the form planner can optimize, say, a LATERAL JOIN.

## 4.7 Consultas Complexas

The *raison d'être* of spatial database functionality is performing queries inside the database which would ordinarily require desktop GIS functionality. Using PostGIS effectively requires knowing what spatial functions are available, and ensuring that appropriate indexes are in place to provide good performance. The SRID of 312 used in these examples is purely for demonstration. You should be using a REAL SRID listed in the `spatial_ref_sys` table and one that matches the projection of your data. If your data has no spatial reference system specified, you should be THINKING very thoughtfully why it doesn't and maybe it should.

If your reason is because you are modeling something that doesn't have a geographic spatial reference system defined such as the internals of a molecule or the floorplan of a not yet built amusement park then that's fine. If the location of the amusement park has been planned however, then it would make sense to use a suitable planar coordinate system for that location if nothing more than to ensure the amusement part is not trespassing on already existing structures.

Even in the case where you are planning a Mars expedition to transport the human race in the event of a nuclear holocaust and you want to map out the Mars planet for rehabilitation, you can use a non-earthly coordinate system such as **Mars 2000** make one up and insert it in the `spatial_ref_sys` table. Though this Mars coordinate system is a non-planar one (it's in degrees spheroidal), you can use it with the `geography` type to have your length and proximity measurements in meters instead of degrees.

### 4.7.1 Tirando vantagem dos índices

Ao construir uma consulta é importante lembrar que somente os operadores baseados na caixa delimitadora como `&&` podem tirar vantagem do índice GiST. Funções como `ST_Distance()` não podem usar o índice para otimizar sua operação. Por exemplo, a consulta a seguir seria devagar em uma tabela grande:

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

This query is selecting all the geometries in `geom_table` which are within 100 units of the point (100000, 200000). It will be slow because it is calculating the distance between each point in the table and our specified point, ie. one `ST_Distance()` calculation for each row in the table. We can avoid this by using the single step index accelerated function `ST_DWithin` to reduce the number of distance calculations required:

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

This query selects the same geometries, but it does it in a more efficient way. Assuming there is a GiST index on `the_geom`, the query planner will recognize that it can use the index to reduce the number of rows before calculating the result of the `ST_Distance()` function. Notice that the `ST_MakeEnvelope` geometry which is used in the `&&` operation is a 200 unit square box centered on the original point - this is our "query box". The `&&` operator uses the index to quickly reduce the result set down to only those geometries which have bounding boxes that overlap the "query box". Assuming that our query box is much smaller than the extents of the entire geometry table, this will drastically reduce the number of distance calculations that need to be done.

### 4.7.2 Exemplos de SQL espacial

Os exemplos desta seção farão uso de duas tabelas, uma tabela de ruas lineares e uma de limites de municípios poligonais. As definições para a tabela `bc_roads` são:

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

A definição para a tabela `bc_municipality` é:

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

1. *Qual é o comprimento total de todas as ruas, expressado em quilômetros?*

Você pode responder esta questão com um pedaço simples de SQL:

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

km_roads
-----
70842.1243039643
(1 row)
```

## 2. *Quão grande é a cidade de Prince George em hectares?*

Esta consulta combina um atributo condição (no nome do município) com um cálculo espacial (da área):

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

hectares
-----
32657.9103824927
(1 row)
```

## 3. *Qual é o maior município na província, por área?*

Esta consulta traz uma media espacial dentro da consulta condição. Existem várias maneiras de abordar este problema, mas o mais eficiente está abaixo:

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

name          | hectares
-----+-----
TUMBLER RIDGE | 155020.02556131
(1 row)
```

Note que para responder esta pesquisa, tivemos que calcular a área de cada polígono. Se estivéssemos acostumados a fazer isto, faria sentido adicionar uma coluna área à tabela que poderíamos classificar separadamente para apresentação. Pedindo os resultados em uma direção descendente, e usando o comando PostgreSQL "LIMITE" podemos escolher o maior valor sem usar uma função agregada como max().

## 4. *Qual é o comprimento de ruas completamente contidas dentro de cada município?*

Este é um exemplo de uma "união espacial", porque estamos unindo dados de duas tabelas (fazendo uma união), mas usando a condição de interação espacial ("contida") como a condição de união em vez da maneira usual de unir a uma chave comum:

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

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

Esta consulta leva alguns instantes, pois cada rua na tabela está resumida no resultado final (por volta de 250K ruas para nossa tabela exemplo). Para coberturas menores (vários milhares de relatos em muitas centenas) a resposta pode ser bem rápida.

##### 5. Criar uma nova tabela como todas as ruas dentro da cidade de Prince George.

Este é um exemplo de uma "cobertura", que pega duas tabelas e gera uma nova tabela que consiste de resultantes cortadas. Diferente da "união espacial" demonstrada acima, esta consulta realmente cria novas geometrias. Uma cobertura é como uma união espacial turbo carregada, e é útil para trabalhos de análise mais exatos:

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

##### 6. Qual é o tamanho em quilômetros da "Douglas St" em Victoria?

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

kilometers
-----
4.89151904172838
(1 row)
```

##### 7. Qual é o maior município poligonal que tem um buraco?

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

gid | name           | area
-----+-----+-----
12  | SPALLUMCHEEN | 257374619.430216
(1 row)
```

## Chapter 5

# Gerência de dados raster, pesquisas e aplicações

### 5.1 Carregando e criando dados matriciais

Para a maioria dos casos, você usará a ferramenta `raster2pgsql` para carregar os dados matriciais para o PostGIS.

#### 5.1.1 Usando o `raster2pgsql` para carregar dados matriciais

O `raster2pgsql` é um carregador raster executável que carrega formatos raster, suportados pelo GDAL, para sql, carregando em uma tabela raster PostGIS. É capaz de carregar pastas de arquivos raster bem como criar overviews de rasters.

Já que o `raster2pgsql` está compilado como parte do PostGIS (a menos que você compile sua própria biblioteca GDAL), os tipos raster suportados pelo executável serão os mesmos que os compilados na biblioteca de dependência GDAL. Para pegar essa lista de tipos de rasters que seu `raster2pgsql` suporta, use o `-G`. Esses devem ser os mesmos daqueles fornecidos pela sua instalação PostGIS documentada aqui [ST\\_GDALDrivers](#) se você está usando a mesma biblioteca `gdal` para ambos.

**Note**

A versão mais antiga dessa ferramenta era um python script. O executável substituiu o python script. Se você continuar achando necessário os exemplos python script, um pode ser encontrado em: [GDAL PostGIS Raster Driver Usage](#). Por favor, note que o `raster2pgsql` python script pode não funcionar com versões futuras do raster PostGIS e não ser mais suportada.

**Note**

Na criação de overviews de um fator específico de um conjunto de rasters que estão alinhados, é possível que as overviews não se alinhem. Visite <http://trac.osgeo.org/postgis/ticket/1764> para ver um exemplo onde as overviews não se alinham.

**USO EXEMPLO:**

```
raster2pgsql raster_options_go_here raster_file someschema.sometable > out.sql
```

**-?** Tela de ajuda. A ajuda também é exibida se você não passar em nenhum argumento.

**-G** Imprimir os formatos de raster suportados.

**(claldp)** Essas são opções mutuamente exclusivas:



- c Criar nova table e popular ela com raster(s), *esse é o modo padrão*
- a Anexar raster(s) à uma table existente.
- d Derrubar table, criar nova e popular ela com raster(s)
- p Preparar modo, somente criar a table.

#### Processo raster: Solicitando restrições para registro apropriado nos catálogos raster

- C Solicitar restrições raster -- srid, pixelsize etc. para assegurar que o raster está registrado corretamente na view `raster_columns`.
- x Desativar configuração da extensão de restrição máxima. Empregado somente se a bandeira -C também for.
- r Configurar as restrições (especialmente únicas e telha de cobertura) para bloqueio normal. Empregado somente se a bandeira -C também for.

#### Processo raster: Parâmetros opcionais usados para manipular a entrada do conjunto de dados raster

- s <SRID> Saída raster designada com SRID específico. Se não for fornecida ou for zero, os metadados raster serão verificados para determinar um SRID apropriado.
- b **BAND** Índice (1-base) da banda para extrair de raster. Para mais de um índice de banda, separe com vírgula(.). Se não especificado, todas as bandas de raster serão extraídas.
- t **TILE\_SIZE** Corte raster em ladrilhos para ser inserido um por fileira na tabela. O `TILE_SIZE` é expressado como `LARGURAxALTURA` ou configurado para o valor "auto" para permitir o carregador computar um tamanho usando o primeiro raster e aplicando para os outros rasters.
- P Preenche a maioria das tiles da direita e de baixo para garantir que todas as tiles tenham a mesma largura e peso.
- R, --register Registrar o raster como o sistema de arquivos raster (out-db).  
Somente os metadados do raster e a localização do caminho para o raster estão armazenados no banco de dados (não os pixels).
- l **OVERVIEW\_FACTOR** Cria uma visão geral do raster. Para mais de um fator, separe com a vírgula(.). A visão geral da tabela de nomes segue o modelo `o_overview_factor_table`, onde `overview_factor` é um marcador de posição para um fator de visão geral numérico e `table` é substituído com a tabela de nome básica. A visão geral criada está armazenada no banco de dados e não é afetada por -R. Note que seu arquivo sql criado contém a tabela principal e as tabelas panoramas.
- N **NODATA** NODATA valor para usar em bandas sem um valor NODATA.

#### Parâmetros opcionais usados para manipular objetos do banco de dados

- q Identificadores wrap PostgreSQL em citações
  - f **COLUMN** Especificar nome de destinação da coluna raster, o padrão é 'rast'
  - F Adicionar uma coluna com o nome do arquivo
  - n **COLUMN** Especificar o nome da coluna filename. Sugere -F.
  - q Identificadores wrap PostgreSQL em citações.
  - I Cria um índice GiST na coluna raster.
  - M Vácuo analise a tabela raster.
  - k Pula NODATA verificações de valores para cada banda raster.
  - T **tablespace** Especificar o espaço da tabela para a nova tabela. Note que os índices (incluindo a chave primária) continuarão sendo usados o espaço de tabela padrão, a menos que a bandeira -X também esteja sendo usada.
  - X **tablespace** Especificar o espaço da tabela para a nova tabela. Isto se aplica à chave primária e ao índice espacial se a bandeira -I estiver sendo usada.
  - Y Utilize declarações copiadas em vez de inserir declarações.
- e Execute cada declaração individualmente, não use uma transação.
- E **ENDIAN** Controla endianidade da saída binária gerada do raster; especificamos 0 para XDR e 1 para NDR (padrão); somente a saída NDR é suportada agora

**-V version** Especificamos uma versão de formato de saída. O padrão é 0. Somente o 0 é suportado neste momento.

Uma sessão de exemplo usando o carregador para criar um arquivo de entrada fazendo upload de seus azulejos fragmentados em 100x100, pode ficar parecido com:



**Note**

Você pode nomear o esquema ex: `demelevation` em vez de `public.demelevation` e a tabela raster será criada no esquema padrão do banco de dados ou usuário

```
raster2pgsql -s 4326 -I -C -M *.tif -F -t 100x100 public.demelevation
> elev.sql
psql -d gisdb -f elev.sql
```

Uma conversão e u upload podem ser feitos em apenas um passo usando encadeamento UNIX:

```
raster2pgsql -s 4326 -I -C -M *.tif -F -t 100x100 public.demelevation | psql -d gisdb
```

Carregue as tiles em metros dos rasters do estado plano de Massachusetts no esquema chamado: `aerial` e crie uma view completa, tabelas panoramas de níveis 2 e 4, use o modo cópia para inserir (sem arquivos intermediários), e -e não força tudo em uma transação (é bom se você quiser ver dados em tabelas sem esperar nada por isso). Quebre os rasters em 128x128 pixel tiles e aplique restrições de rasters. Utilize o modo cópia em vez da tabela inserida. (-F) Inclui um campo chamado filename para conter o nome do arquivo de onde as tiles cortam.

```
raster2pgsql -I -C -e -Y -F -s 26986 -t 128x128 -l 2,4 bostonaerials2008/*.jpg aerials. ←
    boston | psql -U postgres -d gisdb -h localhost -p 5432
```

```
--get a list of raster types supported:
raster2pgsql -G
```

Os comandos -G geram uma lista parecida com

```
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
JAXA PALSAR Product Reader (Level 1.1/1.5)
Ground-based SAR Applications Testbed File Format (.gff)
ELAS
Arc/Info Binary Grid
Arc/Info ASCII Grid
GRASS ASCII Grid
SDTS Raster
DTED Elevation Raster
Portable Network Graphics
JPEG JFIF
In Memory Raster
Japanese DEM (.mem)
Graphics Interchange Format (.gif)
Graphics Interchange Format (.gif)
Envisat Image Format
Maptech BSB Nautical Charts
X11 PixMap Format
MS Windows Device Independent Bitmap
```

SPOT DIMAP  
AirSAR Polarimetric Image  
RadarSat 2 XML Product  
PCIDSK Database File  
PCRaster Raster File  
ILWIS Raster Map  
SGI Image File Format 1.0  
SRTMHGT File Format  
Leveller heightfield  
Terragen heightfield  
USGS Astrogeology ISIS cube (Version 3)  
USGS Astrogeology ISIS cube (Version 2)  
NASA Planetary Data System  
EarthWatch .TIL  
ERMapper .ers Labelled  
NOAA Polar Orbiter Level 1b Data Set  
FIT Image  
GRIdded Binary (.grb)  
Raster Matrix Format  
EUMETSAT Archive native (.nat)  
Idrisi Raster A.1  
Intergraph Raster  
Golden Software ASCII Grid (.grd)  
Golden Software Binary Grid (.grd)  
Golden Software 7 Binary Grid (.grd)  
COSAR Annotated Binary Matrix (TerraSAR-X)  
TerraSAR-X Product  
DRDC COASP SAR Processor Raster  
R Object Data Store  
Portable Pixmap Format (netpbm)  
USGS DOQ (Old Style)  
USGS DOQ (New Style)  
ENVI .hdr Labelled  
ESRI .hdr Labelled  
Generic Binary (.hdr Labelled)  
PCI .aux Labelled  
Vexcel MFF Raster  
Vexcel MFF2 (HKV) Raster  
Fuji BAS Scanner Image  
GSC Geogrid  
EOSAT FAST Format  
VTP .bt (Binary Terrain) 1.3 Format  
Erdas .LAN/.GIS  
Convair PolGASP  
Image Data and Analysis  
NLAPS Data Format  
Erdas Imagine Raw  
DIPEX  
FARSITE v.4 Landscape File (.lcp)  
NOAA Vertical Datum .GTX  
NADCON .los/.las Datum Grid Shift  
NTv2 Datum Grid Shift  
ACE2  
Snow Data Assimilation System  
Swedish Grid RIK (.rik)  
USGS Optional ASCII DEM (and CDED)  
GeoSoft Grid Exchange Format  
Northwood Numeric Grid Format .grd/.tab  
Northwood Classified Grid Format .grc/.tab  
ARC Digitized Raster Graphics  
Standard Raster Product (ASRP/USRP)  
Magellan topo (.blx)

```
SAGA GIS Binary Grid (.sdatt)
Kml Super Overlay
ASCII Gridded XYZ
HF2/HFZ heightfield raster
OziExplorer Image File
USGS LULC Composite Theme Grid
Arc/Info Export E00 GRID
ZMap Plus Grid
NOAA NGS Geoid Height Grids
```

### 5.1.2 Criando rasters utilizando as funções rasters do PostGIS

Em várias ocasiões, você vai querer criar rasters e tabelas rasters no banco de dados. Existe uma superabundância de funções que fazem isto. Os passos gerais para seguir.

1. Cria uma tabela com uma coluna raster para segurar os novos relatos rasters que são efetuados com:

```
CREATE TABLE myrasters(rid serial primary key, rast raster);
```

2. Existem várias funções para auxiliar com este objetivo. Se você não estiver criando rasters como derivados de outros rasters, você precisará de começar com: [ST\\_MakeEmptyRaster](#), seguido por [ST\\_AddBand](#)

Você também pode criar rasters de geometrias. Para alcançar seu objetivo, você irá querer usar [ST\\_AsRaster](#) talvez acompanhado com outras funções como: [ST\\_Union](#) ou [ST\\_MapAlgebraFct](#) ou qualquer um da família de outras funções álgebra de mapa.

Existem ainda mais opções para a criação de novas tabelas rasters a partir das tabelas existentes. Você pode criar uma tabela raster em uma projeção diferente de uma existente, por exemplo, utilizando: [ST\\_Transform](#)

3. Uma vez que você houver terminado de popular sua tabela inicialmente, você vai querer criar um índice espacial na coluna raster com algo parecido com:

```
CREATE INDEX myrasters_rast_st_convexhull_idx ON myrasters USING gist( ST_ConvexHull( ↵
rast) );
```

Note o uso do [ST\\_ConvexHull](#) já que a maioria dos operados raster são baseados no casco convexo dos rasters.



#### Note

Versões pre-2.0 do raster PostGIS eram baseadas no envelope em vez do casco convexo, Para os índices espaciais funcionar propriamente, você precisará derrubar estes e substituí-los com índice de casco convexo.

4. Aplique restrições rasters usando

## 5.2 Catálogos Raster

Existem duas view raster catalogadas que vêm compactadas com o PostGIS. Ambas views utilizam informações embutidas em restrições das tabelas rasters. Como resultado as views catalogadas são sempre consistentes com os dados raster nas tabelas já que as restrições são impostas.

1. `raster_columns` esta view cataloga todas as tabelas de colunas rasters no seu banco de dados.
2. `raster_overviews` esta view cataloga todas as tabelas de colunas raster no seu banco de dados que servem como um panorama para uma tabela granulada melhor. Tabelas deste tipo são geradas quando você utiliza o interruptor `-l` durante o carregamento.

## 5.2.1 Catálogo de Colunas Raster

O `raster_columns` é um catálogo de todas as colunas de tabela raster no seu banco de dados que são do tipo raster. É uma view utilizando as restrições nas tabelas para que a informação seja sempre consistente, mesmo se você tiver restaurado uma tabela raster de um backup de outro banco de dados. As seguintes colunas existem no catálogo `raster_columns`.

Se você criou suas tabelas sem o carregador ou esqueceu de especificar a bandeira `-C` durante o carregamento, você pode forçar as restrições depois de usá-las de fato `AddRasterConstraints` para que o catálogo `raster_columns` registre as informações comuns sobre as tiles raster.

- `r_table_catalog` O banco de dados que a tabela está. Isto irá sempre ler o banco de dados atual.
- `r_table_schema` O esquema do banco de dados que o raster pertence.
- `r_table_name` raster table
- `r_raster_column` a coluna é a `r_table_name` tabela que é do tipo raster. Não há nada no PostGIS que previna múltiplas colunas raster por tabela, assim, é possível haver uma tabela raster listada várias vezes com uma coluna raster diferente pra cada uma.
- `srid` O identificador de referência espacial do raster. Deve ser uma entrada no Section 4.3.1.
- `scale_x` A escala entre coordenadas geométricas espaciais e pixel. Isto só está disponível se todas as tiles na coluna raster tiverem a mesma `scale_x` e esta restrição for aplicada. Recorra a `ST_ScaleX` para mais detalhes.
- `scale_y` A escala entre coordenadas geométricas espaciais e pixel. Isto só está disponível se todas as tiles na coluna raster tiverem a mesma `scale_y` e a restrição `scale_y` for aplicada. Recorra a `ST_ScaleY` para mais detalhes.
- `blocksize_x` A largura (número de pixels obliquamente) de cada raster tile. Recorrer a `ST_Width` para mais detalhes.
- `blocksize_y` A largura (número de pixels para baixo) de cada raster tile. Recorrer a `ST_Height` para mais detalhes.
- `same_alignment` Uma booleana que é verdade se todos os rasters tiles têm o mesmo alinhamento. Recorrer a `ST_SameAlignment` para mais detalhes.
- `regular_blocking` Se a coluna raster possui a espacialidade única e cobre restrições tiles, o valor com ela se torna VERDADE. Senão, será FALSO.
- `num_bands` O número de bandas em cada tile do seu conjunto de raster. É a mesma informação da que é fornecida por `ST_NumBands`
- `pixel_types` Um arranjo definindo o tipo de pixel para cada banda. Você terá o mesmo número de elementos e bandas nesse arranjo. Os `pixel_types` são uns dos definidos em `ST_BandPixelType`.
- `nodata_values` Um arranjo de números preciso dobrados indicando o `nodata_value` para cada banda. Você terá o mesmo número de elementos e de bandas neste arranjo. Esses números definem o valor do pixel para cada banda que deveria ser ignorada para a maioria das operações. Uma informação parecida é fornecida por: `ST_BandNoDataValue`.
- `out_db` Um arranjo de bandeiras booleanas indicando se os dados das bandas rasters são mantidos de fora do banco de dados. Você terá o mesmo número de elementos e bandas neste arranjo.
- `extent` Isto é uma extensão de todas as filas raster no sua configuração raster. Se você planeja carregar mais dados que irão modificar a extensão de configuração, precisará executar a função `DropRasterConstraints` antes de carregar e então reaplicar as restrições com `AddRasterConstraints` depois carregar.
- `spatial_index` Uma booleana que é verdade se uma coluna raster possui um índice espacial.

## 5.2.2 Panoramas Raster

`raster_overviews` cataloga informação sobre as colunas de tabelas raster usadas para panoramas e informações adicionais sobre elas, que são úteis para saber quando usar os panoramas. As tabelas de panoramas são catalogadas em `raster_columns` e `raster_overviews`, porque elas são raster, mas também têm um propósito especial de serem uma caricatura de baixa resolução de uma tabela de alta resolução. Elas são geradas ao longo do lado da tabela raster principal quando você usa a troca `-l` no carregamento raster ou podem ser geradas manualmente, utilizando: [AddOverviewConstraints](#).

Tabelas resumidas contêm as mesmas restrições que as outras tabelas raster bem como informações adicionais somente restrições específicas para panoramas.



### Note

A informação em `raster_overviews` não duplica a informação em `raster_columns`. Se você precisa da informação sobre uma tabela panorama presente em `raster_columns`, você pode unir as `raster_overviews` e `raster_columns` para obter o conjunto completo de informações que precisa.

Duas razões principais para panoramas são:

1. Baixa resolução das tabelas de núcleo comumente usadas para um mapeamento de aproximação mais rápido.
2. Os cálculos são, geralmente, mais rápidos de serem feitos em si mesmos que a resolução mais alta de seus parentes, porque são relatos menores e cada pixel cobre mais território. Embora os cálculos não são tão atuais quanto as tabelas high-res que eles suportam, eles pode ser suficientes em vários cálculos da regra do polegar.

O catálogo `raster_overviews` contém as seguintes colunas de informação.

- `o_table_catalog` O banco de dados que o panorama está localizado. Isto sempre irá ler o banco de dados atual.
- `o_table_schema` O esquema do banco de dados que a tabela do panorama raster pertence.
- `o_table_name` nome da tabela de panorama raster
- `o_raster_column` a coluna raster na tabela panorama
- `r_table_catalog` O banco de dados que a tabela raster que este panorama está. Isto sempre lerá o banco de dados atual.
- `r_table_schema` O esquema do banco de dados da tabela raster que os serviços do panorama pertence.
- `r_table_name` tabela raster que este panorama fornece.
- `r_raster_column` a coluna raster que esta coluna panorama fornece.
- `overview_factor` - este é o nível da pirâmide da tabela panorama. Quanto maior o número menor a resolução da tabela. `raster2pgsql` se dada uma pasta de imagens, irá calcular panorama de cada arquivo de imagem e carregar separadamente. O nível 1 é assumido e sempre o arquivo original. Nível 2 terá que cada tile representa 4 do original. Então, por exemplo, se você tem uma pasta com arquivos de imagens de 5000x5000 pixel e você escolhe ordenar 125x125, para cada arquivo de imagem sua tabela base terá  $(5000*5000)/(125*125)$  records = 1600, your (l=2) `o_2` will have  $\text{ceiling}(1600/\text{Power}(2,2)) = 400$  rows, your (l=3) `o_3` will have  $\text{ceiling}(1600/\text{Power}(2,3)) = 200$  rows. Se seus pixels não são visíveis pelo tamanho das suas tiles, você pegará algumas tiles sobras (que não estão completamente cheias). Note que cada tile panorama gerada pelo `raster2pgsql` tem o meso número de pixels de seus pais, mas é de uma resolução menor onde cada pixel dele representa  $(\text{Power}(2,\text{overview\_factor})$  pixels do original).

## 5.3 Construindo Aplicações Personalizadas com o PostGIS Raster

O fato do PostGIS raster fornecer para você funções SQL para reproduzir rasters em imagens com formatos conhecidos, dá a você muitas opções para reproduzi-los. Por exemplo, você pode usar OpenOffice / LibreOffice para reproduzir como está demonstrado em: [Rendering PostGIS Raster graphics with LibreOffice Base Reports](#). Além disso, você pode usar uma grande variedade de linguagens como demonstrado nesta seção.

### 5.3.1 PHP Exemplo Outputting usando ST\_AsPNG em consenso co outras funções raster

Nesta seção, demonstraremos como usar o driver PHP PostgreSQL e a família **ST\_AsGDALRaster** de funções para gerar banda 1,2,3 de um raster para um fluxo de solicitação PHP que pode ser inserido em uma `img src html` tag.

A consulta exemplo demonstra como combinar um conjunto de funções raster para apanhar todas as tiles que intersectam uma caixa delimitadora wgs 84 e então une com **ST\_Union** as tiles intersectando retornando todas as bandas, transforma para projeção de usuário específico usando **ST\_Transform**, e gera os resultados como um png usando **ST\_AsPNG**.

Você poderia chamar o abaixo usando

```
http://mywebserver/test_raster.php?srid=2249
```

para obter a imagem raster no Massachusetts state plane feet.

```
<?php
/** contents of test_raster.php */
$conn_str = 'dbname=mydb host=localhost port=5432 user=myuser password=mypwd';
$dbconn = pg_connect($conn_str);
header('Content-Type: image/png');
/**If a particular projection was requested use it otherwise use mass state plane meters ←
**/
if (!empty( $_REQUEST['srid'] ) && is_numeric( $_REQUEST['srid'] ) ){
    $input_srid = intval($_REQUEST['srid']);
}
else { $input_srid = 26986; }
/** The set bytea_output may be needed for PostgreSQL 9.0+, but not for 8.4 **/
$sql = "set bytea_output='escape';
SELECT ST_AsPNG(ST_Transform(
    ST_AddBand(ST_Union(rast,1), ARRAY[ST_Union(rast,2),ST_Union(rast ←
    ,3]))
    , $input_srid) ) As new_rast
FROM aerials.boston
WHERE
    ST_Intersects(rast, ST_Transform(ST_MakeEnvelope(-71.1217, 42.227, -71.1210, ←
    42.218,4326),26986) )";
$result = pg_query($sql);
$row = pg_fetch_row($result);
pg_free_result($result);
if ($row === false) return;
echo pg_unescape_bytea($row[0]);
?>
```

### 5.3.2 ASP.NET C# Exemplo gerado usando ST\_AsPNG em consenso com outras funções raster

Nesta seção, demonstraremos como usar o driver PHP PostgreSQL .NET driver e a família **ST\_AsGDALRaster** de funções para gerar banda 1,2,3 de um raster para um fluxo de solicitação PHP que pode ser inserido em uma `img src html` tag.

Você precisará do driver `npgsql` .NET PostgreSQL para este exercício que pode ser obtido em: <http://npgsql.projects.postgresql.org/>. Apenas faça o download e coloque na sua pasta ASP.NET bin e você estará pronto.

A consulta exemplo demonstra como combinar um conjunto de funções raster para apanhar todas as tiles que intersectam uma caixa delimitadora wgs 84 e então une com **ST\_Union** as tiles intersectando retornando todas as bandas, transforma para projeção de usuário específico usando **ST\_Transform**, e gera os resultados como um png usando **ST\_AsPNG**.

Este é o mesmo exemplo de Section 5.3.1 exceto implementado em C#.

Você poderia chamar o abaixo usando

```
http://mywebserver/TestRaster.ashx?srid=2249
```

para obter a imagem raster no Massachusetts state plane feet.

```
-- web.config connection string section --
<connectionStrings>
  <add name="DSN"
    connectionString="server=localhost;database=mydb;Port=5432;User Id=myuser;password= ←
    mypwd"/>
</connectionStrings>
>
```

```
// Code for TestRaster.ashx
<%@ WebHandler Language="C#" Class="TestRaster" %>
using System;
using System.Data;
using System.Web;
using Npgsql;

public class TestRaster : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "image/png";
        context.Response.BinaryWrite(GetResults(context));
    }

    public bool IsReusable {
        get { return false; }
    }

    public byte[] GetResults(HttpContext context)
    {
        byte[] result = null;
        NpgsqlCommand command;
        string sql = null;
        int input_srid = 26986;
    try {
        using (NpgsqlConnection conn = new NpgsqlConnection(System. ←
            Configuration.ConfigurationManager.ConnectionStrings["DSN"]. ←
            ConnectionString)) {
            conn.Open();

            if (context.Request["srid"] != null)
            {
                input_srid = Convert.ToInt32(context.Request["srid"]);
            }
            sql = @"SELECT ST_Aspng(
                ST_Transform(
                    ST_AddBand(
                        ST_Union(rast,1), ARRAY[ST_Union(rast,2),ST_Union(rast,3)] ←
                            ,:input_srid) ) As new_rast
                FROM aerials.boston
                WHERE
                    ST_Intersects(rast,
                        ST_Transform(ST_MakeEnvelope(-71.1217, 42.227, ←
                            -71.1210, 42.218,4326),26986) )";
            command = new NpgsqlCommand(sql, conn);
            command.Parameters.Add(new NpgsqlParameter("input_srid", input_srid));

            result = (byte[]) command.ExecuteScalar();
            conn.Close();
        }
    }
}
```



```

    }

    }
    catch (Exception ex)
    {
        result = null;
        context.Response.Write(ex.Message.Trim());
    }
    return result;
}
}
}

```

### 5.3.3 O app console Java que gera a consulta raster como arquivo de imagem

Este é um exemplo de aplicativo console java que utiliza uma consulta que retorna uma imagem e gera um arquivo específico.

Você pode baixar os últimos drivers PostgreSQL JDBC de <http://jdbc.postgresql.org/download.html>

Você pode compilar o código seguinte usando um comando como:

```

set env CLASSPATH ../\postgresql-9.0-801.jdbc4.jar
javac SaveQueryImage.java
jar cfm SaveQueryImage.jar Manifest.txt *.class

```

E chama da linha de comando com algo tipo

```

java -jar SaveQueryImage.jar "SELECT ST_AsPNG(ST_AsRaster(ST_Buffer(ST_Point(1,5),10, '↔
quad_segs=2'),150, 150, '8BUI',100));" "test.png"

```

```

-- Manifest.txt --
Class-Path: postgresql-9.0-801.jdbc4.jar
Main-Class: SaveQueryImage

```

```

// Code for SaveQueryImage.java
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.io.*;

public class SaveQueryImage {
    public static void main(String[] argv) {
        System.out.println("Checking if Driver is registered with DriverManager.");

        try {
            //java.sql.DriverManager.registerDriver (new org.postgresql.Driver());
            Class.forName("org.postgresql.Driver");
        }
        catch (ClassNotFoundException cnfe) {
            System.out.println("Couldn't find the driver!");
            cnfe.printStackTrace();
            System.exit(1);
        }

        Connection conn = null;

        try {
            conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/mydb","myuser ↔
            ", "mypwd");
            conn.setAutoCommit(false);

```

```

PreparedStatement sGetImg = conn.prepareStatement(argv[0]);

ResultSet rs = sGetImg.executeQuery();

    FileOutputStream fout;
    try
    {
        rs.next();
        /** Output to file name requested by user **/
        fout = new FileOutputStream(new File(argv[1]) );
        fout.write(rs.getBytes(1));
        fout.close();
    }
    catch(Exception e)
    {
        System.out.println("Can't create file");
        e.printStackTrace();
    }

    rs.close();
    sGetImg.close();
    conn.close();
}
catch (SQLException se) {
    System.out.println("Couldn't connect: print out a stack trace and exit.");
    se.printStackTrace();
    System.exit(1);
}
}
}

```

### 5.3.4 Use PLPython para excluir imagens via SQL

Esta é uma função plpython armazenada que cria um arquivo no diretório do servidor para cada relato. Requer que tenha instalado plpython. Deve funcionar bem com plpythonu e plpython3u.

```

CREATE OR REPLACE FUNCTION write_file (param_bytes bytea, param_filepath text)
RETURNS text
AS $$
f = open(param_filepath, 'wb+')
f.write(param_bytes)
return param_filepath
$$ LANGUAGE plpythonu;

```

```

--write out 5 images to the PostgreSQL server in varying sizes
-- note the postgresql daemon account needs to have write access to folder
-- this echos back the file names created;
SELECT write_file(ST_AsPNG(
    ST_AsRaster(ST_Buffer(ST_Point(1,5),j*5, 'quad_segs=2'),150*j, 150*j, '8BUI',100)),
    'C:/temp/slices'|| j || '.png')
FROM generate_series(1,5) As j;

```

write\_file

```

-----
C:/temp/slices1.png
C:/temp/slices2.png
C:/temp/slices3.png
C:/temp/slices4.png
C:/temp/slices5.png

```

### 5.3.5 Rasters de saída com PSQL

Infelizmente o PSQL não tem facilidade em usar funcionalidade embutida para binários gerados. Isto é um pequeno hack no legado PostgreSQL de suporte de objetos grandes. Para usar, primeiro lance sua linha de comando psql conectada no seu banco de dados.

Diferente da aproximação python, esta cria o arquivo no seu computador local.

```
SELECT oid, lowrite(lo_open(oid, 131072), png) As num_bytes
FROM
( VALUES (lo_create(0),
  ST_AsPNG( (SELECT rast FROM aerals.boston WHERE rid=1) )
) ) As v(oid,png);
-- you'll get an output something like --
  oid  | num_bytes
-----+-----
2630819 |      74860

-- next note the oid and do this replacing the c:/test.png to file path location
-- on your local computer
\lo_export 2630819 'C:/temp/aerial_samp.png'

-- this deletes the file from large object storage on db
SELECT lo_unlink(2630819);
```

## Chapter 6

# Usando a Geometria do PostGIS: Criando aplicativos

## 6.1 Usando o MapServer

O MapServer de Minnesota é um servidor de mapas web que esta em conformidade com a especificação do OpenGIS Web Mapping Server.

- A página do MapServer esta em <http://mapserver.org>.
- A especificação OpenGIS Web Map encontra-se em <http://www.opengeospatial.org/standards/wms>.

### 6.1.1 Uso Básico

Para utilizar o PostGIS com o MapServer, você precisa saber como configurar o MapServer, o que esta além do escopo desta documentação. Esta seção ira cobrir questões relativas ao PostGIS, além de detalhes de configuração.

Para utilizar o PostGIS com o MapServer, você vai precisar:

- Versão 0.6 ou mais recente do PostGIS.
- Versão 3.5 ou mais recente do MapServer.

O MapServer acessa os dados do PostGIS/PostgreSQL como qualquer outro cliente PostgreSQL -- utilizando a interface libpq. Isso significa que o MapServer pode ser instalado em qualquer máquina com acesso à rede para o servidor do PostGIS, e usar o PostGIS como uma fonte de dados. Quanto mais rápida a conexão entre os sistemas, melhor.

1. Compile e instale o MapServer, com quaisquer opções que desejar, incluindo a opção de configuração "--with-postgis".
2. No seu arquivo de mapeamento do MapServer, adicione uma camada PostGIS. Por exemplo:

```
LAYER
  CONNECTIONTYPE postgis
  NAME "widehighways"
  # Connect to a remote spatial database
  CONNECTION "user=dbuser dbname=gisdatabase host=bigserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
  # Get the lines from the 'geom' column of the 'roads' table
  DATA "geom from roads using srid=4326 using unique gid"
  STATUS ON
  TYPE LINE
  # Of the lines in the extents, only render the wide highways
```

```

FILTER "type = 'highway' and numlanes >= 4"
CLASS
  # Make the superhighways brighter and 2 pixels wide
  EXPRESSION ([numlanes] >= 6)
  STYLE
    COLOR 255 22 22
    WIDTH 2
  END
END
CLASS
  # All the rest are darker and only 1 pixel wide
  EXPRESSION ([numlanes] < 6)
  STYLE
    COLOR 205 92 82
  END
END
END

```

No exemplo acima, as instruções específicas do PostGIS estão como segue:

**CONNECTIONTYPE** Para camadas PostGIS, este é sempre "postgis".

**CONEXÃO** A conexão do banco de dados é governada pela "string conexão" que é uma configuração padrão de chaves e valores como essa (com os valores padrões em <>):

```
user=<username> password=<password> dbname=<username> hostname=<server> port=<5432>
```

Uma string de conexão vazia continua válida, e quaisquer outros valores/chaves podem ser omitidos. No mínimo você fornece o nome e nome de usuário para o banco de dados para conectar.

**DADOS** Para formar esse parâmetro é "<geocolumn> de <tablename> using srid=<srid> usando único <primary key>" onde a coluna é a coluna espacial para ser reproduzida no mapa, a SRID é SRID usada pela coluna e a chave primária é a chave primária da table (ou qualquer outra coluna de valor único com um index).

Você pode omitir as orações "using srid" e "using unique" e o MapServer irá determinar automaticamente os valores possíveis se possível, mas ao custo de executar algumas pesquisas extras no servidor para cada desenho de mapa.

**PROCESSAMENTO** Colocando em um CLOSE\_CONNECTION=DEFER se você tem múltiplas camadas reutilizando conexões existentes ao invés de fechar elas. Isso melhora a velocidade. Para informações mais detalhadas vá para: [MapServer PostGIS Performance Tips](#).

**FILTRO** O filtro deve ser uma string SQL válida correspondente à normalidade lógica seguindo a palavra-chave "ONDE" em uma consulta SQL. Então, por exemplo, para representar caminhos com 6 ou mais pistas, use um filtro de "num\_lanes >= 6".

3. No seu banco de dados espacial, certifique-se que tenha indexes espaciais (GiST) construídos para qualquer uma das camadas que você irá desenhar.

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

4. Se você irá consultar suas camadas usando o MapServer, você também vai precisar usar a oração "using unique" no sua declaração de DADOS.

O MapServer requer identificadores únicos para cada registro espacial quando fazem-se consultas, e o módulo do PostGIS do MapServer usa o único valor que você especifica a fim de fornecer esse identificadores. Utilizar a chave primária da table é a melhor prática.

## 6.1.2 Perguntas Frequentes

1. *Quando eu uso uma EXPRESSÃO no meu arquivo de mapa, a condição nunca retorna como verdadeira, mesmo que eu saiba os valores que existem na minha table.*

Diferentemente dos shape files, os campos de nomes do PostGIS precisam ser referenciados em EXPRESSÕES usando letra minúscula.

```
EXPRESSÃO ([numlanes] >= 6)
```

2. *O FILTRO que eu uso para meus shape files não estão funcionando para minha table PostGIS dos mesmo dados.*

Diferentemente dos shpe files, filtros para as camadas PPostGIS usam sintaxe SQL (elas estão anexadas à declaração SQL que o conector do PostGIS gera para desenhar camadas no MapServer).

```
FILTER "type = 'highway' and numlanes >= 4"
```

3. *Minha camada PostGIS desenha muito mais devagar que minha camada shape file, isso é normal?*

Em geral, quanto mais características você desenhar em um mapa mais o PostGIS se tornará mais devagar que os shape files. Para mapas com poucas características (100s), o PostGIS será mais rápido. Para mapas com características de alta densidade (1000s), o PostGIS sempre será mais devagar. Se você está encontrando problemas substanciais de apresentação de desenho, é possível que você não tenha construído um index espacial na sua table.

```
postgis# CREATE INDEX geotable_gix ON geotable USING GIST ( geocolumn );
postgis# VACUUM ANALYZE;
```

4. *Minha camada PostGIS desenha bem, mas as consultas são bastante devagar. O que está errado?*

Para as pesquisas serem mais rápidas, você deve ter uma única chave para sua spatial table e deve ter um index nessa chave única. Você pode especificar qual chave única para o mapserver para usar com a oração `USING UNIQUE` na sua linha DATA:

```
DADOS "geom FROM geotable USING UNIQUE gid"
```

5. *Posso usar colunas "geografia" (novo em PostGIS 1.5) como fonte para minhas camadas MapServer?*

Sim! O MapServer entende colunas geografia sendo o mesmo que colunas geometria, mas sempre usando uma SRID de 4326. Só certifique-se de incluir uma oração "using srid=4326" na sua declaração DATA. Todo o resto funciona exatamente da mesma forma que com geometria.

```
DADOS "geog FROM geogtable USING SRID=4326 USING UNIQUE gid"
```

### 6.1.3 Uso Avançado

A pseudo oração SQL `USING` é usada para adicionar algumas informações para ajudar o mapserver a entender os resultados de pesquisas mais avançadas. Mais especificamente, quando uma view ou uma subselect são usadas como a source table (a coisa para a direita do "DE" em uma definição DATA) é mais difícil para o mapserver determinar automaticamente um identificador único para cada fila e também a SRID para a table. A oração `USING` pode fornecer o mapserver com essas duas informações:

```
DATA "geom FROM (
  SELECT
    table1.geom AS geom,
    table1.gid AS gid,
    table2.data AS data
  FROM table1
  LEFT JOIN table2
  ON table1.id = table2.id
) AS new_table USING UNIQUE gid USING SRID=4326"
```

**USING UNIQUE <uniqueid>** O MapServer requer uma id única para cada fila a fim de identificar a linha quando estiver fazendo consultas de mapa. Normalmente, ele identifica a chave primária do sistema de tables. Contudo, as views e subselects não têm automaticamente uma coluna única conhecida. Se você quiser usar a funcionalidade pesquisa do MapServer, você precisa certificar-se que sua view ou subselect inclui uma única coluna com valor, e declare isso com: `USING UNIQUE`. Por exemplo, você poderia explicitamente selecionar o nascimento dos valores da chave primária da table para esse propósito, ou qualquer outra coluna que se garante ser única para o resultado estabelecido.

**Note**

"Pesquisando um Mapa" é a ação de clicar em um mapa para perguntar sobre informações sobre as características naquela localização. Não confunda "mapa pesquisa" com a pesquisa SQL em uma definição DATA.

**USING SRID=<srid>** O PostGIS precisa saber qual sistema de referência espacial está sendo utilizado pelas geometrias a fim de retornar os dados corretos para o MapServer. Normalmente, é possível encontrar essa informação na table "geometry\_columns" no banco de dados do PostGIS, porém, não é possível para as tables que são criadas rapidamente como subselects e views. Então, a opção `USING SRID=` permite a SRID correta ser especificada na definição DATA.

### 6.1.4 Exemplos

Vamos começar com um exemplo simples e trabalhar com ele. Considere a seguinte definição de camada MapServer:

```
LAYER
  CONNECTIONTYPE postgis
  NAME "roads"
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  DATA "geom from roads"
  STATUS ON
  TYPE LINE
  CLASS
    STYLE
      COLOR 0 0 0
    END
  END
END
```

Essa camada irá expor todas as geometrias de rua nas roads tables como linhas pretas.

Agora, digamos que queremos mostrar somente as estradas antes de aproximarmos para uma escala 1:100000 - as próximas duas camadas irão alcançar esse efeito:

```
LAYER
  CONNECTIONTYPE postgis
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
  DATA "geom from roads"
  MINSCALE 100000
  STATUS ON
  TYPE LINE
  FILTER "road_type = 'highway'"
  CLASS
    COLOR 0 0 0
  END
END
LAYER
  CONNECTIONTYPE postgis
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
  DATA "geom from roads"
  MAXSCALE 100000
  STATUS ON
  TYPE LINE
  CLASSITEM road_type
  CLASS
    EXPRESSION "highway"
    STYLE
      WIDTH 2
      COLOR 255 0 0
```

```

    END
  END
  CLASS
  STYLE
    COLOR 0 0 0
  END
END
END
END

```

A primeira camada é usada quando a escala é maior que 1:100000, e exibe apenas as ruas do tipo "estrada" como linhas pretas. A opção `FILTRO` faz com que apenas as ruas do tipo "estradas" sejam exibidas.

A segunda camada é usada quando a escala é menor que 1:100000, e irá exibir estradas como duas linhas vermelhas grossas, e outras ruas como linhas pretas normais.

Portanto, fizemos algumas coisas interessantes utilizando apenas a funcionalidade MapServer, mas nossa declaração SQL `DATA` continuou simples. Suponha que o nome da rua está guardado em outra table (por alguma razão) e precisamos ingressar para pegar ele e etiquetar nossas ruas.

```

LAYER
  CONNECTIONTYPE postgis
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  DATA "geom FROM (SELECT roads.gid AS gid, roads.geom AS geom,
    road_names.name as name FROM roads LEFT JOIN road_names ON
    roads.road_name_id = road_names.road_name_id)
    AS named_roads USING UNIQUE gid USING SRID=4326"
  MAXSCALE 20000
  STATUS ON
  TYPE ANNOTATION
  LABELITEM name
  CLASS
  LABEL
    ANGLE auto
    SIZE 8
    COLOR 0 192 0
    TYPE truetype
    FONT arial
  END
END
END
END

```

Essa camada de comentário adiciona etiquetas verdes a todas as ruas quando a escala fica abaixo de 1:20000 ou menor que isso. Ela também demonstra como usar um ingresso SQL em uma definição `DATA`.

## 6.2 Clientes Java (JDBC)

Os clientes Java podem acessar os objetos "geometria" do PostGIS no banco de dados PostgreSQL diretamente como representações de textos ou usando a extensão JDBC de objetos empacotados com PostGIS. A fim de usar os objetos da extensão, o arquivo "postgis.jar" deve estar no seu `CLASSPATH` junto com o "postgresql.jar" do pacote de dispositivos JDBC.

```

import java.sql.*;
import java.util.*;
import java.lang.*;
import org.postgis.*;

public class JavaGIS {

public static void main(String[] args) {

    java.sql.Connection conn;

```



```

try {
    /*
    * Load the JDBC driver and establish a connection.
    */
    Class.forName("org.postgresql.Driver");
    String url = "jdbc:postgresql://localhost:5432/database";
    conn = DriverManager.getConnection(url, "postgres", "");
    /*
    * Add the geometry types to the connection. Note that you
    * must cast the connection to the postgres-specific connection
    * implementation before calling the addDataType() method.
    */
    ((org.postgresql.PGConnection) conn).addDataType("geometry", Class.forName("org.postgis. ↵
        PGgeometry"));
    ((org.postgresql.PGConnection) conn).addDataType("box3d", Class.forName("org.postgis. ↵
        PGbox3d"));
    /*
    * Create a statement and execute a select query.
    */
    Statement s = conn.createStatement();
    ResultSet r = s.executeQuery("select geom,id from geomtable");
    while( r.next() ) {
        /*
        * Retrieve the geometry as an object then cast it to the geometry type.
        * Print things out.
        */
        PGgeometry geom = (PGgeometry)r.getObject(1);
        int id = r.getInt(2);
        System.out.println("Row " + id + ":");
        System.out.println(geom.toString());
    }
    s.close();
    conn.close();
}
catch( Exception e ) {
    e.printStackTrace();
}
}
}

```

O objeto "PGgeometry" é um objeto wrapper que contém um objeto específico de geometria topológica (subclasse da classe abstrata "Geometria") dependendo do tipo: Ponto, LineString, Polígono, MultiPonto, MultiLineString, MultiPolígono.

```

PGgeometry geom = (PGgeometry)r.getObject(1);
if( geom.getType() == Geometry.POLYGON ) {
    Polygon pl = (Polygon)geom.getGeometry();
    for( int r = 0; r < pl.numRings(); r++ ) {
        LinearRing rng = pl.getRing(r);
        System.out.println("Ring: " + r);
        for( int p = 0; p < rng.numPoints(); p++ ) {
            Point pt = rng.getPoint(p);
            System.out.println("Point: " + p);
            System.out.println(pt.toString());
        }
    }
}
}
}

```

O JavaDoc para os objetos de extensão fornece uma referência para os dados variados das funções accessor nos objetos geométricos.

## **6.3 Clientes C (libpq)**

...

### **6.3.1 Cursores de Texto**

...

### **6.3.2 Cursores Binários**

...

---

## Chapter 7

# Dicas de desempenho

## 7.1 Pequenas tabelas de grandes geometrias

### 7.1.1 Descrição do problema

Versões atuais do PostgreSQL (incluindo a 8.0) sofrem de um problema no otimizador de queries quando falamos de tabelas TOAST. As tabelas TOAST são extensões utilizadas para armazenamento de grandes valores (no sentido de tamanho do dado) que não cabem normalmente nas páginas de dados (grandes blocos de texto, imagens ou geometrias complexas com muitos vértices, veja [a documentação oficial](#) para maiores informações).

Este problema ocorre se você possui tabelas com geometrias grandes, mas não muitas linhas (uma tabela dos limites todos os países europeus em alta resolução). A tabela em si, é pequena, mas utiliza muito espaço TOAST. Em nosso exemplo, a tabela em si possuía apenas 80 linhas e utilizava apenas 3 páginas de dados, mas a tabela TOAST utilizava 8225 páginas de dados.

Emita uma pesquisa onde você utiliza o operador `&&` para pesquisa por um retângulo envolvente que bate com poucas dessas linhas. O otimizador de pesquisas ve esta tabela contendo apenas 3 páginas e 80 linhas. Como a tabela é pequena, ele estima que um scan sequencial em uma tabela tão pequena será mais rápida do que utilizar um índice, ignorando o mesmo. Geralmente esta estimativa é correta, mas em nosso caso o operador `&&` tem que buscar todas as geometrias em disco para comparação dos retângulos envolventes, lendo todas as páginas TOAST também.

Para visualizar se você sofre com este bug, utilize um "EXPLAIN ANALYZE" na pesquisa em questão. Para maiores informações e detalhes técnicos, você pode recorrer a lista do postgres sobre desempenho: <http://archives.postgresql.org/pgsql-performance/2005-02/msg00030.php>

and newer thread on PostGIS <https://lists.osgeo.org/pipermail/postgis-devel/2017-June/026209.html>

### 7.1.2 Soluções

O pessoal responsável pelo PostgreSQL está tentando resolver esta questão por transformar o otimizador de pesquisas ciente das tabelas TOAST. Por enquanto, existem duas soluções:

A primeira solução é forçar o estimador de pesquisar a utilizar o índice. Emita um comando "SET enable\_seqscan TO off" ao servidor antes de emitir a pesquisa. Isto força o estimador a evitar scans sequenciais sempre que possível, utilizando o índice GIST como de costume. Mas esta flag deve ser setada para cada conexão e causa o estimador a decidir mal em outros casos, portanto, você deve habilitar "SET enable\_seqscan TO on;" após a pesquisa.

A segunda solução é fazer a pesquisa sequencial tão rápida quanto o estimador imagina. Isto pode ser feito criando uma coluna adicional que cacheia o retângulo envolvente e realizando as pesquisas em cima desta coluna. Em nosso exemplo, os comandos são:

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

Altere sua query para usar o operador && contra o retângulo envolvente ao invés da colunas geométrica, assim:

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

Claro, se você alterar ou adicionar colunas a mytable, você deve manter o retângulo envolvente em sincronia. A forma mais transparente de fazer isto seria através de triggers, mas você também quer modificar sua aplicação para manter a coluna do retângulo envolvente atualizada or executar a query de UPDATE após cada modificação.

## 7.2 CLUSTERizando índices geométricos

Para tabelas que são basicamente somente-leitura, e onde um único índice é utilizado pela maioria das queries, PostgreSQL oferece o comando CLUSTER. Este comando fisicamente reordena todas as linhas da tabela assim como as do índice, assim possibilitando duas melhorias de desempenho: primeiro, para pesquisas de intervalo de índice, o número de pesquisas na tabela de dados é dramaticamente reduzido. Segundo, se seu conjunto de trabalho concentra-se em pequenos intervalos nos índices, você tem um cache mais eficiente, pois todas as informações estão divididas em poucas páginas de dados. (Sinta se convidado para ler a documentação do comando CLUSTER do manual do PostgreSQL.)

Contudo, atualmente o Postgresql não permite a clusterização de índices geométricos GIST, pois estes índices simplesmente ignoram valores nulos, retornando um erro como:

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

Como a HINT da mensagem te diz, você pode adicionar uma constraint "not null" na tabela para contornar o problema.

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

Claro, isto não vai funcionar se você de fato precisa de valores NULL em sua coluna geométrica. Adicionalmente, você deve usar o método acima para adicionar a constraint. Utilizar uma constraint do tipo CHECK como "ALTER TABLE blubb ADD CHECK (geometry is not null);" não irá funcionar.

## 7.3 Evitando conversão de dimensões

Algumas vezes, você tem dados que são 3D ou 4D em sua tabela, mas sempre acessa-os usando métodos OpenGIS, como ST\_AsText() ou ST\_AsBinary(), que somente funcionam em geometrias 2D. Eles fazem isso internamente chamando a função ST\_Force2D(), que introduza um gasto extra para grandes geometrias. Para evitar este gasto extra, pode ser viável dropar essas dimensões adicionais para sempre:

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

Note que se você adicionou sua coluna geométrica utilizando o método AddGeometryColumn(), existirá uma constraint na dimensão da geometria. Para contornar isto, você precisará dropar a constraint também. Lembre-se de atualizar a entrada na tabela geometry\_columns e recriar a constraint posteriormente.

No caso de grandes tabelas, pode ser sábio dividir este UPDATE em porções menores, restringindo o UPDATE a pequenas partes da tabela com o uso de uma cláusula WHERE sobre sua PRIMARY KEY ou outro critério, rodando um VACUUM, entre os UPDATES. Isto reduz drasticamente a necessidade de espaço em disco temporário. Adicionalmente, se você tem geometrias de dimensões mistas, restringir o UPDATE por "WHERE dimension(the\_geom)>2" pula as geometrias que já estão em 2D.

## 7.4 Tunando sua configuração

Tuning for PostGIS is much like tuning for any PostgreSQL workload. The only additional note to keep in mind is that geometries and rasters are heavy so memory related optimizations generally have more of an impact on PostGIS than other types of PostgreSQL queries.

For general details about optimizing PostgreSQL, refer to [Tuning your PostgreSQL Server](#).

For PostgreSQL 9.4+ all these can be set at the server level without touching `postgresql.conf` or `postgresql.auto.conf` by using the `ALTER SYSTEM . .` command.

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

In addition to these settings, PostGIS also has some custom settings which you can find listed in [Section 8.2](#).

### 7.4.1 Início

Estas configuração estão no arquivo `postgresql.conf`:

#### `constraint_exclusion`

- Padrão: 1MB
- Isto é geralmente utilizado para particionamento de tabelas. Se você está rodando versões anteriores do 8.4, configure para "on" garantindo que o planejador de queries irá otimizar como desejado. Depois do PostgreSQL 8.4, o padrão é `partition`, que é ideal, já que irá forçar o planejador a analisar somente as tabelas se elas estão na hierarquia de herança e não irá pagar outras penalidades com o planejador.

#### `shared_buffers`

- Default: ~128MB in PostgreSQL 9.6
- Set to about 25% to 40% of available RAM. On windows you may not be able to set as high.

#### `work_mem` (memória utilizada para operações de ordenamento e queries complexas)

- Padrão: 1MB
- Sets the maximum number of background processes that the system can support. This parameter can only be set at server start.

### 7.4.2 Runtime

#### `work_mem` (memória utilizada para operações de ordenamento e queries complexas)

- Padrão: 1MB
- Ajuste para cima para grandes bancos de dados, queries complexas e muito RAM
- Ajuste para baixo para muitos usuários concorrentes ou pouca RAM disponível
- Se você tem muita memória RAM e poucos desenvolvedores:

```
SET work_mem TO 1200000;
```

**maintenance\_work\_mem** (utilizado para VACUUM, CREATE INDEX, etc.)

- Padrão: 16MB
- Geralmente é muito pouco - amarra I/O, trava objetos enquanto swappa memória
- Recomenda-se entre 32MB até 256MB em produção com muita memória RAM, mas depende do número concorrente de usuários. Se você tem muita RAM e poucos desenvolvedores:

```
SET maintenance_work_mem TO 1200000;
```

**max\_parallel\_workers\_per\_gather** This setting is only available for PostgreSQL 9.6+ and will only affect PostGIS 2.3+, since only PostGIS 2.3+ supports parallel queries. If set to higher than 0, then some queries such as those involving relation functions like `ST_Intersects` can use multiple processes and can run more than twice as fast when doing so. If you have a lot of processors to spare, you should change the value of this to as many processors as you have. Also make sure to bump up `max_worker_processes` to at least as high as this number.

- Padrão: 1MB
  - Sets the maximum number of workers that can be started by a single `Gather` node. Parallel workers are taken from the pool of processes established by `max_worker_processes`. Note that the requested number of workers may not actually be available at run time. If this occurs, the plan will run with fewer workers than expected, which may be inefficient. Setting this value to 0, which is the default, disables parallel query execution.
-

## Chapter 8

# Referência do PostGIS

As funções descritas abaixo são as que um usuário do PostGIS devem precisar. Existem outras funções que são necessárias para suportar os objetos PostGIS mas que não são de uso comum pelo usuário.

---

**Note**

O PostGIS iniciou uma transição da convenção de nomenclatura existente para uma convenção em torno do SQL-MM. Como resultado, a maioria das funções que você conhece e ama foram renomeadas usando o padrão de tipo espacial (com o prefixo ST). As funções anteriores ainda existem, porém não são listadas nesta documentação onde as funções atualizadas são equivalentes. As funções que não possuem prefixo ST\_ não listadas nesta documentação estão obsoletas e serão removidas em futuros lançamentos, então PAREM DE UTILIZÁ-LAS.

---

### 8.1 PostgreSQL PostGIS Geometry/Geography/Box Types

#### 8.1.1 box2d

box2d — Uma caixa composta de x min, ymin, xmax, ymax. Normalmente usada para retornar a caixa 2d enclosing de uma geometria.

**Descrição**

a caixa2d é um tipo de dados espaciais usados para representar a caixa enclosing de uma geometria ou conjunto de geometrias. A ST\_Extent em versões mais antigas, anteriores a PostGIS 1.4, retornariam uma caixa2d.

#### 8.1.2 box3d

box3d — Uma caixa composta de x min, ymin, zmin, xmax, ymax, zmax. Normalmente usada para a extensão 3d de uma geometria ou coleção de geometrias.

**Descrição**

a caixa3d é um tipo de dados postgis usados para representar a caixa enclosing de uma geometria ou conjunto de geometrias. A ST\_3DExtent retorna um objeto caixa3d.

**Comportamento Casting**

Essa seção lista os casts automáticos bem como os explícitos permitidos para esse tipo de dados

---

| Cast To  | Comportamento |
|----------|---------------|
| box      | automático    |
| box2d    | automático    |
| geometry | automático    |

### 8.1.3 geometry

geometry — Tipo de dados espaciais planares

#### Descrição

geometry is a fundamental PostGIS spatial data type used to represent a feature in the Euclidean coordinate system.

All spatial operations on geometry are using units of the Spatial Reference System the geomtry is in.

#### Comportamento Casting

Essa seção lista os casts automáticos bem como os explícitos permitidos para esse tipo de dados

| Cast To   | Comportamento |
|-----------|---------------|
| box       | automático    |
| box2d     | automático    |
| box3d     | automático    |
| bytea     | automático    |
| geography | automático    |
| texto     | automático    |

#### Veja também

Section [4.1](#)

### 8.1.4 geometry\_dump

geometry\_dump — Um tipo de dados espaciais com dois campos - geom (possuindo um objeto geométrico) e path[] (uma ordem 1-d possuindo a posição da geometria com o objeto rejeitado).

#### Descrição

geometry\_dump é um tipo composto de dados, que consiste em um objeto referido pelo campo e path[] .geom, uma ordem de inteiros 1-dimesional (começando por 1 ex.: path[1] para pegar o primeiro elemento), a qual define o path de navegação com a geometria rejeitada para encontrar esse elemento. Isso é usado pela família de funções ST\_Dump\*, como tipo de saída para detonar uma geometria mais complexa dentro de suas partes constituintes e localização de partes.

#### Veja também

Section [14.6](#)

### 8.1.5 geografia

geografia — Tipo de dado espacial elipsoidal.



**Descrição**

geografia é um tipo de dado espacial usado para representar uma característica no sistema de coordenada da terra-redonda.

**Comportamento Casting**

Essa seção lista os casts automáticos bem como os explícitos permitidos para esse tipo de dados

| Cast To  | Comportamento |
|----------|---------------|
| geometry | explícito     |

**Veja também**

Section [14.4](#), Section [4.2](#)

## 8.2 Grandes Variáveis Unificadas Personalizadas do PostGIS (GUCs)

### 8.2.1 `postgis.backend`

`postgis.backend` — O backend para fazer a manutenção de uma função onde GEOS e SFCGAL sobrepõe. Opções: `geos` ou `sfcgal`. Padrão para `geos`.

**Descrição**

Essa GUC só é relevante se você compilou o PostGIS com o suporte `sfcgal`. Por padrão o backend `geos` é usado por funções onde o GEOS e o SFCGAL têm o mesmo nome. Essa variável permite exceder e fazer o `sfcgal` ser o backend para a solicitação do serviço.

Disponibilidade: 2.1.0

**Exemplos**

Configura backend apenas para vida de conexão

```
set postgis.backend = sfcgal;
```

Configura backend para novas conexões para o banco de dados

```
ALTER DATABASE mygisdb SET postgis.backend = sfcgal;
```

**Veja também.**

Section [8.10](#)

### 8.2.2 `postgis.gdal_datapath`

`postgis.gdal_datapath` — Uma opção de configuração para designar o valor da opção `GDAL_DATA` do GDAL. Se não funcionar, a variável ambiental `GDAL_DATA` é usada.

## Descrição

Uma variável GUC do PostgreSQL para configurar o valor da opção `GDAL_DATA` do GDAL. O valor `postgis.gdal_datapath` deve ser o path físico completo para os arquivos de dados do GDAL.

Essa opção de configuração é mais usada para plataformas do Windows, onde os arquivos de dados path do GDAL's não estão hard-coded. Essa opção deve também ser configurada quando esses arquivos não estiverem no path esperado.



### Note

Essa opção pode ser configurada no arquivo de configuração `postgresql.conf`. Pode ser configurado por conexão ou transação.

Disponibilidade: 2.2.0



### Note

Informação adicional sobre o `GDAL_DATA` está disponível em GDAL's [Configuration Options](#).

## Exemplos

### Configurar e resetar `postgis.gdal_datapath`

```
SET postgis.gdal_datapath TO '/usr/local/share/gdal.hidden';  
SET postgis.gdal_datapath TO default;
```

### Configurando no Windows para um banco de dados específico

```
ALTER DATABASE gisdb  
SET postgis.gdal_datapath = 'C:/Program Files/PostgreSQL/9.3/gdal-data';
```

## Veja também.

[PostGIS\\_GDAL\\_Version](#), [ST\\_Transform](#)

## 8.2.3 `postgis.gdal_enabled_drivers`

`postgis.gdal_enabled_drivers` — Uma opção de configuração para estabelecer os drivers GDAL ativados no ambiente POstGIS. Afeta a variável `GDAL_SKIP` do GDAL.

## Descrição

Uma opção de configuração para estabelecer os drivers GDAL ativados no PostGIS. Afeta a variável de configuração `GDAL_SKIP`. Essa opção pode ser estabelecida no arquivo de configuração do PostgreSQL: `postgresql.conf`. Ela também pode ser estabelecida por conexão ou transação.

O valor inicial do `postgis.gdal_enabled_drivers` também pode ser estabelecido passando a variável de ambiente `POSTGIS_GDAL_ENABLED_DRIVERS` com a lista de drivers ativados para o processo de começar o PostgreSQL.

Dispositivos ativados específicos GDAL podem ser especificados pelos dispositivos de nome ou código curto. Dispositivos com nomes ou códigos curtos podem ser encontrados em [GDAL Raster Formats](#). Vários dispositivos podem ser encontrados, colocando um espaço entre cada um deles.

**Note**

Existem três códigos especiais disponíveis para `postgis.gdal_enabled_drivers`. Os códigos são case-sensitive.



- `DISABLE_ALL` desabilita todos os drivers GDAL. Se presente, `DISABLE_ALL` excede todos os outros valores em `postgis.gdal_enabled_drivers`.
- `ENABLE_ALL` ativa todos os drivers GDAL.
- `VSI_CURL` ativa o arquivo do sistema virtual `/vsicurl/` do GDAL.

Quando `postgis.gdal_enabled_drivers` é configurado para `DESABILITAR_TODOS`, tenta usar `out-db rasters`, `ST_FromGDALRaster()`, `ST_AsGDALRaster()`, `ST_AsTIFF()`, `ST_AsJPEG()` e `ST_AsPNG()` resultará em mensagens de erro.

**Note**

Na instalação padrão do PostGIS, `postgis.gdal_enabled_drivers` é configurado para `DESABILITAR_TODOS`.

**Note**

Informações adicionais sobre `GDAL_SKIP` estão disponíveis em [Opções de Configuração](#).

Disponibilidade: 2.2.0

**Exemplos**

Configurar e resetar `postgis.gdal_enabled_drivers`

Configura backend para todas as novas conexões para o banco de dados

```
ALTER DATABASE mygisdb SET postgis.gdal_enabled_drivers TO 'GTiff PNG JPEG';
```

Estabelece drivers ativados padrões para todas as conexões para fazer a manutenção. Requer acesso super do usuário e PostgreSQL 9.4+. Aquele banco de dados, sessão e usuário não excedem isso.

```
ALTER SYSTEM SET postgis.gdal_enabled_drivers TO 'GTiff PNG JPEG';  
SELECT pg_reload_conf();
```

```
SET postgis.gdal_enabled_drivers TO 'GTiff PNG JPEG';  
SET postgis.gdal_enabled_drivers = default;
```

Ativar todos os dispositivos GDAL

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
```

Desativar todos os dispositivos GDAL

```
SET postgis.gdal_enabled_drivers = 'DISABLE_ALL';
```

**Veja também.**

[ST\\_FromGDALRaster](#), [ST\\_AsGDALRaster](#), [ST\\_AsTIFF](#), [ST\\_AsPNG](#), [ST\\_AsJPEG](#), [postgis.enable\\_outdb\\_rasters](#)

## 8.2.4 `postgis.enable_outdb_rasters`

`postgis.enable_outdb_rasters` — Uma opção de configuração booleana para ativar o acesso ao out-db raster bands.

### Descrição

Uma opção de configuração booleana para ativar o acesso ao ut-db raster bands. Essa opção pode ser estabelecida no arquivo de configuração: `postgresql.conf`. Ela também pode ser estabelecida por conexão ou transação.

O valor inicial de `postgis.enable_outdb_rasters` também pode ser estabelecido passando a variável de ambiente `POSTGIS_ENABLE_OUTDB_RASTERS` com um valor não-zero para o processo de começar o PostgreSQL.



#### Note

Mesmo se `postgis.enable_outdb_rasters` is é verdade, o GUC `postgis.enable_outdb_rasters` determina os formatos raster acessíveis.



#### Note

Na instalação padrão do PostGIS, `postgis.enable_outdb_rasters` é colocado como Falso.

Disponibilidade: 2.2.0

### Exemplos

Configurar e resetar `postgis.enable_outdb_rasters`

```
SET postgis.enable_outdb_rasters TO True;
SET postgis.enable_outdb_rasters = default;
SET postgis.enable_outdb_rasters = True;
SET postgis.enable_outdb_rasters = False;
```

Veja também.

[postgis.gdal\\_enabled\\_drivers](#)

## 8.3 Funções de Gestão

### 8.3.1 `AddGeometryColumn`

`AddGeometryColumn` — Adiciona uma coluna geometria à uma table de atributos existente. Por padrão usa um tipo modificador em vez de restrições. Passa em falso por usar `use_typmod` para obter uma restrição antiga baseada em comportamento.

#### 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);
```

## Descrição

Adiciona uma coluna geometria à uma table de atributos. O `schema_name` é o nome da table esquema. O `srid` deve ser um valor de referência inteiro para uma entrada na table `SPATIAL_REF_SYS`. O `tipo` deve ser uma string correspondente ao tipo da geometria, por exemplo: 'POLÍGONO' ou 'MULTILINSTRING'. Um erro é descartado se o esquema não existe (ou não é visível no `search_path` atual) ou a `SRID` especificada, tipo de geometria ou dimensão é inválida.

### Note



Alterado: 2.0.0 Essa função não atualiza mais a `geometry_columns` desde que ela é a view que lê dos catálogos de sistema. Por padrão, isso não cria restrições, mas usa a construção no comportamento do tipo modificador do PostgreSQL. Então, por exemplo, construir uma coluna `wgs84 POINT` com essa função é equivalente a: `ALTER TABLE some_table ADD COLUMN geom geometry(Point,4326);`

Alterado: 2.0.0 Se você exige o comportamento antigo de restrições use o padrão `use_typmod`, mas configure isso para falso.

### Note



Alterações: 2.0.0 Views não podem ser registradas manualmente mais em `geometry_columns`, porém as views construídas contra as geometrias `typmod` tables e usadas sem as funções wrapper irão se registrar corretamente, porque elas herdam um comportamento `typmod` da table column mãe. As views que usam funções geométricas que fazem outras geometrias saírem, precisarão de ser lançadas para as geometrias `typmod`, para essas colunas serem registradas corretamente em `geometry_columns`. Use Section 4.3.4.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Melhorias: 2.0.0 argumento `use_typmod` introduzido. Padrões para criar colunas de geometria `typmod` ao invés das baseadas em obstáculos.

## Exemplos

```
-- Create schema to hold data
CREATE SCHEMA my_schema;
-- Create a new simple PostgreSQL table
CREATE TABLE my_schema.my_spatial_table (id serial);

-- Describing the table shows a simple table with a single "id" column.
postgis=# \d my_schema.my_spatial_table
          Table "my_schema.my_spatial_table"
Column | Type          | Modifiers
-----+-----+-----
 id    | integer      | not null default nextval('my_schema.my_spatial_table_id_seq'::regclass)

-- Add a spatial column to the table
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geom',4326,'POINT',2);

-- Add a point using the old constraint based behavior
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geom_c',4326,'POINT',2, false);

--Add a curvepolygon using old constraint behavior
```

```

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

-- Describe the table again reveals the addition of a new geometry columns.
\d my_schema.my_spatial_table
          addgeometrycolumn
-----
my_schema.my_spatial_table.geomcp_c SRID:4326 TYPE:CURVEPOLYGON DIMS:2
(1 row)

Table "my_schema.my_spatial_table"
Column |          Type          | Modifiers
-----+-----+-----
id      | integer                | not null default nextval('my_schema.my_spatial_table_id_seq'::regclass)
geom    | geometry(Point,4326)   |
geom_c  | geometry                |
geomcp_c | geometry                |

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

-- geometry_columns view also registers the new columns --
SELECT f_geometry_column As col_name, type, srid, coord_dimension As ndims
FROM geometry_columns
WHERE f_table_name = 'my_spatial_table' AND f_table_schema = 'my_schema';

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

```

**Veja também**

[DropGeometryColumn](#), [DropGeometryTable](#), [Section 4.3.2](#), [Section 4.3.4](#)

**8.3.2 DropGeometryColumn**

DropGeometryColumn — Remove uma coluna geometria de uma spatial table.

**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);

```

**Descrição**

Remove uma coluna geometria de uma table espacial. Note que o `schema_name` precisará combinar com o campo `f_table_schema` da fila da table na table `geometry_columns`.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

**Note**

Alterações: 2.0.0 Essa função é fornecida para compatibilidade atrasada. Desde que `geometry_columns` é uma view contra os sistemas catalogados, você pode derrubar uma coluna geométrica como qualquer outra table column usando `ALTERAR TABLE`

**Exemplos**

```
SELECT DropGeometryColumn ('my_schema', 'my_spatial_table', 'geom');
      ----RESULT output ----
                        dropgeometrycolumn
-----
my_schema.my_spatial_table.geom effectively removed.

-- In PostGIS 2.0+ the above is also equivalent to the standard
-- the standard alter table. Both will deregister from geometry_columns
ALTER TABLE my_schema.my_spatial_table DROP column geom;
```

**Veja também**

[AddGeometryColumn](#), [DropGeometryTable](#), [Section 4.3.2](#)

**8.3.3 DropGeometryTable**

`DropGeometryTable` — Derruba uma table e todas suas referências em `geometry_columns`.

**Synopsis**

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

**Descrição**

Derruba uma table e todas as suas referências em `geometry_columns`. Nota: use `current_schema()` nas instalações schema-aware pgsqll se o esquema não for fornecido.

**Note**

Alterações: 2.0.0 Essa função é fornecida para compatibilidade atrasada. Desde que `geometry_columns` é uma view contra os sistemas catalogados, você pode derrubar uma table com colunas geométricas como qualquer outra table usando `DERRUBAR TABLE`

**Exemplos**

```
SELECT DropGeometryTable ('my_schema', 'my_spatial_table');
----RESULT output ---
my_schema.my_spatial_table dropped.

-- The above is now equivalent to --
DROP TABLE my_schema.my_spatial_table;
```

**Veja também**

[AddGeometryColumn](#), [DropGeometryColumn](#), [Section 4.3.2](#)

**8.3.4 PostGIS\_Extensions\_Upgrade**

**PostGIS\_Extensions\_Upgrade** — Upgrades installed postgis packaged extensions (e.g. `postgis_sfcgal`, `postgis_topology`, `postgis_sfcgal`) to latest installed version. Reports full postgis version and build configuration infos after.

**Synopsis**

```
text PostGIS_Extensions_Upgrade();
```

**Descrição**

Upgrades installed postgis packaged extensions to latest installed version. Only extensions you have installed in the database will be upgraded and if they are already at last installed version, they will not be upgraded. Reports full postgis version and build configuration infos after. This is short-hand for doing multiple `ALTER EXTENSION .. UPDATE` for each postgis extension. Currently only tries to upgrade extensions `postgis`, `postgis_sfcgal`, `postgis_topology`, and `postgis_tiger_geocoder`.

Availability: 2.5.0

**Exemplos**

```
SELECT PostGIS_Extensions_Upgrade();
```

```
NOTICE: ALTER EXTENSION postgis_tiger_geocoder UPDATE TO "2.5.0dev";
CONTEXT: PL/pgSQL function postgis_extensions_upgrade() line 10 at RAISE
NOTICE: ALTER EXTENSION postgis_topology UPDATE TO "2.5.0dev";
CONTEXT: PL/pgSQL function postgis_extensions_upgrade() line 10 at RAISE

                                postgis_extensions_upgrade
-----
POSTGIS="2.5.0dev r15966" [EXTENSION] PGSQL="100"
GEOS="3.7.0dev-CAPI-1.11.0 8fe2ce6" SFCGAL="1.3.1"
PROJ="Rel. 4.9.3, 15 August 2016" GDAL="GDAL 2.2.2, released 2017/09/15"
LIBXML="2.7.8" LIBJSON="0.12" LIBPROTOBUF="1.2.1" TOPOLOGY RASTER
(1 row)
```

**Veja também**

[Section 2.10](#), [PostGIS\\_GEOS\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_PROJ\\_Version](#), [PostGIS\\_Version](#)



### 8.3.5 PostGIS\_Full\_Version

PostGIS\_Full\_Version — Relata a versão completa do postgis e constrói informações de configuração.

#### Synopsis

```
text PostGIS_Full_Version();
```

#### Descrição

Relata a versão completa do postgis e constrói infos. Também informa sobre sincronização entre bibliotecas e scripts, sugerindo atualizações, se necessário.

#### Exemplos

```
SELECT PostGIS_Full_Version();
                                     postgis_full_version
-----
POSTGIS="2.2.0dev r12699" GEOS="3.5.0dev-CAPI-1.9.0 r3989" SFCGAL="1.0.4" PROJ="Rel. 4.8.0, ↵
 6 March 2012"
GDAL="GDAL 1.11.0, released 2014/04/16" LIBXML="2.7.8" LIBJSON="0.12" RASTER
(1 row)
```

#### Veja também

Section 2.10, [PostGIS\\_GEOS\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_PROJ\\_Version](#), [PostGIS\\_Version](#)

### 8.3.6 PostGIS\_GEOS\_Version

PostGIS\_GEOS\_Version — Retorna à versão da biblioteca GEOS.

#### Synopsis

```
text PostGIS_GEOS_Version();
```

#### Descrição

Retorna à versão da biblioteca GEOS ou NULO se o suporte GEOS não estiver ativado.

#### Exemplos

```
SELECT PostGIS_GEOS_Version();
postgis_geos_version
-----
3.1.0-CAPI-1.5.0
(1 row)
```

**Veja também**

[PostGIS\\_Full\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_PROJ\\_Version](#), [PostGIS\\_Version](#)

### 8.3.7 PostGIS\_Liblwgeom\_Version

PostGIS\_Liblwgeom\_Version — Returns the version number of the liblwgeom library. This should match the version of PostGIS.

**Synopsis**

```
text PostGIS_Liblwgeom_Version();
```

**Descrição**

Returns the version number of the liblwgeom library/

**Exemplos**

```
SELECT PostGIS_Liblwgeom_Version();
postgis_liblwgeom_version
-----
2.3.3 r15473
(1 row)
```

**Veja também**

[PostGIS\\_Full\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_PROJ\\_Version](#), [PostGIS\\_Version](#)

### 8.3.8 PostGIS\_LibXML\_Version

PostGIS\_LibXML\_Version — Retorna à versão da biblioteca libxml2.

**Synopsis**

```
text PostGIS_LibXML_Version();
```

**Descrição**

Retorna à versão da biblioteca LibXML2.

Disponibilidade: 1.5

**Exemplos**

```
SELECT PostGIS_LibXML_Version();
postgis_libxml_version
-----
2.7.6
(1 row)
```

**Veja também**

[PostGIS\\_Full\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_PROJ\\_Version](#), [PostGIS\\_GEOS\\_Version](#), [PostGIS\\_Version](#)

**8.3.9 PostGIS\_Lib\_Build\_Date**

PostGIS\_Lib\_Build\_Date — Returns build date of the PostGIS library.

**Synopsis**

```
text PostGIS_Lib_Build_Date();
```

**Descrição**

Returns build date of the PostGIS library.

**Exemplos**

```
SELECT PostGIS_Lib_Build_Date();

postgis_lib_build_date
-----
2008-06-21 17:53:21

(1 row)
```

**8.3.10 PostGIS\_Lib\_Version**

PostGIS\_Lib\_Version — Returns the version number of the PostGIS library.

**Synopsis**

```
text PostGIS_Lib_Version();
```

**Descrição**

Returns the version number of the PostGIS library.

**Exemplos**

```
SELECT PostGIS_Lib_Version();

postgis_lib_version
-----
1.3.3

(1 row)
```

**Veja também**

[PostGIS\\_Full\\_Version](#), [PostGIS\\_GEOS\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_PROJ\\_Version](#), [PostGIS\\_Version](#)

**8.3.11 PostGIS\_PROJ\_Version**

PostGIS\_PROJ\_Version — Retorna à versão da biblioteca do PROJ4.

**Synopsis**

```
text PostGIS_PROJ_Version();
```

**Descrição**

Retorna à versão da biblioteca PROJ4 ou NULO se o suporte PROJ4 não estiver ativado.

**Exemplos**

```
SELECT PostGIS_PROJ_Version();

postgis_proj_version
-----
Rel. 4.4.9, 29 Oct 2004

(1 row)
```

**Veja também**

[PostGIS\\_Full\\_Version](#), [PostGIS\\_GEOS\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_Version](#)

**8.3.12 PostGIS\_Scripts\_Build\_Date**

PostGIS\_Scripts\_Build\_Date — Returns build date of the PostGIS scripts.

**Synopsis**

```
text PostGIS_Scripts_Build_Date();
```

**Descrição**

Returns build date of the PostGIS scripts.

Disponibilidade: 1.0.0RC1

---

## Exemplos

```
SELECT PostGIS_Scripts_Build_Date();

postgis_scripts_build_date
-----
2007-08-18 09:09:26
(1 row)
```

## Veja também

[PostGIS\\_Full\\_Version](#), [PostGIS\\_GEOS\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_Version](#)

### 8.3.13 PostGIS\_Scripts\_Installed

PostGIS\_Scripts\_Installed — Retorna a versão das scripts do postgis instaladas nesse banco de dados.

## Synopsis

texto **PostGIS\_Scripts\_Installed()**;

## Descrição

Retorna a versão das scripts do postgis instaladas nesse banco de dados.



### Note

Se a saída dessa função não se encaixa com a saída da [PostGIS\\_Scripts\\_Released](#), você fracassou em atualizar propriamente um banco de dados existente. Veja a seção [Upgrading](#) para mais informação.

Disponibilidade: 0.9.0

## Exemplos

```
SELECT PostGIS_Scripts_Installed();

postgis_scripts_installed
-----
1.5.0SVN
(1 row)
```

## Veja também

[PostGIS\\_Full\\_Version](#), [PostGIS\\_Scripts\\_Released](#), [PostGIS\\_Version](#)

### 8.3.14 PostGIS\_Scripts\_Released

PostGIS\_Scripts\_Released — Retorna a versão da script do postgresql com o postgres lib instalado.

#### Synopsis

texto **PostGIS\_Scripts\_Released()**;

#### Descrição

Retona a versão da script do postgresql com a postgres lib. instalada.



#### Note

Começando com a versão 1.1.0, essa função retorna o mesmo valor da [PostGIS\\_Lib\\_Version](#). Manteve para compatibilidade atrasada.

Disponibilidade: 0.9.0

#### Exemplos

```
SELECT PostGIS_Scripts_Released();

postgres_scripts_released
-----
1.3.4SVN
(1 row)
```

#### Veja também

[PostGIS\\_Full\\_Version](#), [PostGIS\\_Scripts\\_Installed](#), [PostGIS\\_Lib\\_Version](#)

### 8.3.15 PostGIS\_Version

PostGIS\_Version — Retorna a versão do PostGIS e compila opções de tempo.

#### Synopsis

texto **PostGIS\_Version()**;

#### Descrição

Retorna a versão do PostGIS e compila opções de tempo.

## Exemplos

```
SELECT PostGIS_Version();

postgis_version
-----
1.3 USE_GEOS=1 USE_PROJ=1 USE_STATS=1

(1 row)
```

## Veja também

[PostGIS\\_Full\\_Version](#), [PostGIS\\_GEOS\\_Version](#), [PostGIS\\_Lib\\_Version](#), [PostGIS\\_LibXML\\_Version](#), [PostGIS\\_PROJ\\_Version](#)

### 8.3.16 Populate\_Geometry\_Columns

`Populate_Geometry_Columns` — Assegura que as colunas geométricas são definidas com modificadores de tipo ou têm obstáculos espaciais apropriados. Isso garante que serão registrados corretamente na view `geometry_columns`. Por padrão, irá converter todas as colunas geométricas com nenhum modificador de tipo para os que têm o modificador. para obter esse comportamento antigo use `use_typmod=false`

## Synopsis

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

## Descrição

Assegura que as colunas geométricas tenham tipos modificadores ou restrições espaciais apropriadas para certificar que elas estão registradas corretamente na table `geometry_columns`.

Para compatibilidades atrasadas e necessidades espaciais como a herança das tables, onde cada table child talvez tenha um tipo geométrico diferente, a última verificação do comportamento ainda é suportada. Se você precisar do último comportamento, você tem de passar o novo argumento opcional como falso `use_typmod=false`. Quando isso for feito, as colunas geométricas serão criadas sem modificadores de tipo, mas terão 3 obstáculos definidos. Isso significa que cada coluna geométrica pertencente a uma table tem, pelo menos, três obstáculos:

- `enforce_dims_the_geom` - assegura que toda geometria tenha a mesma dimensão (veja [ST\\_NDims](#))
- `enforce_geotype_the_geom` - assegura que toda geometria seja do mesmo tipo (veja [Tipo de geometria](#))
- `enforce_srid_the_geom` - assegura que toda geometria tenha a mesma projeção (veja [ST\\_SRID](#))

Se uma table `oid` é fornecida, essa função tenta determinar a `srid`, a dimensão e o tipo geométrico de todas as colunas geométricas na table, adicionando restrições se necessário. Se for bem-sucedido, uma fila apropriada é inserida na table `geometry_columns`, senão, a exceção é pega e uma notificação de erro surge, descrevendo o problema.

Se o `oid` de uma view é fornecido, como com uma table `oid`, essa função tenta determinar a `srid`, dimensão e tipo de todas as geometrias na view, inserindo entradas apropriadas na table `geometry_columns`, mas nada é feito para executar obstáculos.

A variante sem parâmetro é um simples wrapper para a variante parametrizada que trunca primeiro e repopula a table `geometry_columns` para cada table espacial e view no banco de dados, adicionando obstáculos espaciais para tables onde são apropriados. Isso retorna um resumo do número de colunas geométricas detectadas no banco de dados e o número que foi

inserido na table `geometry_columns`. A versão parametrizada retorna, simplesmente, o número de filas inseridas na table `geometry_columns`.

Disponibilidade: 1.4.0

Alterações: 2.0.0 Por padrão, utilize modificadores de tipo ao invés de verificar restrições para restringir os tipos de geometria. Você pode verificar restrições de comportamento ao invés de usar o novo `use_typmod` e configurá-lo para falso.

Melhorias: 2.0.0 `use_typmod` argumento opcional foi introduzido, permitindo controlar se as colunas forem criadas com modificadores de tipo ou com verificação de restrições.

## Exemplos

```
CREATE TABLE public.myspatial_table(gid serial, geom geometry);
INSERT INTO myspatial_table(geom) VALUES(ST_GeomFromText('LINESTRING(1 2, 3 4)',4326) );
-- This will now use typ modifiers. For this to work, there must exist data
SELECT Populate_Geometry_Columns('public.myspatial_table'::regclass);
```

```
populate_geometry_columns
-----
                1

\d myspatial_table

Table "public.myspatial_table"
Column |          Type          | Modifiers
-----+-----+-----
gid    | integer                | not null default nextval('myspatial_table_gid_seq'::<
  regclass)
geom   | geometry(LineString,4326) |
```

```
-- This will change the geometry columns to use constraints if they are not typmod or have <
  constraints already.
--For this to work, there must exist data
CREATE TABLE public.myspatial_table_cs(gid serial, geom geometry);
INSERT INTO myspatial_table_cs(geom) VALUES(ST_GeomFromText('LINESTRING(1 2, 3 4)',4326) );
SELECT Populate_Geometry_Columns('public.myspatial_table_cs'::regclass, false);
populate_geometry_columns
-----
                1

\d myspatial_table_cs

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

### 8.3.17 UpdateGeometrySRID

`UpdateGeometrySRID` — Atualiza a SRID de todas as características em uma coluna geométrica, `geometry_columns` metadados e `srid`. Se foi executado com restrições, elas serão atualizadas com a nova restrição `srid`. Se a antiga foi executada pelo definição de tipo, ela será alterada.



## 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);
```

## Descrição

Atualiza a SRID de todas as características em uma coluna geométrica, atualizando restrições e referências na `geometry_columns`.  
Nota: use `current_schema()` nas instalações `schema-aware` `pgsql` se o esquema não for fornecido.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

Isso irá alterar a srid das roads tables para 4326 de qualquer coisa que tenha sido antes

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

O exemplo anterior é equivalente a esta declaração DDL

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

Se você obteve a projeção errada (ou comprou como desconhecido) no carregamento e quer transformar para mercator, tudo de uma vez, você pode fazer isso com DDL, mas não existe uma função de gestão equivalente do PostGIS.

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

## Veja também

[UpdateRasterSRID](#), [ST\\_SetSRID](#), [ST\\_Transform](#)

## 8.4 Construtores de geometria

### 8.4.1 ST\_BdPolyFromText

`ST_BdPolyFromText` — Constrói um polígono dada uma coleção arbitrária de linestrings fechadas como uma representação de texto de uma multilinestring bem conhecida.

## Synopsis

```
geometria ST_BdPolyFromText(texto WKT, inteiro srid);
```

## Descrição

Constrói um polígono dada uma coleção arbitrária de linestrings fechadas como uma representação de texto de uma multilinestring bem conhecida.



### Note

Lança um erro se WKT não é uma MULTILINESTRING. Lança um erro se a saída não é um MULTIPOLÍGONO; use `ST_BdMPolyFromText` nesse caso, ou veja `ST_BuildArea()` para uma aproximação postgis-specific.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2

Disponibilidade: 1.1.0 - requer GEOS >= 2.1.0.

## Examples

Próximo

## Veja também

[ST\\_BuildArea](#), [ST\\_BdMPolyFromText](#)

## 8.4.2 ST\_BdMPolyFromText

`ST_BdMPolyFromText` — Constrói um polígono dada uma coleção arbitrária de linestrings fechadas como uma representação de texto de uma multilinestring bem conhecida.

## Synopsis

geometria `ST_BdMPolyFromText`(text WKT, inteiro srid);

## Descrição

Constrói um um polígono dada uma coleção arbitrária de linestrings, polígonos, multilinestrings fechados como uma representação de texto bem conhecida.



### Note

Lança um erro se WKT não é uma MULTILINESTRING. Força a saída MULTIPOLÍGONO mesmo quando o resultado não é composto somente por um POLÍGONO único; use `ST_BdPolyFromText` se você tem certeza de que um único POLÍGONO irá resultar de uma operação, ou veja `ST_BuildArea()` para uma aproximação postgis-specific.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2

Disponibilidade: 1.1.0 - requer GEOS >= 2.1.0.

## Examples

Próximo

**Veja também**

[ST\\_BuildArea](#), [ST\\_BdPolyFromText](#)

**8.4.3 ST\_Box2dFromGeoHash**

`ST_Box2dFromGeoHash` — Retorna uma CAIXA2D de uma string GeoHash.

**Synopsis**

```
box2d ST_Box2dFromGeoHash(text geohash, integer precision=full_precision_of_geohash);
```

**Descrição**

Retorna uma CAIXA2D de uma string GeoHash.

Se nenhuma `precisão` é especificada, a `ST_Box2dFromGeoHash` retorna uma CAIXA2D baseada em uma precisão completa da string de entrada GeoHash.

Se a `precisão` é especificada, a `ST_Box2dFromGeoHash` irá usar aqueles vários caracteres do GeoHash para criar a CAIXA2D. Valores de precisão mais baixos resultam em CAIXAS2D maiores e valores maiores aumentam a precisão.

Disponibilidade: 2.1.0

**Examples**

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxnccgyy4d0dbxqz0');
           st_geomfromgeohash
-----
BOX(-115.172816 36.114646,-115.172816 36.114646)

SELECT ST_Box2dFromGeoHash('9qqj7nmxnccgyy4d0dbxqz0', 0);
           st_box2dfromgeohash
-----
BOX(-180 -90,180 90)

SELECT ST_Box2dFromGeoHash('9qqj7nmxnccgyy4d0dbxqz0', 10);
           st_box2dfromgeohash
-----
BOX(-115.17282128334 36.1146408319473,-115.172810554504 36.1146461963654)
```

**Veja também**

[ST\\_GeoHash](#), [ST\\_GeomFromGeoHash](#), [ST\\_PointFromGeoHash](#)

**8.4.4 ST\_GeogFromText**

`ST_GeogFromText` — Retorna um valor de geografia específico de uma representação bem conhecida de texto ou estendida (WKT).

## Synopsis

geografia **ST\_GeogFromText**(texto EWKT);

## Descrição

Retorna um objeto de geografia de um texto bem conhecido ou representação estendida bem conhecida. SRID 4326 é suposta se não for especificada. Isso é um heterônimo para `ST_GeographyFromText`. Os pontos são sempre expressados em uma forma long lat.

## Examples

```
--- converting lon lat coords to geography
ALTER TABLE sometable ADD COLUMN geog geography(POINT,4326);
UPDATE sometable SET geog = ST_GeogFromText('SRID=4326;POINT(' || lon || ' ' || lat || ')') ←
;

--- specify a geography point using EPSG:4267, NAD27
SELECT ST_AsEWKT(ST_GeogFromText('SRID=4267;POINT(-77.0092 38.889588)'));
```

## Veja também

[ST\\_AsText](#), [ST\\_GeographyFromText](#)

### 8.4.5 ST\_GeographyFromText

`ST_GeographyFromText` — Retorna um valor de geografia específico de uma representação bem conhecida de texto ou estendida (WKT).

## Synopsis

geografia **ST\_GeographyFromText**(texto EWKT);

## Descrição

Retorna um objeto de geografia de um texto bem conhecido ou representação estendida bem conhecida. SRID 4326 é suposta se não for especificada

## Veja também

[ST\\_GeogFromText](#), [ST\\_AsText](#)

### 8.4.6 ST\_GeogFromWKB

`ST_GeogFromWKB` — Cria uma ocasião geografia de uma geometria binária bem conhecida (WKB) ou binário estendido bem conhecido (EWKB).

## Synopsis

geografia **ST\_GeogFromWKB**(bytea wkb);

---

## Descrição

A função `ST_GeogFromWKB`, pega uma representação binária bem conhecida (WKB) de uma geometria ou WKB estendida do PostGIS e cria uma ocasião do tipo de geografia apropriado. Essa função cumpre o papel da Fábrica de Geometria em SQL.

Se a SRID não está especificado, isso leva a 4326 (WGS 84 long lat).



This method supports Circular Strings and Curves

## Examples

```
--Although bytea rep contains single \, these need to be escaped when inserting into a table
SELECT ST_AsText(
ST_GeogFromWKB(E'\001\002\000\000\000\002\000\000\000\037\205\353Q
  \270~\300\323Mb\020X\231C@020X9\264\310~\300)\217\302\365\230
  C@')
);

```

| st_astext                                  |
|--------------------------------------------|
| LINESTRING(-113.98 39.198,-113.981 39.195) |

(1 row)

## Veja também

[ST\\_GeogFromText](#), [ST\\_AsBinary](#)

## 8.4.7 ST\_GeomFromTWKB

`ST_GeomFromTWKB` — Cria uma ocasião de uma TWKB ("Tiny Well-Known Binary") representação de geometria.

## Synopsis

geometria `ST_GeomFromTWKB`(bytea twkb);

## Descrição

A função `ST_GeomFromTWKB`, pega uma TWKB ("Tiny Well-Known Binary") representação geométrica (WKB) e cria uma ocasião do tipo apropriado de geometria.

## Examples

```
SELECT ST_AsText(ST_GeomFromTWKB(ST_ASTWKB('LINESTRING(126 34, 127 35)::geometry')));

```

| st_astext                  |
|----------------------------|
| LINESTRING(126 34, 127 35) |

(1 row)

```
SELECT ST_AsEWKT(
  ST_GeomFromTWKB(E'\x620002f7f40dbce4040105')
);
```

```
----- st_asewkt
-----
```

```
LINestring(-113.98 39.198,-113.981 39.195)
(1 row)
```

### Veja também

[ST\\_AsTWKB](#)

## 8.4.8 ST\_GeomCollFromText

**ST\_GeomCollFromText** — Makes a collection Geometry from collection WKT with the given SRID. If SRID is not given, it defaults to 0.

### Synopsis

```
geometria ST_GeomCollFromText(text WKT, inteiro srid);
geometria ST_GeomCollFromText(texto WKT);
```

### Descrição

Makes a collection Geometry from the Well-Known-Text (WKT) representation with the given SRID. If SRID is not given, it defaults to 0.

OGC SPEC 3.2.6.2 - opção SRID é de uma suíte de conformação

Retorna nula se a WKT não for uma GEOMETRYCOLLECTION



#### Note

se você não tem total certeza de que todas suas geometrias WKT são coleções, não use essa função. Ela é mais devagar que a `ST_GeomFromText`, já que adiciona um passo de validação adicional.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification.

### Examples

```
SELECT ST_GeomCollFromText('GEOMETRYCOLLECTION(POINT(1 2),LINestring(1 2, 3 4))');
```

### Veja também

[ST\\_GeomFromText](#), [ST\\_SRID](#)

## 8.4.9 ST\_GeomFromEWKB

**ST\_GeomFromEWKB** — Retorna um valor `ST_Geometry` específico da representação binária estendida bem conhecida (EWKB).

## Synopsis

geometria **ST\_GeomFromEWKB**(bytea EWKB);

## Descrição

Constrói um objeto PostGIS ST\_Geometry da representação binária estendida bem conhecida OGC (EWKT).



### Note

O formato EWKB não é um padrão OGC, mas um formato específico PostGIS que inclui o identificador de sistema de referência espacial (SRID).

Melhorias: 2.0.0 suporte para superfícies poliédricas e TIN foi introduzido.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Examples

line string binary rep of **LINestring**(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.25932) in NAD 83 long lat (4269).



### Note

NOTA: Mesmo que os byte arrays seja delimitados com \ e talvez tenham ', precisamos escapar ambos com \ e " se as `standard_conforming_strings` estão deligadas. então, isso não parece exatamente como uma representação `AsEWKB`.

```
SELECT ST_GeomFromEWKB(E'\001\002\000\000 \255\020\000\000\003\000\000\000\344 ←
J=
\013B\312Q\300n\303(\010\036!E@'\277E'K
\312Q\300\366{b\235*!E@\225|\354.P\312Q
\300p\231\323e1!E@');
```



### Note

In PostgreSQL 9.1+ - `standard_conforming_strings` is set to on by default, where as in past versions it was set to off. You can change defaults as needed for a single query or at the database or server level. Below is how you would do it with `standard_conforming_strings = on`. In this case we escape the ' with standard ansi ', but slashes are not escaped

```
set standard_conforming_strings = on;
SELECT ST_GeomFromEWKB('\001\002\000\000 \255\020\000\000\003\000\000\000\344J=\012\013B
\312Q\300n\303(\010\036!E@'\277E'K\012\312Q\300\366{b\235*!E@\225|\354.P\312Q\012\300 ←
p\231\323e1')
```

**Veja também**

[ST\\_AsBinary](#), [ST\\_AsEWKB](#), [ST\\_GeomFromWKB](#)

**8.4.10 ST\_GeomFromEWKT**

ST\_GeomFromEWKT — Retorna um valor ST\_Geometry específico da representação de texto estendida bem conhecida (EWKT).

**Synopsis**

geometria **ST\_GeomFromEWKT**(texto EWKT);

**Descrição**

Constrói um objeto PostGIS ST\_Geometry da representação de texto estendida bem conhecida OGC (EWKT).

**Note**

O formato EWKT não é um padrão OGC, mas um formato específico PostGIS que inclui o identificador de sistema de referência espacial (SRID).

Melhorias: 2.0.0 suporte para superfícies poliédricas e TIN foi introduzido.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

**Examples**

```
SELECT ST_GeomFromEWKT('SRID=4269;LINESTRING(-71.160281 42.258729,-71.160837 ↵
  42.259113,-71.161144 42.25932)');
SELECT ST_GeomFromEWKT('SRID=4269;MULTILINESTRING((-71.160281 42.258729,-71.160837 ↵
  42.259113,-71.161144 42.25932)');

SELECT ST_GeomFromEWKT('SRID=4269;POINT(-71.064544 42.28787)');

SELECT ST_GeomFromEWKT('SRID=4269;POLYGON((-71.1776585052917 ↵
  42.3902909739571,-71.1776820268866 42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↵
  42.3902909739571))');

SELECT ST_GeomFromEWKT('SRID=4269;MULTIPOLYGON((( -71.1031880899493 42.3152774590236,
-71.1031627617667 42.3152960829043,-71.102923838298 42.3149156848307,
-71.1023097974109 42.3151969047397,-71.1019285062273 42.3147384934248,
-71.102505233663 42.3144722937587,-71.10277487471 42.3141658254797,
-71.103113945163 42.3142739188902,-71.10324876416 42.31402489987,
-71.1033002961013 42.3140393340215,-71.1033488797549 42.3139495090772,
-71.103396240451 42.3138632439557,-71.1041521907712 42.3141153348029,
```



```
-71.1041411411543 42.3141545014533,-71.1041287795912 42.3142114839058,
-71.1041188134329 42.3142693656241,-71.1041112482575 42.3143272556118,
-71.1041072845732 42.3143851580048,-71.1041057218871 42.3144430686681,
-71.1041065602059 42.3145009876017,-71.1041097995362 42.3145589148055,
-71.1041166403905 42.3146168544148,-71.1041258822717 42.3146748022936,
-71.1041375307579 42.3147318674446,-71.1041492906949 42.3147711126569,
-71.1041598612795 42.3148085717339,-71.1042515013869 42.3151287620809,
-71.1041173835118 42.3150739481917,-71.1040809891419 42.3151344119048,
-71.1040438678912 42.3151191367447,-71.1040194562988 42.3151832057859,
-71.1038734225584 42.3151140942995,-71.1038446938243 42.3151006300338,
-71.1038315271889 42.315094347535,-71.1037393329282 42.315054824985,
-71.1035447555574 42.3152608696313,-71.1033436658644 42.3151648370544,
-71.1032580383161 42.3152269126061,-71.103223066939 42.3152517403219,
-71.1031880899493 42.3152774590236)),
((-71.1043632495873 42.315113108546,-71.1043583974082 42.3151211109857,
-71.1043443253471 42.3150676015829,-71.1043850704575 42.3150793250568,-71.1043632495873 ↔
 42.315113108546)))');
```

```
--3d circular string
SELECT ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)');
```

```
--Polyhedral Surface example
SELECT ST_GeomFromEWKT('POLYHEDRALSURFACE (
  ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
  ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
  ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
  ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
  ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
  ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1))
)');
```

## Veja também

[ST\\_AsEWKT](#), [ST\\_GeomFromText](#), [ST\\_GeomFromEWKT](#)

### 8.4.11 ST\_GeometryFromText

**ST\_GeometryFromText** — Retorna um valor **ST\_Geometry** específico da representação de texto estendida bem conhecida (EWKT). Isso é um heterônimo para **ST\_GeomFromText**

#### Synopsis

```
geometria ST_GeometryFromText(texto WKT);
geometria ST_GeometryFromText(texto WKT, inteiro srid);
```

#### Descrição



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 5.1.40

## Veja também

[ST\\_GeomFromText](#)

### 8.4.12 ST\_GeomFromGeoHash

ST\_GeomFromGeoHash — Retorna uma geometria de uma string GeoHash.

#### Synopsis

geometria **ST\_GeomFromGeoHash**(texto geohash, inteiro precision=full\_precision\_of\_geohash);

#### Descrição

Retorna uma geometria de uma string GeoHash. A geometria será um polígono representando os limites GeoHash.

Se nenhuma *precisão* for especificada, a ST\_GeomFromGeoHash retorna um polígono baseado na precisão completa da string de entrada GeoHash.

Se a *precisão* for especificada, a ST\_GeomFromGeoHash irá usar aqueles vários caracteres do GeoHash para criar o polígono.

Disponibilidade: 2.1.0

#### Examples

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncggy4d0dbxqz0'));
           st_astext
-----
POLYGON((-115.172816 36.114646,-115.172816 36.114646,-115.172816 36.114646,-115.172816 36.114646,-115.172816 36.114646))

SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncggy4d0dbxqz0', 4));
           st_astext
-----
POLYGON((-115.3125 36.03515625,-115.3125 36.2109375,-114.9609375 36.2109375,-114.9609375 36.03515625,-115.3125 36.03515625))

SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncggy4d0dbxqz0', 10));
           st_astext
-----
POLYGON((-115.17282128334 36.1146408319473,-115.17282128334 36.1146461963654,-115.172810554504 36.1146461963654,-115.172810554504 36.1146408319473,-115.17282128334 36.1146408319473))
```

#### Veja também

[ST\\_GeoHash](#), [ST\\_Box2dFromGeoHash](#), [ST\\_PointFromGeoHash](#)

### 8.4.13 ST\_GeomFromGML

ST\_GeomFromGML — Utiliza como entrada uma representação GML de geometria e como saída um objeto de geometria PostGIS

## Synopsis

geometria **ST\_GeomFromGML**(text geomgml);  
 geometria **ST\_GeomFromGML**(texto geomgml, inteiro srid);

## Descrição

Constrói um objeto PostGIS ST\_Geometry de uma representação OGC GML.

A ST\_GeomFromGML funciona apenas para fragmentos da geometria GML. Ela descarta um erro, se você tentar usá-la em um documento inteiro GML.

OGC GML versions supported:

- GML 3.2.1 Namespace
- GML 3.1.1 Simple Features profile SF-2 (with GML 3.1.0 and 3.0.0 backward compatibility)
- GML 2.1.2

OGC GML standards, cf: <http://www.opengeospatial.org/standards/gml>:

Disponibilidade: 1.5, requer libxml2 1.6+

Melhorias: 2.0.0 suporte para superfícies poliédricas e TIN foi introduzido.

Melhorias: 2.0.0 parâmetro opcional padrão srid adicionado.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

GML permite dimensões mescladas (2D e 3D dentro da mesma MultiGeometria, por exemplo). ST\_GeomFromGML converte a geometria inteira para 2D se uma dimensão Z perdida for encontrada uma vez, as geometrias do PostGIS não fazem isso.

A GML suporta SRS misturadas dentro da mesma MultiGeometria. As ST\_GeomFromGML, reprojeta todas as sub geometrias para o nó raiz da SRS, as geometrias PostGIS não fazem isso. Se nenhum srsNome atribuí disponível para o nó raiz GML, a função descarta um erro.

A função ST\_GeomFromGML não é afetada sobre um espaço de nome específico GML. Você poderia evitar mencionar ela explicitamente para usos comuns. Mas você precisa dela se quiser usar XLink dentro de GML.



### Note

A função ST\_GeomFromGML não suporta geometrias SQL/MM curvas.

## Exemplos - Uma única geometria com srsNome

```
SELECT ST_GeomFromGML ('
    <gml:LineString srsName="EPSG:4269">
      <gml:coordinates>
        -71.16028,42.258729 -71.160837,42.259112 ↔
        -71.161143,42.25932
      </gml:coordinates>
    </gml:LineString
>');
```

**Exemplos - Uso XLink**

```

SELECT ST_GeomFromGML('
    <gml:LineString xmlns:gml="http://www.opengis.net/gml"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        srsName="urn:ogc:def:crs:EPSG::4269">
        <gml:pointProperty>
            <gml:Point gml:id="p1"
                ><gml:pos
                >42.258729 -71.16028</gml:pos
            </gml:Point>
            </gml:pointProperty>
            <gml:pos
            >42.259112 -71.160837</gml:pos>
            <gml:pointProperty>
                <gml:Point xlink:type="simple" xlink:href="#p1"/>
            </gml:pointProperty>
        </gml:LineString
    >'););

```

**Exemplos - Superfícies Poliédricas**

```

SELECT ST_AsEWKT(ST_GeomFromGML('
<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:polygons>
</gml:PolyhedralSurface
>');

-- result --
POLYHEDRALSURFACE(((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)))

```

### Veja também

Section [2.4.1](#), [ST\\_AsGML](#), [ST\\_GMLToSQL](#)

## 8.4.14 ST\_GeomFromGeoJSON

**ST\_GeomFromGeoJSON** — Utiliza como entrada uma representação geojson de uma geometria e como saída um objeto de geometria PostGIS

### Synopsis

```

geometry ST_GeomFromGeoJSON(text geomjson);
geometry ST_GeomFromGeoJSON(json geomjson);
geometry ST_GeomFromGeoJSON(jsonb geomjson);

```

### Descrição

Constrói um objeto de geometria PostGIS de uma representação GeoJSON.

A **ST\_GeomFromGeoJSON** funciona apenas para fragmentos da geometria JSON. Ela descarta um erro se você tentar usá-la em um documento JSON inteiro.

Enhanced: 2.5.0 can now accept json and jsonb as inputs.

Disponibilidade: 2.0.0 requer - JSON-C >= 0.9



#### Note

Se você não tem JSON-C ativada, o suporte apresentará uma notificação de erro ao invés de uma saída. Para ativar JSON-C, execute a configuração `--with-jsondir=/path/to/json-c`. Veja mais detalhes em: Section [2.4.1](#).



This function supports 3d and will not drop the z-index.

## Examples

```
SELECT ST_AsText(ST_GeomFromGeoJSON('{"type":"Point","coordinates":[-48.23456,20.12345]}')) ←
  As wkt;
wkt
-----
POINT(-48.23456 20.12345)
```

```
-- a 3D linestring
SELECT ST_AsText(ST_GeomFromGeoJSON('{"type":"LineString","coordinates ←
  ":[ [1,2,3], [4,5,6], [7,8,9]]}')) As wkt;

wkt
-----
LINESTRING(1 2,4 5,7 8)
```

## Veja também

[ST\\_AsText](#), [ST\\_AsGeoJSON](#), [Section 2.4.1](#)

### 8.4.15 ST\_GeomFromKML

**ST\_GeomFromKML** — Utiliza como entrada uma representação KML de geometria e como saída um objeto de geometria PostGIS

#### Synopsis

geometria **ST\_GeomFromKML**(texto geomkml);

#### Descrição

Constrói um objeto PostGIS **ST\_Geometry** de uma representação OGC KML.

A **ST\_GeomFromKML** funciona apenas para fragmentos da geometria KML. Ela descarta um erro, se você tentar usá-la em um documento inteiro KML.

OGC KML versões suportadas:

- KML 2.2.0 Namespace

OGC KML standards, cf: <http://www.opengeospatial.org/standards/kml>:

Disponibilidade: 1.5, libxml2 2.6+



This function supports 3d and will not drop the z-index.



#### Note

A função **ST\_GeomFromKML** não suporta geometrias SQL/MM curvas.

### Exemplos - Uma única geometria com srsNome

```
SELECT ST_GeomFromKML('
    <LineString>
      <coordinates>
>-71.1663,42.2614
                                -71.1667,42.2616</coordinates>
      </LineString>
>');
```

### Veja também

Section [2.4.1](#), [ST\\_AsKML](#)

## 8.4.16 ST\_GMLToSQL

**ST\_GMLToSQL** — Retorna um valor **ST\_Geometry** específico da representação GML. Esse é um heterônimo para **ST\_GeomFromGML**.

### Synopsis

geometry **ST\_GMLToSQL**(text geomgml);  
geometry **ST\_GMLToSQL**(text geomgml, integer srid);

### Descrição



This method implements the SQL/MM specification. SQL-MM 3: 5.1.50 (exceto para curvas suporte).

Disponibilidade: 1.5, requer libxml2 1.6+

Melhorias: 2.0.0 suporte para superfícies poliédricas e TIN foi introduzido.

Melhorias: 2.0.0 parâmetro opcional padrão srid adicionado.

### Veja também

Section [2.4.1](#), [ST\\_GeomFromGML](#), [ST\\_AsGML](#)

## 8.4.17 ST\_GeomFromText

**ST\_GeomFromText** — Retorna um valor **ST\_Geometry** específico da representação de texto bem conhecida (WKT).

### Synopsis

geometria **ST\_GeomFromText**(texto WKT);  
geometria **ST\_GeomFromText**(texto WKT, inteiro srid);

## Descrição

Constrói um objeto PostGIS `ST_Geometry` de uma representação de texto bem conhecida OGC.



### Note

Existem duas variantes da função `ST_GeomFromText`. A primeira não pega nenhuma SRID e retorna uma geometria com um sistema de referência espacial indefinido (SRID=0). A segunda pega uma SRID como o segundo argumento e retorna uma geometria que inclui essa SRID como parte dos seus metadados.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2 - opção SRID é da suíte de conformidade.



This method implements the SQL/MM specification. SQL-MM 3: 5.1.40



This method supports Circular Strings and Curves



### Warning

Alterações: 2.0.0 Nas primeiras versões do PostGIS, `ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)')` foi permitida. Ela agora é ilegal no PostGIS 2.0.0 para melhor se adequar aos padrões SQL/MM. Ela deverá se escrita como `ST_GeomFromText('GEOMETRYCOLLECTION EMPTY')`

## Examples

```
SELECT ST_GeomFromText ('LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.25932) ');
SELECT ST_GeomFromText ('LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.25932) ', 4269);

SELECT ST_GeomFromText ('MULTILINESTRING((-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.25932) ');

SELECT ST_GeomFromText ('POINT(-71.064544 42.28787) ');

SELECT ST_GeomFromText ('POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 42.3902909739571) ');

SELECT ST_GeomFromText ('MULTIPOLYGON((( -71.1031880899493 42.3152774590236,
-71.1031627617667 42.3152960829043,-71.102923838298 42.3149156848307,
-71.1023097974109 42.3151969047397,-71.1019285062273 42.3147384934248,
-71.102505233663 42.3144722937587,-71.10277487471 42.3141658254797,
-71.103113945163 42.3142739188902,-71.10324876416 42.31402489987,
-71.1033002961013 42.3140393340215,-71.1033488797549 42.3139495090772,
-71.103396240451 42.3138632439557,-71.1041521907712 42.3141153348029,
-71.1041411411543 42.3141545014533,-71.1041287795912 42.3142114839058,
-71.1041188134329 42.3142693656241,-71.1041112482575 42.3143272556118,
-71.1041072845732 42.3143851580048,-71.1041057218871 42.3144430686681,
-71.1041065602059 42.3145009876017,-71.1041097995362 42.3145589148055,
-71.1041166403905 42.3146168544148,-71.1041258822717 42.3146748022936,
-71.1041375307579 42.3147318674446,-71.1041492906949 42.3147711126569,
-71.1041598612795 42.314808571739,-71.1042515013869 42.3151287620809,
-71.1041173835118 42.3150739481917,-71.1040809891419 42.3151344119048,
-71.1040438678912 42.3151191367447,-71.1040194562988 42.3151832057859,
```



```

-71.1038734225584 42.3151140942995,-71.1038446938243 42.3151006300338,
-71.1038315271889 42.315094347535,-71.1037393329282 42.315054824985,
-71.1035447555574 42.3152608696313,-71.1033436658644 42.3151648370544,
-71.1032580383161 42.3152269126061,-71.103223066939 42.3152517403219,
-71.1031880899493 42.3152774590236)),
((-71.1043632495873 42.315113108546,-71.1043583974082 42.3151211109857,
-71.1043443253471 42.3150676015829,-71.1043850704575 42.3150793250568,-71.1043632495873 ↔
 42.315113108546)))',4326);

SELECT ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)');

```

## Veja também

[ST\\_GeomFromEWKT](#), [ST\\_GeomFromWKB](#), [ST\\_SRID](#)

### 8.4.18 ST\_GeomFromWKB

**ST\_GeomFromWKB** — Criar uma geometria exemplo de um representação bem conhecida de geometria binária (WKB) e SRID opcional.

#### Synopsis

geometria **ST\_GeomFromWKB**(bytea geom);  
 geometry **ST\_GeomFromWKB**(bytea geom, inteiro srid);

#### Descrição

A função **ST\_GeomFromWKB**, pega uma representação binária bem conhecida de uma geometria e um sistema de referência espacial ID (SRID) e cria um exemplo do tipo apropriado de geometria. Essa função cumpre o papel da Fábrica de Geometria na SQL. Isso é um nome alternativo para **ST\_WKBToSQL**.

Se o SRID não for especificado, leva a 0 (desconhecido).



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.7.2 - o SRID opcional é da suíte de conformidade.



This method implements the SQL/MM specification. SQL-MM 3: 5.1.41



This method supports Circular Strings and Curves

#### Examples

```

--Although bytea rep contains single \, these need to be escaped when inserting into a ↔
table
        -- unless standard_conforming_strings is set to on.
SELECT ST_AsEWKT(
ST_GeomFromWKB(E'\001\002\000\000\000\002\000\000\000\0037\205\353Q ↔
  \270~\300\323Mb\020X\231C@020X9\264\310~\300)\217\302\365\230 ↔
  C@',4326)
);
          st_asewkt
-----
SRID=4326;LINESTRING(-113.98 39.198,-113.981 39.195)

```

```
(1 row)

SELECT
  ST_AsText (
    ST_GeomFromWKB (
      ST_AsEWKB ('POINT(2 5)')::geometry
    )
  );
 st_astext
-----
POINT(2 5)
(1 row)
```

**Veja também**

[ST\\_WKBToSQL](#), [ST\\_AsBinary](#), [ST\\_GeomFromEWKB](#)

**8.4.19 ST\_LineFromEncodedPolyline**

`ST_LineFromEncodedPolyline` — Cria uma `LineString` de uma Encoded Polyline.

**Synopsis**

geometria `ST_LineFromEncodedPolyline`(texto polyline, inteiro precision=5);

**Descrição**

Cria uma `LineString` de uma string Encoded Polyline.

Optional `precision` specifies how many decimal places will be preserved in Encoded Polyline. Value should be the same on encoding and decoding, or coordinates will be incorrect.

Veja <http://developers.google.com/maps/documentation/utilities/polylinealgorithm>

Disponibilidade: 2.2.0

**Examples**

```
-- Create a line string from a polyline
SELECT ST_AsEWKT(ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@'));
-- result --
SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)

-- Select different precision that was used for polyline encoding
SELECT ST_AsEWKT(ST_LineFromEncodedPolyline('_p~iF~ps|U_ulLnnqC_mqNvxq`@',6));
-- result --
SRID=4326;LINESTRING(-12.02 3.85,-12.095 4.07,-12.6453 4.3252)
```

**Veja também**

[ST\\_AsEncodedPolyline](#)

## 8.4.20 ST\_LineFromMultiPoint

ST\_LineFromMultiPoint — Cria uma linestring de um multiponto geométrico.

### Synopsis

geometria **ST\_LineFromMultiPoint**(geometria ummultiponto);

### Descrição

Cria uma LineString de uma geometria MultiPointo.



This function supports 3d and will not drop the z-index.

### Examples

```
--Create a 3d line string from a 3d multipoint
SELECT ST_AsEWKT(ST_LineFromMultiPoint(ST_GeomFromEWKT('MULTIPOINT(1 2 3, 4 5 6, 7 8 9)'))) ←
;
--result--
LINESTRING(1 2 3,4 5 6,7 8 9)
```

### Veja também

[ST\\_AsEWKT](#), [ST\\_Collect](#), [ST\\_MakeLine](#)

## 8.4.21 ST\_LineFromText

ST\_LineFromText — Faz uma geometria de uma representação WKT com a SRID dada. Se a SRID não for dada, isso leva a 0.

### Synopsis

geometria **ST\_LineFromText**(texto WKT);  
geometria **ST\_LineFromText**(texto WKT, inteiro srid);

### Descrição

Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0. If WKT passed in is not a LINESTRING, then null is returned.



#### Note

OGC SPEC 3.2.6.2 - opção SRID é de uma suíte de conformação.



#### Note

Se você sabe que todas as suas geometrias são LINESTRINGS, é mais eficiente usar somente ST\_GeomFromText. Isso só convida a ST\_GeomFromText e adiciona validação extra que ela retorna uma linestring.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification. SQL-MM 3: 7.2.8

### Examples

```
SELECT ST_LineFromText('LINESTRING(1 2, 3 4)') AS aline, ST_LineFromText('POINT(1 2)') AS ←
    null_return;

aline | null_return
-----
010200000002000000000000000000000F ... | t
```

### Veja também

[ST\\_GeomFromText](#)

## 8.4.22 ST\_LineFromWKB

`ST_LineFromWKB` — Faz uma `LINESTRING` de uma WKB com o SRID dado

### Synopsis

geometria `ST_LineFromWKB`(bytea WKB);  
geometria `ST_LineFromWKB`(bytea WKB, inteiro srid);

### Descrição

A função `ST_LineFromWKB`, pega uma representação binária bem conhecida de geometria e um sistema de referência espacial ID (SRID) e cria um exemplo do tipo apropriado de geometria - nesse caso, uma geometria `LINESTRING`. Essa função cumpre o papel da Fábrica de Geometria SQL.

Se um SRID não estiver especificado, isso leva a 0. Retorna NULA se a entrada `bytea` não representa uma `LINESTRING`.



#### Note

OGC SPEC 3.2.6.2 - opção SRID é de uma suíte de conformação.



#### Note

Se você sabe que todas suas geometrias são `LINESTRINGs`, é mais eficaz usar `ST_GeomFromWKB`. Essa função convoca `ST_GeomFromWKB` e adiciona validação extra que ela retorna uma `linestring`.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification. SQL-MM 3: 7.2.9

## Examples

```
SELECT ST_LineFromWKB(ST_AsBinary(ST_GeomFromText('LINESTRING(1 2, 3 4)'))) AS aline,
       ST_LineFromWKB(ST_AsBinary(ST_GeomFromText('POINT(1 2)'))) IS NULL AS ←
       null_return;
-----
aline | null_return
-----
01020000000200000000000000000000F ... | t
```

## Veja também

[ST\\_GeomFromWKB](#), [ST\\_LinestringFromWKB](#)

### 8.4.23 ST\_LinestringFromWKB

`ST_LinestringFromWKB` — Faz uma geometria de uma WKB com o SRID dado.

#### Synopsis

```
geometry ST_LinestringFromWKB(bytea WKB);
geometry ST_LinestringFromWKB(bytea WKB, integer srid);
```

#### Descrição

A função `ST_LinestringFromWKB`, pega uma representação binária bem conhecida de geometria e um sistema de referência espacial ID (SRID) e cria um exemplo do tipo apropriado de geometria - nesse caso, uma geometria `LINESTRING`. Essa função cumpre o papel da Fábrica de Geometria SQL.

Se um SRID não estiver especificado, isso leva a 0. Retorna NULA se a entrada `bytea` não representa uma geometria `LINESTRING`. Isso é um heterônimo para [ST\\_LineFromWKB](#).



#### Note

OGC SPEC 3.2.6.2 - o SRID opcional é da suíte de conformação.



#### Note

Se você sabe que todas suas geometrias são `LINESTRING`s, é mais eficaz usar [ST\\_GeomFromWKB](#). Essa função convida [ST\\_GeomFromWKB](#) e adiciona validação extra que ela retorna uma `LINESTRING`.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification. SQL-MM 3: 7.2.9

## Examples

```
SELECT
  ST_LineStringFromWKB(
    ST_AsBinary(ST_GeomFromText('LINESTRING(1 2, 3 4)'))
  ) AS aline,
  ST_LineStringFromWKB(
    ST_AsBinary(ST_GeomFromText('POINT(1 2)'))
  ) IS NULL AS null_return;
-----
aline                               | null_return
-----
0102000000020000000000000000000000F ... | t
```

## Veja também

[ST\\_GeomFromWKB](#), [ST\\_LineFromWKB](#)

### 8.4.24 ST\_MakeBox2D

`ST_MakeBox2D` — Cria uma CAIXA2D definida pelos pontos dados das geometrias.

#### Synopsis

box2d `ST_MakeBox2D`(geometria pointLowLeft, geometria pointUpRight);

#### Descrição

Cria uma CAIXA2D definida pelos pontos dados das geometrias. Isso é útil para fazer pesquisas de alcance

#### Examples

```
--Return all features that fall reside or partly reside in a US national atlas coordinate ↔
  bounding box
--It is assumed here that the geometries are stored with SRID = 2163 (US National atlas ↔
  equal area)
SELECT feature_id, feature_name, the_geom
FROM features
WHERE the_geom && ST_SetSRID(ST_MakeBox2D(ST_Point(-989502.1875, 528439.5625),
  ST_Point(-987121.375 , 529933.1875)), 2163)
```

## Veja também

[ST\\_MakePoint](#), [ST\\_Point](#), [ST\\_SetSRID](#), [ST\\_SRID](#)

### 8.4.25 ST\_3DMakeBox

`ST_3DMakeBox` — Cria uma CAIXA2D definida pelos pontos 3d dados das geometrias.

#### Synopsis

box3d `ST_3DMakeBox`(geometry point3DLowLeftBottom, geometry point3DUpRightTop);

## Descrição

Cria uma CAIXA2D definida pelos pontos 2 3D dados das geometrias.



Essa função suporta 3d e não irá derrubar o z-index.

Alterações: 2.0.0 Nas versões anteriores era chamado de ST\_MakeBox3D

## Examples

```
SELECT ST_3DMakeBox(ST_MakePoint(-989502.1875, 528439.5625, 10),
                    ST_MakePoint(-987121.375, 529933.1875, 10)) As abb3d

--bb3d--
-----
BOX3D(-989502.1875 528439.5625 10,-987121.375 529933.1875 10)
```

## Veja também

[ST\\_MakePoint](#), [ST\\_SetSRID](#), [ST\\_SRID](#)

## 8.4.26 ST\_MakeLine

ST\_MakeLine — Cria uma Linestring de ponto, multiponto ou linha das geometrias.

### Synopsis

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

### Descrição

A ST\_MakeLine vem de 3 maneiras: um agregado espacial que pega filas de ponto, multiponto ou linhas das geometrias e retorna uma line string; uma função que pega uma ordem de ponto, multiponto ou linha e uma função normal que pega dois pontos, multipontos ou linhas das geometrias. Você pode querer usar um subselect para ordenar antes de sustentar eles para a versão agregada dessa função.

Entradas diferentes de ponto, multiponto ou linhas, serão ignoradas.

Quando adicionar linhas componentes, nó comuns no começo das linhas são removidos da saída. Nós comuns em entradas ponto e multiponto não são removidos.



This function supports 3d and will not drop the z-index.

Availability: 2.3.0 - Support for multipoint input elements was introduced

Disponibilidade: 2.0.0 - Suporte para elementos de entrada linestring foi introduzido

Disponibilidade: 1.4.0 - ST\_MakeLine(geomarray) foi introduzida. A ST\_MakeLine agrega funções que foram melhoradas para lidar com mais pontos mais rápido.

**Exemplos: Versão espacial agregada**

Esse exemplo pega uma sequência de pontos do GPS e cria um relato para cada torre gps onde o campo geométrico é uma line string composta com os pontos do gps na ordem da viagem.

```
-- For pre-PostgreSQL 9.0 - this usually works,
-- but the planner may on occasion choose not to respect the order of the subquery
SELECT gps.gps_track, ST_MakeLine(gps.the_geom) As newgeom
      FROM (SELECT gps_track, gps_time, the_geom
            FROM gps_points ORDER BY gps_track, gps_time) As gps
      GROUP BY gps.gps_track;
```

```
-- If you are using PostgreSQL 9.0+
-- (you can use the new ORDER BY support for aggregates)
-- this is a guaranteed way to get a correctly ordered linestring
-- Your order by part can order by more than one column if needed
SELECT gps.gps_track, ST_MakeLine(gps.the_geom ORDER BY gps_time) As newgeom
      FROM gps_points As gps
      GROUP BY gps.gps_track;
```

**Exemplos: Versão não espacial agregada**

O primeiro exemplo é um one off line string composto de 2 pontos. O segundo formula line strings de 2 pontos que um usuário desenha. O terceiro é um one-off que junta pontos 2 3d para criar uma linha em um espaço 3d.

```
SELECT ST_AsText(ST_MakeLine(ST_MakePoint(1,2), ST_MakePoint(3,4)));
      st_astext
-----
LINESTRING(1 2,3 4)

SELECT userpoints.id, ST_MakeLine(startpoint, endpoint) As drawn_line
      FROM userpoints ;

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

**Exemplos: Utilizando versão banco de dados**

```
SELECT ST_MakeLine(ARRAY(SELECT ST_Centroid(the_geom) FROM visit_locations ORDER BY ↵
      visit_time));

--Making a 3d line with 3 3-d points
SELECT ST_AsEWKT(ST_MakeLine(ARRAY[ST_MakePoint(1,2,3),
      ST_MakePoint(3,4,5), ST_MakePoint(6,6,6)]));
      st_asewkt
-----
LINESTRING(1 2 3,3 4 5,6 6 6)
```

**Veja também**

[ST\\_AsEWKT](#), [ST\\_AsText](#), [ST\\_GeomFromText](#), [ST\\_MakePoint](#)



### 8.4.27 ST\_MakeEnvelope

`ST_MakeEnvelope` — Cria um polígono retangular formado a partir dos mínimos e máximos dados. Os valores de entrada devem ser em SRS especificados pelo SRID.

#### Synopsis

geometry **ST\_MakeEnvelope**(double precision xmin, double precision ymin, double precision xmax, double precision ymax, integer srid=unknown);

#### Descrição

Cria um polígono retangular formado a partir do mínimo e máximo, pela dada shell. Os valores de entradas devem ser SRS especificados pelo SRID. Se nenhum SRID for especificado o sistema de referência espacial desconhecido é assumido

Disponibilidade: 1.5

Melhorias: 2.0: Habilidade para especificar um pacote sem especificar um SRID foi introduzida.

#### Exemplo: Construindo um polígono bounding box

```
SELECT ST_AsText(ST_MakeEnvelope(10, 10, 11, 11, 4326));
st_asewkt
-----
POLYGON((10 10, 10 11, 11 11, 11 10, 10 10))
```

#### Veja também

[ST\\_MakePoint](#), [ST\\_MakeLine](#), [ST\\_MakePolygon](#)

### 8.4.28 ST\_MakePolygon

`ST_MakePolygon` — Cria uma polígono formado pela dada shell. As geometrias de entrada devem ser LINESTRINGS fechadas.

#### Synopsis

geometry **ST\_MakePolygon**(geometry linestring);  
 geometry **ST\_MakePolygon**(geometry outerlinestring, geometry[] interiorlinestrings);

#### Descrição

Cria uma polígono formado pela dada shell. As geometrias de entrada devem ser LINESTRINGS fechadas. Vem em 2 variantes.

Variante 1: Pega uma linestring fechada.

Variante 2: Cria um polígono formado pela dada shell e pela coleção de buracos. Você pode construir um banco de dados de geometrias usando as construções `ST_Accum` ou o PostgreSQL `ARRAY[]` e `ARRAY()`. Geometrias de entrada devem ser LINESTRINGS fechadas.



#### Note

Essa função não aceitará uma MULTILINESTRING. Use [ST\\_LineMerge](#) ou [ST\\_Dump](#) para gerar line strings.



This function supports 3d and will not drop the z-index.

**Exemplos: LINESTRING fechada única**

```
--2d line
SELECT ST_MakePolygon(ST_GeomFromText('LINESTRING(75.15 29.53,77 29,77.6 29.5, 75.15 29.53) ←
  '));
--If linestring is not closed
--you can add the start point to close it
SELECT ST_MakePolygon(ST_AddPoint(foo.open_line, ST_StartPoint(foo.open_line)))
FROM (
SELECT ST_GeomFromText('LINESTRING(75.15 29.53,77 29,77.6 29.5)') As open_line) As foo;

--3d closed line
SELECT ST_MakePolygon(ST_GeomFromText('LINESTRING(75.15 29.53 1,77 29 1,77.6 29.5 1, 75.15 ←
  29.53 1)'));

st_asewkt
-----
POLYGON((75.15 29.53 1,77 29 1,77.6 29.5 1,75.15 29.53 1))

--measured line --
SELECT ST_MakePolygon(ST_GeomFromText('LINESTRINGM(75.15 29.53 1,77 29 1,77.6 29.5 2, 75.15 ←
  29.53 2)'));

st_asewkt
-----
POLYGONM((75.15 29.53 1,77 29 1,77.6 29.5 2,75.15 29.53 2))
```

**Exemplos: estrutura de dentro e estrutura de fora****Construir um donut com um buraco de formiga**

```
SELECT ST_MakePolygon(
    ST_ExteriorRing(ST_Buffer(foo.line,10)),
    ARRAY[ST_Translate(foo.line,1,1),
    ST_ExteriorRing(ST_Buffer(ST_MakePoint(20,20),1)) ]
)
FROM
    (SELECT ST_ExteriorRing(ST_Buffer(ST_MakePoint(10,10),10,10))
    As line )
    As foo;
```

Construir limites de província com buracos representando lagos na província de um conjunto de polígonos/multipolígonos e linestrings de água das províncias. Esse é um exemplo do uso do PostGIS ST\_Accum.

**Note**

A construção CASE é usada porque sustentar uma coleção de nulos em ST\_MakePolygon resulta em NULO.

**Note**

Um ingresso esquerdo é usado para garantir que temos todas as províncias de volta, mesmo se elas não tiverem lagos.

```
SELECT p.gid, p.province_name,
    CASE WHEN
```

```

        ST_Accum(w.the_geom) IS NULL THEN p.the_geom
    ELSE ST_MakePolygon(ST_LineMerge(ST_Boundary(p.the_geom)), ST_Accum(w. ↔
        the_geom)) END
FROM
    provinces p LEFT JOIN waterlines w
        ON (ST_Within(w.the_geom, p.the_geom) AND ST_IsClosed(w.the_geom))
GROUP BY p.gid, p.province_name, p.the_geom;

--Same example above but utilizing a correlated subquery
--and PostgreSQL built-in ARRAY() function that converts a row set to an array

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

**Veja também**

[ST\\_Boundary](#), [ST\\_Accum](#), [ST\\_AddPoint](#), [ST\\_GeometryType](#), [ST\\_IsClosed](#), [ST\\_LineMerge](#), [ST\\_BuildArea](#)

**8.4.29 ST\_MakePoint**

`ST_MakePoint` — Creates a 2D, 3DZ or 4D point geometry.

**Synopsis**

geometry **ST\_MakePoint**(double precision x, double precision y);

geometry **ST\_MakePoint**(double precision x, double precision y, double precision z);

geometry **ST\_MakePoint**(double precision x, double precision y, double precision z, double precision m);

**Descrição**

Creates a 2D, 3DZ or 4D point geometry (geometry with measure). `ST_MakePoint` while not being OGC compliant is generally faster and more precise than [ST\\_GeomFromText](#) and [ST\\_PointFromText](#). It is also easier to use if you have raw coordinates rather than WKT.

**Note**

Note que x é longitude e y é latitude

**Note**

Use [ST\\_MakePointM](#) if you need to make a point with x, y and m.



This function supports 3d and will not drop the z-index.

### Examples

```
--Return point with unknown SRID
SELECT ST_MakePoint(-71.1043443253471, 42.3150676015829);

--Return point marked as WGS 84 long lat
SELECT ST_SetSRID(ST_MakePoint(-71.1043443253471, 42.3150676015829), 4326);

--Return a 3D point (e.g. has altitude)
SELECT ST_MakePoint(1, 2, 1.5);

--Get z of point
SELECT ST_Z(ST_MakePoint(1, 2, 1.5));
result
-----
1.5
```

### Veja também

[ST\\_GeomFromText](#), [ST\\_PointFromText](#), [ST\\_SetSRID](#), [ST\\_MakePointM](#)

## 8.4.30 ST\_MakePointM

**ST\_MakePointM** — Cria um ponto com uma coordenada x y e m.

### Synopsis

geometry **ST\_MakePointM**(float x, float y, float m);

### Descrição

Cria um ponto com uma coordenada x y e medida.



#### Note

Note que x é longitude e y é latitude.

### Examples

Usamos `ST_AsEWKT` nesses exemplos para demonstrar a representação de texto ao invés de `ST_AsText` porque `ST_AsText` não suporta o M que retorna.

```
--Return EWKT representation of point with unknown SRID
SELECT ST_AsEWKT(ST_MakePointM(-71.1043443253471, 42.3150676015829, 10));

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

```

--Return EWKT representation of point with measure marked as WGS 84 long lat
SELECT ST_AsEWKT(ST_SetSRID(ST_MakePointM(-71.1043443253471, 42.3150676015829,10),4326));

                                     st_asewkt
-----
SRID=4326;POINTM(-71.1043443253471 42.3150676015829 10)

--Return a 3d point (e.g. has altitude)
SELECT ST_MakePoint(1, 2,1.5);

--Get m of point
SELECT ST_M(ST_MakePointM(-71.1043443253471, 42.3150676015829,10));
result
-----
10

```

### Veja também

[ST\\_AsEWKT](#), [ST\\_MakePoint](#), [ST\\_SetSRID](#)

### 8.4.31 ST\_MLineFromText

ST\_MLineFromText — Retorna um valor específico ST\_MultiLineString de uma representação WKT.

#### Synopsis

geometria **ST\_MLineFromText**(texto WKT, integer srid);  
 geometria **ST\_MLineFromText**(texto WKT);

#### Descrição

Makes a Geometry from Well-Known-Text (WKT) with the given SRID. If SRID is not given, it defaults to 0.

OGC SPEC 3.2.6.2 - opção SRID é de uma suíte de conformação

Retorna nulo se o WKT não é uma MULTILINESTRING



#### Note

Se você tem total certeza de que todas suas geometrias WKT são pontos, não use essa função. Ela é mais devagar que a ST\_GeomFromText, já que adiciona um passo de validação adicional.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification. SQL-MM 3: 9.4.4

#### Examples

```
SELECT ST_MLineFromText('MULTILINESTRING((1 2, 3 4), (4 5, 6 7))');
```

## Veja também

[ST\\_GeomFromText](#)

### 8.4.32 ST\_MPointFromText

ST\_MPointFromText — Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

#### Synopsis

```
geometry ST_MPointFromText(text WKT, integer srid);  
geometry ST_MPointFromText(text WKT);
```

#### Descrição

Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

OGC SPEC 3.2.6.2 - opção SRID é de uma suíte de conformação

Retorna nulo se o WKT não é um MULTIPONTO



#### Note

Se você tem total certeza de que todas suas geometrias WKT são pontos, não use essa função. Ela é mais devagar que a ST\_GeomFromText, já que adiciona um passo de validação adicional.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). 3.2.6.2



This method implements the SQL/MM specification. SQL-MM 3: 9.2.4

#### Examples

```
SELECT ST_MPointFromText('MULTIPOINT(1 2, 3 4)');  
SELECT ST_MPointFromText('MULTIPOINT(-70.9590 42.1180, -70.9611 42.1223)', 4326);
```

## Veja também

[ST\\_GeomFromText](#)

### 8.4.33 ST\_MPolyFromText

ST\_MPolyFromText — Makes a MultiPolygon Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

#### Synopsis

```
geometria ST_MPolyFromText(text WKT, inteiro srid);  
geometria ST_MPolyFromText(texto WKT);
```

**Descrição**

Makes a MultiPolygon from WKT with the given SRID. If SRID is not given, it defaults to 0.

OGC SPEC 3.2.6.2 - opção SRID é de uma suíte de conformação

Descarta um erro se o WKT não for um MULTIPOLÍGONO

**Note**

Se você tem total certeza de que todas suas geometrias WKT são multipoligonos, não use essa função. Ela é mais devagar que a ST\_GeomFromText, já que adiciona um passo de validação adicional.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification. SQL-MM 3: 9.6.4

**Examples**

```
SELECT ST_MPolyFromText('MULTIPOLYGON(((0 0 1,20 0 1,20 20 1,0 20 1,0 0 1),(5 5 3,5 7 3,7 7 ←
  3,7 5 3,5 5 3)))');
SELECT ST_MPolyFromText('MULTIPOLYGON((-70.916 42.1002,-70.9468 42.0946,-70.9765 ←
  42.0872,-70.9754 42.0875,-70.9749 42.0879,-70.9752 42.0881,-70.9754 42.0891,-70.9758 ←
  42.0894,-70.9759 42.0897,-70.9759 42.0899,-70.9754 42.0902,-70.9756 42.0906,-70.9753 ←
  42.0907,-70.9753 42.0917,-70.9757 42.0924,-70.9755 42.0928,-70.9755 42.0942,-70.9751 ←
  42.0948,-70.9755 42.0953,-70.9751 42.0958,-70.9751 42.0962,-70.9759 42.0983,-70.9767 ←
  42.0987,-70.9768 42.0991,-70.9771 42.0997,-70.9771 42.1003,-70.9768 42.1005,-70.977 ←
  42.1011,-70.9766 42.1019,-70.9768 42.1026,-70.9769 42.1033,-70.9775 42.1042,-70.9773 ←
  42.1043,-70.9776 42.1043,-70.9778 42.1048,-70.9773 42.1058,-70.9774 42.1061,-70.9779 ←
  42.1065,-70.9782 42.1078,-70.9788 42.1085,-70.9798 42.1087,-70.9806 42.109,-70.9807 ←
  42.1093,-70.9806 42.1099,-70.9809 42.1109,-70.9808 42.1112,-70.9798 42.1116,-70.9792 ←
  42.1127,-70.979 42.1129,-70.9787 42.1134,-70.979 42.1139,-70.9791 42.1141,-70.9987 ←
  42.1116,-71.0022 42.1273,
  -70.9408 42.1513,-70.9315 42.1165,-70.916 42.1002)))',4326);
```

**Veja também**

[ST\\_GeomFromText](#), [ST\\_SRID](#)

**8.4.34 ST\_Point**

ST\_Point — Retorna uma ST\_Point com os valores de coordenada dados. Heterônimo OGC para ST\_MakePoint.

**Synopsis**

geometria **ST\_Point**(float x\_lon, float y\_lat);

**Descrição**

Retorna uma ST\_Point com os valores de coordenada dados. Heterônimo submisso para ST\_MakePoint que pega somente o x e o y.



This method implements the SQL/MM specification. SQL-MM 3: 6.1.2

**Exemplos: Geometria**

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829),4326)
```

**Exemplos: Geografia**

```
SELECT CAST(ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829),4326) Como geografia);
```

```
-- the :: is PostgreSQL short-hand for casting.
```

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829),4326)::geography;
```

```
--If your point coordinates are in a different spatial reference from WGS-84 long lat, then ↵  
you need to transform before casting
```

```
-- This example we convert a point in Pennsylvania State Plane feet to WGS 84 and then ↵  
geography
```

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

**Veja também**

Section [4.2.1](#), [ST\\_MakePoint](#), [ST\\_SetSRID](#), [ST\\_Transform](#)

**8.4.35 ST\_PointFromGeoHash**

`ST_PointFromGeoHash` — Retorna um ponto de uma string GeoHash.

**Synopsis**

ponto `ST_PointFromGeoHash`(texto geohash, inteiro precision=full\_precision\_of\_geohash);

**Descrição**

Retorna um ponto de uma string GeoHash. O ponto representa o ponto central do GeoHash.

Se nenhuma `precisão` for especificada, a `ST_PointFromGeoHash` retorna um ponto baseado na precisão completa da string da entrada GeoHash.

Se a `precisão` for especificada, a `ST_PointFromGeoHash` irá usar aqueles vários caracteres do GeoHash para criar o ponto.

Disponibilidade: 2.1.0

**Examples**

```
SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxcggy4d0dbxqz0'));
      st_astext
-----
POINT(-115.172816 36.114646)

SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxcggy4d0dbxqz0', 4));
      st_astext
-----
POINT(-115.13671875 36.123046875)

SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxcggy4d0dbxqz0', 10));
      st_astext
```



```
POINT(-115.172815918922 36.1146435141563)
```

### Veja também

[ST\\_GeoHash](#), [ST\\_Box2dFromGeoHash](#), [ST\\_GeomFromGeoHash](#)

## 8.4.36 ST\_PointFromText

`ST_PointFromText` — Faz um ponto de um WKT com o SRID dado. Se o SRID não for dado, isso leva a desconhecido.

### Synopsis

```
geometry ST_PointFromText(texto WKT);  
geometria ST_PointFromText(texto WKT, inteiro srid);
```

### Descrição

Constructs a PostGIS `ST_Geometry` point object from the OGC Well-Known text representation. If SRID is not given, it defaults to unknown (currently 0). If geometry is not a WKT point representation, returns null. If completely invalid WKT, then throws an error.



#### Note

Existem 2 variantes da função `ST_PointFromText`, a primeira não pega nenhuma SRID e retorna uma geometria sem sistema de referência espacial definido. A segunda, pega uma id referência espacial como o segundo argumento e retorna uma `ST_Geometry` que inclui esse srid como parte dos seus metadados. O srid deve ser definido na `spatial_ref_sys` table.



#### Note

Se você tem total certeza de que todas suas geometrias WKT são pontos, não use essa função. Ela é mais devagar que a `ST_GeomFromText`, já que adiciona um passo de validação adicional. Se você está construindo pontos de coordenadas long lat e se importa mais com apresentação e precisão do que com concordância OGC, use: [ST\\_MakePoint](#) ou [ST\\_Point](#).



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2 - opção SRID é da suíte de conformidade.



This method implements the SQL/MM specification. SQL-MM 3: 6.1.8

### Examples

```
SELECT ST_PointFromText ('POINT(-71.064544 42.28787) ');  
SELECT ST_PointFromText ('POINT(-71.064544 42.28787)', 4326);
```

### Veja também

[ST\\_GeomFromText](#), [ST\\_MakePoint](#), [ST\\_Point](#), [ST\\_SRID](#)

### 8.4.37 ST\_PointFromWKB

ST\_PointFromWKB — Faz uma geometria a partir de um WKB com o SRID dado

#### Synopsis

geometria **ST\_GeomFromWKB**(bytea geom);  
geometry **ST\_GeomFromWKB**(bytea geom, inteiro srid);

#### Descrição

A função `ST_PointFromWKB`, pega uma representação binária bem conhecida de geometria e um sistema de referência espacial ID (SRID) e cria um exemplo do tipo apropriado de geometria - nesse caso, uma geometria `PONTO`. Essa função cumpre o papel da Fábrica de Geometria SQL.

Se uma SRID não for especificada, leva a 0. NULO é retornado se a entrada `bytea` não representar uma `PONTO` geometria.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.7.2



This method implements the SQL/MM specification. SQL-MM 3: 6.1.9



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

#### Examples

```
SELECT
  ST_AsText (
    ST_PointFromWKB (
      ST_AsEWKB ('POINT(2 5) '::geometry)
    )
  );
st_astext
-----
POINT(2 5)
(1 row)

SELECT
  ST_AsText (
    ST_PointFromWKB (
      ST_AsEWKB ('LINESTRING(2 5, 2 6) '::geometry)
    )
  );
st_astext
-----

(1 row)
```

#### Veja também

[ST\\_GeomFromWKB](#), [ST\\_LineFromWKB](#)

### 8.4.38 ST\_Polygon

ST\_Polygon — Retorna um polígono construído de linestring e SRID especificados.

#### Synopsis

```
geometry ST_Polygon(geometry aLineString, integer srid);
```

#### Descrição

Retorna um polígono construído de linestring e SRID especificados.



#### Note

ST\_Polygon is similar to first version of ST\_MakePolygon except it also sets the spatial ref sys (SRID) of the polygon. Will not work with MULTILINESTRINGS so use LineMerge to merge multilines. Also does not create polygons with holes. Use ST\_MakePolygon for that.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 8.3.2



This function supports 3d and will not drop the z-index.

#### Examples

```
--a 2d polygon
SELECT ST_Polygon(ST_GeomFromText('LINESTRING(75.15 29.53,77 29,77.6 29.5, 75.15 29.53)'), ←
  4326);

--result--
POLYGON((75.15 29.53,77 29,77.6 29.5,75.15 29.53))
--a 3d polygon
SELECT ST_AsEWKT(ST_Polygon(ST_GeomFromEWKT('LINESTRING(75.15 29.53 1,77 29 1,77.6 29.5 1, ←
  75.15 29.53 1)'), 4326));

result
-----
SRID=4326;POLYGON((75.15 29.53 1,77 29 1,77.6 29.5 1,75.15 29.53 1))
```

#### Veja também

[ST\\_AsEWKT](#), [ST\\_AsText](#), [ST\\_GeomFromEWKT](#), [ST\\_GeomFromText](#), [ST\\_LineMerge](#), [ST\\_MakePolygon](#)

### 8.4.39 ST\_PolygonFromText

ST\_PolygonFromText — Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

#### Synopsis

```
geometry ST_PolygonFromText(text WKT);
geometry ST_PolygonFromText(text WKT, integer srid);
```

**Descrição**

Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0. Returns null if WKT is not a polygon.  
OGC SPEC 3.2.6.2 - opção SRID é de uma suíte de conformação

**Note**

Se você tem total certeza de que todas suas geometrias WKT são polígonos, não use essa função. Ela é mais devagar que a ST\_GeomFromText, já que adiciona um passo de validação adicional.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification. SQL-MM 3: 8.3.6

**Examples**

```
SELECT ST_PolygonFromText('POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 ↵
 42.3903701743239,
-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 ↵
 42.3902909739571))');
st_polygonfromtext
-----
010300000001000000050000006...
```

```
SELECT ST_PolygonFromText('POINT(1 2)') IS NULL as point_is_notpoly;
point_is_not_poly
-----
t
```

**Veja também**

[ST\\_GeomFromText](#)

**8.4.40 ST\_WKBToSQL**

ST\_WKBToSQL — Retorna um valor ST\_Geometry específico da representação de texto binário bem conhecida (WKB). Isso é um heterônimo para ST\_GeomFromWKB que não pega nenhum srid

**Synopsis**

geometry ST\_WKBToSQL(bytea WKB);

**Descrição**

This method implements the SQL/MM specification. SQL-MM 3: 5.1.36

**Veja também**[ST\\_GeomFromWKB](#)**8.4.41 ST\_WKTToSQL**

**ST\_WKTToSQL** — Retorna um valor `ST_Geometry` específico da representação de texto estendida bem conhecida (EWKT). Isso é um heterônimo para `ST_GeomFromText`

**Synopsis**

geometria **ST\_WKTToSQL**(texto WKT);

**Descrição**

This method implements the SQL/MM specification. SQL-MM 3: 5.1.34

**Veja também**[ST\\_GeomFromText](#)**8.5 Acessors de Geometria****8.5.1 Tipo de geometria**

**Tipo de geometria** — Retorna o tipo de geometria como uma string. Exemplos: 'LINESTRING', 'POLÍGONO', 'MULTIPOINT', etc.

**Synopsis**

texto **GeometryType**(geometria geomA);

**Descrição**

Retorna o tipo de geometria como uma string. Exemplos: 'LINESTRING', 'POLÍGONO', 'MULTIPOINT', etc.

OGC SPEC s2.1.1.1 - Retorna o nome do sub tipo ocasional da geometria da qual essa geometria ocasiona é um membro. O nome do sub tipo ocasional retorna como uma string.

**Note**

Essa função também indica se a geometria é medida, retornando uma string da forma 'POINTM'.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TIN introduzido.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### Exemplos

```
SELECT GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
geometrytype
-----
LINESTRING
```

```
SELECT ST_GeometryType(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)
) )'));
--result
POLYHEDRALSURFACE
```

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

### Veja também

[ST\\_GeometryType](#)

## 8.5.2 ST\_Boundary

**ST\_Boundary** — Retorna o encerramento da borda combinatória dessa geometria.

## Synopsis

geometria **ST\_Boundary**(geometria geomA);

## Descrição

Retorna o encerramento do limite combinatório dessa geometria. O limite combinatório é definido com descrito na seção 3.12.3.2 do OGC SPEC. Porque o resultado dessa função é um encerramento, e por isso topologicamente fechado, o limite resultante pode ser representado usando geometrias primitivas representacionais como foi discutido no OGC SPEC, seção 3.12.2.

Desempenhado pelo módulo GEOS



### Note

Anterior a 2.0.0, essa função abre uma exceção se usada com `GEOMETRYCOLLECTION`. A partir do 2.0.0 ela vai retornar NULA (entrada não suportada).

---



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). OGC SPEC s2.1.1.1



This method implements the SQL/MM specification. SQL-MM 3: 5.1.14




This function supports 3d and will not drop the z-index.

Melhorias: 2.1.0 suporte para Triângulo foi introduzido

## Exemplos

---




*Linestring com pontos de limite cobertos*

```

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

-- ST_AsText output
MULTIPOINT(100 150,160 170)
    
```



*furos de polígono com multilinestring limite*

```

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

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

```

SELECT ST_AsText(ST_Boundary(ST_GeomFromText('LINESTRING(1 1,0 0, -1 1)')));
st_astext
-----
MULTIPOINT(1 1,-1 1)

SELECT ST_AsText(ST_Boundary(ST_GeomFromText('POLYGON((1 1,0 0, -1 1, 1 1)'))));
st_astext
-----
LINESTRING(1 1,0 0,-1 1,1 1)

--Using a 3d polygon
SELECT ST_AsEWKT(ST_Boundary(ST_GeomFromEWKT('POLYGON((1 1 1,0 0 1, -1 1 1, 1 1 1)'))));
st_asewkt
-----
LINESTRING(1 1 1,0 0 1,-1 1 1,1 1 1)

--Using a 3d multilinestring
SELECT ST_AsEWKT(ST_Boundary(ST_GeomFromEWKT('MULTILINESTRING((1 1 1,0 0 0.5, -1 1 1),(1 1 ←
      0.5,0 0 0.5, -1 1 0.5, 1 1 0.5) ))'));
st_asewkt
-----
MULTIPOINT(-1 1 1,1 1 0.75)
    
```



**Veja também**

[ST\\_AsText](#), [ST\\_ExteriorRing](#), [ST\\_MakePolygon](#)

### 8.5.3 ST\_CoordDim

ST\_CoordDim — Retorna a dimensão da coordenada do valor ST\_Geometry.

**Synopsis**

inteiro **ST\_CoordDim**(geometria geomA);

**Descrição**

Retorna a dimensão da coordenada do valor ST\_Geometry.

Esse é o pseudônimo condescendente do MM para [ST\\_NDims](#)



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 5.1.3



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

**Exemplos**

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

```
---result--
```

```
3
```

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

```
--result--
```

```
2
```

**Veja também**

[ST\\_NDims](#)

## 8.5.4 ST\_Dimension

`ST_Dimension` — A dimensão herdada desse objeto geométrico, o qual deve ser menor ou igual à dimensão coordenada.

### Synopsis

inteiro `ST_Dimension`(geometria g);

### Descrição

A dimensão herdada desse objeto geométrico, que deve ser menor que ou igual à dimensão coordenada. OGC SPEC s2.1.1.1 - retorna 0 para PUNTO, 1 para LINESTRING, 2 para POLÍGONO, e a dimensão mais larga dos componentes de uma COLEÇÃO DE GEOMETRIAS. Se desconhecida (geometria vazia) nula é retornada.



This method implements the SQL/MM specification. SQL-MM 3: 5.1.2

Melhorias: 2.0.0 suporte para superfícies poliédricas e TINs foi introduzido. Não abre mais exceção se uma geometria vazia é dada.



#### Note

Anterior à 2.0.0, essa função abre uma exceção se usada com uma geometria vazia.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### Exemplos

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

ST_Dimension
-----
1
```

### Veja também

[ST\\_NDims](#)

## 8.5.5 ST\_EndPoint

`ST_EndPoint` — Retorna ao último ponto de uma LINESTRING ou CIRCULARLINESTRING geometria como um PONTO.

### Synopsis

booleana `ST_EndPoint`(geometria g);

## Descrição

Retorna ao último ponto de uma `LINestring` geometria como um `PONTO` ou `NULO` se o parâmetro de entrada não é uma `LINestring`.



This method implements the SQL/MM specification. SQL-MM 3: 7.1.4



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

### Note



Alterações: 2.0.0 não funciona mais com geometrias de multilinesstrings. Em verões mais antigas do PostGIS -- uma linha multilinesstring sozinha trabalharia normalmente com essa função e voltaria o ponto de início. Na 2.0.0 ela retorna NULA como qualquer outra multilinesstring. O antigo comportamento não foi uma característica documentada, mas as pessoas que consideravam que tinham seus dados armazenados como uma `LINestring`, agora podem experimentar essas que retornam NULAS em 2.0.

## Exemplos

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

postgis=# SELECT ST_EndPoint('POINT(1 1)::geometry') IS NULL AS is_null;
is_null
-----
t
(1 row)

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

## Veja também

[ST\\_PointN](#), [ST\\_StartPoint](#)

## 8.5.6 ST\_Envelope

`ST_Envelope` — Retorna uma geometria representando a precisão da dobrada (`float8`) da caixa limitada da geometria fornecida.

### Synopsis

```
geometria ST_Envelope(geometria g1);
```

## Descrição

Retorna o limite mínimo da caixa float8 para a geometria fornecida, com uma geometria. O polígono é definido pelos pontos de canto da caixa limitada ((MINX, MINY), (MINX, MAXY), (MAXX, MAXY), (MAXX, MINY), (MINX, MINY)). (PostGIS irá adicionar uma ZMIN/ZMAX coordenada também).

Casos degenerados (linhas verticais, pontos) irão retornar como uma geometria de dimensão menor que POLÍGONO, ie. PONTO ou LINESTRING.

Disponibilidade: 1.5.0 comportamento alterado para saída de precisão dupla ao invés de float4



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.1



This method implements the SQL/MM specification. SQL-MM 3: 5.1.15

## Exemplos

```
SELECT ST_AsText(ST_Envelope('POINT(1 3)::geometry'));
 st_astext
-----
POINT(1 3)
(1 row)

SELECT ST_AsText(ST_Envelope('LINESTRING(0 0, 1 3)::geometry'));
          st_astext
-----
POLYGON((0 0,0 3,1 3,1 0,0 0))
(1 row)

SELECT ST_AsText(ST_Envelope('POLYGON((0 0, 0 1, 1.0000001 1, 1.0000001 0, 0 0))::geometry ←
));
          st_astext
-----
POLYGON((0 0,0 1,1.00000011920929 1,1.00000011920929 0,0 0))
(1 row)
SELECT ST_AsText(ST_Envelope('POLYGON((0 0, 0 1, 1.0000000001 1, 1.0000000001 0, 0 0))':: ←
geometry));
          st_astext
-----
POLYGON((0 0,0 1,1.00000011920929 1,1.00000011920929 0,0 0))
(1 row)

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



*Envelope of a point and linestring.*

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

#### Veja também

[Caixa2D](#), [Caixa3D](#), [ST\\_OrientedEnvelope](#)

### 8.5.7 ST\_BoundingDiagonal

`ST_BoundingDiagonal` — Retorna a diagonal da geometria fornecida da caixa limitada.

#### Synopsis

geometria `ST_BoundingDiagonal`(geometria geom, booleana fits=false);

#### Descrição

Retorna a diagonal da geometria fornecida da caixa limitada em linestring. Se a entrada da geometria está vazia, a linha diagonal também está, caso contrário é uma linestring de 2-pontos com valores mínimos de cada dimensão no ponto de início e com valores máximos no ponto de fim.

A linestring da geometria retornada sempre retém SRID e dimensionalidade (Z e M presentes) da geometria de entrada.

O parâmetro `fits` especifica se o que se encaixa melhor é necessário. Se negativo, a diagonal de uma caixa limitadora de alguma forma pode ser aceita (é mais rápido obter para geometrias com muitos vértices). De qualquer forma, a caixa limitadora da linha diagonal retornada sempre cobre a geometria de entrada.

**Note**

Em casos degenerados (um único vértice na entrada) a linestring retornada será topologicamente inválida (sem interior). Isso não torna o retorno semanticamente inválido.

Disponibilidade: 2.2.0



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

**Exemplos**

```
-- Get the minimum X in a buffer around a point
SELECT ST_X(ST_StartPoint(ST_BoundingDiagonal(
  ST_Buffer(ST_MakePoint(0,0),10)
)));
st_x
-----
-10
```

**Veja também**

[ST\\_StartPoint](#), [ST\\_EndPoint](#), [ST\\_X](#), [ST\\_Y](#), [ST\\_Z](#), [ST\\_M](#), &&&

**8.5.8 ST\_ExteriorRing**

`ST_ExteriorRing` — Retorna uma line string representando o anel exterior da geometria POLÍGONO. Retorna NULA se a geometria não for um polígono. Não funcionará com MULTIPOLÍGONO.

**Synopsis**

geometria `ST_ExteriorRing`(geometry a\_polygon);

**Descrição**

Retorna uma line string representando o anel exterior da geometria POLÍGONO. Retorna NULA se a geometria não for um polígono.

**Note**

Funciona somente com geometrias tipo POLÍGONO.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). 2.1.5.1



This method implements the SQL/MM specification. SQL-MM 3: 8.2.3, 8.3.3



This function supports 3d and will not drop the z-index.

## Exemplos

```
--If you have a table of polygons
SELECT gid, ST_ExteriorRing(the_geom) AS ering
FROM sometable;

--If you have a table of MULTIPOLYGONS
--and want to return a MULTILINESTRING composed of the exterior rings of each polygon
SELECT gid, ST_Collect(ST_ExteriorRing(the_geom)) AS erings
      FROM (SELECT gid, (ST_Dump(the_geom)).geom As the_geom
            FROM sometable) As foo
GROUP BY gid;

--3d Example
SELECT ST_AsEWKT(
      ST_ExteriorRing(
      ST_GeomFromEWKT('POLYGON((0 0 1, 1 1 1, 1 2 1, 1 1 1, 0 0 1))')
      )
);

st_asewkt
-----
LINESTRING(0 0 1,1 1 1,1 2 1,1 1 1,0 0 1)
```

## Veja também

[ST\\_InteriorRingN](#), [ST\\_Boundary](#), [ST\\_NumInteriorRings](#)

## 8.5.9 ST\_GeometryN

**ST\_GeometryN** — Retorna a geometria de 1-base Nth se a geometria é uma **GEOMETRYCOLLECTION**, **(MULTI)POINT**, **(MULTI)LINESTRING**, **MULTICURVE** ou **(MULTI)POLYGON**, **POLYHEDRALSURFACE**. Senão, retorna NULA.

### Synopsis

geometria **ST\_GeometryN**(geometria geomA, inteiro n);

### Descrição

Retorna a geometria de 1-base Nth se a geometria é uma **GEOMETRYCOLLECTION**, **(MULTI)POINT**, **(MULTI)LINESTRING**, **MULTICURVE** ou **(MULTI)POLYGON**, **POLYHEDRALSURFACE**. Senão, retorna NULA.



#### Note

O Index é 1-base como para OGC specs desde a versão 0.8.0. Versões anteriores implementaram isso como 0-base.



#### Note

Se você quiser extrair todas as geometrias, de uma geometria, **ST\_Dump** é mais eficiente e também funcionará para geometrias singulares.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TIN introduzido.

Alterações: 2.0.0. Versões anteriores voltariam NULAS para geometrias únicas. Isso foi alterado para voltar a geometria para o caso ST\_GeometryN(...,1).



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 9.1.5



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### Exemplos Padrão

```
--Extracting a subset of points from a 3d multipoint
SELECT n, ST_AsEWKT(ST_GeometryN(the_geom, n)) As geomewkt
FROM (
VALUES (ST_GeomFromEWKT('MULTIPOINT(1 2 7, 3 4 7, 5 6 7, 8 9 10)') ),
( ST_GeomFromEWKT('MULTICURVE(CIRCULARSTRING(2.5 2.5,4.5 2.5, 3.5 3.5), (10 11, 12 11))') )
)As foo(the_geom)
CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(the_geom);
```

| n | geomewkt                                |
|---|-----------------------------------------|
| 1 | POINT(1 2 7)                            |
| 2 | POINT(3 4 7)                            |
| 3 | POINT(5 6 7)                            |
| 4 | POINT(8 9 10)                           |
| 1 | CIRCULARSTRING(2.5 2.5,4.5 2.5,3.5 3.5) |
| 2 | LINestring(10 11,12 11)                 |

```
--Extracting all geometries (useful when you want to assign an id)
SELECT gid, n, ST_GeometryN(the_geom, n)
FROM sometable CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(the_geom);
```

### Exemplos de Superfícies Poliédricas, TIN e Triângulos

```
-- Polyhedral surface example
-- Break a Polyhedral surface into its faces
SELECT ST_AsEWKT(ST_GeometryN(p_geom,3)) As geom_ewkt
FROM (SELECT ST_GeomFromEWKT('POLYHEDRALSURFACE(
((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1))
```



```

)') AS p_geom ) AS a;

-----
geom_ewkt
-----
POLYGON((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0))

-- TIN --
SELECT ST_AsEWKT(ST_GeometryN(geom,2)) as wkt
FROM
  (SELECT
    ST_GeomFromEWKT('TIN (((
      0 0 0,
      0 0 1,
      0 1 0,
      0 0 0
    )), ((
      0 0 0,
      0 1 0,
      1 1 0,
      0 0 0
    )))
  ) AS geom
) AS g;
-- result --
-----
wkt
-----
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))

```

## Veja também

[ST\\_Dump](#), [ST\\_NumGeometries](#)

### 8.5.10 ST\_GeometryType

ST\_GeometryType — Retorna o tipo de geometria de valor ST\_Geometry.

#### Synopsis

texto **ST\_GeometryType**(geometria g1);

#### Descrição

Retorna o tipo da geometria como uma string. EX: 'ST\_Linestring', 'ST\_Polygon', 'ST\_MultiPolygon' etc. Essa função difere de GeometryType(geometria) no caso da string e ST na frente que é retornada, bem como o fato que isso não indicará se a geometria é medida.

Melhorias: 2.0.0 suporte a superfícies poliédricas foi introduzido.



This method implements the SQL/MM specification. SQL-MM 3: 5.1.4



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

**Exemplos**

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

```
--result
```

```
ST_LineString
```

```
SELECT ST_GeometryType(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)
) )'));
--result
```

```
ST_PolyhedralSurface
```

```
SELECT ST_GeometryType(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)
) )'));
--result
```

```
ST_PolyhedralSurface
```

```
SELECT ST_GeometryType(geom) as result
FROM
  (SELECT
    ST_GeomFromEWKT('TIN (((
      0 0 0,
      0 0 1,
      0 1 0,
      0 0 0
    )), ((
      0 0 0,
      0 1 0,
      1 1 0,
      0 0 0
    ))) AS geom
  ) AS g;
result
-----
ST_Tin
```

**Veja também**

[Tipo de geometria](#)

**8.5.11 ST\_InteriorRingN**

**ST\_InteriorRingN** — Retorna o anel linestring Nth interior do polígono. Retorna NULO se a geometria não for um polígono ou o dado N está fora da extensão.

**Synopsis**

geometria **ST\_InteriorRingN**(geometria a\_polygon, inteiro n);

**Descrição**

Retorna o anel linestring Nth interior do polígono. Retorna NULO se a geometria não for um polígono ou o dado N está fora da extensão. index começa em 1.

**Note**

Isso não funcionará para MULTIPOLÍGONOS. Use em conjunção com ST\_Dump para MULTIPOLÍGONOS.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 8.2.6, 8.3.5



This function supports 3d and will not drop the z-index.

**Exemplos**

```
SELECT ST_AsText(ST_InteriorRingN(the_geom, 1)) As the_geom
FROM (SELECT ST_BuildArea(
        ST_Collect(ST_Buffer(ST_Point(1,2), 20,3),
                ST_Buffer(ST_Point(1, 2), 10,3))) As the_geom
      ) as foo
```

**Veja também**

[ST\\_ExteriorRing](#) [ST\\_BuildArea](#), [ST\\_Collect](#), [ST\\_Dump](#), [ST\\_NumInteriorRing](#), [ST\\_NumInteriorRings](#)

**8.5.12 ST\_IsPolygonCCW**

**ST\_IsPolygonCCW** — Returns true if all exterior rings are oriented counter-clockwise and all interior rings are oriented clockwise.

**Synopsis**

boolean **ST\_IsPolygonCCW** ( geometry geom );

**Descrição**

Returns true if all polygonal components of the input geometry use a counter-clockwise orientation for their exterior ring, and a clockwise direction for all interior rings.

Returns true if the geometry has no polygonal components.

**Note**

Closed linestrings are not considered polygonal components, so you would still get a true return by passing a single closed linestring no matter its orientation.

**Note**

If a polygonal geometry does not use reversed orientation for interior rings (i.e., if one or more interior rings are oriented in the same direction as an exterior ring) then both `ST_IsPolygonCW` and `ST_IsPolygonCCW` will return false.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

**Veja também**

[ST\\_ForcePolygonCW](#) , [ST\\_ForcePolygonCCW](#) , [ST\\_IsPolygonCW](#)

### 8.5.13 ST\_IsPolygonCW

`ST_IsPolygonCW` — Returns true if all exterior rings are oriented clockwise and all interior rings are oriented counter-clockwise.

**Synopsis**

```
boolean ST_IsPolygonCW ( geometry geom );
```

**Descrição**

Returns true if all polygonal components of the input geometry use a clockwise orientation for their exterior ring, and a counter-clockwise direction for all interior rings.

Returns true if the geometry has no polygonal components.

**Note**

Closed linestrings are not considered polygonal components, so you would still get a true return by passing a single closed linestring no matter its orientation.

**Note**

If a polygonal geometry does not use reversed orientation for interior rings (i.e., if one or more interior rings are oriented in the same direction as an exterior ring) then both `ST_IsPolygonCW` and `ST_IsPolygonCCW` will return false.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

**Veja também**

[ST\\_ForcePolygonCW](#) , [ST\\_ForcePolygonCCW](#) , [ST\\_IsPolygonCW](#)

**8.5.14 ST\_IsClosed**

`ST_IsClosed` — Retorna `VERDADEIRO` se os pontos de começo e fim da `LINESTRING` são coincidentes. Para superfície poliédrica está fechada (volumétrica).

**Synopsis**

booleana `ST_IsClosed`(geometria g);

**Descrição**

Retorna `VERDADEIRO` se os pontos de começo e fim da `LINESTRING` são coincidentes. Para superfícies poliédricas, isso lhe diz se a superfície é territorial (aberta) ou volumétrica (fechada).



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 7.1.5, 9.3.3

**Note**

SQL-MM define o resultado do `ST_IsClosed` (NULO) para ser 0, enquanto o PostGIS retorna NULO.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Melhorias: 2.0.0 suporte a superfícies poliédricas foi introduzido.



This function supports Polyhedral surfaces.

**Exemplos de line string e ponto**

```
postgis=# SELECT ST_IsClosed('LINESTRING(0 0, 1 1)::geometry);
 st_isclosed
-----
 f
(1 row)

postgis=# SELECT ST_IsClosed('LINESTRING(0 0, 0 1, 1 1, 0 0)::geometry);
 st_isclosed
-----
 t
(1 row)

postgis=# SELECT ST_IsClosed('MULTILINESTRING((0 0, 0 1, 1 1, 0 0),(0 0, 1 1))::geometry);
 st_isclosed
-----
```

```
f
(1 row)

postgis=# SELECT ST_IsClosed('POINT(0 0)::geometry);
 st_isclosed
-----
t
(1 row)

postgis=# SELECT ST_IsClosed('MULTIPOINT((0 0), (1 1))::geometry);
 st_isclosed
-----
t
(1 row)
```

### Exemplos de Superfície Poliedral

```
-- A cube --
SELECT ST_IsClosed(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 ←
1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0) ←
),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1) ←
) )'));

 st_isclosed
-----
t

-- Same as cube but missing a side --
SELECT ST_IsClosed(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 ←
0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0) ←
),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)) )'));

 st_isclosed
-----
f
```

### Veja também

[ST\\_IsRing](#)

### 8.5.15 ST\_IsCollection

**ST\_IsCollection** — Retorna VERDADEIRO se o argumento é uma coleção (MULTI\*, GEOMETRYCOLLECTION, ...)

#### Synopsis

booleana **ST\_IsCollection**(geometria g);

## Descrição

Retorna VERDADEIRO se o tipo da geometria do argumento é:

- COLEÇÃO DE GEOMETRIA
- MULTI{PONTO, POLÍGONO, LINESTRING, CURVA, SUPERFÍCIE}
- CURVA COMPOSTA



### Note

Essa função analisa o tipo da geometria. Isso significa que vai retornar VERDADEIRO nas coleções que são vazias ou que contêm apenas um elemento.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```

postgis=# SELECT ST_IsCollection('LINESTRING(0 0, 1 1)::geometry);
 st_iscollection
-----
 f
(1 row)

postgis=# SELECT ST_IsCollection('MULTIPOINT EMPTY)::geometry);
 st_iscollection
-----
 t
(1 row)

postgis=# SELECT ST_IsCollection('MULTIPOINT((0 0))::geometry);
 st_iscollection
-----
 t
(1 row)

postgis=# SELECT ST_IsCollection('MULTIPOINT((0 0), (42 42))::geometry);
 st_iscollection
-----
 t
(1 row)

postgis=# SELECT ST_IsCollection('GEOMETRYCOLLECTION(POINT(0 0))::geometry);
 st_iscollection
-----
 t
(1 row)

```

## Veja também

[ST\\_NumGeometries](#)

## 8.5.16 ST\_IsEmpty

ST\_IsEmpty — Retorna verdadeiro se essa geometria é uma coleção vazia, polígono, ponto etc.

### Synopsis

booleana **ST\_IsEmpty**(geometria geomA);

### Descrição

Retorna verdadeiro se essa geometria se é vazia. Se verdadeira, ela representa uma coleção vazia, polígono, ponto etc.



#### Note

SQL-MM define o resultado da ST\_IsEmpty(NULA) para ser 0, enquanto o PostGIS retorna NULO.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.1



This method implements the SQL/MM specification. SQL-MM 3: 5.1.7



This method supports Circular Strings and Curves



#### Warning

Alterações: 2.0.0 Nas versões anteriores do PostGIS ST\_GeomFromText('GEOMETRYCOLLECTION(EMPTY)') era permitido. Agora isso é ilegal no PostGIS 2.0.0 para se adequar aos padrões SQL/MM.

### Exemplos

```
SELECT ST_IsEmpty(ST_GeomFromText('GEOMETRYCOLLECTION EMPTY'));
 st_isempty
-----
 t
(1 row)

SELECT ST_IsEmpty(ST_GeomFromText('POLYGON EMPTY'));
 st_isempty
-----
 t
(1 row)

SELECT ST_IsEmpty(ST_GeomFromText('POLYGON((1 2, 3 4, 5 6, 1 2))'));

 st_isempty
-----
 f
(1 row)

SELECT ST_IsEmpty(ST_GeomFromText('POLYGON((1 2, 3 4, 5 6, 1 2))')) = false;
?column?
-----
 t
```



```
(1 row)

SELECT ST_IsEmpty(ST_GeomFromText('CIRCULARSTRING EMPTY'));
   st_isempty
-----
t
(1 row)
```

## 8.5.17 ST\_IsRing

**ST\_IsRing** — Retorna VERDADEIRO se essa LINESTRING for fechada e simples.

### Synopsis

booleana **ST\_IsRing**(geometria g);

### Descrição

Retorna VERDADEIRO se essa LINESTRING for **ST\_IsClosed** ( $ST\_StartPoint((g)) \sim ST\_EndPoint((g))$ ) e **ST\_IsSimple** (não cruzar consigo mesma).



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). 2.1.5.1



This method implements the SQL/MM specification. SQL-MM 3: 7.1.6



### Note

SQL-MM define o resultado do **ST\_IsRing** (NULO) para ser 0, enquanto o PostGIS retorna NULO.

### Exemplos

```
SELECT ST_IsRing(the_geom), ST_IsClosed(the_geom), ST_IsSimple(the_geom)
FROM (SELECT 'LINESTRING(0 0, 0 1, 1 1, 1 0, 0 0)::geometry AS the_geom) AS foo;
   st_isring | st_isclosed | st_issimple
-----+-----+-----
t           | t           | t
(1 row)

SELECT ST_IsRing(the_geom), ST_IsClosed(the_geom), ST_IsSimple(the_geom)
FROM (SELECT 'LINESTRING(0 0, 0 1, 1 0, 1 1, 0 0)::geometry AS the_geom) AS foo;
   st_isring | st_isclosed | st_issimple
-----+-----+-----
f           | t           | f
(1 row)
```

### Veja também

[ST\\_IsClosed](#), [ST\\_IsSimple](#), [ST\\_StartPoint](#), [ST\\_EndPoint](#)

## 8.5.18 ST\_IsSimple

`ST_IsSimple` — Retorna (VERDADEIRA) se essa geometria não tem nenhum ponto irregular, como auto intersecção ou tangenciação.

### Synopsis

booleana `ST_IsSimple`(geometria geomA);

### Descrição

Retorna verdadeira se essa geometria não tem nenhum ponto geométrico irregular, como auto intersecção ou tangenciação. Para maiores informações na definição OGC da simplicidade e validade das geometrias, use "[Ensuring OpenGIS compliancy of geometries](#)"



#### Note

SQL-MM define o resultado da `ST_IsSimple(NULA)` para ser 0, enquanto o PostGIS retorna NULO.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.1



This method implements the SQL/MM specification. SQL-MM 3: 5.1.8



This function supports 3d and will not drop the z-index.

### Exemplos

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

```
st_issimple
```

```
-----
```

```
t
```

```
(1 row)
```

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

```
st_issimple
```

```
-----
```

```
f
```

```
(1 row)
```

### Veja também

[ST\\_IsValid](#)

### 8.5.19 ST\_IsValid

ST\_IsValid — Retorna verdadeira se a ST\_Geometry é bem formada.

#### Synopsis

```
booleana ST_IsValid(geometria g);
booleana ST_IsValid(geometria g, inteiro flags);
```

#### Descrição

Testa se um valor ST\_Geometry é bem formado. Para geometrias que são inválidas, o PostgreSQL NOTICE irá fornecer detalhes da razão de não serem válidas. Para maiores informações na definição OGC da simplicidade e validade de geometrias, use "[Ensuring OpenGIS compliancy of geometries](#)"



#### Note

SQL-MM define o resultado da ST\_IsValid(NULA) para ser 0, enquanto o PostGIS retorna NULO.

A versão que aceita bandeiras está disponível com a 2.0.0 e requer GEOS >= 3.3.0. Tal versão não imprime um AVISO explicando a invalidade. As bandeiras permitidas estão documentadas em [ST\\_IsValidDetail](#).



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 5.1.9



#### Note

Nem as especificações do OGC-SFS ou do SQL-MM incluem um argumento bandeira para ST\_IsValid. A bandeira é uma extensão do PostGIS.

#### Exemplos

```
SELECT ST_IsValid(ST_GeomFromText('LINESTRING(0 0, 1 1)')) As good_line,
       ST_IsValid(ST_GeomFromText('POLYGON((0 0, 1 1, 1 2, 1 1, 0 0))')) As bad_poly
--results
NOTICE: Self-intersection at or near point 0 0
good_line | bad_poly
-----+-----
t         | f
```

#### Veja também

[ST\\_IsSimple](#), [ST\\_IsValidReason](#), [ST\\_IsValidDetail](#), [ST\\_Summary](#)

### 8.5.20 ST\_IsValidReason

ST\_IsValidReason — Retorna texto declarando se uma geometria é válida ou não e se não for válida, uma razão do porquê.

## Synopsis

texto **ST\_IsValidReason**(geometria geomA);  
 text **ST\_IsValidReason**(geometria geomA, inteiro flags);

## Descrição

Retorna texto declarando se uma geometria é válida ou não e se não for válida, uma razão do porquê.

Vantajoso em combinação com **ST\_IsValid** para gerar um relato detalhado das geometrias inválidas e seus motivos.

As bandeiras permitidas estão documentadas em **ST\_IsValidDetail**.

Disponibilidade: 1.4 - requer GEOS >= 3.1.0.

Disponibilidade: 2.0 - requer GEOS >= 3.3.0 para a versão pegando bandeiras.

## Exemplos

```
--First 3 Rejects from a successful quintuplet experiment
SELECT gid, ST_IsValidReason(the_geom) as validity_info
FROM
(SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), ST_Accum(f.line)) As the_geom, gid
FROM (SELECT ST_Buffer(ST_MakePoint(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_MakePoint(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(the_geom) = false
ORDER BY gid
LIMIT 3;
```

| gid  | validity_info            |
|------|--------------------------|
| 5330 | Self-intersection [32 5] |
| 5340 | Self-intersection [42 5] |
| 5350 | Self-intersection [52 5] |

```
--simple example
SELECT ST_IsValidReason('LINESTRING(220227 150406,2220227 150407,222020 150410)');
```

| st_isvalidreason |
|------------------|
| Valid Geometry   |

## Veja também

[ST\\_IsValid](#), [ST\\_Summary](#)

### 8.5.21 ST\_IsValidDetail

`ST_IsValidDetail` — Retorna uma fila `valid_detail` (válida, motivo, localização) se uma geometria é válida ou não e, se não for, uma razão do porquê e uma localização.

#### Synopsis

```
valid_detail ST_IsValidDetail(geometry geom);
valid_detail ST_IsValidDetail(geometria geom, inteiro flags);
```

#### Descrição

Retorna uma linha `valid_detail`, formada por uma declaração booleana (válida) se uma geometria é válida, uma declaração `varchar` (motivo) uma razão de porquê é inválida e uma geometria (localização) pontuando onde ela é inválida.

Vantajoso para substituir e melhorar a combinação de `ST_IsValid` e `ST_IsValidReason` para gerar um relato detalhado de geometrias inválidas.

O argumento das "bandeiras" é um bitfield. Não pode ter os seguintes valores:

- 1: Considere anéis que tem auto intersecção formando buracos como válidos. Isso também pode ser conhecido como "a bandeira ESRI". Note que isso vai contra o modelo OGC.

Disponibilidade: 2.0.0 - requer GEOS >= 3.3.0.

#### Exemplos

```
--First 3 Rejects from a successful quintuplet experiment
SELECT gid, reason(ST_IsValidDetail(the_geom)), ST_AsText(location(ST_IsValidDetail(↵
    the_geom))) as location
FROM
(SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), ST_Accum(f.line)) As the_geom, gid
FROM (SELECT ST_Buffer(ST_MakePoint(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_MakePoint(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(the_geom) = false
ORDER BY gid
LIMIT 3;
```

| gid  | reason            | location    |
|------|-------------------|-------------|
| 5330 | Self-intersection | POINT(32 5) |
| 5340 | Self-intersection | POINT(42 5) |
| 5350 | Self-intersection | POINT(52 5) |

```
--simple example
SELECT * FROM ST_IsValidDetail('LINESTRING(220227 150406,220227 150407,22020 150410)');
```

```

valid | reason | location
-----+-----+-----
t      |         |

```

**Veja também**

[ST\\_IsValid](#), [ST\\_IsValidReason](#)

**8.5.22 ST\_M**

**ST\_M** — Retorna a coordenada M do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.

**Synopsis**

```
float ST_M(geometria a_point);
```

**Descrição**

Retorna a coordenada M do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.

**Note**

Isso não faz parte (ainda) do OGC spec, mas está listado aqui para completar a função lista do ponto coordenado extrator.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification.



This function supports 3d and will not drop the z-index.

**Exemplos**

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

```
st_m
```

```
-----
```

```
4
```

```
(1 row)
```

**Veja também**

[ST\\_GeomFromEWKT](#), [ST\\_X](#), [ST\\_Y](#), [ST\\_Z](#)

### 8.5.23 ST\_NDims

ST\_NDims — Retorna a dimensão coordenada da geometria como uma small int. Os valores são: 2, 3 ou 4.

#### Synopsis

integer **ST\_NDims**(geometria g1);

#### Descrição

Retorna a dimensão coordenada da geometria. O PostGIS suporta 2 - (x,y) , 3 - (x,y,z) ou 2D com medida - x,y,m, e 4 - 3D com espaço de medida x,y,z,m



This function supports 3d and will not drop the z-index.

#### Exemplos

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

```

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

#### Veja também

[ST\\_CoordDim](#), [ST\\_Dimension](#), [ST\\_GeomFromEWKT](#)

### 8.5.24 ST\_NPoints

ST\_NPoints — Retorna o número de pontos (vértices) em uma geometria.

#### Synopsis

inteiro **ST\_NPoints**(geometria g1);

#### Descrição

Retorna o número de pontos em uma geometria. Funciona para todas as geometrias.

Melhorias: 2.0.0 suporte a superfícies poliédricas foi introduzido.



#### Note

Anteriores a 1.3.4, essa função falha se usada com geometrias que contêm CURVAS. Isso é consertado em 1.3.4+



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

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

--Polygon in 3D space
SELECT ST_NPoints(ST_GeomFromEWKT('LINESTRING(77.29 29.07 1,77.42 29.26 0,77.27 29.31
  -1,77.29 29.07 3)'));
--result
4
```

## Veja também

[ST\\_NumPoints](#)

### 8.5.25 ST\_NRings

ST\_NRings — Se a geometria for um polígono ou multi polígono, retorna o número de anéis.

#### Synopsis

inteiro ST\_NRings(geometria geomA);

#### Descrição

Se a geometria for um polígono ou multi polígono, retorna o número de anéis. Diferente do NumInteriorRings, esse conta os anéis de fora também.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

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

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

## Veja também

[ST\\_NumInteriorRings](#)

### 8.5.26 ST\_NumGeometries

ST\_NumGeometries — Se a geometria é uma GEOMETRYCOLLECTION (ou MULTI\*), retorna o número de geometria, para geometrias únicas retornará 1, senão retorna NULO.



**Synopsis**

inteiro **ST\_NumGeometries**(geometria geom);

**Descrição**

Retorna o número de geometrias. Se a geometria é uma **GEOMETRYCOLLECTION** (ou **MULTI\***), retorna o número de geometria, para geometrias únicas retornará 1, senão retorna **NULO**.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TIN introduzido.

Alterações: 2.0.0 Em versões anteriores retornaria **NULO** se a geometria não fosse do tipo coleção/**MULTI**. 2.0.0+ agora retorna 1 para geometrias únicas ex: **POLÍGONO**, **LINestring**, **PONTO**.



This method implements the SQL/MM specification. SQL-MM 3: 9.1.4



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

**Exemplos**

```
--Prior versions would have returned NULL for this -- in 2.0.0 this returns 1
SELECT ST_NumGeometries(ST_GeomFromText('LINestring(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
--result
1

--Geometry Collection Example - multis count as one geom in a collection
SELECT ST_NumGeometries(ST_GeomFromEWKT('GEOMETRYCOLLECTION(MULTIPOINT(-2 3 , -2 2),
LINestring(5 5 ,10 10),
POLYGON((-7 4.2,-7.1 5,-7.1 4.3,-7 4.2))'));
--result
3
```

**Veja também**

[ST\\_GeometryN](#), [ST\\_Multi](#)

**8.5.27 ST\_NumInteriorRings**

**ST\_NumInteriorRings** — Retorna o número de anéis interiores de um polígono.

**Synopsis**

inteiro **ST\_NumInteriorRings**(geometria a\_polygon);

## Descrição

Retorna o número de anéis interiores de um polígono. Retorna NULO se a geometria não for um polígono.



This method implements the SQL/MM specification. SQL-MM 3: 8.2.5

Alterações: 2.0.0 - nas versões anteriores isso permitiria um MULTIPOLÍGONO, retornando o número de anéis interiores do primeiro POLÍGONO.

## Exemplos

```
--If you have a regular polygon
SELECT gid, field1, field2, ST_NumInteriorRings(the_geom) AS numholes
FROM sometable;

--If you have multipolygons
--And you want to know the total number of interior rings in the MULTIPOLYGON
SELECT gid, field1, field2, SUM(ST_NumInteriorRings(the_geom)) AS numholes
FROM (SELECT gid, field1, field2, (ST_Dump(the_geom)).geom As the_geom
      FROM sometable) As foo
GROUP BY gid, field1,field2;
```

## Veja também

[ST\\_NumInteriorRing](#)

### 8.5.28 ST\_NumInteriorRing

ST\_NumInteriorRing — Retorna o número de anéis interiores de um polígono na geometria. Sinônimo para ST\_NumInteriorRings.

#### Synopsis

inteiro **ST\_NumInteriorRing**(geometria a\_polygon);

## Veja também

[ST\\_NumInteriorRings](#)

### 8.5.29 ST\_NumPatches

ST\_NumPatches — Retorna o número de faces em uma superfícies poliédrica. Retornará nulo para geometrias não poliédricas.

#### Synopsis

inteiro **ST\_NumPatches**(geometria g1);

## Descrição

Retorna o número de faces em uma superfície poliédrica. Retornará nulo para geometrias não poliédricas. Isso é um heterônimo para `ST_NumGeometries` para suportar a nomeação MM. É mais rápido utilizar `ST_NumGeometries` se você não se importa com a convenção MM.

Disponibilidade: 2.0.0



This function supports 3d and will not drop the z-index.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: ?



This function supports Polyhedral surfaces.

## Exemplos

```
SELECT ST_NumPatches(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 ←
  0)),
    ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0) ←
    ),
    ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
    ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1) ←
    ) )');
--result
6
```

## Veja também

[ST\\_GeomFromEWKT](#), [ST\\_NumGeometries](#)

### 8.5.30 ST\_NumPoints

`ST_NumPoints` — Retorna o número de pontos em um valor `ST_LineString` ou `ST_CircularString`.

#### Synopsis

inteiro `ST_NumPoints`(geometria g1);

#### Descrição

Retorna o número de pontos em um valor `ST_LineString` ou `ST_CircularString`. Anteriores a 1.4 só funcionam com `LineStrings` como as specs declaram. A partir de 1.4 isso é um heterônimo para `ST_NPoints`, que retorna o número de vértices apenas para as `line strings`. Considere utilizar `ST_NPoints` que tem vários objetivos e funciona com vários tipos de geometrias.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 7.2.4

## Exemplos

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

## Veja também

[ST\\_NPoints](#)

### 8.5.31 ST\_PatchN

**ST\_PatchN** — Retorna a geometria (face) de 1-base Nth se a geometria é uma POLYHEDRALSURFACE, POLYHEDRALSURFACEM. Senão, retorna NULA.

#### Synopsis

geometria **ST\_PatchN**(geometria geomA, inteiro n);

#### Descrição

> Retorna a geometria (face) de 1-base Nth se a geometria é POLYHEDRALSURFACE, POLYHEDRALSURFACEM. Senão, retorna NULA. Retorna a mesma resposta como ST\_GeometryN para superfícies poliédricas. Utilizar ST\_GeometryN é mais rápido.



**Note**  
Index é 1-base.



**Note**  
Se você quiser extrair todas as geometrias, de uma geometria, ST\_Dump é mais eficiente.

Disponibilidade: 2.0.0



This method implements the SQL/MM specification. SQL-MM 3: ?



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

## Exemplos

```
--Extract the 2nd face of the polyhedral surface
SELECT ST_AsEWKT(ST_PatchN(geom, 2)) As geomewkt
FROM (
VALUES (ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )') ) ←
      As foo(geom);

      geomewkt
-----+-----
POLYGON((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0))
```

## Veja também

[ST\\_AsEWKT](#), [ST\\_GeomFromEWKT](#), [ST\\_Dump](#), [ST\\_GeometryN](#), [ST\\_NumGeometries](#)

### 8.5.32 ST\_PointN

**ST\_PointN** — Retorna o ponto Nth na primeira linestring ou linestring circular na geometria. Valores negativos são contados tardiamente do fim da linestring. Retorna NULA se não há uma linestring na geometria.

#### Synopsis

geometria **ST\_PointN**(geometria a\_linestring, inteiro n);

#### Descrição

Retorna o ponto Nth na primeira linestring ou linestring circular na geometria. Valores negativos são contados tardiamente do fim da linestring, tornando o ponto -1 o último ponto. Retorna NULA se não há uma linestring na geometria.



#### Note

O Index é 1-base como para OGC specs desde a versão 0.8.0. Indexing atrasado (negativo) não está nas versões OGC anteriores implementadas com 0-base.



#### Note

Se você quiser o ponto nth de cada line string em uma multilinestring, utilize em conjunção com [ST\\_Dump](#)



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 7.2.5, 7.3.5



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

**Note**

Alterações: 2.0.0 não funciona mais com geometrias multilinestrings únicas. Em verões mais antigas do PostGIS -- uma única linha multilinestring trabalharia normalmente e retornaria o ponto inicial. Na 2.0.0 só retorna NULA como qualquer outra multilinestring.

Alterações: 2.3.0 : indexing negativo disponível (-1 é o último ponto)

**Exemplos**

```
-- Extract all POINTs from a LINESTRING
SELECT ST_AsText(
  ST_PointN(
    column1,
    generate_series(1, ST_NPoints(column1))
  ))
FROM ( VALUES ('LINESTRING(0 0, 1 1, 2 2)::geometry) ) AS foo;

st_astext
-----
POINT(0 0)
POINT(1 1)
POINT(2 2)
(3 rows)

--Example circular string
SELECT ST_AsText(ST_PointN(ST_GeomFromText('CIRCULARSTRING(1 2, 3 2, 1 2)'),2));

st_astext
-----
POINT(3 2)

SELECT st_astext(f)
FROM ST_GeometryFromtext('LINESTRING(0 0 0, 1 1 1, 2 2 2)') as g
      ,ST_PointN(g, -2) AS f -- 1 based index

st_astext
-----
"POINT Z (1 1 1)"
```

**Veja também**

[ST\\_NPoints](#)

**8.5.33 ST\_Points**

**ST\_Points** — Retorna uma multilinestring contendo todas as coordenadas de uma geometria.

**Synopsis**

geometria **ST\_Points**( geometria geom );

**Descrição**

Retorna uma multilinestring contendo todas as coordenadas de uma geometria. Não remove pontos que são duplicados na entrada da geometria, incluindo pontos iniciais e finais de geometrias de anéis. (Se esse comportamento não é desejável, duplicadas podem ser removidas utilizando [ST\\_RemoveRepeatedPoints](#)).

As ordenadas serão preservadas, se existentes.



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.

Disponibilidade: 2.3.0

**Exemplos**

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

**Veja também**

[ST\\_RemoveRepeatedPoints](#)

**8.5.34 ST\_SRID**

**ST\_SRID** — Retorna o identificador de referência espacial para a *ST\_Geometry* como definido na table *spatial\_ref\_sys*.

**Synopsis**

inteiro **ST\_SRID**(geometria g1);

**Descrição**

Retorna o identificador de referência espacial para a *ST\_Geometry* como definido na table *spatial\_ref\_sys*. Section [4.3.1](#)

**Note**

A table *spatial\_ref\_sys* cataloga todos os sistemas de referência espacial conhecidos pelo PostGIS e é usada para transformações de um sistema de referência espacial para outro. Portanto, verificar que você possui o identificador de sistema de referência espacial é importante se planeja transformar suas geometrias em algum momento.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.1



This method implements the SQL/MM specification. SQL-MM 3: 5.1.5



This method supports Circular Strings and Curves

## Exemplos

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)',4326));
--result
4326
```

## Veja também

Section [4.3.1](#), [ST\\_GeomFromText](#), [ST\\_SetSRID](#), [ST\\_Transform](#)

### 8.5.35 ST\_StartPoint

`ST_StartPoint` — Retorna ao último ponto de uma `LINestring` geometria como um `PONTO`.

#### Synopsis

geometria `ST_StartPoint`(geometria geomA);

#### Descrição

Retorna ao último ponto de uma `LINestring` ou `CIRCULARLINestring` geometria como um `PONTO` ou `NULO` se o parâmetro de entrada não é uma `LINestring`.



This method implements the SQL/MM specification. SQL-MM 3: 7.1.3



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

#### Note



Alterações: 2.0.0 não funciona mais com geometrias de multilinesstrings. Em versões mais antigas do PostGIS -- uma linha multilinesstring sozinha trabalharia normalmente com essa função e voltaria o ponto de início. Na 2.0.0 ela retorna `NULA` como qualquer outra multilinesstring. O antigo comportamento não foi uma característica documentada, mas as pessoas que consideravam que tinham seus dados armazenados como uma `LINestring`, agora podem experimentar essas que retornam `NULAS` em 2.0.

## Exemplos

```
SELECT ST_AsText(ST_StartPoint('LINestring(0 1, 0 2)::geometry'));
 st_astext
-----
POINT(0 1)
(1 row)

SELECT ST_StartPoint('POINT(0 1)::geometry') IS NULL AS is_null;
 is_null
-----
t
(1 row)
```



```
--3d line
SELECT ST_AsEWKT(ST_StartPoint('LINESTRING(0 1 1, 0 2 2)')::geometry);
  st_asewkt
-----
POINT(0 1 1)
(1 row)

-- circular linestring --
SELECT ST_AsText(ST_StartPoint('CIRCULARSTRING(5 2,-3 1.999999, -2 1, -4 2, 5 2)')::geometry ←
  ));
  st_astext
-----
POINT(5 2)
```

### Veja também

[ST\\_EndPoint](#), [ST\\_PointN](#)

## 8.5.36 ST\_Summary

**ST\_Summary** — Retorna um texto resumo dos conteúdos da geometria.

### Synopsis

```
texto ST_Summary(geometria g);
text ST_Summary(geografia g);
```

### Descrição

Retorna um texto resumo dos conteúdos da geometria.

As bandeiras mostraram colchetes depois do tipo de geometria ter o seguinte significado:

- M: tem ordenada M
- Z: tem ordenada Z
- B: tem uma caixa limitante salva
- G: é geodésico (geografia)
- S: tem um sistema de referência espacial



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Disponibilidade: 1.2.2

Melhorias: 2.0.0 suporte para geografia adicionado

melhorias: 2.1.0 Bandeira S para indicar se existe um sistema de referência espacial conhecido

Melhorias: 2.2.0 Suporte para TIN e Curvas adicionado

## Exemplos

```

=# SELECT ST_Summary(ST_GeomFromText('LINESTRING(0 0, 1 1)')) as geom,
           ST_Summary(ST_GeogFromText('POLYGON((0 0, 1 1, 1 2, 1 1, 0 0))')) geog;
-----+-----
geom | geog
-----+-----
LineString[B] with 2 points | Polygon[BGS] with 1 rings
                               | ring 0 has 5 points
                               :
(1 row)

=# SELECT ST_Summary(ST_GeogFromText('LINESTRING(0 0 1, 1 1 1)')) As geog_line,
           ST_Summary(ST_GeomFromText('SRID=4326;POLYGON((0 0 1, 1 1 2, 1 2 3, 1 1 1, 0 0 1)) ←
           ') As geom_poly;
;
-----+-----
geog_line | geom_poly
-----+-----
LineString[ZBGS] with 2 points | Polygon[ZBS] with 1 rings
                               :   ring 0 has 5 points
                               :
(1 row)

```

## Veja também

[PostGIS\\_DropBBox](#), [PostGIS\\_AddBBox](#), [ST\\_Force3DM](#), [ST\\_Force3DZ](#), [ST\\_Force2D](#), [geografia](#)  
[ST\\_IsValid](#), [ST\\_IsValid](#), [ST\\_IsValidReason](#), [ST\\_IsValidDetail](#)

## 8.5.37 ST\_X

**ST\_X** — Retorna a coordenada X do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.

### Synopsis

```
float ST_X(geometria a_point);
```

### Descrição

Retorna a coordenada X do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.



#### Note

Se você quer valores máximos e mínimos de x de qualquer geometria, veja nas funções [ST\\_XMin](#), [ST\\_XMax](#).



This method implements the SQL/MM specification. SQL-MM 3: 6.1.3



This function supports 3d and will not drop the z-index.

## Exemplos

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

```
st_x
```

```
-----
```

```
1
```

```
(1 row)
```

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

```
st_y
```

```
-----
```

```
1.5
```

```
(1 row)
```

## Veja também

[ST\\_Centroid](#), [ST\\_GeomFromEWKT](#), [ST\\_M](#), [ST\\_XMax](#), [ST\\_XMin](#), [ST\\_Y](#), [ST\\_Z](#)

### 8.5.38 ST\_XMax

**ST\_XMax** — Retorna o X máximo de uma caixa limitante 2d ou 3d ou uma geometria.

#### Synopsis

```
float ST_XMax(box3d aGeomorBox2DorBox3D);
```

#### Descrição

Retorna o X máximo de uma caixa limitante 2d ou 3d ou uma geometria.



#### Note

Embora essa função só seja definida para caixa3d, ela funcionará para caixa2d e geometria, por causa do comportamento auto-casting definido para geometria e caixa2d. Entretanto você não pode abastecer uma geometria ou uma representação de texto de uma caixa2d com ele, já que não irá auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
SELECT ST_XMax('BOX3D(1 2 3, 4 5 6)');
st_xmax
-----
4

SELECT ST_XMax(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_xmax
-----
5

SELECT ST_XMax(CAST('BOX(-3 2, 3 4)' As box2d));
st_xmax
-----
3
--Observe THIS DOES NOT WORK because it will try to autocast the string representation to a ←
BOX3D
SELECT ST_XMax('LINESTRING(1 3, 5 6)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_XMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ←
150406 3)'));
st_xmax
-----
220288.248780547
```

## Veja também

[ST\\_XMin](#), [ST\\_YMax](#), [ST\\_YMin](#), [ST\\_ZMax](#), [ST\\_ZMin](#)

### 8.5.39 ST\_XMin

**ST\_XMin** — Retorna o X mínimo de uma caixa limitante 2d ou 3d ou uma geometria.

#### Synopsis

```
float ST_XMin(box3d aGeomorBox2DorBox3D);
```

#### Descrição

Retorna o X mínimo de uma caixa limitante 2d ou 3d ou uma geometria.



#### Note

Embora essa função só seja definida para caixa3d, ela funcionará para caixa2d e geometria, por causa do comportamento auto-casting definido para geometria e caixa2d. Entretanto você não pode abastecer uma geometria ou uma representação de texto de uma caixa2d com ele, já que não irá auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```

SELECT ST_XMin('BOX3D(1 2 3, 4 5 6)');
st_xmin
-----
1

SELECT ST_XMin(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_xmin
-----
1

SELECT ST_XMin(CAST('BOX(-3 2, 3 4)' As box2d));
st_xmin
-----
-3
--Observe THIS DOES NOT WORK because it will try to autocast the string representation to a ←
  BOX3D
SELECT ST_XMin('LINESTRING(1 3, 5 6)');

--ERROR:  BOX3D parser - doesn't start with BOX3D(

SELECT ST_XMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ←
  150406 3)'));
st_xmin
-----
220186.995121892

```

## Veja também

[ST\\_XMax](#), [ST\\_YMax](#), [ST\\_YMin](#), [ST\\_ZMax](#), [ST\\_ZMin](#)

### 8.5.40 ST\_Y

ST\_Y — Retorna a coordenada Y do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.

#### Synopsis

```
float ST_Y(geometria a_point);
```

#### Descrição

Retorna a coordenada Y do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 6.1.4



This function supports 3d and will not drop the z-index.

## Exemplos

```
SELECT ST_Y(ST_GeomFromEWKT('POINT(1 2 3 4)'));
st_y
-----
      2
(1 row)

SELECT ST_Y(ST_Centroid(ST_GeomFromEWKT('LINESTRING(1 2 3 4, 1 1 1 1)')));
st_y
-----
  1.5
(1 row)
```

## Veja também

[ST\\_Centroid](#), [ST\\_GeomFromEWKT](#), [ST\\_M](#), [ST\\_X](#), [ST\\_YMax](#), [ST\\_YMin](#), [ST\\_Z](#)

### 8.5.41 ST\_YMax

**ST\_YMax** — Retorna o Y máximo de uma caixa limitante 2d ou 3d ou uma geometria.

#### Synopsis

```
float ST_YMax(box3d aGeomorBox2DorBox3D);
```

#### Descrição

Retorna o Y máximo de uma caixa limitante 2d ou 3d ou uma geometria.



#### Note

Embora essa função só seja definida para caixa3d, ela funcionará para caixa2d e geometria, por causa do comportamento auto-casting definido para geometria e caixa2d. Entretanto você não pode abastecer uma geometria ou uma representação de texto de uma caixa2d com ele, já que não irá auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
SELECT ST_YMax('BOX3D(1 2 3, 4 5 6)');
st_ymax
-----
      5

SELECT ST_YMax(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_ymax
-----
```

```

6
SELECT ST_YMax(CAST('BOX(-3 2, 3 4)' As box2d));
st_ymax
-----
4
--Observe THIS DOES NOT WORK because it will try to autocast the string representation to a ←
BOX3D
SELECT ST_YMax('LINESTRING(1 3, 5 6)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_YMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ←
150406 3)'));
st_ymax
-----
150506.126829327

```

### Veja também

[ST\\_XMin](#), [ST\\_XMax](#), [ST\\_YMin](#), [ST\\_ZMax](#), [ST\\_ZMin](#)

## 8.5.42 ST\_YMin

ST\_YMin — Retorna o Y mínimo de uma caixa limitante 2d ou 3d ou uma geometria.

### Synopsis

float ST\_YMin(box3d aGeomorBox2DorBox3D);

### Descrição

Retorna o Y mínimo de uma caixa limitante 2d ou 3d ou uma geometria.



#### Note

Embora essa função só seja definida para caixa3d, ela funcionará para caixa2d e geometria, por causa do comportamento auto-casting definido para geometria e caixa2d. Entretanto você não pode abastecer uma geometria ou uma representação de texto de uma caixa2d com ele, já que não irá auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

### Exemplos

```

SELECT ST_YMin('BOX3D(1 2 3, 4 5 6)');
st_ymin
-----
2

SELECT ST_YMin(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));

```

```

st_ymin
-----
3

SELECT ST_YMin(CAST('BOX(-3 2, 3 4)' As box2d));
st_ymin
-----
2
--Observe THIS DOES NOT WORK because it will try to autocast the string representation to a ←
BOX3D
SELECT ST_YMin('LINESTRING(1 3, 5 6)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_YMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ←
150406 3)'));
st_ymin
-----
150406

```

**Veja também**

[ST\\_GeomFromEWKT](#), [ST\\_XMin](#), [ST\\_XMax](#), [ST\\_YMax](#), [ST\\_ZMax](#), [ST\\_ZMin](#)

**8.5.43 ST\_Z**

**ST\_Z** — Retorna a coordenada Z do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.

**Synopsis**

```
float ST_Z(geometria a_point);
```

**Descrição**

Retorna a coordenada Z do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.



This method implements the SQL/MM specification.



This function supports 3d and will not drop the z-index.

**Exemplos**

```

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

st_z
-----
3

(1 row)

```



**Veja também**

[ST\\_GeomFromEWKT](#), [ST\\_M](#), [ST\\_X](#), [ST\\_Y](#), [ST\\_ZMax](#), [ST\\_ZMin](#)

**8.5.44 ST\_ZMax**

`ST_ZMax` — Retorna o Z mínimo de uma caixa limitante 2d ou 3d ou uma geometria.

**Synopsis**

```
float ST_ZMax(box3d aGeomorBox2DorBox3D);
```

**Descrição**

Retorna o Z máximo de uma caixa limitante 2d ou 3d ou uma geometria.

**Note**

Embora essa função só seja definida para caixa3d, ela funcionará para caixa2d e geometria, por causa do comportamento auto-casting definido para geometria e caixa2d. Entretanto você não pode abastecer uma geometria ou uma representação de texto de uma caixa2d com ele, já que não irá auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

**Exemplos**

```
SELECT ST_ZMax('BOX3D(1 2 3, 4 5 6)');
st_zmax
-----
6

SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING(1 3 4, 5 6 7)'));
st_zmax
-----
7

SELECT ST_ZMax('BOX3D(-3 2 1, 3 4 1)');
st_zmax
-----
1
--Observe THIS DOES NOT WORK because it will try to autocast the string representation to a ←
BOX3D
SELECT ST_ZMax('LINESTRING(1 3 4, 5 6 7)');
--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_ZMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ←
150406 3)'));
st_zmax
-----
3
```

**Veja também**

[ST\\_GeomFromEWKT](#), [ST\\_XMin](#), [ST\\_XMax](#), [ST\\_YMax](#), [ST\\_YMin](#), [ST\\_ZMax](#)

**8.5.45 ST\_Zmflag**

`ST_Zmflag` — Retorna a bandeira ZM (dimensão semântica) das geometrias como uma small int. Os valores são: 0=2d, 1=3dm, 2=3dz, 3=4d.

**Synopsis**

```
smallint ST_Zmflag(geometria geomA);
```

**Descrição**

Retorna a bandeira ZM (dimensão semântica) das geometrias como uma small int. Os valores são: 0=2d, 1=3dm, 2=3dz, 3=4d.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

**Exemplos**

```
SELECT ST_Zmflag(ST_GeomFromEWKT('LINESTRING(1 2, 3 4)'));
 st_zmflag
-----
                0

SELECT ST_Zmflag(ST_GeomFromEWKT('LINESTRINGM(1 2 3, 3 4 3)'));
 st_zmflag
-----
                1

SELECT ST_Zmflag(ST_GeomFromEWKT('CIRCULARSTRING(1 2 3, 3 4 3, 5 6 3)'));
 st_zmflag
-----
                2

SELECT ST_Zmflag(ST_GeomFromEWKT('POINT(1 2 3 4)'));
 st_zmflag
-----
                3
```

**Veja também**

[ST\\_CoordDim](#), [ST\\_NDims](#), [ST\\_Dimension](#)

**8.5.46 ST\_ZMin**

`ST_ZMin` — Retorna o Z mínimo de uma caixa limitante 2d ou 3d ou uma geometria.

**Synopsis**

```
float ST_ZMin(box3d aGeomorBox2DorBox3D);
```

## Descrição

Retorna o Z mínimo de uma caixa limitante 2d ou 3d ou uma geometria.



### Note

Embora essa função só seja definida para caixa3d, ela funcionará para caixa2d e geometria, por causa do comportamento auto-casting definido para geometria e caixa2d. Entretanto você não pode abastecer uma geometria ou uma representação de texto de uma caixa2d com ele, já que não irá auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
SELECT ST_ZMin('BOX3D(1 2 3, 4 5 6)');
st_zmin
-----
3

SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING(1 3 4, 5 6 7)'));
st_zmin
-----
4

SELECT ST_ZMin('BOX3D(-3 2 1, 3 4 1)');
st_zmin
-----
1
--Observe THIS DOES NOT WORK because it will try to autocast the string representation to a ←
BOX3D
SELECT ST_ZMin('LINESTRING(1 3 4, 5 6 7)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_ZMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 ←
150406 3)'));
st_zmin
-----
1
```

## Veja também

[ST\\_GeomFromEWKT](#), [ST\\_GeomFromText](#), [ST\\_XMin](#), [ST\\_XMax](#), [ST\\_YMax](#), [ST\\_YMin](#), [ST\\_ZMax](#)

## 8.6 Editores de geometria

### 8.6.1 ST\_AddPoint

`ST_AddPoint` — Adicione um ponto para uma `LineString`.

**Synopsis**

```
geometry ST_AddPoint(geometry linestring, geometry point);
geometry ST_AddPoint(geometry linestring, geometry point, integer position);
```

**Descrição**

Adiciona um ponto a uma LineString antes do ponto <position> (índice inicia-se em 0). O terceiro parâmetro pode ser omitido ou configurado como -1 para acrescentar ao final.

Disponibilidade: 1.1.0



This function supports 3d and will not drop the z-index.

**Exemplos**

```
--guarantee all linestrings in a table are closed
--by adding the start point of each linestring to the end of the line ↔
string
--only for those that are not closed
UPDATE sometable
SET the_geom = ST_AddPoint(the_geom, ST_StartPoint(the_geom))
FROM sometable
WHERE ST_IsClosed(the_geom) = false;

--Adding point to a 3-d line
SELECT ST_AsEWKT(ST_AddPoint(ST_GeomFromEWKT('LINESTRING(0 0 1, 1 1 1)'), ↔
    ST_MakePoint(1, 2, 3)));

--result
st_asewkt
-----
LINESTRING(0 0 1,1 1 1,1 2 3)
```

**Veja também**

[ST\\_RemovePoint](#), [ST\\_SetPoint](#)

**8.6.2 ST\_Affine**

**ST\_Affine** — Aplique uma 3a transformação afim em uma geometria.

**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);
```

**Descrição**

Aplica uma 3a transformação afim na geometria para fazer coisas como translação, rotação e escala em apenas um passo.

Versão 1: A chamada

```
ST_Affine(geom, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff)
```

representa a matriz de transformação

```
/ a b c xoff \  
| d e f yoff |  
| g h i zoff |  
\ 0 0 0 1 /
```

e os vértices são transformados como segue:

```
x' = a*x + b*y + c*z + xoff  
y' = d*x + e*y + f*z + yoff  
z' = g*x + h*y + i*z + zoff
```

Todas as funções de translação / escala estão expressadas via tal qual uma transformação afim.

Versão 2: Aplica uma 2a transformação afim para a geometria. A chamada

```
ST_Affine(geom, a, b, d, e, xoff, yoff)
```

representa a matriz de transformação

```
/ a b 0 xoff \  
| d e 0 yoff |  
| 0 0 1 0 |  
\ 0 0 0 1 /  
rsp. / a b xoff \  
| d e yoff |  
| 0 0 1 /
```

e os vértices são transformados como segue:

```
x' = a*x + b*y + xoff  
y' = d*x + e*y + yoff  
z' = z
```

Esse método é um subcaso do caso 3D acima.

Melhorias: 2.0.0 suporte para superfícies polihédricas, triângulos e TINs introduzido.

Disponibilidade: 1.1.2. Mudança de nome de Affine para ST\_Affine na versão 1.2.2.



**Note**

Antes da versão 1.3.4, esta função falha se utilizada em geometrias que continham curvas. Resolvido nas versões 1.3.4+.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
--Rotaciona uma linha 3d 180 graus no eixo z. Note que esta é uma maneira difícil de ←
  utilizar o método ST_Rotate (que teria o mesmo efeito prático)
SELECT ST_AsEWKT(ST_Affine(the_geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), 0, 0, ←
  0, 1, 0, 0, 0)) As using_affine,
ST_AsEWKT(ST_Rotate(the_geom, pi())) As using_rotate
FROM (SELECT ST_GeomFromEWKT('LINESTRING(1 2 3, 1 4 3)') As the_geom) As foo;
using_affine | using_rotate
-----+-----
LINESTRING(-1 -2 3,-1 -4 3) | LINESTRING(-1 -2 3,-1 -4 3)
(1 row)

--Rotaciona uma linha 3d em 180 graus nos eixos x e y
SELECT ST_AsEWKT(ST_Affine(the_geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), -sin( ←
  pi()), 0, sin(pi()), cos(pi()), 0, 0, 0))
FROM (SELECT ST_GeomFromEWKT('LINESTRING(1 2 3, 1 4 3)') As the_geom) As foo;
st_asewkt
-----
LINESTRING(-1 -2 -3,-1 -4 -3)
(1 row)
```

## Veja também

[ST\\_Rotate](#), [ST\\_Scale](#), [ST\\_Translate](#), [ST\\_TransScale](#)

### 8.6.3 ST\_Force2D

ST\_Force2D — Força a geometria para o modo de 2 dimensões.

#### Synopsis

```
geometry ST_Force2D(geometry geomA);
```

#### Descrição

Força a geometria a possuir apenas duas dimensões, para que todas saídas tenham apenas as coordenadas X e Y. Esta função é útil para forçar geometrias de acordo a norma OGC (a OGC apenas especifica geometrias de duas dimensões).

Melhorias: 2.0.0 suporte a superfícies polihédricas foi introduzido.

Alterado: 2.1.0. Até versão 2.0.x isto era chamado de ST\_Force\_2D.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.

## Exemplos

```
SELECT ST_AsEWKT(ST_Force2D(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5 6 ←
2)')));
          st_asewkt
-----
CIRCULARSTRING(1 1,2 3,4 5,6 7,5 6)

SELECT ST_AsEWKT(ST_Force2D('POLYGON((0 0 2,0 5 2,5 0 2,0 0 2),(1 1 2,3 1 2,1 3 2,1 1 2)) ←
'));
          st_asewkt
-----
POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))
```

## Veja também

[ST\\_Force3D](#)

### 8.6.4 ST\_Force3D

ST\_Force3D — Força a geometria para um modo XYZ. Este é um apelido para a função ST\_Force\_3DZ.

#### Synopsis

geometry **ST\_Force3D**(geometry geomA);

#### Descrição

Força a geometria a possuir 3 dimensões. Este é um apelido para a função ST\_Force\_3DZ. Se a geometria não possuir um componente Z, então uma coordenada Z de valor 0 será adicionada.

Melhorias: 2.0.0 suporte a superfícies polihédricas foi introduzido.

Alterado: 2.1.0. Até versão 2.0.x isto era chamado de ST\_Force\_3D.



This function supports Polyhedral surfaces.



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.

## Exemplos

```
--Nada acontece com uma geometria que já é 3D.

SELECT ST_AsEWKT(ST_Force3D(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5 6 ←
2)')));
          st_asewkt
-----
CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2)
```

```
SELECT ST_AsEWKT(ST_Force3D('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))'));

```

|                                                              | st_asewkt |
|--------------------------------------------------------------|-----------|
| -----                                                        | -----     |
| POLYGON((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0)) |           |

### Veja também

[ST\\_AsEWKT](#), [ST\\_Force2D](#), [ST\\_Force3DM](#), [ST\\_Force3DZ](#)

## 8.6.5 ST\_Force3DZ

ST\_Force3DZ — Força as geometrias para o modo XYZ.

### Synopsis

geometry **ST\_Force3DZ**(geometry geomA);

### Descrição

Força a geometria para o modo XYZ. Este é um sinônimo para ST\_Force3DZ. Se a geometria não possuir um componente Z, então uma coordenada Z de valor 0 será adicionada.

Melhorias: 2.0.0 suporte a superfícies polihédricas foi introduzido.

Alterado: 2.1.0. Até versão 2.0.x isto era chamado de ST\_Force\_3DZ.



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

### Exemplos

```
--Nothing happens to an already 3D geometry
SELECT ST_AsEWKT(ST_Force3DZ(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5 6 2)')));

```

|                                               | st_asewkt |
|-----------------------------------------------|-----------|
| -----                                         | -----     |
| CIRCULARSTRING(1 1 2,2 3 2,4 5 2,6 7 2,5 6 2) |           |

```
SELECT ST_AsEWKT(ST_Force3DZ('POLYGON((0 0,0 5,5 0,0 0),(1 1,3 1,1 3,1 1))'));

```

|                                                              | st_asewkt |
|--------------------------------------------------------------|-----------|
| -----                                                        | -----     |
| POLYGON((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0)) |           |

### Veja também

[ST\\_AsEWKT](#), [ST\\_Force2D](#), [ST\\_Force3DM](#), [ST\\_Force3D](#)



## 8.6.6 ST\_Force3DM

ST\_Force3DM — Força as geometrias para o modo XYM.

### Synopsis

geometry **ST\_Force3DM**(geometry geomA);

### Descrição

Força a geometria para o modo XYM. Se uma geometria não possui componente M, então uma ordenada M é associada a mesma. Se ela possui um componente Z, a ordenada Z é removida.

Alterado: 2.1.0. Até a versão 2.0.x esta função era chamada de ST\_Force\_3DM.



This method supports Circular Strings and Curves

### Exemplos

```
--Nada ocorre com uma geometria já 3D.
SELECT ST_AsEWKT(ST_Force3DM(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5
6 2)')));
          st_asewkt
-----
CIRCULARSTRINGM(1 1 0,2 3 0,4 5 0,6 7 0,5 6 0)

SELECT ST_AsEWKT(ST_Force3DM('POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))
'));
          st_asewkt
-----
POLYGONM((0 0 0,0 5 0,5 0 0,0 0 0),(1 1 0,3 1 0,1 3 0,1 1 0))
```

### Veja também

[ST\\_AsEWKT](#), [ST\\_Force2D](#), [ST\\_Force3DM](#), [ST\\_Force3D](#), [ST\\_GeomFromEWKT](#)

## 8.6.7 ST\_Force4D

ST\_Force4D — Força as geometrias para o modo XYZM.

### Synopsis

geometry **ST\_Force4D**(geometry geomA);

## Descrição

Forças as geometrias para o modo XYZM. 0 é utilizado nas componentes Z e M faltantes.

Alterado: 2.1.0. Até a versão 2.0.x esta função era chamada ST\_Force\_4D.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
--Nada ocorre com uma geometria já 4D.
SELECT ST_AsEWKT(ST_Force4D(ST_GeomFromEWKT('CIRCULARSTRING(1 1 2, 2 3 2, 4 5 2, 6 7 2, 5 6 2)')));
```

|  | st_asewkt                                               |
|--|---------------------------------------------------------|
|  | CIRCULARSTRING(1 1 2 0,2 3 2 0,4 5 2 0,6 7 2 0,5 6 2 0) |

```
SELECT ST_AsEWKT(ST_Force4D('MULTILINESTRINGM((0 0 1,0 5 2,5 0 3,0 0 4),(1 1 1,3 1 1,1 3 1,1 1 1))'));

```

|  | st_asewkt                                                                            |
|--|--------------------------------------------------------------------------------------|
|  | MULTILINESTRING((0 0 0 1,0 5 0 2,5 0 0 3,0 0 0 4),(1 1 0 1,3 1 0 1,1 3 0 1,1 1 0 1)) |

## Veja também

[ST\\_AsEWKT](#), [ST\\_Force2D](#), [ST\\_Force3DM](#), [ST\\_Force3D](#)

### 8.6.8 ST\_ForcePolygonCCW

ST\_ForcePolygonCCW — Orients all exterior rings counter-clockwise and all interior rings clockwise.

## Synopsis

```
geometry ST_ForcePolygonCCW ( geometry geom );
```

## Descrição

Forces (Multi)Polygons to use a counter-clockwise orientation for their exterior ring, and a clockwise orientation for their interior rings. Non-polygonal geometries are returned unchanged.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

## Veja também

[ST\\_ForcePolygonCW](#) , [ST\\_IsPolygonCCW](#) , [ST\\_IsPolygonCW](#)

## 8.6.9 ST\_ForceCollection

ST\_ForceCollection — Converte a geometria para um GEOMETRYCOLLECTION.

### Synopsis

geometry **ST\_ForceCollection**(geometry geomA);

### Descrição

Converte a geometria em um GEOMETRYCOLLECTION. Isto é útil para simplificar a representação WKB.

Melhorias: 2.0.0 suporte a superfícies polihédricas foi introduzido.

Disponibilidade: 1.2.2, antes da versão 1.3.4 esta função irá reportar um erro com curvas. Resolvido na versão 1.3.4+.

Alterado: 2.1.0. Até a versão 2.0.x esta função era chamada de ST\_Force\_Collection.



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

### Exemplos

```
SELECT ST_AsEWKT(ST_ForceCollection('POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))'));

```

st\_asewkt

---

```
GEOMETRYCOLLECTION(POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))
```

```
SELECT ST_AsText(ST_ForceCollection('CIRCULARSTRING(220227 150406,220227 150407,220227 150406)'));

```

st\_astext

---

```
GEOMETRYCOLLECTION(CIRCULARSTRING(220227 150406,220227 150407,220227 150406))
(1 row)
```

-- exemplo POLYHEDRAL --

```
SELECT ST_AsEWKT(ST_ForceCollection('POLYHEDRALSURFACE(((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0)),
((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0)),
((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0)),
((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0)),
((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0)),
((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1)))'));

```

st\_asewkt

---

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

```
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))  
)
```

#### Veja também

[ST\\_AsEWKT](#), [ST\\_Force2D](#), [ST\\_Force3DM](#), [ST\\_Force3D](#), [ST\\_GeomFromEWKT](#)

### 8.6.10 ST\_ForcePolygonCW

`ST_ForcePolygonCW` — Orients all exterior rings clockwise and all interior rings counter-clockwise.

#### Synopsis

geometry `ST_ForcePolygonCW` ( geometry geom );

#### Descrição

Forces (Multi)Polygons to use a clockwise orientation for their exterior ring, and a counter-clockwise orientation for their interior rings. Non-polygonal geometries are returned unchanged.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

#### Veja também

[ST\\_ForcePolygonCCW](#) , [ST\\_IsPolygonCCW](#) , [ST\\_IsPolygonCW](#)

### 8.6.11 ST\_ForceSFS

`ST_ForceSFS` — Força as geometrias a utilizarem os tipos disponíveis na especificação SFS 1.1.

#### Synopsis

geometry `ST_ForceSFS`(geometry geomA);  
geometry `ST_ForceSFS`(geometry geomA, text version);

#### Descrição



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.

## 8.6.12 ST\_ForceRHR

ST\_ForceRHR — Força a orientação dos vértices em um polígono a seguir a regra da mão direita.

### Synopsis

geometry **ST\_ForceRHR**(geometry g);

### Descrição

Forces the orientation of the vertices in a polygon to follow a Right-Hand-Rule, in which the area that is bounded by the polygon is to the right of the boundary. In particular, the exterior ring is orientated in a clockwise direction and the interior rings in a counter-clockwise direction. This function is a synonym for [ST\\_ForcePolygonCW](#)



#### Note

The above definition of the Right-Hand-Rule conflicts with definitions used in other contexts. To avoid confusion, it is recommended to use [ST\\_ForcePolygonCW](#).

Melhorias: 2.0.0 suporte a superfícies polihédricas foi introduzido.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

### Exemplos

```
SELECT ST_AsEWKT(
  ST_ForceRHR(
    'POLYGON((0 0 2, 5 0 2, 0 5 2, 0 0 2),(1 1 2, 1 3 2, 3 1 2, 1 1 2))'
  )
);
```

|         | st_asewkt                                                    |
|---------|--------------------------------------------------------------|
|         | POLYGON((0 0 2,0 5 2,5 0 2,0 0 2),(1 1 2,3 1 2,1 3 2,1 1 2)) |
| (1 row) |                                                              |

### Veja também

[ST\\_ForcePolygonCCW](#) , [ST\\_ForcePolygonCW](#) , [ST\\_IsPolygonCCW](#) , [ST\\_IsPolygonCW](#) , [ST\\_BuildArea](#), [ST\\_Polygonize](#), [ST\\_Reverse](#)

## 8.6.13 ST\_ForceCurve

ST\_ForceCurve — Converte para cima uma geometria para seu tipo curvo, se aplicável.

### Synopsis

geometry **ST\_ForceCurve**(geometry g);

## Descrição

Transforma uma geometria em sua representação curva, se aplicável. linhas se transformar em compoundcurves, multi-linhas se transformam em multicurves, polígonos em curvepolygons, multi-polígonos em multisurfaces. Se a entrada já é do tipo curvo, a função retorna a mesma entrada.

Disponibilidade: 2.2.0



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
SELECT ST_AsText (
  ST_ForceCurve (
    'POLYGON((0 0 2, 5 0 2, 0 5 2, 0 0 2),(1 1 2, 1 3 2, 3 1 2, 1 1 2))'::geometry
  )
);
```

|         | st_astext                                                            |
|---------|----------------------------------------------------------------------|
|         | CURVEPOLYGON Z ((0 0 2,5 0 2,0 5 2,0 0 2),(1 1 2,1 3 2,3 1 2,1 1 2)) |
| (1 row) |                                                                      |

## Veja também

[ST\\_LineToCurve](#)

### 8.6.14 ST\_LineMerge

**ST\_LineMerge** — Retorna um (conjunto de) LineString(s), costuradas em uma MULTILINESTRING.

#### Synopsis

geometry **ST\_LineMerge**(geometry amultilinestring);

#### Descrição

Retorna um conjunto de LineStrings, costuradas em uma MULTILINESTRING.



#### Note

Somente use com MULTILINESTRING/LINESTRINGs. Se você utilizar um polígono ou uma coleção de geometrias como entrada desta função, o retorno será um GEOMETRYCOLLECTION vazio.

Disponibilidade: 1.1.0



#### Note

Requer GEOS >= 2.1.0

**Warning**

Will strip the M dimension.

**Exemplos**

```

SELECT ST_AsText(ST_LineMerge(
ST_GeomFromText('MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33), (-45 -33,-46 -32))')
)
);
st_astext
-----
LINESTRING(-29 -27,-30 -29.7,-36 -31,-45 -33,-46 -32)
(1 row)

--If can't be merged - original MULTILINESTRING is returned
SELECT ST_AsText(ST_LineMerge(
ST_GeomFromText('MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33), (-45.2 -33.2,-46 -32))')
)
);
st_astext
-----
MULTILINESTRING((-45.2 -33.2,-46 -32), (-29 -27,-30 -29.7,-36 -31,-45 -33))

-- example with Z dimension
SELECT ST_AsText(ST_LineMerge(
ST_GeomFromText('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))')
)
);
st_astext
-----
LINESTRING Z (-30 -29.7 5,-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 1,-46 -32 11)
(1 row)

```

**Veja também**

[ST\\_Segmentize](#), [ST\\_LineSubstring](#)

**8.6.15 ST\_CollectionExtract**

**ST\_CollectionExtract** — Dada uma (multi)geometria, retorna uma (multi)geometria consistida apenas por elementos do tipo especificado.

**Synopsis**

geometry **ST\_CollectionExtract**(geometry collection, integer type);

## Descrição

Dada uma (multi)geometria, retorna uma (multi)geometria, apenas do tipo geométrico especificado. Sub-geometrias que não são dos tipos especificados são ignorados. Se não existem sub-geometrias do tipo escolhido, uma geometria vazia será retornada. Somente pontos, linhas e polígonos são suportados. Os tipos numéricos são 1 == POINT 2 == LINESTRING, 3 == POLYGON.

Disponibilidade: 1.5.0



### Note

Antes da versão 1.5.3, esta função retornava entradas que não eram coleções sem alterá-las, independente do tipo. Na versão 1.5.3, geometrias solitárias retornam NULL. Na versão 2.0.0, todo caso de resultados não encontrados retornam uma geometria VAZIA do tipo escolhido.



### Warning

Quando especificar 3 == POLYGON, um multi-polígono é retornado, mesmo quando os limites são compartilhados. Isto resulta em multi-polígonos inválidos em vários casos, como aplicar esta função ao resultado de [ST\\_Split](#).

## Exemplos

```
-- Constantes: 1 == POINT, 2 == LINESTRING, 3 == POLYGON
SELECT ST_AsText(ST_CollectionExtract(ST_GeomFromText('GEOMETRYCOLLECTION(↔
    GEOMETRYCOLLECTION(POINT(0 0)))'),1));
st_astext
-----
MULTIPOINT(0 0)
(1 row)

SELECT ST_AsText(ST_CollectionExtract(ST_GeomFromText('GEOMETRYCOLLECTION(↔
    GEOMETRYCOLLECTION(LINESTRING(0 0, 1 1),LINESTRING(2 2, 3 3))'),2));
st_astext
-----
MULTILINESTRING((0 0, 1 1), (2 2, 3 3))
(1 row)
```

## Veja também

[ST\\_Multi](#), [ST\\_Dump](#), [ST\\_CollectionHomogenize](#)

### 8.6.16 ST\_CollectionHomogenize

`ST_CollectionHomogenize` — Dada uma coleção geométrica, retorna a representação "mais simples" dos conteúdos.

#### Synopsis

```
geometry ST_CollectionHomogenize(geometry collection);
```



## Descrição

Dada uma coleção geométrica, retorna a representação mais simples de seu conteúdo. Geometrias solitárias serão retornadas como solitárias. Coleções homogêneas serão retornadas com o tipo múltiplo apropriado.



### Warning

Quando especificar `3 == POLYGON`, um multi-polígono é retornado, mesmo quando os limites são compartilhados. Isto resulta em multi-polígonos inválidos em vários casos, como aplicar esta função ao resultado de [ST\\_Split](#).

Disponibilidade: 2.0.0

## Exemplos

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0))'));

 st_astext
-----
 POINT(0 0)
 (1 row)

SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0),POINT(1 1))'));

 st_astext
-----
 MULTIPOINT(0 0,1 1)
 (1 row)
```

## Veja também

[ST\\_Multi](#), [ST\\_CollectionExtract](#)

### 8.6.17 ST\_Multi

`ST_Multi` — Restitui a geometria como uma `MULTI*` geometria.

#### Synopsis

```
geometry ST_Multi(geometry g1);
```

#### Descrição

Retorna a geometria como uma `MULTI*` geometria. Se a geometria já é `MULTI*`, ela retorna inalterada.

#### Exemplos

```
SELECT ST_AsText(ST_Multi(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450,
743265 2967450,743265.625 2967416,743238 2967416))')));
 st_astext
-----
```

```
MULTIPOLYGON(((743238 2967416,743238 2967450,743265 ↵
                2967450,743265.625 2967416,
                743238 2967416)))
(1 row)
```

### Veja também

[ST\\_AsText](#)

## 8.6.18 ST\_Normalize

ST\_Normalize — Retorna a geometria na sua forma canônica.

### Synopsis

geometry **ST\_Normalize**(geometry geom);

### Descrição

Retorna a geometria na sua forma normalizada/canônica. Talvez rearranja vértices em anéis de polígonos, anéis em um polígono, elementos em um complexo de multi-geometria.

Mais usada para teste (comparando resultados obtidos e esperados).

Disponibilidade: 2.3.0

### Exemplos

```
SELECT ST_AsText(ST_Normalize(ST_GeomFromText (
  'GEOMETRYCOLLECTION(
    POINT(2 3),
    MULTILINESTRING((0 0, 1 1),(2 2, 3 3)),
    POLYGON(
      (0 10,0 0,10 0,10 10,0 10),
      (4 2,2 2,2 4,4 4,4 2),
      (6 8,8 8,8 6,6 6,6 8)
    )
  )'
)));
```

st\_astext

---

```
GEOMETRYCOLLECTION(POLYGON((0 0,0 10,10 10,10 0,0 0),(6 6,8 6,8 8,6 8,6 6),(2 2,4 2,4 4,2 ↵
  4,2 2)),MULTILINESTRING((2 2,3 3),(0 0,1 1)),POINT(2 3))
(1 row)
```

### Veja também

[ST\\_Equals](#),

## 8.6.19 ST\_QuantizeCoordinates

ST\_QuantizeCoordinates — Sets least significant bits of coordinates to zero

## Synopsis

```
geometry ST_QuantizeCoordinates ( geometry g , int prec_x , int prec_y , int prec_z , int prec_m );
```

## Descrição

`ST_QuantizeCoordinates` determines the number of bits (N) required to represent a coordinate value with a specified number of digits after the decimal point, and then sets all but the N most significant bits to zero. The resulting coordinate value will still round to the original value, but will have improved compressibility. This can result in a significant disk usage reduction provided that the geometry column is using a **compressible storage type**. The function allows specification of a different number of digits after the decimal point in each dimension; unspecified dimensions are assumed to have the precision of the x dimension. Negative digits are interpreted to refer digits to the left of the decimal point, (i.e., `prec_x=-2` will preserve coordinate values to the nearest 100).

The coordinates produced by `ST_QuantizeCoordinates` are independent of the geometry that contains those coordinates and the relative position of those coordinates within the geometry. As a result, existing topological relationships between geometries are unaffected by use of this function. The function may produce invalid geometry when it is called with a number of digits lower than the intrinsic precision of the geometry.

Availability: 2.5.0

## Technical Background

PostGIS stores all coordinate values as double-precision floating point integers, which can reliably represent 15 significant digits. However, PostGIS may be used to manage data that intrinsically has fewer than 15 significant digits. An example is TIGER data, which is provided as geographic coordinates with six digits of precision after the decimal point (thus requiring only nine significant digits of longitude and eight significant digits of latitude.)

When 15 significant digits are available, there are many possible representations of a number with 9 significant digits. A double precision floating point number uses 52 explicit bits to represent the significand (mantissa) of the coordinate. Only 30 bits are needed to represent a mantissa with 9 significant digits, leaving 22 insignificant bits; we can set their value to anything we like and still end up with a number that rounds to our input value. For example, the value 100.123456 can be represented by the floating point numbers closest to 100.123456000000, 100.123456000001, and 100.123456432199. All are equally valid, in that `ST_AsText(geom, 6)` will return the same result with any of these inputs. As we can set these bits to any value, `ST_QuantizeCoordinates` sets the 22 insignificant bits to zero. For a long coordinate sequence this creates a pattern of blocks of consecutive zeros that is compressed by PostgreSQL more efficiently.



### Note

Only the on-disk size of the geometry is potentially affected by `ST_QuantizeCoordinates`. `ST_MemSize`, which reports the in-memory usage of the geometry, will return the the same value regardless of the disk space used by a geometry.

## Exemplos

```
SELECT ST_AsText(ST_QuantizeCoordinates('POINT (100.123456 0)::geometry, 4));
st_astext
-----
POINT(100.123455047607 0)
```

```
WITH test AS (SELECT 'POINT (123.456789123456 123.456789123456)::geometry AS geom)
SELECT
  digits,
  encode(ST_QuantizeCoordinates(geom, digits), 'hex'),
  ST_AsText(ST_QuantizeCoordinates(geom, digits))
FROM test, generate_series(15, -15, -1) AS digits;
```

| digits | encode                                         | st_astext                     |
|--------|------------------------------------------------|-------------------------------|
| 15     | 010100000005f9a72083cdd5e405f9a72083cdd5e40    | POINT (123.456789123456 ↔     |
|        | 123.456789123456)                              |                               |
| 14     | 010100000005f9a72083cdd5e405f9a72083cdd5e40    | POINT (123.456789123456 ↔     |
|        | 123.456789123456)                              |                               |
| 13     | 010100000005f9a72083cdd5e405f9a72083cdd5e40    | POINT (123.456789123456 ↔     |
|        | 123.456789123456)                              |                               |
| 12     | 010100000005c9a72083cdd5e405c9a72083cdd5e40    | POINT (123.456789123456 ↔     |
|        | 123.456789123456)                              |                               |
| 11     | 0101000000409a72083cdd5e40409a72083cdd5e40     | POINT (123.456789123456 ↔     |
|        | 123.456789123456)                              |                               |
| 10     | 0101000000009a72083cdd5e40009a72083cdd5e40     | POINT (123.456789123455 ↔     |
|        | 123.456789123455)                              |                               |
| 9      | 0101000000009072083cdd5e40009072083cdd5e40     | POINT (123.456789123418 ↔     |
|        | 123.456789123418)                              |                               |
| 8      | 0101000000008072083cdd5e40008072083cdd5e40     | POINT (123.45678912336 ↔      |
|        | 123.45678912336)                               |                               |
| 7      | 0101000000000070083cdd5e40000070083cdd5e40     | POINT (123.456789121032 ↔     |
|        | 123.456789121032)                              |                               |
| 6      | 0101000000000040083cdd5e40000040083cdd5e40     | POINT (123.456789076328 ↔     |
|        | 123.456789076328)                              |                               |
| 5      | 0101000000000000083cdd5e400000000083cdd5e40    | POINT (123.456789016724 ↔     |
|        | 123.456789016724)                              |                               |
| 4      | 01010000000000000003cdd5e400000000003cdd5e40   | POINT (123.456787109375 ↔     |
|        | 123.456787109375)                              |                               |
| 3      | 0101000000000000000003cdd5e400000000003cdd5e40 | POINT (123.456787109375 ↔     |
|        | 123.456787109375)                              |                               |
| 2      | 01010000000000000000038dd5e4000000000038dd5e40 | POINT (123.45654296875 ↔      |
|        | 123.45654296875)                               |                               |
| 1      | 0101000000000000000000dd5e400000000000dd5e40   | POINT (123.453125 123.453125) |
| 0      | 0101000000000000000000dc5e400000000000dc5e40   | POINT (123.4375 123.4375)     |
| -1     | 0101000000000000000000c05e400000000000c05e40   | POINT (123 123)               |
| -2     | 0101000000000000000000005e400000000000005e40   | POINT (120 120)               |
| -3     | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -4     | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -5     | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -6     | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -7     | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -8     | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -9     | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -10    | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -11    | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -12    | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -13    | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -14    | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |
| -15    | 01010000000000000000000058400000000000005840   | POINT (96 96)                 |

**Veja também**

[ST\\_SnapToGrid](#)

**8.6.20 ST\_RemovePoint**

ST\_RemovePoint — Remove um ponto de uma linestring.

**Synopsis**

geometry **ST\_RemovePoint**(geometry linestring, integer offset);

**Descrição**

Remove um ponto de uma linestring, dado seu index com base 0. Útil para transformar um anel fechado em uma line string aberta.

Disponibilidade: 1.1.0



This function supports 3d and will not drop the z-index.

**Exemplos**

```
--garante que as LINESTRINGS não são fechadas removendo o ponto final
UPDATE sometable
   SET the_geom = ST_RemovePoint(the_geom, ST_NPoints(the_geom) - 1)
   FROM sometable
  WHERE ST_IsClosed(the_geom) = true;
```

**Veja também**

[ST\\_AddPoint](#), [ST\\_NPoints](#), [ST\\_NumPoints](#)

**8.6.21 ST\_Reverse**

**ST\_Reverse** — Retorna a geometria com a ordem dos vértices revertida.

**Synopsis**

geometry **ST\_Reverse**(geometry g1);

**Descrição**

Pode ser usado em qualquer geometria e reverte a ordem dos vértices.

Enhanced: 2.4.0 support for curves was introduced.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

**Exemplos**

```
SELECT ST_AsText(the_geom) as line, ST_AsText(ST_Reverse(the_geom)) As reverseline
FROM
  (SELECT ST_MakeLine(ST_MakePoint(1,2),
                    ST_MakePoint(1,10)) As the_geom) as foo;
--result
      line          | reverseline
-----+-----
LINESTRING(1 2,1 10) | LINESTRING(1 10,1 2)
```

## 8.6.22 ST\_Rotate

ST\_Rotate — Rotaciona uma geometria em rotRadians em sentido anti-horário de sua origem.

### 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);
```

### Descrição

Rotaciona uma geometria rotRadians em sentido anti-horário da origem. O ponto de origem da rotação pode ser especificado como uma ponto, ou como coordenadas XY. Se a origem não é especificada a geometria é rotacionada na origem POINT(0 0).

Melhorias: 2.0.0 suporte para superfícies polihédricas, triângulos e TINs introduzido.

Melhoria: 2.0.0 parâmetros adicionais para especificação da origem de rotação adicionados.

Disponibilidade: 1.1.2. Mudança de nome de Affine para ST\_Affine na versão 1.2.2.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### Exemplos

```
--Rotate 180 degrees
SELECT ST_AsEWKT(ST_Rotate('LINESTRING (50 160, 50 50, 100 50)', pi()));
      st_asewkt
-----
LINESTRING(-50 -160,-50 -50,-100 -50)
(1 row)

--Rotate 30 degrees counter-clockwise at x=50, y=160
SELECT ST_AsEWKT(ST_Rotate('LINESTRING (50 160, 50 50, 100 50)', pi()/6, 50, 160));
      st_asewkt
-----
LINESTRING(50 160,105 64.7372055837117,148.301270189222 89.7372055837117)
(1 row)

--Rotate 60 degrees clockwise from centroid
SELECT ST_AsEWKT(ST_Rotate(geom, -pi()/3, ST_Centroid(geom)))
FROM (SELECT 'LINESTRING (50 160, 50 50, 100 50)'::geometry AS geom) AS foo;
      st_asewkt
-----
LINESTRING(116.4225 130.6721,21.1597 75.6721,46.1597 32.3708)
(1 row)
```

### Veja também

[ST\\_Affine](#), [ST\\_RotateX](#), [ST\\_RotateY](#), [ST\\_RotateZ](#)

### 8.6.23 ST\_RotateX

ST\_RotateX — Rotaciona uma geometria rotRadians em cima do eixo X.

#### Synopsis

geometry **ST\_RotateX**(geometry geomA, float rotRadians);

#### Descrição

Rotaciona uma geometria geomA - rotRadians sobre o eixo X.



#### Note

ST\_RotateX(geomA, rotRadians) é um atalho para ST\_Affine(geomA, 1, 0, 0, 0, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0).

Melhorias: 2.0.0 suporte para superfícies polihédricas, triângulos e TINs introduzido.

Disponibilidade: 1.1.2. Mudança de nome de Affine para ST\_Affine na versão 1.2.2.



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

#### Exemplos

```
--Rotaciona uma linha 90 graus ao longo do eixo X
SELECT ST_AsEWKT(ST_RotateX(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
           st_asewkt
-----
LINESTRING(1 -3 2,1 -1 1)
```

#### Veja também

[ST\\_Affine](#), [ST\\_RotateY](#), [ST\\_RotateZ](#)

### 8.6.24 ST\_RotateY

ST\_RotateY — Rotaciona uma geometria rotRadians em cima do eixo Y.

#### Synopsis

geometry **ST\_RotateY**(geometry geomA, float rotRadians);

## Descrição

Rotaciona uma geometria geomA - rotRadians sobre o eixo Y.



### Note

`ST_RotateY(geomA, rotRadians)` é um atalho para `ST_Affine(geomA, cos(rotRadians), 0, sin(rotRadians), 0, 1, 0, -sin(rotRadians), 0, cos(rotRadians), 0, 0, 0)`.

Disponibilidade: 1.1.2. Mudança de nome de Affine para `ST_Affine` na versão 1.2.2.

Melhorias: 2.0.0 suporte para superfícies polihédricas, triângulos e TINs introduzido.



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

```
--Rotaciona uma linha 90 graus ao longo do eixo Y
SELECT ST_AsEWKT(ST_RotateY(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
           st_asewkt
-----
LINESTRING(3 2 -1,1 1 -1)
```

## Veja também

[ST\\_Affine](#), [ST\\_RotateX](#), [ST\\_RotateZ](#)

### 8.6.25 ST\_RotateZ

`ST_RotateZ` — Rotaciona uma geometria rotRadians em cima do eixo Z.

## Synopsis

geometry **ST\_RotateZ**(geometry geomA, float rotRadians);

## Descrição

Rotaciona uma geometria geomA - rotRadians sobre o eixo Z.



### Note

Esta função é um sinônimo para `ST_Rotate`



**Note**

`ST_RotateZ(geomA, rotRadians)` é um atalho para `SELECT ST_Affine(geomA, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0, 1, 0, 0, 0)`.

Melhorias: 2.0.0 suporte para superfícies polihédricas, triângulos e TINs introduzido.

Disponibilidade: 1.1.2. Mudança de nome de `Affine` para `ST_Affine` na versão 1.2.2.

**Note**

Antes da versão 1.3.4, esta função falha se utilizada em geometrias que continham curvas. Resolvido nas versões 1.3.4+.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

**Exemplos**

```
--Rotaciona uma linha 90 graus ao longo do eixo Z
SELECT ST_AsEWKT(ST_RotateZ(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
          st_asewkt
-----
LINESTRING(-2 1 3,-1 1 1)

--Rotate a curved circle around z-axis
SELECT ST_AsEWKT(ST_RotateZ(the_geom, pi()/2))
FROM (SELECT ST_LineToCurve(ST_Buffer(ST_GeomFromText('POINT(234 567)'), 3)) As the_geom) ←
      As foo;

-----

CURVEPOLYGON(CIRCULARSTRING(-567 237,-564.87867965644 236.12132034356,-564 ←
234,-569.12132034356 231.87867965644,-567 237))
```

**Veja também**

[ST\\_Affine](#), [ST\\_RotateX](#), [ST\\_RotateY](#)

**8.6.26 ST\_Scale**

`ST_Scale` — Escala uma geometria pelos fatores dados.

## 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);
```

## Descrição

Escala a geometria para um tamanho novo multiplicando as ordenadas com os parâmetros correspondentes do coeficiente.

A versão tomando uma geometria como `fator` o parâmetro permite passar um 2d, 3dm, 3dz ou 4d ponto para configurar coeficiente de escala para todas as dimensões suportadas. Dimensões perdidas no `fator` ponto são equivalentes a nenhuma escala na dimensão correspondente.

The three-geometry variant allows a "false origin" for the scaling to be passed in. This allows "scaling in place", for example using the centroid of the geometry as the false origin. Without a false origin, scaling takes place relative to the actual origin, so all coordinates are just multiplied by the scale factor.



### Note

Antes da versão 1.3.4, esta função falha se utilizada em geometrias que continham curvas. Resolvido nas versões 1.3.4+.

Disponibilidade: 1.1.0

Melhorias: 2.0.0 suporte para superfícies polihédricas, triângulos e TINs introduzido.

Melhorias: 2.2.0 suporte para escalar todas as dimensões (parâmetro de geometria) foi introduzido.



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports M coordinates.

## Exemplos

```
--Version 1: scale X, Y, Z
SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 0.75, 0.8));
-----
LINESTRING(0.5 1.5 2.4,0.5 0.75 0.8)

--Version 2: Scale X Y
SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 0.75));
-----
LINESTRING(0.5 1.5 3,0.5 0.75 1)

--Version 3: Scale X Y Z M
```

```

SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3 4, 1 1 1 1)'),
  ST_MakePoint(0.5, 0.75, 2, -1)));
          st_asewkt
-----
LINESTRING(0.5 1.5 6 -4,0.5 0.75 2 -1)

--Version 4: Scale X Y using false origin
SELECT ST_AsText(ST_Scale('LINESTRING(1 1, 2 2)', 'POINT(2 2)', 'POINT(1 1)::geometry'));
          st_astext
-----
LINESTRING(1 1,3 3)

```

**Veja também**[ST\\_Affine](#), [ST\\_TransScale](#)**8.6.27 ST\_Segmentize**

`ST_Segmentize` — Retorna uma geometria/geografia alterada não tendo nenhum segmento maior que a distância dada.

**Synopsis**

```

geometry ST_Segmentize(geometry geom, float max_segment_length);
geography ST_Segmentize(geography geog, float max_segment_length);

```

**Descrição**

Retorna a geometria alterada não tendo nenhum segmento maior que o dado `max_segment_length`. O cálculo de distância é efetuado somente no 2o dia. Para geometria, unidades de comprimento estão em unidades de referência espacial. Para geografia, unidades estão em metros.

Disponibilidade: 1.2.2

Enhanced: 3.0.0 Segmentize geometry now uses equal length segments

Enhanced: 2.3.0 Segmentize geography now uses equal length segments

Melhorias: 2.1.0 suporte para geografia foi introduzido.

Alteração: 2.1.0 Como um resultado da introdução do suporte de geografia: A construção `SELECT ST_Segmentize('LINESTRING(1 2, 3 4)', 0.5)`; irá resultar em uma função de erro ambíguo. Você precisa ter o objeto propriamente digitado ex. uma coluna geometria/geografia, use `ST_GeomFromText`, `ST_GeogFromText` or `SELECT ST_Segmentize('LINESTRING(1 2, 3 4)::geometry, 0.5)`;

**Note**

Isso só irá aumentar segmentos. Não irá alongar segmentos menores que o comprimento máximo

**Exemplos**

```

SELECT ST_AsText(ST_Segmentize(
ST_GeomFromText('MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33), (-45 -33,-46 -32))')
, 5)
);
-----
st_astext
-----
MULTILINESTRING((-29 -27,-30 -29.7,-34.886615700134 -30.758766735029,-36 -31,
-40.8809353009198 -32.0846522890933,-45 -33),
(-45 -33,-46 -32))
(1 row)

SELECT ST_AsText(ST_Segmentize(ST_GeomFromText('POLYGON((-29 28, -30 40, -29 28))'),10));
st_astext
-----
POLYGON((-29 28,-29.8304547985374 37.9654575824488,-30 40,-29.1695452014626 ←
30.0345424175512,-29 28))
(1 row)

```

**Veja também**[ST\\_LineSubstring](#)**8.6.28 ST\_SetPoint**

ST\_SetPoint — Substitui ponto de uma linestring com um dado ponto.

**Synopsis**

geometry **ST\_SetPoint**(geometry linestring, integer zerobasedposition, geometry point);

**Descrição**

Substitui ponto N de linstring com um dado ponto. Index é de base 0. Index negativo são contados atrasados, logo -1 é o último ponto. Isso é especialmente usado em causas tentando manter relações juntas quando um vértice se move.

Disponibilidade: 1.1.0

Atualizado 2.3.0: indexing negativo



This function supports 3d and will not drop the z-index.

**Exemplos**

```

--Change first point in line string from -1 3 to -1 1
SELECT ST_AsText(ST_SetPoint('LINESTRING(-1 2,-1 3)', 0, 'POINT(-1 1)'));
-----
st_astext
-----
LINESTRING(-1 1,-1 3)

---Change last point in a line string (lets play with 3d linestring this time)
SELECT ST_AsEWKT(ST_SetPoint(foo.the_geom, ST_NumPoints(foo.the_geom) - 1, ST_GeomFromEWKT ←
('POINT(-1 1 3)')))
FROM (SELECT ST_GeomFromEWKT('LINESTRING(-1 2 3,-1 3 4, 5 6 7)') As the_geom) As foo;

```

```

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

SELECT ST_AsText(ST_SetPoint(g, -3, p))
FROM ST_GeomFromText('LINESTRING(0 0, 1 1, 2 2, 3 3, 4 4)') AS g
      , ST_PointN(g,1) as p;
          st_astext
-----
LINESTRING(0 0,1 1,0 0,3 3,4 4)

```

**Veja também**

[ST\\_AddPoint](#), [ST\\_NPoints](#), [ST\\_NumPoints](#), [ST\\_PointN](#), [ST\\_RemovePoint](#)

**8.6.29 ST\_SetSRID**

**ST\_SetSRID** — Configure SRID em uma geometria para um valor inteiro específico.

**Synopsis**

geometry **ST\_SetSRID**(geometry geom, integer srid);

**Descrição**

Configura SRID em uma geometria para um valor inteiro específico. Útil em construir caixas seguras para pesquisas.

**Note**

Essa função não transforma as coordenadas da geometria em nenhuma maneira - ela simplesmente configura os dados meta definindo o sistema de referência espacial que a geometria supostamente está. Use [ST\\_Transform](#) se você quiser transformar a geometria em uma nova projeção.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method supports Circular Strings and Curves

**Exemplos**

-- Marque um ponto como WGS 84 extensa latitude --

```

SELECT ST_SetSRID(ST_Point(-123.365556, 48.428611),4326) As wgs84long_lat;
-- the ewkt representation (wrap with ST_AsEWKT) -
SRID=4326;POINT(-123.365556 48.428611)

```

-- Marque um ponto com WGS 84 extensa latitude e então transforme para mercator web (Mercator Esférico) --

```

SELECT ST_Transform(ST_SetSRID(ST_Point(-123.365556, 48.428611),4326),3785) As spere_merc;
-- the ewkt representation (wrap with ST_AsEWKT) -
SRID=3785;POINT(-13732990.8753491 6178458.96425423)

```

**Veja também**

Section [4.3.1](#), [ST\\_AsEWKT](#), [ST\\_Point](#), [ST\\_SRID](#), [ST\\_Transform](#), [UpdateGeometrySRID](#)

**8.6.30 ST\_SnapToGrid**

`ST_SnapToGrid` — Rompe todos os pontos da geometria de entrada para uma rede regular.

**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);
```

**Descrição**

Variante1,2,3: Rompe todos os pontos da geometria de entrada para a rede definida por sua origem e tamanho da célula. Remove pontos consecutivos caindo na mesma célula, finalmente retornando NULO se os pontos de saída não são suficientes para definir uma geometria do tipo dado. Geometrias colapsadas em uma coleção são desguarnecidas disso. Útil para reduzi a precisão.

Variante4: Introduzido 1.1.0 - Rompe todos os pontos da geometria de entrada para a rede definida por sua origem (o segundo argumento deve ser um ponto) e tamanhos de células. Especifica 0 como um tamanho para qualquer dimensão que você não quer romper para uma rede.

**Note**

A geometria de retorno pode perder sua simplicidade (veja [ST\\_IsSimple](#)).

**Note**

Antes de lançar 1.1.0, essa função sempre retornou uma geometria 2d. Começando em 1.1.0 a geometria de retorno terá a mesma dimensionalidade da entrada com maiores valores intocados de dimensão. Use a versão pegando um segundo argumento de geometria para definir todas as dimensões de rede.

Disponibilidade: 1.0.0RC1

Disponibilidade: 1.1.0 - suporte a Z e M



This function supports 3d and will not drop the z-index.

**Exemplos**

```
--Snap your geometries to a precision grid of 10^-3
UPDATE mytable
  SET the_geom = ST_SnapToGrid(the_geom, 0.001);

SELECT ST_AsText(ST_SnapToGrid(
  ST_GeomFromText('LINESTRING(1.1115678 2.123, 4.111111 3.2374897, ↵
    4.11112 3.23748667)'),
  0.001)
  );
      st_astext
```

```

-----
LINESTRING(1.112 2.123,4.111 3.237)
--Snap a 4d geometry
SELECT ST_AsEWKT(ST_SnapToGrid(
    ST_GeomFromEWKT('LINESTRING(-1.1115678 2.123 2.3456 1.11111,
        4.111111 3.2374897 3.1234 1.1111, -1.11111112 2.123 2.3456 1.111112)'),
    ST_GeomFromEWKT('POINT(1.12 2.22 3.2 4.4444)'),
    0.1, 0.1, 0.1, 0.01) );
                                                                    st_asewkt
-----
LINESTRING(-1.08 2.12 2.3 1.1144,4.12 3.22 3.1 1.1144,-1.08 2.12 2.3 1.1144)

--With a 4d geometry - the ST_SnapToGrid(geom,size) only touches x and y coords but keeps m ←
and z the same
SELECT ST_AsEWKT(ST_SnapToGrid(ST_GeomFromEWKT('LINESTRING(-1.1115678 2.123 3 2.3456,
    4.111111 3.2374897 3.1234 1.1111)'),
    0.01) );
                                                                    st_asewkt
-----
LINESTRING(-1.11 2.12 3 2.3456,4.11 3.24 3.1234 1.1111)

```

**Veja também**

[ST\\_Snap](#), [ST\\_AsEWKT](#), [ST\\_AsText](#), [ST\\_GeomFromText](#), [ST\\_GeomFromEWKT](#), [ST\\_Simplify](#)

**8.6.31 ST\_Snap**

**ST\_Snap** — Rompe segmentos e vértices de geometria de entrada para vértices de uma geometria de referência.

**Synopsis**

geometry **ST\_Snap**(geometry input, geometry reference, float tolerance);

**Descrição**

Snaps the vertices and segments of a geometry another Geometry's vertices. A snap distance tolerance is used to control where snapping is performed. The result geometry is the input geometry with the vertices snapped. If no snapping occurs then the input geometry is returned unchanged.

Romper uma geometria para outra pode melhorar robusteza para operações de cobertura eliminando limites quase coincidentes (os quais causam problemas durante o sinal e cálculo de intersecção).

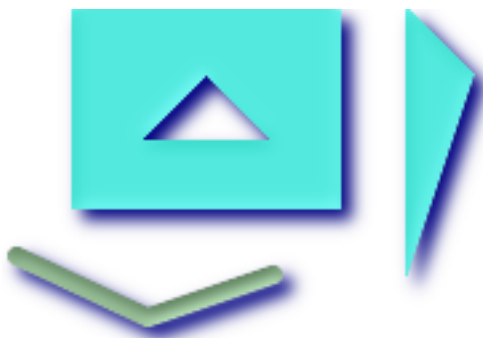
Romper muito pode resultar na criação de topologia inválida, então o número e localização dos vértices rompidos são decididos usando heurísticos para determinar quando é seguro romper. Entretanto, isso pode resultar em alguns rompimentos potencialmente omitidos.

**Note**

A geometria devolvida pode perder sua simplicidade (veja [ST\\_IsSimple](#)) e validade (veja [ST\\_IsValid](#)).

Disponibilidade: 2.0.0. requer GEOS >=3.3.0

**Exemplos**



*Um multi polígono apresentado com uma linestring (antes de qualquer rompimento)*





*Um multi polígono rompido para linestring para tolerância: 1.01 de distância. O novo multi polígono é mostrado com linestring de referência*

```
SELECT ST_AsText(ST_Snap(poly,line, ←
    ST_Distance(poly,line)*1.01)) AS polysnapped
FROM (SELECT
    ST_GeomFromText('MULTIPOLYGON(
    ((26 125, 26 200, 126 200, 126 125, ←
    26 125 ),
    ( 51 150, 101 150, 76 175, 51 150 ) ←
    ),
    (( 151 100, 151 200, 176 175, 151 ←
    100 )))') As poly,
    ST_GeomFromText('LINESTRING (5 ←
    107, 54 84, 101 100)') As line
    ) As foo;
```

polysnapped

```
MULTIPOLYGON(((26 125,26 200,126 200,126 ←
    125,101 100,26 125),
(51 150,101 150,76 175,51 150)),((151 ←
    100,151 200,176 175,151 100)))
```



*Um multi polígono rompido para linestring para tolerância: 1.25 de distância. O novo multi polígono é mostrado com linestring de referência*

```
SELECT ST_AsText (
    ST_Snap(poly,line, ST_Distance(poly, ←
    line)*1.25)
    ) AS polysnapped
FROM (SELECT
    ST_GeomFromText('MULTIPOLYGON(
    (( 26 125, 26 200, 126 200, 126 125, ←
    26 125 ),
    ( 51 150, 101 150, 76 175, 51 150 ) ←
    ),
    (( 151 100, 151 200, 176 175, 151 ←
    100 )))') As poly,
    ST_GeomFromText('LINESTRING (5 ←
    107, 54 84, 101 100)') As line
    ) As foo;
```

polysnapped

```
MULTIPOLYGON(((5 107,26 200,126 200,126 ←
    125,101 100,54 84,5 107),
(51 150,101 150,76 175,51 150)),((151 ←
    100,151 200,176 175,151 100)))
```



*A linestring rompida para o multi polígono original em tolerância de 1.01 de distância. As nova linestring é mostrada com multi polígono de referência*

```
SELECT ST_AsText (
  ST_Snap(line, poly, ST_Distance(poly, ↵
    line)*1.01)
) AS linesnapped
FROM (SELECT
  ST_GeomFromText ('MULTIPOLYGON (
    ((26 125, 26 200, 126 200, 126 125, ↵
    26 125),
    (51 150, 101 150, 76 175, 51 150 )) ↵
  ',
  ((151 100, 151 200, 176 175, 151 ↵
  100)))') As poly,
  ST_GeomFromText ('LINESTRING (5 ↵
  107, 54 84, 101 100)') As line
) As foo;

          linesnapped
-----
LINESTRING(5 107,26 125,54 84,101 100)
```



*A linestring rompida para o polígono original de tolerância 1.25 de distância. A nova linestring é mostrada com multi polígono de referência*

```
SELECT ST_AsText (
  ST_Snap(line, poly, ST_Distance(poly, ↵
    line)*1.25)
) AS linesnapped
FROM (SELECT
  ST_GeomFromText ('MULTIPOLYGON (
    (( 26 125, 26 200, 126 200, 126 125, ↵
    26 125 ),
    (51 150, 101 150, 76 175, 51 150 )) ↵
  ',
  ((151 100, 151 200, 176 175, 151 ↵
  100 )))') As poly,
  ST_GeomFromText ('LINESTRING (5 ↵
  107, 54 84, 101 100)') As line
) As foo;

          linesnapped
-----
LINESTRING(26 125,54 84,101 100)
```

## Veja também

[ST\\_SnapToGrid](#)

## 8.6.32 ST\_Transform

**ST\_Transform** — Retorna uma nova geometria com suas coordenadas transformadas em diferentes referências espaciais.

### 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);

## Descrição

Retorna uma nova geometria com suas coordenadas transformadas em um sistema de referência espacial diferente. O destino da referência espacial `to_srid` pode ser identificado por um parâmetro inteiro válido SRID (ele deve existir na `spatial_ref_sys` table). Alternativamente, uma referência espacial é definida como uma string PROJ.4 pode ser usada para `to_proj` e/ou `from_proj`, entretanto esses métodos não são otimizados. Se o destino do sistema de referência espacial é expressado com uma string PROJ.4 ao invés de uma SRID, a SRID da geometria de saída será configurada para zero. Com exceção das funções com `from_proj`, geometrias de entrada devem ter uma SRID definida.

ST\_Transform é confundido com ST\_SetSRID(). ST\_Transform na verdade altera as coordenadas de uma geometria de um sistema de referência espacial para outro, enquanto ST\_SetSRID() simplesmente muda o identificador SRID da geometria.



### Note

Requer que PostGIS seja compilado com o suporte Proj. Use [PostGIS\\_Full\\_Version](#) para confirmar que você o suporte proj compilado.



### Note

Se utilizar mais de uma transformação, é útil ter um index prático nas transformações comumente usadas para ter vantagem do uso index.



### Note

Antes da versão 1.3.4, esta função falha se utilizada em geometrias que continham curvas. Resolvido nas versões 1.3.4+.

Melhorias: 2.0.0 suporte a superfícies polihédricas foi introduzido.

Melhorias: 2.3.0 suporte para texto direto PROJ.4 foi introduzido.



This method implements the SQL/MM specification. SQL-MM 3: 5.1.6



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

Alterar a geometria do estado plano de Massachusetts EUA para WGS 84 long lat

```
SELECT ST_AsText(ST_Transform(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450,
743265 2967450,743265.625 2967416,743238 2967416))',2249),4326)) As wgs_geom;
```

```
wgs_geom
-----
POLYGON((-71.1776848522251 42.3902896512902,-71.1776843766326 42.3903829478009,
-71.1775844305465 42.3903826677917,-71.1775825927231 42.3902893647987,-71.177684
8522251 42.3902896512902));
(1 row)
```

```
--3D Circular String example
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromEWKT('SRID=2249;CIRCULARSTRING(743238 2967416 4326) ←
  1,743238 2967450 2,743265 2967450 3,743265.625 2967416 3,743238 2967416 4)'),4326));

                                st_asewkt
-----
SRID=4326;CIRCULARSTRING(-71.1776848522251 42.3902896512902 1,-71.1776843766326 4326 ←
  42.3903829478009 2,
-71.1775844305465 42.3903826677917 3,
-71.1775825927231 42.3902893647987 3,-71.1776848522251 42.3902896512902 4)
```

Exemplo da criação de um index funcional parcial. Para tables onde você não tem certeza que todas as geometrias caberão, é melhor usar um index parcial que deixa geometrias nulas que irão conservar seu espaço e farão seu index menor e mais eficiente.

```
CREATE INDEX idx_the_geom_26986_parcelas
  ON parcels
  USING gist
  (ST_Transform(the_geom, 26986))
  WHERE the_geom IS NOT NULL;
```

Exemplos usando PROJ.4 para transformar com referências espaciais personalizadas.

```
-- Find intersection of two polygons near the North pole, using a custom Gnostic projection
-- See http://boundlessgeo.com/2012/02/flattening-the-peel/
WITH data AS (
  SELECT
    ST_GeomFromText('POLYGON((170 50,170 72,-130 72,-130 50,170 50))', 4326) AS p1,
    ST_GeomFromText('POLYGON((-170 68,-170 90,-141 90,-141 68,-170 68))', 4326) AS p2,
    '+proj=gnom +ellps=WGS84 +lat_0=70 +lon_0=-160 +no_defs'::text AS gnom
  )
SELECT ST_AsText(
  ST_Transform(
    ST_Intersection(ST_Transform(p1, gnom), ST_Transform(p2, gnom)),
    gnom, 4326))
FROM data;

                                st_astext
-----
POLYGON((-170 74.053793645338,-141 73.4268621378904,-141 68,-170 68,-170 74.053793645338) ←
)
```

## Configurando comportamento de transformação

Algumas vezes transformações coordenadas envolvendo mudança de rede podem falhar, por exemplo se PROJ.4 não tiver sido construída com arquivos de mudança de rede ou a coordenada não se encontra com a extensão para qual a mudança de rede é definida. Por padrão, PostGIS irá lançar um erro se um arquivo de mudança de rede não estiver presente, mas esse comportamento pode ser configurado em uma base per-SRID e até mesmo testando diferentes `to_proj` valores do texto PROJ.4, ou alterando o `proj4text` valor com a `spatial_ref_sys` table.

Por exemplo, o `proj4text` parameter `+datum=NAD87` é uma forma estenográfica para os parâmetros `+nadgrids` seguintes:

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat
```

O prefixo `@` significa que nenhum erro foi relatado se os arquivos não estão presentes, mas se o fim da lista chega sem nenhum arquivo ser apropriado (encontrado ou sobreposto) então um erro é o problema.

Se, por outro lado, você queria garantir que pelo menos os arquivos padrões estivessem presentes, mas que se todos os arquivos tivessem sido escaneados sem atingir uma transformação nula é empregada, você poderia usar:

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat,null
```

O arquivo nulo de mudança de rede é um arquivo válido cobrindo o mundo todo e não aplicando nenhuma mudança. Então para um exemplo completo, se você quisesse alterar PostGIS para que as transformações para o SRID 4267 que não se encontraram com a variação correta não lançassem um erro, você deveria utilizar o seguinte:

```
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;
```

## Veja também

[PostGIS\\_Full\\_Version](#), [ST\\_AsText](#), [ST\\_SetSRID](#), [UpdateGeometrySRID](#)

### 8.6.33 ST\_Translate

ST\_Translate — Translação de uma geometria pelos dados deslocamentos.

#### Synopsis

```
geometry ST_Translate(geometry g1, float deltax, float deltay);
geometry ST_Translate(geometry g1, float deltax, float deltay, float deltaz);
```

#### Descrição

Retorna uma nova geometria cujas coordenadas são transladações unidades delta x, delta y, delta z. Unidades são baseadas nas unidades definidas na referência espacial (SRID) para essa geometria.



#### Note

Antes da versão 1.3.4, esta função falha se utilizada em geometrias que continham curvas. Resolvido nas versões 1.3.4+.

Disponibilidade: 1.2.2



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

#### Exemplos

Move um ponto 1 grau de latitude

```
SELECT ST_AsText(ST_Translate(ST_GeomFromText('POINT(-71.01 42.37)', 4326), 1, 0)) As ↵
    wgs_transgeomttx;

    wgs_transgeomttx
    -----
    POINT(-70.01 42.37)
```

Move uma linestring 1 grau de longitude e 1/2 grau latitude

```
SELECT ST_AsText(ST_Translate(ST_GeomFromText('LINESTRING(-71.01 42.37,-71.11 42.38)', 4326) ↵
    , 1, 0.5)) As wgs_transgeomttx;

    wgs_transgeomttx
    -----
    LINESTRING(-70.01 42.87,-70.11 42.88)
```

### Move um ponto 3d

```
SELECT ST_AsEWKT(ST_Translate(CAST('POINT(0 0 0)' As geometry), 5, 12,3));
      st_asewkt
      -----
      POINT(5 12 3)
```

### Move uma curva e um ponto

```
SELECT ST_AsText(ST_Translate(ST_Collect('CURVEPOLYGON(CIRCULARSTRING(4 3,3.12 0.878,1 ↔
0,-1.121 5.1213,6 7, 8 9,4 3))','POINT(1 3)'),1,2));
-----
GEOMETRYCOLLECTION(CURVEPOLYGON(CIRCULARSTRING(5 5,4.12 2.878,2 2,-0.121 7.1213,7 9,9 11,5 ↔
5)),POINT(2 5))
```

### Veja também

[ST\\_Affine](#), [ST\\_AsText](#), [ST\\_GeomFromText](#)

## 8.6.34 ST\_TransScale

`ST_TransScale` — Translada uma geometria dando coeficientes e deslocamentos.

### Synopsis

geometry **ST\_TransScale**(geometry geomA, float deltaX, float deltaY, float XFactor, float YFactor);

### Descrição

Translada a geometria utilizando o `deltaX` e `deltaY` args, depois escala ela utilizando o `XFactor`, `YFactor` args, trabalhando somente com 2D.

**Note**

`ST_TransScale(geomA, deltaX, deltaY, XFactor, YFactor)` is short-hand for `ST_Affine(geomA, XFactor, 0, 0, 0, YFactor, 0, 0, 0, 1, deltaX*XFactor, deltaY*YFactor, 0)`.

**Note**

Antes da versão 1.3.4, esta função falha se utilizada em geometrias que continham curvas. Resolvido nas versões 1.3.4+.

Disponibilidade: 1.1.0



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
SELECT ST_AsEWKT(ST_TransScale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 1, 1, 2));
-----
LINESTRING(1.5 6 3,1.5 4 1)

--Buffer a point to get an approximation of a circle, convert to curve and then translate ←
1,2 and scale it 3,4
SELECT ST_AsText(ST_Transscale(ST_LineToCurve(ST_Buffer('POINT(234 567)', 3)),1,2,3,4));
-----

CURVEPOLYGON(CIRCULARSTRING(714 2276,711.363961030679 2267.51471862576,705 ←
2264,698.636038969321 2284.48528137424,714 2276))
```

## Veja também

[ST\\_Affine](#), [ST\\_Translate](#)

## 8.7 Saídas de geometria

### 8.7.1 ST\_AsBinary

**ST\_AsBinary** — Retorna a representação do binário bem conhecido (WKB) da geometria/geografia sem os meta dados SRID.

#### Synopsis

```
bytea ST_AsBinary(geometry g1);
bytea ST_AsBinary(geometry g1, text NDR_or_XDR);
bytea ST_AsBinary(geography g1);
bytea ST_AsBinary(geography g1, text NDR_or_XDR);
```

#### Descrição

Retorna a representação binária bem conhecida da geometria. Existem 2 variantes da função. A primeira variante não pega nenhum parâmetro endian encoding e descumpra a máquina endian. A segunda variante pega um segundo argumento indicando o encoding - usando little-endian ('NDR') ou big-endian ('XDR') encoding.

Isso é útil em cursores binários para empurrar dados do banco de dados sem converter para uma representação de string.



#### Note

O WKB spec não inclui o SRID. Para obter o WKB com o formato SRID use: `ST_AsEWKB`



#### Note

`ST_AsBinary` é o reverso de `ST_GeomFromWKB` para geometria. Use `ST_GeomFromWKB` para converter para uma geometria postgis de uma representação `ST_AsBinary`.





**Veja também.**

[ST\\_GeomFromWKB](#), [ST\\_AsEWKB](#), [ST\\_AsTWKB](#), [ST\\_AsText](#),

## 8.7.2 ST\_AsEncodedPolyline

`ST_AsEncodedPolyline` — Retorna uma Polilinha Encoded de uma geometria `LineString`.

### Synopsis

```
text ST_AsEncodedPolyline(geometry geom, integer precision=5);
```

### Descrição

Retorna a geometria como uma Polilinha Encoded. Este é um formato muito útil se você estiver usando o google maps

Optional `precision` specifies how many decimal places will be preserved in Encoded Polyline. Value should be the same on encoding and decoding, or coordinates will be incorrect.

Disponibilidade: 2.2.0

### Examples

#### Básico

```
SELECT ST_AsEncodedPolyline(GeomFromEWKT('SRID=4326;LINESTRING(-120.2 38.5,-120.95 40.7,-126.453 43.252)'));
--result--
|_p~iF~ps|U_ulLnnqC_mqNvxq`@
```

Use em conjunto com a `linestring` geografia e `segmentize` geografia, e coloque no google maps

```
-- the SQL for Boston to San Francisco, segments every 100 KM
SELECT ST_AsEncodedPolyline(
    ST_Segmentize(
        ST_GeogFromText('LINESTRING(-71.0519 42.4935,-122.4483 37.64)'),
        100000)::geometry) As encodedFlightPath;
```

javascript irá parecer em algo com isso, onde a variável `$` você substitui com o resultado da pesquisa

```
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?libraries=
  geometry"
></script>
<script type="text/javascript">
    flightPath = new google.maps.Polyline({
        path: google.maps.geometry.encoding.decodePath("$encodedFlightPath
        "),
        map: map,
        strokeColor: '#0000CC',
        strokeOpacity: 1.0,
        strokeWeight: 4
    });
</script>
```

**Veja também.**

[ST\\_LineFromEncodedPolyline](#), [ST\\_Segmentize](#)

### 8.7.3 ST\_AsEWKB

ST\_AsEWKB — Retorna a representação binária bem conhecida (WKB) da geometria com os meta dados SRID.

#### Synopsis

```
bytea ST_AsEWKB(geometry g1);
bytea ST_AsEWKB(geometry g1, text NDR_or_XDR);
```

#### Descrição

Retorna a representação binária bem conhecida da geometria com os meta dados SRID. Existem 2 variantes da função. A primeira variante não pega nenhum parâmetro endian encoding e descumpre o pequeno endian. A segunda variante pega um segundo argumento indicando o encoding - usando little-endian ('NDR') ou big-endian ('XDR') encoding.

Isso é útil em cursores binários para empurrar dados do banco de dados sem converter para uma representação de string.



#### Note

O WKB spec não inclui o SRID. Para obter o formato OGC WKB use:



#### Note

ST\_AsEWKB é o reverso de ST\_GeomFromEWKB. Use ST\_GeomFromEWKB para converter uma geometria postgis de uma representação ST\_AsEWKB.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

#### Examples

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_asewkb
```

```
-----
\001\003\000\000 \346\020\000\000\001\000
\000\000\005\000\000\000\000
\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000
\000\000\360?\000\000\000\000\000\000\360?
\000\000\000\000\000\000\360?\000\000\000\000\000
\000\360?\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000
(1 row)
```

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326), 'XDR');
      st_asewkb
-----
\000 \000\000\003\000\000\020\346\000\000\000\001\000\000\000\005\000\000\000\000\
000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\
\360\000\000\000\000\000\000?\360\000\000\000\000\000\000?\360\000\000\000\000
\000\000?\360\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
```

**Veja também.**

[ST\\_AsBinary](#), [ST\\_AsEWKT](#), [ST\\_AsText](#), [ST\\_GeomFromEWKT](#), [ST\\_SRID](#)

### 8.7.4 ST\_AsEWKT

ST\_AsEWKT — Retorna a representação de texto bem conhecida (WKT) da geometria com os meta dados SRID.

#### Synopsis

```
text ST_AsEWKT(geometry g1);
text ST_AsEWKT(geography g1);
```

#### Descrição

Retorna a representação de texto bem conhecida da geometria prefixada com o SRID.



#### Note

O WKT spec não inclui o SRID. Para obter o formato OGC WKT use: [ST\\_AsText](#)



O formato WKT não mantém a precisão, então para prevenir truncamento flutuante, use formato [ST\\_AsBinary](#) ou [ST\\_AsEWKB](#) para o transporte.



#### Note

ST\_AsEWKT é o reverso de [ST\\_GeomFromEWKT](#). Use [ST\\_GeomFromEWKT](#) para converter para uma geometria de uma representação ST\_AsEWKT.

Melhorias: 2.0.0 suporte para geografia, superfícies poliédricas, triângulos e TIN foi introduzido.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



Versão 1: `ST_AsGeoJSON(geom) / precisão=15 versão=1 opções=0`

Versão 2: `ST_AsGeoJSON(geom, precision) / versão=1 opções=0`

Versão 3: `ST_AsGeoJSON(geom, precisão, opções) / versão=1`

Versão 4: `ST_AsGeoJSON(gj_versão, geom) / precisão=15 opções=0`

Versão 5: `ST_AsGeoJSON(gj_versão, geom, precisão) /opções=0`

Versão 6: `ST_AsGeoJSON(gj_versão, geom, precisão,opções)`

Disponibilidade: 1.3.4

Disponibilidade: 1.5.0 suporte para geografia foi introduzido.

Alterações: 2.0.0 suporte padrão args e args nomeados.



This function supports 3d and will not drop the z-index.

## Examples

O formato GeoJSON é, geralmente, mais eficiente que outros formatos para o uso em mapeamentos ajax. Um cliente popular javascript que suporta ele é o Open Layers. Exemplo deste use é: [OpenLayers GeoJSON Example](#)

```
SELECT ST_AsGeoJSON(the_geom) from fe_edges limit 1;
                st_asgeojson
-----
{"type":"MultiLineString","coordinates":[[[-89.734634999999997,31.492072000000000],
[-89.734955999999997,31.492237999999997]]]}
(1 row)
--3d point
SELECT ST_AsGeoJSON('LINESTRING(1 2 3, 4 5 6)');

st_asgeojson
-----
{"type":"LineString","coordinates":[[1,2,3],[4,5,6]]}
```

## 8.7.6 ST\_AsGML

`ST_AsGML` — Retorna a geometria como uma versão GML com 2 ou 3 elementos.

### Synopsis

`text ST_AsGML(geometry geom, integer maxdecimaldigits=15, integer options=0);`

`text ST_AsGML(geography geog, integer maxdecimaldigits=15, integer options=0);`

`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);`

### Descrição

Retorna a geometria como um elemento Geography Markup Language (GML). Esta versão de parâmetro, se especificada, pode ser 2 ou 3. Se nenhuma versão de parâmetro estiver especificada, então, o padrão é assumido para ser 2. O argumento de precisão pode ser usado para reduzir o número máximo de casas decimais (`maxdecimaldigits`) usado na saída (padrão 15).

GML 2 refere-se a versão 2.1.2 , GML 3 para a versão 3.1.1

O argumento "opções" é um bitfield. Ele poderia ser usado para definir o tipo de saída CRS na saída GML, e para declarar dados como lat/lon:

- 0: GML Short CRS (ex: EPSG:4326), valor padrão
- 1: GML Long CRS (ex: urn:ogc:def:crs:EPSG::4326)
- 2: Para GML 3 somente, remove srsDimension atribuída da saída.
- 4: Para GML 3 somente, use <LineString> em vez de <Curve> tag para linhas.
- 16: Declara que dados são lat/lon (ex: srid=4326). O padrão é supor que os dados são planos. Esta opção é útil apenas para saída GML 3.1.1, relacionada a ordem do eixo. Então, se você configurá-la, ela irá trocar as coordenadas, deixando a ordem sendo lat lon em vez do banco de dados.
- 32: Gera a caixa da geometria (envelope).

O argumento 'namespace prefix' pode ser usado para especificar um namespace prefix personalizado ou nenhum prefixo (se vazio). Se nulo ou omitido, o prefixo 'gml' é usado

Disponibilidade: 1.3.2

Disponibilidade: 1.5.0 suporte para geografia foi introduzido.

Melhorias: 2.0.0 prefixo suportado foi introduzido. A opção 4 para o GML3 foi introduzida para permitir a utilização da LineString em vez da tag Curva para linhas. O suporte GML3 para superfícies poliédricas e TINS foi introduzidos. A Opção 32 foi introduzida para gerar a caixa.

Alterações: 2.0.0 use argumentos nomeados por padrão

Melhorias: 2.1.0 suporte para id foi introduzido, para GML 3.



#### Note

Somente a versão 3+ de ST\_AsGML suporta superfícies poliédricas e TINS.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

#### Exemplos: Versão 2

```
SELECT ST_AsGML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
      st_asgml
-----
      <gml:Polygon srsName="EPSG:4326"
><gml:outerBoundaryIs
><gml:LinearRing
><gml:coordinates
>0,0 0,1 1,1 1,0 0,0</gml:coordinates
></gml:LinearRing
></gml:outerBoundaryIs
></gml:Polygon
>
```

**Exemplos: Versão 3**

```
-- Flip coordinates and output extended EPSG (16 | 1)--
SELECT ST_AsGML(3, ST_GeomFromText('POINT(5.234234233242 6.34534534534)',4326), 5, 17);
      st_asgml
      -----
      <gml:Point srsName="urn:ogc:def:crs:EPSG::4326"
><gml:pos
>6.34535 5.23423</gml:pos
></gml:Point
>
```

```
-- Output the envelope (32) --
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 32);
      st_asgml
      -----
      <gml:Envelope srsName="EPSG:4326">
        <gml:lowerCorner
>1 2</gml:lowerCorner>
        <gml:upperCorner
>10 20</gml:upperCorner>
      </gml:Envelope
>
```

```
-- Output the envelope (32) , reverse (lat lon instead of lon lat) (16), long srs (1)= 32 | ←
16 | 1 = 49 --
SELECT ST_AsGML(3, ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 49);
      st_asgml
      -----
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
        <gml:lowerCorner
>2 1</gml:lowerCorner>
        <gml:upperCorner
>20 10</gml:upperCorner>
      </gml:Envelope
>
```

```
-- Polyhedral Example --
SELECT ST_AsGML(3, ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0) ←
),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )' ));
      st_asgml
      -----
      <gml:PolyhedralSurface>
<gml:polygonPatches>
  <gml:PolygonPatch>
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList srsDimension="3"
>0 0 0 0 0 1 0 1 1 0 1 0 0 0 0</gml:posList>
      </gml:LinearRing>
    </gml:exterior>
  </gml:PolygonPatch>
  <gml:PolygonPatch>
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList srsDimension="3"
>0 0 0 0 1 0 1 1 0 1 0 0 0 0 0</gml:posList>
      </gml:LinearRing>
```

```

        </gml:exterior>
    </gml:PolygonPatch>
    <gml:PolygonPatch>
        <gml:exterior>
            <gml:LinearRing>
                <gml:posList srsDimension="3"
>0 0 0 1 0 0 1 0 1 0 0 1 0 0 0</gml:posList>
                </gml:LinearRing>
            </gml:exterior>
        </gml:PolygonPatch>
    <gml:PolygonPatch>
        <gml:exterior>
            <gml:LinearRing>
                <gml:posList srsDimension="3"
>1 1 0 1 1 1 1 0 1 1 0 0 1 1 0</gml:posList>
                </gml:LinearRing>
            </gml:exterior>
        </gml:PolygonPatch>
    <gml:PolygonPatch>
        <gml:exterior>
            <gml:LinearRing>
                <gml:posList srsDimension="3"
>0 1 0 0 1 1 1 1 1 1 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:polygons>
    </gml:PolyhedralSurface
>

```

**Veja também.**

[ST\\_GeomFromGML](#)

### 8.7.7 ST\_AsHEXEWKB

**ST\_AsHEXEWKB** — Retorna uma geometria no formato HEXEWKB (como texto) usando little-endian (NDR) ou big-endian (XDR) encoding.

#### Synopsis

```
text ST_AsHEXEWKB(geometry g1, text NDRorXDR);
text ST_AsHEXEWKB(geometry g1);
```

#### Descrição

Retorna uma geometria no formato HEXEWKB (como texto) usando little-endian (NDR) ou big-endian (XDR) encoding. Se nenhum encoding estiver especificado, então o NDR é usado.





**Note**

Melhorias: 2.0.0 - Adiciona namespace prefixo. O padrão é não ter nenhum prefixo

**Note**

Alterações: 2.0.0 - suporte padrão args e suporta args nomeados

**Note**

A saída AsKML não funcionará com geometrias que não possuem um SRID



This function supports 3d and will not drop the z-index.

**Examples**

```
SELECT ST_AsKML(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

      st_askml
      -----
      <Polygon
><outerBoundaryIs
><LinearRing
><coordinates
>0,0 0,1 1,1 1,0 0,0</coordinates
></LinearRing
></outerBoundaryIs
></Polygon>

--3d linestring
SELECT ST_AsKML('SRID=4326;LINESTRING(1 2 3, 4 5 6)');
      <LineString
><coordinates
>1,2,3 4,5,6</coordinates
></LineString>
```

**Veja também.**

[ST\\_AsSVG](#), [ST\\_AsGML](#)

**8.7.9 ST\_AsLatLonText**

ST\_AsLatLonText — Retorna a representação de Graus, Minutos, Segundos do ponto dado.

**Synopsis**

```
text ST_AsLatLonText(geometry pt, text format=’');
```

## Descrição

Returns the Degrees, Minutes, Seconds representation of the point.



### Note

É suposto que o ponto é uma projeção lat/lon. As coordenadas X (lon) e Y (lat), são normalizadas na saída para o alcance "normal" (-180 to +180 para lon, -90 para +90 para lat).

O texto parâmetro é um formato string que contém o formato do texto resultante, parecido com uma string de formato data. Tokens válidos são "D" para graus, "M" para minutos, "S" para segundos e "C" para direções cardiais (NSLO). Os tokens DMS podem se repetir para indicar a largura e precisão desejadas ("SSS.SSSS" significa "1.0023").

"M", "S", e "C" são opcionais. Se "C" estiverem omitidas, os graus são mostrados com um "-" se sul ou oeste. Se "S" estiver omitido, os minutos serão mostrados como decimais com com tanta precisão de dígitos quanto você especificar. Se "M" também estiver omitido, os graus serão mostrados como decimais com tanta precisão de dígitos quanto você especificar.

Se a string formato for omitida (ou tiver tamanho zero) um formato padrão será usado.

Disponibilidade: 2.0

## Examples

Formato padrão.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)'));
      st_aslatlon
-----
2\textdegree{}19'29.928"S 3\textdegree{}14'3.243"W
```

Fornecendo um formato (o mesmo do padrão).

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D\textdegree{}M'S.SSS"C'));
      st_aslatlon
-----
2\textdegree{}19'29.928"S 3\textdegree{}14'3.243"W
```

Outros caracteres além de D, M, S, C e . são somente passados.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D degrees, M minutes, S seconds to the C'));
      st_aslatlon
-----
2 degrees, 19 minutes, 30 seconds to the S 3 degrees, 14 minutes, 3 seconds to the W
```

Graus assinados em vez de direções cardiais.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D\textdegree{}M'S.SSS"));
      st_aslatlon
-----
-2\textdegree{}19'29.928" -3\textdegree{}14'3.243"
```

Graus decimais.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D.DDDD degrees C'));
      st_aslatlon
-----
2.3250 degrees S 3.2342 degrees W
```

Valores excessivamente grandes são normalizados.

```
SELECT (ST_AsLatLonText('POINT (-302.2342342 -792.32498)'));
      st_aslatlon
-----
72\textdegree{}19'29.928"S 57\textdegree{}45'56.757"E
```

### 8.7.10 ST\_AsSVG

ST\_AsSVG — Retorna uma geometria em dados SVG path, dado um objeto de geometria ou geografia.

#### Synopsis

```
text ST_AsSVG(geometry geom, integer rel=0, integer maxdecimaldigits=15);
text ST_AsSVG(geography geog, integer rel=0, integer maxdecimaldigits=15);
```

#### Descrição

Retorna a geometria como dados Scalar Vector Graphics (SVG). Use 1 como segundo argumento para ter os dados path implementados em termo de movimentos relacionados, o padrão (ou 0) utiliza movimento absolutos. O terceiro argumento pode ser usado para reduzir o máximo número de dígitos decimais usados na saída (padrão 15). Geometrias pontuais, serão renderizadas como cx/cy quando o argumento 'rel' for 0, x/y quando 'rel' for 1. Geometrias multipontuais são delimitadas por vírgulas (","). As geometrias GeometryCollection são delimitadas por ponto e vírgula (";").



#### Note

Disponibilidade: 1.2.2. Disponibilidade: 1.4.0 Alterado em PostGIS 1.4.0 para incluir comando L em path absoluto para entrar em conformidade com <http://www.w3.org/TR/SVG/paths.html#PathDataBNF>

Alterações: 2.0.0 para usar args padrão e suporta args nomeados

#### Examples

```
SELECT ST_AsSVG(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
      st_assvg
-----
M 0 0 L 0 -1 1 -1 1 0 Z
```

### 8.7.11 ST\_AsText

ST\_AsText — Retorna a representação de texto bem conhecida (WKT) da geometria/geografia sem os meta dados do SRID.

#### Synopsis

```
bytea ST_AsBinary(geometry g1);
bytea ST_AsBinary(geometry g1, text NDR_or_XDR);
bytea ST_AsBinary(geography g1);
bytea ST_AsBinary(geography g1, text NDR_or_XDR);
```



Veja também.

[ST\\_AsBinary](#), [ST\\_AsEWKB](#), [ST\\_AsEWKT](#), [ST\\_GeomFromText](#)

### 8.7.12 ST\_AsTWKB

ST\_AsTWKB — Retorna a geometria como TWKB, também conhecido como "Tiny Well-Known Binary"

#### Synopsis

bytea **ST\_AsTWKB**(geometry g1, integer decimaldigits\_xy=0, integer decimaldigits\_z=0, integer decimaldigits\_m=0, boolean include\_sizes=false, boolean include\_bounding\_boxes=false);

bytea **ST\_AsTWKB**(geometry[] geometries, bigint[] unique\_ids, integer decimaldigits\_xy=0, integer decimaldigits\_z=0, integer decimaldigits\_m=0, boolean include\_sizes=false, boolean include\_bounding\_boxes=false);

#### Descrição

Retorna a geometria no formato TWKB (Tiny Well-Known Binary). TWKB é um **compressed binary format** com foco em minimizar o tamanho da saída.

Os parâmetros de dígitos decimais controlam quanta precisão está armazenada na saída. Por padrão, valores são arredondados para a unidade mais próxima antes de encoding. Por exemplo: um valor de 1 implica que o primeiro dígito a direita do ponto decimal será preservado.

Os tamanhos e os parâmetros das caixas limitadoras controlam onde as informações opcionais sobre o tamanho do encoding do objeto e os limites do objeto estão incluídas na saída. Por padrão elas não estão. Não as inclua a menos que o software do seu cliente tenha um uso para elas, como elas só ocupa espaço (e economizar espaço é o objeto do TWKB).

A forma arranjo entrada da função é usada para converter uma coleção de geometrias e identificadores únicos em uma coleção TWKB que preserva os identificadores. Isto é útil para clientes que esperam desempacotar uma coleção e acessar informações futuras sobre os objetos que estão dentro. Você pode criar os arranjos usando a função [array\\_agg](#). Os outros parâmetros funcionam da mesma forma para o formato simples da função.



#### Note

O formato de especificação está disponível online em <https://github.com/TWKB/Specification>, e o código para construir um cliente JavaScript pode ser encontrado em <https://github.com/TWKB/twkb.js>.

Enhanced: 2.4.0 memory and speed improvements.

Disponibilidade: 2.2.0

#### Examples

```
SELECT ST_AsTWKB('LINESTRING(1 1,5 5)::geometry);
          st_astwkb
-----
\x0200020202020808
```

Para criar um objeto TWKB agregado, incluir identificadores agrega as geometrias e objetos desejado primeiro, utilizando "array\_agg()", então, utilize a função TWKB apropriada.

```
SELECT ST_AsTWKB(array_agg(geom), array_agg(gid)) FROM mytable;
          st_astwkb
-----
\x040402020400000202
```

Veja também.

[ST\\_GeomFromTWKB](#), [ST\\_AsEWKB](#), [ST\\_AsEWKT](#), [ST\\_GeomFromText](#)

### 8.7.13 ST\_AsX3D

ST\_AsX3D — Retorna uma geometria em X3D nó xml formato do elemento: ISO-IEC-19776-1.2-X3DEncodings-XML

#### Synopsis

```
text ST_AsX3D(geometry g1, integer maxdecimaldigits=15, integer options=0);
```

#### Descrição

Retorna uma geometria como um elemento nó formatado X3D xml <http://www.web3d.org/standards/number/19776-1>. Se `maxdecimal` (precisão) não estiver especificada, então, leva para 15.

#### Note



Existem vários motivos para traduzir as geometrias PostGIS para X3D já que os tipos de geometria X3D não mapeiam diretamente para os tipos de geometria do PostGIS e alguns tipos X3D mais novos, que podem ser os melhores mapeadores que estávamos evitando já que a maioria das ferramentas renderizadoras não suportam eles. Sinta-se livre para postar um comentário se você tiver ideias de como podemos permitir as pessoas a indicarem seus mapeamentos preferidos.

Abaixo está como nós mapeamos os tipos 2D/3D do PostGIS para os tipos X3D, no momento

O argumento 'opções' é um bitfield. Para o PostGIS 2.2+, isto é usado para indicar onde representar as coordenadas atuais com o nó X3D GeoCoordinates Geospatial e, além disso, onde derrubar os eixos x/y. Por padrão, ST\_AsX3D gera na forma de banco de dados (long,lat or X,Y), mas X3D de lat/lon, y/x podem ser preferidos.

- 0: X/Y na ordem de banco de dados (ex: long/lat = X,Y é a ordem padrão de banco de dados), valor padrão e coordenadas não-espaciais (somente coordenada tag antiga).
- 1: Lançar X e Y. Se usado em conjunção com a opção de trocar a geocoordenada, então, a saída será "latitude\_first" e as coordenadas serão lançadas também.
- 2: Gera coordenadas no GeoSpatial GeoCoordinates. Esta opção lançará um erro se as geometrias não estiverem na WGS 84 long lat (srid: 4326). Este é o único tipo GeoCoordinate suportado. [Refer to X3D specs specifying a spatial reference system..](#) Saída padrão será: `GeoCoordinate geoSystem='GD' 'WE' 'longitude_first''`. If you prefer the X3D default of `GeoCoordinate geoSystem='GD' 'WE' 'latitude_first''` use  $(2 + 1) = 3$

| Tipo PostGIS                            | Tipo 2D X3D                                   | Tipo 3D X3D                                                             |
|-----------------------------------------|-----------------------------------------------|-------------------------------------------------------------------------|
| LINestring                              | ainda não foi implementado - será PoliLinha2D | LineSet                                                                 |
| MULTILINestring                         | ainda não foi implementado - será PoliLinha2D | IndexedLineSet                                                          |
| MULTIPONTO                              | Poliponto2D                                   | PointSet                                                                |
| PONTO                                   | gera as coordenadas delimitadas pelo espaço   | gera as coordenadas delimitadas pelo espaço                             |
| (MULTI) POLÍGONO, SUPERFÍCIE POLIÉDRICA | Marcação X3D inválida                         | IndexedFaceSet (anéis interiores atualmente gerados como outro faceset) |
| TIN                                     | TriangleSet2D (ainda não implementado)        | IndexedTriangleSet                                                      |

**Note**

O suporte para geometrias 2D ainda não está completo. Os anéis interiores apenas desenhados como polígonos separados. Estamos trabalhando nisto.

Muitos avanços acontecendo no espaço 3D particularmente com [X3D Integration with HTML5](#)

Existe uma ótima fonte de visualizador X3D que você pode usar para ver as geometrias renderizadas. Free Wrl <http://freewrl.sourceforge.net>. binários para Mac, Linux, and Windows. Use FreeWRL\_Launcher compactados para visualizar as geometrias.

Veja também [PostGIS minimalist X3D viewer](#) que utiliza esta função e [x3dDom html/js fonte aberta toolkit](#).

Disponibilidade: 2.0.0: ISO-IEC-19776-1.2-X3DEncodings-XML

Melhorias: 2.2.0: Suporte para GeoCoordinates e eixos (x/y, long/lat) lançando. Observe as opções para mais detalhes.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

**Exemplo: Cria um documento X3D completamente funcional - Isto irá gerar um cubo visível no FreeWrl e outros visualizadores X3D.**

```
SELECT '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d ←
-3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
        <Appearance>
          <Material emissiveColor='0 0 1' />
        </Appearance>
      </Shape>
    </Transform>
  </Scene>
</X3D>
>' As x3ddoc;

          x3ddoc
          -----
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d ←
-3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
        <Appearance>
          <Material emissiveColor='0 0 1' />
        </Appearance>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```



```

    <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
>

```

### Exemplo: Um octógono elevado 3 unidades e com precisão decimal de 6

```

SELECT ST_AsX3D(
ST_Translate(
  ST_Force_3d(
    ST_Buffer(ST_Point(10,10),5, 'quad_segs=2')), 0,0,
    3)
,6) As x3dfrag;

x3dfrag
-----
<IndexedFaceSet coordIndex="0 1 2 3 4 5 6 7">
  <Coordinate point="15 10 3 13.535534 6.464466 3 10 5 3 6.464466 6.464466 3 5 10 3 ←
    6.464466 13.535534 3 10 15 3 13.535534 13.535534 3 " />
</IndexedFaceSet
>

```

### Exemplo: TIN

```

SELECT ST_AsX3D(ST_GeomFromEWKT('TIN (((
  0 0 0,
  0 0 1,
  0 1 0,
  0 0 0
)), ((
  0 0 0,
  0 1 0,
  1 1 0,
  0 0 0
))
)')) As x3dfrag;

x3dfrag
-----
<IndexedTriangleSet index='0 1 2 3 4 5'
><Coordinate point='0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0' /></IndexedTriangleSet
>

```

### Exemplo: Multilinestring fechada (o limite de um polígono com buracos)

```

SELECT ST_AsX3D(
  ST_GeomFromEWKT('MULTILINESTRING((20 0 10,16 -12 10,0 -16 10,-12 -12 ←
    10,-20 0 10,-12 16 10,0 24 10,16 16 10,20 0 10),
    (12 0 10,8 8 10,0 12 10,-8 8 10,-8 0 10,-8 -4 10,0 -8 10,8 -4 10,12 0 10))')
)

```

```

) As x3dfrag;

          x3dfrag
          -----
<IndexedLineSet coordIndex='0 1 2 3 4 5 6 7 0 -1 8 9 10 11 12 13 14 15 8'>
  <Coordinate point='20 0 10 16 -12 10 0 -16 10 -12 -12 10 -20 0 10 -12 16 10 0 24 10 16 ←
    16 10 12 0 10 8 8 10 0 12 10 -8 8 10 -8 0 10 -8 -4 10 0 -8 10 8 -4 10 ' />
</IndexedLineSet
>

```

### 8.7.14 ST\_GeoHash

ST\_GeoHash — Retorna uma representação GeoHash da geometria.

#### Synopsis

```
text ST_GeoHash(geometry geom, integer maxchars=full_precision_of_point);
```

#### Descrição

Retorna uma representação GeoHash (<http://en.wikipedia.org/wiki/Geohash>) da geometria. Uma GeoHash codifica um ponto dentro de uma forma de texto que é classificável e pesquisável baseado em prefixos. Um GeoHash menor é uma representação menos precisa de um ponto. Pode ser pensado como uma caixa que contém o ponto atual.

Se nenhum `maxchars` é especificado, `ST_GeoHash` retorna um GeoHash baseado na precisão completa do tipo de geometria de entrada. Os pontos retornam um GeoHash com 20 caracteres de precisão (o suficiente para segurar a precisão dupla da entrada). Or outros tipos retornam um GeoHash com uma quantidade de precisão variável, baseada no tamanho da característica. Traços maiores são representados com menos precisão, traços menores com mais precisão. A ideia é que a caixa sugerida pelo GeoHash sempre conterá o traço de entrada.

Se `maxchars` está especificado, `ST_GeoHash` retorna um GeoHash com no máximo muitos caracteres, então uma representação de precisão menor da geometria de entrada. Para não pontos, o ponto de início do cálculo é o centro da caixa limitadora da geometria.

Disponibilidade: 1.4.0



#### Note

ST\_GeoHash não funcionará com geometrias que não estão nas coordenadas geográficas (lon/lat).



This method supports Circular Strings and Curves

#### Examples

```

SELECT ST_GeoHash(ST_SetSRID(ST_MakePoint(-126,48),4326));

          st_geohash
          -----
c0w3hf1s70w3hf1s70w3

SELECT ST_GeoHash(ST_SetSRID(ST_MakePoint(-126,48),4326),5);

          st_geohash

```

```
-----
c0w3h
```

**Veja também.**

[ST\\_GeomFromGeoHash](#)

### 8.7.15 ST\_AsGeobuf

ST\_AsGeobuf — Return a Geobuf representation of a set of rows.

#### Synopsis

```
bytea ST_AsGeobuf(anyelement set row);
bytea ST_AsGeobuf(anyelement row, text geom_name);
```

#### Descrição

Return a Geobuf representation (<https://github.com/mapbox/geobuf>) of a set of rows corresponding to a FeatureCollection. Every input geometry is analyzed to determine maximum precision for optimal storage. Note that Geobuf in its current form cannot be streamed so the full output will be assembled in memory.

row row data with at least a geometry column.

geom\_name is the name of the geometry column in the row data. If NULL it will default to the first found geometry column.

Availability: 2.4.0

#### Examples

```
SELECT encode(ST_AsGeobuf(q, 'geom'), 'base64')
         FROM (SELECT ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))') AS geom) AS q;
 st_asgeobuf
-----
GAAiEAoOCgwIBBoIAAAAAgIAAAE=
```

### 8.7.16 ST\_AsMVTGeom

ST\_AsMVTGeom — Transform a geometry into the coordinate space of a [Mapbox Vector Tile](#).

#### Synopsis

```
text ST_AsSVG(geometry geom, integer rel=0, integer maxdecimaldigits=15);
text ST_AsSVG(geography geog, integer rel=0, integer maxdecimaldigits=15);
```

## Descrição

Transform a geometry into the coordinate space of a [Mapbox Vector Tile](#) of a set of rows corresponding to a Layer. Makes best effort to keep and even correct validity and might collapse geometry into a lower dimension in the process.

`geom` is the geometry to transform.

`bounds` is the geometric bounds of the tile contents without buffer.

`extent` is the tile extent in tile coordinate space as defined by the [specification](#). If NULL it will default to 4096.

`buffer` is the buffer distance in tile coordinate space to optionally clip geometries. If NULL it will default to 256.

`clip_geom` is a boolean to control if geometries should be clipped or encoded as is. If NULL it will default to true.

Availability: 2.4.0

## Examples

```
SELECT ST_AsText(ST_AsMVTGeom(
    ST_GeomFromText('POLYGON ((0 0, 10 0, 10 5, 0 -5, 0 0))'),
    ST_MakeBox2D(ST_Point(0, 0), ST_Point(4096, 4096)),
    4096, 0, false));
           st_astext
-----
MULTIPOLYGON(((5 4096,10 4096,10 4091,5 4096)),((5 4096,0 4096,0 4101,5 4096)))
```

### 8.7.17 ST\_AsMVT

`ST_AsMVT` — Return a [Mapbox Vector Tile](#) representation of a set of rows.

#### Synopsis

```
text ST_AsKML(geometry geom, integer maxdecimaldigits=15);
text ST_AsKML(geography geog, integer maxdecimaldigits=15);
text ST_AsKML(integer version, geometry geom, integer maxdecimaldigits=15, text nprefix=NULL);
text ST_AsKML(integer version, geography geog, integer maxdecimaldigits=15, text nprefix=NULL);
```

#### Descrição

Return a [Mapbox Vector Tile](#) representation of a set of rows corresponding to a Layer. Multiple calls can be concatenated to a tile with multiple Layers. Geometry is assumed to be in tile coordinate space and valid as per [specification](#). Typically `ST_AsMVTGeom` can be used to transform geometry into tile coordinate space. Other row data will be encoded as attributes.

The [Mapbox Vector Tile](#) format can store features with a different set of attributes per feature. To make use of this feature supply a JSONB column in the row data containing Json objects one level deep. The keys and values in the object will be parsed into feature attributes.



#### Important

Do not call with a `GEOMETRYCOLLECTION` as an element in the row. However you can use `ST_AsMVTGeom` to prep a geometry collection for inclusion.

---

`row` row data with at least a geometry column.

`name` is the name of the Layer. If NULL it will use the string "default".

`extent` is the tile extent in screen space as defined by the specification. If NULL it will default to 4096.

`geom_name` is the name of the geometry column in the row data. If NULL it will default to the first found geometry column.

Melhorias: 2.1.0 suporte para id foi introduzido, para GML 3.

Availability: 2.4.0

## Examples

```
SELECT ST_AsmVT(q, 'test', 4096, 'geom') FROM (SELECT 1 AS c1,
  ST_AsmVTGeom(ST_GeomFromText('POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 ←
    35, 30 20, 20 30))'),
  ST_MakeBox2D(ST_Point(0, 0), ST_Point(4096, 4096)), 4096, 0, false) AS geom) AS q;
-----
\ ←
  x1a320a0474657374121d12020000180322150946ec3f1a14453b0a09280f091413121e09091e0f1a02633122022801
```

Veja também.

[ST\\_AsmVTGeom](#)

## 8.8 Operadores

### 8.8.1 &&

`&&` — Retorna VERDADE se a caixa limitadora 2D de A intersecta a caixa limitadora 2D de B.

#### Synopsis

```
boolean &&( geometry A , geometry B );
boolean &&( geography A , geography B );
```

#### Descrição

O operador `&&` retorna VERDADE se a caixa limitadora 2D da geometria A intersecta a caixa limitadora 2D da geometria B.



#### Note

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

Melhorias: 2.0.0 suporte a superfícies poliédricas foi introduzido.

Disponibilidade: 1.5.0 Suporte para geografia foi introduzido



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 && tbl2.column2 AS overlaps
FROM ( VALUES
      (1, 'LINESTRING(0 0, 3 3)::geometry),
      (2, 'LINESTRING(0 1, 0 5)::geometry)) AS tbl1,
( VALUES
      (3, 'LINESTRING(1 2, 4 6)::geometry)) AS tbl2;
```

column1	column1	overlaps
1	3	t
2	3	f

(2 rows)

## Veja também.

[|&>](#), [&>](#), [&<|](#), [&<](#), [~](#), [@](#)

## 8.8.2 &&(geometry,box2df)

`&&(geometry,box2df)` — Returns TRUE if a geometry's (cached) 2D bounding box intersects a 2D float precision bounding box (BOX2DF).

### Synopsis

boolean `&&( geometry A , box2df B );`

### Descrição

The `&&` operator returns TRUE if the cached 2D bounding box of geometry A intersects the 2D bounding box B, using float precision. This means that if B is a (double precision) box2d, it will be internally converted to a float precision 2D bounding box (BOX2DF)



#### Note

This operand is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

```
SELECT ST_MakePoint(1,1) && ST_MakeBox2D(ST_MakePoint(0,0), ST_MakePoint(2,2)) AS overlaps;
```

overlaps
t

(1 row)

**Veja também.**

[&&\(box2df,geometry\)](#), [&&\(box2df,box2df\)](#), [~\(geometry,box2df\)](#), [~\(box2df,geometry\)](#), [~\(box2df,box2df\)](#), [@\(geometry,box2df\)](#), [@\(box2df,geometry\)](#), [@\(box2df,box2df\)](#)

**8.8.3 &&(box2df,geometry)**

`&&(box2df,geometry)` — Returns `TRUE` if a 2D float precision bounding box (BOX2DF) intersects a geometry's (cached) 2D bounding box.

**Synopsis**

boolean `&&( box2df A , geometry B );`

**Descrição**

The `&&` operator returns `TRUE` if the 2D bounding box A intersects the cached 2D bounding box of geometry B, using float precision. This means that if A is a (double precision) `box2d`, it will be internally converted to a float precision 2D bounding box (BOX2DF)

**Note**

This operand is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

**Exemplos**

```
SELECT ST_MakeBox2D(ST_MakePoint(0,0), ST_MakePoint(2,2)) && ST_MakePoint(1,1) AS overlaps;

 overlaps
-----
 t
(1 row)
```

**Veja também.**

[&&\(geometry,box2df\)](#), [&&\(box2df,box2df\)](#), [~\(geometry,box2df\)](#), [~\(box2df,geometry\)](#), [~\(box2df,box2df\)](#), [@\(geometry,box2df\)](#), [@\(box2df,geometry\)](#), [@\(box2df,box2df\)](#)

**8.8.4 &&(box2df,box2df)**

`&&(box2df,box2df)` — Returns `TRUE` if two 2D float precision bounding boxes (BOX2DF) intersect each other.

## Synopsis

boolean **&&**( box2df A , box2df B );

## Descrição

The **&&** operator returns `TRUE` if two 2D bounding boxes A and B intersect each other, using float precision. This means that if A (or B) is a (double precision) `box2d`, it will be internally converted to a float precision 2D bounding box (`BOX2DF`)



### Note

This operator is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

```
SELECT ST_MakeBox2D(ST_MakePoint(0,0), ST_MakePoint(2,2)) && ST_MakeBox2D(ST_MakePoint(1,1) ↔
, ST_MakePoint(3,3)) AS overlaps;
```

```
overlaps
-----
t
(1 row)
```

## Veja também.

[&&\(geometry,box2df\)](#), [&&\(box2df,geometry\)](#), [~\(geometry,box2df\)](#), [~\(box2df,geometry\)](#), [~\(box2df,box2df\)](#), [@\(geometry,box2df\)](#), [@\(box2df,geometry\)](#), [@\(box2df,box2df\)](#)

## 8.8.5 &&&

**&&&** — Retorna `VERDADE` se a caixa limitadora n-D de A intersecta a caixa limitadora n-D de B.

## Synopsis

boolean **&&&**( geometry A , geometry B );

## Descrição

O operador **&&&** retorna `VERDADE` se a caixa limitadora n-D da geometria A intersecta a caixa limitadora n-D da geometria B.



### Note

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.



Disponibilidade: 2.0.0



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.

### Exemplos: LineStrings 3D

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &&& tbl2.column2 AS overlaps_3d,
       tbl1.column2 && tbl2.column2 AS overlaps_2d
FROM ( VALUES
      (1, 'LINESTRING Z(0 0 1, 3 3 2)::geometry),
      (2, 'LINESTRING Z(1 2 0, 0 5 -1)::geometry)) AS tbl1,
 ( VALUES
      (3, 'LINESTRING Z(1 2 1, 4 6 1)::geometry)) AS tbl2;
```

column1	column1	overlaps_3d	overlaps_2d
1	3	t	t
2	3	f	t

### Exemplos: LineStrings 3M

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &&& tbl2.column2 AS overlaps_3zm,
       tbl1.column2 && tbl2.column2 AS overlaps_2d
FROM ( VALUES
      (1, 'LINESTRING M(0 0 1, 3 3 2)::geometry),
      (2, 'LINESTRING M(1 2 0, 0 5 -1)::geometry)) AS tbl1,
 ( VALUES
      (3, 'LINESTRING M(1 2 1, 4 6 1)::geometry)) AS tbl2;
```

column1	column1	overlaps_3zm	overlaps_2d
1	3	t	t
2	3	f	t

Veja também.

**&&**

### 8.8.6 &&&(geometry,gidx)

**&&&(geometry,gidx)** — Returns TRUE if a geometry's (cached) n-D bounding box intersects a n-D float precision bounding box (GIDX).

#### Synopsis

boolean **&&&**( geometry A , gidx B );

## Descrição

The `&&&` operator returns `TRUE` if the cached n-D bounding box of geometry A intersects the n-D bounding box B, using float precision. This means that if B is a (double precision) `box3d`, it will be internally converted to a float precision 3D bounding box (GIDX)



### Note

This operator is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.

## Exemplos

```
SELECT ST_MakePoint(1,1,1) &&& ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) AS ↔
       overlaps;
```

```
overlaps
-----
t
(1 row)
```

## Veja também.

[&&&\(gidx,geometry\)](#), [&&&\(gidx,gidx\)](#)

### 8.8.7 &&&(gidx,geometry)

`&&&(gidx,geometry)` — Returns `TRUE` if a n-D float precision bounding box (GIDX) intersects a geometry's (cached) n-D bounding box.

## Synopsis

```
boolean &&&( gidx A , geometry B );
```

## Descrição

The `&&&` operator returns `TRUE` if the n-D bounding box A intersects the cached n-D bounding box of geometry B, using float precision. This means that if A is a (double precision) `box3d`, it will be internally converted to a float precision 3D bounding box (GIDX)

**Note**

This operator is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.

**Exemplos**

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_MakePoint(1,1,1) AS ↔
    overlaps;

overlaps
-----
t
(1 row)
```

**Veja também.**

[&&&\(geometry,gidx\), &&&\(gidx,gidx\)](#)

**8.8.8 &&&(gidx,gidx)**

[&&&\(gidx,gidx\)](#) — Returns TRUE if two n-D float precision bounding boxes (GIDX) intersect each other.

**Synopsis**

boolean [&&&\( gidx A , gidx B \)](#);

**Descrição**

The [&&&](#) operator returns TRUE if two n-D bounding boxes A and B intersect each other, using float precision. This means that if A (or B) is a (double precision) box3d, it will be internally converted to a float precision 3D bounding box (GIDX)

**Note**

This operator is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.

## Exemplos

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_3DMakeBox(ST_MakePoint(1,1,1), ST_MakePoint(3,3,3)) AS overlaps;

overlaps
-----
t
(1 row)
```

## Veja também.

[&&&\(geometry,gidx\)](#), [&&&\(gidx,geometry\)](#)

## 8.8.9 &<

**&<** — Retorna VERDADE se a caixa limitadora de A sobrepõe ou está à esquerda de B.

### Synopsis

boolean **&<**( geometry A , geometry B );

### Descrição

O operador **&<** retorna VERDADE se a caixa limitadora da geometria A sobrepõe ou está à esquerda da caixa da geometria B, ou mais precisamente, sobrepõe ou NÃO está à direita da caixa limitadora da geometria B.



#### Note

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

## Exemplos

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &< tbl2.column2 AS overleft
FROM
  ( VALUES
    (1, 'LINESTRING(1 2, 4 6)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)::geometry),
```

```
(3, 'LINESTRING(0 1, 0 5)::geometry),
(4, 'LINESTRING(6 0, 6 1)::geometry) AS tbl2;
```

```
column1 | column1 | overleft
-----+-----+-----
      1 |      2 | f
      1 |      3 | f
      1 |      4 | t
(3 rows)
```

### Veja também.

[&&](#), [l&>](#), [&>](#), [&<](#)

### 8.8.10 &<

**&<** — Retorna VERDADE se a caixa limitadora de A sobrepõe ou está abaixo de B.

### Synopsis

boolean **&<**( geometry A , geometry B );

### Descrição

O operador **&<** retorna VERDADE se a caixa limitadora da geometria A sobrepõe ou está abaixo da caixa da geometria B, ou mais precisamente, sobrepõe ou NÃO está acima da caixa limitadora da geometria B.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



#### Note

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

### Exemplos

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &<| tbl2.column2 AS overbelow
FROM
  ( VALUES
    (1, 'LINESTRING(6 0, 6 4)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)::geometry),
    (3, 'LINESTRING(0 1, 0 5)::geometry),
    (4, 'LINESTRING(1 2, 4 6)::geometry) AS tbl2;

column1 | column1 | overbelow
-----+-----+-----
      1 |      2 | f
      1 |      3 | t
      1 |      4 | t
(3 rows)
```

**Veja também.**[&&](#), [|&>](#), [&>](#), [&<](#)**8.8.11 &>**

**&>** — Retorna VERDADE se a caixa limitadora de A sobrepõe ou está à direita de B.

**Synopsis**

boolean **&>**( geometry A , geometry B );

**Descrição**

O operador **&>** retorna VERDADE se a caixa limitadora da geometria A sobrepõe ou está à direita da caixa da geometria B, ou mais precisamente, sobrepõe ou NÃO está à esquerda da caixa limitadora da geometria B.

**Note**

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

**Exemplos**

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 &> tbl2.column2 AS overright
FROM
  ( VALUES
    (1, 'LINESTRING(1 2, 4 6)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)::geometry),
    (3, 'LINESTRING(0 1, 0 5)::geometry),
    (4, 'LINESTRING(6 0, 6 1)::geometry) AS tbl2;
```

column1	column1	overright
1	2	t
1	3	t
1	4	f

(3 rows)

**Veja também.**[&&](#), [|&>](#), [&<](#), [&<](#)**8.8.12 <<**

**<<** — Retorna VERDADE se uma caixa limitadora de A está estritamente à esquerda da de B.

**Synopsis**

boolean **<<**( geometry A , geometry B );

**Descrição**

O operador << retorna VERDADE se a caixa limitadora da geometria A está estritamente à esquerda da caixa limitadora da geometria B.

**Note**

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

**Exemplos**

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 << tbl2.column2 AS left
FROM
  ( VALUES
    (1, 'LINESTRING (1 2, 1 5)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0, 4 3)::geometry),
    (3, 'LINESTRING (6 0, 6 5)::geometry),
    (4, 'LINESTRING (2 2, 5 6)::geometry) AS tbl2;
```

```
column1 | column1 | left
-----+-----+-----
         1 |         2 | f
         1 |         3 | t
         1 |         4 | t
(3 rows)
```

**Veja também.**

>>, |>>, <<|

**8.8.13 <<|**

<<| — Retorna VERDADE se uma caixa limitadora de A está estritamente abaixo da de B.

**Synopsis**

```
boolean <<|( geometry A , geometry B );
```

**Descrição**

O operador <<| retorna VERDADE se a caixa limitadora da geometria A está estritamente à esquerda da caixa limitadora da geometria B.

**Note**

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

## Exemplos

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 <<| tbl2.column2 AS below
FROM
  ( VALUES
    (1, 'LINESTRING (0 0, 4 3)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (1 4, 1 7)::geometry),
    (3, 'LINESTRING (6 1, 6 5)::geometry),
    (4, 'LINESTRING (2 3, 5 6)::geometry) AS tbl2;

column1 | column1 | below
-----+-----+-----
        1 |         2 | t
        1 |         3 | f
        1 |         4 | f
(3 rows)
```

## Veja também.

<<, >>, |>>

### 8.8.14 =

= — Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B.

## Synopsis

```
boolean =( geometry A , geometry B );
boolean =( geography A , geography B );
```

## Descrição

The = operator returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B. PostgreSQL uses the =, <, and > operators defined for geometries to perform internal orderings and comparison of geometries (ie. in a GROUP BY or ORDER BY clause).



### Note

Only geometry/geography that are exactly equal in all respects, with the same coordinates, in the same order, are considered equal by this operator. For "spatial equality", that ignores things like coordinate order, and can detect features that cover the same spatial area with different representations, use [ST\\_OrderingEquals](#) or [ST\\_Equals](#)



### Caution

This operand will NOT make use of any indexes that may be available on the geometries. For an index assisted exact equality test, combine = with &&.

Changed: 2.4.0, in prior versions this was bounding box equality not a geometric equality. If you need bounding box equality, use ~= instead.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



## Exemplos

```

SELECT 'LINESTRING(0 0, 0 1, 1 0)::geometry = 'LINESTRING(1 1, 0 0)::geometry;
?column?
-----
f
(1 row)

SELECT ST_AsText(column1)
FROM ( VALUES
      ('LINESTRING(0 0, 1 1)::geometry),
      ('LINESTRING(1 1, 0 0)::geometry)) AS foo;
      st_astext
-----
LINESTRING(0 0,1 1)
LINESTRING(1 1,0 0)
(2 rows)

-- Note: the GROUP BY uses the "=" to compare for geometry equivalency.
SELECT ST_AsText(column1)
FROM ( VALUES
      ('LINESTRING(0 0, 1 1)::geometry),
      ('LINESTRING(1 1, 0 0)::geometry)) AS foo
GROUP BY column1;
      st_astext
-----
LINESTRING(0 0,1 1)
LINESTRING(1 1,0 0)
(2 rows)

-- In versions prior to 2.0, this used to return true --
SELECT ST_GeomFromText('POINT(1707296.37 4820536.77)') =
      ST_GeomFromText('POINT(1707296.27 4820536.87)') As pt_intersect;

--pt_intersect --
f

```

## Veja também.

[ST\\_Equals](#), [ST\\_OrderingEquals](#), [~=](#)

### 8.8.15 >>

>> — Returns TRUE if A's bounding box is strictly to the right of B's.

#### Synopsis

```
boolean >>( geometry A , geometry B );
```

#### Descrição

O operador >> retorna VERDADE se a caixa limitadora da geometria A está estritamente à direita da caixa limitadora da geometria B.

**Note**

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

**Exemplos**

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 >> tbl2.column2 AS right
FROM
  ( VALUES
    (1, 'LINESTRING (2 3, 5 6)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (1 4, 1 7)::geometry),
    (3, 'LINESTRING (6 1, 6 5)::geometry),
    (4, 'LINESTRING (0 0, 4 3)::geometry) AS tbl2;
```

column1	column1	right
1	2	t
1	3	f
1	4	f

(3 rows)

**Veja também.**

<<, >>, <<

**8.8.16 @**

@ — Retorna VERDADE se uma caixa limitadora de A está contida pela de B.

**Synopsis**

boolean @( geometry A , geometry B );

**Descrição**

O operador @ retorna VERDADE se a caixa limitadora da geometria A estiver completamente contida pela caixa limitadora da geometria B.

**Note**

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

**Exemplos**

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 @ tbl2.column2 AS contained
FROM
  ( VALUES
    (1, 'LINESTRING (1 1, 3 3)::geometry) AS tbl1,
  ( VALUES
```

```
(2, 'LINESTRING (0 0, 4 4)::geometry),
(3, 'LINESTRING (2 2, 4 4)::geometry),
(4, 'LINESTRING (1 1, 3 3)::geometry)) AS tbl2;
```

column1	column1	contained
1	2	t
1	3	f
1	4	t

(3 rows)

**Veja também.**

~, &&

### 8.8.17 @(geometry,box2df)

@(geometry,box2df) — Returns TRUE if a geometry's 2D bounding box is contained into a 2D float precision bounding box (BOX2DF).

#### Synopsis

```
boolean @( geometry A , box2df B );
```

#### Descrição

The @ operator returns TRUE if the A geometry's 2D bounding box is contained the 2D bounding box B, using float precision. This means that if B is a (double precision) box2d, it will be internally converted to a float precision 2D bounding box (BOX2DF)



#### Note

This operand is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

#### Exemplos

```
SELECT ST_Buffer(ST_GeomFromText('POINT(2 2)'), 1) @ ST_MakeBox2D(ST_MakePoint(0,0), ↔
ST_MakePoint(5,5)) AS is_contained;
```

is_contained
t

(1 row)

**Veja também.**

[&&\(geometry,box2df\)](#), [&&\(box2df,geometry\)](#), [&&\(box2df,box2df\)](#), [~\(geometry,box2df\)](#), [~\(box2df,geometry\)](#), [~\(box2df,box2df\)](#), [@\(box2df,geometry\)](#), [@\(box2df,box2df\)](#)

**8.8.18 @(box2df,geometry)**

`@(box2df,geometry)` — Returns `TRUE` if a 2D float precision bounding box (`BOX2DF`) is contained into a geometry's 2D bounding box.

**Synopsis**

```
boolean @( box2df A , geometry B );
```

**Descrição**

The `@` operator returns `TRUE` if the 2D bounding box `A` is contained into the `B` geometry's 2D bounding box, using float precision. This means that if `B` is a (double precision) `box2d`, it will be internally converted to a float precision 2D bounding box (`BOX2DF`).

**Note**

This operand is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

**Exemplos**

```
SELECT ST_MakeBox2D(ST_MakePoint(2,2), ST_MakePoint(3,3)) @ ST_Buffer(ST_GeomFromText(' ↵
POINT(1 1)'), 10) AS is_contained;
```

```
is_contained
-----
t
(1 row)
```

**Veja também.**

[&&\(geometry,box2df\)](#), [&&\(box2df,geometry\)](#), [&&\(box2df,box2df\)](#), [~\(geometry,box2df\)](#), [~\(box2df,geometry\)](#), [~\(box2df,box2df\)](#), [@\(geometry,box2df\)](#), [@\(box2df,box2df\)](#)

**8.8.19 @(box2df,box2df)**

`@(box2df,box2df)` — Returns `TRUE` if a 2D float precision bounding box (`BOX2DF`) is contained into another 2D float precision bounding box.

## Synopsis

```
boolean @( box2df A , box2df B );
```

## Descrição

The @ operator returns TRUE if the 2D bounding box A is contained into the 2D bounding box B, using float precision. This means that if A (or B) is a (double precision) box2d, it will be internally converted to a float precision 2D bounding box (BOX2DF)



### Note

This operand is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

```
SELECT ST_MakeBox2D(ST_MakePoint(2,2), ST_MakePoint(3,3)) @ ST_MakeBox2D(ST_MakePoint(0,0), ←
      ST_MakePoint(5,5)) AS is_contained;
```

```
is_contained
-----
t
(1 row)
```

## Veja também.

[&&\(geometry,box2df\)](#), [&&\(box2df,geometry\)](#), [&&\(box2df,box2df\)](#), [~\(geometry,box2df\)](#), [~\(box2df,geometry\)](#), [~\(box2df,box2df\)](#), [@\(geometry,box2df\)](#), [@\(box2df,geometry\)](#)

### 8.8.20 |&>

|&> — Retorna VERDADE se a caixa limitadora de A sobrepõe ou está acima de B.

## Synopsis

```
boolean |&>( geometry A , geometry B );
```

## Descrição

O operador |&> retorna VERDADE se a caixa limitadora da geometria A sobrepõe ou está acima da caixa da geometria B, ou mais precisamente, sobrepõe ou NÃO está abaixo da caixa limitadora da geometria B.



### Note

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

## Exemplos

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 |&> tbl2.column2 AS overabove
FROM
  ( VALUES
    (1, 'LINESTRING(6 0, 6 4)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING(0 0, 3 3)::geometry),
    (3, 'LINESTRING(0 1, 0 5)::geometry),
    (4, 'LINESTRING(1 2, 4 6)::geometry) AS tbl2;
```

column1	column1	overabove
1	2	t
1	3	f
1	4	f

(3 rows)

## Veja também.

[&&](#), [&>](#), [&<](#), [&<](#)

### 8.8.21 |>>

|>> — Retorna VERDADE se uma caixa limitadora de A está estritamente acima da de B.

## Synopsis

boolean |>>( geometry A , geometry B );

## Descrição

The |>> operator returns TRUE if the bounding box of geometry A is strictly above the bounding box of geometry B.



### Note

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

## Exemplos

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 |>> tbl2.column2 AS above
FROM
  ( VALUES
    (1, 'LINESTRING (1 4, 1 7)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0, 4 2)::geometry),
    (3, 'LINESTRING (6 1, 6 5)::geometry),
    (4, 'LINESTRING (2 3, 5 6)::geometry) AS tbl2;
```

column1	column1	above
1	2	t
1	3	f
1	4	f

(3 rows)

**Veja também.**[<<, >>, <<|](#)**8.8.22 ~**

~ — Retorna VERDADE se uma caixa limitadora de A contém a de B.

**Synopsis**

```
boolean ~( geometry A , geometry B );
```

**Descrição**

O operador ~ retorna VERDADE se a caixa limitadora da geometria A estiver completamente contida pela caixa limitadora da geometria B.

**Note**

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

**Exemplos**

```
SELECT tbl1.column1, tbl2.column1, tbl1.column2 ~ tbl2.column2 AS contains
FROM
  ( VALUES
    (1, 'LINESTRING (0 0, 3 3)::geometry) AS tbl1,
  ( VALUES
    (2, 'LINESTRING (0 0, 4 4)::geometry),
    (3, 'LINESTRING (1 1, 2 2)::geometry),
    (4, 'LINESTRING (0 0, 3 3)::geometry) AS tbl2;
```

```
column1 | column1 | contains
-----+-----+-----
         1 |         2 | f
         1 |         3 | t
         1 |         4 | t
(3 rows)
```

**Veja também.**[@, &&](#)**8.8.23 ~(geometry,box2df)**

~(geometry,box2df) — Returns TRUE if a geometry's 2D bonding box contains a 2D float precision bounding box (GIDX).

**Synopsis**

```
boolean ~( geometry A , box2df B );
```

## Descrição

The `~` operator returns `TRUE` if the 2D bounding box of a geometry A contains the 2D bounding box B, using float precision. This means that if B is a (double precision) `box2d`, it will be internally converted to a float precision 2D bounding box (`BOX2DF`)



### Note

This operand is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

```
SELECT ST_Buffer(ST_GeomFromText('POINT(1 1)'), 10) ~ ST_MakeBox2D(ST_MakePoint(0,0), ↔
    ST_MakePoint(2,2)) AS contains;
```

```
contains
-----
t
(1 row)
```

## Veja também.

[&&\(geometry,box2df\)](#), [&&\(box2df,geometry\)](#), [&&\(box2df,box2df\)](#), [~\(box2df,geometry\)](#), [~\(box2df,box2df\)](#), [@\(geometry,box2df\)](#), [@\(box2df,geometry\)](#), [@\(box2df,box2df\)](#)

### 8.8.24 ~(box2df,geometry)

`~(box2df,geometry)` — Returns `TRUE` if a 2D float precision bounding box (`BOX2DF`) contains a geometry's 2D bonding box.

## Synopsis

```
boolean ~( box2df A , geometry B );
```

## Descrição

The `~` operator returns `TRUE` if the 2D bounding box A contains the B geometry's bounding box, using float precision. This means that if A is a (double precision) `box2d`, it will be internally converted to a float precision 2D bounding box (`BOX2DF`)



### Note

This operand is intended to be used internally by BRIN indexes, more than by users.



Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

### Exemplos

```
SELECT ST_MakeBox2D(ST_MakePoint(0,0), ST_MakePoint(5,5)) ~ ST_Buffer(ST_GeomFromText(' ↵
POINT(2 2)'), 1) AS contains;
```

```
contains
-----
t
(1 row)
```

### Veja também.

[&&\(geometry,box2df\)](#), [&&\(box2df,geometry\)](#), [&&\(box2df,box2df\)](#), [~\(geometry,box2df\)](#), [~\(box2df,box2df\)](#), [@\(geometry,box2df\)](#), [@\(box2df,geometry\)](#), [@\(box2df,box2df\)](#)

### 8.8.25 ~(box2df,box2df)

~(box2df,box2df) — Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).

#### Synopsis

```
boolean ~( box2df A , box2df B );
```

#### Descrição

The ~ operator returns TRUE if the 2D bounding box A contains the 2D bounding box B, using float precision. This means that if A is a (double precision) box2d, it will be internally converted to a float precision 2D bounding box (BOX2DF)



#### Note

This operand is intended to be used internally by BRIN indexes, more than by users.

Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

## Exemplos

```
SELECT ST_MakeBox2D(ST_MakePoint(0,0), ST_MakePoint(5,5)) ~ ST_MakeBox2D(ST_MakePoint(2,2),
    ST_MakePoint(3,3)) AS contains;
```

```
contains
-----
t
(1 row)
```

## Veja também.

[&&\(geometry,box2df\)](#), [&&\(box2df,geometry\)](#), [&&\(box2df,box2df\)](#), [~\(geometry,box2df\)](#), [~\(box2df,geometry\)](#), [@\(geometry,box2df\)](#), [@\(box2df,geometry\)](#), [@\(box2df,box2df\)](#)

### 8.8.26 ~=

`~=` — Retorna VERDADE se a caixa limitadora de A é a mesma de B.

## Synopsis

boolean `~=`( geometry A , geometry B );

## Descrição

O operador `~` retorna VERDADE se a caixa limitadora da geometria/geografia A for a mesma da caixa limitadora da geometria/geografia B.



### Note

Esse operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

Disponibilidade: 1.5.0 comportamento alterado



This function supports Polyhedral surfaces.



### Warning

This operator has changed behavior in PostGIS 1.5 from testing for actual geometric equality to only checking for bounding box equality. To complicate things it also depends on if you have done a hard or soft upgrade which behavior your database has. To find out which behavior your database has you can run the query below. To check for true equality use [ST\\_OrderingEquals](#) or [ST\\_Equals](#).

## Exemplos

```
select 'LINESTRING(0 0, 1 1)::geometry ~= 'LINESTRING(0 1, 1 0)::geometry as equality;
equality |
-----+
t       |
```

Veja também.

[ST\\_Equals](#), [ST\\_OrderingEquals](#), [=](#)

### 8.8.27 <->

<-> — Retorna a distância 2D entre A e B.

#### Synopsis

```
double precision <->( geometry A , geometry B );
double precision <->( geography A , geography B );
```

#### Descrição

O operador <-> retorna a distância 2D entre duas geometrias. Usado nas orações "ORDEM" que fornecem configurações de resultado index-assisted nearest-neighbor. Para o PostgreSQL menor que 9.5 somente fornece a distância centroide das caixas limitadoras e para PostgreSQL 9.5+, a verdadeira distância KNN procura dando verdadeiras distâncias entre geometrias, e distância esférica para geografias.



#### Note

Esse operador fará uso dos indexes 2D GiST que podem estar disponíveis nas geometrias. É diferente de outros operadores que usam indexes espaciais em que eles só são usados quando o operador está na oração ORDEM.



#### Note

O index só rejeita se uma das geometrias é uma constante (não em uma subquery/cte). ex. 'SRID=3005;POINT(1011102 450541)::geometria ao invés de uma .geom

Vá para [OpenGeo workshop: Nearest-Neighbour Searching](#) para um exemplo real.

melhorias: 2.2.0 -- Verdadeiro comportamento KNN ("vizinho mais perto de K") para geometria e geografia para PostgreSQL 9.5+. Note que para geografia o KNN é baseado em esfera ao invés de esferoide. Para o PostgreSQL 9.4 ou menor, o suporte para geografia é novo, mas só suporta caixa centroide.

Alterações: 2.2.0 -- Para usuários do PostgreSQL 9.5, a sintaxe Hybrid antiga pode ser mais lenta, então, você vai querer se livrar daquele hack se você está executando seu código só no PostGIS 2.2+ 9.5+. Veja os exemplos abaixo.

Disponibilidade: 2.0.0 -- O KNN mais fraco fornece vizinho mais próximos baseados em distâncias centroides de geometrias, ao invés de distâncias reais. Resultados corretos para pontos, incorretos para todos os outros tipos. Disponível para PostgreSQL 9.1+

#### Exemplos

```
SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)::geometry) as d,edabbr, vaabbr
FROM va2005
ORDER BY d limit 10;
```

d	edabbr	vaabbr
0	ALQ	128
5541.57712511724	ALQ	129A
5579.67450712005	ALQ	001

```

6083.4207708641 | ALQ      | 131
7691.2205404848 | ALQ      | 003
7900.75451037313 | ALQ      | 122
8694.20710669982 | ALQ      | 129B
9564.24289057111 | ALQ      | 130
12089.665931705  | ALQ      | 127
18472.5531479404 | ALQ      | 002
(10 rows)

```

Então, a resposta KNN crua:

```

SELECT st_distance(geom, 'SRID=3005;POINT(1011102 450541)::geometry) as d,edabbr, vaabbr
FROM va2005
ORDER BY geom <-> 'SRID=3005;POINT(1011102 450541)::geometry' limit 10;

```

```

      d      | edabbr | vaabbr
-----+-----+-----
          0 | ALQ    | 128
5541.57712511724 | ALQ    | 129A
5579.67450712005 | ALQ    | 001
6083.4207708641 | ALQ    | 131
7691.2205404848 | ALQ    | 003
7900.75451037313 | ALQ    | 122
8694.20710669982 | ALQ    | 129B
9564.24289057111 | ALQ    | 130
12089.665931705  | ALQ    | 127
18472.5531479404 | ALQ    | 002
(10 rows)

```

Se você executar "ANÁLISE EXPLICATIVA" nas duas pesquisas, você verá uma apresentação melhorada para a segunda.

Para usuários com PostgreSQL < 9.5, use uma pesquisa hybrid para encontrar os vizinhos verdadeiros mais próximos. Primeiro, uma pesquisa CTE usando o index-assisted KNN, e depois, uma pesquisa exata para pegar a ordem certa:

```

WITH index_query AS (
  SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)::geometry) as d,edabbr, vaabbr
  FROM va2005
  ORDER BY geom <-> 'SRID=3005;POINT(1011102 450541)::geometry' LIMIT 100)
SELECT *
  FROM index_query
 ORDER BY d limit 10;

```

```

      d      | edabbr | vaabbr
-----+-----+-----
          0 | ALQ    | 128
5541.57712511724 | ALQ    | 129A
5579.67450712005 | ALQ    | 001
6083.4207708641 | ALQ    | 131
7691.2205404848 | ALQ    | 003
7900.75451037313 | ALQ    | 122
8694.20710669982 | ALQ    | 129B
9564.24289057111 | ALQ    | 130
12089.665931705  | ALQ    | 127
18472.5531479404 | ALQ    | 002
(10 rows)

```

**Veja também.**

[ST\\_DWithin](#), [ST\\_Distance](#), [<#>](#)

### 8.8.28 `|=|`

`|=|` — Retorna a distância entre As trajetórias A e B ao ponto de aproximação mais perto.

#### Synopsis

```
double precision |=|( geometry A , geometry B );
```

#### Descrição

O operador `|=|` retorna a distância 3D entre duas trajetórias (Veja [ST\\_IsValidTrajectory](#)). Isso é o mesmo que [ST\\_DistanceCPA](#), mas como um operador pode ser usado para fazer pesquisas de vizinhos próximos usando um index n-dimensional (requer PostgreSQL 9.5.0 ou superior).



#### Note

Esse operador fará uso dos indexes ND GiST que podem estar disponíveis nas geometrias. É diferente de outros operadores que usam indexes espaciais em que eles só são usados quando o operador está na oração ORDER.



#### Note

O index só rejeita se uma das geometrias é uma constante (não em uma subquery/cte). ex: `'SRID=3005;LINESTRINGM(0 0 0,0 0 1)'`::geometria ao invés de uma `.geom`

Disponibilidade: 2.2.0. Index suportado disponível somente para PostgreSQL 9.5+

#### Exemplos

```
-- Save a literal query trajectory in a psql variable...
\set qt 'ST_AddMeasure(ST_MakeLine(ST_MakePointM(-350,300,0),ST_MakePointM(-410,490,0)) ←
,10,20)'
-- Run the query !
SELECT track_id, dist FROM (
  SELECT track_id, ST_DistanceCPA(tr,:qt) dist
  FROM trajectories
  ORDER BY tr |=| :qt
  LIMIT 5
) foo;
 track_id      dist
-----+-----
    395 | 0.576496831518066
    380 |  5.06797130410151
    390 |  7.72262293958322
    385 |  9.8004461358071
    405 | 10.9534397988433
(5 rows)
```

#### Veja também.

[ST\\_DistanceCPA](#), [ST\\_ClosestPointOfApproach](#), [ST\\_IsValidTrajectory](#)

### 8.8.29 <#>

<#> — Retorna a distância 2D entre as caixas limitadoras de A e B.

#### Synopsis

```
double precision <#>( geometry A , geometry B );
```

#### Descrição

O operador <#> retorna a distância entre dois pontos flutuantes, possivelmente lendo eles de um index espacial (PostgreSQL 9.1+ requerido). Útil para tornar vizinhos mais próximos **aproximar** a distância pedida.



#### Note

Esse operador fará uso dos indexes que podem estar disponíveis nas geometrias. É diferente de outros operadores que usam indexes espaciais em que eles só são usados quando o operador está na oração ORDER.



#### Note

O index só rejeita se uma das geometrias é uma constante ex. ORDER BY (ST\_GeomFromText('POINT(1 2)') <#> geom) ao invés de uma g1.geom <#>.

Disponibilidade: 2.0.0 -- KNN só está disponível para PostgreSQL 9.1+

#### Exemplos

```
SELECT *
FROM (
SELECT b.tlid, b.mtfcc,
       b.geom <#
> ST_GeomFromText('LINESTRING(746149 2948672,745954 2948576,
745787 2948499,745740 2948468,745712 2948438,
745690 2948384,745677 2948319)',2249) As b_dist,
       ST_Distance(b.geom, ST_GeomFromText('LINESTRING(746149 2948672,745954 ↵
2948576,
745787 2948499,745740 2948468,745712 2948438,
745690 2948384,745677 2948319)',2249)) As act_dist
FROM bos_roads As b
ORDER BY b_dist, b.tlid
LIMIT 100) As foo
ORDER BY act_dist, tlid LIMIT 10;
```

tlid	mtfcc	b_dist	act_dist
85732027	S1400	0	0
85732029	S1400	0	0
85732031	S1400	0	0
85734335	S1400	0	0
85736037	S1400	0	0
624683742	S1400	0	128.528874268666
85719343	S1400	260.839270432962	260.839270432962
85741826	S1400	164.759294123275	260.839270432962
85732032	S1400	277.75	311.830282365264
85735592	S1400	222.25	311.830282365264

(10 rows)

Veja também.

[ST\\_DWithin](#), [ST\\_Distance](#), [<->](#)

### 8.8.30 <<->>

<<->> — Retorna a distância n-D entre as centroides das caixas limitadoras de A e B.

#### Synopsis

```
double precision <<->>( geometry A , geometry B );
```

#### Descrição

O operador <<->> retorna a distância (euclidiana) n-D entre as centroides das caixas limitadoras de duas geometrias. Útil para tornar vizinhos mais próximos **aproximar** a distância perdida.



#### Note

Esse operador fará uso dos indexes n-D GiST que podem estar disponíveis nas geometrias. É diferente de outros operadores que usam indexes espaciais em que eles só são usados quando o operador está na oração ORDER.



#### Note

O index só rejeita se uma das geometrias é uma constante (não em uma subquery/cte). ex. 'SRID=3005;POINT(1011102 450541)::geometria ao invés de uma .geom

Disponibilidade: 2.2.0 -- KNN só está disponível para PostgreSQL 9.1+

Veja também.

[<<#>>](#), [<->](#)

### 8.8.31 <<#>>

<<#>> — Retorna a distância n-D entre as caixas limitadoras de A e B.

#### Synopsis

```
double precision <<#>>( geometry A , geometry B );
```

#### Descrição

O operador <<#>> retorna a distância entre dois pontos flutuantes, possivelmente lendo eles de um index espacial (PostgreSQL 9.1+ requerido). Útil para tornar vizinhos mais próximos **aproximar** uma distância pedida.



#### Note

Esse operador fará uso dos indexes que podem estar disponíveis nas geometrias. É diferente de outros operadores que usam indexes espaciais em que eles só são usados quando o operador está na oração ORDER.

**Note**

O index só rejeita se uma das geometrias é ua constante ex. ORDER BY (ST\_GeomFromText('POINT(1 2)') <<#>> geom) ao invés de g1.geom <<#>>.

Disponibilidade: 2.2.0 -- KNN só está disponível para PostgreSQL 9.1+

**Veja também.**

<<->, <#>

## 8.9 Relações espaciais e medidas

### 8.9.1 ST\_3DClosestPoint

ST\_3DClosestPoint — Retorna o ponto 3 dimensional em g1 que é o mais próximo de g2. Este é o primeiro ponto da linha mais curta em três dimensões.

**Synopsis**

geometry **ST\_3DClosestPoint**(geometry g1, geometry g2);

**Descrição**

Retorne o ponto 3-dimensional no g1 que é mais perto ao g2. Esse é o primeiro ponto da menor linha 3D. O comprimento 3D da menor linha 3D é a distância 3D.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

Disponibilidade: 2.0.0

Alterações: 2.2.0 - se 2 geometrias 2D são entradas, um ponto 2D retorna (em vez do antigo comportamento assumindo 0 para Z perdido). Em caso de 2D e 3D, o Z não é mais 0 para Z perdido.

**Exemplos**

linestring e ponto -- pontos 3d e 2d mais próximos

```
SELECT ST_AsEWKT(ST_3DClosestPoint(line,pt)) AS cp3d_line_pt,
       ST_AsEWKT(ST_ClosestPoint(line,pt)) As cp2d_line_pt
FROM (SELECT 'POINT(100 100 30)::geometry As pt,
            'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 1000)::' ←
       geometry As line
       ) As foo;

cp3d_line_pt |                               ←
cp2d_line_pt
-----+-----
POINT(54.6993798867619 128.935022917228 11.5475869506606) | POINT(73.0769230769231 ←
115.384615384615)
```



**linestring e multiponto -- pontos 3d e 2d mais próximos**

```
SELECT ST_AsEWKT(ST_3DClosestPoint(line,pt)) AS cp3d_line_pt,
       ST_AsEWKT(ST_ClosestPoint(line,pt)) As cp2d_line_pt
FROM (SELECT 'MULTIPOINT(100 100 30, 50 74 1000) '::geometry As pt,
            'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 900) ':: geometry As line
      ) As foo;
```

cp3d_line_pt	cp2d_line_pt
POINT(54.69937988867619 128.935022917228 11.5475869506606)	POINT(50 75)

**Multilinestring e polígono pontos 3d e 2d mais próximos**

```
SELECT ST_AsEWKT(ST_3DClosestPoint(poly, mline)) As cp3d,
       ST_AsEWKT(ST_ClosestPoint(poly, mline)) As cp2d
FROM (SELECT ST_GeomFromEWKT('POLYGON((175 150 5, 20 40 5, 35 45 5, 50 60 5,
100 100 5, 175 150 5))') As poly,
       ST_GeomFromEWKT('MULTILINESTRING((175 155 2, 20 40 20, 50 60 -2, 125
100 1, 175 155 1),
       (1 10 2, 5 20 1))') As mline ) As foo;
```

cp3d	cp2d
POINT(39.993580415989 54.1889925532825 5)	POINT(20 40)

**Veja também**

[ST\\_AsEWKT](#), [ST\\_ClosestPoint](#), [ST\\_3DDistance](#), [ST\\_3DShortestLine](#)

**8.9.2 ST\_3DDistance**

**ST\_3DDistance** — Para tipo geometria, retorna a menor distância cartesiana 3-dimensional (baseado no sistema de referência espacial) entre duas geometrias em unidades projetadas.

**Synopsis**

```
float ST_3DDistance(geometry g1, geometry g2);
```

**Descrição**

Para tipo geometria, retorna a menor distância cartesiana 3-dimensional entre duas geometrias em unidades projetadas (spatial ref units).



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This method implements the SQL/MM specification. SQL-MM ?



This method is also provided by SFCGAL backend.

Disponibilidade: 2.0.0

Alterações: 2.2.0 - Em caso de 2D e 3D, o Z não é mais 0 para Z perdido.

## Exemplos

```
-- Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point ←
  and line compared 2D point and line)
-- Note: currently no vertical datum support so Z is not transformed and assumed to be same ←
  units as final.
SELECT ST_3DDistance(
    ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 4)') ←
      ,2163),
    ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 ←
      15, -72.123 42.1546 20)'),2163)
  ) As dist_3d,
  ST_Distance(
    ST_Transform(ST_GeomFromText('POINT(-72.1235 42.3521)',4326),2163),
    ST_Transform(ST_GeomFromText('LINESTRING(-72.1260 42.45, -72.123 ←
      42.1546)', 4326),2163)
  ) As dist_2d;

  dist_3d      |      dist_2d
-----+-----
127.295059324629 | 126.66425605671
```

```
-- Multilinestring and polygon both 3d and 2d distance
-- Same example as 3D closest point example
SELECT ST_3DDistance(poly, mline) As dist3d,
  ST_Distance(poly, mline) As dist2d
  FROM (SELECT 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;

  dist3d      |      dist2d
-----+-----
0.716635696066337 | 0
```

## Veja também

[ST\\_Distance](#), [ST\\_3DClosestPoint](#), [ST\\_3DDWithin](#), [ST\\_3DMaxDistance](#), [ST\\_3DShortestLine](#), [ST\\_Transform](#)

### 8.9.3 ST\_3DDWithin

**ST\_3DDWithin** — Para tipo de geometria 3d (z) verdadeiro se a distância entre duas geometrias 3d está dentro de números de unidades.

#### Synopsis

boolean **ST\_3DDWithin**(geometry g1, geometry g2, double precision distance\_of\_srid);

#### Descrição

Para tipo de geometria retorna verdade se a distância 3d entre dois objetos estiver dentro da unidades projetadas distance\_of\_srid especificadas (unidades de referência espacial).



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This method implements the SQL/MM specification. SQL-MM ?

Disponibilidade: 2.0.0

### Exemplos

```
-- Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point ←
  and line compared 2D point and line)
-- Note: currently no vertical datum support so Z is not transformed and assumed to be same ←
  units as final.
SELECT ST_3DDWithin(
           ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 4)') ←
             ,2163),
           ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 ←
             15, -72.123 42.1546 20)'),2163),
           126.8
         ) As within_dist_3d,
ST_DWithin(
           ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 4)') ←
             ,2163),
           ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 ←
             15, -72.123 42.1546 20)'),2163),
           126.8
         ) As within_dist_2d;

within_dist_3d | within_dist_2d
-----+-----
f              | t
```

### Veja também

[ST\\_3DDistance](#), [ST\\_Distance](#), [ST\\_DWithin](#), [ST\\_3DMaxDistance](#), [ST\\_Transform](#)

## 8.9.4 ST\_3DDFullyWithin

**ST\_3DDFullyWithin** — Retorna verdade se todas as geometrias 3D estiverem dentro da distância especificada de um outro.

### Synopsis

boolean **ST\_3DDFullyWithin**(geometry g1, geometry g2, double precision distance);

### Descrição

Retorna verdade se todas as geometrias 3D estiverem completamente dentro da distância especificada de um outro. A distância é especificada em unidade definidas pelo sistema de referência espacial de geometrias. Para esta função fazer sentido, as geometrias fonte devem ser da mesma projeção coordenada, tendo o mesmo SRID.



#### Note

Esta função pode, automaticamente, incluir uma comparação de caixa delimitadora que usará quaisquer índices que estejam disponíveis nas geometrias.

Disponibilidade: 2.0.0



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

### Exemplos

```
-- This compares the difference between fully within and distance within as well
-- as the distance fully within for the 2D footprint of the line/point vs.
-- the 3d fully within
SELECT ST_3DDFullyWithin(geom_a, geom_b, 10) as D3DFullyWithin10,
       ST_3DDWithin(geom_a, geom_b, 10) as D3DWithin10,
       ST_DFullyWithin(geom_a, geom_b, 20) as D2DFullyWithin20,
       ST_3DDFullyWithin(geom_a, geom_b, 20) as D3DFullyWithin20 from
(select ST_GeomFromEWKT('POINT(1 1 2)') as geom_a,
       ST_GeomFromEWKT('LINESTRING(1 5 2, 2 7 20, 1 9 100, 14 12 3)') as geom_b)
t1;
-----+-----+-----+-----+
d3dfullywithin10 | d3dwithin10 | d2dfullywithin20 | d3dfullywithin20
-----+-----+-----+-----+
f                 | t           | t                 | f
```

### Veja também

[ST\\_3DMaxDistance](#), [ST\\_3DDWithin](#), [ST\\_DWithin](#), [ST\\_DFullyWithin](#)

## 8.9.5 ST\_3DIntersects

**ST\_3DIntersects** — Retorna VERDADE se as geometrias "intersectadas espacialmente" em 3d - somente para pontos, linestrings, polígonos, superfícies poliédricas (área). Com o backend SFCGAL ativado, também suporta TINS

### Synopsis

boolean **ST\_3DIntersects**( geometry geomA , geometry geomB );

### Descrição

Sobreposição, toca, dentro de todas as interseções espaciais implicadas. Se qualquer dos anteriores retornarem verdade, então as geometrias também intersectam espacialmente. Desconjunção implica falso para interseção espacial.

Disponibilidade: 2.0.0



#### Note

Esta função pode, automaticamente, incluir uma comparação de caixa delimitadora que usará quaisquer índices que estejam disponíveis nas geometrias.



#### Note

Com o objetivo de obter vantagem para TINS, você precisa ativar o backend SFCGAL. Isto pode ser feito na seção `set postgis.backend = sfcgal;` ou no banco de dados ou sistema de níveis. O nível do banco de dados pode ser feito com: `set postgis.backend = sfcgal;`



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This method is also provided by SFCGAL backend.



This method implements the SQL/MM specification. SQL-MM 3: ?

### Exemplos de Geometria

```
SELECT ST_3DIntersects(pt, line), ST_Intersects(pt,line)
       FROM (SELECT 'POINT(0 0 2)::geometry As pt,
                   'LINESTRING (0 0 1, 0 2 3 )::geometry As line) As foo;
st_3dintersects | st_intersects
-----+-----
f                | t
(1 row)
```

### Exemplos TIN

```
set postgis.backend = sfcgal;
SELECT ST_3DIntersects('TIN(((0 0,1 0,0 1,0 0)))::geometry, 'POINT(.1 .1)::geometry);
st_3dintersects
-----
t
```

### Veja também

[ST\\_Intersects](#)

## 8.9.6 ST\_3DLongestLine

**ST\_3DLongestLine** — Retorna a linha 3-dimensional mais longa entre duas geometrias

### Synopsis

```
geometry ST_3DLongestLine(geometry g1, geometry g2);
```

### Descrição

Retorna a linha 3-dimensional mais longa entre duas geometrias. A função só retornará a primeira linha, se existirem mais de uma. A linha retornada sempre começará em g1 e acabará em g2. O comprimento da linha essa função sempre será o mesmo do [ST\\_3DMaxDistance](#) retorna para g1 e g2.

Disponibilidade: 2.0.0

Alterações: 2.2.0 - se 2 geometrias 2D são entradas, um ponto 2D retorna (em vez do antigo comportamento assumindo 0 para Z perdido). Em caso de 2D e 3D, o Z não é mais 0 para Z perdido.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

## Exemplos

### linestring e ponto -- linhas 3d e 2d mais longas

```
SELECT ST_AsEWKT(ST_3DLongestLine(line,pt)) AS lol3d_line_pt,
       ST_AsEWKT(ST_LongestLine(line,pt)) As lol2d_line_pt
FROM (SELECT 'POINT(100 100 30)::geometry As pt,
            'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 1000)::' ←
       geometry As line
       ) As foo;
```

lol3d_line_pt		lol2d_line_pt
-----+-----		
LINESTRING(50 75 1000,100 100 30)		LINESTRING(98 190,100 100)

### linestring e multiponto -- linhas 3d e 2d mais longas

```
SELECT ST_AsEWKT(ST_3DLongestLine(line,pt)) AS lol3d_line_pt,
       ST_AsEWKT(ST_LongestLine(line,pt)) As lol2d_line_pt
FROM (SELECT 'MULTIPOINT(100 100 30, 50 74 1000)::geometry As pt,
            'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 900)::' ←
       geometry As line
       ) As foo;
```

lol3d_line_pt		lol2d_line_pt
-----+-----		
LINESTRING(98 190 1,50 74 1000)		LINESTRING(98 190,50 74)

### Multilinestring e polígono linhas 3d e 2d mais longas

```
SELECT ST_AsEWKT(ST_3DLongestLine(poly, mline)) As lol3d,
       ST_AsEWKT(ST_LongestLine(poly, mline)) As lol2d
FROM (SELECT ST_GeomFromEWKT('POLYGON((175 150 5, 20 40 5, 35 45 5, 50 60 5, ←
100 100 5, 175 150 5))') As poly,
       ST_GeomFromEWKT('MULTILINESTRING((175 155 2, 20 40 20, 50 60 -2, 125 ←
100 1, 175 155 1),
       (1 10 2, 5 20 1))') As mline ) As foo;
```

lol3d		lol2d
-----+-----		
LINESTRING(175 150 5,1 10 2)		LINESTRING(175 150,1 10)

## Veja também

[ST\\_3DClosestPoint](#), [ST\\_3DDistance](#), [ST\\_LongestLine](#), [ST\\_3DShortestLine](#), [ST\\_3DMaxDistance](#)

## 8.9.7 ST\_3DMaxDistance

**ST\_3DMaxDistance** — Para tipo de geometria retorna a maior distância 3-dimensional cartesiana (baseada na referência espacial) entre duas geometrias em unidade projetadas.

**Synopsis**

```
float ST_3DMaxDistance(geometry g1, geometry g2);
```

**Descrição**

Para tipo geometria, retorna a menor distância cartesiana 3-dimensional entre duas geometrias em unidades projetadas (spatial ref units).



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

Disponibilidade: 2.0.0

Alterações: 2.2.0 - Em caso de 2D e 3D, o Z não é mais 0 para Z perdido.

**Exemplos**

```
-- Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (3D point ↔
  and line compared 2D point and line)
-- Note: currently no vertical datum support so Z is not transformed and assumed to be same ↔
  units as final.
SELECT ST_3DMaxDistance(
    ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 ↔
      10000)'),2163),
    ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 ↔
      15, -72.123 42.1546 20)'),2163)
  ) As dist_3d,
  ST_MaxDistance(
    ST_Transform(ST_GeomFromEWKT('SRID=4326;POINT(-72.1235 42.3521 ↔
      10000)'),2163),
    ST_Transform(ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45 ↔
      15, -72.123 42.1546 20)'),2163)
  ) As dist_2d;

  dist_3d      |      dist_2d
-----+-----
24383.7467488441 | 22247.8472107251
```

**Veja também**

[ST\\_Distance](#), [ST\\_3DDWithin](#), [ST\\_3DMaxDistance](#), [ST\\_Transform](#)

**8.9.8 ST\_3DShortestLine**

ST\_3DShortestLine — Retorna a menor linha 3-dimensional entre duas geometrias

**Synopsis**

```
geometry ST_3DShortestLine(geometry g1, geometry g2);
```

## Descrição

Retorna a menor linha 3-dimensional entre duas geometrias, a função só irá retornar a primeira linha menor se houverem mais de um, este a função encontra. Se g1 e g2 intersecta em apenas um ponto, a função retornará uma linha com os pontos de interseção da direita e esquerda. Se g1 e g2 estão intersectando em mais de um ponto, a função retornará uma linha com começo e fim no mesmo ponto, mas também pode ser qualquer um dos outros pontos. A linha que retorna sempre começará com g2 e acabará em g2. O comprimento 3D da linha, os retornos desta função serão sempre os mesmos dos retornos para g1 e g2 [ST\\_3DDistance](#).

Disponibilidade: 2.0.0

Alterações: 2.2.0 - se 2 geometrias 2D são entradas, um ponto 2D retorna (em vez do antigo comportamento assumindo 0 para Z perdido). Em caso de 2D e 3D, o Z não é mais 0 para Z perdido.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

## Exemplos

### linestring e ponto -- linhas 3d e 2d mais curtas

```
SELECT ST_AsEWKT(ST_3DShortestLine(line,pt)) AS shl3d_line_pt,
       ST_AsEWKT(ST_ShortestLine(line,pt)) As shl2d_line_pt
FROM (SELECT 'POINT(100 100 30)::geometry As pt,
            'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 1000)::' ←
       geometry As line
       ) As foo;

shl3d_line_pt                                     | ←
-----
shl2d_line_pt                                     +-----+
LINESTRING(54.69937988867619 128.935022917228 11.5475869506606,100 100 30) | ←
LINESTRING(73.0769230769231 115.384615384615,100 100)
```

### linestring e multiponto -- linhas 3d e 2d mais curtas

```
SELECT ST_AsEWKT(ST_3DShortestLine(line,pt)) AS shl3d_line_pt,
       ST_AsEWKT(ST_ShortestLine(line,pt)) As shl2d_line_pt
FROM (SELECT 'MULTIPOINT(100 100 30, 50 74 1000)::geometry As pt,
            'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 900)::' ←
       geometry As line
       ) As foo;

shl2d_line_pt                                     shl3d_line_pt                                     | ←
-----+-----+
LINESTRING(54.69937988867619 128.935022917228 11.5475869506606,100 100 30) | LINESTRING ←
(50 75,50 74)
```



**Multilinestring e polígono linhas 3d e 2d mais curtas**

```

SELECT ST_AsEWKT(ST_3DShortestLine(poly, mline)) As sh13d,
       ST_AsEWKT(ST_ShortestLine(poly, mline)) As sh12d
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;
sh13d ←
| sh12d
-----
LINESTRING(39.993580415989 54.1889925532825 5,40.4078575708294 53.6052383805529 ←
5.03423778139177) | LINESTRING(20 40,20 40)

```

**Veja também**

[ST\\_3DClosestPoint](#), [ST\\_3DDistance](#), [ST\\_LongestLine](#), [ST\\_ShortestLine](#), [ST\\_3DMaxDistance](#)

**8.9.9 ST\_Area**

**ST\_Area** — Retorna a área da superfície se ela for um polígono ou multipolígono. Para geometria, uma área cartesiana 2D é determinada com unidades especificadas pelo SRID. Para geografia, a área é determinada em uma superfície com unidades em metros quadrados.

**Synopsis**

```
float ST_Area(geometry g1);
float ST_Area(geography geog, boolean use_spheroid=true);
```

**Descrição**

Retorna a área da geometria se for um polígono ou multipolígono. Retorna a medida do comprimento de um valor `ST_Surface` ou `ST_MultiSurface`. Para geometria, uma área cartesiana 2D é determinada com unidades especificadas pelo SRID. Para geografia, por padrão, ela é determinada em um esferoide com unidade em metros quadrados. Para medir a esfera mais rápida, mas menos precisa, use: `ST_Area(geog,false)`.

Melhorias: 2.0.0 - suporte a superfícies 2D poliédricas foi introduzido.

Melhorias: 2.2.0 - medição em esferoides desempenhada com GeographicLib para uma melhor precisão e força. Requer Proj >= 4.9.0 para tirar vantagem da nova característica.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 8.1.2, 9.5.3



This function supports Polyhedral surfaces.

**Note**

Para superfícies poliédricas, somente suporta superfícies poliédricas 2D (não 2.5D). Para 2.5D, pode ser dada uma resposta não zero, mas somente para as faces que se encaixam completamente no plano XY.



This method is also provided by SFCGAL backend.

## Exemplos

Retorna uma área em pés quadrado para um terreno de Massachusetts e multiplica pela conversão para metros quadrados. Note que isto é em pés quadrados porque EPSG:2249 é o Massachusetts State Plane Feet

```
SELECT ST_Area(the_geom) As sqft, ST_Area(the_geom)*POWER(0.3048,2) As sqm
      FROM (SELECT
            ST_GeomFromText ('POLYGON((743238 2967416,743238 2967450,
            743265 2967450,743265.625 2967416,743238 2967416))',2249) ) As foo( ←
            the_geom);
```

sqft	sqm
928.625	86.27208552

Retorna uma área em pés quadrados e transforma para Massachusetts state plane em metros (EPSG:26986) para pegar metros quadrados. Note que ele é em pés quadrados, porque 2249 é Massachusetts State Plane Feet e a área transformada está em em metros quadrados já que EPSG:26986 é o state plane Massachusetts em metros

```
SELECT ST_Area(the_geom) As sqft, ST_Area(ST_Transform(the_geom,26986)) As sqm
      FROM (SELECT
            ST_GeomFromText ('POLYGON((743238 2967416,743238 2967450,
            743265 2967450,743265.625 2967416,743238 2967416))',2249) ) As foo( ←
            the_geom);
```

sqft	sqm
928.625	86.2724304199219

Retorna uma área em pés quadrados e metros quadrados usado tipo de dados geografia. Note que transformamos nossa geometria para geografia (antes você pode certificar que sua geometria está em WGS 84 long lat 4326). A geografia sempre mede em metros. Isto é só para demonstração para comparar. Normalmente sua tabela já será armazenada no tipo de dados geografia.

```
SELECT ST_Area(the_geog)/POWER(0.3048,2) As sqft_spheroid, ST_Area(the_geog,false)/POWER( ←
(0.3048,2) As sqft_sphere, ST_Area(the_geog) As sqm_spheroid
      FROM (SELECT
            geography(
            ST_Transform(
            ST_GeomFromText ('POLYGON((743238 2967416,743238 2967450,743265 ←
            2967450,743265.625 2967416,743238 2967416))',
            2249
            ) ,4326
            )
            ) As foo(the_geog);
```

sqft_spheroid	sqft_sphere	sqm_spheroid
928.684403538925	927.049336105925	86.2776042893529

```
--if your data is in geography already
SELECT ST_Area(the_geog)/POWER(0.3048,2) As sqft, ST_Area(the_geog) As sqm
      FROM somegeogtable;
```

## Veja também

[ST\\_GeomFromText](#), [ST\\_GeographyFromText](#), [ST\\_SetSRID](#), [ST\\_Transform](#)

### 8.9.10 ST\_Azimuth

**ST\_Azimuth** — Retorna o azimute baseado em norte como o ângulo em radianos medidos em sentido horário da vertical no pontoA para o pontoB.

## Synopsis

```
float ST_Azimuth(geometria pointA, geometria pointB);
float ST_Azimuth(geografia pointA, geografia pointB);
```

## Descrição

Retorna o azimute em radianos do segmento definido pelos pontos dados, ou NULO se os dois pontos forem coincidentes. O azimute é um ângulo, está referenciado em norte e em sentido horário: Norte = 0; Leste =  $\pi/2$ ; Sul =  $\pi$ ; Oeste =  $3\pi/2$ .

Para o tipo de geografia, o próximo azimute resolvido como parte do problema geodésico inverso.

O azimute é matematicamente conceituado como o ângulo entre um plano de referência e um ponto, com unidades angulares em radianos. As unidades podem ser convertidas para graus usando uma função PostgreSQL graus () embutida, como mostrado no exemplo.

Disponibilidade: 1.1.0

Melhorias: 2.0.0 suporte para geografia foi introduzido.

Melhorias: 2.2.0 medição em esferoides desempenhada com GeographicLib para uma melhor precisão e força. Requer Proj >= 4.9.0 para tirar vantagem da nova característica.

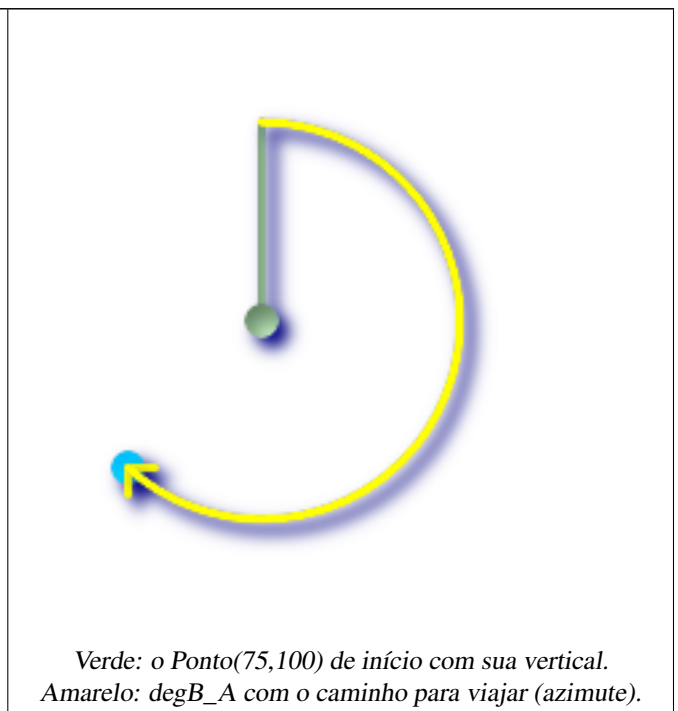
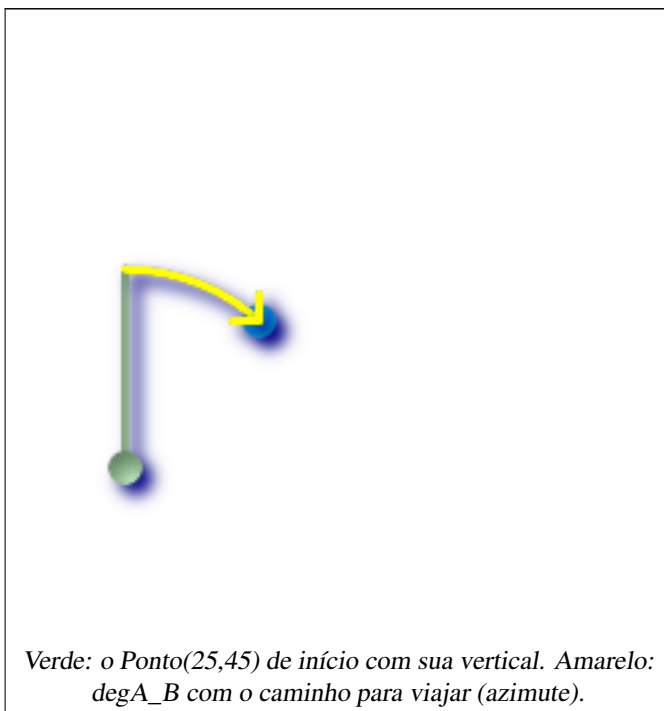
Azimute é especialmente útil em conjunto com ST\_Translate para mudar um objeto ao longo do seu eixo perpendicular. Veja [upgis\\_lineshift](#) [Pgsqlfunctions PostGIS wiki section](#) para um exemplo disto.

## Exemplos

### Azimute da geometria em graus

```
SELECT degrees(ST_Azimuth(ST_Point(25, 45), ST_Point(75, 100))) AS degA_B,
       degrees(ST_Azimuth(ST_Point(75, 100), ST_Point(25, 45))) AS degB_A;
```

dega_b	degb_a
42.2736890060937	222.273689006094



**Veja também**

[ST\\_Point](#), [ST\\_Translate](#), [ST\\_Project](#), [PostgreSQL Math Functions](#)

**8.9.11 ST\_Angle**

`ST_Angle` — Returns the angle between 3 points, or between 2 vectors (4 points or 2 lines).

**Synopsis**

```
float ST_Angle(geometry point1, geometry point2, geometry point3, geometry point4);
float ST_Angle(geometry line1, geometry line2);
```

**Descrição**

For 3 points, computes the angle measured clockwise of P1P2P3. If input are 2 lines, get first and last point of the lines as 4 points. For 4 points, compute the angle measured clockwise of P1P2,P3P4. Results are always positive, between 0 and 2\*Pi radians. Uses azimuth of pairs or points.

`ST_Angle(P1,P2,P3) = ST_Angle(P2,P1,P2,P3)`

Result is in radian and can be converted to degrees using a built-in PostgreSQL function `degrees()`, as shown in the example.

Availability: 2.5.0

**Exemplos****Azimute da geometria em graus**

```
WITH rand AS (
    SELECT s, random() * 2 * PI() AS rad1
           , random() * 2 * PI() AS rad2
    FROM generate_series(1,2,2) AS s
)
, points AS (
    SELECT s, rad1, rad2, ST_MakePoint(cos1+s, sin1+s) as p1, ST_MakePoint(s, s) ←
           AS p2, ST_MakePoint(cos2+s, sin2+s) as p3
    FROM rand
           , cos(rad1) cos1, sin(rad1) sin1
           , cos(rad2) cos2, sin(rad2) sin2
)
SELECT s, ST_AsText(ST_SnapToGrid(ST_MakeLine(ARRAY[p1,p2,p3]),0.001)) AS line
       , degrees(ST_Angle(p1,p2,p3)) as computed_angle
       , round(degrees(2*PI()-rad2 -2*PI()+rad1+2*PI()))::int%360 AS reference
       , round(degrees(2*PI()-rad2 -2*PI()+rad1+2*PI()))::int%360 AS reference
FROM points ;
```

```
1 | line |~computed_angle |~reference
-----+-----
1 | LINESTRING(1.511 1.86,1 1,0.896 0.005) | 155.27033848688 | 155
```

**8.9.12 ST\_Centroid**

`ST_Centroid` — Retorna o centro geométrico de uma geometria.

## Synopsis

```
geometry ST_Centroid(geometry g1);  
geography ST_Centroid(geography g1, boolean use_spheroid=true);
```

## Descrição

Calcula o centro geométrico de uma geometria, ou equivalentemente, o centro de massa da geometria como um POINT. Para [MULTI]POINTS, isto é calculado como o significado aritmético das coordenadas de entrada. Para [MULTI]LINESTRINGS, é calculado como o peso do comprimento de cada segmento de linha. Para [MULTI]POLYGONS, "peso" é pensado em termos de área. Se uma geometria vazia é fornecida, uma GEOMETRYCOLLECTION é retornada. Se NULL é fornecido, NULL retorna. Se CIRCULARSTRING ou COMPOUNDCURVE são fornecidos, eles são convertidos para linestring com CurveToLine primeiro, depois o mesmo para LINESTRING

Novo em 2.3.0 : suporte para CIRCULARSTRING e COMPOUNDCURVE (usando CurveToLine)

Availability: 2.4.0 support for geography was introduced.

O centroide é igual ao centroide do conjunto de componentes de geometria da maior dimensão (já que as geometrias de menor dimensão contribuem com "peso" zero no centroide).



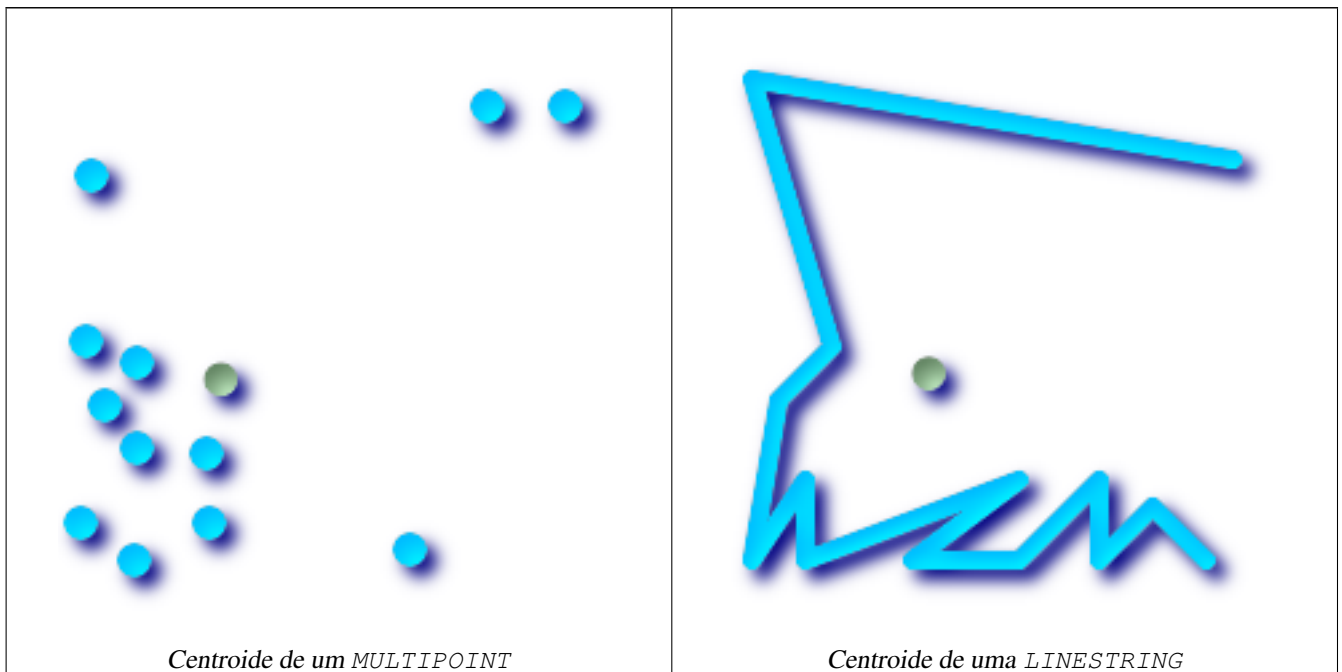
This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).

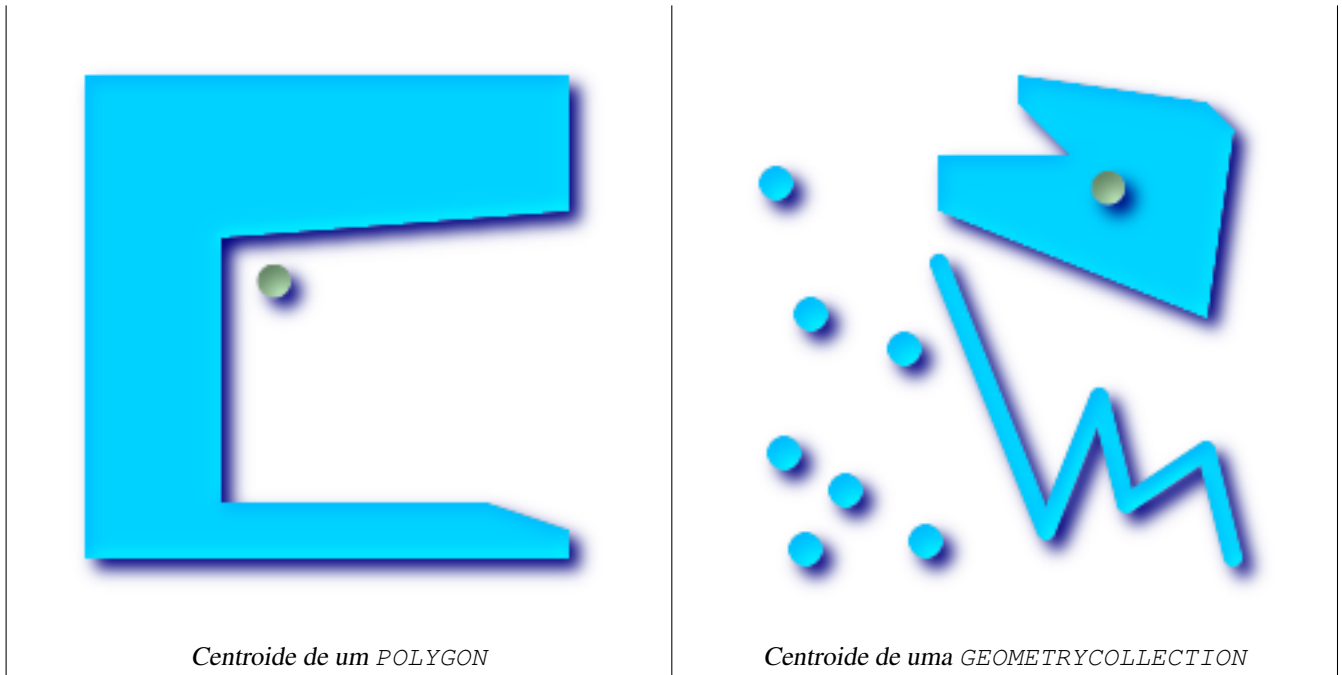


This method implements the SQL/MM specification. SQL-MM 3: 8.1.4, 9.5.5

## Exemplos

Em cada uma das ilustrações seguintes, o ponto azul representa o centroide da geometria fonte.





```

SELECT ST_AsText(ST_Centroid('MULTIPOINT ( -1 0, -1 2, -1 3, -1 4, -1 7, 0 1, 0 3, 1 1, 2 0, 6 0, 7 8, 9 8, 10 6 )'));
                                     st_astext
-----
POINT(2.30769230769231 3.30769230769231)
(1 row)

SELECT ST_AsText(ST_centroid(g))
FROM   ST_GeomFromText ('CIRCULARSTRING(0 2, -1 1,0 0, 0.5 0, 1 0, 2 1, 1 2, 0.5 2, 0 2)') AS g ;
-----
POINT(0.5 1)

SELECT ST_AsText(ST_centroid(g))
FROM   ST_GeomFromText ('COMPOUNDCURVE (CIRCULARSTRING(0 2, -1 1,0 0),(0 0, 0.5 0, 1 0), CIRCULARSTRING( 1 0, 2 1, 1 2),(1 2, 0.5 2, 0 2))' ) AS g;
-----
POINT(0.5 1)

```

#### Veja também

[ST\\_PointOnSurface](#), [ST\\_GeometricMedian](#)

### 8.9.13 ST\_ClosestPoint

`ST_ClosestPoint` — Retorna o ponto 2-dimensional no `g1` que está mais perto de `g2`. Este é o primeiro ponto da menor linha.

#### Synopsis

geometria `ST_ClosestPoint`(geometria `g1`, geometria `g2`);

**Descrição**

Retorna o ponto 2-dimensional no g1 que está mais perto de g2. Este é primeiro ponto da menor linha.



**Note**

Se você tem uma geometria 3D, talvez prefira usar [ST\\_3DClosestPoint](#).

Disponibilidade: 1.5.0

**Exemplos**

*O mais perto entre um ponto e uma linestring é o ponto, mas o ponto mais perto entre uma linestring e um ponto é o ponto na line string que está mais perto.*

```
SELECT ST_AsText(ST_ClosestPoint(pt,line) ↔
) AS cp_pt_line,
      ST_AsText(ST_ClosestPoint(line,pt ↔
)) AS cp_line_pt
FROM (SELECT 'POINT(100 100)::geometry ↔
As pt,
      'LINESTRING (20 80, 98 ↔
190, 110 180, 50 75 )::geometry As line
) As foo;
```

cp_pt_line		↔
cp_line_pt		
-----+		
POINT(100 100)		POINT(73.0769230769231 ↔ 115.384615384615)

*ponto no polígono A mais próximo do polígono B*

```
SELECT ST_AsText (
      ST_ClosestPoint (
        ST_GeomFromText (' ↔
POLYGON((175 150, 20 40, 50 60, 125 100, 175 150 ↔
ST_Buffer( ↔
ST_GeomFromText('POINT(110 170)'), 20)
)
) As ptwkt;

ptwkt
----- ↔
POINT(140.752120669087 125.695053378061)
```

## Veja também

[ST\\_3DClosestPoint](#), [ST\\_Distance](#), [ST\\_LongestLine](#), [ST\\_ShortestLine](#), [ST\\_MaxDistance](#)

### 8.9.14 ST\_ClusterDBSCAN

`ST_ClusterDBSCAN` — Função do Windows que retorna id inteira para o grupo que cada geometria de entrada está, baseada na implementação 2D de agrupamento de densidade espacial de aplicações com algoritmo barulhento (DBSCAN).

#### Synopsis

inteiro `ST_ClusterDBSCAN`(geometria winset geom, float8 eps, inteiro minpoints);

#### Descrição

Returns cluster number for each input geometry, based on a 2D implementation of the [Density-based spatial clustering of applications with noise \(DBSCAN\)](#) algorithm. Unlike `ST_ClusterKMeans`, it does not require the number of clusters to be specified, but instead uses the desired `distance` (eps) and density (`minpoints`) parameters to construct each cluster.

Uma geometria de entrada será adicionada para um grupo se ele for:

- A "core" geometry, that is within eps `distance` of at least `minpoints` input geometries (including itself) or
- A "border" geometry, that is within eps `distance` of a core geometry.

Note que as geometrias de borda podem estar dentro da distância eps de geometrias centrais e mais de um grupo; neste caso, qualquer atribuição estaria correta, e a geometria de borda seria designada arbitrariamente para um dos grupos disponíveis. Nestes casos, é possível um grupo correto ser gerado com menos que geometrias `minpoints`. Quando a atribuição de uma geometria de borda for ambígua, chamadas repetidas para `ST_ClusterDBSCAN` irão produzir resultados idênticos se uma frase `ORDER BY` estiver incluída na definição da janela, mas as atribuições podem diferenciar de outras implementações do mesmo algoritmo.



#### Note

Geometrias de entrada que não encontram o critério para ingressar a qualquer outro grupo serão designadas para um número de grupo NULO.

---

Disponibilidade: 2.3.0 - requer GEOS

#### Exemplos

Assigning a cluster number to each polygon within 50 meters of each other. Require at least 2 polygons per cluster

---





*within 50 meters at least 2 per cluster. singletons have NULL for cid*

```
SELECT name, ST_ClusterDBSCAN(geom, eps ←
:= 50, minpoints := 2) over () AS cid
FROM boston_polys
WHERE name > '' AND building > ''
AND ST_DWithin(geom,
ST_Transform(
ST_GeomFromText('POINT ←
(-71.04054 42.35141)', 4326), 26986),
500);
```

bucket	name		↔
0	Manulife Tower		↔
0	Park Lane Seaport I		↔
0	Park Lane Seaport II		↔
0	Renaissance Boston Waterfront Hotel		↔
0	Seaport Boston Hotel		↔
0	Seaport Hotel & World Trade Center		↔
0	Waterside Place		↔
0	World Trade Center East		↔
1	100 Northern Avenue		↔
1	100 Pier 4		↔
1	The Institute of Contemporary Art		↔
2	101 Seaport		↔
2	District Hall		↔
2	One Marina Park Drive		↔
2	Twenty Two Liberty		↔
2	Vertex		↔
2	Vertex		↔
2	Watermark Seaport		↔
NULL	Blue Hills Bank Pavilion		↔
NULL	World Trade Center West		↔
(20 rows)			

Combinando parcelas com o mesmo número de grupo dentro de uma única geometria. Utiliza chamada de argumentos nomeados

```
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;
```

**Veja também**

[ST\\_DWithin](#), [ST\\_ClusterKMeans](#), [ST\\_ClusterIntersecting](#), [ST\\_ClusterWithin](#)

### 8.9.15 ST\_ClusterIntersecting

`ST_ClusterIntersecting` — Agregado. Retorna um arranjo com os componentes conectados de um conjunto de geometrias

#### Synopsis

```
geometry[] ST_ClusterIntersecting(geometry set g);
```

#### Descrição

A `ST_ClusterIntersecting` é uma função agregada que retorna um arranjo de coleções de geometrias, onde cada coleção representa um conjunto de geometrias interconectadas.

Disponibilidade: 2.2.0 - requer GEOS

#### Exemplos

```
WITH testdata AS
  (SELECT unnest(ARRAY['LINESTRING (0 0, 1 1)::geometry',
                      'LINESTRING (5 5, 4 4)::geometry',
                      'LINESTRING (6 6, 7 7)::geometry',
                      'LINESTRING (0 0, -1 -1)::geometry',
                      'POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0))::geometry']) AS geom)

SELECT ST_AsText(unnest(ST_ClusterIntersecting(geom))) FROM testdata;

--result

st_astext
-----
GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),LINESTRING(5 5,4 4),LINESTRING(0 0,-1 -1),POLYGON((0 ←
  0,4 0,4 4,0 4,0 0)))
GEOMETRYCOLLECTION(LINESTRING(6 6,7 7))
```

#### Veja também

[ST\\_ClusterDBSCAN](#), [ST\\_ClusterKMeans](#), [ST\\_ClusterWithin](#)

### 8.9.16 ST\_ClusterKMeans

`ST_ClusterKMeans` — Função do Windows que retorna id inteira para o grupo que cada geometria de entrada está.

#### Synopsis

```
inteiro ST_ClusterKMeans(geometria winset geom, inteiro number_of_clusters);
```

#### Descrição

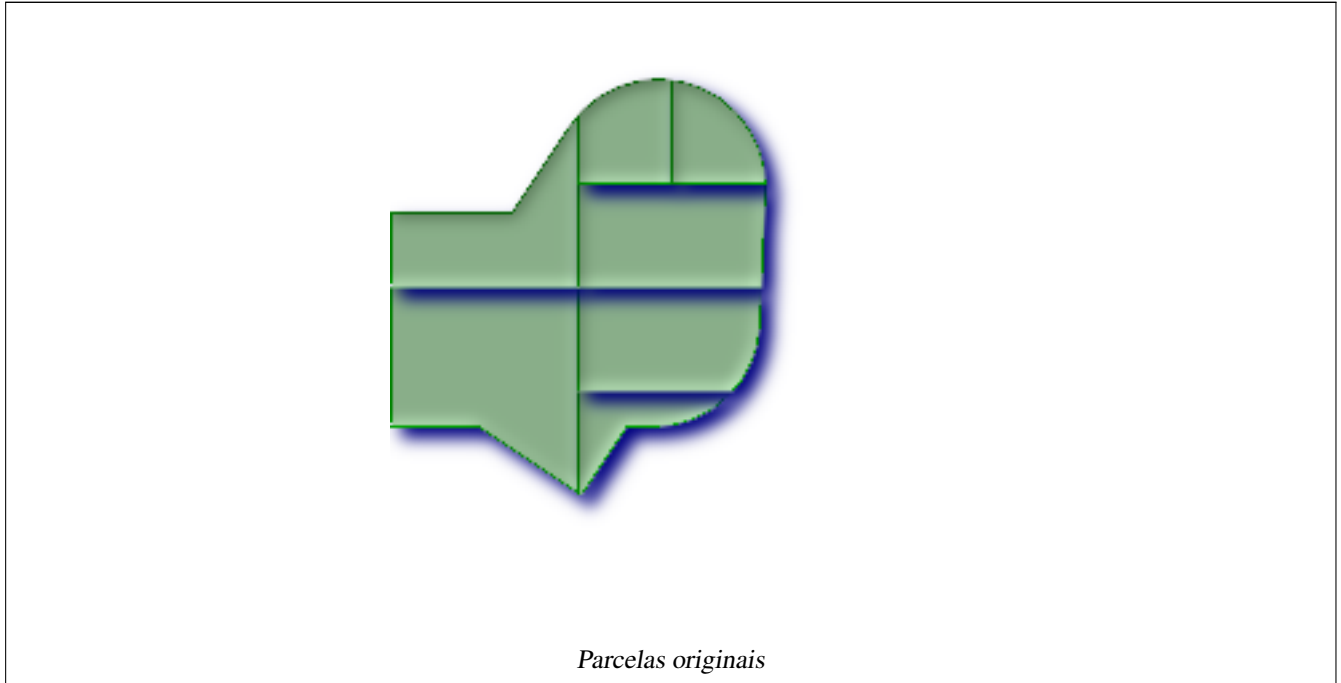
Retorna uma distância 2D **k-means** número de grupo para cada geometria de entrada. A distância usada para agrupar é a distância entre as centroides das geometrias.

Disponibilidade: 2.3.0 - requer GEOS

## Exemplos

### Gera conjunto fictício de parcelas para exemplos

```
CREATE TABLE parcels AS
SELECT lpad((row_number() over())::text,3,'0') As parcel_id, geom,
('{residential, commercial}'::text[])[1 + mod(row_number()OVER(),2)] As type
FROM
  ST_Subdivide(ST_Buffer('LINESTRING(40 100, 98 100, 100 150, 60 90)'::geometry,
    40, 'endcap=square'),12) As geom;
```



```
-- Partitioning parcel clusters by type
SELECT ST_ClusterKMeans(geom,3) over (PARTITION BY type) AS cid, parcel_id, type
FROM parcels;
-- result
cid | parcel_id | type
-----+-----+-----
  1 | 005      | commercial
  1 | 003      | commercial
  2 | 007      | commercial
  0 | 001      | commercial
  1 | 004      | residential
  0 | 002      | residential
  2 | 006      | residential
(7 rows)
```

### Veja também

[ST\\_ClusterDBSCAN](#), [ST\\_ClusterIntersecting](#), [ST\\_ClusterWithin](#), [ST\\_Subdivide](#)

### 8.9.17 ST\_ClusterWithin

**ST\_ClusterWithin** — Agregado. Retorna um arranjo de coleções de geometrias, onde cada coleção representa um conjunto de geometrias separados por nada mais que a distância especificada.

**Synopsis**

geometria[] **ST\_ClusterWithin**(geometry set g, float8 distância);

**Descrição**

**ST\_ClusterWithin** is an aggregate function that returns an array of **GeometryCollections**, where each **GeometryCollection** represents a set of geometries separated by no more than the specified distance. (Distances are Cartesian distances in the units of the SRID.)

Disponibilidade: 2.2.0 - requer GEOS

**Exemplos**

```
WITH testdata AS
  (SELECT unnest(ARRAY['LINESTRING (0 0, 1 1)::geometry,
                      'LINESTRING (5 5, 4 4)::geometry,
                      'LINESTRING (6 6, 7 7)::geometry,
                      'LINESTRING (0 0, -1 -1)::geometry,
                      'POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0))::geometry']) AS geom)

SELECT ST_AsText(unnest(ST_ClusterWithin(geom, 1.4))) FROM testdata;

--result

st_astext
-----
GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),LINESTRING(5 5,4 4),LINESTRING(0 0,-1 -1),POLYGON((0 ←
  0,4 0,4 4,0 4,0 0)))
GEOMETRYCOLLECTION(LINESTRING(6 6,7 7))
```

**Veja também**

[ST\\_ClusterDBSCAN](#), [ST\\_ClusterKMeans](#), [ST\\_ClusterIntersecting](#)

**8.9.18 ST\_Contains**

**ST\_Contains** — Retorna verdade se nenhum ponto de B estiverem no exterior de A, e pelo menos um ponto do interior de B estiver no interior de A.

**Synopsis**

booleana **ST\_Contains**(geometria geomA, geometria geomB);

**Descrição**

Geometria A contém Geometria B se e somente se nenhum ponto de B estiver no exterior de A, e pelo menos um ponto do interior de B estiver no interior de A. Uma sutileza importante desta definição é que A não contém seu limite, mas A se contém. Contraste isto a [ST\\_ContainsProperly](#) onde a geometria A não se contenha propriamente.

Retorna VERDADE se a geometria B estiver completamente dentro da A. Para esta função funcionar, as geometrias fonte devem ser da mesma projeção de coordenada, tendo o mesmo SRID. **ST\_Contains** é o inverso de **ST\_Within**. Então, **ST\_Contains(A,B)** sugere que **ST\_Within(B,A)**, exceto nos casos de geometrias inválidas, onde o resultado é sempre falso ou não definido.

Desempenhado pelo módulo GEOS

Melhorias: 2.3.0 Aprimoramento para o pequeno circuito estendido PIP para suportar MultiPOntos com pouco pontos. Versões anteriores só suportavam pontos em polígonos.



### Important

Não chama com uma `GEOMETRYCOLLECTION` como um argumento



### Important

Não usa esta função com geometrias inválidas. Você obterá resultados inesperados.

A chamada desta função irá, automaticamente, incluir uma comparação de caixa delimitadora que fará uso de quaisquer índices que esteja disponíveis nas geometrias. Para evitar o uso de índice, utilize a função `_ST_Contains`.

NOTA: esta é a versão "permissível" que retorna um booleano, não um inteiro.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 // s2.1.13.3 - same as `within(geometry B, geometry A)`

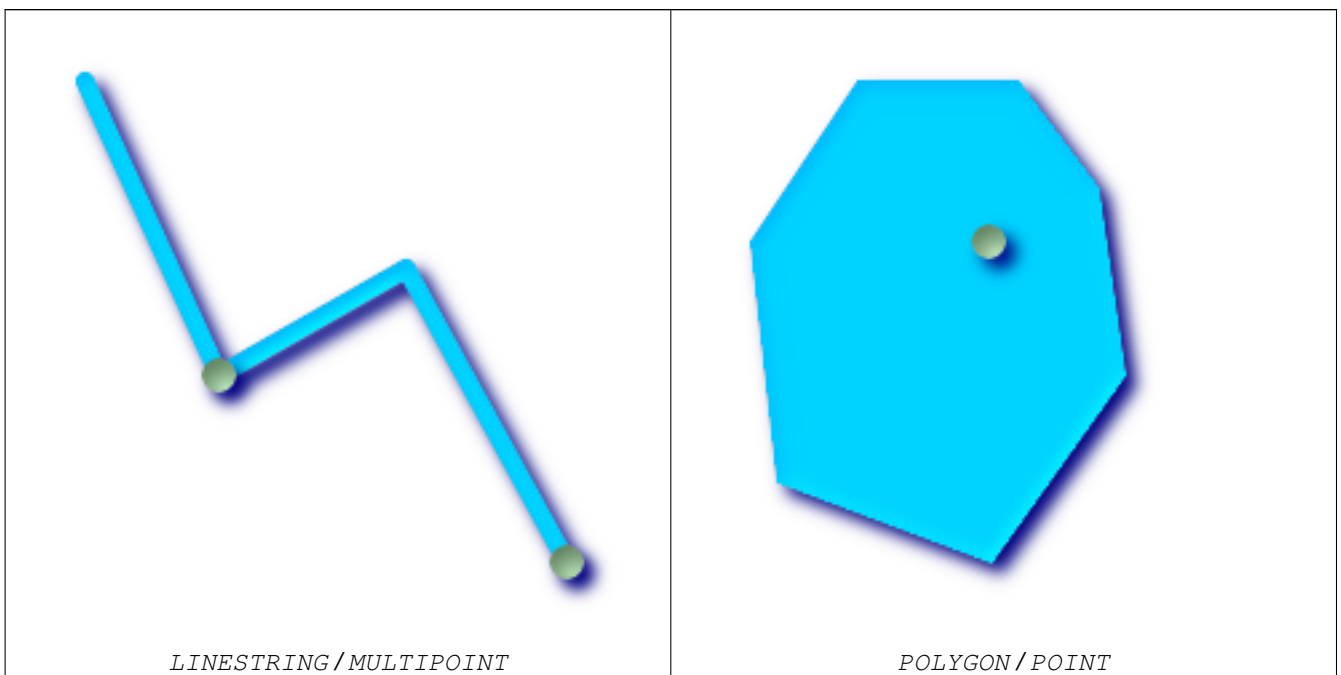


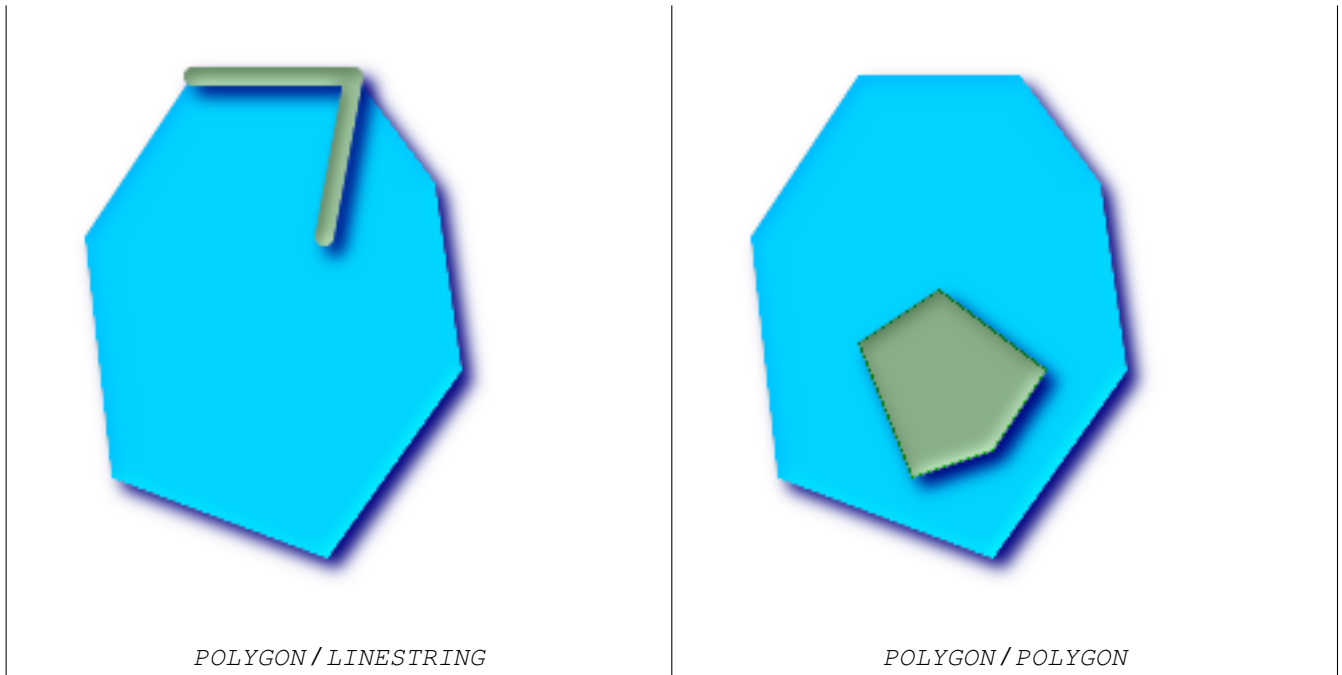
This method implements the SQL/MM specification. SQL-MM 3: 5.1.31

Existem certas sutilezas para `ST_Contains` e `ST_Within` que não são tão óbvias. Para detalhes, veja: [Subtleties of OGC Covers, Contains, Within](#)

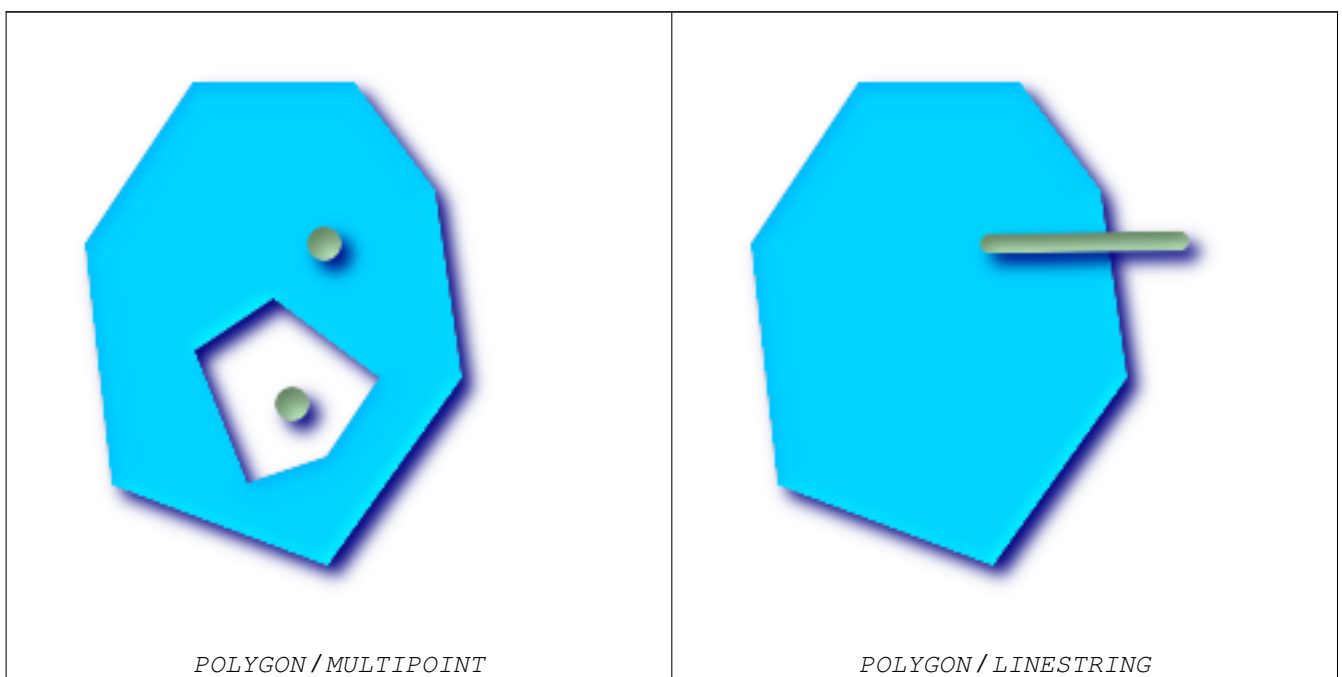
## Exemplos

O predicado `ST_Contains` retorna `TRUE` em todas as ilustrações seguintes.





O predicado `ST_Contains` retorna FALSE em todas as ilustrações seguintes.



```
-- A circle within a circle
SELECT ST_Contains(smallc, bigc) As smallcontainsbig,
       ST_Contains(bigc, smallc) As bigcontainssmall,
       ST_Contains(bigc, ST_Union(smallc, bigc)) as bigcontainsunion,
       ST_Equals(bigc, ST_Union(smallc, bigc)) as bigisunion,
       ST_Covers(bigc, ST_ExteriorRing(bigc)) As bigcoversexterior,
       ST_Contains(bigc, ST_ExteriorRing(bigc)) As bigcontainsexterior
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
          ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;

-- Result
```





**Important**

Não chama com uma `GEOMETRYCOLLECTION` como um argumento



**Important**

Não usa esta função com geometrias inválidas. Você obterá resultados inesperados.

A chamada desta função irá, automaticamente, incluir uma comparação de caixa delimitadora que fará uso de quaisquer índices que esteja disponíveis nas geometrias. Para evitar o uso de índice, utilize a função `ST_ContainsProperly`.

**Exemplos**

```
--a circle within a circle
SELECT ST_ContainsProperly(smallc, bigc) As smallcontainspropbig,
       ST_ContainsProperly(bigc,smallc) As bigcontainspropsmall,
       ST_ContainsProperly(bigc, ST_Union(smallc, bigc)) as bigcontainspropunion,
       ST_Equals(bigc, ST_Union(smallc, bigc)) as bigisunion,
       ST_Covers(bigc, ST_ExteriorRing(bigc)) As bigcoversexterior,
       ST_ContainsProperly(bigc, ST_ExteriorRing(bigc)) As bigcontainsexterior
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
       ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;
--Result
smallcontainspropbig | bigcontainspropsmall | bigcontainspropunion | bigisunion |
bigcoversexterior | bigcontainsexterior
-----+-----+-----+-----+-----+-----
f                    | t                    | f                    | t          |
                    | f                    |                      |            |

--example demonstrating difference between contains and contains properly
SELECT ST_GeometryType(geomA) As geomtype, ST_Contains(geomA,geomA) AS acontainsa,
       ST_ContainsProperly(geomA, geomA) AS acontainspropa,
       ST_Contains(geomA, ST_Boundary(geomA)) As acontainsba, ST_ContainsProperly(geomA,
       ST_Boundary(geomA)) As acontainspropba
FROM (VALUES ( ST_Buffer(ST_Point(1,1), 5,1) ),
            ( ST_MakeLine(ST_Point(1,1), ST_Point(-1,-1) ) ),
            ( ST_Point(1,1) )
        ) As foo(geomA);

geomtype | acontainsa | acontainspropa | acontainsba | acontainspropba
-----+-----+-----+-----+-----
ST_Polygon | t         | f             | f           | f
ST_LineString | t        | f             | f           | f
ST_Point | t         | t             | f           | f
```

**Veja também**

[ST\\_GeometryType](#), [ST\\_Boundary](#), [ST\\_Contains](#), [ST\\_Covers](#), [ST\\_CoveredBy](#), [ST\\_Equals](#), [ST\\_Relate](#), [ST\\_Within](#)

**8.9.20 ST\_Covers**

`ST_Covers` — Retorna 1 (VERDADE) se nenhum ponto na geometria B estiver fora da geometria A



## Synopsis

```
boolean ST_Covers(geometry geomA, geometry geomB);
boolean ST_Covers(geography geogpolyA, geography geogpointB);
```

## Descrição

Retorna 1 (VERDADE) se nenhum ponto na geometria/geografia B estiver fora da geometria/geografia A  
Desempenhado pelo módulo GEOS



### Important

Não chama com uma `GEOMETRYCOLLECTION` como um argumento



### Important

Não usa esta função com geometrias inválidas. Você obterá resultados inesperados.

A chamada desta função irá, automaticamente, incluir uma comparação de caixa delimitadora que fará uso de quaisquer índices que esteja disponíveis nas geometrias. Para evitar o uso de índice, utilize a função `_ST_Covers`.

Enhanced: 2.4.0 Support for polygon in polygon and line in polygon added for geography type

Melhorias: 2.3.0 Aprimoramento para o pequeno circuito estendido PIP para suportar MultiPontos com poucos pontos. Versões anteriores só suportavam pontos em polígonos.

Disponibilidade: 1.5 - suporte para geografia foi introduzido.

Disponibilidade: 1.2.2 - requer GEOS >= 3.0

NOTA: esta é a versão "permissível" que retorna um booleano, não um inteiro.

Não é um padrão OGC, mas o Oracle tem também.

Existem certas sutilezas para `ST_Contains` e `ST_Within` que não são tão óbvias. Para detalhes, veja: [Subtleties of OGC Covers, Contains, Within](#)

## Exemplos

### Exemplo de geometria

```
--a circle covering a circle
SELECT ST_Covers(smallc,smallc) As smallinsmall,
       ST_Covers(smallc, bigc) As smallcoversbig,
       ST_Covers(bigc, ST_ExteriorRing(bigc)) As bigcoversexterior,
       ST_Contains(bigc, ST_ExteriorRing(bigc)) As bigcontainsexterior
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
         ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;
--Result
 smallinsmall | smallcoversbig | bigcoversexterior | bigcontainsexterior
-----+-----+-----+-----
 t           | f              | t                 | f
(1 row)
```

### Exemplo de Geografia

```
-- a point with a 300 meter buffer compared to a point, a point and its 10 meter buffer
SELECT ST_Covers(geog_poly, geog_pt) As poly_covers_pt,
       ST_Covers(ST_Buffer(geog_pt,10), geog_pt) As buff_10m_covers_cent
FROM (SELECT ST_Buffer(ST_GeogFromText('SRID=4326;POINT(-99.327 31.4821)'), 300) As ←
      geog_poly,
      ST_GeogFromText('SRID=4326;POINT(-99.33 31.483)') As ←
      geog_pt ) As foo;
```

poly_covers_pt	buff_10m_covers_cent
f	t

## Veja também

[ST\\_Contains](#), [ST\\_CoveredBy](#), [ST\\_Within](#)

### 8.9.21 ST\_CoveredBy

`ST_CoveredBy` — Retorna 1 (VERDADE) se nenhum ponto na geometria/geografia A estiver fora da geometria/geografia B

#### Synopsis

boolean `ST_CoveredBy`(geometry geomA, geometry geomB);  
 boolean `ST_CoveredBy`(geography geogA, geography geogB);

#### Descrição

Retorna 1 (VERDADE) se nenhum ponto na geometria/geografia A estiver fora da geometria/geografia B

Desempenhado pelo módulo GEOS



#### Important

Não chama com uma `GEOMETRYCOLLECTION` como um argumento



#### Important

Não usa esta função com geometrias inválidas. Você obterá resultados inesperados.

Disponibilidade: 1.2.2 - requer GEOS >= 3.0

A chamada desta função irá, automaticamente, incluir uma comparação de caixa delimitadora que fará uso de quaisquer índices que esteja disponíveis nas geometrias. Para evitar o uso de índice, utilize a função `_ST_CoveredBy`.

NOTA: esta é a versão "permissível" que retorna um booleano, não um inteiro.

Não é um padrão OGC, mas o Oracle tem também.

Existem certas sutilezas para `ST_Contains` e `ST_Within` que não são tão óbvias. Para detalhes, veja: [Subtleties of OGC Covers, Contains, Within](#)

## Exemplos

```
--a circle coveredby a circle
SELECT ST_CoveredBy(smallc,smallc) As smallinsmall,
       ST_CoveredBy(smallc, bigc) As smallcoveredbybig,
       ST_CoveredBy(ST_ExteriorRing(bigc), bigc) As exteriorcoveredbybig,
       ST_Within(ST_ExteriorRing(bigc),bigc) As exeriorwithinbig
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
       ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;
--Result
smallinsmall | smallcoveredbybig | exteriorcoveredbybig | exeriorwithinbig
-----+-----+-----+-----
t           | t                 | t                     | f
(1 row)
```

## Veja também

[ST\\_Contains](#), [ST\\_Covers](#), [ST\\_ExteriorRing](#), [ST\\_Within](#)


## 8.9.22 ST\_Crosses

`ST_Crosses` — Retorna TRUE se as geometrias fornecidas têm alguns, não todos, pontos em comum.

### Synopsis

boolean `ST_Crosses`(geometry g1, geometry g2);

### Descrição

`ST_Crosses` pega dois objetos de geometria e retorna TRUE se sua interseção "cruza espacialmente", isto é, as geometrias têm alguns, não todos, pontos interiores em comum. A interseção dos interiores das geometrias não deve ser vazia e deve ter uma dimensionalidade menor que o máximo das duas geometrias de entrada. Além disso, a interseção das duas geometrias não deve ser igual às geometrias fonte. Caso contrário,  retorna FALSE.

Em termos matemáticos, é expressado como:

$$a.Crosses(b) \Leftrightarrow (dim(I(a) \cap I(b)) < max(dim(I(a)), dim(I(b)))) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

A DE-9IM Intersection Matrix para as duas geometrias é:

- T\*T\*\*\*\*\* (para Ponto/Linha, Ponto/Área, e situações de Linha/Área)
- T\*\*\*\*\*T\*\* (para Linha/Ponto, Área/Ponto, e situações de Área/Linha)
- 0\*\*\*\*\* (para situações de Linha/Linha)

Para qualquer outra combinação de dimensões, este predicado retorna falso.

As especificações de características simples definem este predicado somente para Ponto/Linha, Ponto/Área, Linha/Linha, e Linha/Área. JTS/ GEOS estende a definição para aplicar a Linha/Ponto, Área/Ponto e Área/Linha também. Isto torna a relação simétrica.



### Important

Não chama com uma `GEOMETRYCOLLECTION` como um argumento

**Note**

Esta função pode, automaticamente, incluir uma comparação de caixa delimitadora que usará quaisquer índices que estejam disponíveis nas geometrias.



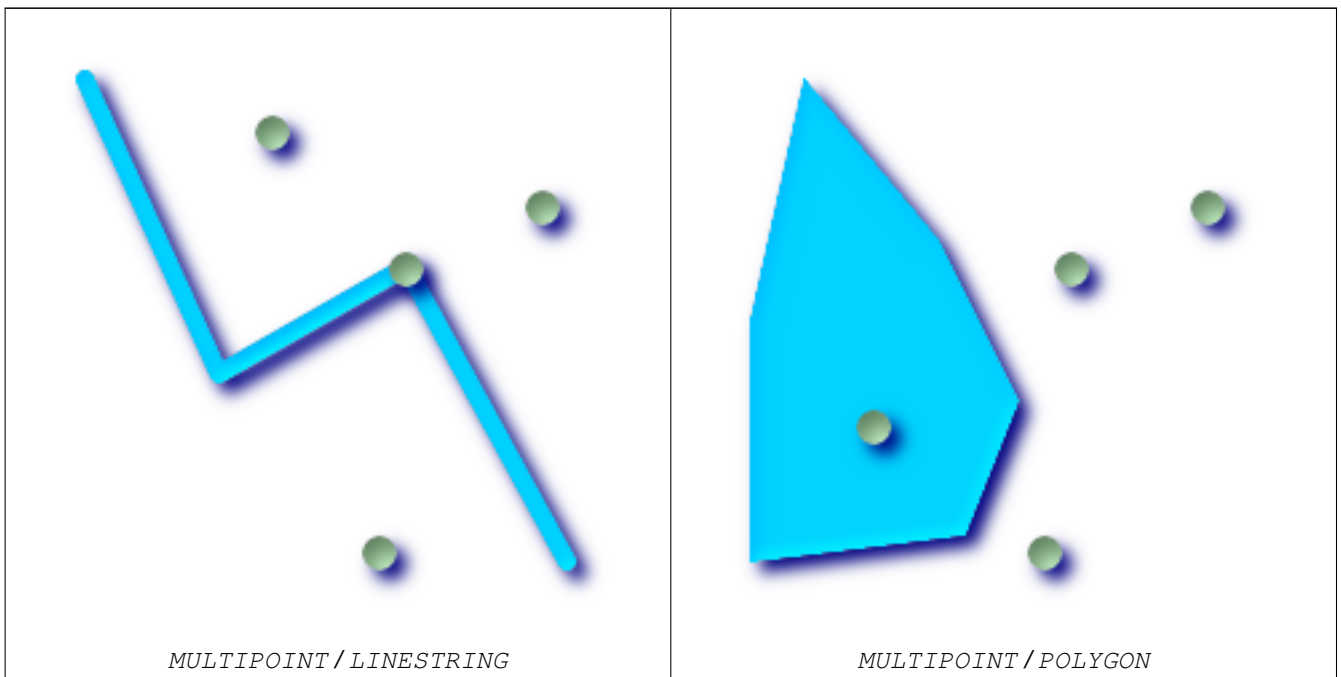
This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1. s2.1.13.3](#)

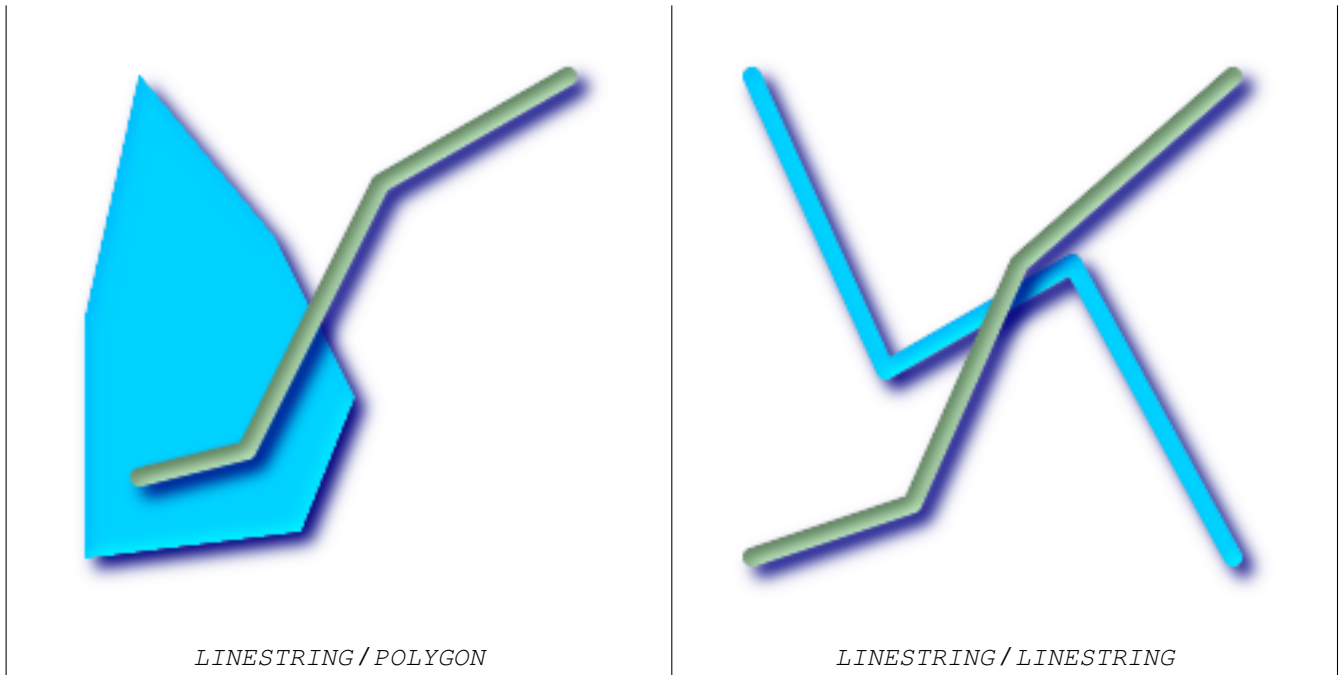


This method implements the SQL/MM specification. SQL-MM 3: 5.1.29

**Exemplos**

Todas as ilustrações seguintes retornam TRUE.





Considere uma situação onde um usuário tem duas tabelas: uma tabela de ruas e outra de estradas.

```
CREATE TABLE roads (
  id serial NOT NULL,
  the_geom geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↔
    road_id)
);
```

```
CREATE TABLE highways (
  id serial NOT NULL,
  the_gem geometry,
  CONSTRAINT roads_pkey PRIMARY KEY ( ↔
    road_id)
);
```

Para determinar uma lista de ruas que cruzam uma estrada, use uma questão parecida com:

```
SELECT roads.id
FROM roads, highways
WHERE ST_Crosses(roads.the_geom, highways.the_geom);
```

### 8.9.23 ST\_LineCrossingDirection

**ST\_LineCrossingDirection** — Dadas as 2 linestrings, retorna um número entre -3 e 3 indicando qual tipo de comportamento de travessia. 0 não é travessia.

#### Synopsis

integer **ST\_LineCrossingDirection**(geometry linestringA, geometry linestringB);

#### Descrição

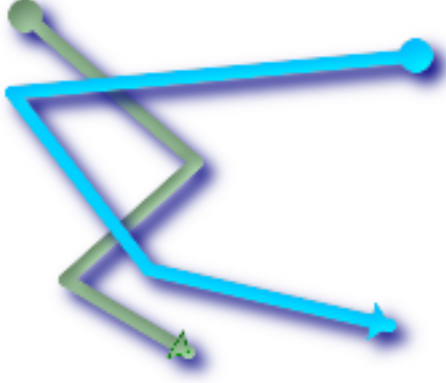
Dadas as 2 linestrings, retorna um número entre -3 e 3 indicando qual tipo de comportamento de travessia. 0 não é travessia. Só é suportado para **LINESTRING**

Definição das constantes inteiras a seguir:

- 0: LINHA NÃO CRUZA
- -1: LINHA CRUZA À ESQUERDA
- 1: LINHA CRUZA À DIREITA
- -2: FIM LINHA MULTICRUZA À ESQUERDA
- 2: FIM LINHA MULTICRUZA À DIREITA
- -3: LINHA MULTICRUZA MESMO FIM PRIMEIRO À ESQUERDA
- 3: LINHA MULTICRUZA MESMO FIM PRIMEIRO À DIREITA

Disponibilidade: 1.4

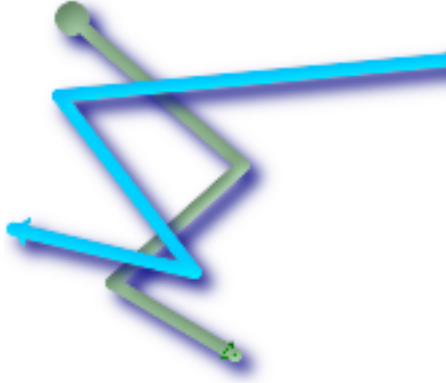
### Exemplos



*Linha 1 (verde), Linha 2 a bola é o ponto inicial, triângulos são pontos finais. Consulta abaixo.*

```
SELECT ST_LineCrossingDirection(foo.line1 ↔
, foo.line2) As l1_cross_l2 ,
ST_LineCrossingDirection(foo. ↔
line2, foo.line1) As l2_cross_l1
FROM (
SELECT
ST_GeomFromText('LINESTRING(25 169,89 ↔
114,40 70,86 43)') As line1,
ST_GeomFromText('LINESTRING(171 154,20 ↔
140,71 74,161 53)') As line2
) As foo;
```

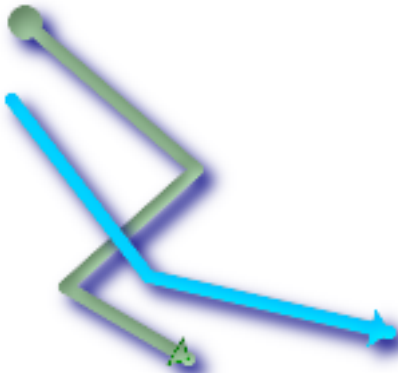
l1_cross_l2	l2_cross_l1
3	-3



*Linha 1 (verde), Linha 2 (azul) a bola é o ponto inicial, triângulos são pontos finais. Consulta abaixo.*

```
SELECT ST_LineCrossingDirection(foo.line1 ↔
, foo.line2) As l1_cross_l2 ,
ST_LineCrossingDirection(foo. ↔
line2, foo.line1) As l2_cross_l1
FROM (
SELECT
ST_GeomFromText('LINESTRING(25 169,89 ↔
114,40 70,86 43)') As line1,
ST_GeomFromText('LINESTRING (171 154, ↔
20 140, 71 74, 2.99 90.16)') As line2
) As foo;
```

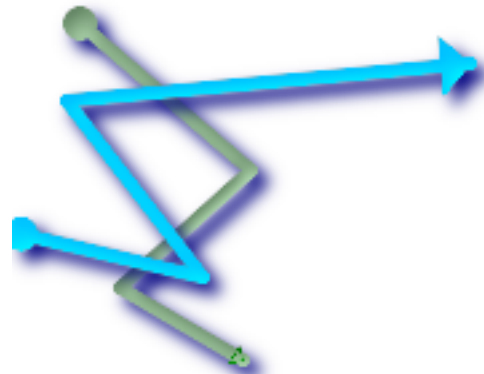
l1_cross_l2	l2_cross_l1
2	-2



*Linha 1 (verde), Linha 2 (azul) a bola é o ponto inicial, triângulos são pontos finais. Consulta abaixo.*

```
SELECT
    ST_LineCrossingDirection(foo. ↵
line1, foo.line2) As l1_cross_l2 ,
    ST_LineCrossingDirection(foo. ↵
line2, foo.line1) As l2_cross_l1
FROM (
    SELECT
        ST_GeomFromText('LINESTRING(25 169,89 ↵
114,40 70,86 43)') As line1,
        ST_GeomFromText('LINESTRING (20 140, 71 ↵
74, 161 53)') As line2
    ) As foo;

l1_cross_l2 | l2_cross_l1
-----+-----
-1 |          1
```



*Linha 1 (verde), Linha 2 (azul) a bola é o ponto inicial, triângulos são pontos finais. Consulta abaixo.*

```
SELECT ST_LineCrossingDirection(foo.line1 ↵
, foo.line2) As l1_cross_l2 ,
    ST_LineCrossingDirection(foo. ↵
line2, foo.line1) As l2_cross_l1
FROM (SELECT
    ST_GeomFromText('LINESTRING(25 ↵
169,89 114,40 70,86 43)') As line1,
    ST_GeomFromText('LINESTRING(2.99 ↵
90.16,71 74,20 140,171 154)') As line2
    ) As foo;

l1_cross_l2 | l2_cross_l1
-----+-----
-2 |          2
```

```
SELECT s1.gid, s2.gid, ST_LineCrossingDirection(s1.the_geom, s2.the_geom)
FROM streets s1 CROSS JOIN streets s2 ON (s1.gid != s2.gid AND s1.the_geom && s2. ↵
the_geom )
WHERE ST_CrossingDirection(s1.the_geom, s2.the_geom)
> 0;
```

**Veja também**

[ST\\_Crosses](#)

**8.9.24 ST\_Disjoint**

ST\_Disjoint — Retorna VERDADE se as geometrias não se "intersectam espacialmente" - se elas não dividem nenhum espaço.

## Synopsis

boolean **ST\_Disjoint**( geometry A , geometry B );

## Descrição

Sobrepõe, toca, dentro de todas as interseções espaciais implicadas. Se qualquer dos anteriores retornarem verdade, então as geometrias não são desconjuntadas. Desconjuntar implica em falso para interseção espacial.



### Important

Não chama com uma `GEOMETRYCOLLECTION` como um argumento

Desempenhado pelo módulo GEOS



### Note

A chamada desta função não usa índices



### Note

NOTA: esta é a versão "permissível" que retorna um booleano, não um inteiro.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). `s2.1.1.2 //s2.1.13.3 - a.Relate(b, 'FF*FF****')`



This method implements the SQL/MM specification. SQL-MM 3: 5.1.26

## Exemplos

```
SELECT ST_Disjoint('POINT(0 0)::geometry, 'LINESTRING ( 2 0, 0 2 ) '::geometry);
st_disjoint
-----
t
(1 row)
SELECT ST_Disjoint('POINT(0 0)::geometry, 'LINESTRING ( 0 0, 0 2 ) '::geometry);
st_disjoint
-----
f
(1 row)
```

## Veja também

[ST\\_Intersects](#)

## 8.9.25 ST\_Distance

**ST\_Distance** — For geometry type returns the 2D Cartesian distance between two geometries in projected units (based on spatial reference system). For geography type defaults to return minimum geodesic distance between two geographies in meters.



## Synopsis

```
float ST_Distance(geometry g1, geometry g2);
float ST_Distance(geography gg1, geography gg2);
float ST_Distance(geography gg1, geography gg2, boolean use_spheroid);
```

## Descrição

For **geometry** type returns the minimum 2D Cartesian distance between two geometries in projected units (spatial ref units). For **geografia** type defaults to return the minimum geodesic distance between two geographies in meters. If `use_spheroid` is false, a faster sphere calculation is used instead of a spheroid.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 5.1.23



This method supports Circular Strings and Curves



This method is also provided by SFCGAL backend.

Disponibilidade: 1.5.0 suporte de geografia foi introduzido em 1.5. Melhorias na velocidade para planar para lidar melhor com mais ou maiores vértices de geometrias.

Melhorias: 2.1.0 velocidade melhorada para geografia. Veja [Making Geography faster](#) para mais detalhes.

Melhorias: 2.1.0 - suporte para geometrias curvas foi introduzido.

Melhorias: 2.2.0 - medição em esferoides desempenhada com GeographicLib para uma melhor precisão e força. Requer Proj >= 4.9.0 para tirar vantagem da nova característica.

## Exemplos de Geometria Básicos

```
--Geometry example - units in planar degrees 4326 is WGS 84 long lat unit=degrees
SELECT ST_Distance(
    'SRID=4326;POINT(-72.1235 42.3521)::geometry',
    'SRID=4326;LINESTRING(-72.1260 42.45, -72.123 42.1546)::geometry'
);
st_distance
-----
0.00150567726382282

-- Geometry example - units in meters (SRID: 3857, proportional to pixels on popular web ↔
  maps)
-- although the value is off, nearby ones can be compared correctly,
-- which makes it a good choice for algorithms like KNN or KMeans.
SELECT ST_Distance(
    ST_Transform('SRID=4326;POINT(-72.1235 42.3521)::geometry', 3857),
    ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45, -72.123 42.1546) ↔
    '::geometry', 3857)
);
st_distance
-----
167.441410065196

-- Geometry example - units in meters (SRID: 3857 as above, but corrected by cos(lat) to ↔
  account for distortion)
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);
st_distance
-----
123.742351254151

-- Geometry example - units in meters (SRID: 26986 Massachusetts state plane meters) (most ←
  accurate for Massachusetts)
SELECT ST_Distance(
        ST_Transform('SRID=4326;POINT(-72.1235 42.3521)::geometry, 26986),
        ST_Transform('SRID=4326;LINESTRING(-72.1260 42.45, -72.123 42.1546) ←
        '::geometry, 26986)
    );
st_distance
-----
123.797937878454

-- Geometry example - units in meters (SRID: 2163 US National Atlas Equal area) (least ←
  accurate)
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)
    );
st_distance
-----
126.664256056812

```

## Exemplos de Geografia

```

-- same as geometry example but note units in meters - use sphere for slightly faster less ←
  accurate
SELECT ST_Distance(gg1, gg2) As spheroid_dist, ST_Distance(gg1, gg2, false) As sphere_dist
FROM (SELECT
        'SRID=4326;POINT(-72.1235 42.3521)::geography as gg1,
        'SRID=4326;LINESTRING(-72.1260 42.45, -72.123 42.1546)::geography as gg2
    ) As foo ;

spheroid_dist | sphere_dist
-----+-----
123.802076746848 | 123.475736916397

```

## Veja também

[ST\\_3DDistance](#), [ST\\_DWithin](#), [ST\\_DistanceSphere](#), [ST\\_DistanceSpheroid](#), [ST\\_MaxDistance](#), [ST\\_HausdorffDistance](#), [ST\\_FrechetDistance](#), [ST\\_Transform](#)

### 8.9.26 ST\_MinimumClearance

`ST_MinimumClearance` — Retorna a liquidação mínima de uma geometria, uma medida de uma robustez de uma geometria.

#### Synopsis

```
float ST_MinimumClearance(geometry g);
```

## Descrição

Não é incomum ter uma geometria que, enquanto o critério de validade de acordo com `ST_IsValid` (polígonos) ou `ST_IsSimple` (linhas), se tornaria inválida se um de seus vértices de movesse por uma pequena distância, como pode acontecer durante uma conversão para formatos baseados em textos (como WKT, KML, GML GeoJSON), ou formatos binários que não usam coordenadas de pontos flutuantes de precisão dupla (MapInfo TAB).

A "liquidação mínima" de uma geometria é a menor distância pela qual um vértice da geometria poderia ser movido para produzir uma geometria inválida. Pode ser pensado como uma medida quantitativa da robustez de uma geometria, onde quanto maiores os valores da liquidação maiores os valores da robustez.

Se uma geometria tem uma liquidação mínima de  $\epsilon$ , pode ser dito que:

- Dois vértices distintos na geometria não são separados por menos que  $\epsilon$ .
- Nenhum vértice está mais perto que  $\epsilon$  a um segmento de linha do qual ele não é o endpoint.

Se nenhuma liquidação existe para uma geometria (por exemplo, um único ponto, ou um multiponto cujos pontos são idênticos), então a retornará infinita.

Disponibilidade: 2.3.0 - requer GEOS  $\geq$  3.6.0

## Exemplos

```
SELECT ST_MinimumClearance('POLYGON ((0 0, 1 0, 1 1, 0.5 3.2e-4, 0 0))');
st_minimumclearance
-----
0.00032
```

## Veja também

[ST\\_MinimumClearanceLine](#)

## 8.9.27 ST\_MinimumClearanceLine

`ST_MinimumClearanceLine` — Retorna a `LineString` de dois pontos abrangendo a liquidação mínima de uma geometria.

### Synopsis

Geometry `ST_MinimumClearanceLine`(geometry g);

### Descrição

Retorna a `LineString` de dois pontos abrangendo a liquidação mínima de uma geometria. Se a geometria não possui uma liquidação mínima, uma `LINestring EMPTY` vai retornar.

Disponibilidade: 2.3.0 - requer GEOS  $\geq$  3.6.0

## Exemplos

```
SELECT ST_AsText(ST_MinimumClearanceLine('POLYGON ((0 0, 1 0, 1 1, 0.5 3.2e-4, 0 0))'));
st_astext
-----
LINESTRING(0.5 0.00032,0.5 0)
```

**Veja também**[ST\\_MinimumClearance](#)**8.9.28 ST\_HausdorffDistance**

`ST_HausdorffDistance` — Retorna a distância Hausdorff entre duas geometrias. Basicamente, uma medida de quão parecidas ou diferentes 2 geometrias são. As unidades estão nas medidas do sistema de referência espacial das geometrias.

**Synopsis**

```
float ST_HausdorffDistance(geometry g1, geometry g2);
float ST_HausdorffDistance(geometry g1, geometry g2, float densifyFrac);
```

**Descrição**

Implementa algoritmo para calcular a distância métrica que pode ser pensada como a "Distância Discreta Hausdorff". Esta é a distância de Hausdorff restrita para pontos discretos para as geometrias. [Wikipedia article on Hausdorff distance](#) [Martin Davis nota como o cálculo dessa distância de Hausdorff era usado para provar a exatidão da proximidade CascadePolygonUnion](#).

Quando `densifyFrac` for especificado, esta função representa uma densificação de segmento antes de calcular a distância hausdorff discreta. O parâmetro `densifyFrac` configura a fração pela qual o segmento será densificado. Cada segmento será dividido em um número de subsegmentos com o mesmo comprimento, de quem a fração do comprimento total está mais perto da fração dada.

**Note**

A implementação atual suporta somente vértices como as localizações completas. Isto poderia ser expandido para permitir densidade arbitrária de pontos a serem usados.

**Note**

Este algoritmo NÃO é equivalente ao modelo Hausdorff de distância. Entretanto, ele calcula uma aproximação que é correta para um grande subset de casos úteis. Uma parte importante deste subset são as linestrings que são aproximadamente paralelas umas às outras e aproximadamente iguais em comprimento. É um métrico útil para combinação de linhas.

Disponibilidade: 1.5.0 - requer GEOS >= 3.2.0

**Exemplos**

Para cada construção, encontre a parcela que melhor representa ela. Primeiro, solicitamos que a parcela intersecte com a geometria. `DISTINCT ON` garante que cada construção seja listada apenas uma vez, a `ORDER BY .. ST_HausdorffDistance` nos dá uma preferência de parcela que é mais parecida com a construção.

```
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);
```

```
postgis=# SELECT ST_HausdorffDistance(
           'LINESTRING (0 0, 2 0)::geometry,
           'MULTIPOINT (0 1, 1 0, 2 1)::geometry);
st_hausdorffdistance
-----
```

1

(1 row)

```

postgis=# SELECT st_hausdorffdistance('LINESTRING (130 0, 0 0, 0 150)::geometry, ' ↔
        LINESTRING (10 10, 10 150, 130 10)::geometry, 0.5);
 st_hausdorffdistance
-----
                          70
(1 row)

```

## Veja também

[ST\\_FrechetDistance](#)

### 8.9.29 ST\_FrechetDistance

`ST_FrechetDistance` — Returns the Fréchet distance between two geometries. This is a measure of similarity between curves that takes into account the location and ordering of the points along the curves. Units are in the units of the spatial reference system of the geometries.

#### Synopsis

```
float ST_FrechetDistance(geometry g1, geometry g2, float densifyFrac = -1);
```

#### Descrição

Implements algorithm for computing the Fréchet distance restricted to discrete points for both geometries, based on [Computing Discrete Fréchet Distance](#). The Fréchet distance is a measure of similarity between curves that takes into account the location and ordering of the points along the curves. Therefore it is often better than the Hausdorff distance.

When the optional `densifyFrac` is specified, this function performs a segment densification before computing the discrete Fréchet distance. The `densifyFrac` parameter sets the fraction by which to densify each segment. Each segment will be split into a number of equal-length subsegments, whose fraction of the total length is closest to the given fraction.



#### Note

A implementação atual suporta somente vértices como as localizações completas. Isto poderia ser expandido para permitir densidade arbitrária de pontos a serem usados.



#### Note

The smaller `densifyFrac` we specify, the more accurate Fréchet distance we get. But, the computation time and the memory usage increase with the square of the number of subsegments.

Availability: 2.4.0 - requires GEOS >= 3.7.0

#### Exemplos

```

postgres=# SELECT st_frechetdistance('LINESTRING (0 0, 100 0)::geometry, 'LINESTRING (0 0, ↔
        50 50, 100 0)::geometry);
 st_frechetdistance
-----
        70.7106781186548
(1 row)

```

```
SELECT st_frechetdistance('LINESTRING (0 0, 100 0)::geometry, 'LINESTRING (0 0, 50 50, 100 50)::geometry, 0.5);
st_frechetdistance
-----
                    50
(1 row)
```

**Veja também**[ST\\_HausdorffDistance](#)**8.9.30 ST\_MaxDistance**

`ST_MaxDistance` — Retorna a maior distância 2-dimensional entre duas geometrias em unidades projetadas.

**Synopsis**

```
float ST_MaxDistance(geometry g1, geometry g2);
```

**Descrição****Note**

Retorna a distância 2-dimensional máxima entre duas geometrias em unidades projetadas. Se g1 e g2 forem a mesma geometria, a função retornará a distância entre os dois vértices mais longes um do outro naquela geometria.

Disponibilidade: 1.5.0

**Exemplos**

Distância básica mais longe que o ponto está de qualquer parte da linha

```
postgres=# SELECT ST_MaxDistance('POINT(0 0)::geometry, 'LINESTRING ( 2 0, 0 2 ) '::geometry);
st_maxdistance
-----
                2
(1 row)
```

```
postgres=# SELECT ST_MaxDistance('POINT(0 0)::geometry, 'LINESTRING ( 2 2, 2 2 ) '::geometry);
st_maxdistance
-----
2.82842712474619
(1 row)
```

**Veja também**

[ST\\_Distance](#), [ST\\_LongestLine](#), [ST\\_DFullyWithin](#)

### 8.9.31 ST\_DistanceSphere

`ST_DistanceSphere` — Retorna a distância mínima em metros entre duas geometrias long/lat. Usa uma terra esférica e raio derivado do esferoide definido pelo SRID. Mais rápido que a `ST_DistanceSpheroid` [ST\\_DistanceSpheroid](#), mas menos preciso. As versões do PostGIS anteriores a 1.5 só implementavam para pontos.

#### Synopsis

```
float ST_DistanceSphere(geometry geomlonlatA, geometry geomlonlatB);
```

#### Descrição

Retorna a distância mínima em metros entre dois pontos long/lat. Usa uma terra esférica e raio derivado do esferoide definido pelo SRID. Mais rápido que [ST\\_DistanceSpheroid](#), mas menos preciso. As versões do PostGIS anteriores a 1.5 só implementavam para pontos.

Disponibilidade: 1.5 - suporte para outros tipos de geometria além de pontos foi introduzido. As versões anteriores só funcionam com pontos.

Alterações: 2.2.0 Em versões anteriores era chamada de `ST_Distance_Sphere`

#### Exemplos

```
SELECT round(CAST(ST_DistanceSphere(ST_Centroid(the_geom), ST_GeomFromText('POINT(-118 38) ←
',4326)) As numeric),2) As dist_meters,
round(CAST(ST_Distance(ST_Transform(ST_Centroid(the_geom),32611),
ST_Transform(ST_GeomFromText('POINT(-118 38)',4326),32611)) As numeric),2) ←
As dist_utm11_meters,
round(CAST(ST_Distance(ST_Centroid(the_geom), ST_GeomFromText('POINT(-118 38)',4326)) As ←
numeric),5) As dist_degrees,
round(CAST(ST_Distance(ST_Transform(the_geom,32611),
ST_Transform(ST_GeomFromText('POINT(-118 38)',4326),32611)) As numeric),2) ←
As min_dist_line_point_meters
FROM
(SELECT ST_GeomFromText('LINESTRING(-118.584 38.374,-118.583 38.5)',4326) As ←
the_geom) as foo;
dist_meters | dist_utm11_meters | dist_degrees | min_dist_line_point_meters
-----+-----+-----+-----
70424.47 | 70438.00 | 0.72900 | 65871.18
```

#### Veja também

[ST\\_Distance](#), [ST\\_DistanceSpheroid](#)

### 8.9.32 ST\_DistanceSpheroid

`ST_DistanceSpheroid` — Retorna a menor distância entre duas geometrias lon/lat dado um esferoide específico. As versões anteriores a 1.5 só suportam pontos.

#### Synopsis

```
float ST_DistanceSpheroid(geometry geomlonlatA, geometry geomlonlatB, spheroid measurement_spheroid);
```

## Descrição

Retorna a distância mínima em metros entre duas geometrias long/lat dado um esferoide específico. Veja a explanação dos esferoides dados pela [ST\\_LengthSpheroid](#). As versões do PostGIS anteriores a 1.5 só suportam pontos.



### Note

Esta função não olha o SRID de uma geometria e sempre irá assumir que está representada nas coordenadas do esferoide passado. AS versões anteriores desta função só suportam pontos.

Disponibilidade: 1.5 - suporte para outros tipos de geometria além de pontos foi introduzido. As versões anteriores só funcionam com pontos.

Alterações: 2.2.0 Em versões anteriores era chamada de `ST_Distance_Spheroid`

## Exemplos

```
SELECT round(CAST (
    ST_DistanceSpheroid(ST_Centroid(the_geom), ST_GeomFromText('POINT(-118 38) ←
    ', 4326), 'SPHEROID["WGS 84",6378137,298.257223563]')
    As numeric),2) As dist_meters_spheroid,
    round(CAST(ST_DistanceSphere(ST_Centroid(the_geom), ST_GeomFromText('POINT ←
    (-118 38)', 4326)) As numeric),2) As dist_meters_sphere,
round(CAST(ST_Distance(ST_Transform(ST_Centroid(the_geom), 32611),
    ST_Transform(ST_GeomFromText('POINT(-118 38)', 4326), 32611)) As numeric),2) ←
    As dist_utm11_meters
FROM
    (SELECT ST_GeomFromText('LINESTRING(-118.584 38.374,-118.583 38.5)', 4326) As ←
    the_geom) as foo;
dist_meters_spheroid | dist_meters_sphere | dist_utm11_meters
-----+-----+-----
70454.92 | 70424.47 | 70438.00
```

## Veja também

[ST\\_Distance](#), [ST\\_DistanceSphere](#)

### 8.9.33 ST\_DFullyWithin

`ST_DFullyWithin` — Retorna verdade se todas as geometrias estiverem dentro da distância especificada de um outro.

#### Synopsis

boolean `ST_DFullyWithin`(geometry g1, geometry g2, double precision distance);

#### Descrição

Retorna verdade se todas as geometrias estiverem completamente dentro da distância especificada de um outro. A distância é especificada em unidade definidas pelo sistema de referência espacial de geometrias. Para esta função fazer sentido, as geometrias fonte devem ser da mesma projeção coordenada, tendo o mesmo SRID.



**Note**

Esta função pode, automaticamente, incluir uma comparação de caixa delimitadora que usará quaisquer índices que estejam disponíveis nas geometrias.

Disponibilidade: 1.5.0

**Exemplos**

```
postgis=# 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
      (select ST_GeomFromText('POINT(1 1)') as geom_a, ST_GeomFromText('LINESTRING ←
      (1 5, 2 7, 1 9, 14 12)') as geom_b) t1;
```

```
-----
 DFullyWithin10 | DWithin10 | DFullyWithin20 |
-----+-----+-----+
 f              | t        | t              |
```

**Veja também**

[ST\\_MaxDistance](#), [ST\\_DWithin](#)

**8.9.34 ST\_DWithin**

**ST\_DWithin** — Retorna verdade se as geometrias estiverem dentro da distância especificada de outra. Para geometria, as unidades estão na referência espacial e para geografia, elas estão em metros e a medição é `use_spheroid=true` (medida em volta do esferoide), para uma verificação mais rápida, `use_spheroid=false` para medir ao longo da esfera.

**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 ST_DWithin(geography gg1, geography gg2, double precision distance_meters, boolean use_spheroid);
```

**Descrição**

Retorna verdade se as geometrias estiverem dentro da distância especificada de outra.

Para geometrias: A distância é especificada em unidades definidas pelo sistema de referência espacial das geometrias. Para esta função fazer sentido, as geometrias fonte devem ser ambas da mesma projeção de coordenada, tendo o mesmo SRID.

Para geografia, as unidades estão em metros e a medida usa `use_spheroid=verdade`, para uma verificação mais rápida, `use_spheroid=falso` para medir ao longo da esfera.

**Note**

Esta função pode, automaticamente, incluir uma comparação de caixa delimitadora que usará quaisquer índices que estejam disponíveis nas geometrias.

**Note**

Versões anteriores a 1.3, a `ST_Expand` era comumente usada em conjunto com `&&` e a `ST_Distance` para alcançar o mesmo efeito e nas versões anteriores a pre-1.3.4, esta função era basicamente uma abreviação para aquela construção. A partir da 1.3.4, a `ST_DWithin` usa uma função de distância de curto-circuito que deveria ser mais eficiente do que as versões anteriores para regiões buffer maiores.

**Note**

Use `ST_3DDWithin` se você tem geometrias 3D.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).

Disponibilidade: 1.5.0 suporte para geografia foi introduzido

Melhorias: 2.1.0 velocidade melhorada para geografia. Veja [Making Geography faster](#) para mais detalhes.

Melhorias: 2.1.0 suporte para geometrias curvas foi introduzido.

**Exemplos**

```
--Find the nearest hospital to each school
--that is within 3000 units of the school.
-- We do an ST_DWithin search to utilize indexes to limit our search list
-- that the non-indexable ST_Distance needs to process
--If the units of the spatial reference is meters then units would be meters
SELECT DISTINCT ON (s.gid) s.gid, s.school_name, s.the_geom, h.hospital_name
   FROM schools s
      LEFT JOIN hospitals h ON ST_DWithin(s.the_geom, h.the_geom, 3000)
   ORDER BY s.gid, ST_Distance(s.the_geom, h.the_geom);

--The schools with no close hospitals
--Find all schools with no hospital within 3000 units
--away from the school. Units is in units of spatial ref (e.g. meters, feet, degrees)
SELECT s.gid, s.school_name
   FROM schools s
      LEFT JOIN hospitals h ON ST_DWithin(s.the_geom, h.the_geom, 3000)
  WHERE h.gid IS NULL;
```

**Veja também**

[ST\\_Distance](#), [ST\\_CoveredBy](#), [ST\\_Within](#)

**8.9.35 ST\_Equals**

`ST_Equals` — Retorna verdade se as geometrias representam a mesma geometria. A direcionalidade é ignorada.

**Synopsis**

boolean `ST_Equals`(geometry A, geometry B);

## Descrição

Retorna VERDADE se as geometrias dadas forem "espacialmente iguais". Use isto para uma resposta "melhor" do que "=". Note que com espacialmente iguais queremos dizer que `ST_Within(A,B) = verdade` e `ST_Within(B,A) = verdade` e também que a ordenação de pontos pode ser diferente, mas representa a mesma estrutura de geometria. Para verificar se a ordenação de pontos é consistente, use `ST_OrderingEquals` (deve-se notar que `ST_OrderingEquals` é um pouco mais limitado que simplesmente verificar se as ordenações são as mesmas).



### Important

This function will return false if either geometry is invalid except in the case where they are binary equal.



### Important

Não chama com uma GEOMETRYCOLLECTION como um argumento.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2



This method implements the SQL/MM specification. SQL-MM 3: 5.1.24

Changed: 2.2.0 Returns true even for invalid geometries if they are binary equal

## Exemplos

```
SELECT ST_Equals(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
                ST_GeomFromText('LINESTRING(0 0, 5 5, 10 10)'));

 st_equals
-----
t
(1 row)

SELECT ST_Equals(ST_Reverse(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
                          ST_GeomFromText('LINESTRING(0 0, 5 5, 10 10)'));

 st_equals
-----
t
(1 row)
```

## Veja também

[ST\\_IsValid](#), [ST\\_OrderingEquals](#), [ST\\_Reverse](#), [ST\\_Within](#)

### 8.9.36 ST\_GeometricMedian

`ST_GeometricMedian` — Retorna a mediana de um MultiPonto.

#### Synopsis

geometry **ST\_GeometricMedian** ( geometry g , float8 tolerance , int max\_iter , boolean fail\_if\_not\_converged );

## Descrição

Computes the approximate geometric median of a MultiPoint geometry using the Weiszfeld algorithm. The geometric median provides a centrality measure that is less sensitive to outlier points than the centroid.

The algorithm will iterate until the distance change between successive iterations is less than the supplied `tolerance` parameter. If this condition has not been met after `max_iterations` iterations, the function will produce an error and exit, unless `fail_if_not_converged` is set to false.

If a `tolerance` value is not provided, a default tolerance value will be calculated based on the extent of the input geometry.

M value of points, if present, is interpreted as their relative weight.

Disponibilidade: 2.3.0

Enhanced: 2.5.0 Added support for M as weight of points.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

## Exemplos



*Comparação do centroide (ponto turquesa) e mediana (ponto vermelho) de um MultiPonto de quatro pontos (pontos amarelos).*

```
WITH test AS (
SELECT 'MULTIPOINT((0 0), (1 1), (2 2), (200 200))'::geometry geom)
SELECT
  ST_AsText(ST_Centroid(geom)) centroid,
  ST_AsText(ST_GeometricMedian(geom)) median
FROM test;
   centroid          |          median
-----+-----
POINT(50.75 50.75) | POINT(1.9761550281255 1.9761550281255)
(1 row)
```

## Veja também

[ST\\_Centroid](#)

### 8.9.37 ST\_HasArc

ST\_HasArc — Retorna verdade se uma geometria ou coleção de geometria contém uma string circular

#### Synopsis

boolean **ST\_HasArc**(geometry geomA);

#### Descrição

Retorna verdade se uma geometria ou coleção de geometria contém uma string circular

Disponibilidade: 1.2.3?



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

#### Exemplos

```
SELECT ST_HasArc(ST_Collect('LINESTRING(1 2, 3 4, 5 6)', 'CIRCULARSTRING(1 1, 2 3, 4 5, 6 6, 7, 5 6)'));
           st_hasarc
           -
           t
```

#### Veja também

[ST\\_CurveToLine](#), [ST\\_LineToCurve](#)

### 8.9.38 ST\_Intersects

ST\_Intersects — Retorna VERDADE se as geometrias/geografia 'intersectam espacialmente em 2D" - (dividem qualquer porção de espaço) e FALSO se elas não (estão disjuntas). Para geografia -- a tolerância é de 0.00001 metros (então quaisquer pontos que estão mais perto estão intersectando)

#### Synopsis

boolean **ST\_Intersects**( geometry geomA , geometry geomB );  
boolean **ST\_Intersects**( geography geogA , geography geogB );

#### Descrição

Se uma geometria ou geografia divide qualquer porção de espaço, elas se intersectam. Para geografia -- a tolerância é 0.00001 metros (então, quaisquer pontos mais perto são considerados como interseção)

Sobreposição, toca, dentro de todas as interseções espaciais implicadas. Se qualquer dos anteriores retornarem verdade, então as geometrias também intersectam espacialmente. Desconjuntar implica falso para interseção espacial.

Enhanced: 2.5.0 Supports GEOMETRYCOLLECTION.

Melhorias: 2.3.0 Aprimoramento para o pequeno circuito estendido PIP para suportar MultiPontos com pouco pontos. Versões anteriores só suportavam pontos em polígonos.

Desempenhado pelo módulo GEOS (para geometria), geografia é natural

Disponibilidade: 1.5 suporte para geografia foi introduzido.

**Note**

Esta função pode, automaticamente, incluir uma comparação de caixa delimitadora que usará quaisquer índices que estejam disponíveis nas geometrias.

**Note**

Para geografia, esta função tem uma tolerância de distância de 0.00001 metros e usa o cálculo da esfera em vez do esferoide.

**Note**

NOTA: esta é a versão "permissível" que retorna um booleano, não um inteiro.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 //s2.1.13.3 - ST\_Intersects(g1, g2) --> Not (ST\_Disjoint(g1, g2))



This method implements the SQL/MM specification. SQL-MM 3: 5.1.27



This method is also provided by SFCGAL backend.

**Exemplos de Geometria**

```
SELECT ST_Intersects('POINT(0 0)::geometry, 'LINESTRING ( 2 0, 0 2 ) '::geometry);
st_intersects
-----
f
(1 row)
SELECT ST_Intersects('POINT(0 0)::geometry, 'LINESTRING ( 0 0, 0 2 ) '::geometry);
st_intersects
-----
t
(1 row)
```

**Exemplos de Geografia**

```
SELECT ST_Intersects (
    ST_GeographyFromText ('SRID=4326;LINESTRING(-43.23456 72.4567,-43.23456 72.4568) '),
    ST_GeographyFromText ('SRID=4326;POINT(-43.23456 72.456772) ')
);
st_intersects
-----
t
```

**Veja também**

[ST\\_3DIntersects](#), [ST\\_Disjoint](#)

### 8.9.39 ST\_Length

**ST\_Length** — Retorna o comprimento 2D da geometria se ela é uma `LineString` ou `MultiLineString`. A geometria está em unidades da referência espacial e geografia em metros (padrão esferoide)

#### Synopsis

```
float ST_Length(geometry a_2dlinestring);
float ST_Length(geography geog, boolean use_spheroid=true);
```

#### Descrição

Para geometria: Retorna o comprimento cartesiano 2D se for uma `LineString`, `MultiLineString`, `ST_Curve`, `ST_MultiCurve`. Retorna 0 para geometrias areais. Use **ST\_Perimeter**. Para tipos de geometrias, unidades para medição de comprimento estão especificadas pelo sistema de referência espacial da geometria.

Para tipos de geografia, os cálculos são representados usando o problema geodésico inverso, onde as unidades do comprimento estão em metros. Se o PostGIS estiver compilado com a versão 4.8.0 ou superior do PROJ, o esferoide é especificado pelo SRID, senão é exclusivo do WGS84. Se `use_spheroid=false`, os cálculos irão aproximar uma esfera em vez de um esferoide.

No momento, para geometria, isto é heterônimo para `ST_Length2D`, mas isto pode mudar para dimensões maiores.



#### Warning

Alterações: 2.0.0 Quebrando a mudança -- nas versões anteriores aplicar isto a um MULTI/POLÍGONO de tipo de geografia lhe daria o perímetro do POLÍGONO/MULTIPOLÍGONO. Na 2.0.0 isso é alterado para retornar 0 a estar na linha com o comportamento da geometria. Por favor, utilize a `ST_Perimeter` se quiser o perímetro de um polígono



#### Note

Para a medição de geografia o padrão é a medição do esferoide. Para usar a esfera mais rápida e menos precisa, use `ST_Length(gg,false)`;



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.5.1



This method implements the SQL/MM specification. SQL-MM 3: 7.1.2, 9.3.4

Disponibilidade: 1.5.0 suporte para geografia foi introduzido em 1.5.



This method is also provided by SFCGAL backend.

#### Exemplos de Geometria

Retorna o comprimento em pés para line string. Note que é em pés, porque EPSG:2249 é Massachusetts State Plane Feet

```
SELECT ST_Length(ST_GeomFromText('LINESTRING(743238 2967416,743238 2967450,743265 2967450,743265.625 2967416,743238 2967416)',2249));
```

```
st_length
```

```
-----
```

```
122.630744000095
```

```
--Transforming WGS 84 LineString to Massachusetts state plane meters
```

```
SELECT ST_Length(
```

```

        ST_Transform(
            ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45, -72.1240 42.45666, ↵
                -72.123 42.1546)'),
            26986
        )
);
st_length
-----
34309.4563576191

```

## Exemplos de Geografia

Retorna o comprimento de WGS 84 linha de geografia

```

-- default calculation is using a sphere rather than spheroid
SELECT ST_Length(the_geog) As length_spheroid, ST_Length(the_geog,false) As length_sphere
FROM (SELECT ST_GeographyFromText(
'SRID=4326;LINESTRING(-72.1260 42.45, -72.1240 42.45666, -72.123 42.1546)') As the_geog)
As foo;
length_spheroid | length_sphere
-----+-----
34310.5703627288 | 34346.2060960742

```

## Veja também

[ST\\_GeographyFromText](#), [ST\\_GeomFromEWKT](#), [ST\\_LengthSpheroid](#), [ST\\_Perimeter](#), [ST\\_Transform](#)

### 8.9.40 ST\_Length2D

**ST\_Length2D** — Retorna o comprimento 2-dimensional da geometria se for uma linestring ou multi-linestring. Isto é um heterônimo para `ST_Length`

#### Synopsis

```
float ST_Length2D(geometry a_2dlinestring);
```

#### Descrição

Retorna o comprimento 2-dimensional da geometria se for uma linestring ou multi-linestring. Isto é um heterônimo para `ST_Length`

## Veja também

[ST\\_Length](#), [ST\\_3DLength](#)

### 8.9.41 ST\_3DLength

**ST\_3DLength** — Retorna o comprimento 3-dimensional ou 2-dimensional da geometria se for uma linestring ou multi-linestring.

#### Synopsis

```
float ST_3DLength(geometry a_3dlinestring);
```



## Descrição

Retorna o comprimento 3-dimensional ou 2-dimensional da geometria se for uma linestring ou multi-linestring. Para linhas 2-d, ela só retornará o comprimento 2-d (o mesmo da `ST_Length` e `ST_Length2D`)



This function supports 3d and will not drop the z-index.

Alterações: 2.0.0 Nas versões anteriores era chamado de `ST_Length3D`

## Exemplos

Retorna o comprimento em pés para um cabo 3D. Note que é em pés, porque EPSG:2249 é Massachusetts State Plane Feet

```
SELECT ST_3DLength(ST_GeomFromText('LINESTRING(743238 2967416 1,743238 2967450 1,743265 2967450 3,743265.625 2967416 3,743238 2967416 3)',2249));
ST_3DLength
-----
122.704716741457
```

## Veja também

[ST\\_Length](#), [ST\\_Length2D](#)

## 8.9.42 ST\_LengthSpheroid

`ST_LengthSpheroid` — Calcula o comprimento/perímetro 2D ou 3D de uma geometria em um elipsoide. É útil se as coordenadas da geometria estão em longitude/latitude e um comprimento é desejado sem reprojeção.

### Synopsis

```
float ST_LengthSpheroid(geometry a_geometry, spheroid a_spheroid);
```

### Descrição

Calcula o comprimento/perímetro de uma geometria em um elipsoide. É útil se as coordenadas da geometria estão em longitude/latitude e um comprimento é desejado sem reprojeção. O elipsoide é um tipo de banco de dados separado e pode ser construído como segue:

```
SPHEROID [<NAME>, <SEMI-MAJOR AXIS>, <INVERSE FLATTENING>]
```

```
SPHEROID ["GRS_1980", 6378137, 298.257222101]
```

Disponibilidade: 1.2.2

Alterações: 2.2.0 Em versões anteriores era chamada de `ST_Length_Spheroid` e costumava ter um heterônimo `ST_3DLength_Spheroid`



This function supports 3d and will not drop the z-index.

## Exemplos

```

SELECT ST_LengthSpheroid( geometry_column,
                          'SPHEROID["GRS_1980",6378137,298.257222101]' )
FROM geometry_table;

SELECT ST_LengthSpheroid( the_geom, sph_m ) As tot_len,
ST_LengthSpheroid(ST_GeometryN(the_geom,1), sph_m) As len_line1,
ST_LengthSpheroid(ST_GeometryN(the_geom,2), sph_m) As len_line2
FROM (SELECT ST_GeomFromText('MULTILINESTRING((-118.584 38.374,
-118.583 38.5),
(-71.05957 42.3589 , -71.061 43))') As the_geom,
CAST('SPHEROID["GRS_1980",6378137,298.257222101]' As spheroid) As sph_m) as foo;
-----+-----+-----
tot_len      | len_line1   | len_line2
-----+-----+-----
85204.5207562955 | 13986.8725229309 | 71217.6482333646

--3D
SELECT ST_LengthSpheroid( the_geom, sph_m ) As tot_len,
ST_LengthSpheroid(ST_GeometryN(the_geom,1), sph_m) As len_line1,
ST_LengthSpheroid(ST_GeometryN(the_geom,2), sph_m) As len_line2
FROM (SELECT ST_GeomFromEWKT('MULTILINESTRING((-118.584 38.374 20,
-118.583 38.5 30),
(-71.05957 42.3589 75, -71.061 43 90))') As the_geom,
CAST('SPHEROID["GRS_1980",6378137,298.257222101]' As spheroid) As sph_m) as foo;
-----+-----+-----
tot_len      | len_line1   | len_line2
-----+-----+-----
85204.5259107402 | 13986.876097711 | 71217.6498130292

```

## Veja também

[ST\\_GeometryN](#), [ST\\_Length](#)

### 8.9.43 ST\_Length2D\_Spheroid

**ST\_Length2D\_Spheroid** — Calcula o comprimento/perímetro 2D de uma geometria em um elipsoide. É útil se as coordenadas da geometria estão em longitude/latitude e um comprimento é desejado sem reprojeção.

#### Synopsis

```
float ST_Length2D_Spheroid(geometry a_geometry, spheroid a_spheroid);
```

#### Descrição

Calcula o comprimento/perímetro 2D de uma geometria em um elipsoide. É útil se as coordenadas da geometria estão em longitude/latitude e um comprimento é desejado sem reprojeção. O elipsoide é um tipo de banco de dados separado e pode ser construído como segue:

```
SPHEROID [<NAME>, <SEMI-MAJOR AXIS>, <INVERSE FLATTENING>]
```

```
SPHEROID ["GRS_1980", 6378137, 298.257222101]
```

**Note**

Isto se parece bastante com `ST_LengthSpheroid` exceto que irá ignorar a ordenada Z nos cálculos.

**Exemplos**

```
SELECT ST_Length2D_Spheroid( geometry_column,
                             'SPHEROID["GRS_1980",6378137,298.257222101]' )
FROM geometry_table;

SELECT ST_Length2D_Spheroid( the_geom, sph_m ) As tot_len,
ST_Length2D_Spheroid(ST_GeometryN(the_geom,1), sph_m) As len_line1,
ST_Length2D_Spheroid(ST_GeometryN(the_geom,2), sph_m) As len_line2
FROM (SELECT ST_GeomFromText('MULTILINESTRING((-118.584 38.374,
-118.583 38.5),
(-71.05957 42.3589 , -71.061 43))') As the_geom,
CAST('SPHEROID["GRS_1980",6378137,298.257222101]' As spheroid) As sph_m) as foo;
-----+-----+-----
tot_len      | len_line1   | len_line2
-----+-----+-----
85204.5207562955 | 13986.8725229309 | 71217.6482333646

--3D Observe same answer
SELECT ST_Length2D_Spheroid( the_geom, sph_m ) As tot_len,
ST_Length2D_Spheroid(ST_GeometryN(the_geom,1), sph_m) As len_line1,
ST_Length2D_Spheroid(ST_GeometryN(the_geom,2), sph_m) As len_line2
FROM (SELECT ST_GeomFromEWKT('MULTILINESTRING((-118.584 38.374 30,
20,-118.583 38.5 30),
(-71.05957 42.3589 75, -71.061 43 90))') As the_geom,
CAST('SPHEROID["GRS_1980",6378137,298.257222101]' As spheroid) As sph_m) as foo;

tot_len      | len_line1   | len_line2
-----+-----+-----
85204.5207562955 | 13986.8725229309 | 71217.6482333646
```

**Veja também**

[ST\\_GeometryN](#), [ST\\_LengthSpheroid](#)

**8.9.44 ST\_LongestLine**

`ST_LongestLine` — Retorna a linha de pontos 2-dimensional mais longa entre duas geometrias. A função só retornará a primeira linha, se existirem mais de uma que a função encontra. A linha retornada sempre começará em g1 e acabará em g2. O comprimento da linha essa função sempre será o mesmo da `st_maxdistance` retorna para g1 e g2.

**Synopsis**

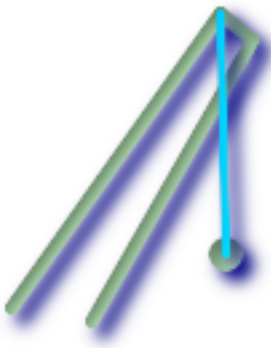
```
geometry ST_LongestLine(geometry g1, geometry g2);
```

**Descrição**

Retorna a linha 2-dimensional mais longa entre os pontos de duas geometrias.

Disponibilidade: 1.5.0

**Exemplos**

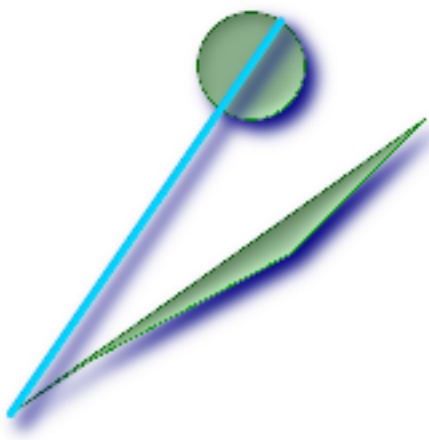


*Linha mais longa entre ponto e linha*

```

SELECT ST_AsText (
  ST_LongestLine('POINT(100 100) ':: geometry,
    'LINESTRING (20 80, 98 190, 110 180, 50 75 ) '::geometry)
) As lline;

lline
-----
LINESTRING(100 100,98 190)
    
```

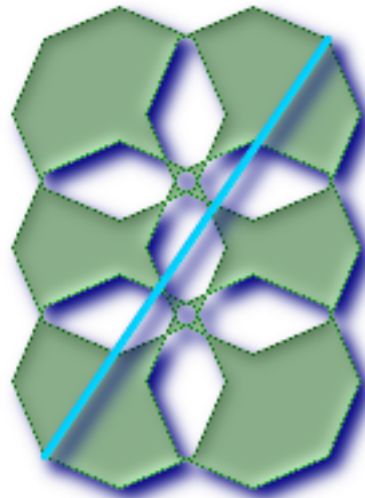


*linha mais longa entre polígono e polígono*

```

SELECT ST_AsText (
  ST_LongestLine (
    ST_GeomFromText ('POLYGON ↵
((175 150, 20 40, 50 60, 125 100, ↵
175 150))'),
    ST_Buffer(ST_GeomFromText ↵
('POINT(110 170)'), 20)
)
) As llinewkt;

lline
-----
LINESTRING(20 40,121.111404660392 ↵
186.629392246051)
    
```



*distância direta mais longa para se viajar de uma cidade elegante para outra. Note que a distância máxima = ao comprimento da linha.*

```
SELECT ST_AsText(ST_LongestLine(c.the_geom, c.the_geom)) As llinekwt,
       ST_MaxDistance(c.the_geom,c.the_geom) As max_dist,
       ST_Length(ST_LongestLine(c.the_geom, c.the_geom)) As lenll
FROM (SELECT ST_BuildArea(ST_Collect(the_geom)) As the_geom
      FROM (SELECT ST_Translate(ST_SnapToGrid(ST_Buffer(ST_Point(50 ,generate_series ←
(50,190, 50)
                                     ),40, 'quad_segs=2'),1), x, 0) As the_geom
      FROM generate_series(1,100,50) As x) AS foo
) As c;
```

llinekwt	max_dist	lenll
LINESTRING(23 22,129 178)	188.605408193933	188.605408193933

### Veja também

[ST\\_MaxDistance](#), [ST\\_ShortestLine](#), [ST\\_LongestLine](#)

### 8.9.45 ST\_OrderingEquals

`ST_OrderingEquals` — Retorna verdade se as geometrias dadas representam a mesma geometrias e os pontos estão na mesma ordem direcional.

#### Synopsis

boolean `ST_OrderingEquals`(geometry A, geometry B);

#### Descrição

`ST_OrderingEquals` compara duas geometrias e retorna t (VERDADE) se as geometrias forem iguais e as coordenadas estiverem na mesma ordem; senão, retorna f (FALSO).

**Note**

Esta função é implementada como a especificação ArcSDE SQL em vez da SQL-MM. [http://edndoc.esri.com/arcsde/9.1/sql\\_api/sqlapi3.htm#ST\\_OrderingEquals](http://edndoc.esri.com/arcsde/9.1/sql_api/sqlapi3.htm#ST_OrderingEquals)



This method implements the SQL/MM specification. SQL-MM 3: 5.1.43

**Exemplos**

```
SELECT ST_OrderingEquals(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
                        ST_GeomFromText('LINESTRING(0 0, 5 5, 10 10)'));
 st_orderingequals
-----
 f
(1 row)

SELECT ST_OrderingEquals(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
                        ST_GeomFromText('LINESTRING(0 0, 0 0, 10 10)'));
 st_orderingequals
-----
 t
(1 row)

SELECT ST_OrderingEquals(ST_Reverse(ST_GeomFromText('LINESTRING(0 0, 10 10)'),
                                ST_GeomFromText('LINESTRING(0 0, 0 0, 10 10)'));
 st_orderingequals
-----
 f
(1 row)
```

**Veja também**

[ST\\_Equals](#), [ST\\_Reverse](#)

**8.9.46 ST\_Overlaps**

**ST\_Overlaps** — Retorna VERDADE se as geometrias dividem espaço, são da mesma dimensão, mas não estão completamente contidas uma pela outra.

**Synopsis**

boolean **ST\_Overlaps**(geometry A, geometry B);

**Descrição**

Retorna VERDADE se as geometrias se "sobrepoem espacialmente". Com isso, queremos dizer que elas se intersectam, mas uma não contém completamente a outra.

Desempenhado pelo módulo GEOS

**Note**

Não chama com uma GeometryCollection com um argumento

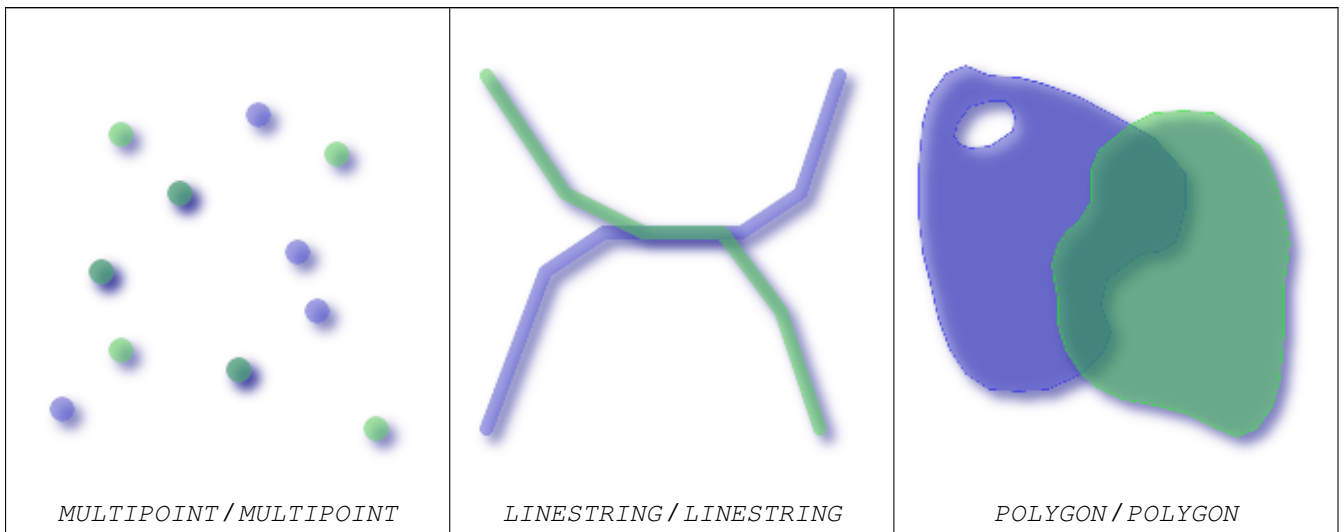
A chamada desta função irá, automaticamente, incluir uma comparação de caixa delimitadora que fará uso de quaisquer índices que estejam disponíveis nas geometrias. Para evitar o uso de índice, utilize a função `_ST_Overlaps`.

NOTA: esta é a versão "permissível" que retorna um booleano, não um inteiro.

- ✓ This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1. s2.1.1.2 // s2.1.13.3](#)
- ✓ This method implements the SQL/MM specification. SQL-MM 3: 5.1.32

**Exemplos**

Todas as ilustrações seguintes retornam TRUE.



```
--a point on a line is contained by the line and is of a lower dimension, and therefore ↩
  does not overlap the line
      nor crosses

SELECT ST_Overlaps(a,b) As a_overlap_b,
       ST_Crosses(a,b) As a_crosses_b,
       ST_Intersects(a, b) As a_intersects_b, ST_Contains(b,a) As b_contains_a
FROM (SELECT ST_GeomFromText('POINT(1 0.5)') As a, ST_GeomFromText('LINESTRING(1 0, 1 1, 3 ↩
  5)') As b)
     As foo

a_overlap_b | a_crosses_b | a_intersects_b | b_contains_a
-----+-----+-----+-----
f           | f           | t              | t

--a line that is partly contained by circle, but not fully is defined as intersecting and ↩
  crossing,
-- but since of different dimension it does not overlap
SELECT ST_Overlaps(a,b) As a_overlap_b, ST_Crosses(a,b) As a_crosses_b,
       ST_Intersects(a, b) As a_intersects_b,
       ST_Contains(a,b) As a_contains_b
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 0.5)'), 3) As a, ST_GeomFromText(' ↩
  LINESTRING(1 0, 1 1, 3 5)') As b)
     As foo;

a_overlap_b | a_crosses_b | a_intersects_b | a_contains_b
-----+-----+-----+-----
```

```

f          | t          | t          | f
-- a 2-dimensional bent hot dog (aka buffered line string) that intersects a circle,
--   but is not fully contained by the circle is defined as overlapping since they ←
--   are of the same dimension,
--   but it does not cross, because the intersection of the 2 is of the same dimension
--   as the maximum dimension of the 2

SELECT ST_Overlaps(a,b) As a_overlap_b, ST_Crosses(a,b) As a_crosses_b, ST_Intersects(a, b) ←
       As a_intersects_b,
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 ←
       dima_intersection_b
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 0.5)'), 3) As a,
       ST_Buffer(ST_GeomFromText('LINESTRING(1 0, 1 1, 3 5)'),0.5) As b)
       As foo;

a_overlap_b | a_crosses_b | a_intersects_b | b_contains_a | dim_a | dim_b | ←
-----+-----+-----+-----+-----+-----+-----
t          | f          | t          | f          | 2    | 2    | ←

```

### Veja também

[ST\\_Contains](#), [ST\\_Crosses](#), [ST\\_Dimension](#), [ST\\_Intersects](#)

### 8.9.47 ST\_Perimeter

**ST\_Perimeter** — Retorna o comprimento do limite de uma geometria ou geografia `ST_Surface` ou `ST_MultiSurface`. (Polígono, Multipolígono). A medição das unidades de geometria está na referência espacial e a da geografia em metros.

#### Synopsis

```
float ST_Perimeter(geometry g1);
float ST_Perimeter(geography geog, boolean use_spheroid=true);
```

#### Descrição

Retorna o perímetro 2D da geometria/geografia se for uma `ST_Surface`, `ST_MultiSurface` (Polygon, MultiPolygon). Retorna 0 para geometrias não areais. Para geometrias lineares, use [ST\\_Length](#). Para tipos de geometria, unidades para medição de perímetro estão especificadas pelo sistema de referência espacial da geometria.

Para tipos de geografia, os cálculos são representados usando o problema geodésico inverso, onde as unidades do perímetro estão em metros. Se o PostGIS estiver compilado com a versão 4.8.0 ou superior do PROJ, o esferoide é especificado pelo SRID, senão é exclusivo do WGS84. Se `use_spheroid=false`, os cálculos irão aproximar uma esfera em vez de um esferoide.

No momento isto é um heterônimo para `ST_Perimeter2D`, mas pode ser alterado para suportar dimensões maiores.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.5.1



This method implements the SQL/MM specification. SQL-MM 3: 8.1.3, 9.5.4

Disponibilidade 2.0.0: Suporte para geografia foi introduzido



**Exemplos: Geometria**

Retorna o perímetro em pés para Polígono e Multipolígono. Note que é em pés, porque EPSG:2249 é Massachusetts State Plane Feet

```
SELECT ST_Perimeter(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450,743265 2967450,
743265.625 2967416,743238 2967416))', 2249));
st_perimeter
-----
 122.630744000095
(1 row)

SELECT ST_Perimeter(ST_GeomFromText('MULTIPOLYGON(((763104.471273676 2949418.44119003,
763104.477769673 2949418.42538203,
763104.189609677 2949418.22343004,763104.471273676 2949418.44119003)),
((763104.471273676 2949418.44119003,763095.804579742 2949436.33850239,
763086.132105649 2949451.46730207,763078.452329651 2949462.11549407,
763075.354136904 2949466.17407812,763064.362142565 2949477.64291974,
763059.953961626 2949481.28983009,762994.637609571 2949532.04103014,
762990.568508415 2949535.06640477,762986.710889563 2949539.61421415,
763117.237897679 2949709.50493431,763235.236617789 2949617.95619822,
763287.718121842 2949562.20592617,763111.553321674 2949423.91664605,
763104.471273676 2949418.44119003)))', 2249));
st_perimeter
-----
 845.227713366825
(1 row)
```

**Exemplos: Geografia**

Retorna perímetro em metros e pés para Polígono e MultiPolígono. Note que isso é geografia (WGS 84 long lat)

```
SELECT ST_Perimeter(geog) As per_meters, ST_Perimeter(geog)/0.3048 As per_ft
FROM ST_GeogFromText('POLYGON((-71.1776848522251 42.3902896512902,-71.1776843766326 ↵
 42.3903829478009,
-71.1775844305465 42.3903826677917,-71.1775825927231 42.3902893647987,-71.1776848522251 ↵
 42.3902896512902))') As geog;

  per_meters      |      per_ft
-----+-----
37.3790462565251 | 122.634666195949

-- MultiPolygon example --
SELECT ST_Perimeter(geog) As per_meters, ST_Perimeter(geog,false) As per_sphere_meters, ↵
  ST_Perimeter(geog)/0.3048 As per_ft
FROM ST_GeogFromText('MULTIPOLYGON(((71.1044543107478 42.340674480411,-71.1044542869917 ↵
 42.3406744369506,
-71.1044553562977 42.340673886454,-71.1044543107478 42.340674480411)),
((-71.1044543107478 42.340674480411,-71.1044860600303 42.3407237015564,-71.1045215770124 ↵
 42.3407653385914,
-71.1045498002983 42.3407946553165,-71.1045611902745 42.3408058316308,-71.1046016507427 ↵
 42.340837442371,
-71.104617893173 42.3408475056957,-71.1048586153981 42.3409875993595,-71.1048736143677 ↵
 42.3409959528211,
-71.1048878050242 42.3410084812078,-71.1044020965803 42.3414730072048,
-71.1039672113619 42.3412202916693,-71.1037740497748 42.3410666421308,
-71.1044280218456 42.3406894151355,-71.1044543107478 42.340674480411)))') As geog;

  per_meters      | per_sphere_meters |      per_ft
-----+-----+-----
```

257.634283683311 | 257.412311446337 | 845.256836231335

### Veja também

[ST\\_GeogFromText](#), [ST\\_GeomFromText](#), [ST\\_Length](#)

## 8.9.48 ST\_Perimeter2D

`ST_Perimeter2D` — Retorna o perímetro 2-dimensional da geometria, se for um polígono ou multi-polígono. Isto é, no momento, um heterônimo para `ST_Perimeter`.

### Synopsis

```
float ST_Perimeter2D(geometry geomA);
```

### Descrição

Retorna o perímetro 2-dimensional da geometria, se for uma polígono ou multi-polígono.



#### Note

Isto é um heterônimo para `ST_Perimeter`. Nas próximas versões a `ST_Perimeter`. pode retornar a maior dimensão de perímetro para uma geometria. Continua abaixo de consideração

### Veja também

[ST\\_Perimeter](#)

## 8.9.49 ST\_3DPerímetro

`ST_3DPerímetro` — Retorna o perímetro 3-dimensional da geometria, se for uma polígono ou multi-polígono.

### Synopsis

```
float ST_3DPerimeter(geometry geomA);
```

### Descrição

Retorna o perímetro 3-dimensional da geometria, se for um polígono ou multi-polígono. Se a geometria for 2-dimensional, então retorna o perímetro 2-dimensional.



This function supports 3d and will not drop the z-index.

Alterações: 2.0.0 Nas versões anteriores era chamado de `ST_Perimeter3D`

## Exemplos

O perímetro de um polígono levemente elevado no ar no Massachusetts state plane feet

```
SELECT ST_3DPerimeter(the_geom), ST_Perimeter2d(the_geom), ST_Perimeter(the_geom) FROM
      (SELECT ST_GeomFromEWKT('SRID=2249;POLYGON((743238 2967416 2,743238 ←
          2967450 1,
743265.625 2967416 1,743238 2967416 2))') As the_geom) As foo;

  ST_3DPerimeter | st_perimeter2d | st_perimeter
-----+-----+-----
105.465793597674 | 105.432997272188 | 105.432997272188
```

## Veja também

[ST\\_GeomFromEWKT](#), [ST\\_Perimeter](#), [ST\\_Perimeter2D](#)

### 8.9.50 ST\_PointOnSurface

ST\_PointOnSurface — Retorna um POINT garantido a ficar na superfície.

#### Synopsis

geometry **ST\_PointOnSurface**(geometry g1);

#### Descrição

Retorna um POINT garantido a intersectar uma superfície.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s3.2.14.2 // s3.2.18.2



This method implements the SQL/MM specification. SQL-MM 3: 8.1.5, 9.5.6. De acordo com as specs, a ST\_PointOnSurface funciona para superfícies (POLÍGONOS, MULTIPOLÍGONOS, POÍGONOS CURVOS). Então o PostGIS parece estar estendendo o que a spec permite aqui. A maioria dos banco de dados Oracle,DB II, ESRI SDE parecem suportar esta função para superfícies. Assim como o PostGIS o SQL Server 2008 suporta todas as geometrias comuns.



This function supports 3d and will not drop the z-index.

## Exemplos

```
SELECT ST_AsText(ST_PointOnSurface('POINT(0 5)::geometry'));
 st_astext
-----
POINT(0 5)
(1 row)

SELECT ST_AsText(ST_PointOnSurface('LINESTRING(0 5, 0 10)::geometry'));
 st_astext
-----
POINT(0 5)
(1 row)
```

```

SELECT ST_AsText(ST_PointOnSurface('POLYGON((0 0, 0 5, 5 5, 5 0, 0 0))'::geometry));
      st_astext
-----
POINT(2.5 2.5)
(1 row)

SELECT ST_AsEWKT(ST_PointOnSurface(ST_GeomFromEWKT('LINESTRING(0 5 1, 0 0 1, 0 10 2)')));
      st_asewkt
-----
POINT(0 0 1)
(1 row)

```

**Veja também**

[ST\\_Centroid](#), [ST\\_PointInsideCircle](#)

**8.9.51 ST\_Project**

**ST\_Project** — Retorna um POINT projetado de um ponto inicial usando uma distância em metros e suportando (azimute) em radianos.

**Synopsis**

geography **ST\_Project**(geography g1, float distance, float azimuth);

**Descrição**

Retorna um POINT projetado ao longo de um geodésico de um azimute (suportando) medido em radianos e distância em metros. Também é chamado de um problema direto geodésico.

O azimute é chamado de guia ou sustentador na navegação. É medido relativo ao norte verdadeiro (azimute zero). Leste é azimute 90 ( $\pi/2$ ), sul é azimute 180 ( $\pi$ ), oeste é azimute 270 ( $3\pi/2$ ).

A distância é dada em metros.

Disponibilidade: 2.0.0

Enhanced: 2.4.0 Allow negative distance and non-normalized azimuth.

**Exemplo: Usando graus - ponto projetado 100,000 metros e assumindo 45 graus**

```

SELECT ST_AsText(ST_Project('POINT(0 0)'::geography, 100000, radians(45.0)));
      st_astext
-----
POINT(0.635231029125537 0.639472334729198)
(1 row)

```

**Veja também**

[ST\\_Azimuth](#), [ST\\_Distance](#), [PostgreSQL Math Functions](#)

## 8.9.52 ST\_Relate

**ST\_Relate** — Retorna verdade se esta geometria estiver relacionada a outra geometria, testando interseções entre o interior, limite e exterior das duas geometrias como especificado pelos valores na `intersectionMatrixPattern`. Se nenhuma `intersectionMatrixPattern` passa, retorna a `intersectionMatrixPattern` máxima que relaciona as 2 geometrias.

### 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);
```

### Descrição

Versão 1: Pega a `geomA`, `geomB`, `intersectionMatrix` e Retorna 1 (VERDADE) se esta geometria for relacionada espacialmente com outra geometria, testando interseções entre o interior, limite e exterior das duas geometrias pelos valores na [DE-9IM matrix pattern](#).

Isto é especialmente útil para testar verificações compostas de interseção, travessias etc., em um passo.

Não chama com uma `GeometryCollection` com um argumento



#### Note

Isto é a versão "permissível" que retorna um booleano, não um inteiro. É definido na spec do OGC



#### Note

Isto NÃO inclui uma chamada de índice. A razão para isso é que algumas relações são anti ex. Disjuntas. Se você estiver usando uma relação padrão que requer interseção, então inclui `&& index call`.

Versão 2: Pega `geomA` e `geomB` e retorna a [Section 4.3.6](#)

Versão 3: a mesma da versão 2, mas permite especificar uma regra de nó delimitador (1:OGC/MOD2, 2:Endpoint, 3:MultivalentEndpoint, 4:MonovalentEndpoint)



#### Note

Não chama com uma `GeometryCollection` com um argumento

não na spec OGC, mas sugerido. veja [s2.1.13.2](#)

Desempenhado pelo módulo GEOS



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). [s2.1.1.2](#) // [s2.1.13.3](#)



This method implements the SQL/MM specification. SQL-MM 3: 5.1.25

Melhorias: 2.0.0 - suporte para especificar regra de nó delimitador adicionado (requer GEOS >= 3.0).

## Exemplos

```
--Find all compounds that intersect and not touch a poly (interior intersects)
SELECT l.* , b.name As poly_name
      FROM polys As b
INNER JOIN compounds As l
ON (p.the_geom && b.the_geom
AND ST_Relate(l.the_geom, b.the_geom, 'T*****'));

SELECT ST_Relate(ST_GeometryFromText('POINT(1 2)'), ST_Buffer(ST_GeometryFromText('POINT(1 2)'),2));
st_relate
-----
0FFFFFF212

SELECT ST_Relate(ST_GeometryFromText('LINESTRING(1 2, 3 4)'), ST_GeometryFromText('LINESTRING(5 6, 7 8)'));
st_relate
-----
FF1FF0102

SELECT ST_Relate(ST_GeometryFromText('POINT(1 2)'), ST_Buffer(ST_GeometryFromText('POINT(1 2)'),2), '0FFFFFF212');
st_relate
-----
t

SELECT ST_Relate(ST_GeometryFromText('POINT(1 2)'), ST_Buffer(ST_GeometryFromText('POINT(1 2)'),2), '*FF*FF212');
st_relate
-----
t
```

## Veja também

[ST\\_Crosses](#), [Section 4.3.6](#), [ST\\_Disjoint](#), [ST\\_Intersects](#), [ST\\_Touches](#)

### 8.9.53 ST\_RelateMatch

`ST_RelateMatch` — Retorna verdade se `intersectionMatrixPattern1` insinuar `intersectionMatrixPattern2`

#### Synopsis

boolean **ST\_RelateMatch**(text intersectionMatrix, text intersectionMatrixPattern);

#### Descrição

Pega a `intersectionMatrix` e `intersectionMatrixPattern` e retorna verdade se a `intersectionMatrix` satisfizer a `intersectionMatrixPattern`. Para maiores informações, recorra a [Section 4.3.6](#).

Disponibilidade: 2.0.0 - requer GEOS >= 3.3.0.

## Exemplos

```
SELECT ST_RelateMatch('101202FFF', 'TTTTTFFF') ;
-- result --
t
--example of common intersection matrix patterns and example matrices
-- comparing relationships of involving one invalid geometry and ( a line and polygon that ←
  intersect at interior and boundary)
SELECT mat.name, pat.name, ST_RelateMatch(mat.val, pat.val) As satisfied
FROM
  ( VALUES ('Equality', 'T1FF1FFF1'),
    ('Overlaps', 'T*T***T**'),
    ('Within', 'T*F**F***'),
    ('Disjoint', 'FF*FF****') As pat(name,val)
CROSS JOIN
  ( VALUES ('Self intersections (invalid)', '111111111'),
    ('IE2_BI1_BB0_BE1_EI1_EE2', 'FF2101102'),
    ('IB1_IE1_BB0_BE0_EI2_EI1_EE2', 'F11F00212')
  ) As mat(name,val);
```

## Veja também

Section [4.3.6, ST\\_Relate](#)

## 8.9.54 ST\_ShortestLine

ST\_ShortestLine — Retorna a menor linha 2-dimensional entre duas geometrias

### Synopsis

geometry **ST\_ShortestLine**(geometry g1, geometry g2);

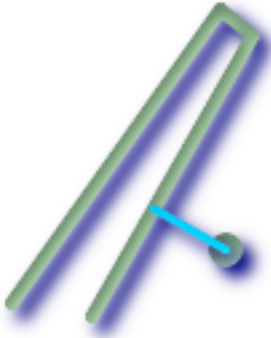
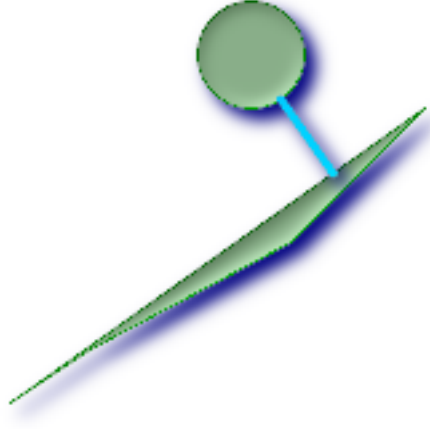
### Descrição

Retorna a menor linha 2-dimensional entre duas geometrias, a função só irá retornar a primeira linha menor se houverem mais de um, este a função encontra. Se g1 e g2 intersecta em apenas um ponto, a função retornará uma linha com os pontos de interseção da direita e esquerda. Se g1 e g2 estão intersectando em mais de um ponto, a função retornará uma linha com começo e fim no mesmo ponto, mas também pode ser qualquer um dos outros pontos. A linha que retorna sempre começará com g2 e acabará em g2. O comprimento da linha que esta função retorna será sempre o mesmo que a para **ST\_Distance** retorna para g1 e g2.

Disponibilidade: 1.5.0

## Exemplos

---

 <p style="text-align: center;"><i>Menor linha entre ponto e linestring</i></p> <pre>SELECT ST_AsText (   ST_ShortestLine('POINT(100 100) ←     '::geometry,     'LINESTRING (20 80, 98 ←       190, 110 180, 50 75 )'::geometry)   ) As sline;  sline ----- LINESTRING(100 100,73.0769230769231 ←   115.384615384615)</pre>	 <p style="text-align: center;"><i>menor linha entre polígono e polígono</i></p> <pre>SELECT ST_AsText (   ST_ShortestLine (     ST_GeomFromText (' ←       POLYGON((175 150, 20 40, 50 60, 125         ST_Buffer( ←           ST_GeomFromText ('POINT(110 170)'), 2         )       ) As slinewkt;  LINESTRING(140.752120669087 ←   125.695053378061,121.111404660392 15</pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Veja também**

[ST\\_ClosestPoint](#), [ST\\_Distance](#), [ST\\_LongestLine](#), [ST\\_MaxDistance](#)

**8.9.55 ST\_Touches**

**ST\_Touches** — Retorna TRUE se as geometrias têm pelo menos um ponto em comum, mas seus interiores não se intersectam.

**Synopsis**

boolean **ST\_Touches**(geometry g1, geometry g2);

**Descrição**

Retorna TRUE se somente os pontos em comum entr  $g1$  e  $g2$  estiverem na união dos limites de  $g1$  e  $g2$ . A **ST\_Touches** relação aplica a todos Area/Area, Line/Line, Line/Area, Point/Area and Point/Line pares de relações, mas *não* ao par Point/Point.

Em termos matemáticos, este atributo é expressado como:

$$a.Touches(b) \Leftrightarrow (I(a) \cap I(b) = \emptyset) \wedge (a \cap b) \neq \emptyset$$



As matrizes de interseção DE-9IM permissíveis para as duas geometrias são:

- FT\*\*\*\*\*
- F\*\*T\*\*\*\*\*
- F\*\*\*T\*\*\*\*



#### Important

Não chama com uma `GEOMETRYCOLLECTION` como um argumento



#### Note

A chamada desta função irá, automaticamente, incluir uma comparação de caixa delimitadora que fará uso de quaisquer índices que esteja disponíveis nas geometrias. Para evitar o uso de índice, utilize `_ST_Touches`.



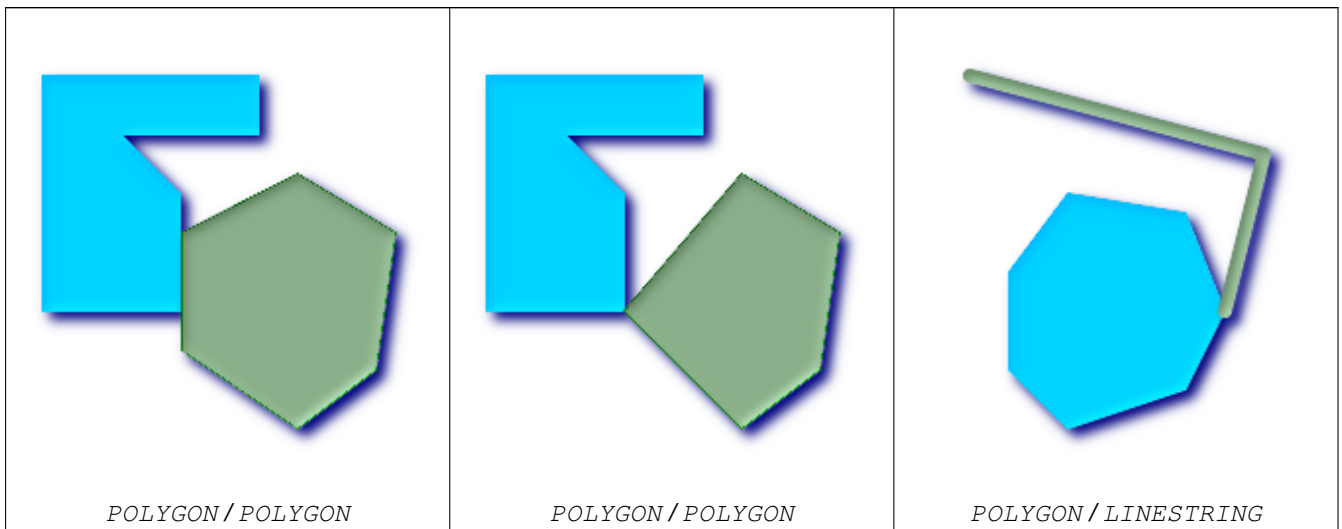
This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 // s2.1.13.3

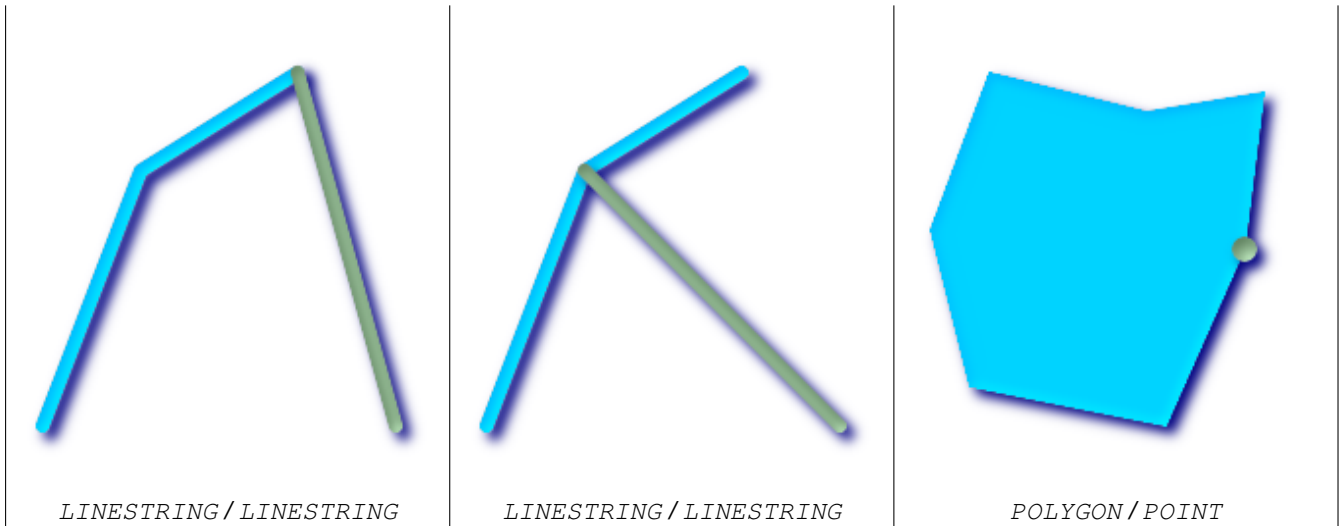


This method implements the SQL/MM specification. SQL-MM 3: 5.1.28

#### Exemplos

O predicado `_ST_Touches` retorna `TRUE` em todas as ilustrações seguintes.





```
SELECT ST_Touches('LINESTRING(0 0, 1 1, 0 2)::geometry, 'POINT(1 1)::geometry');
st_touches
-----
f
(1 row)

SELECT ST_Touches('LINESTRING(0 0, 1 1, 0 2)::geometry, 'POINT(0 2)::geometry');
st_touches
-----
t
(1 row)
```

### 8.9.56 ST\_Within

**ST\_Within** — Retorna verdade se a geometria A estiver completamente dentro da geometria B

#### Synopsis

boolean **ST\_Within**(geometry A, geometry B);

#### Descrição

Retorna VERDADE se a geometria A estiver completamente dentro da B. Para esta função funcionar, as geometrias fonte devem ser da mesma projeção de coordenada, tendo o mesmo SRID. **ST\_Contains** é o inverso de **ST\_Within**. Então, dado que se **ST\_Within(A,B)** é verdade e **ST\_Within(B,A)** é verdade, então as duas geometrias são consideradas espacialmente iguais.

Desempenhado pelo módulo GEOS

Melhorias: 2.3.0 Aprimoramento para o pequeno circuito estendido PIP para suportar MultiPontos com poucos pontos. Versões anteriores só suportavam pontos em polígonos.



#### Important

Não chama com uma **GEOMETRYCOLLECTION** como um argumento

**Important**

Não usa esta função com geometrias inválidas. Você obterá resultados inesperados.

A chamada desta função irá, automaticamente, incluir uma comparação de caixa delimitadora que fará uso de quaisquer índices que esteja disponíveis nas geometrias. Para evitar o uso de índice, utilize a função `_ST_Within`.

NOTA: esta é a versão "permissível" que retorna um booleano, não um inteiro.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 // s2.1.13.3 - a.Relate(b, 'T\*\*F\*\*F\*\*\*')



This method implements the SQL/MM specification. SQL-MM 3: 5.1.30

**Exemplos**

```
--a circle within a circle
SELECT ST_Within(smallc,smallc) As smallinsmall,
       ST_Within(smallc, bigc) As smallinbig,
       ST_Within(bigc,smallc) As biginsmall,
       ST_Within(ST_Union(smallc, bigc), bigc) as unioninbig,
       ST_Within(bigc, ST_Union(smallc, bigc)) as biginunion,
       ST_Equals(bigc, ST_Union(smallc, bigc)) as bigisunion
FROM
(
SELECT ST_Buffer(ST_GeomFromText('POINT(50 50)'), 20) As smallc,
       ST_Buffer(ST_GeomFromText('POINT(50 50)'), 40) As bigc) As foo;
--Result
smallinsmall | smallinbig | biginsmall | unioninbig | biginunion | bigisunion
-----+-----+-----+-----+-----+-----
t            | t          | f          | t          | t          | t
(1 row)
```

**Veja também**

[ST\\_Contains](#), [ST\\_Equals](#), [ST\\_IsValid](#)

## 8.10 SFCGAL Funções

### 8.10.1 `postgis_sfcgal_version`

`postgis_sfcgal_version` — retorna a versão do SFCGAL em uso

#### Synopsis

texto `postgis_sfcgal_version`(void);

#### Descrição

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### 8.10.2 `ST_Extrude`

`ST_Extrude` — Extrude uma superfície a um volume relacionado

#### Synopsis

geometry `ST_Extrude`(geometry geom, float x, float y, float z);

#### Descrição

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



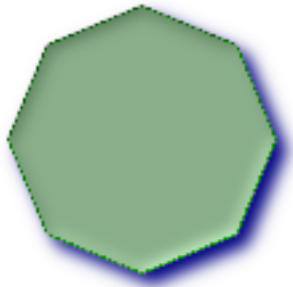


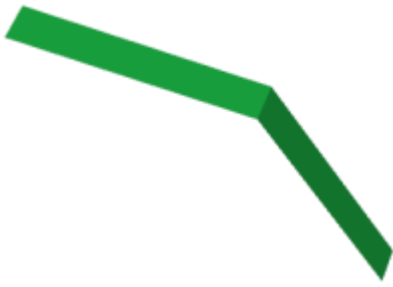
This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

As imagens 3D foram criadas usando o PostGIS xref linkend="ST\_AsX3D"/> e interpretadas no HTML usando: [X3Dom HTML Javascript rendering library](#).

<pre>SELECT ST_Buffer(ST_GeomFromText('POINT ↵ (100 90)'), 50, 'quad_segs=2'),0,0,30);</pre>  <p><i>Octágono original formado a partir do ponto buffering</i></p>	<pre>ST_Extrude(ST_Buffer(ST_GeomFromText(' ↵ POINT(100 90)'), 50, 'quad_segs=2'),0,0,30);</pre>  <p><i>30 unidades expelidas do hexágono com Z produz uma PolyhedralSurfaceZ</i></p>
<pre>SELECT ST_GeomFromText('LINESTRING(50 50, ↵ 100 90, 95 150)')</pre>  <p><i>Linestring original</i></p>	<pre>SELECT ST_Extrude( ST_GeomFromText('LINESTRING(50 50, 100 ↵ 90, 95 150)'),0,0,10);</pre>  <p><i>Linestring expelida com Z produz uma PolyhedralSurfaceZ</i></p>

**Veja também**[ST\\_AsX3D](#)**8.10.3 ST\_StraightSkeleton**

ST\_StraightSkeleton — Calcule um esqueleto em linha reta de uma geometria

**Synopsis**

geometry **ST\_StraightSkeleton**(geometry geom);

**Descrição**

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



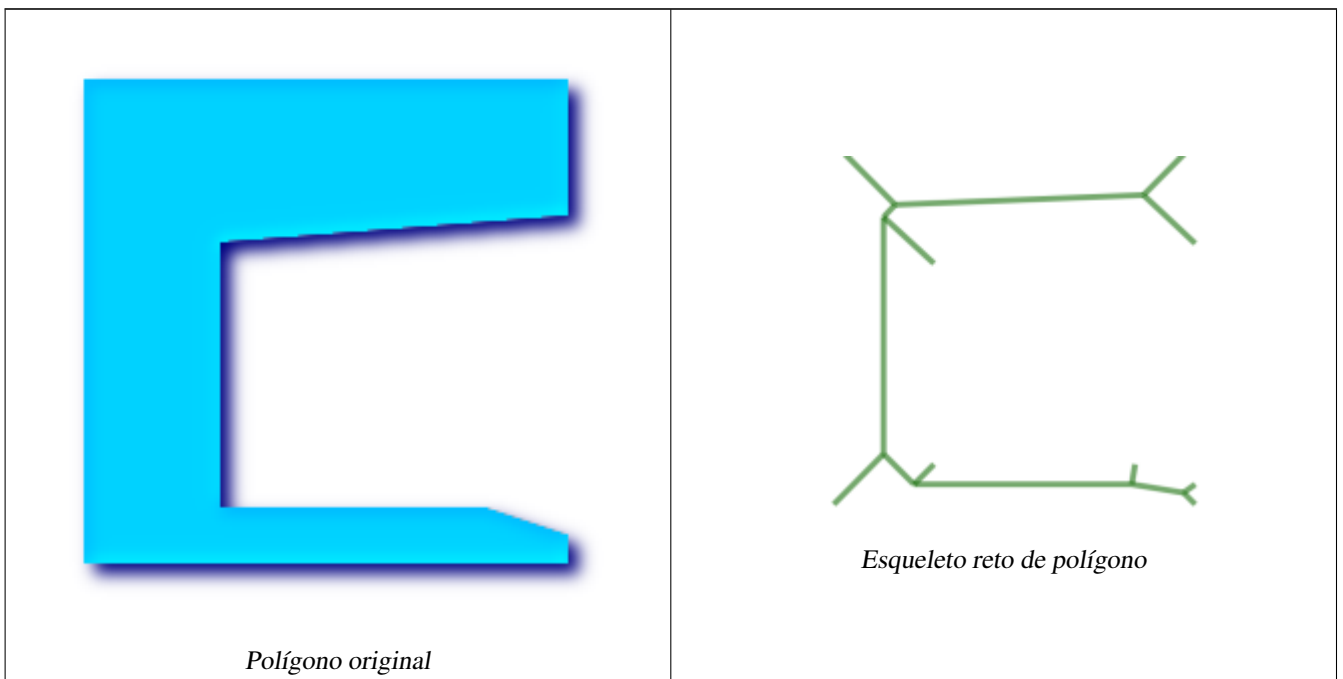
This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

**Exemplos**

```
SELECT ST_StraightSkeleton(ST_GeomFromText('POLYGON (( 190 190, 10 190, 10 10, 190 10, 190 ←  
20, 160 30, 60 30, 60 130, 190 140, 190 190 ))'));
```



### 8.10.4 ST\_ApproximateMedialAxis

ST\_ApproximateMedialAxis — Computa o eixo mediano aproximado de uma geometria territorial.

#### Synopsis

geometria **ST\_ApproximateMedialAxis**(geometria geom);

#### Descrição

Retorna um eixo mediano aproximado para a entrada territorial baseada do seu esqueleto reto. Usa uma API específica do SFCGAL quando construída contra uma versão capaz (1.2.0+). Senão a função é somente um wrapper em volta do ST\_StraightSkeleton (caso menor).

Disponibilidade: 2.2.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

#### Exemplos

```
SELECT ST_ApproximateMedialAxis(ST_GeomFromText('POLYGON (( 190 190, 10 190, 10 10, 190 10, ←  
190 20, 160 30, 60 30, 60 130, 190 140, 190 190 ))'));
```



*Um polígono e seu eixo mediano aproximado*

**Veja também**

[ST\\_StraightSkeleton](#)

### 8.10.5 ST\_IsPlanar

ST\_IsPlanar — Verifique se a superfície é ou não planar

**Synopsis**

boolean **ST\_IsPlanar**(geometry geom);

**Descrição**

Disponibilidade: 2.2.0: Isso foi documentado em 2.1.0, mas foi deixado de fora acidentalmente na 2.1.



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### 8.10.6 ST\_Orientation

ST\_Orientation — Determine orientação da superfície

**Synopsis**

integer **ST\_Orientation**(geometry geom);

**Descrição**

A função só se aplica a polígonos. Ela retorna -1 se o polígono estiver orientado no sentido anti-horário e 1 se estiver no sentido horário.

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.

### 8.10.7 ST\_ForceLHR

ST\_ForceLHR — Orientação força LHR

**Synopsis**

geometry **ST\_ForceLHR**(geometry geom);

---



## Descrição

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## 8.10.8 ST\_MinkowskiSum

ST\_MinkowskiSum — Representar soma Minkowski

### Synopsis

```
geometry ST_MinkowskiSum(geometry geom1, geometry geom2);
```

## Descrição

Essa função representa uma soma minkowski 2D de um ponto, linha ou polígono com um polígono.

Uma soma minkowski de duas geometrias A e B é o conjunto de todos os pontos que estão somados a quaisquer pontos A e B. As somas minkowski são usadas em planos em movimento e design de ajuda de computadores. Maiores detalhes em [Wikipedia Minkowski addition](#).

O primeiro parâmetro pode ser qualquer geometria 2D (ponto, linestring, polígono). Se uma geometria 3D é passada, ela será convertida para 2D forçando Z para 0, levando a possíveis casos de invalidade. O segundo parâmetro deve ser um polígono 2D.

Implementação utiliza [CGAL 2D Minkowskisum](#).

Disponibilidade: 2.1.0

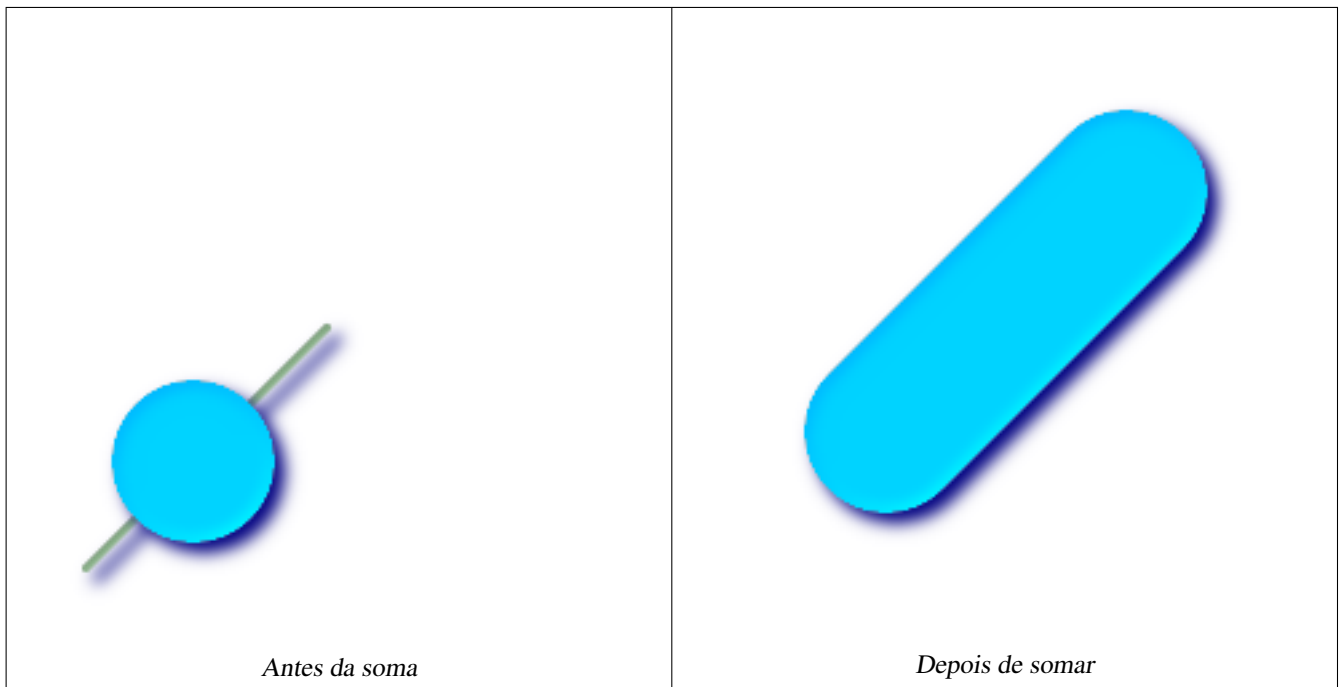


This method needs SFCGAL backend.

## Exemplos

Soma minkowski de linestring e polígono circular onde a linestring corta através do círculo

---



```

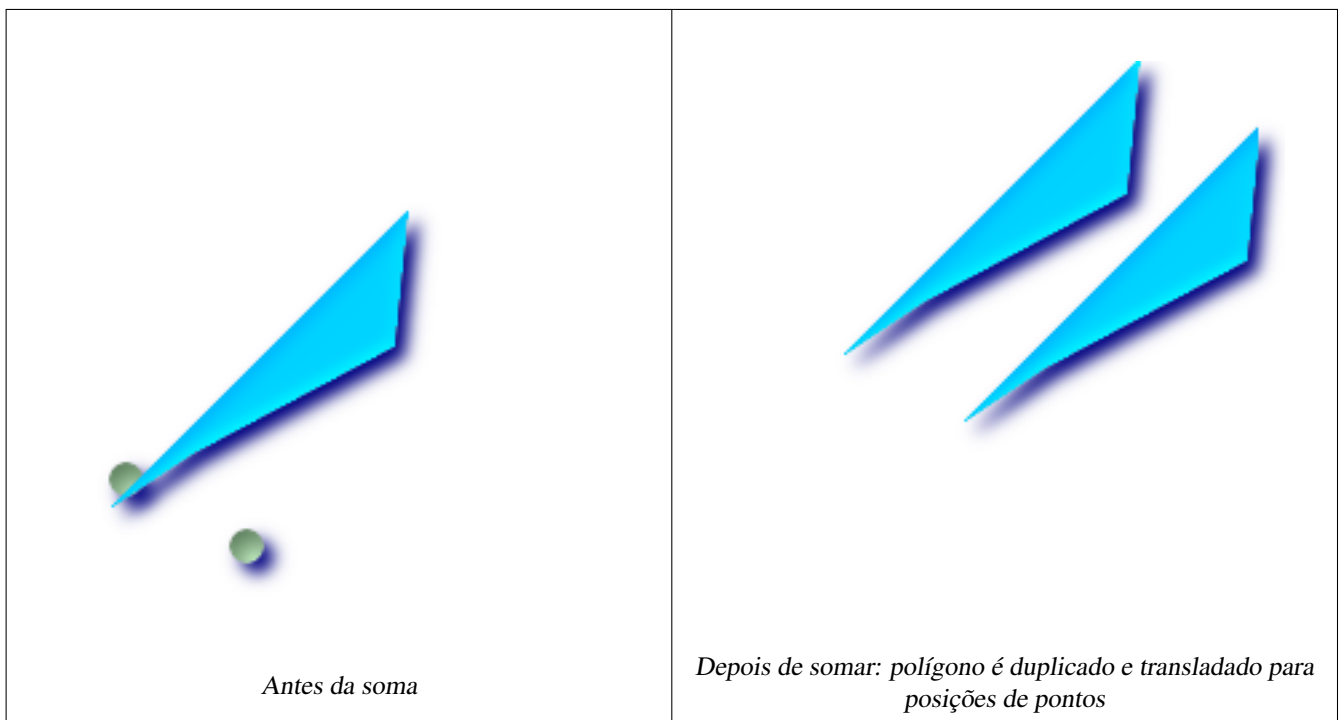
SELECT ST_MinkowskiSum(line, circle)
FROM (SELECT
  ST_MakeLine(ST_MakePoint(10, 10),ST_MakePoint(100, 100)) As line,
  ST_Buffer(ST_GeomFromText('POINT(50 50)'), 30) As circle) As foo;

-- wkt --
MULTIPOLYGON(((30 59.9999999999999,30.5764415879031 54.1472903395161,32.2836140246614 ↔
48.5194970290472,35.0559116309237 43.3328930094119,38.7867965644036 ↔
38.7867965644035,43.332893009412 35.0559116309236,48.5194970290474 ↔
32.2836140246614,54.1472903395162 30.5764415879031,60.0000000000001 30,65.8527096604839 ↔
30.5764415879031,71.4805029709527 32.2836140246614,76.6671069905881 ↔
35.0559116309237,81.2132034355964 38.7867965644036,171.213203435596 ↔
128.786796564404,174.944088369076 133.332893009412,177.716385975339 ↔
138.519497029047,179.423558412097 144.147290339516,180 150,179.423558412097 ↔
155.852709660484,177.716385975339 161.480502970953,174.944088369076 ↔
166.667106990588,171.213203435596 171.213203435596,166.667106990588 174.944088369076,
161.480502970953 177.716385975339,155.852709660484 179.423558412097,150 ↔
180,144.147290339516 179.423558412097,138.519497029047 177.716385975339,133.332893009412 ↔
174.944088369076,128.786796564403 171.213203435596,38.7867965644035 ↔
81.2132034355963,35.0559116309236 76.667106990588,32.2836140246614 ↔
71.4805029709526,30.5764415879031 65.8527096604838,30 59.9999999999999)))

```

Soma minkowski de um polígono e multiponto

---



```
SELECT ST_MinkowskiSum(mp, poly)
FROM (SELECT 'MULTIPOINT(25 50,70 25)::geometry As mp,
      'POLYGON((130 150, 20 40, 50 60, 125 100, 130 150))::geometry As poly
      ) As foo

-- wkt --
MULTIPOLYGON(
  ((70 115,100 135,175 175,225 225,70 115)),
  ((120 65,150 85,225 125,275 175,120 65))
)
```

### 8.10.9 ST\_3DIntersection

ST\_3DIntersection — Representar intersecção 3D

#### Synopsis

geometry **ST\_3DIntersection**(geometry geom1, geometry geom2);

#### Descrição

Retorna uma geometria que é dividida entre geom1 e geom2

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



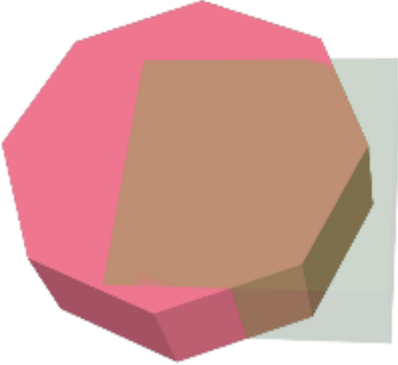
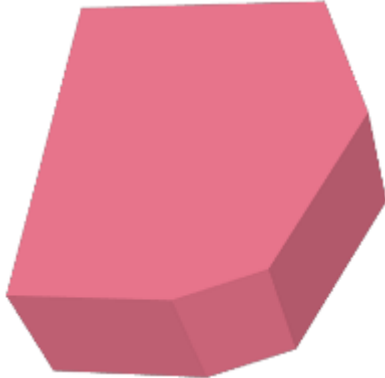
This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

As imagens 3D foram criadas usando o PostGIS xref linkend="ST\_AsX3D"/> e interpretadas no HTML usando: [X3Dom HTML Javascript rendering library](#).

<pre>SELECT ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(100 90)'),     50, 'quad_segs=2'),0,0,30) AS geom1,     ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(80 80)'),     50, 'quad_segs=1'),0,0,30) AS geom2;</pre>  <p><i>Geometrias 3D originais cobertas. geom2 é apresentada semitransparente</i></p>	<pre>SELECT ST_3DIntersection(geom1,geom2) FROM ( SELECT ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(100 90)'),     50, 'quad_segs=2'),0,0,30) AS geom1,     ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(80 80)'),     50, 'quad_segs=1'),0,0,30) AS geom2 ) As ←     t;</pre>  <p><i>Intersecção de geom1 e geom2</i></p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Linestrings 3D e polígonos

```
SELECT ST_AsText(ST_3DIntersection(linestring, polygon)) As wkt
FROM ST_GeomFromText('LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)') AS ←
    linestring
CROSS JOIN ST_GeomFromText('POLYGON((0 0 8, 0 1 8, 1 1 8, 1 0 8, 0 0 8))') AS polygon;
```

wkt

-----

LINESTRING Z (1 1 8,0.5 0.5 8)

## Cubo (superfície poliédrica fechada) e polígono Z

```
SELECT ST_AsText(ST_3DIntersection(
    ST_GeomFromText('POLYHEDRALSURFACE Z( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)) ←
    /
    ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
    ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
    ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )'),
    'POLYGON Z ((0 0 0, 0 0 0.5, 0 0.5 0.5, 0 0.5 0, 0 0 0))'::geometry))
```

```
TIN Z (((0 0 0,0 0 0.5,0 0.5 0.5,0 0 0)),((0 0.5 0,0 0 0,0 0.5 0.5,0 0.5 0)))
```

Intersecção de 2 sólidos, que resulta em uma intersecção volumétrica, também é um sólido (ST\_Dimension retorna 3)

```
SELECT ST_AsText(ST_3DIntersection( ST_Extrude(ST_Buffer('POINT(10 20) '::geometry,10,1) ←
,0,0,30),
ST_Extrude(ST_Buffer('POINT(10 20) '::geometry,10,1),2,0,10) ));
```

```
POLYHEDRALSURFACE Z (((13.3333333333333 13.3333333333333 10,20 20 0,20 20 ←
10,13.3333333333333 13.3333333333333 10)),
((20 20 10,16.6666666666667 23.3333333333333 10,13.3333333333333 13.3333333333333 ←
10,20 20 10)),
((20 20 0,16.6666666666667 23.3333333333333 10,20 20 10,20 20 0)),
((13.3333333333333 13.3333333333333 10,10 10 0,20 20 0,13.3333333333333 ←
13.3333333333333 10)),
((16.6666666666667 23.3333333333333 10,12 28 10,13.3333333333333 13.3333333333333 ←
10,16.6666666666667 23.3333333333333 10)),
((20 20 0,9.99999999999995 30 0,16.6666666666667 23.3333333333333 10,20 20 0)),
((10 10 0,9.99999999999995 30 0,20 20 0,10 10 0)),((13.3333333333333 ←
13.3333333333333 10,12 12 10,10 10 0,13.3333333333333 13.3333333333333 10)),
((12 28 10,12 12 10,13.3333333333333 13.3333333333333 10,12 28 10)),
((16.6666666666667 23.3333333333333 10,9.99999999999995 30 0,12 28 ←
10,16.6666666666667 23.3333333333333 10)),
((10 10 0,0 20 0,9.99999999999995 30 0,10 10 0)),
((12 12 10,11 11 10,10 10 0,12 12 10)),((12 28 10,11 11 10,12 12 10,12 28 10)),
((9.99999999999995 30 0,11 29 10,12 28 10,9.99999999999995 30 0)),((0 20 0,2 20 ←
10,9.99999999999995 30 0,0 20 0)),
((10 10 0,2 20 10,0 20 0,10 10 0)),((11 11 10,2 20 10,10 10 0,11 11 10)),((12 28 ←
10,11 29 10,11 11 10,12 28 10)),
((9.99999999999995 30 0,2 20 10,11 29 10,9.99999999999995 30 0)),((11 11 10,11 29 ←
10,2 20 10,11 11 10)))
```

### 8.10.10 ST\_3DDifference

ST\_3DDifference — Representar diferença 3D

#### Synopsis

geometry **ST\_3DDifference**(geometry geom1, geometry geom2);

#### Descrição

Retorna aquela parte de geom1 que não faz parte de geom2.

Disponibilidade: 2.2.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



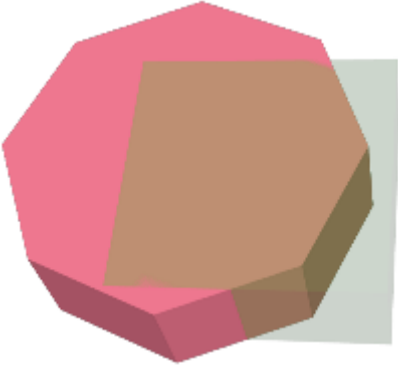
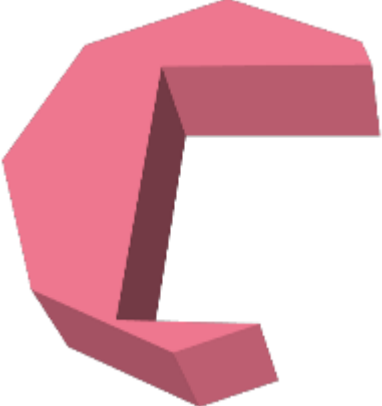
This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

As imagens 3D foram criadas usando o PostGIS xref linkend="ST\_AsX3D"/> e interpretadas no HTML usando: [X3Dom HTML Javascript rendering library](#).

<pre>SELECT ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(100 90)'),     50, 'quad_segs=2'),0,0,30) AS geom1,     ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(80 80)'),     50, 'quad_segs=1'),0,0,30) AS geom2;</pre>  <p><i>Geometrias 3D originais cobertas. geom2 é a parte que será removida.</i></p>	<pre>SELECT ST_3DDifference(geom1,geom2) FROM ( SELECT ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(100 90)'),     50, 'quad_segs=2'),0,0,30) AS geom1,     ST_Extrude(ST_Buffer( ←     ST_GeomFromText('POINT(80 80)'),     50, 'quad_segs=1'),0,0,30) AS geom2 ) As ←     t;</pre>  <p><i>O que restou depois de remover geom2</i></p>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Veja também

[ST\\_Extrude](#), [ST\\_AsX3D](#), [ST\\_3DIntersection](#) [ST\\_3DUnion](#)

### 8.10.11 ST\_3DUnion

ST\_3DUnion — Representar união 3D

#### Synopsis

geometry **ST\_3DUnion**(geometry geom1, geometry geom2);

#### Descrição

Disponibilidade: 2.2.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



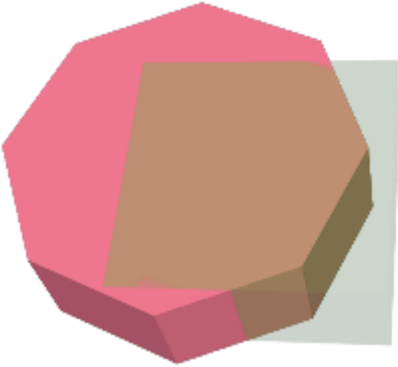
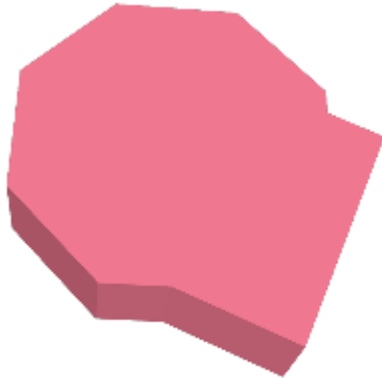
This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### Exemplos

As imagens 3D foram criadas usando o PostGIS xref linkend="ST\_AsX3D"/> e interpretadas no HTML usando: [X3Dom HTML Javascript rendering library](#).

<pre>SELECT ST_Extrude(ST_Buffer( ↵     ST_GeomFromText ('POINT(100 90)'),     50, 'quad_segs=2'),0,0,30) AS geom1,     ST_Extrude(ST_Buffer( ↵     ST_GeomFromText ('POINT(80 80)'),     50, 'quad_segs=1'),0,0,30) AS geom2;</pre>  <p><i>Geometrias 3D originais cobertas. geom2 é a com transparência.</i></p>	<pre>SELECT ST_3DUnion(geom1,geom2) FROM ( SELECT ST_Extrude(ST_Buffer( ↵     ST_GeomFromText ('POINT(100 90)'),     50, 'quad_segs=2'),0,0,30) AS geom1,     ST_Extrude(ST_Buffer( ↵     ST_GeomFromText ('POINT(80 80)'),     50, 'quad_segs=1'),0,0,30) AS geom2 ) As ↵ t;</pre>  <p><i>União de geom1 e geom2</i></p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Veja também

[ST\\_Extrude](#), [ST\\_AsX3D](#), [ST\\_3DIntersection](#) [ST\\_3DDifference](#)

### 8.10.12 ST\_3DArea

ST\_3DArea — Computa a área de geometrias de superfície 3D. Irá retornar 0 para sólidos.

#### Synopsis

```
floatST_3DArea(geometry geom1);
```

#### Descrição

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

Nota: Por padrão uma superfície poliédrica construída de um WKT, é uma superfície de geometria, não sólida. Ela, portanto, tem uma área de superfície. Uma vez convertido em sólido, não tem nenhuma área.

```
SELECT ST_3DArea(geom) As cube_surface_area,
       ST_3DArea(ST_MakeSolid(geom)) As solid_surface_area
FROM (SELECT 'POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
  ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
  ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
  ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
  ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
  ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )'::geometry) As f(geom);
```

cube_surface_area	solid_surface_area
6	0

## Veja também

[ST\\_Area](#), [ST\\_MakeSolid](#), [ST\\_IsSolid](#), [ST\\_Area](#)

### 8.10.13 ST\_Tessellate

**ST\_Tessellate** — Representa superfície tesselação de um polígono ou superfície poliédrica e retorna como uma TIN ou coleção de TINS

#### Synopsis

```
geometry ST_Tessellate(geometry geom);
```

#### Descrição

Usa como entrada uma superfície como um MULTI(POLÍGONO) ou SUPERFÍCIEPOLIÉDRICA e retorna uma representação TIN via o processo de tesselação usando triângulos.

Disponibilidade: 2.1.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.





This function supports Polyhedral surfaces.





This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos



<pre>SELECT ST_GeomFromText('POLYHEDRALSURFACE ↵   Z( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)), ↵   ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ↵   ((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 1), ↵   (0 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0</pre>  <p><i>Cubo original</i></p>	<pre>SELECT ST_Tessellate(ST_GeomFromText(' ↵   POLYHEDRALSURFACE Z( ((0 0 0, 0 0 1, 0 1 ↵   ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 ↵   0)), ((0 0 0, 1 0 0, 1 0 1, 0 1, 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, 0</pre> <p><b>ST_AsText output:</b></p> <pre>SELECT ST_AsText(ST_GeomFromText('POLYHEDRALSURFACE ↵   Z( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)), ↵   ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ↵   ((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 1), ↵   (0 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0</pre>  <p><i>Cubo tesselado com triângulos coloridos</i></p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<pre>SELECT 'POLYGON (( 10 190, 10 70, 80 70, ↵ 80 130, 50 160, 120 160,</pre>  <p><i>Polígono original</i></p>	<pre>SELECT   ST_Tessellate('POLYGON (( 10 190, ↵ 120 190, 10 190 ))',:geometry; FIN(((80 130,50 160,80 70,80 130)),((50 ↵ 160,10 190,10 70,50 160)), ((80 70,50 160,10 70,80 70)) ↵ ,((120 160,120 190,50 160,120 1 ((120 190,10 190,50 160,120 190)))</pre>  <p><i>Polígono tesselado</i></p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### 8.10.14 ST\_Volume

**ST\_Volume** — Computa o volume de um sólido 3D. Se aplicado a geometrias com superfícies (mesmo fechadas), irão retornar 0.

##### Synopsis

```
float ST_Volume(geometry geom1);
```

##### Descrição

Disponibilidade: 2.2.0

- ✓ This method needs SFCGAL backend.
- ✓ This function supports 3d and will not drop the z-index.
- ✓ This function supports Polyhedral surfaces.
- ✓ This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplo

Quando superfícies fechadas são criadas com WKT, elas são tratadas como territoriais ao invés de sólido. Para torná-las sólidos, você precis usar [ST\\_MakeSolid](#). Geometrias territoriais não têm volume. Aqui está um exemplo demonstrativo.

```
SELECT ST_Volume(geom) As cube_surface_vol,
       ST_Volume(ST_MakeSolid(geom)) As solid_surface_vol
FROM (SELECT 'POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
  ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
  ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
  ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
  ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
  ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )'::geometry) As f(geom);
```

cube_surface_vol	solid_surface_vol
0	1

## Veja também

[ST\\_3DArea](#), [ST\\_MakeSolid](#), [ST\\_IsSolid](#)

### 8.10.15 ST\_MakeSolid

**ST\_MakeSolid** — Molde a geometria para um sólido. Nenhuma verificação é apresentada. Para obter um sólido válido, a geometria de entrada deve ser uma superfície poliédrica fechada ou um TIN fechado.

#### Synopsis

```
geometry ST_MakeSolid(geometry geom1);
```

#### Descrição

Disponibilidade: 2.2.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### 8.10.16 ST\_IsSolid

**ST\_IsSolid** — teste se a geometria é um sólido. Nenhuma verificação de validade é representada.

#### Synopsis

```
boolean ST_IsSolid(geometry geom1);
```

**Descrição**

Disponibilidade: 2.2.0



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

**8.11 Processamento de Geometria****8.11.1 ST\_Buffer**

`ST_Buffer` — (T) Retorna uma geometria cobrindo todos os pontos com a dada distância da geometria de entrada.

**Synopsis**

```
geometry ST_Buffer(geometry g1, float radius_of_buffer);
geometry ST_Buffer(geometry g1, float radius_of_buffer, integer num_seg_quarter_circle);
geometry ST_Buffer(geometry g1, float radius_of_buffer, text buffer_style_parameters);
geography ST_Buffer(geography g1, float radius_of_buffer_in_meters);
geography ST_Buffer(geography g1, float radius_of_buffer, integer num_seg_quarter_circle);
geography ST_Buffer(geography g1, float radius_of_buffer, text buffer_style_parameters);
```

**Descrição**

Retorna uma geometria/geografia que representa todos os pontos cujas distâncias desta geometria/geografia são menores ou iguais à distância.

Geometria: Cálculos estão no Sistema de Referência Espacial da geometria. Introduzido no suporte 1.5 para um fim diferente e configurações para fazer uma esquadria para controlar o formato.

**Note**

Alcance negativo: Para polígonos, um rádio negativo pode ser usado, ele irá reduzir em vez de expandir.

**Note**

Geografia: Para geografia, isto é uma fina embalagem em torna da implementação de geometria. Isso primeiro determina o melhor SRID que se encaixa na caixa limitadora do objeto geográfico (favorecendo UTM, Lambert Azimuthal Equal Area (LAEA) pólo sul/norte, e caindo no pior caso de cenário mercator) e os buffers naquela referência espacial plana na geografia WGS84.



para geografia, isto pode não se comportar da melhor maneira se o objeto é grande o suficiente para cair entre as zonas UTM ou cruzar a linha da data

Enhanced: 2.5.0 - `ST_Buffer` geometry support was enhanced to allow for side buffering specification `side=both|left|right`.

Disponibilidade: 1.5 - `ST_Buffer` foi melhorado para suportar diferentes tampas de extremidade. Estas são úteis para converter linestrings de estrada para estradas de polígono com limites quadrados ou planos em vez dos novos. A embalagem fina para geografia foi acionada. - GEOS  $\geq$  3.2 para tomar vantagem da funcionalidade avançada de geometria.

O terceiro parâmetro opcional (que só se aplica à geometria, no momento) pode ou especificar números de segmentos usados para aproximar um quarto de círculo (caso inteireza, leva a 8) ou uma lista de espaços em branco separando pares de valores (caso string) para ajusta operações como segue:

- `'quad_segs=#'` : número de segmentos usado para aproximar um quarto de círculo (leva a 8).
- `'endcap=round|flat|square'` : endcap style (vai para "round", precisa de GEOS-3.2 para valores diferentes). 'butt' também é aceitado como um sinônimo para 'flat'.
- `'join=round|mitre|bevel'` : join style (vai para "round", precisa de GEOS-3.2 para valores diferentes). 'miter' também é aceitado como um sinônimo para 'mitre'.
- `'mitre_limit=#.#'` : mitre ratio limit (só afeta o estilo mitered join). 'miter\_limit' também é aceito como sinônimo para 'mitre\_limit'.
- `'side=both|left|right'` : 'left' or 'right' performs a single-sided buffer on the geometry, with the buffered side relative to the direction of the line. This is only really relevant to LINestring geometry and does not affect POINT or POLYGON geometries. By default end caps are square.

Unidades de raio são medidas nas unidades do sistema de referência espacial.

As saídas podem ser: POINTS, MULTIPOINTS, LINESTRINGS, MULTILINESTRINGS, POLYGONS, MULTIPOLYGONS, and GeometryCollections.



#### Note

Esta função ignora a terceira dimensão (z) e sempre oferece um amortecedor 2-d mesmo com uma geometria 3-d presente.

Desempenhado pelo módulo GEOS.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3



This method implements the SQL/MM specification. SQL-MM 3: 5.1.17



#### Note

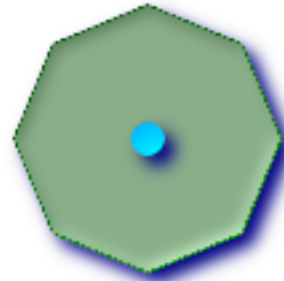
As pessoas cometem o erro de utilizar esta função para tentar fazer pesquisas de raio. Criar um buffer para essa pesquisa é retrocesso e sem sentido. Utilize [ST\\_DWithin](#).

## Exemplos



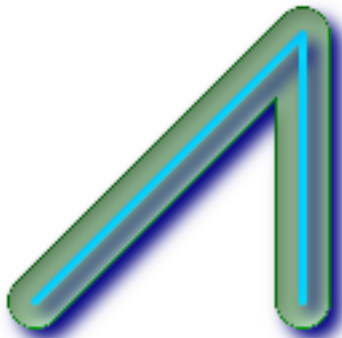
*quad\_segs=8 (default)*

```
SELECT ST_Buffer(
  ST_GeomFromText('POINT(100 90)'),
  50, 'quad_segs=8');
```



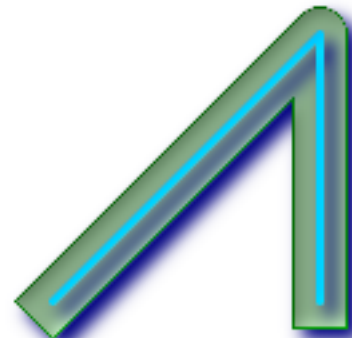
*quad\_segs=2 (lame)*

```
SELECT ST_Buffer(
  ST_GeomFromText('POINT(100 90)'),
  50, 'quad_segs=2');
```



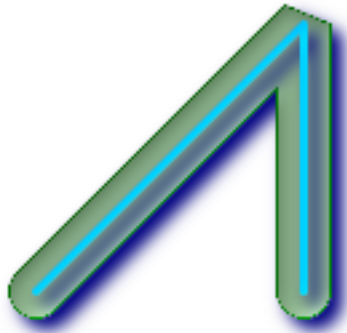
*endcap=round join=round (default)*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'endcap=round join=round');
```



*endcap=square*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'endcap=square join=round');
```



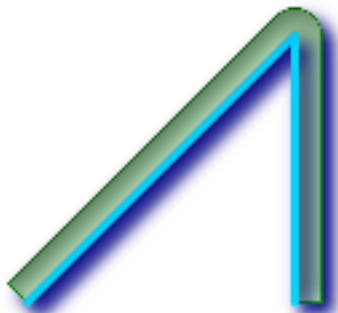
*join=bevel*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=bevel');
```



*join=mitre mitre\_limit=5.0 (default mitre limit)*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=mitre mitre_limit=5.0');
```



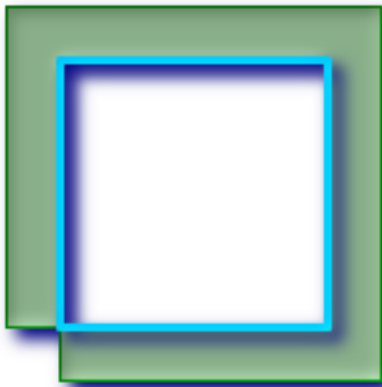
*side=left*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=bevel');
```



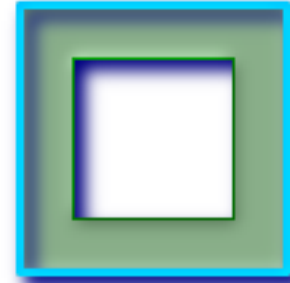
*side=right*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=bevel');
```



*right-hand-winding, polygon boundary side=left*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=bevel');
```



*right-hand-winding, polygon boundary side=right*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=bevel');
```

```
--A buffered point approximates a circle
-- A buffered point forcing approximation of (see diagram)
-- 2 points per circle is poly with 8 sides (see diagram)
SELECT ST_NPoints(ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50)) As ←
  promisingcircle_pcount,
ST_NPoints(ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50, 2)) As lamecircle_pcount;

promisingcircle_pcount | lamecircle_pcount
-----+-----
                        33 | 9

--A lighter but lamer circle
-- only 2 points per quarter circle is an octagon
--Below is a 100 meter octagon
-- Note coordinates are in NAD 83 long lat which we transform
to Mass state plane meter and then buffer to get measurements in meters;
SELECT ST_AsText(ST_Buffer(
  ST_Transform(
    ST_SetSRID(ST_MakePoint(-71.063526, 42.35785),4269), 26986)
  ,100,2)) As octagon;
-----
POLYGON((236057.59057465 900908.759918696,236028.301252769 900838.049240578,235
957.59057465 900808.759918696,235886.879896532 900838.049240578,235857.59057465
900908.759918696,235886.879896532 900979.470596815,235957.59057465 901008.759918
696,236028.301252769 900979.470596815,236057.59057465 900908.759918696))
```

**Veja também.**

[ST\\_Collect](#), [ST\\_DWithin](#), [ST\\_SetSRID](#), [ST\\_Transform](#), [ST\\_Union](#)



## 8.11.2 ST\_BuildArea

ST\_BuildArea — Cria uma geometria formada pela linwork constituinte da geometria

### Synopsis

geometry **ST\_BuildArea**(geometry A);

### Descrição

Cria uma geometria formada pela linwork constituinte da dada geometria. O tipo que retorna pode ser um polígono ou um multipolígono, depende da entrada. Se a entrada não formar polígonos, retorna NULA. As entradas podem ser LINESTRINGS, MULTILINESTRINGS, POLÍGONOS, MULTIPOLÍGONOS, e GeometryCollections.

Esta função irá supor que todas as geometrias internas representam buracos



#### Note

A linework de entrada deve ser nodificada corretamente para esta função trabalhar normalmente.

Disponibilidade: 1.1.0 - requer GEOS >= 2.1.0.

### Exemplos



*Isto irá criar um donut*

```
SELECT ST_BuildArea(ST_Collect(smallc,bigc))
FROM (SELECT
      ST_Buffer(
        ST_GeomFromText('POINT(100 90)'), 25) As smallc,
      ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50) As bigc) As foo;
```



*Isto criará um buraco aberto dentro do círculo com dentes saindo*

```
SELECT ST_BuildArea(ST_Collect(line,circle))
FROM (SELECT
  ST_Buffer(
    ST_MakeLine(ST_MakePoint(10, 10),ST_MakePoint(190, 190)),
    5) As line,
  ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50) As circle) As foo;

--this creates the same gaping hole
--but using linestrings instead of polygons
SELECT ST_BuildArea(
  ST_Collect(ST_ExteriorRing(line),ST_ExteriorRing(circle))
)
FROM (SELECT ST_Buffer(
  ST_MakeLine(ST_MakePoint(10, 10),ST_MakePoint(190, 190))
  ,5) As line,
  ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50) As circle) As foo;
```

#### **Veja também.**

[ST\\_Node](#), [ST\\_MakePolygon](#), [ST\\_BdPolyFromText](#), [ST\\_BdMPolyFromText](#) wrappers to this function with standard OGC interface

### **8.11.3 ST\_ClipByBox2D**

`ST_ClipByBox2D` — Retorna a porção de uma geometria caindo dentro de um retângulo.

#### **Synopsis**

geometry `ST_ClipByBox2D`(geometry geom, box2d box);

## Descrição

Corta uma geometria através de uma caixa 2D de uma maneira rápida, mas não muito limpa. A geometria de saída não está garantida para ser válida (atuas interseções para um polígono podem ser introduzidas). Geometrias de entrada topologicamente inválidas não resultam em exceções.

Desempenhado pelo módulo GEOS.



**Note**  
Requer GEOS 3.5.0+

Disponibilidade: 2.2.0 - requer GEOS >= 3.5.0.

## Exemplos

```
-- Rely on implicit cast from geometry to box2d for the second parameter
SELECT ST_ClipByBox2D(the_geom, ST_MakeEnvelope(0,0,10,10)) FROM mytab;
```

## Veja também.

[ST\\_Intersection](#), [ST\\_MakeBox2D](#), [ST\\_MakeEnvelope](#)

### 8.11.4 ST\_Collect

`ST_Collect` — Retorna um valor `ST_Geometry` específico de uma coleção de outras geometrias.

#### Synopsis

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

## Descrição

O tipo de saída pode ser uma `MULTI *` ou uma `GEOMETRYCOLLECTION`. Vem com 2 variantes. A variante 1 coleta 2 geometrias. E variante 2 é uma função agregada que pega um conjunto de geometrias e as coloca em uma única `ST_Geometry`.

Versão agregada: Esta função retorna uma `GEOMETRYCOLLECTION` ou um `MULTI` objeto de um conjunto de geometrias. A função `ST_Collect()` é uma função "agregada" na terminologia do PostgreSQL. Isso significa que ela opera em filas de dados, da mesma maneira que as funções `SUM()` e `AVG()`. Por exemplo: "`SELECT ST_Collect(GEOM) FROM GEOMTABLE GROUP BY ATTRCOLUMN`" irá retornar uma `GEOMETRYCOLLECTION` para cada valor distinto de `ATTRCOLUMN`.

Versão não agregada: Esta função retorna uma geometria sendo uma coleção de duas geometrias de entrada. O tipo de saída pode ser uma `MULTI*` ou `GEOMETRYCOLLECTION`.



**Note**  
As `ST_Collect` e `ST_Union` são imutáveis, normalmente. `ST_Collect` está nos comandos gerais de grandeza mais rápida que `ST_Union`, porque ela não tenta desfazer limites ou validar que um MultiPolígono construído não possui regiões sobrepostas. Isto apenas puxa geometrias únicas para dentro de `MULTI` e `MULTI` ou mistura tipos de geometria dentro de Coleções de Geometrias. Infelizmente, as coleções não são bem suportadas pelas ferramentas GIS. Para prevenir que a `ST_Collect` retorne uma coleção de geometrias quando estiver coletando `MULTI` geometrias, pode-se usar o artifício abaixo o qual utiliza `ST_Dump` para expandir as `MULTI`s para únicas e depois reagrupá-las.

Disponibilidade: 1.4.0 - `ST_Collect(geomarray)` foi introduzida. A `ST_Collect` foi melhorada para lidar com mais geometrias mais rápido.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves Este método suporta Strings Circulares e Curvas, mas nunca retornará uma MULTICURVA ou MULTI como o esperado e o PostGIS não suporta elas no momento.

## Exemplos

### Exemplo agregado

```
SELECT stusps,
       ST_Multi(ST_Collect(f.the_geom)) as singlegeom
FROM (SELECT stusps, (ST_Dump(the_geom)).geom As the_geom
      FROM
      somestatable ) As f
GROUP BY stusps
```

### Exemplo não agregado

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

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

--Collect 2 d points
SELECT ST_AsText(ST_Collect(ST_GeomFromText('POINT(1 2)'),
                          ST_GeomFromText('POINT(1 2)')));

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

--Collect 3d points
SELECT ST_AsEWKT(ST_Collect(ST_GeomFromEWKT('POINT(1 2 3)'),
                          ST_GeomFromEWKT('POINT(1 2 4)')));

          st_asewkt
-----
MULTIPOINT(1 2 3,1 2 4)

--Example with curves
SELECT ST_AsText(ST_Collect(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'),
                          ST_GeomFromText('CIRCULARSTRING(220227 150406,220227 150407,220227 150406)')));

-----
GEOMETRYCOLLECTION(CIRCULARSTRING(220268 150415,220227 150505,220227 150406),
                   CIRCULARSTRING(220227 150406,220227 150407,220227 150406))

--New ST_Collect array construct
SELECT ST_Collect(ARRAY(SELECT the_geom FROM sometable));

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

```
--wkt collect --  
MULTILINESTRING((1 2,3 4),(3 4,4 5))
```

Veja também.

[ST\\_Dump](#), [ST\\_Union](#)

### 8.11.5 ST\_ConcaveHull

`ST_ConcaveHull` — O casco côncavo de uma geometrias representa uma possível geometria côncava que envolve todas as geometrias dentro do conjunto. Você pode pensar nele como um encolhimento.

#### Synopsis

geometry **ST\_ConcaveHull**(geometry geomA, float target\_percent, boolean allow\_holes=false);

#### Descrição

O casco côncavo de uma geometrias representa uma possível geometria côncava que envolve todas as geometrias dentro do conjunto. É falsa por permitir polígonos com buracos. O resultado nunca é maior que um único polígono.

O `target_percent` é o percentual alvo da área de casco convexo que a solução PostGIS tentará aproximar antes de desistir ou sair. Você pode pensar no casco côncavo como a geometria que conseguirá pegar através da vedação a vácuo de um conjunto de geometrias. O `target_percent` de 1 fornecerá a mesma resposta do casco convexo. Um `target_percent` entre 0 e 0.99 fornecerá algo que com uma área menor que o casco convexo. Isto é diferente de um casco convexo que é mais propenso a empacotar uma banda envolta de um conjunto de geometrias.

Frequentemente usado com `MULTI` e Coleções de Geometrias. Embora não seja um agregado - você pode usar em conjunto com `ST_Collect` or `ST_Union` para obter o casco côncavo de um conjunto de pontos/linestring/polígonos `ST_ConcaveHull(ST_Collect(somepolygons))` (0.80).

É muito mais lento calcular do que o casco convexo mas envolve a geometria melhor e também é útil para reconhecimento de imagem.

Desempenhado pelo módulo GEOS



#### Note

Nota - Se você está usando com pontos, linestrings ou coleções de geometrias, use `ST_Collect`. Se estiver usando com polígonos, use `ST_Union` já que pode falhar com geometrias inválidas.



#### Note

Nota - Quanto menor você fizer a porcentagem alvo mais tempo demorará para processar o casco côncavo e a chance de colidir com exceções topológicas é maior. Além disso, quanto mais pontos flutuantes mais números de pontos você acumula. Primeiramente, tente 0.99 que faz um primeiro salto, muito rápido, às vezes tão rápido como o casco convexo, e geralmente oferece muito melhor que 99% de encolher já que quase sempre passa dos limites. O segundo salto de 0.98 mai lento, os outros ficam mais lentos quadraticamente. Para reduzir a precisão e os pontos flutuantes, use `ST_SimplifyPreserveTopology` ou `ST_SnapToGrid` depois `ST_ConcaveHull`. A `ST_SnapToGrid` é um pouco mais rápida, não é resultante em geometrias onde, como `ST_SimplifyPreserveTopology` quase sempre preserva a validade da geometria.

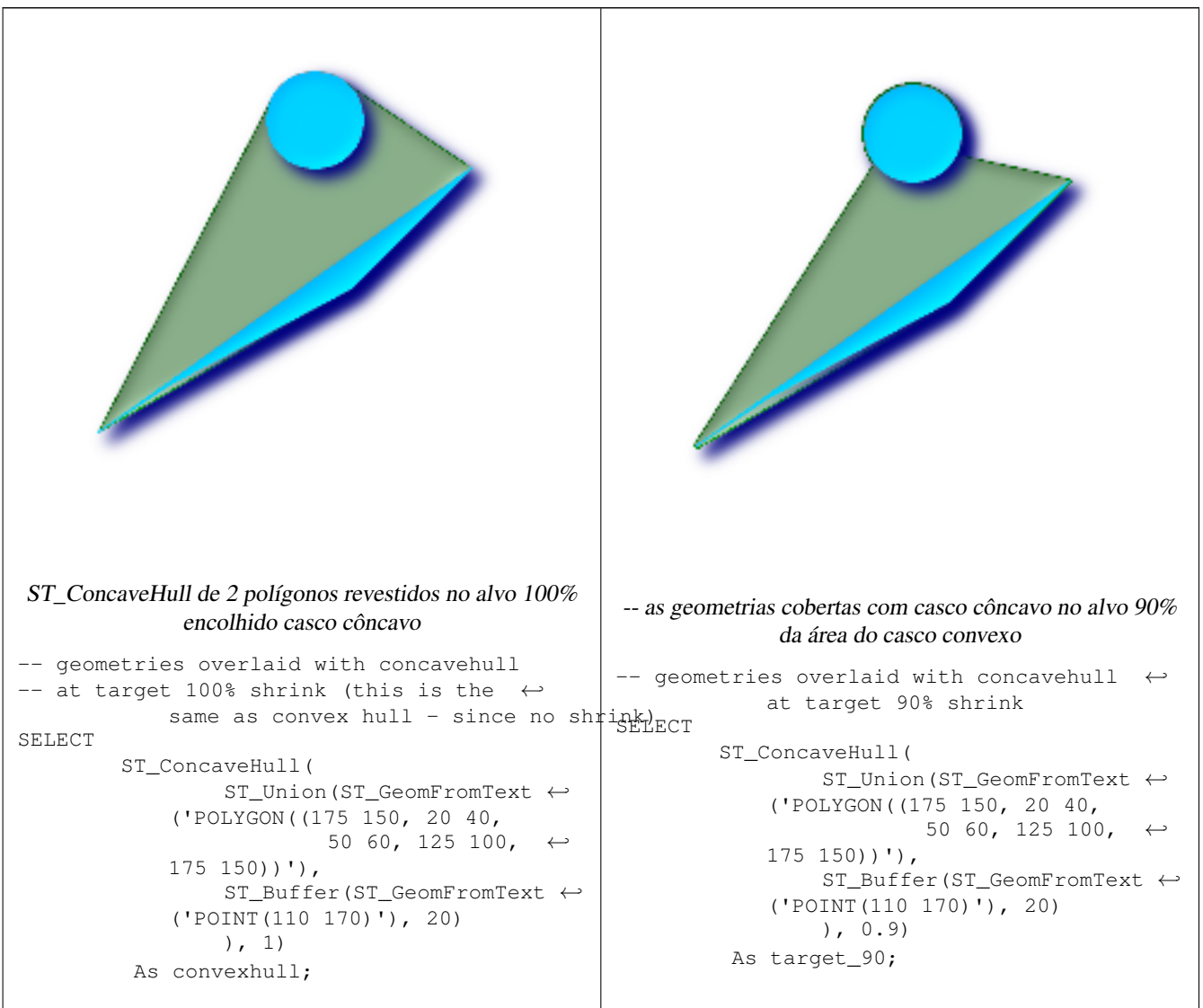
Mais exemplos do mundo real e uma breve explicação da técnica em: [http://www.bostongis.com/postgis\\_concavehull.snippet](http://www.bostongis.com/postgis_concavehull.snippet)

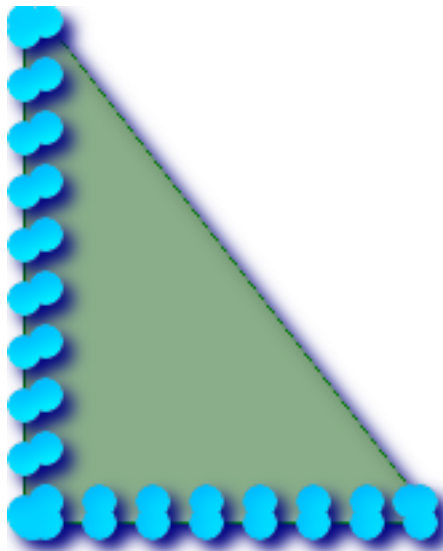
Veja também o artigo do Simon Greener mostrando o ConcaveHull introduzido no Oracle 11G R2. [http://www.spatialdbadvisor.com/oracle\\_spatial\\_tips\\_tricks/172/concave-hull-geometries-in-oracle-11gr2](http://www.spatialdbadvisor.com/oracle_spatial_tips_tricks/172/concave-hull-geometries-in-oracle-11gr2). A solução que pegamos na porcentagem do alvo 0.75 do casco convexo é parecida com o formato que o Simon pega com with Oracle SDO\_CONCAVEHULL\_BOUNDARY.

Disponibilidade: 2.0.0

## Exemplos

```
--Get estimate of infected area based on point observations
SELECT d.disease_type,
       ST_ConcaveHull(ST_Collect(d.pnt_geom), 0.99) As geom
FROM disease_obs As d
GROUP BY d.disease_type;
```

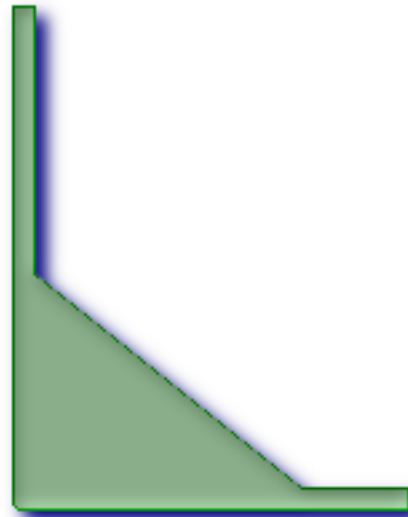




*Os pontos L Shapes revistem isso com um casco convexo*

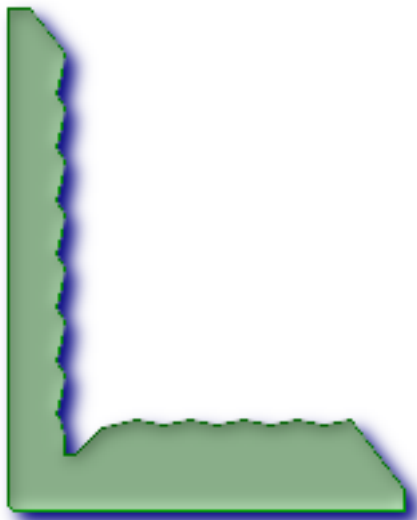
```
-- this produces a table of 42 points ←
      that form an L shape
SELECT (ST_DumpPoints(ST_GeomFromText(
'MULTIPOINT(14 14,34 14,54 14,74 14,94 ←
      14,114 14,134 14,
150 14,154 14,154 6,134 6,114 6,94 6,74 ←
      6,54 6,34 6,
14 6,10 6,8 6,7 7,6 8,6 10,6 30,6 50,6 ←
      70,6 90,6 110,6 130,
6 150,6 170,6 190,6 194,14 194,14 174,14 ←
      154,14 134,14 114,
14 94,14 74,14 54,14 34,14 14)'))).geom
      INTO TABLE l_shape;

SELECT ST_ConvexHull(ST_Collect(geom))
FROM l_shape;
```



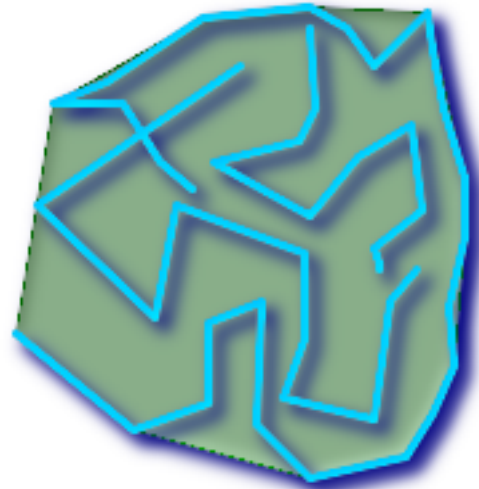
*ST\_ConcaveHull of L points at target 99% of convex hull*

```
SELECT ST_ConcaveHull(ST_Collect(geom), ←
      0.99)
FROM l_shape;
```

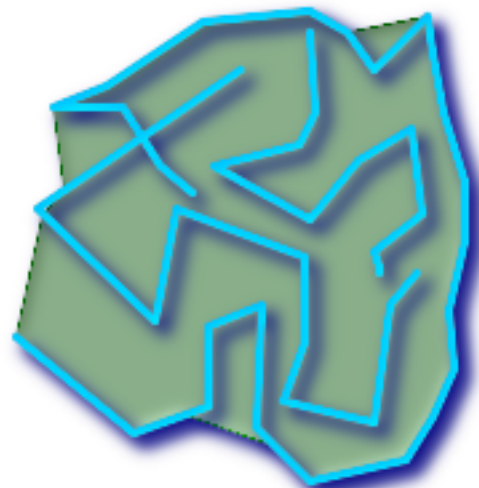


*Casco cômcavo de L pontos no alvo 80%*

```
-- Concave Hull L shape points
-- at target 80% of convexhull
SELECT ST_ConcaveHull(ST_Collect(↵
    geom), 0.80)
FROM l_shape;
```



*multilinestring reveste com Casco convexo.*



*multilinestring com coberturas com cascos cômcavos de linestrings em um alvo de 99% -- primeiro salto*

```
SELECT ST_ConcaveHull(ST_GeomFromText('↵
    MULTILINESTRING((106 164,30 112,74 70,82↵
    130 62,122 40,156 32,162 76,172 ↵
    88),↵
    (132 178,134 148,128 136,96 128,132 ↵
    108,150 130,↵
    170 142,174 110,156 96,158 90,158 88),↵
    (22 64,66 28,94 38,94 68,114 76,112 30,↵
    132 10,168 18,178 34,186 52,184 74,190 ↵
    100,↵
    190 122,182 148,178 170,176 184,156 ↵
    164,146 178,↵
    132 186,92 182,56 158,36 150,62 150,76 ↵
    128,88 118))'),0.99)
```



Veja também.

[ST\\_Collect](#), [ST\\_ConvexHull](#), [ST\\_SimplifyPreserveTopology](#), [ST\\_SnapToGrid](#)

### 8.11.6 ST\_ConvexHull

`ST_ConvexHull` — O casco convexo de uma geometria representa a mínima geometria convexa que envolve todas as geometrias dentro do conjunto.

#### Synopsis

```
geometry ST_ConvexHull(geometry geomA);
```

#### Descrição

O casco convexo de uma geometria representa a mínima geometria convexa que envolve todas as geometrias dentro do conjunto.

Pode-se pensar no casco convexo como a geometria que se obtém envolvendo uma banda elástica em volta de um conjunto de geometrias. Isto é diferente de um casco côncavo que é análogo ao encolhimento nas suas geometrias.

Comumente usado com `MULTI` e Coleções de Geometrias. Embora não seja um agregado - você pode usar em conjunto com `ST_Collect` para obter o casco convexo de um conjunto de pontos. `ST_ConvexHull(ST_Collect(somepointfield))`.

Regularmente usado para determinar uma área afetada baseado em um conjunto de observações pontuais.

Desempenhado pelo módulo GEOS



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3



This method implements the SQL/MM specification. SQL-MM 3: 5.1.16



This function supports 3d and will not drop the z-index.

#### Exemplos

```
--Get estimate of infected area based on point observations
SELECT d.disease_type,
       ST_ConvexHull(ST_Collect(d.the_geom)) As the_geom
FROM disease_obs As d
GROUP BY d.disease_type;
```



*Casco convexo de uma MultiLineString e um MultiPonto vistos juntos com a MultiLineString e MultiPonto*

```
SELECT ST_AsText(ST_ConvexHull(
  ST_Collect(
    ST_GeomFromText('MULTILINESTRING((100 190,10 8),(150 10, 20 30)'),
    ST_GeomFromText('MULTIPOINT(50 5, 150 30, 50 10, 10 10)')
  )));
---st_astext---
POLYGON((50 5,10 8,10 10,100 190,150 30,150 10,50 5))
```

**Veja também.**

[ST\\_Collect](#), [ST\\_ConcaveHull](#), [ST\\_MinimumBoundingCircle](#)

### 8.11.7 ST\_CurveToLine

ST\_CurveToLine — Converte uma CIRCULARSTRING/CURVEPOLYGON para uma LINESTRING/POLYGON

#### Synopsis

geometry **ST\_OffsetCurve**(geometry line, float signed\_distance, text style\_parameters=’);

#### Descrição

Converte uma CIRCULAR STRING para uma LINESTRING normal ou um CURVEPOLYGON para um POLYGON. Útil para aparelhos que não suportam tipos de geometria CIRCULARSTRING

Converte uma dada geometria para uma geometrias linear. Cada geometria curvada ou segmento é convertido para uma aproximação linear usando o valor padrão de 32 segmentos por quarto de círculo

The ‘tolerance\_type’ argument determines interpretation of the ‘tolerance’ argument. It can take the following values:

- 0 (default): Tolerance is max segments per quadrant.
- 1: Tolerance is max-deviation of line from curve, in source units.
- 2: Tolerance is max-angle, in radians, between generating radii.

The 'flags' argument is a bitfield. 0 by default. Supported bits are:

- 1: Symmetric (orientation independent) output.
- 2: Retain angle, avoids reducing angles (segment lengths) when producing symmetric output. Has no effect when Symmetric flag is off.

Disponibilidade: 1.3.3

Enhanced: 2.4.0 added support for max-deviation and max-angle tolerance, and for symmetric output.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 7.1.7



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

## Exemplos

```
SELECT ST_AsText(ST_CurveToLine(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)')));
```

--Result --

```
LINESTRING(220268 150415,220269.95064912 150416.539364228,220271.823415575 150418.17258804,220273.613787707 150419.895736857,
220275.317452352 150421.704659462,220276.930305234 150423.594998003,220278.448460847 150425.562198489,
220279.868261823 150427.60152176,220281.186287736 150429.708054909,220282.399363347 150431.876723113,
220283.50456625 150434.10230186,220284.499233914 150436.379429536,220285.380970099 150438.702620341,220286.147650624 150441.066277505,
220286.797428488 150443.464706771,220287.328738321 150445.892130112,220287.740300149 150448.342699654,
220288.031122486 150450.810511759,220288.200504713 150453.289621251,220288.248038775 150455.77405574,
220288.173610157 150458.257830005,220287.977398166 150460.734960415,220287.659875492 150463.199479347,
220287.221807076 150465.64544956,220286.664248262 150468.066978495,220285.988542259 150470.458232479,220285.196316903 150472.81345077,
220284.289480732 150475.126959442,220283.270218395 150477.39318505,220282.140985384 150479.606668057,
220280.90450212 150481.762075989,220279.5637474 150483.85421628,220278.12195122 150485.87804878,
220276.582586992 150487.828697901,220274.949363179 150489.701464356,220273.226214362 150491.491836488,
220271.417291757 150493.195501133,220269.526953216 150494.808354014,220267.559752731 150496.326509628,
220265.520429459 150497.746310603,220263.41389631 150499.064336517,220261.245228106 150500.277412127,
220259.019649359 150501.38261503,220256.742521683 150502.377282695,220254.419330878 150503.259018879,
220252.055673714 150504.025699404,220249.657244448 150504.675477269,220247.229821107 150505.206787101,
220244.779251566 150505.61834893,220242.311439461 150505.909171266,220239.832329968 150506.078553494,
```

```

220237.347895479 150506.126087555,220234.864121215 150506.051658938,220232.386990804 ↔
  150505.855446946,
220229.922471872 150505.537924272,220227.47650166 150505.099855856,220225.054972724 ↔
  150504.542297043,
220222.663718741 150503.86659104,220220.308500449 150503.074365683,
220217.994991777 150502.167529512,220215.72876617 150501.148267175,
220213.515283163 150500.019034164,220211.35987523 150498.7825509,
220209.267734939 150497.441796181,220207.243902439 150496,
220205.293253319 150494.460635772,220203.420486864 150492.82741196,220201.630114732 ↔
  150491.104263143,
220199.926450087 150489.295340538,220198.313597205 150487.405001997,220196.795441592 ↔
  150485.437801511,
220195.375640616 150483.39847824,220194.057614703 150481.291945091,220192.844539092 ↔
  150479.123276887,220191.739336189 150476.89769814,
220190.744668525 150474.620570464,220189.86293234 150472.297379659,220189.096251815 ↔
  150469.933722495,
220188.446473951 150467.535293229,220187.915164118 150465.107869888,220187.50360229 ↔
  150462.657300346,
220187.212779953 150460.189488241,220187.043397726 150457.710378749,220186.995863664 ↔
  150455.22594426,
220187.070292282 150452.742169995,220187.266504273 150450.265039585,220187.584026947 ↔
  150447.800520653,
220188.022095363 150445.35455044,220188.579654177 150442.933021505,220189.25536018 ↔
  150440.541767521,
220190.047585536 150438.18654923,220190.954421707 150435.873040558,220191.973684044 ↔
  150433.60681495,
220193.102917055 150431.393331943,220194.339400319 150429.237924011,220195.680155039 ↔
  150427.14578372,220197.12195122 150425.12195122,
220198.661315447 150423.171302099,220200.29453926 150421.298535644,220202.017688077 ↔
  150419.508163512,220203.826610682 150417.804498867,
220205.716949223 150416.191645986,220207.684149708 150414.673490372,220209.72347298 ↔
  150413.253689397,220211.830006129 150411.935663483,
220213.998674333 150410.722587873,220216.22425308 150409.61738497,220218.501380756 ↔
  150408.622717305,220220.824571561 150407.740981121,
220223.188228725 150406.974300596,220225.586657991 150406.324522731,220227 150406)

```

--3d example

```

SELECT ST_AsEWKT(ST_CurveToLine(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 ↔
  150505 2,220227 150406 3)')));

```

Output

-----

```

LINESTRING(220268 150415 1,220269.95064912 150416.539364228 1.0181172856673,
220271.823415575 150418.17258804 1.03623457133459,220273.613787707 150419.895736857 ↔
  1.05435185700189,...AD INFINITUM ....
  220225.586657991 150406.324522731 1.32611114201132,220227 150406 3)

```

--use only 2 segments to approximate quarter circle

```

SELECT ST_AsText(ST_CurveToLine(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 ↔
  150505,220227 150406)'),2));

```

st\_astext

-----

```

LINESTRING(220268 150415,220287.740300149 150448.342699654,220278.12195122 ↔
  150485.87804878,
220244.779251566 150505.61834893,220207.243902439 150496,220187.50360229 150462.657300346,
220197.12195122 150425.12195122,220227 150406)

```

Veja também.

[ST\\_LineToCurve](#)

### 8.11.8 ST\_DelaunayTriangles

ST\_DelaunayTriangles — Retorna uma triangulação de Delaunay em volta dos pontos de entrada.

#### Synopsis

```
geometry ST_DelaunayTriangles(geometry g1, float tolerance, int4 flags);
```

#### Descrição

Retorna uma **Delaunay triangulation** em volta dos vértices da geometria de entrada. A saída é uma COLEÇÃO de polígonos (para bandeiras=0) ou MULTILINESTRING (para bandeiras=1) ou TIN (para bandeiras=2). A tolerância é usada para romper vértices de entrada juntos.

Disponibilidade: 2.1.0 - requer GEOS >= 3.4.0.



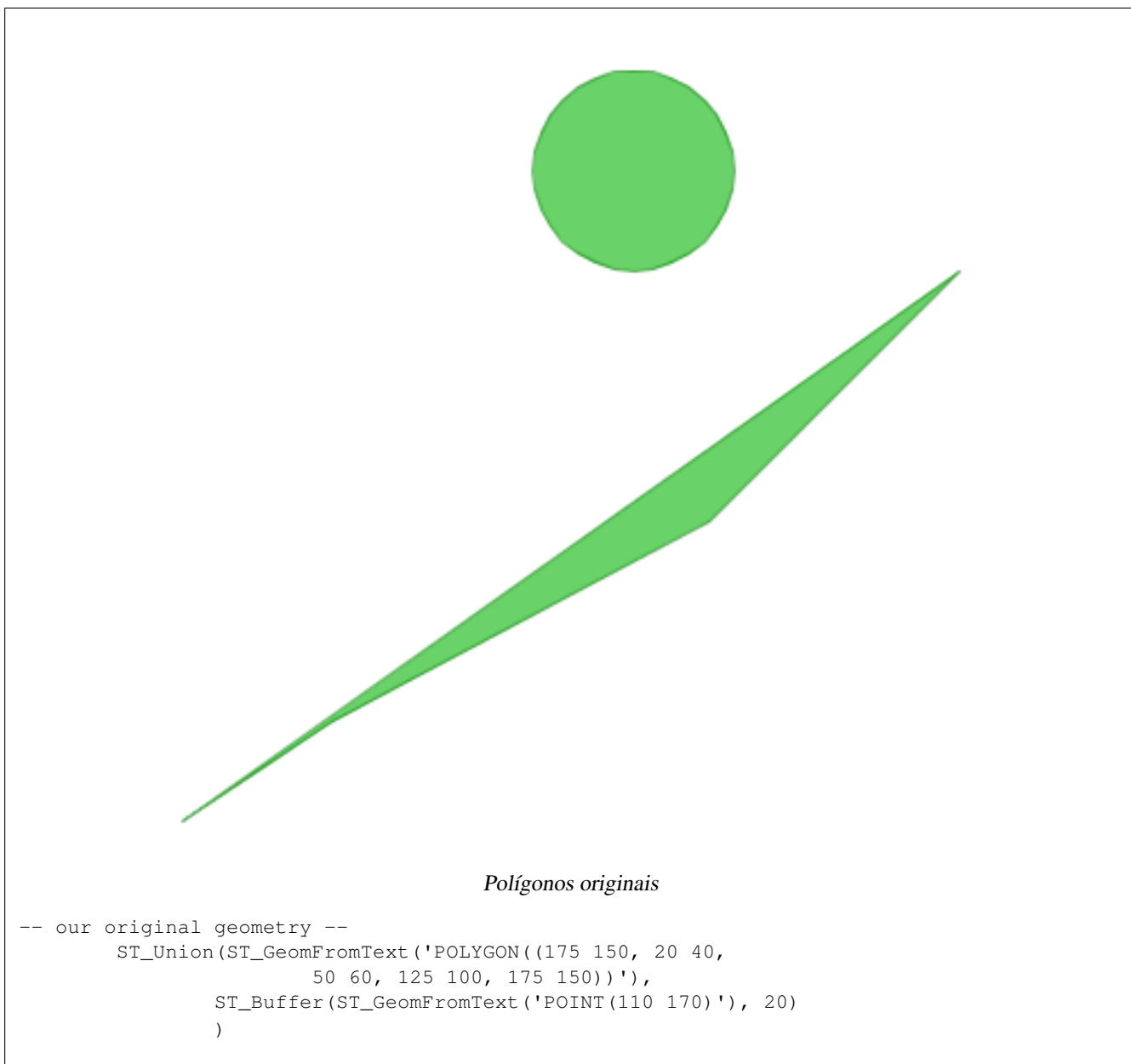
This function supports 3d and will not drop the z-index.

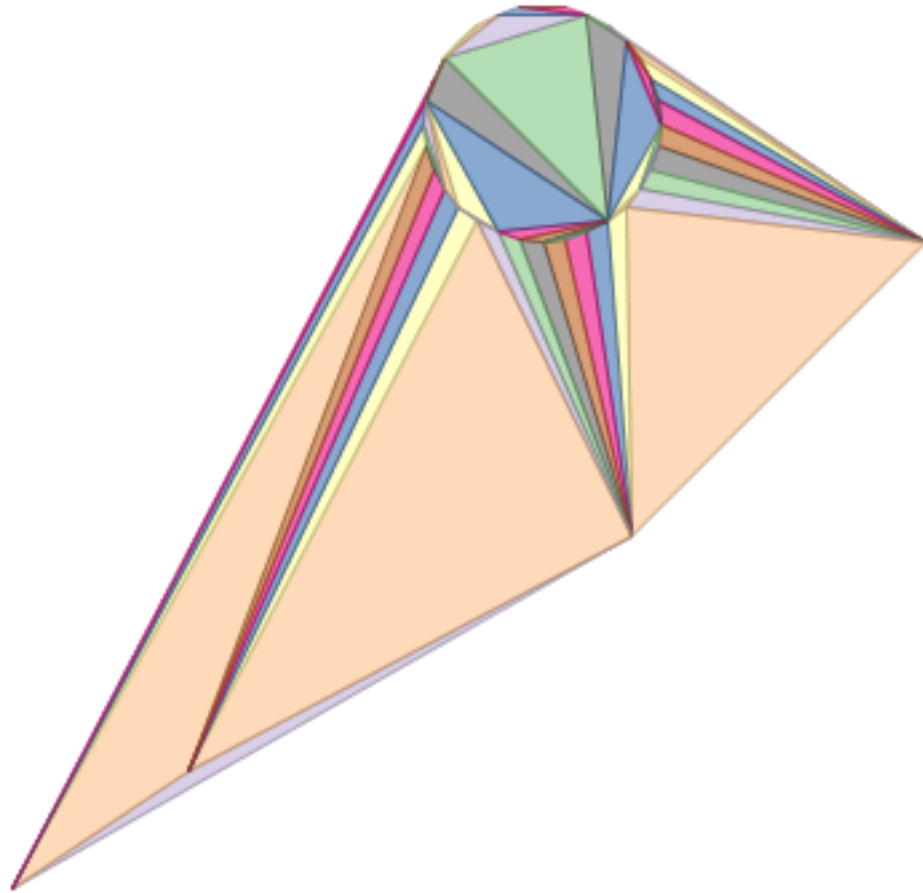


This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

#### Exemplos 2D

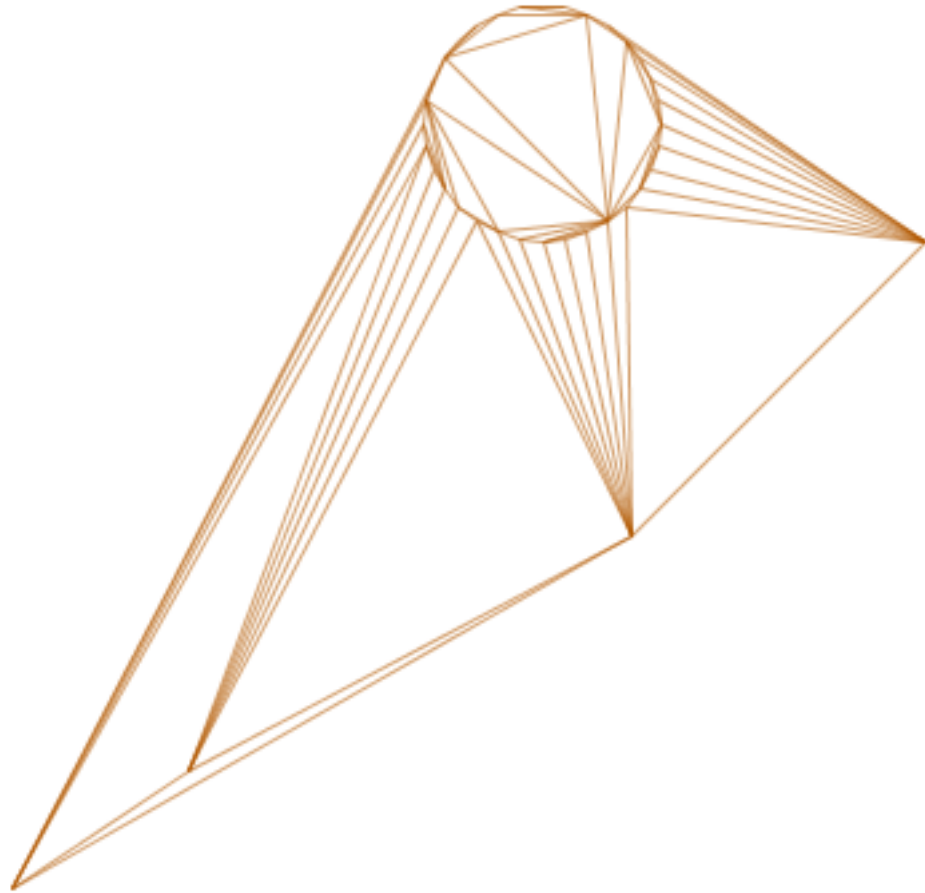
---





*ST\_DelaunayTriangles de 2 polígonos: polígonos da triangulação de Delaunay, cada triângulo de uma cor diferente*

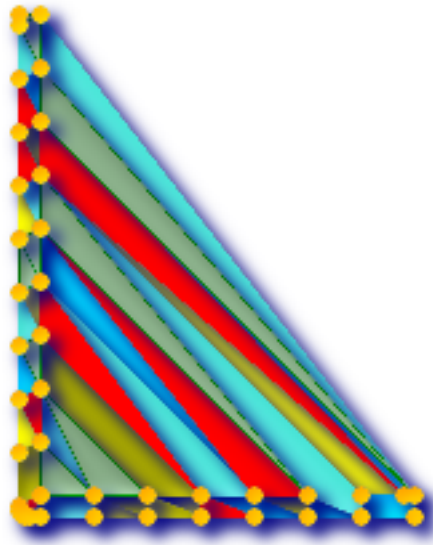
```
-- geometries overlaid multilinestring triangles
SELECT
  ST_DelaunayTriangles(
    ST_Union(ST_GeomFromText('POLYGON((175 150, 20 40,
      50 60, 125 100, 175 150))'),
    ST_Buffer(ST_GeomFromText('POINT(110 170)'), 20)
  )
  As dtriag;
```



*-- triângulos de Delaunay como multilinestring*

```
SELECT
  ST_DelaunayTriangles(
    ST_Union(ST_GeomFromText('POLYGON((175 150, 20 40,
      50 60, 125 100, 175 150))'),
    ST_Buffer(ST_GeomFromText('POINT(110 170)'), 20)
  ),0.001,1)
  As dtriag;
```





*-- triângulos delaunay de 45 pontos como 55 polígonos triangulares*

```
-- this produces a table of 42 points that form an L shape
SELECT (ST_DumpPoints(ST_GeomFromText (
'MULTIPOINT(14 14,34 14,54 14,74 14,94 14,114 14,134 14,
150 14,154 14,154 6,134 6,114 6,94 6,74 6,54 6,34 6,
14 6,10 6,8 6,7 7,6 8,6 10,6 30,6 50,6 70,6 90,6 110,6 130,
6 150,6 170,6 190,6 194,14 194,14 174,14 154,14 134,14 114,
14 94,14 74,14 54,14 34,14 14)'))).geom
      INTO TABLE l_shape;
-- output as individual polygon triangles
SELECT ST_AsText((ST_Dump(geom)).geom) As wkt
FROM ( SELECT ST_DelaunayTriangles(ST_Collect(geom)) As geom
FROM l_shape) As foo;

---wkt ---
POLYGON((6 194,6 190,14 194,6 194))
POLYGON((14 194,6 190,14 174,14 194))
POLYGON((14 194,14 174,154 14,14 194))
POLYGON((154 14,14 174,14 154,154 14))
POLYGON((154 14,14 154,150 14,154 14))
POLYGON((154 14,150 14,154 6,154 14))
:
:
```

### Exemplos 3D

```
-- 3D multipoint --
SELECT ST_AsText(ST_DelaunayTriangles(ST_GeomFromText (
'MULTIPOINT Z(14 14 10,
150 14 100,34 6 25, 20 10 150)')))) As wkt;

-----wkt-----
GEOMETRYCOLLECTION Z (POLYGON Z ((14 14 10,20 10 150,34 6 25,14 14 10))
,POLYGON Z ((14 14 10,34 6 25,150 14 100,14 14 10)))
```

Veja também.

[ST\\_ConcaveHull](#), [ST\\_Dump](#)

### 8.11.9 ST\_Difference

ST\_Difference — Retorna uma geometria que representa aquela parte de geometria A que não intersecta com a geometria B.

#### Synopsis

```
geometry ST_Difference(geometry geomA, geometry geomB);
```

#### Descrição

Retorna uma geometria que representa aquela parte de geometria A que não intersecta com a geometria B. Pode-se pensar nela como uma  $GeometryA - ST\_Intersection(A,B)$ . Se A está completamente contida em B, então uma coleção vazia de geometria retorna.



#### Note

Nota - questões de ordem. B - B sempre retornará uma porção de B

---

Desempenhado pelo módulo GEOS

---



#### Note

Não chama com uma GeometryCollection com um argumento

---



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3



This method implements the SQL/MM specification. SQL-MM 3: 5.1.20

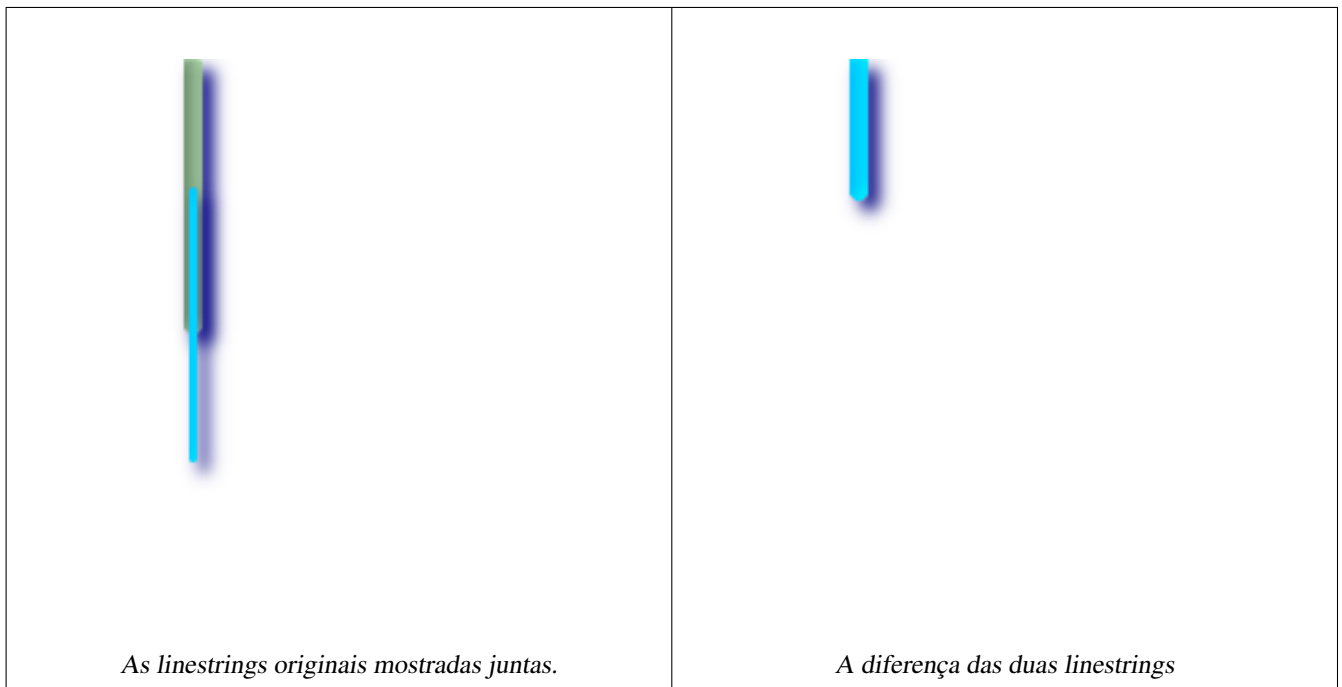


This function supports 3d and will not drop the z-index. Entretanto aparenta considerar apenas x y quando fazem diferença e retorna no Z-índice

#### Exemplos

---

---



```
--Safe for 2d. This is same geometries as what is shown for st_symdifference
SELECT ST_AsText(
  ST_Difference(
    ST_GeomFromText('LINESTRING(50 100, 50 200)'),
    ST_GeomFromText('LINESTRING(50 50, 50 150)')
  )
);
```

```
st_astext
-----
LINESTRING(50 150,50 200)
```

```
--When used in 3d doesn't quite do the right thing
SELECT ST_AsEWKT(ST_Difference(ST_GeomFromEWKT('MULTIPOINT(-118.58 38.38 5,-118.60 38.329 6,-118.614 38.281 7)'), ST_GeomFromEWKT('POINT(-118.614 38.281 5)')));
st_asewkt
-----
MULTIPOINT(-118.6 38.329 6,-118.58 38.38 5)
```

**Veja também.**

[ST\\_SymDifference](#)

### 8.11.10 ST\_Dump

**ST\_Dump** — Retorna um conjunto de filas `geometry_dump` (`geom,path`), que fazem uma geometria `g1`.

#### Synopsis

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

## Descrição

Esta é uma função set-returning (SRF). Ela retorna um conjunto de filas `geometry_dump`, formado por uma geometria (`geom`) e um arranjo de inteiros (caminho). Quando a geometria de entrada é um tipo simples (PONTO, LINESTRING, POLÍGONO), um único relato retornará com um arranjo caminho vazio e a geometria de entrada como `geom`. Quando a geometria de entrada for uma coleção ou multi, retornará um relato para cada um dos componentes da coleção, e o caminho vai expressar a posição do componente de dentro da coleção.

`ST_Dump` é útil para expandir geometrias. É o oposto de um GRUPO POR onde cria novas filas. Por exemplo, pode ser usada para expandir MULTIPOLÍGONOS dentro de POLÍGONOS.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.

Disponibilidade: PostGIS 1.0.0RC1. Requer PostgreSQL 7.3 ou superior.



### Note

Antes da versão 1.3.4, esta função falha se utilizada em geometrias que continham curvas. Resolvido nas versões 1.3.4+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.

## Exemplos Padrão

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

-- Break a compound curve into its constituent linestrings and circularstrings
SELECT ST_AsEWKT(a.geom), ST_HasArc(a.geom)
FROM ( SELECT (ST_Dump(p_geom)).geom AS geom
       FROM (SELECT ST_GeomFromEWKT('COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0), (1 0, 0
         1))') AS p_geom) AS b
       ) AS a;
  st_asewkt          | st_hasarc
-----+-----
CIRCULARSTRING(0 0,1 1,1 0) | t
LINESTRING(1 0,0 1)      | f
(2 rows)
```

## Exemplos de Superfícies Poliédricas, TIN e Triângulos

```
-- Polyhedral surface example
-- Break a Polyhedral surface into its faces
SELECT (a.p_geom).path[1] As path, ST_AsEWKT((a.p_geom).geom) As geom_ewkt
FROM (SELECT ST_Dump(ST_GeomFromEWKT('POLYHEDRALSURFACE(
((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)), ((1 1 0, 1 1
  1, 1 0 1, 1 0 0, 1 1 0)),
```

```
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1))
)') ) AS p_geom ) AS a;
```

path	geom_ewkt
1	POLYGON((0 0 0,0 0 1,0 1 1,0 1 0,0 0 0))
2	POLYGON((0 0 0,0 1 0,1 1 0,1 0 0,0 0 0))
3	POLYGON((0 0 0,1 0 0,1 0 1,0 0 1,0 0 0))
4	POLYGON((1 1 0,1 1 1,1 0 1,1 0 0,1 1 0))
5	POLYGON((0 1 0,0 1 1,1 1 1,1 1 0,0 1 0))
6	POLYGON((0 0 1,1 0 1,1 1 1,0 1 1,0 0 1))

```
-- TIN --
SELECT (g.gdump).path, ST_AsEWKT((g.gdump).geom) as wkt
FROM
  (SELECT
    ST_Dump( ST_GeomFromEWKT('TIN (((
      0 0 0,
      0 0 1,
      0 1 0,
      0 0 0
    )), ((
      0 0 0,
      0 1 0,
      1 1 0,
      0 0 0
    ))
    )') ) AS gdump
  ) AS g;
-- result --
path | wkt
-----+-----
{1} | TRIANGLE((0 0 0,0 0 1,0 1 0,0 0 0))
{2} | TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
```

### Veja também.

[geometry\\_dump](#), [Section 14.6](#), [ST\\_Collect](#), [ST\\_Collect](#), [ST\\_GeometryN](#)

### 8.11.11 ST\_DumpPoints

**ST\_DumpPoints** — Retorna um conjunto de filas `geometry_dump` (`geom,path`) de todos os pontos que fazem uma geometria.

#### Synopsis

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

#### Descrição

Esta função set-returning (SRF) retorna um conjunto de filas `geometry_dump` formadas por uma geometria (`geom`) e um arranjo de inteiros (`path`).

O componente `geom` do `geometry_dump` são todos os POINTs que fazem as geometrias fornecidas

O componente `path` do `geometry_dump` (um `integer[]`) é um índice referência enumerando os POINTs da geometria fornecida. Por exemplo, se uma `LINESTRING` é fornecida, um caminho de `{i}` é retornado onde `i` é a `n`th coordenada na

**LINestring**. Se um **POLYGON** é fornecido, um caminho de  $\{i, j\}$  é retornado onde  $i$  é o número anel (1 é de fora; anéis de dentro seguem) e  $j$  enumera o **POINTS** (novamente índice 1-based).

Melhorias: 2.1.0 Velocidade mais rápida. Reimplementada como native-C.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.

Disponibilidade: 1.5.0



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



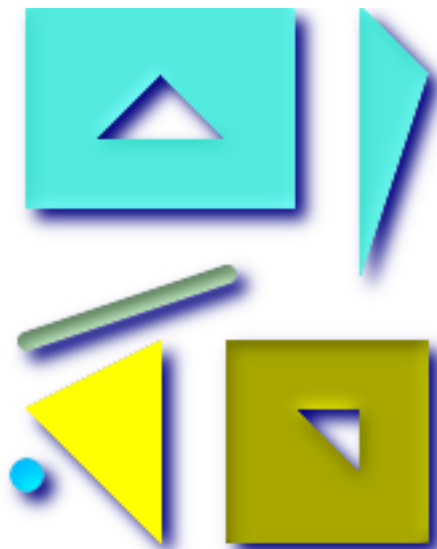
This function supports 3d and will not drop the z-index.

### Clássico Explode uma Tabela de LineStrings em nós

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

edge_id	index	wktnode
1	1	POINT(1 2)
1	2	POINT(3 4)
1	3	POINT(10 10)
2	1	POINT(3 5)
2	2	POINT(5 6)
2	3	POINT(9 10)

### Exemplos de geometria padrão



```

SELECT path, ST_AsText(geom)
FROM (
  SELECT (ST_DumpPoints(g.geom)).*
  FROM
    (SELECT
      'GEOMETRYCOLLECTION(
        POINT ( 0 1 ),
        LINESTRING ( 0 3, 3 4 ),
        POLYGON (( 2 0, 2 3, 0 2, 2 0 )),
        POLYGON (( 3 0, 3 3, 6 3, 6 0, 3 0 ),
          ( 5 1, 4 2, 5 2, 5 1 )),
        MULTIPOLYGON (
          (( 0 5, 0 8, 4 8, 4 5, 0 5 ),
            ( 1 6, 3 6, 2 7, 1 6 )),
            (( 5 4, 5 8, 6 7, 5 4 ))
          )
        )
      )::geometry AS geom
    ) AS g
  ) j;

```

path	st_astext
{1,1}	POINT(0 1)
{2,1}	POINT(0 3)
{2,2}	POINT(3 4)
{3,1,1}	POINT(2 0)
{3,1,2}	POINT(2 3)
{3,1,3}	POINT(0 2)
{3,1,4}	POINT(2 0)
{4,1,1}	POINT(3 0)
{4,1,2}	POINT(3 3)
{4,1,3}	POINT(6 3)
{4,1,4}	POINT(6 0)
{4,1,5}	POINT(3 0)
{4,2,1}	POINT(5 1)
{4,2,2}	POINT(4 2)
{4,2,3}	POINT(5 2)
{4,2,4}	POINT(5 1)
{5,1,1,1}	POINT(0 5)
{5,1,1,2}	POINT(0 8)
{5,1,1,3}	POINT(4 8)
{5,1,1,4}	POINT(4 5)
{5,1,1,5}	POINT(0 5)
{5,1,2,1}	POINT(1 6)
{5,1,2,2}	POINT(3 6)
{5,1,2,3}	POINT(2 7)
{5,1,2,4}	POINT(1 6)
{5,2,1,1}	POINT(5 4)
{5,2,1,2}	POINT(5 8)
{5,2,1,3}	POINT(6 7)
{5,2,1,4}	POINT(5 4)

(29 rows)

### Exemplos de Superfícies Poliédricas, TIN e Triângulos

```

-- Polyhedral surface cube --
SELECT (g.gdump).path, ST_AsEWKT((g.gdump).geom) as wkt
FROM
  (SELECT

```

```

        ST_DumpPoints(ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 ←
        0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )') ) AS gdump
    ) AS g;
-- result --
 path | wkt
-----+-----
{1,1,1} | POINT(0 0 0)
{1,1,2} | POINT(0 0 1)
{1,1,3} | POINT(0 1 1)
{1,1,4} | POINT(0 1 0)
{1,1,5} | POINT(0 0 0)
{2,1,1} | POINT(0 0 0)
{2,1,2} | POINT(0 1 0)
{2,1,3} | POINT(1 1 0)
{2,1,4} | POINT(1 0 0)
{2,1,5} | POINT(0 0 0)
{3,1,1} | POINT(0 0 0)
{3,1,2} | POINT(1 0 0)
{3,1,3} | POINT(1 0 1)
{3,1,4} | POINT(0 0 1)
{3,1,5} | POINT(0 0 0)
{4,1,1} | POINT(1 1 0)
{4,1,2} | POINT(1 1 1)
{4,1,3} | POINT(1 0 1)
{4,1,4} | POINT(1 0 0)
{4,1,5} | POINT(1 1 0)
{5,1,1} | POINT(0 1 0)
{5,1,2} | POINT(0 1 1)
{5,1,3} | POINT(1 1 1)
{5,1,4} | POINT(1 1 0)
{5,1,5} | POINT(0 1 0)
{6,1,1} | POINT(0 0 1)
{6,1,2} | POINT(1 0 1)
{6,1,3} | POINT(1 1 1)
{6,1,4} | POINT(0 1 1)
{6,1,5} | POINT(0 0 1)
(30 rows)

```

```

-- Triangle --
SELECT (g.gdump).path, ST_AsText((g.gdump).geom) as wkt
FROM
  (SELECT
    ST_DumpPoints( ST_GeomFromEWKT('TRIANGLE ((
      0 0,
      0 9,
      9 0,
      0 0
    ))') ) AS gdump
  ) AS g;
-- result --
 path | wkt
-----+-----
{1} | POINT(0 0)
{2} | POINT(0 9)
{3} | POINT(9 0)
{4} | POINT(0 0)

```

```

-- TIN --

```



```

SELECT (g.gdump).path, ST_AsEWKT((g.gdump).geom) as wkt
FROM
  (SELECT
    ST_DumpPoints( ST_GeomFromEWKT('TIN (((
      0 0 0,
      0 0 1,
      0 1 0,
      0 0 0
    ))), ((
      0 0 0,
      0 1 0,
      1 1 0,
      0 0 0
    ))
  ) AS g;
-- result --
 path | wkt
-----+-----
{1,1,1} | POINT(0 0 0)
{1,1,2} | POINT(0 0 1)
{1,1,3} | POINT(0 1 0)
{1,1,4} | POINT(0 0 0)
{2,1,1} | POINT(0 0 0)
{2,1,2} | POINT(0 1 0)
{2,1,3} | POINT(1 1 0)
{2,1,4} | POINT(0 0 0)
(8 rows)

```

**Veja também.**

[geometry\\_dump](#), Section 14.6, [ST\\_Dump](#), [ST\\_DumpRings](#)

**8.11.12 ST\_DumpRings**

**ST\_DumpRings** — Retorna um conjunto de filas `geometry_dump`, representando os anéis interiores e exteriores de um polígono.

**Synopsis**

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

**Descrição**

Esta é uma função set-returning (SRF). Ela retorna um conjunto de filas `geometry_dump`, definidas como um `integer[]` e uma `geometry`, pseudônimos "caminho" e "geom" respectivamente. o campo "caminho" detém o anel índice do polígono que contém um único inteiro: 0 para shell, >0 para buracos. O caminho "geom" contém o anel correspondente como um polígono.

Disponibilidade: PostGIS 1.1.3. Requer PostgreSQL 7.3 ou superior.

**Note**

Funciona somente para POLÍGONOS. Não irá funcionar para MULTIPOLÍGONOS



This function supports 3d and will not drop the z-index.

## Exemplos

```
SELECT sometable.field1, sometable.field1,
       (ST_DumpRings(sometable.the_geom)).geom As the_geom
FROM sometableOfpolys;

SELECT ST_AsEWKT(geom) As the_geom, path
FROM ST_DumpRings(
  ST_GeomFromEWKT('POLYGON((-8149064 5133092 1,-8149064 5132986 1,-8148996 ↵
    5132839 1,-8148972 5132767 1,-8148958 5132508 1,-8148941 5132466 ↵
    1,-8148924 5132394 1,
    -8148903 5132210 1,-8148930 5131967 1,-8148992 5131978 1,-8149237 5132093 ↵
    1,-8149404 5132211 1,-8149647 5132310 1,-8149757 5132394 1,
    -8150305 5132788 1,-8149064 5133092 1),
    (-8149362 5132394 1,-8149446 5132501 1,-8149548 5132597 1,-8149695 5132675 ↵
    1,-8149362 5132394 1))')
  ) as foo;
```

path	the_geom
{0}	POLYGON((-8149064 5133092 1,-8149064 5132986 1,-8148996 5132839 1,-8148972 5132767 ↵ 1,-8148958 5132508 1,   -8148941 5132466 1,-8148924 5132394 1,   -8148903 5132210 1,-8148930 5131967 1,   -8148992 5131978 1,-8149237 5132093 1,   -8149404 5132211 1,-8149647 5132310 1,-8149757 5132394 1,-8150305 ↵ 5132788 1,-8149064 5133092 1))
{1}	POLYGON((-8149362 5132394 1,-8149446 5132501 1,   -8149548 5132597 1,-8149695 5132675 1,-8149362 5132394 1))

### Veja também.

[geometry\\_dump](#), [Section 14.6](#), [ST\\_Dump](#), [ST\\_ExteriorRing](#), [ST\\_InteriorRingN](#)

### 8.11.13 ST\_FlipCoordinates

**ST\_FlipCoordinates** — Retorna uma versão da geometria dada com os eixos X e Y virados. Útil para pessoas que construíram características latitude/longitude e precisam consertá-las.

#### Synopsis

```
geometry ST_FlipCoordinates(geometry geom);
```

#### Descrição

Retorna uma versão da geometria dada com os eixos X e Y virados.

Disponibilidade: 2.0.0



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.



This function supports M coordinates.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### Exemplo

```
SELECT ST_AsEWKT(ST_FlipCoordinates(GeomFromEWKT('POINT(1 2)')));
 st_asewkt
-----
POINT(2 1)
```

### Veja também.

[ST\\_SwapOrdinates](#)

## 8.11.14 ST\_GeneratePoints

**ST\_GeneratePoints** — Converte um polígono ou multi-polígono para um multi-ponto composto por pontos de localização aleatórios dentro das áreas originais.

### Synopsis

geometry **ST\_GeneratePoints**( g geometry , npoints numeric );

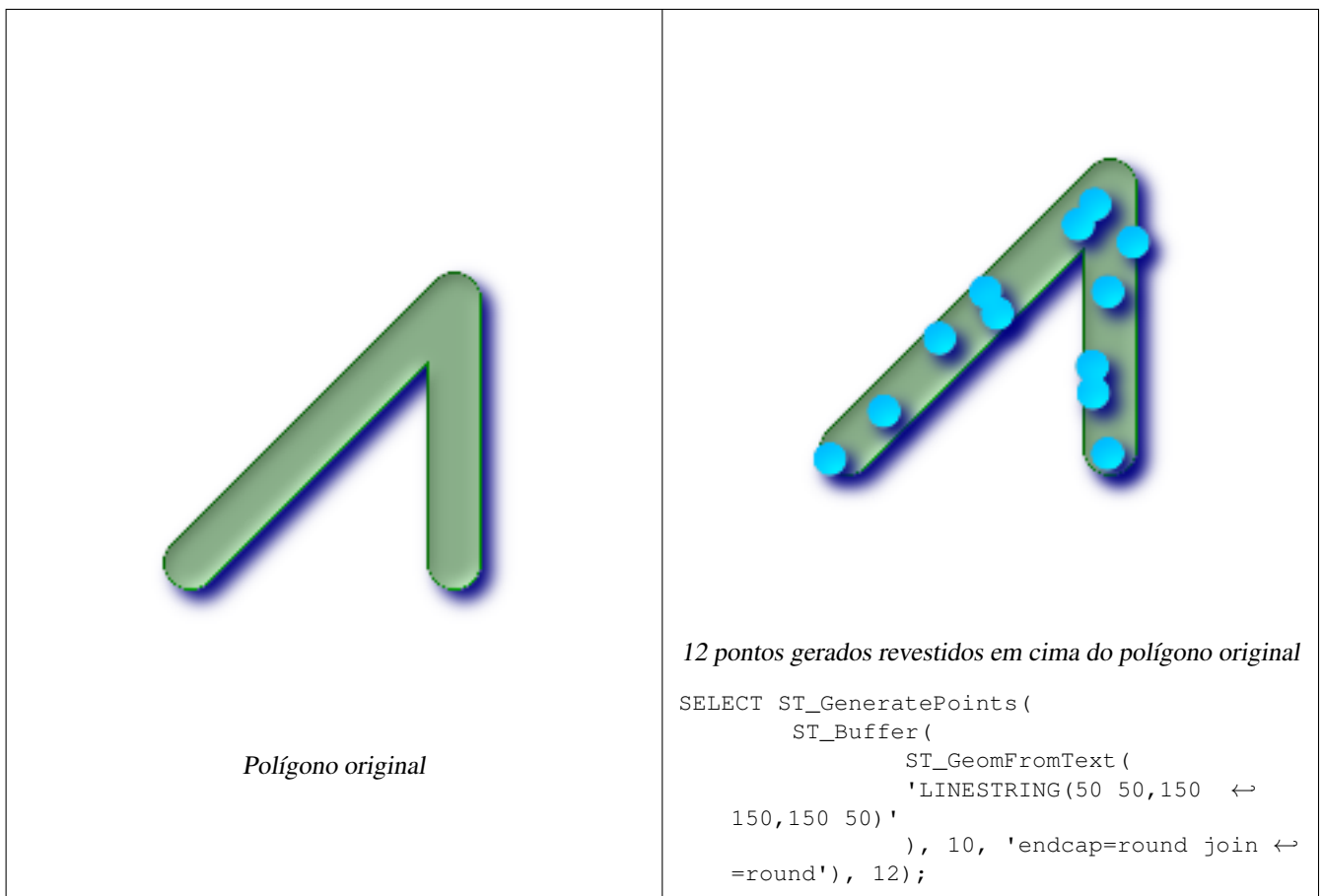
### Descrição

**ST\_GeneratePoints** gera pontos pseudo aleatórios antes do número solicitado são encontrados dentro da área de entrada.

Disponibilidade: 2.3.0

### Exemplos

---



### 8.11.15 ST\_Intersection

**ST\_Intersection** — (T) Retorna uma geometria que representa a porção dividida da geomA e geomB.

#### Synopsis

```
geometry ST_Intersection( geometry geomA , geometry geomB );
geography ST_Intersection( geography geogA , geography geogB );
```

#### Descrição

Retorna uma geometria que representa a interseção de pontos das Geometrias.

Em outras palavras - aquela porção de geometria A e geometria B que é dividido entre as duas geometrias.

Se as geometrias não dividem nenhum espaço (são disjuntas), uma coleção vazia de geometrias retorna.

A **ST\_Intersection**, em conjunto com a **ST\_Intersects**, é bastante útil para aparar geometrias como na caixa limitadora, buffer, consultas de região onde você só quer retornar aquela porção de uma geometria que se encaixa em um país ou região de interesse.

#### Note



Geografia: Para geografia isto é um envoltório bastante fino em torno da implementação geométrica. Ela determina, primeiramente, o melhor SRID que se encaixa na caixa limitadora dos 2 objetos geográficos (se eles estão dentro de meia zona UTM, mas não a mesma UTM escolherá um deles) (favorecendo UTM ou Lambert Azimuthal Equal Area (LAEA) polo norte/sul, e caindo no pior cenário mercator) e, então, interseção naquela referência plana que se encaixa melhor e transforma novamente de volta para a geografia WGS84.

**Important**

Não chama com uma `GEOMETRYCOLLECTION` como um argumento

**Warning**

This function will drop the M coordinate values if present.

**Warning**

Se você estiver trabalhando com geometrias 3D, talvez queira usar SFCGAL baseado [ST\\_3DIntersection](#) que faz uma interseção 3D apropriada para as geometrias 3D. Embora esta função funcione com a coordenada-Z, ela faz uma média dos valores de coordenada-Z quando `postgis.backend=geos`. `postgis.backend=sfcgal`, retornará uma geometria 2D de qualquer forma, ignorando a coordenada-Z. Recorra a [postgis.backend](#) para mais detalhes.

Desempenhado pelo módulo GEOS



This method is also provided by SFCGAL backend.

Disponibilidade: 1.5 suporte para tipo de dados de geografia foi introduzido.



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3



This method implements the SQL/MM specification. SQL-MM 3: 5.1.18

**Exemplos**

```
SELECT ST_AsText(ST_Intersection('POINT(0 0)::geometry, 'LINESTRING ( 2 0, 0 2 )':: ←
  geometry));
  st_astext
-----
GEOMETRYCOLLECTION EMPTY
(1 row)
SELECT ST_AsText(ST_Intersection('POINT(0 0)::geometry, 'LINESTRING ( 0 0, 0 2 )':: ←
  geometry));
  st_astext
-----
POINT(0 0)
(1 row)

---Clip all lines (trails) by country (here we assume country geom are POLYGON or ←
  MULTIPOLYGONS)
-- NOTE: we are only keeping intersections that result in a LINESTRING or MULTILINESTRING ←
  because we don't
-- care about trails that just share a point
-- the dump is needed to expand a geometry collection into individual single MULT* parts
-- the below is fairly generic and will work for polys, etc. by just changing the where ←
  clause
SELECT clipped.gid, clipped.f_name, clipped_geom
FROM (SELECT trails.gid, trails.f_name, (ST_Dump(ST_Intersection(country.the_geom, trails. ←
  the_geom))).geom As clipped_geom
FROM country
```

```

INNER JOIN trails
ON ST_Intersects(country.the_geom, trails.the_geom) As clipped
WHERE ST_Dimension(clipped.clipped_geom) = 1 ;

--For polys e.g. polygon landmarks, you can also use the sometimes faster hack that ↔
  buffering anything by 0.0
-- except a polygon results in an empty geometry collection
--(so a geometry collection containing polys, lines and points)
-- buffered by 0.0 would only leave the polygons and dissolve the collection shell
SELECT poly.gid, ST_Multi(ST_Buffer(
                                ST_Intersection(country.the_geom, poly.the_geom),
                                0.0)
                            ) As clipped_geom
FROM country
INNER JOIN poly
ON ST_Intersects(country.the_geom, poly.the_geom)
WHERE Not ST_IsEmpty(ST_Buffer(ST_Intersection(country.the_geom, poly.the_geom) ↔
,0.0));

```

### Exemplos: 2.5Dish

Geos é o backend padrão se não definido. Note que isto não é uma interseção verdadeira, compare ao mesmo exemplo usando: [ST\\_3DIntersection](#).

```

set postgis.backend=geos;
select ST_AsText(ST_Intersection(linestring, polygon)) As wkt
from ST_GeomFromText('LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)') AS ↔
  linestring
CROSS JOIN ST_GeomFromText('POLYGON((0 0 8, 0 1 8, 1 1 8, 1 0 8, 0 0 8))') AS polygon;

          st_astext
-----
LINESTRING Z (1 1 8,0.5 0.5 8,0 0 10)

```

Se seu PostGIS estiver compilado com o suporte sfcgal, tem opção de usar sfcgal, mas note se casos básicos abaixo das geometrias para 2D antes de fazer a interseção e retorna o resultado equivalente da ST\_Force2D, a qual é uma geometria 2D

```

set postgis.backend=sfcgal;
select ST_AsText(ST_Intersection(linestring, polygon)) As wkt
from ST_GeomFromText('LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)') AS ↔
  linestring
CROSS JOIN ST_GeomFromText('POLYGON((0 0 8, 0 1 8, 1 1 8, 1 0 8, 0 0 8))') AS polygon;

          wkt
-----
MULTILINESTRING((0.5 0.5,0 0),(1 1,0.5 0.5))

```

### Veja também.

[ST\\_3DIntersection](#), [ST\\_Difference](#), [ST\\_Dimension](#), [ST\\_Dump](#), [ST\\_Force2D](#), [ST\\_SymDifference](#), [ST\\_Intersects](#), [ST\\_Multi](#)

## 8.11.16 ST\_LineToCurve

ST\_LineToCurve — Converte uma LINESTRING/POLYGON para um CIRCULARSTRING, CURVEPOLYGON

### Synopsis

geometry **ST\_LineToCurve**(geometry geomANoncircular);

**Descrição**

Converte LINESTRING/POLÍGONOS simples para STRINGs CIRCULARES e Polígonos Curvados. Observe que pouquíssimos pontos são necessários para descrever a curva equivalente.



**Note**

If the input LINESTRING/POLYGON is not curved enough to clearly represent a curve, the function will return the same input geometry.

Disponibilidade: 1.3.3



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

**Exemplos**

```
SELECT ST_AsText(ST_LineToCurve(foo.the_geom)) As curvedastext,ST_AsText(foo.the_geom) As
non_curvedastext
FROM (SELECT ST_Buffer('POINT(1 3)::geometry, 3) As the_geom) As foo;
```

curvedastext	non_curvedastext
CURVEPOLYGON(CIRCULARSTRING(4 3,3.12132034355964 0.878679656440359,   POLYGON((4	3,3.94235584120969 2.41472903395162,3.77163859753386 1.85194970290473,
1 0,-1.12132034355965 5.12132034355963,4 3))	3.49440883690764
1.33328930094119,3.12132034355964 0.878679656440359,	2.66671069905881
	0.505591163092366,2.14805029
	0.228361402466141,
	1.58527096604839
	0.0576441587903094,1
	0,
	0.414729033951621
	0.0576441587903077,-0.1480502
	0.228361402466137,
	-0.666710699058802
	0.505591163092361,-1.12132034
	0.878679656440353,
	-1.49440883690763
	1.33328930094119,-1.77163859
	1.85194970290472
	--ETC--
	,3.94235584120969
	3.58527096604839,4
	3))

```
--3D example
SELECT ST_AsEWKT(ST_LineToCurve(ST_GeomFromEWKT('LINESTRING(1 2 3, 3 4 8, 5 6 4, 7 8 4, 9
10 4)')));
```

st_asewkt
CIRCULARSTRING(1 2 3,5 6 4,9 10 4)

**Veja também.**

[ST\\_CurveToLine](#)

### 8.11.17 ST\_MakeValid

ST\_MakeValid — Tenta tornar uma geometria inválida válida sem perder vértices.

#### Synopsis

```
geometry ST_MakeValid(geometry input);
```

#### Descrição

A função tenta criar uma representação válida de uma geometria dada sem perder nenhum dos vértices de entrada. Geometrias já validadas retornam sem mais intervenções.

As entradas suportadas são: PONTOS, MULTIPONTOS, LINESTRINGS, MULTILINESTRINGS, POLÍGONOS, MULTIPOLÍGONOS e GEOMETRYCOLLECTIONS que contenham qualquer mistura deles.

Em caso de colapsos completos ou parciais, a geometria de saída pode ser uma coleção de geometrias de dimensão inferior a igual ou uma geometria de dimensão inferior.

Polígonos únicos podem se tornar multi-geometrias no caso de auto-intersecções.

Disponibilidade: 2.0.0, requer GEOS-3.3.0

Melhorias: 2.0.1, melhorias na velocidade requer GEOS-3.3.4

Melhorias: 2.1.0 suporte para GEOMETRYCOLLECTION e MULTIPONTO adicionado.



This function supports 3d and will not drop the z-index.

**Veja também.**

[ST\\_IsValid](#) [ST\\_CollectionExtract](#)

### 8.11.18 ST\_MemUnion

ST\_MemUnion — O mesmo que ST\_Union, somente memória amigável (utiliza menos memória e mais tempo de processador).

#### Synopsis

```
geometry ST_MemUnion(geometry set geomfield);
```

#### Descrição

Alguma descrição útil aqui.



#### Note

O mesmo que ST\_Union, somente memória amigável (utiliza menos memória e mais tempo de processador). Esta função agregada trabalha unindo uma geometria de cada vez para resultados anteriores como opostos a ST\_Union agregada, a qual primeiro cria um arranjo e então uniões



This function supports 3d and will not drop the z-index.



## Exemplos

See `ST_Union`

## Veja também.

[ST\\_Union](#)

### 8.11.19 ST\_MinimumBoundingCircle

`ST_MinimumBoundingCircle` — Retorna o menor polígono circular que pode conter uma geometria. O padrão usa 48 segmentos por quarto de círculo.

#### Synopsis

```
geometry ST_MinimumBoundingCircle(geometry geomA, integer num_segs_per_qt_circ=48);
```

#### Descrição

retorna o menor polígono circular que pode conter uma geometria.

---

#### Note



O círculo é aproximado por um polígono com um padrão de 48 segmentos por quarto de círculo. Devido ao polígono ser uma aproximação do círculo delimitador, alguns pontos na geometria de entrada podem não estar contidos dentro do polígono. A aproximação pode ser melhorada pelo aumento do número de segmentos, com uma pequena penalidade de desempenho. Para aplicações nas quais uma aproximação poligonal não se encaixa, a `ST_MinimumBoundingRadius` pode ser usada.

---

Regularmente usado com `MULTI` e Coleções de Geometrias. Embora não seja um agregado - você pode usar em conjunto com a `ST_Collect` para obter o menor círculo delimitador de um conjunto de geometrias. `ST_MinimumBoundingCircle(ST_Collect(somepoint...`

A razão da área de um polígono dividido pela área do seu menor círculo delimitador é referenciada com o teste Roeck.

Disponibilidade: 1.4.0 - requer GEOS

## Veja também.

[ST\\_Collect](#), [ST\\_MinimumBoundingRadius](#)

## Exemplos

```
SELECT d.disease_type,  
       ST_MinimumBoundingCircle(ST_Collect(d.the_geom)) As the_geom  
FROM disease_obs As d  
GROUP BY d.disease_type;
```



*Menor círculo delimitador de um ponto e linestring. Utilizando 8 segmentos para aproximar um quarto de círculo*

```
SELECT ST_AsText(ST_MinimumBoundingCircle(
    ST_Collect(
        ST_GeomFromEWKT('LINESTRING(55 75,125 150)'),
        ST_Point(20, 80)), 8
    )) As wktmbc;

wktmbc
-----
POLYGON((135.59714732062 115,134.384753327498 102.690357210921,130.79416296937 90.8537670908995,124.963360620072 79.9451031602111,117.116420743937 70.3835792560632,107.554896839789 62.5366393799277,96.6462329091006 56.70583703063,84.8096427890789 53.115246672502,72.5000000000001 51.9028526793802,60.1903572109213 53.1152466725019,48.3537670908996 56.7058370306299,37.4451031602112 62.5366393799276,27.8835792560632 70.383579256063,20.0366393799278 79.9451031602109,14.20583703063 90.8537670908993,10.615246672502 102.690357210921,9.40285267938019 115,10.6152466725019 127.309642789079,14.2058370306299 139.1462329091,20.0366393799275 150.054896839789,27.883579256063 159.616420743937,37.4451031602108 167.463360620072,48.3537670908992 173.29416296937,60.190357210921 176.884753327498,72.4999999999998 178.09714732062,84.8096427890786 176.884753327498,96.6462329091003 173.29416296937,107.554896839789 167.463360620072,117.116420743937 159.616420743937,124.963360620072 150.054896839789,130.79416296937 139.146232909101,134.384753327498 127.309642789079,135.59714732062 115))
```

**Veja também.**

[ST\\_Collect](#), [ST\\_MinimumBoundingRadius](#)

### 8.11.20 ST\_MinimumBoundingRadius

`ST_MinimumBoundingRadius` — Retorna o ponto central e o alcance do menor círculo que pode conter uma geometria.

#### Synopsis

(geometry, double precision) `ST_MinimumBoundingRadius`(geometry geom);

**Descrição**

Retorna um relato contendo o ponto central e o alcance do menor círculo que pode conter uma geometria.

Pode ser usado em conjunto com [ST\\_Collect](#) para obter o menor círculo delimitador de um conjunto de geometrias.

Disponibilidade - 2.3.0

**Veja também.**

[ST\\_Collect](#), [ST\\_MinimumBoundingCircle](#)

**Exemplos**

```
SELECT ST_AsText(center), radius FROM ST_MinimumBoundingRadius('POLYGON((26426 65078,26531 ←
  65242,26075 65136,26096 65427,26426 65078))');
```

st_astext	radius
POINT(26284.8418027133 65267.1145090825)	247.436045591407

**8.11.21 ST\_OrientedEnvelope**

[ST\\_OrientedEnvelope](#) — Retorna uma coleção de geometrias resultante da divisão de uma geometria.

**Synopsis**

geometry [ST\\_Node](#)(geometry geom);

**Descrição**

Returns a minimum rotated rectangle enclosing a geometry. Note that more than one minimum rotated rectangle may exist. May return a Point or LineString in the case of degenerate inputs.

Disponibilidade: 2.0.0

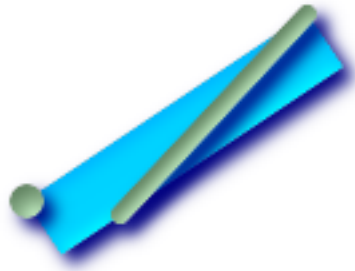
**Veja também.**

[ST\\_Envelope](#) [ST\\_MinimumBoundingCircle](#)

**Exemplos**

```
SELECT ST_AsText(ST_OrientedEnvelope('MULTIPOINT ((0 0), (-1 -1), (3 2))'));
```

st_astext
POLYGON((3 2,2.88 2.16,-1.12 -0.84,-1 -1,3 2))



*Oriented envelope of a point and linestring.*

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

wktenv
-----
POLYGON((19.9999999999997 79.9999999999999,33.0769230769229 ↔
        60.3846153846152,138.076923076924 130.384615384616,125.000000000001 ↔
        150.000000000001,19.9999999999997 79.9999999999999))
```

### 8.11.22 ST\_Polygonize

**ST\_Polygonize** — Agregado. Cria uma GeometryCollection contendo possíveis polígonos formados a partir da linework constituente de um conjunto de geometrias.

#### Synopsis

```
geometry ST_Polygonize(geometry set geomfield);
geometry ST_Polygonize(geometry[] geom_array);
```

#### Descrição

Cria uma GeometryCollection contendo possíveis polígonos formados a partir da linework constituente de um conjunto de geometrias.



#### Note

As Coleções de Geometria são difíceis de lidar, normalmente, com ferramentas de terceira parte, então use a **ST\_Polygonize** com **ST\_Dump** para descarregar os polígonos em polígonos individuais.



#### Note

A linework de entrada deve ser nodificada corretamente para esta função trabalhar normalmente.

Disponibilidade: 1.0.0RC1 - requer GEOS >= 2.1.0.

### Exemplos: Poligonizando linestrings únicas

```
SELECT ST_AsEWKT(ST_Polygonize(the_geom_4269)) As geomtextrep
FROM (SELECT the_geom_4269 FROM ma.suffolk_edges ORDER BY tlid LIMIT 45) As foo;

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
SELECT ST_AsEWKT((ST_Dump(foofoo.polycoll)).geom) As geomtextrep
FROM (SELECT ST_Polygonize(the_geom_4269) As polycoll
      FROM (SELECT the_geom_4269 FROM ma.suffolk_edges
            ORDER BY tlid LIMIT 45) As foo) As foofoo;

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)
```

### Veja também.

[ST\\_Node](#), [ST\\_Dump](#)

## 8.11.23 ST\_Node

ST\_Node — Nodar um conjunto de linestrings.

### Synopsis

geometry **ST\_Node**(geometry geom);

### Descrição

Nodar completamente um conjunto de linestrings usando o menor número de nós possíveis enquanto preserva todos os de entrada.



This function supports 3d and will not drop the z-index.

Disponibilidade: 2.0.0 - requer GEOS >= 3.3.0.



#### Note

Devido a um bug no GEOS 3.3.1, esta função falha em nodar linhas que se auto intersectam. Ele é consertado no GEOS 3.3.2 ou superior.

**Note**

Changed: 2.4.0 this function uses GEOSNode internally instead of GEOSUnaryUnion. This may cause the resulting linestrings to have a different order and direction compared to Postgis < 2.4.

**Exemplos**

```
SELECT ST_AsEWKT(
    ST_Node('LINESTRINGZ(0 0 0, 10 10 10, 0 10 5, 10 0 3)::geometry')
) As output;
output
-----
MULTILINESTRING((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))
```

**Veja também.**

[ST\\_UnaryUnion](#)

**8.11.24 ST\_OffsetCurve**

**ST\_OffsetCurve** — Retorna uma linha deslocada em uma dada distância e lado de uma linha de entrada. Útil para calcular linhas paralelas sobre uma linha central

**Synopsis**

geometry **ST\_OffsetCurve**(geometry line, float signed\_distance, text style\_parameters=’');

**Descrição**

Retorna uma linha deslocada em uma dada distância e ponto de uma linha de entrada. Todos os pontos das geometrias retornadas não são mais que a dada distância da geometria de entrada.

Para distância positiva, o deslocamento será do lado esquerdo da linha de entrada e mantida na mesma direção. Para distância negativa, ele será no lado direito e na direção oposta.

Disponibilidade: 2.0 - requer GEOS >= 3.2, melhorado com GEOS >= 3.3

Melhorias: 2.1.0 suporte para GEOMETRYCOLLECTION e MULTIPONTO adicionado.

O terceiro parâmetro opcional permite especificar uma lista de chave=valor em branco separados em pares para ajustar operações como segue:

- 'quad\_segs=#' : número de segmentos usado para aproximar um quarto de círculo (leva a 8).
- 'join=round|mitre|bevel' : join style (padrão para "round"). 'miter' também é aceitado como sinônimo de 'mitre'.
- 'mitre\_limit=#.#' : mitre ratio limit (só afeta o estilo mitred join). 'miter\_limit' também é aceito como sinônimo para 'mitre\_limit'.

Unidades de distância são medidas em unidades do sistema de referência espacial.

Desempenhado pelo módulo GEOS.

**Note**

Esta função ignora a terceira dimensão (z) e sempre oferece um resultado 2-d mesmo com uma geometria 3-d presente.

## Exemplos

### Calcula um buffer aberto em volta das ruas

```
SELECT ST_Union(
  ST_OffsetCurve(f.the_geom, f.width/2, 'quad_segs=4 join=round'),
  ST_OffsetCurve(f.the_geom, -f.width/2, 'quad_segs=4 join=round')
) as track
FROM someroadstable;
```



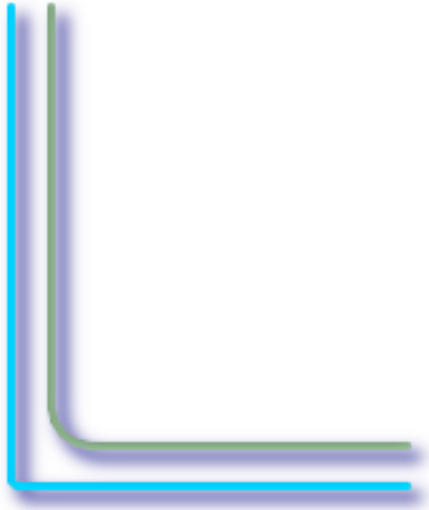
15, 'quad\_segs=4 join=round' original line and its offset 15 units.

```
SELECT ST_AsText(ST_OffsetCurve(↵
  ST_GeomFromText(↵
'LINESTRING(164 16,144 16,124 16,104 ↵
  16,84 16,64 16,↵
    44 16,24 16,20 16,18 16,17 17,↵
    16 18,16 20,16 40,16 60,16 80,16 ↵
  100,↵
    16 120,16 140,16 160,16 180,16 ↵
  195)'),↵
  15, 'quad_segs=4 join=round'));↵
--output --↵
LINESTRING(164 1,18 1,12.2597485145237 ↵
  2.1418070123307,↵
    7.39339828220179 ↵
  5.39339828220179,↵
    5.39339828220179 ↵
  7.39339828220179,↵
    2.14180701233067 ↵
  12.2597485145237,1 18,1 195)
```



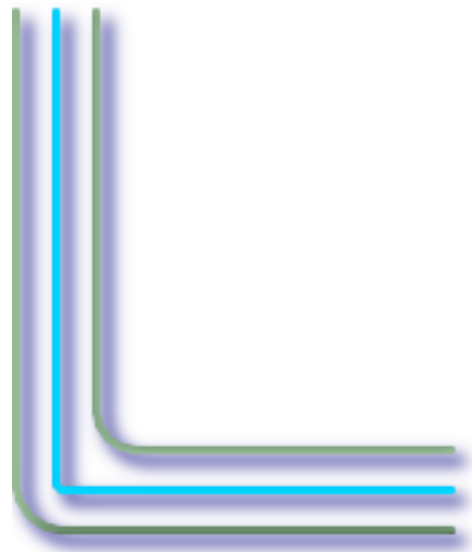
-15, 'quad\_segs=4 join=round' original line and its offset -15 units

```
SELECT ST_AsText(ST_OffsetCurve(geom,↵
  -15, 'quad_segs=4 join=round')) ↵
  As notsocurvy↵
  FROM ST_GeomFromText(↵
'LINESTRING(164 16,144 16,124 16,104 ↵
  16,84 16,64 16,↵
    44 16,24 16,20 16,18 16,17 17,↵
    16 18,16 20,16 40,16 60,16 80,16 ↵
  100,↵
    16 120,16 140,16 160,16 180,16 ↵
  195)') As geom;↵
-- notsocurvy --↵
LINESTRING(31 195,31 31,164 31)
```



*double-offset para ficar mais curvo, note que o primeiro reverte a direção, então  $-30 + 15 = -15$*

```
SELECT ST_AsText(ST_OffsetCurve(↵
  ST_OffsetCurve(geom,↵
    -30, 'quad_segs=4 join=round'),↵
    -15, 'quad_segs=4 join=round')) As morecurvy
FROM ST_GeomFromText(
'LINESTRING(164 16,144 16,124 16,104 ↵
  16,84 16,64 16,↵
    44 16,24 16,20 16,18 16,17 17,↵
    16 18,16 20,16 40,16 60,16 80,16 ↵
    100,↵
    16 120,16 140,16 160,16 180,16 ↵
    195)') As geom;
-- morecurvy --
LINESTRING(164 31,46 31,40.2597485145236 ↵
  32.1418070123307,↵
  35.3933982822018 35.3933982822018,↵
  32.1418070123307 40.2597485145237,31 ↵
  46,31 195)
```



*double-offset para ficar mais curvo, combinado com offset 15 para obter linhas paralelas. Coberto com o original.*

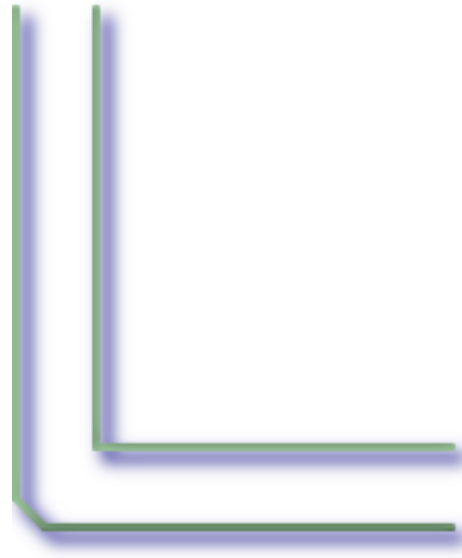
```
SELECT ST_AsText(ST_Collect(↵
  ST_OffsetCurve(geom, 15, '↵
    quad_segs=4 join=round'),↵
  ST_OffsetCurve(ST_OffsetCurve(↵
    geom,↵
    -30, 'quad_segs=4 join=round'),↵
    -15, 'quad_segs=4 join=round')↵
  )
  ) As parallel_curves
FROM ST_GeomFromText(
'LINESTRING(164 16,144 16,124 16,104 ↵
  16,84 16,64 16,↵
    44 16,24 16,20 16,18 16,17 17,↵
    16 18,16 20,16 40,16 60,16 80,16 ↵
    100,↵
    16 120,16 140,16 160,16 180,16 ↵
    195)') As geom;
-- parallel curves --
MULTILINESTRING((164 1,18 ↵
  1,12.2597485145237 2.1418070123307,↵
  7.39339828220179 ↵
  5.39339828220179,5.39339828220179 7.39339828220179,↵
  2.14180701233067 12.2597485145237,1 18,1 ↵
  195),↵
  (164 31,46 31,40.2597485145236 ↵
  32.1418070123307,35.3933982822018 35.3933982822018,↵
  32.1418070123307 40.2597485145237,31 ↵
  46,31 195))
```





15, 'quad\_segs=4 join=bevel' mostrado com a linha original

```
SELECT ST_AsText(ST_OffsetCurve(↵
  ST_GeomFromText(↵
    'LINESTRING(164 16,144 16,124 16,104 ↵
      16,84 16,64 16,↵
        44 16,24 16,20 16,18 16,17 17,↵
          16 18,16 20,16 40,16 60,16 80,16 ↵
            100,↵
              16 120,16 140,16 160,16 180,16 ↵
                195)'),↵
              15, 'quad_segs=4 join=↵
                bevel'));↵
-- output --↵
LINESTRING(164 1,18 1,7.39339828220179 ↵
  5.39339828220179,↵
  5.39339828220179 ↵
  7.39339828220179,1 18,1 195)
```



15,-15 collected, join=mitre mitre\_limit=2.1

```
SELECT ST_AsText(ST_Collect(↵
  ST_OffsetCurve(geom, 15, '↵
    quad_segs=4 join=mitre mitre_limit=2.2'),↵
  ST_OffsetCurve(geom, -15, '↵
    quad_segs=4 join=mitre mitre_limit=2.2')↵
  ) )↵
FROM ST_GeomFromText(↵
  'LINESTRING(164 16,144 16,124 16,104 ↵
    16,84 16,64 16,↵
      44 16,24 16,20 16,18 16,17 17,↵
        16 18,16 20,16 40,16 60,16 80,16 ↵
          100,↵
            16 120,16 140,16 160,16 180,16 ↵
              195)'),↵
  As geom;↵
-- output --↵
MULTILINESTRING((164 1,11.7867965644036 ↵
  1,1 11.7867965644036,1 195),↵
  (31 195,31 31,164 31))
```

Veja também.

[ST\\_Buffer](#)

### 8.11.25 ST\_RemoveRepeatedPoints

`ST_RemoveRepeatedPoints` — Retorna uma versão da geometria dada com pontos removidos duplicados.

#### Synopsis

geometry `ST_RemoveRepeatedPoints`(geometry geom, float8 tolerance);

#### Descrição

Retorna uma versão da geometria dada com pontos removidos duplicados. Só irá fazer algo com (multi)lines, (multi)polígonos e multipontos, mas você pode usar com qualquer tipo de geometria. Já que ocorre a simplificação em uma base objeto por objeto,

you can also feed a `GeometryCollection` for this function.

If the tolerance parameter is provided, vertices within the tolerance of another will be considered "the same" for the purposes of removal.

Disponibilidade: 2.2.0



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.

**Veja também.**

[ST\\_Simplify](#)

### 8.11.26 ST\_SharedPaths

`ST_SharedPaths` — Retorna uma coleção contendo caminhos compartilhados pelas duas `linestrings/multilinestrings` de entrada.

#### Synopsis

```
geometry ST_SharedPaths(geometry lineal1, geometry lineal2);
```

#### Descrição

Retorna uma coleção contendo caminhos compartilhados pelas duas geometrias de entrada. Aquelas indo na mesma direção estão no primeiro elemento da coleção, aquelas indo na direção oposta estão no segundo elemento. Os caminhos por si mesmos são dados na direção da primeira geometria.

Disponibilidade: 2.0.0. requer GEOS >= 3.3.0.

#### Exemplos: Encontrando caminhos compartilhados



*Uma multilinestring e uma linestring*



*O caminho compartilhado de multilinestring e linestring revestido com as geometrias originais.*

```
SELECT ST_AsText (
  ST_SharedPaths (
    ST_GeomFromText ('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),
      (51 150,101 150,76 175,51 150))'),
    ST_GeomFromText ('LINESTRING(151 100,126 156.25,126 125,90 161, 76 175)')
  )
) As wkt
```

wkt

```
-----
GEOMETRYCOLLECTION(MULTILINESTRING((126 156.25,126 125),
(101 150,90 161),(90 161,76 175)),MULTILINESTRING EMPTY)
```

-- same example but linestring orientation flipped

```
SELECT ST_AsText (
  ST_SharedPaths (
    ST_GeomFromText ('LINESTRING(76 175,90 161,126 125,126 156.25,151 100)'),
    ST_GeomFromText ('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),
      (51 150,101 150,76 175,51 150))')
  )
) As wkt
```

wkt

```
-----
GEOMETRYCOLLECTION(MULTILINESTRING EMPTY,
MULTILINESTRING((76 175,90 161),(90 161,101 150),(126 125,126 156.25)))
```

**Veja também.**

[ST\\_Dump](#), [ST\\_GeometryN](#), [ST\\_NumGeometries](#)

### 8.11.27 ST\_ShiftLongitude

ST\_ShiftLongitude — Coordenadas de geometria fechada entre -180...180 e 0...360 extensões.

#### Synopsis

geometry **ST\_ShiftLongitude**(geometry geomA);

#### Descrição

Lê todo ponto/vértice em cada componente de cada característica em uma geometria, e se a coordenada de longitude for <0, adiciona 360 a ele. O resultado seria uma versão 0-360 dos dados a serem tramados em um mapa 180 centric



#### Note

Só é útil para dados em long lat ex: 4326 (WGS 84 long lat)



Pre-1.3.4 bug impediu de funcionar para MULTIPONTO. 1.3.4+ funciona com MULTIPONTO também.



This function supports 3d and will not drop the z-index.

Melhorias: 2.0.0 suporte para superfícies poliédricas e TIN foi introduzido.

NOTA: esta função foi renomeada da "ST\_Shift\_Longitude" em 2.2.0



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

#### Exemplos

```
--3d points
SELECT ST_AsEWKT(ST_ShiftLongitude(ST_GeomFromEWKT('SRID=4326;POINT(-118.58 38.38 10)')) ←
  As geomA,
  ST_AsEWKT(ST_ShiftLongitude(ST_GeomFromEWKT('SRID=4326;POINT(241.42 38.38 10)')) ←
  As geomB
geomA                                geomB
-----                                -----
SRID=4326;POINT(241.42 38.38 10) SRID=4326;POINT(-118.58 38.38 10)

--regular line string
SELECT ST_AsText(ST_ShiftLongitude(ST_GeomFromText('LINESTRING(-118.58 38.38, -118.20 ←
  38.45)'))))
st_astext
-----
LINESTRING(241.42 38.38,241.8 38.45)
```

#### Veja também.

[ST\\_WrapX](#)

### 8.11.28 ST\_WrapX

ST\_WrapX — Envolve uma geometria em torno de um valor X.

#### Synopsis

geometry **ST\_WrapX**(geometry geom, float8 wrap, float8 move);

#### Descrição

Esta função divide as geometrias de entrada e então move cada componente caindo na direita (para "movimento" negativo) ou na esquerda (para "movimento" positivo) da dada linha "wrap" na direção especificada pelo parâmetro de "movimento", enfim reunindo as partes.



#### Note

Isto é útil para "recenter" entrada de long-lat para ter características de interesse não gerados de um lado para o outro.

Disponibilidade: 2.3.0



This function supports 3d and will not drop the z-index.

#### Exemplos

```
-- Move all components of the given geometries whose bounding box
-- falls completely on the left of x=0 to +360
select ST_WrapX(the_geom, 0, 360);

-- Move all components of the given geometries whose bounding box
-- falls completely on the left of x=-30 to +360
select ST_WrapX(the_geom, -30, 360);
```

**Veja também.**

[ST\\_ShiftLongitude](#)

### 8.11.29 ST\_Simplify

ST\_Simplify — Retorna uma versão "simplificada" da geometria usando o algoritmo Douglas-Peucker.

#### Synopsis

geometry **ST\_Simplify**(geometry geomA, float tolerance, boolean preserveCollapsed);





**Note**

Note que a geometria retornada pode perder sua simplicidade (veja [ST\\_IsSimple](#))

**Note**

Note que a topologia pode não ser preservada e resultar em geometrias inválidas. Use (veja [ST\\_SimplifyPreserveTopology](#)) para preservar a topologia.

**Note**

Esta função lida com 3D e a terceira dimensão afetará o resultado.

Disponibilidade: 2.2.0

**Exemplos**

Uma LineString é simplificada com uma área mínima a ponto de 30.

```
select ST_AsText(ST_SimplifyVW(geom,30)) simplified
FROM (SELECT 'LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)::geometry geom) As foo;
-result
simplified
-----+-----+
LINESTRING(5 2,7 25,10 10)
```

**Veja também.**

[ST\\_SetEffectiveArea](#), [ST\\_Simplify](#), [ST\\_SimplifyPreserveTopology](#), [Topology ST\\_Simplify](#)

**8.11.32 ST\_ChaikinSmoothing**

`ST_ChaikinSmoothing` — Retorna uma versão "simplificada" da geometria usando o algoritmo Visvalingam-Whyatt.

**Synopsis**

geometry **ST\_ChaikinSmoothing**(geometry geom, integer nIterations = 1, boolean preserveEndPoints = false);

**Descrição**

Returns a "smoothed" version of the given geometry using the Chaikin algorithm. See [Chaikins-Algorithm](#) for an explanation of the process. For each iteration the number of vertex points will double. The function puts new vertex points at 1/4 of the line before and after each point and removes the original point. To reduce the number of points use one of the simplification functions on the result. The new points gets interpolated values for all included dimensions, also z and m.

Second argument, number of iterations is limited to max 5 iterations

Note third argument is only valid for polygons, and will be ignored for linestrings

Esta função lida com 3D e a terceira dimensão afetará o resultado.





**Note**

This function returns all dimensions, also the z and m-value

Disponibilidade: 2.0.0

**Exemplos**

A linestring is filtered

```
select ST_AsText(ST_SimplifyVW(geom,30)) simplified
FROM (SELECT 'LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)::geometry geom) As foo;
-result
  simplified
-----+-----+
LINESTRING(5 2,7 25,10 10)
```

Veja também.

[ST\\_SetEffectiveArea](#), [ST\\_Dump](#)

**8.11.34 ST\_SetEffectiveArea**

**ST\_SetEffectiveArea** — Define a área eficaz para cada vértice, armazenando o valor na ordenada M. Uma geometria simplificada pode ser gerada por filtração da ordenada M.

**Synopsis**

```
geometry ST_SetEffectiveArea(geometry geomA, float threshold = 0, integer set_area = 1);
```

**Descrição**

Define a área eficaz para cada vértice, usando o algoritmo Visvalingam-Whyatt. A área efetiva é armazenada como o valor-M do vértice. Se o parâmetro opcional "limiar" for usado, uma geometria simplificada vai retornar, contendo somente vértices com uma área efetiva maior ou igual ao valor limiar.

Esta função pode ser usada para simplificação do lado do servidor quando uma limiar estiver especificada. Outra opção é usar um valor limiar de zero. Neste caso, a geometria completa retornará com áreas eficazes como valores-M, que podem ser usados pelo cliente para simplificar rapidamente.

Só irá fazer algo com (multi)lines, (multi)polígonos e multipontos, mas você pode usar com qualquer tipo de geometria. Já que ocorre a simplificação em uma base objeto por objeto, você também pode alimentar uma GeometryCollection para esta função.

**Note**

Note que a geometria retornada pode perder sua simplicidade (veja [ST\\_IsSimple](#))

**Note**

Note que a topologia pode não ser preservada e resultar em geometrias inválidas. Use (veja [ST\\_SimplifyPreserveTopology](#)) para preservar a topologia.

**Note**

A geometria de saída perderá todas as informações prévias nos valores-M

**Note**

Esta função lida com 3D e a terceira dimensão afetará a área eficaz.

Disponibilidade: 2.2.0

**Exemplos**

Calculando a área eficaz de uma LineString. Devido ao uso de um valor limiar zero, todos os vértices na geometria de entrada são retornados.

```
select ST_AsText(ST_SetEffectiveArea(geom)) all_pts, ST_AsText(ST_SetEffectiveArea(geom,30) ←
) thrshld_30
FROM (SELECT 'LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)'::geometry geom) As foo;
-result
all_pts | thrshld_30
-----+-----+
LINESTRING M (5 2 3.40282346638529e+38,3 8 29,6 20 1.5,7 25 49.5,10 10 3.40282346638529e ←
+38) | LINESTRING M (5 2 3.40282346638529e+38,7 25 49.5,10 10 3.40282346638529e+38)
```

Veja também.

[ST\\_SimplifyVW](#)

**8.11.35 ST\_Split**

ST\_Split — Retorna uma coleção de geometrias resultante da divisão de uma geometria.

**Synopsis**

```
geometry ST_Split(geometry input, geometry blade);
```

**Descrição**

A função suporta dividir uma linha por (multi)point, (multi)linha ou limite (multi)polígono, um (multi)polígono por linha. A geometria retornada é sempre uma coleção.

Veja esta função como oposta da ST\_Union. Teoricamente, aplicar a ST\_Union aos elementos da coleção retornada deveria sempre dar preferência a geometria original.

Disponibilidade: 2.0.0

Melhorias: 2.2.0 suporte para dividir uma linha por multilinha, um multiponto ou limite de (multi)polígono foi introduzido.

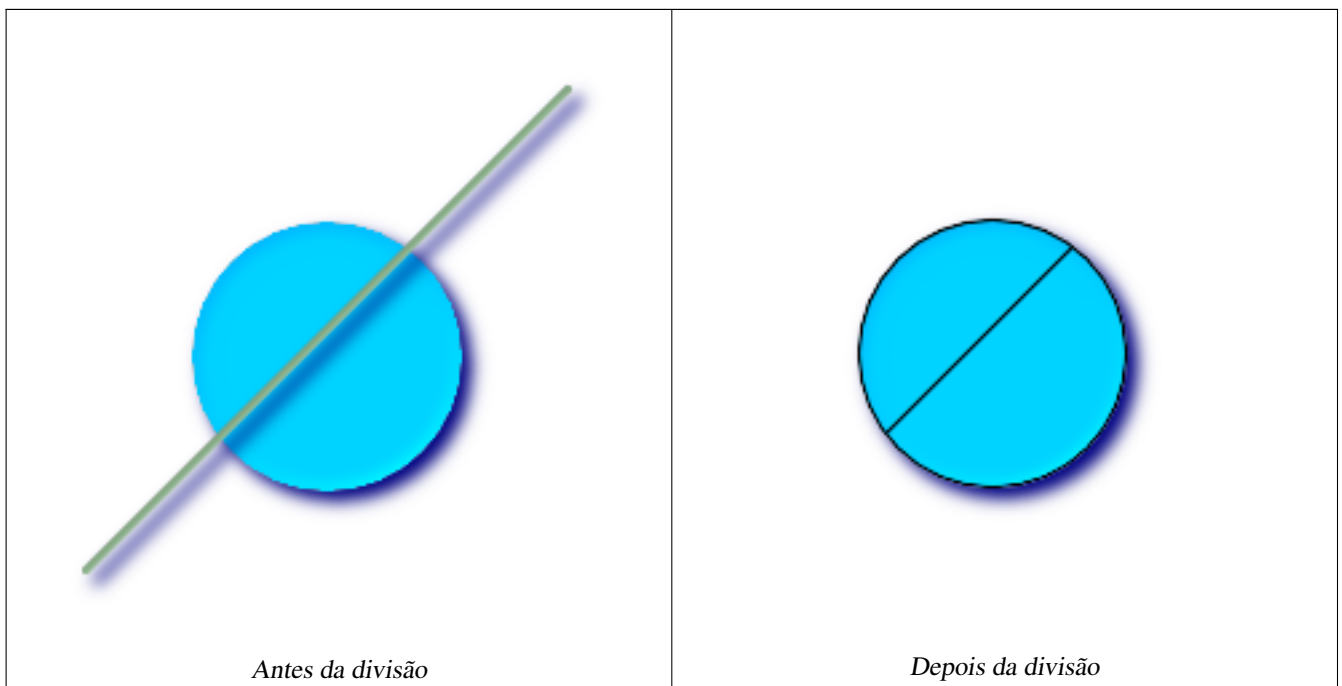
Melhorias: 2.2.0 suporte para dividir uma linha por multilinha, um multiponto ou limite de (multi)polígono foi introduzido.

**Note**

Para melhorar a força da `ST_Split` pode ser conveniente `ST_Snap` a entrada para o blade na frente usando uma tolerância muito baixa. Caso contrário, a coordenada usada internamente pode causar problemas de tolerância, nos quais as coordenadas de entrada e o blade não caem um no outro e a entrada não está sendo dividida corretamente (veja [#2192](#)).

**Note**

Quando um (multi)polígono é passado como o blade, seu componente linear (o limite) é usado para cortar a entrada.

**Exemplos****Polígono cortado pela linha**

```
-- this creates a geometry collection consisting of the 2 halves of the polygon
-- this is similar to the example we demonstrated in ST_BuildArea
SELECT ST_Split(circle, line)
FROM (SELECT
  ST_MakeLine(ST_MakePoint(10, 10),ST_MakePoint(190, 190)) As line,
  ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50) As circle) As foo;

-- result --
GEOMETRYCOLLECTION(POLYGON((150 90,149.039264020162 80.2454838991936,146.193976625564 ↵
  70.8658283817455,..), POLYGON(..)))

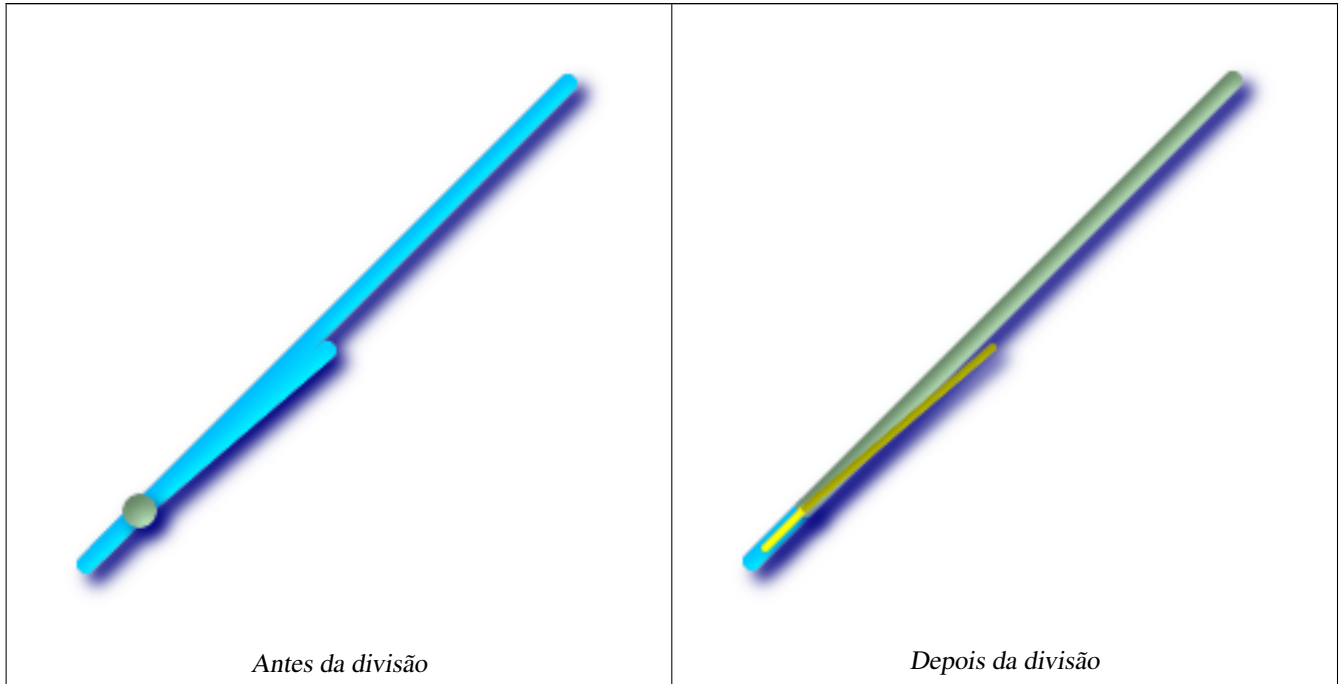
-- To convert to individual polygons, you can use ST_Dump or ST_GeometryN
SELECT ST_AsText((ST_Dump(ST_Split(circle, line)).geom) As wkt
FROM (SELECT
  ST_MakeLine(ST_MakePoint(10, 10),ST_MakePoint(190, 190)) As line,
  ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50) As circle) As foo;

-- result --
```

wkt

```
-----
POLYGON((150 90,149.039264020162 80.2454838991936,..))
POLYGON((60.1371179574584 60.1371179574584,58.4265193848728 ←
        62.2214883490198,53.8060233744357 ..))
```

### Multilinestring cortada por ponto



```
SELECT ST_AsText(ST_Split(mline, pt)) As wktcut
FROM (SELECT
  ST_GeomFromText('MULTILINESTRING((10 10, 190 190), (15 15, 30 30, 100 90))') As mline,
  ST_Point(30,30) As pt) As foo;
```

wktcut

```
-----
GEOMETRYCOLLECTION(
  LINESTRING(10 10,30 30),
  LINESTRING(30 30,190 190),
  LINESTRING(15 15,30 30),
  LINESTRING(30 30,100 90)
)
```

### Veja também.

[ST\\_AsText](#), [ST\\_BuildArea](#), [ST\\_Dump](#), [ST\\_GeometryN](#), [ST\\_Union](#), [ST\\_Subdivide](#)

### 8.11.36 ST\_SymDifference

`ST_SymDifference` — Retorna uma geometria que representa as porções de A e B que não intersectam. É chamado de diferença simétrica, porque  $ST\_SymDifference(A,B) = ST\_SymDifference(B,A)$ .

## Synopsis

geometry **ST\_SymDifference**(geometry geomA, geometry geomB);

## Descrição

Retorna uma geometria que representa as porções de A e B que não intersectam. É chamado de diferença simétrica, porque  $ST\_SymDifference(A,B) = ST\_SymDifference(B,A)$ . Pode-se pensar nisto como  $ST\_Union(geomA,geomB) - ST\_Intersection(A,B)$ .

Desempenhado pelo módulo GEOS



### Note

Não chama com uma GeometryCollection com um argumento



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3

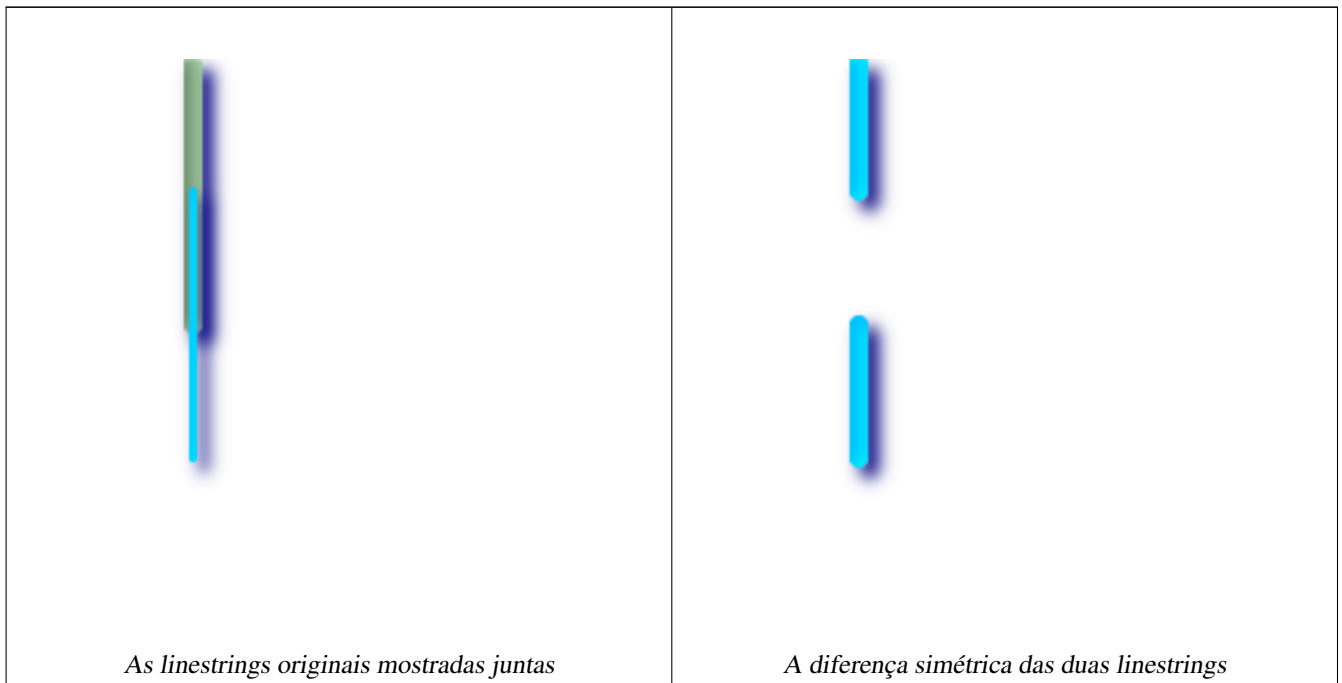


This method implements the SQL/MM specification. SQL-MM 3: 5.1.21



This function supports 3d and will not drop the z-index. Entretanto aparenta considerar apenas x y quando fazem diferença e retorna no Z-índice

## Exemplos



```
--Safe for 2d - symmetric difference of 2 linestrings
SELECT ST_AsText (
  ST_SymDifference (
    ST_GeomFromText ('LINESTRING(50 100, 50 200)'),
    ST_GeomFromText ('LINESTRING(50 50, 50 150)')
```

```

)
);

st_astext
-----
MULTILINESTRING((50 150,50 200),(50 50,50 100))

```

```

--When used in 3d doesn't quite do the right thing
SELECT ST_AsEWKT(ST_SymDifference(ST_GeomFromEWKT('LINESTRING(1 2 1, 1 4 2)'),
    ST_GeomFromEWKT('LINESTRING(1 1 3, 1 3 4)')))

st_astext
-----
MULTILINESTRING((1 3 2.75,1 4 2),(1 1 3,1 2 2.25))

```

**Veja também.**

[ST\\_Difference](#), [ST\\_Intersection](#), [ST\\_Union](#)

### 8.11.37 ST\_Subdivide

**ST\_Subdivide** — Retorna um conjunto de geometrias, no qual nenhuma geometria no conjunto tem mais vértices que o número especificado.

#### Synopsis

setof geometry **ST\_Subdivide**(geometry geom, integer max\_vertices=256);

#### Descrição

Divides geometry into parts until a part can be represented using no more than `max_vertices`. Point-in-polygon and other overlay operations are normally faster for indexed subdivided dataset: "miss" cases are faster to check as boxes for all parts typically cover smaller area than original geometry box, "hit" cases are faster because recheck operates on less points. Uses the same envelope clipping as `ST_ClipByBox2D`. `max_vertices` must be 5 or more, as 5 points are needed to represent a closed box.

Disponibilidade: 2.2.0 requer GEOS >= 3.5.0.

Enhanced: 2.5.0 reuses existing points on polygon split, vertex count is lowered from 8 to 5.

#### Exemplos

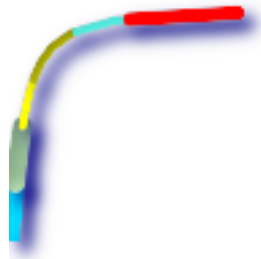
```

-- Subdivide complex geometries in table, in place
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;

```







Útil em conjunto com a `ST_Segmentize` para criar vértices adicionais que podem ser usados para dividir

```
SELECT ST_AsText(ST_SubDivide(ST_Segmentize('LINESTRING(0 0, 100 100, 150 150)'):: geometry, 10), 8);

LINESTRING(0 0,0.487578359029357 5.57659056746196,0.984542144675897 ↔
  11.1527721155093,1.50101059639722 16.7281035483571,1.94532113630331 21.25)
LINESTRING(1.94532113630331 21.25,2.04869538062779 22.3020741387339,2.64204641967673 ↔
  27.8740533545155,3.29994062412787 33.443216802941,4.04836719489742 ↔
  39.0084282520239,4.59890468420694 42.5)
LINESTRING(4.59890468420694 42.5,4.92498503922732 44.5680389206321,5.98737409390639 ↔
  50.1195229244701,7.3290919767674 55.6587646879025,8.79638749938413 60.1969505994924)
LINESTRING(8.79638749938413 60.1969505994924,9.11375579533779 ↔
  61.1785363177625,11.6558166691368 66.6648504160202,15.642041247655 ↔
  72.0867690601745,22.8716627200212 77.3609628116894,24.6991785131552 77.8939011989848)
LINESTRING(24.6991785131552 77.8939011989848,39.4046096622744 ↔
  82.1822848017636,44.7994523421035 82.5156766227011)
LINESTRING(44.7994523421035 82.5156766227011,85 85)
```

Veja também.

[ST\\_AsText](#), [ST\\_ClipByBox2D](#), [ST\\_Segmentize](#), [ST\\_Split](#)

### 8.11.38 ST\_SwapOrdinates

`ST_SwapOrdinates` — Retorna uma versão da geometria dada com os valores ordenados dados trocados.

#### Synopsis

geometry `ST_SwapOrdinates`(geometry geom, cstring ords);

#### Descrição

Retorna uma versão da geometria dada com as ordenadas dadas trocadas.

O parâmetro `ords` é uma string de 2-caracteres nomeando as ordenadas para trocar. Os nomes válidos são: x,y,z e m.

Disponibilidade: 2.2.0



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.



This function supports M coordinates.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

### Exemplo

```
-- Scale M value by 2
SELECT ST_AsText (
  ST_SwapOrdinates (
    ST_Scale (
      ST_SwapOrdinates (g, 'xm'),
      2, 1
    ),
    'xm'
  )
) FROM ( SELECT 'POINT ZM (0 0 0 2):::geometry g ) foo;
-----
POINT ZM (0 0 0 4)
```

Veja também.

[ST\\_FlipCoordinates](#)

### 8.11.39 ST\_Union

**ST\_Union** — Retorna uma geometria que representa a união de pontos das Geometrias.

#### Synopsis

```
geometry ST_Union(geometry set g1field);
geometry ST_Union(geometry g1, geometry g2);
geometry ST_Union(geometry[] g1_array);
```

#### Descrição

O tipo de saída pode ser uma MULTI\*, geometria única, ou Coleção de Geometria. Vem com 2 variantes. A variante 1 une 2 geometrias resultando em uma nova geometria sem regiões de interseção. A variante 2 é uma função agregada que pega um conjunto de geometrias e as une dentro de uma única ST\_Geometry resultando em regiões sem interseção.

Versão agregada: Esta função retorna uma MULTI geometria ou NÃO-MULTI geometria de um conjunto de geometrias. A função ST\_Union() é uma função "agregada" na terminologia do PostgreSQL. Isso significa que ela opera em filas de dados, da mesma forma que as funções SUM() e AVG() e como a maioria dos agregados, também ignora geometrias NULAS.

Versão não-agregada: Esta função retorna uma geometria sendo uma união de duas geometrias de entrada. O tipo de saída pode ser MULTI\*, NÃO-MULTI ou GEOMETRYCOLLECTION. Se algum for NULO, então retorna NULO.

**Note**

As `ST_Collect` e `ST_Union` são imutáveis. `ST_Union` tem ordem de grandeza menor que a `ST_Collect`, porque tenta dissolver limites e reordenar geometrias para assegurar que uma Multi\* construída não tenha regiões intersectando.

Desempenhado pelo módulo GEOS.

NOTA: esta função foi formalmente chamada de `GeomUnion()`, a qual foi renomeada de "Union" porque UNION é uma palavra reservada SQL.

Disponibilidade: 1.4.0 - a `ST_Union` foi melhorada. `ST_Union(geomarray)` foi introduzida e também uma coleção agregada mais rápida no PostgreSQL. Se você está usando GEOS 3.1.0+ a `ST_Union` irá usar o algoritmo Cascaded Union mais rápido, descrito em <http://blog.cleverelephant.ca/2009/01/must-faster-unions-in-postgis-14.html>



This method implements the [OpenGIS Simple Features Implementation Specification for SQL 1.1. s2.1.1.3](#)

**Note**

A versão agregada não está explicitamente definida no OGC SPEC.



This method implements the SQL/MM specification. SQL-MM 3: 5.1.19 o índice-z (elevação) quando polígonos não estão envolvidos.

**Exemplos****Exemplo agregado**

```
SELECT stusps,
       ST_Multi(ST_Union(f.the_geom)) as singlegeom
FROM sometable As f
GROUP BY stusps
```

**Exemplo não agregado**

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

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

SELECT ST_AsText(ST_Union(ST_GeomFromText('POINT(1 2)'),
                          ST_GeomFromText('POINT(1 2)') ) );

st_astext
-----
POINT(1 2)

--3d example - sort of supports 3d (and with mixed dimensions!)
SELECT ST_AsEWKT(st_union(the_geom))
FROM
(SELECT ST_GeomFromEWKT('POLYGON((-7 4.2,-7.1 4.2,-7.1 4.3,
-7 4.2))') as the_geom
UNION ALL
SELECT ST_GeomFromEWKT('POINT(5 5 5)') as the_geom
UNION ALL
```

```

        SELECT ST_GeomFromEWKT('POINT(-2 3 1)') as the_geom
UNION ALL
SELECT ST_GeomFromEWKT('LINESTRING(5 5 5, 10 10 10)') as the_geom ) as foo;

st_asewkt
-----
GEOMETRYCOLLECTION(POINT(-2 3 1),LINESTRING(5 5 5,10 10 10),POLYGON((-7 4.2 5,-7.1 4.2 5,-7.1 4.3 5,-7 4.2 5)));

--3d example not mixing dimensions
SELECT ST_AsEWKT(st_union(the_geom))
FROM
(SELECT ST_GeomFromEWKT('POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2,-7 4.2 2)')') as the_geom
UNION ALL
SELECT ST_GeomFromEWKT('POINT(5 5 5)') as the_geom
UNION ALL
        SELECT ST_GeomFromEWKT('POINT(-2 3 1)') as the_geom
UNION ALL
SELECT ST_GeomFromEWKT('LINESTRING(5 5 5, 10 10 10)') as the_geom ) as foo;

st_asewkt
-----
GEOMETRYCOLLECTION(POINT(-2 3 1),LINESTRING(5 5 5,10 10 10),POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2,-7 4.2 2)));

--Examples using new Array construct
SELECT ST_Union(ARRAY(SELECT the_geom FROM sometable));

SELECT ST_AsText(ST_Union(ARRAY[ST_GeomFromText('LINESTRING(1 2, 3 4)'),
        ST_GeomFromText('LINESTRING(3 4, 4 5)')])) As wktunion;

--wktunion---
MULTILINESTRING((3 4,4 5),(1 2,3 4))

```

**Veja também.**[ST\\_Collect ST\\_UnaryUnion](#)**8.11.40 ST\_UnaryUnion**

ST\_UnaryUnion — Parecido com ST\_Union, mas funcionando no nível do componente da geometria.

**Synopsis**

geometry **ST\_UnaryUnion**(geometry geom);

**Descrição**

Diferente de ST\_Union, ST\_UnaryUnion dissolve limites entre componentes de um multipolígono (inválido) e faz união entre os elementos de uma coleção de geometria. Cada parte da geometria de entrada é válida, então você não pegará um multipolígono de um polígono bow-tie (inválido).

Você pode usar esta função para nodar um conjunto de linestrings. Pode misturar ST\_UnaryUnion com ST\_Collect para sintonizar quantas geometrias você vai querer dissolver de uma vez para ficar bom para o tamanho da memória e a CPU, encontrando o equilíbrio entre ST\_Union e ST\_MemUnion.



This function supports 3d and will not drop the z-index.

Disponibilidade: 2.0.0 - requer GEOS >= 3.3.0.

**Veja também.**

[ST\\_Union](#), [ST\\_MemUnion](#), [ST\\_Collect](#), [ST\\_Node](#)

### 8.11.41 ST\_Voronoi

ST\_Voronoi — Calcula um diagrama de Voronoi dos vértices de uma geometria.

#### Synopsis

geometry **ST\_Voronoi**( g1 geometry , clip geometry , tolerance float8 , return\_polygons boolean );

#### Descrição

ST\_Voronoi calcula um **Voronoi diagram** bidimensional dos vértices da geometria fornecida. Por padrão, o resultado será uma coleção de geometria de polígonos que cobre um envelope maior que a extensão dos vértices de entrada.

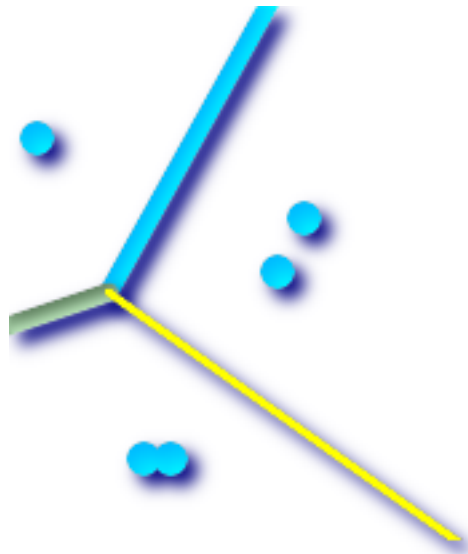
Parâmetros opcionais:

- 'tolerância' : A distância dentro dos vértices serão considerados equivalentes. A força o algoritmo pode ser melhorada fornecendo uma distância de tolerância não zero. (padrão = 0.0)
- 'clip' : Se uma geometria é fornecida como o parâmetro "clip", o diagrama será estendido para cobrir o envelope da geometria "clip", a menos que o envelope seja menor que o envelope padrão. (padrão=NULO)

Disponibilidade: 2.3.0 - requer GEOS >= 3.5.0.

#### Exemplos

---



*Voronoi com tolerância de 30 unidade*

```
SELECT ST_Voronoi(geom, null,30) As geom
FROM (SELECT 'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120)>:::geometry As geom ) ↔
  As g;

-- ST_AsText output
MULTILINESTRING(((135.555555555556 270,36.8181818181818 92.2727272727273) ↔
, (36.8181818181818 92.2727272727273,-110 43.3333333333333), (230 ↔
-45.7142857142858,36.8181818181818 92.2727272727273))
```

**Veja também.**

[ST\\_DelaunayTriangles](#), [ST\\_Intersection](#), [ST\\_Union](#)

### 8.11.42 ST\_Voronoi

ST\_Voronoi — Calcula um diagrama de Voronoi dos vértices de uma geometria.

#### Synopsis

geometry **ST\_Voronoi**( g1 geometry , clip geometry , tolerance float8 , return\_polygons boolean );

#### Descrição

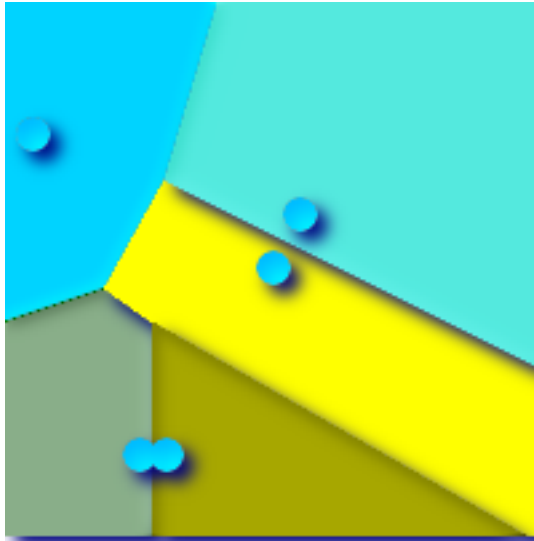
ST\_Voronoi calcula um **Voronoi diagram** bidimensional dos vértices da geometria fornecida. Por padrão, o resultado será uma coleção de geometria de polígonos que cobre um envelope maior que a extensão dos vértices de entrada.

Parâmetros opcionais:

- 'tolerância' : A distância dentro dos vértices serão considerados equivalentes. A força o algoritmo pode ser melhorada fornecendo uma distância de tolerância não zero. (padrão = 0.0)
- 'clip' : Se uma geometria é fornecida como o parâmetro "clip", o diagrama será estendido para cobrir o envelope da geometria "clip", a menos que o envelope seja menor que o envelope padrão. (padrão=NULO)

Disponibilidade: 2.3.0 - requer GEOS >= 3.5.0.

## Exemplos



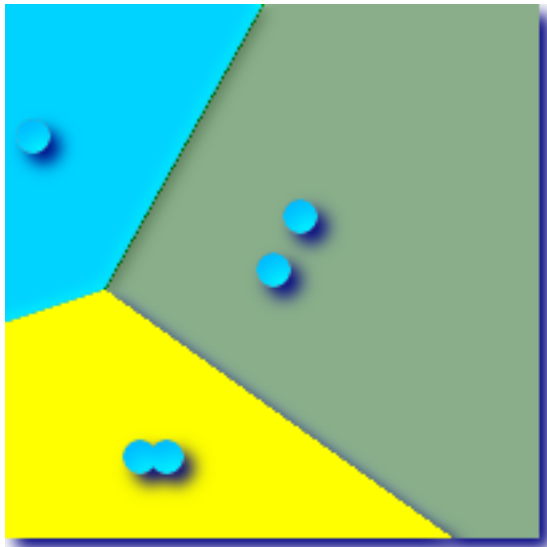
*Pontos revestidos no topo do diagrama de voronoi*

```

SELECT
    ST_Voronoi(geom) As geom
FROM (SELECT 'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120) '::geometry As geom ) ←
    As g;

-- ST_AsText output
GEOMETRYCOLLECTION(POLYGON((-110 43.33333333333333,-110 270,100.5 270,59.3478260869565 ←
    132.826086956522,36.8181818181818 92.2727272727273,-110 43.3333333333333)), ←
POLYGON((55 -90,-110 -90,-110 43.3333333333333,36.8181818181818 92.2727272727273,55 ←
    79.2857142857143,55 -90)), ←
POLYGON((230 47.5,230 -20.7142857142857,55 79.2857142857143,36.8181818181818 ←
    92.2727272727273,59.3478260869565 132.826086956522,230 47.5)),POLYGON((230 ←
    -20.7142857142857,230 -90,55 -90,55 79.2857142857143,230 -20.7142857142857)), ←
POLYGON((100.5 270,230 270,230 47.5,59.3478260869565 132.826086956522,100.5 270)))

```

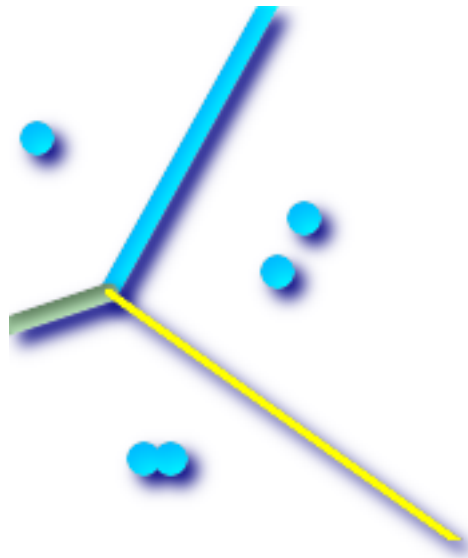


*Voronoi com tolerância de 30 unidade*

```
SELECT ST_Voronoi(geom, null,30) As geom
FROM (SELECT 'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120)>:::geometry As geom ) ↔
      As g;

-- ST_AsText output
GEOMETRYCOLLECTION(POLYGON((-110 43.33333333333333,-110 270,100.5 270,59.3478260869565 ↔
132.826086956522,36.8181818181818 92.2727272727273,-110 43.3333333333333)),
POLYGON((230 47.5,230 -45.7142857142858,36.8181818181818 ↔
92.2727272727273,59.3478260869565 132.826086956522,230 47.5)),POLYGON((230 ↔
-45.7142857142858,230 -90,-110 -90,-110 43.3333333333333,36.8181818181818 ↔
92.2727272727273,230 -45.7142857142858)),
POLYGON((100.5 270,230 270,230 47.5,59.3478260869565 132.826086956522,100.5 270)))
```





*Voronoi com tolerância de 30 unidades como multilinestring*

```
SELECT ST_Voronoi(geom, null,30) As geom
FROM (SELECT 'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120)>:::geometry As geom ) ←
      As g;

-- ST_AsText output
MULTILINESTRING((135.555555555556 270,36.8181818181818 92.2727272727273) ←
, (36.8181818181818 92.2727272727273,-110 43.3333333333333), (230 ←
-45.7142857142858,36.8181818181818 92.2727272727273))
```

**Veja também.**

[ST\\_DelaunayTriangles](#), [ST\\_Intersection](#), [ST\\_Union](#)

## 8.12 Referência linear

### 8.12.1 ST\_LineInterpolatePoint

**ST\_LineInterpolatePoint** — Retorna um ponto interpolar ao longo de uma linha. Segundo argumento é um float8 entre 0 e 1 representando fração do comprimento total da linestring do ponto tem que ser localizado.

#### Synopsis

```
geometria ST_LineInterpolatePoint(geometria a_linestring, float8 a_fraction);
```

#### Descrição

Retorna um ponto interpolar com uma linha. Primeiro argumento deve ser uma LINESTRING. Segundo argumento é um float8 entre 0 e 1 representando fração do comprimento total da linestring do ponto tem que ser localizado.

Veja [ST\\_LineLocatePoint](#) para computar a linha de localização mais perto de um ponto.

**Note**

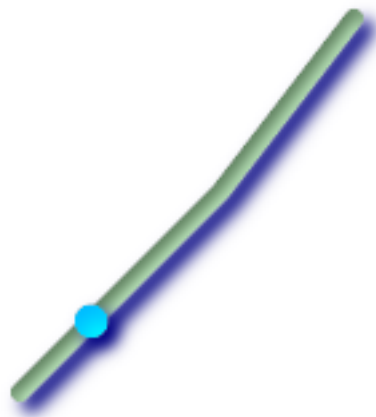
Desde a liberação 1.1.1 essa função também interpola valores M e Z (quando presentes), enquanto as liberações anteriores configura eles para 0.0.

Disponibilidade: 0.8.2, Suporte a Z e M adicionado em 1.1.1

Alterações: 2.1.0 para 2.0.x foi chamada ST\_Line\_Interpolate\_Point.



This function supports 3d and will not drop the z-index.

**Exemplos**

*Uma linestring com o ponto interpolado em uma posição de 20% (0.20)*

```
--Return point 20% along 2d line
SELECT ST_AsEWKT(ST_LineInterpolatePoint(the_line, 0.20))
      FROM (SELECT ST_GeomFromEWKT('LINESTRING(25 50, 100 125, 150 190)') as the_line) As foo;
      st_asewkt
-----
POINT(51.5974135047432 76.5974135047432)
```

```
--Return point mid-way of 3d line
SELECT ST_AsEWKT(ST_LineInterpolatePoint(the_line, 0.5))
      FROM (SELECT ST_GeomFromEWKT('LINESTRING(1 2 3, 4 5 6, 6 7 8)') as the_line) As foo;
      st_asewkt
-----
POINT(3.5 4.5 5.5)
```

```
--find closest point on a line to a point or other geometry
SELECT ST_AsText(ST_LineInterpolatePoint(foo.the_line, ST_LineLocatePoint(foo.the_line, ST_GeomFromText('POINT(4 3)'))))
      FROM (SELECT ST_GeomFromText('LINESTRING(1 2, 4 5, 6 7)') As the_line) As foo;
```

```
st_astext
```

```
POINT(3 4)
```

### Veja Também

[ST\\_AsText](#), [ST\\_AsEWKT](#), [ST\\_Length](#), [ST\\_LineInterpolatePoints](#) [ST\\_LineLocatePoint](#) [ST\\_LineM](#)

## 8.12.2 ST\_LineInterpolatePoints

`ST_LineInterpolatePoints` — Returns one or more points interpolated along a line.

### Synopsis

geometry **ST\_LineInterpolatePoints**(geometry a\_linestring, float8 a\_fraction, boolean repeat);

### Descrição

Returns one or more points interpolated along a line. First argument must be a `LINestring`. Second argument is a float8 between 0 and 1 representing the spacing between the points as a fraction of total `LineString` length. If the third argument is false, at most one point will be constructed (the function will be equivalent to [ST\\_LineInterpolatePoint](#).)

If the result has zero or one points, it will be returned as a `POINT`. If it has two or more points, it will be returned as a `MULTIPOINT`.

Availability: 2.5.0

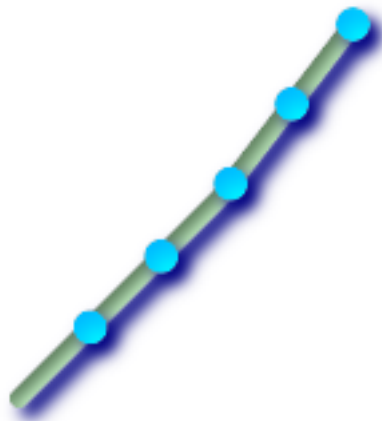


This function supports 3d and will not drop the z-index.



This function supports M coordinates.

### Exemplos



*A linestring with the interpolated points every 20%*

```
--Return points each 20% along a 2D line
SELECT ST_AsText(ST_LineInterpolatePoints('LINESTRING(25 50, 100 125, 150 190)', 0.20))
      st_astext
-----
MULTIPOINT(51.5974135047432 76.5974135047432,78.1948270094864 ↔
          103.194827009486,104.132163186446 130.37181214238,127.066081593223 160.18590607119,150 ↔
          190)
```

## Veja Também

[ST\\_LineInterpolatePoint](#) [ST\\_LineLocatePoint](#)

### 8.12.3 ST\_LineLocatePoint

**ST\_LineLocatePoint** — Retorna um flutuador entre 0 e 1 representando a localização do ponto mais próximo na linestring do ponto dado, como uma fração de uma linha de comprimento 2d total.

#### Synopsis

```
float8 ST_LineLocatePoint(geometria a_linestring, geometria a_point);
```

#### Descrição

Retorna um flutuador entre 0 e 1 representando a localização do ponto mais próximo na linestring do ponto dado, como uma fração de uma **2d line** de comprimento total.

Você pode usar a localização retornada para extrair um ponto ([ST\\_LineInterpolatePoint](#)) ou uma substring ([ST\\_LineSubstring](#)).

Isso é útil para aproximar números de endereços

Disponibilidade: 1.1.0

Alterações: 2.1.0 para 2.0.x foi chamada `ST_Line_Locate_Point`.

#### Exemplos

```
--Rough approximation of finding the street number of a point along the street
--Note the whole foo thing is just to generate dummy data that looks
--like house centroids and street
--We use ST_DWithin to exclude
--houses too far away from the street to be considered on the street
SELECT ST_AsText(house_loc) As as_text_house_loc,
      startstreet_num +
          CAST( (endstreet_num - startstreet_num)
              * ST_LineLocatePoint(street_line, house_loc) As integer) As ↔
          street_num
FROM
  (SELECT ST_GeomFromText('LINESTRING(1 2, 3 4)') As street_line,
        ST_MakePoint(x*1.01,y*1.03) As house_loc, 10 As startstreet_num,
        20 As endstreet_num
  FROM generate_series(1,3) x CROSS JOIN generate_series(2,4) As y)
As foo
WHERE ST_DWithin(street_line, house_loc, 0.2);

as_text_house_loc | street_num
-----+-----
```

```

POINT(1.01 2.06) |          10
POINT(2.02 3.09) |          15
POINT(3.03 4.12) |          20

--find closest point on a line to a point or other geometry
SELECT ST_AsText(ST_LineInterpolatePoint(foo.the_line, ST_LineLocatePoint(foo.the_line, ←
    ST_GeomFromText('POINT(4 3)'))))
FROM (SELECT ST_GeomFromText('LINESTRING(1 2, 4 5, 6 7)') As the_line) As foo;
    st_astext
-----
POINT(3 4)

```

## Veja Também

[ST\\_DWithin](#), [ST\\_Length2D](#), [ST\\_LineInterpolatePoint](#), [ST\\_LineSubstring](#)

### 8.12.4 ST\_LineSubstring

**ST\_LineSubstring** — Retorna uma linestring sendo uma substring da de entrada começando e finalizando nas frações dadas do total 2d de comprimento. Segundo e terceiro argumentos são valores float8 entre 0 e 1.

#### Synopsis

geometria **ST\_LineSubstring**(geometria a\_linestring, float8 startfraction, float8 endfraction);

#### Descrição

Retorna uma linestring sendo uma substring da de entrada começando e finalizando nas frações dadas do total 2d de comprimento. Segundo e terceiro argumentos são valores float8 entre 0 e 1. Isso só funciona com LINESTRINGs. Para usar com MULTILINESTRINGs contíguas, use em conjunção com [ST\\_LineMerge](#).

Se "início" e "fim" tiverem o mesmo valor, isso é equivalente a [ST\\_LineInterpolatePoint](#).

Veja [ST\\_LineLocatePoint](#) para computar a linha de localização mais perto de um ponto.



#### Note

Desde a liberação 1.1.1 essa função também interpola valores M e Z (quando presentes), enquanto as liberações anteriores configura eles para valores não específicos.

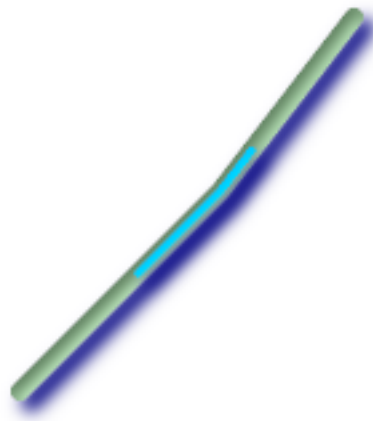
Disponibilidade: 1.1.0, Suporte a Z e M adicionado em 1.1.1

Alterações: 2.1.0 para 2.0.x foi chamada ST\_Line\_Substring.



This function supports 3d and will not drop the z-index.

## Exemplos



### *Uma linestring vista com 1/3 coberto (0.333, 0.666)*

```
--Return the approximate 1/3 mid-range part of a linestring
SELECT ST_AsText(ST_Line_SubString(ST_GeomFromText('LINESTRING(25 50, 100 125, 150 190)'), ←
    0.333, 0.666));
```

st\_astext ←

```
-----
LINESTRING(69.2846934853974 94.2846934853974,100 125,111.700356260683 140.210463138888)
```

```
--The below example simulates a while loop in
--SQL using PostgreSQL generate_series() to cut all
--linestrings in a table to 100 unit segments
-- of which no segment is longer than 100 units
-- units are measured in the SRID units of measurement
-- It also assumes all geometries are LINESTRING or contiguous MULTILINESTRING
--and no geometry is longer than 100 units*10000
--for better performance you can reduce the 10000
--to match max number of segments you expect
```

```
SELECT field1, field2, ST_LineSubstring(the_geom, 100.00*n/length,
    CASE
        WHEN 100.00*(n+1) < length THEN 100.00*(n+1)/length
        ELSE 1
    END) As the_geom
FROM
    (SELECT sometable.field1, sometable.field2,
    ST_LineMerge(sometable.the_geom) AS the_geom,
    ST_Length(sometable.the_geom) As length
    FROM sometable
    ) AS t
CROSS JOIN generate_series(0,10000) AS n
WHERE n*100.00/length < 1;
```

## Veja Também

[ST\\_Length](#), [ST\\_LineInterpolatePoint](#), [ST\\_LineMerge](#)

## 8.12.5 ST\_LocateAlong

ST\_LocateAlong — Retorna um valor de coleção de geometria derivado com elementos que combinam com a medida específica. Elementos polígonos não são suportados.

### Synopsis

geometria **ST\_LocateAlong**(geometria geom\_with\_measure, float8 a\_measure, float8 offset);

### Descrição

Retorna um valor de coleção de geometria derivado com elementos que combinam com a medida específica. Elementos polígonos não são suportados.

Se um deslocamento é fornecido, o resultado será o deslocamento para a direita ou para a esquerda da linha de entrada pelo número específico de unidades. Um deslocamento positivo será para a esquerda e um negativo para a direita.

Semântico é especificado por: ISO/IEC CD 13249-3:200x(E) - Text for Continuation CD Editing Meeting

Disponibilidade: 1.1.0 pelo nome antigo ST\_Locate\_Along\_Measure.

Alterações: 2.0.0 nas versões anteriores era chamado de ST\_Locate\_Along\_Measure. O nome antigo foi menosprezado e será removido no futuro, mas ainda está disponível.



#### Note

Utilize esta função apenas em geometrias com um componente M



This function supports M coordinates.

### Exemplos

```
SELECT ST_AsText(the_geom)
      FROM
      (SELECT ST_LocateAlong(
            ST_GeomFromText('MULTILINESTRINGM((1 2 3, 3 4 2, 9 4 3),
            (1 2 3, 5 4 5))'),3) As the_geom) As foo;

                                     st_asewkt
-----
MULTIPOINT M (1 2 3)

--Geometry collections are difficult animals so dump them
--to make them more digestable
SELECT ST_AsText((ST_Dump(the_geom)).geom)
      FROM
      (SELECT ST_LocateAlong(
            ST_GeomFromText('MULTILINESTRINGM((1 2 3, 3 4 2, 9 4 3),
            (1 2 3, 5 4 5))'),3) As the_geom) As foo;

      st_asewkt
-----
POINTM(1 2 3)
POINTM(9 4 3)
POINTM(1 2 3)
```

**Veja Também**

[ST\\_Dump](#), [ST\\_LocateBetween](#), [ST\\_LocateBetweenElevations](#)

**8.12.6 ST\_LocateBetween**

**ST\_LocateBetween** — Return a derived geometry collection value with elements that match the specified range of measures inclusively. Polygonal elements are not supported.

**Synopsis**

geometria **ST\_LocateBetween**(geometria geomA, float8 measure\_start, float8 measure\_end, float8 offset);

**Descrição**

Return a derived geometry collection with elements that match the specified range of measures inclusively. Polygonal elements are not supported.

Se um deslocamento é fornecido, o resultado será o deslocamento para a direita ou para a esquerda da linha de entrada pelo número específico de unidades. Um deslocamento positivo será para a esquerda e um negativo para a direita.

Semântico é especificado por: ISO/IEC CD 13249-3:200x(E) - Text for Continuation CD Editing Meeting

Disponibilidade: 1.1.0 pelo nome antigo `ST_Locate_Between_Measures`.

Alterações: 2.0.0 - nas versões anteriores era chamado de `ST_Locate_Along_Measure`. O nome antigo foi menosprezado e será removido no futuro, mas ainda está disponível para compatibilidades anteriores.



This function supports M coordinates.

**Exemplos**

```
SELECT ST_AsText(the_geom)
      FROM
      (SELECT ST_LocateBetween(
            ST_GeomFromText('MULTILINESTRING M ((1 2 3, 3 4 2, 9 4 3),
            (1 2 3, 5 4 5))'),1.5, 3) As the_geom) As foo;

                                     st_asewkt
-----
GEOMETRYCOLLECTION M (LINESTRING M (1 2 3,3 4 2,9 4 3),POINT M (1 2 3))

--Geometry collections are difficult animals so dump them
--to make them more digestable
SELECT ST_AsText((ST_Dump(the_geom)).geom)
      FROM
      (SELECT ST_LocateBetween(
            ST_GeomFromText('MULTILINESTRING M ((1 2 3, 3 4 2, 9 4 3),
            (1 2 3, 5 4 5))'),1.5, 3) As the_geom) As foo;

                                     st_asewkt
-----
LINESTRING M (1 2 3,3 4 2,9 4 3)
POINT M (1 2 3)
```



**Veja Também**

[ST\\_Dump](#), [ST\\_LocateAlong](#), [ST\\_LocateBetweenElevations](#)

**8.12.7 ST\_LocateBetweenElevations**

`ST_LocateBetweenElevations` — Retorna um valor de geometria derivada (coleção) com elementos que inserem a variação específica de elevações. Apenas 3D, 4D LINESTRINGS e MULTILINESTRINGS são suportadas.

**Synopsis**

geometria `ST_LocateBetweenElevations`(geometria geom\_mline, float8 elevation\_start, float8 elevation\_end);

**Descrição**

Retorna um valor de geometria derivada (coleção) com elementos que inserem a variação específica de elevações. Apenas 3D, 3DM LINESTRINGS e MULTILINESTRINGS são suportadas.

Disponibilidade: 1.4.0



This function supports 3d and will not drop the z-index.

**Exemplos**

```
SELECT ST_AsEWKT(ST_LocateBetweenElevations(
    ST_GeomFromEWKT('LINESTRING(1 2 3, 4 5 6)'),2,4)) As ewelev;
-----
MULTILINESTRING((1 2 3,2 3 4))

SELECT ST_AsEWKT(ST_LocateBetweenElevations(
    ST_GeomFromEWKT('LINESTRING(1 2 6, 4 5 -1, 7 8 9)'),6,9)) As ewelev ←
;
-----
ewelev
-----
GEOMETRYCOLLECTION(POINT(1 2 6),LINESTRING(6.1 7.1 6,7 8 9))

--Geometry collections are difficult animals so dump them
--to make them more digestable
SELECT ST_AsEWKT((ST_Dump(the_geom)).geom)
    FROM
    (SELECT ST_LocateBetweenElevations(
        ST_GeomFromEWKT('LINESTRING(1 2 6, 4 5 -1, 7 8 9)'),6,9) As ←
        the_geom) As foo;
-----
st_asewkt
-----
POINT(1 2 6)
LINESTRING(6.1 7.1 6,7 8 9)
```

**Veja Também**

[ST\\_Dump](#)

### 8.12.8 ST\_InterpolatePoint

ST\_InterpolatePoint — Retorna o valor da dimensão de medida da geometria no ponto fechado para o ponto fornecido.

#### Synopsis

```
float8 ST_InterpolatePoint(geometria line, geometry ponto);
```

#### Descrição

Retorna o valor da dimensão de medida da geometria no ponto fechado para o ponto fornecido.

Disponibilidade: 2.0.0



This function supports 3d and will not drop the z-index.

#### Exemplos

```
SELECT ST_InterpolatePoint('LINESTRING M (0 0 0, 10 0 20)', 'POINT(5 5)');
 st_interpolatepoint
-----
10
```

#### Veja Também

[ST\\_AddMeasure](#), [ST\\_LocateAlong](#), [ST\\_LocateBetween](#)

### 8.12.9 ST\_AddMeasure

ST\_AddMeasure — Retorna uma geometria derivada com elementos de medida interpolados linearmente entre os pontos de início e de fim.

#### Synopsis

```
geometria ST_AddMeasure(geometria geom_mline, float8 measure_start, float8 measure_end);
```

#### Descrição

Retorna uma geometria derivada com elementos de medida interpolados linearmente entre os pontos de início e de fim. Se a geometria não tem nenhuma dimensão de medida, uma é adicionada. Se a geometria tem dimensão de medida, é sobre escrita com novos valores. Somente LINESTRINGS e MULTILINESTRINGS são suportadas.

Disponibilidade: 1.5.0



This function supports 3d and will not drop the z-index.

## Exemplos

```

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRING(1 0, 2 0, 4 0)'),1,4)) As ewelev;
-----
LINESTRINGM(1 0 1,2 0 2,4 0 4)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRING(1 0 4, 2 0 4, 4 0 4)'),10,40)) As ewelev;
-----
LINESTRING(1 0 4 10,2 0 4 20,4 0 4 40)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRINGM(1 0 4, 2 0 4, 4 0 4)'),10,40)) As ewelev;
-----
LINESTRINGM(1 0 10,2 0 20,4 0 40)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('MULTILINESTRINGM((1 0 4, 2 0 4, 4 0 4),(1 0 4, 2 0 4, 4 0 4)'),10,70)) As ←
      ewelev;
-----
MULTILINESTRINGM((1 0 10,2 0 20,4 0 40),(1 0 40,2 0 50,4 0 70))

```

## 8.13 Suporte Temporal

### 8.13.1 ST\_IsValidTrajectory

`ST_IsValidTrajectory` — Retorna `true` se a geometria possui uma trajetória válida.

#### Synopsis

boolean `ST_IsValidTrajectory`(geometry line);

#### Descrição

Diz se uma geometria codifica uma trajetória válida. Trajetórias válidas são codificadas como `LINESTRING` com valor de `M`, crescente para cada vértice.

Espera-se que trajetórias válidas possuam como entrada pesquisas espaço-temporais, como [ST\\_ClosestPointOfApproach](#)

Disponibilidade: 2.2.0



This function supports 3d and will not drop the z-index.

#### Exemplos

```

-- A valid trajectory
SELECT ST_IsValidTrajectory(ST_MakeLine(
  ST_MakePointM(0,0,1),
  ST_MakePointM(0,1,2))
);

```

```

t

-- An invalid trajectory
SELECT ST_IsValidTrajectory(ST_MakeLine(ST_MakePointM(0,0,1), ST_MakePointM(0,1,0)));
NOTICE: Measure of vertex 1 (0) not bigger than measure of vertex 0 (1)
st_isvalidtrajectory
-----
f

```

## Veja também

[ST\\_ClosestPointOfApproach](#)

### 8.13.2 ST\_ClosestPointOfApproach

`ST_ClosestPointOfApproach` — Retorna a medida em que pontos interpolados na linha são mais próximos.

#### Synopsis

```
float8 ST_ClosestPointOfApproach(geometry track1, geometry track2);
```

#### Descrição

Retorna a menor medida em que o ponto interpolado na linha possuem a menor distância. Entradas devem ser trajetórias válidas como checadas por [ST\\_IsValidTrajectory](#). Retorna null se as trajetórias não se sobrepõe no intervalo M.

Veja [ST\\_LocateAlong](#) para buscar os pontos em uma determinada medida.

Disponibilidade: 2.2.0



This function supports 3d and will not drop the z-index.

#### Exemplos

```

-- Return the time in which two objects moving between 10:00 and 11:00
-- are closest to each other and their distance at that point
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0, 10 0 5)::geometry,
    extract(epoch from '2015-05-26 10:00'::timestampz),
    extract(epoch from '2015-05-26 11:00'::timestampz)
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10, 12 1 2)::geometry,
    extract(epoch from '2015-05-26 10:00'::timestampz),
    extract(epoch from '2015-05-26 11:00'::timestampz)
  ) b
), cpa AS (
  SELECT ST_ClosestPointOfApproach(a,b) m FROM inp
), points AS (
  SELECT ST_Force3DZ(ST_GeometryN(ST_LocateAlong(a,m),1)) pa,
    ST_Force3DZ(ST_GeometryN(ST_LocateAlong(b,m),1)) pb
  FROM inp, cpa
)
SELECT to_timestamp(m) t,
  ST_Distance(pa,pb) distance
FROM points, cpa;

```

t	distance
2015-05-26 10:45:31.034483+02	1.96036833151395

**Veja também**

[ST\\_IsValidTrajectory](#), [ST\\_DistanceCPA](#), [ST\\_LocateAlong](#), [ST\\_AddMeasure](#)

**8.13.3 ST\_DistanceCPA**

`ST_DistanceCPA` — Retorna a distância entre os pontos mais próximos de uma aproximação em duas trajetórias.

**Synopsis**

```
float8 ST_DistanceCPA(geometry track1, geometry track2);
```

**Descrição**

Retorna a menor distância entre dois objetos em movimento que tiveram um do outro. Entradas devem ser trajetórias válidas, como checadas por [ST\\_IsValidTrajectory](#). Retorna null se as trajetórias não se interseccionam no intervalo M.

Disponibilidade: 2.2.0



This function supports 3d and will not drop the z-index.

**Exemplos**

```
-- Return the minimum distance of two objects moving between 10:00 and 11:00
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0, 10 0 5)::geometry,
    extract(epoch from '2015-05-26 10:00'::timestampz),
    extract(epoch from '2015-05-26 11:00'::timestampz)
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10, 12 1 2)::geometry,
    extract(epoch from '2015-05-26 10:00'::timestampz),
    extract(epoch from '2015-05-26 11:00'::timestampz)
  ) b
)
SELECT ST_DistanceCPA(a,b) distance FROM inp;

   distance
-----
1.96036833151395
```

**Veja também**

[ST\\_IsValidTrajectory](#), [ST\\_ClosestPointOfApproach](#), [ST\\_AddMeasure](#), [ST\\_Distance](#)

**8.13.4 ST\_CPAWithin**

`ST_CPAWithin` — Retorna verdadeiro se os pontos mais próximos da trajetória estão até a distância especificada.

## Synopsis

```
float8 ST_CPWithin(geometry track1, geometry track2, float8 maxdist);
```

## Descrição

Checa se dois objetos em movimento chegaram até a distância especificada.

Entradas devem ser trajetórias válidas, conforme checadas por [ST\\_IsValidTrajectory](#). Retorna falso se as trajetórias não se interseccionam no intervalo M.

Disponibilidade: 2.2.0



This function supports 3d and will not drop the z-index.

## Exemplos

```
WITH inp AS ( SELECT
  ST_AddMeasure('LINESTRING Z (0 0 0, 10 0 5)::geometry,
    extract(epoch from '2015-05-26 10:00'::timestampz),
    extract(epoch from '2015-05-26 11:00'::timestampz)
  ) a,
  ST_AddMeasure('LINESTRING Z (0 2 10, 12 1 2)::geometry,
    extract(epoch from '2015-05-26 10:00'::timestampz),
    extract(epoch from '2015-05-26 11:00'::timestampz)
  ) b
)
SELECT ST_CPWithin(a,b,2), ST_DistanceCPA(a,b) distance FROM inp;
```

st_cpawithin	distance
t	1.96521473776207

## Veja também

[ST\\_IsValidTrajectory](#), [ST\\_ClosestPointOfApproach](#), [ST\\_DistanceCPA](#), [|](#)

## 8.14 Suporte de longas transações

Esse módulo e funções pl/pgsql associadas foram implementadas para fornecer suporte de fechamento requerido pela especificação [Web Feature Service](#).



### Note

Os usuários devem usar [serializable transaction level](#), senão o mecanismo de fechamento irá quebrar.

### 8.14.1 AddAuth

AddAuth — Adiciona um token de autorização para ser usado em transação atual.

## Synopsis

boolean **AddAuth**(text auth\_token);

## Descrição

Adiciona um token de autorização para ser usado em transação atual.

Cria/adiciona a uma temp table chamada temp\_lock\_have\_table, o identificador da transação atual e chave de autorização token.

Disponibilidade: 1.1.3

## Exemplos

```
SELECT LockRow('towns', '353', 'priscilla');
      BEGIN TRANSACTION;
          SELECT AddAuth('joey');
          UPDATE towns SET the_geom = ST_Translate(the_geom,2,2) WHERE gid = ←
              353;
      COMMIT;

---Error--
ERROR: UPDATE where "gid" = '353' requires authorization 'priscilla'
```

## Veja também.

[LockRow](#)

## 8.14.2 CheckAuth

CheckAuth — Cria trigger em uma table para prevenir/permitir atualizações e exclusões de filas baseado em token de autorização.

## Synopsis

integer **CheckAuth**(text a\_schema\_name, text a\_table\_name, text a\_key\_column\_name);

integer **CheckAuth**(text a\_table\_name, text a\_key\_column\_name);

## Descrição

Cria trigger em uma table para prevenir/permitir atualizações e exclusões de linhas baseado em token de autorização. identifica filas utilizando <rowid\_col> column.

Se a\_schema\_name não passou, então, pesquise por tables no esquema atual.



### Note

Se um trigger de autorização existe nessa table, função falha

Se o suporte de transação não estiver ativado, a função abre uma exceção.

Disponibilidade: 1.1.3

## Exemplos

```
SELECT CheckAuth('public', 'towns', 'gid');
      result
-----
0
```

Veja também.

[AtivarLongasTransações](#)

### 8.14.3 DesativarLongasTransações

`DesativarLongasTransações` — Desativa suporte de transações longas. Essa função remove as tables de metadados ad transação longa e derruba todos os triggers anexados às tables lock-checked.

#### Synopsis

```
text DisableLongTransactions();
```

#### Descrição

Desativa suporte de transações longas. Essa função remove as tables de metadados ad transação longa e derruba todos os triggers anexados às tables lock-checked.

Derruba a meta table chamada `authorization_table` e uma view chamada `authorized_tables` e todos os triggers chamados `checkauthtrigger`

Disponibilidade: 1.1.3

## Exemplos

```
SELECT DisableLongTransactions();
--result--
Long transactions support disabled
```

Veja também.

[AtivarLongasTransações](#)

### 8.14.4 AtivarLongasTransações

`AtivarLongasTransações` — Ativa o suporte de longas transações. Essa função cria as tables de metadados que são requeridas, precisa ser chamada uma vez antes de usar outras funções nessa seção. Chamá-la duas vezes não tem problema.

#### Synopsis

```
text EnableLongTransactions();
```

---



**Descrição**

Ativa o suporte de longas transações. Essa função cria as tables de metadados que são requeridas, precisa ser chamada uma vez antes de usar outras funções nessa seção. Chamá-la duas vezes não tem problema.

Cria uma meta table chamada `authorization_table` e uma view chamada `authorized_tables`

Disponibilidade: 1.1.3

**Exemplos**

```
SELECT EnableLongTransactions();
--result--
Long transactions support enabled
```

**Veja também.**

[DesativarLongasTransações](#)

**8.14.5 LockRow**

LockRow — Configurar fechamento/autorização para uma fileira específica na table

**Synopsis**

integer **LockRow**(text a\_schema\_name, text a\_table\_name, text a\_row\_key, text an\_auth\_token, timestamp expire\_dt);

integer **LockRow**(text a\_table\_name, text a\_row\_key, text an\_auth\_token, timestamp expire\_dt);

integer **LockRow**(text a\_table\_name, text a\_row\_key, text an\_auth\_token);

**Descrição**

Configurar fechamento/autorização para uma fileira específica na table <authid> é uma valor de texto, <expires> é uma marca temporal padrão para agora()+1hora. Retorna 1 se o fechamento tiver sido designado, 0 senão (já fechado por outra autorização)

Disponibilidade: 1.1.3

**Exemplos**

```
SELECT LockRow('public', 'towns', '2', 'joey');
LockRow
-----
1

--Joey has already locked the record and Priscilla is out of luck
SELECT LockRow('public', 'towns', '2', 'priscilla');
LockRow
-----
0
```

**Veja também.**

[UnlockRows](#)

## 8.14.6 UnlockRows

UnlockRows — Remove todos os fechamentos armazenados por id de autorização específica Retorna o número de fechamentos liberados.

### Synopsis

```
integer UnlockRows(text auth_token);
```

### Descrição

Remove todos os fechamentos armazenados por id de autorização específica Retorna o número de fechamentos liberados.

Disponibilidade: 1.1.3

### Exemplos

```
SELECT LockRow('towns', '353', 'priscilla');
       SELECT LockRow('towns', '2', 'priscilla');
       SELECT UnlockRows('priscilla');
       UnlockRows
       -----
              2
```

Veja também.

[LockRow](#)

## 8.15 Funções Variadas

### 8.15.1 ST\_Accum

ST\_Accum — Agregado. Constrói um banco de dados de geometrias.

### Synopsis

```
geometry[] ST_Accum(geometry set geomfield);
```

### Descrição

Agregado. Constrói um banco de dados de geometrias.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

```
SELECT (ST_Accum(the_geom)) As all_em, ST_AsText((ST_Accum(the_geom))[1]) As grabone,
(ST_Accum(the_geom))[2:4] as grab_rest
      FROM (SELECT ST_MakePoint(a*CAST(random()*10 As integer), a*CAST(↵
random()*10 As integer), a*CAST(random()*10 As integer)) As ↵
the_geom
      FROM generate_series(1,4) a) As foo;

all_em|grabone   | grab_rest
-----+-----
{0101000080000000000000000000144000000000000024400000000000001040:
0101000080000000000000000000
000184000000000000000002C4000000000000003040:
01010000800000000000000000354000000000000038400000000000001840:
010100008000000000000000404000000000000003C400000000000003040} |
POINT(5 10) | {010100008000000000000000001840000000000002C4000000000000003040:
0101000080000000000000000035400000000000003840000000000001840:
01010000800000000000000040400000000000003C400000000000003040}
(1 row)
```

## Veja também.

[ST\\_Collect](#)

## 8.15.2 Caixa2D

Caixa2D — Retorna uma CAIXA2D representando a extensão máxima da geometria.

### Synopsis

```
box2d Box2D(geometry geomA);
```

### Descrição

Retorna uma CAIXA2D representando a extensão máxima da geometria.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

```
SELECT Box2D(ST_GeomFromText('LINESTRING(1 2, 3 4, 5 6)'));
      box2d
      -----
      BOX(1 2,5 6)
```

```
SELECT Box2D(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 ↵
  150406)'));
box2d
-----
BOX(220186.984375 150406,220288.25 150506.140625)
```

**Veja também.**

[Caixa3D](#), [ST\\_GeomFromText](#)

### 8.15.3 Caixa3D

Caixa3D — Retorna uma CAIXA3D representando a extensão máxima da geometria.

#### Synopsis

box3d **Box3D**(geometry geomA);

#### Descrição

Retorna uma CAIXA3D representando a extensão máxima da geometria.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.

#### Exemplos

```
SELECT Box3D(ST_GeomFromEWKT('LINESTRING(1 2 3, 3 4 5, 5 6 5)'));
Box3d
-----
BOX3D(1 2 3,5 6 5)

SELECT Box3D(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 1,220227 ↵
  150406 1)'));
Box3d
-----
BOX3D(220227 150406 1,220268 150415 1)
```

**Veja também.**

[Caixa2D](#), [ST\\_GeomFromEWKT](#)

## 8.15.4 ST\_EstimatedExtent

ST\_EstimatedExtent — Retorna a extensão "estimada" da table espacial dada. O estimada é tirado das estatísticas da coluna geométrica. O esquema atual será usado se não especificado.

### Synopsis

```
box2d ST_EstimatedExtent(text schema_name, text table_name, text geocolumn_name, boolean parent_ony);
```

```
box2d ST_EstimatedExtent(text schema_name, text table_name, text geocolumn_name);
```

```
box2d ST_EstimatedExtent(text table_name, text geocolumn_name);
```

### Descrição

Retorna a extensão "estimada" da table espacial dada. O estimada é tirado das estatísticas da coluna geométrica. O esquema atual será usado se não especificado. O comportamento padrão é usar estatísticas coletadas das children tables (tables HERDAM) se disponíveis. Se 'parent\_ony' é colocado VERDADEIRO, somente as estatísticas da dada table são usadas e as children tables são ignoradas.

Para o PostgreSQL >= 8.0.0, estatísticas são juntadas por VACUUM ANALYZE e a extensão resultante será 95% da real.



#### Note

Na falta de estatísticas (table vazia ou nenhuma ANÁLISE), essa função retorna NULA. Anterior a versão 1.5.4 uma exceção foi descartada.

Para o PostgreSQL < 8.0.0, estatísticas são juntadas pelo update\_geometry\_stats() e a extensão resultante será exata.

Disponibilidade: 1.0.0

Alterações: 2.1.0 para 2.0.x foi chamada ST\_Extended\_Extent.



This method supports Circular Strings and Curves

### Exemplos

```
SELECT ST_EstimatedExtent('ny', 'edges', 'the_geom');
--result--
BOX(-8877653 4912316,-8010225.5 5589284)

SELECT ST_EstimatedExtent('feature_poly', 'the_geom');
--result--
BOX(-124.659652709961 24.6830825805664,-67.7798080444336 49.0012092590332)
```

### Veja também.

[ST\\_Extent](#)

## 8.15.5 ST\_Expand

ST\_Expand — A caixa limitadora retorna expandida em todas as direções da caixa da geometria de entrada. Utiliza precisão dobrada

## 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);
```

## Descrição

Essa função retorna uma caixa limitadora expandida da caixa da geometria de entrada ou especificando uma única distância com a qual a caixa deveria ser expandida em todas as direções ou especificando uma distância de extensão para cada direção.

Juntamente com a geometria de versão `ST_Expand`, a qual é usada normalmente, variantes que aceitam e produzem tipos de dados internos `CAIXA2D` e `CAIXA3D` são fornecidas.

A `ST_Expand` tem o conceito parecido com `ST_Buffer`, exceto que, enquanto o buffer expande a geometria em todas as direções, a `ST_Expand` expande a caixa limitadora em uma quantidade de unidade `x,y,z`.

As unidades estão nas medidas do sistema de referência espacial em uso indicado pelo `SRID`.



### Note

Pre 1.3, `ST_Expand` era utilizada em conjunto com distância para fazer pesquisas indexable. Algo da forma `the_geom && ST_Expand('POINT(10 20)', 10) AND ST_Distance(the_geom, 'POINT(10 20)') < 10` Post 1.2, isso foi substituído pela construção mais fácil `ST_DWithin`.



### Note

Disponibilidade: 1.5.0 comportamento alterado para saída de precisão dupla ao invés das coordenadas float4.  
 Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.  
 Melhorias: 2.3.0 suporte para expandir uma caixa por diferentes porções em dimensões diferentes, foi adicionado.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos



### Note

Exemplos abaixo utilizam o US National Atlas Equal Area (`SRID=2163`) o qual é uma projeção em metros

```
--10 meter expanded box around bbox of a linestring
SELECT CAST(ST_Expand(ST_GeomFromText('LINESTRING(2312980 110676,2312923 110701,2312892  ←
  110714)', 2163),10) As box2d);
                                st_expand
-----
BOX(2312882 110666,2312990 110724)

--10 meter expanded 3d box of a 3d box
SELECT ST_Expand(CAST('BOX3D(778783 2951741 1,794875 2970042.61545891 10)' As box3d),10)
```

```
----- st_expand
```

```
BOX3D(778773 2951731 -9,794885 2970052.61545891 20)
```

```
--10 meter geometry astext rep of a expand box around a point geometry
```

```
SELECT ST_AsEWKT(ST_Expand(ST_GeomFromEWKT('SRID=2163;POINT(2312980 110676)'),10));
```

```
st_asewkt ↔
```

```
-----
```

```
SRID=2163;POLYGON((2312970 110666,2312970 110686,2312990 110686,2312990 110666,2312970 110666))
```

### Veja também.

[ST\\_AsEWKT](#), [ST\\_Buffer](#), [ST\\_DWithin](#), [ST\\_GeomFromEWKT](#), [ST\\_GeomFromText](#), [ST\\_SRID](#)

## 8.15.6 ST\_Extent

`ST_Extent` — uma função agregada que retorna a caixa limitadora que estabelece linhas de geometrias.

### Synopsis

```
box2d ST_Extent(geometry set geomfield);
```

### Descrição

A `ST_Extent` retorna uma caixa limitadora que envolve um conjunto de geometrias. A função `ST_Extent` é uma função "agregada" na terminologia SQL. Isso significa que ela atua em listas de dados, da mesma forma que as funções `SUM()` e `AVG()` fazem.

Já que retorna uma caixa limitadora, as unidades espaciais estão na medida do sistema de referência espacial em uso indicado pelo SRID

A `ST_Extent` parece em conceito com o Oracle Spatial/Locator's `SDO_AGGR_MBR`



#### Note

Já que a `ST_Extent` retorna uma caixa limitadora, os metadados do SRID são perdidos. Use a `ST_SetSRID` para forçar ela de volta para uma geometria com metadados SRID. As coordenadas estão nas unidades do spatial ref das geometrias originais.



#### Note

A `ST_Extent` retornará caixas apenas com os componentes x e y mesmo com as coordenadas (x,y,z) das geometrias. Para manter x,y,z use `ST_3DExtent`.



#### Note

Disponibilidade: 1.4.0

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos



### Note

Exemplos abaixo use Massachusetts State Plane ft (SRID=2249)

```
SELECT ST_Extent(the_geom) as bextent FROM sometable;
                                st_bextent
-----
BOX(739651.875 2908247.25,794875.8125 2970042.75)

--Return extent of each category of geometries
SELECT ST_Extent(the_geom) as bextent
FROM sometable
GROUP BY category ORDER BY category;

                                bextent                                |          name
-----+-----
BOX(778783.5625 2951741.25,794875.8125 2970042.75) | A
BOX(751315.8125 2919164.75,765202.6875 2935417.25) | B
BOX(739651.875 2917394.75,756688.375 2935866)      | C

--Force back into a geometry
-- and render the extended text representation of that geometry
SELECT ST_SetSRID(ST_Extent(the_geom),2249) as bextent FROM sometable;

                                bextent
-----
SRID=2249;POLYGON((739651.875 2908247.25,739651.875 2970042.75,794875.8125 2970042.75,
794875.8125 2908247.25,739651.875 2908247.25))
```

Veja também.

[ST\\_AsEWKT](#), [ST\\_3DExtent](#), [ST\\_SetSRID](#), [ST\\_SRID](#)

### 8.15.7 ST\_3DExtent

ST\_3DExtent — uma função agregada que retorna a caixa3D que estabelece linhas de geometria.

#### Synopsis

box3d **ST\_3DExtent**(geometry set geomfield);



## Descrição

A `ST_3DExtent` retorna uma caixa3d (inclui a coordenada Z) que envolve um conjunto de geometrias. A função `ST_3DExtent` é "agregada" na terminologia SQL. Isso significa que ela opera em listas de dados, da mesma maneira que as funções `SUM()` e `AVG()`.

Já que retorna uma caixa limitadora, as unidades espaciais estão na medida do sistema de referência espacial em uso indicado pelo SRID



### Note

Já que a `ST_3DExtent` retorna uma caixa limitadora, os metadados do SRID são perdidos. Use a `ST_SetSRID` para forçar ela de volta para uma geometria com metadados SRID. As coordenadas estão nas unidades do spatial ref das geometrias originais.

Melhorias: 2.0.0 suporte para superfícies poliédricas, triângulos e TINs introduzido.

Alterações: 2.0.0 Nas versões anteriores era chamado de `ST_Extent3D`



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

## Exemplos

```
SELECT ST_3DExtent(foo.the_geom) As b3extent
FROM (SELECT ST_MakePoint(x,y,z) As the_geom
      FROM generate_series(1,3) As x
      CROSS JOIN generate_series(1,2) As y
      CROSS JOIN generate_series(0,2) As Z) As foo;
-----
b3extent
BOX3D(1 1 0,3 2 2)

--Get the extent of various elevated circular strings
SELECT ST_3DExtent(foo.the_geom) As b3extent
FROM (SELECT ST_Translate(ST_Force_3DZ(ST_LineToCurve(ST_Buffer(ST_MakePoint(x,y),1))),0,0, ←
      z) As the_geom
      FROM generate_series(1,3) As x
      CROSS JOIN generate_series(1,2) As y
      CROSS JOIN generate_series(0,2) As Z) As foo;
-----
b3extent
BOX3D(1 0 0,4 2 2)
```

Veja também.

[ST\\_Extent](#), [ST\\_Force3DZ](#)

### 8.15.8 Find\_SRID

**Find\_SRID** — A sintaxe é `find_srid(a_db_schema, a_table, a_column)` e a função retorna o SRID inteiro das colunas especificadas buscando através da table `GEOMETRY_COLUMNS`.

#### Synopsis

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

#### Descrição

A sintaxe é `find_srid(<db/schema>, <table>, <column>)` e a função retorna o SRID inteiro das colunas especificadas buscando através da table `GEOMETRY_COLUMNS`. Se a coluna geométrica não foi adicionada corretamente com a função `AddGeometryColumns()`, essa função também não funcionará.

#### Exemplos

```
SELECT Find_SRID('public', 'tiger_us_state_2007', 'the_geom_4269');
find_srid
-----
4269
```

Veja também.

[ST\\_SRID](#)

### 8.15.9 ST\_MemSize

**ST\_MemSize** — Retorna a quantidade de espaço (em bytes) que a geometria pega.

#### Synopsis

```
integer ST_MemSize(geometry geomA);
```

#### Descrição

Retorna a quantidade de espaço (em bytes) que a geometria pega.

Isso é um ótimo elogio para o PostgreSQL construído nas funções `pg_column_size`, `pg_size_pretty`, `pg_relation_size`, `pg_total_relation_size`.

#### Note



`pg_relation_size` que fornece o tamanho em bytes de uma table, talvez retorne com o tamanho menor que `ST_MemSize`. Isso acontece porque o `pg_relation_size` não adiciona contribuição toasted table e geometrias largas são guardadas em TOAST tables.

`pg_total_relation_size` - inclui, a table, as toasted tables, e os indexes.

`pg_column_size` retorna quanto espaço uma geometria iria tomar em uma coluna considerando compressão, para talvez ser menor que `ST_MemSize`



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Alterações: 2.2.0 nome mudado para `ST_MemSize` para seguir a convenção de nomes. Nas versões anteriores essa função era chamada de `ST_Mem_Size`, o nome antigo ainda continua disponível.

## Exemplos

```
--Return how much byte space Boston takes up in our Mass data set
SELECT pg_size_pretty(SUM(ST_MemSize(the_geom))) as totgeomsum,
pg_size_pretty(SUM(CASE WHEN town = 'BOSTON' THEN ST_MemSize(the_geom) ELSE 0 END)) As bossum,
CAST(SUM(CASE WHEN town = 'BOSTON' THEN ST_MemSize(the_geom) ELSE 0 END)*1.00 /
SUM(ST_MemSize(the_geom))*100 As numeric(10,2)) As perbos
FROM towns;
```

totgeomsum	bossum	perbos
1522 kB	30 kB	1.99

```
SELECT ST_MemSize(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'));
---
```

73

```
--What percentage of our table is taken up by just the geometry
SELECT pg_total_relation_size('public.neighborhoods') As fulltable_size, sum(ST_MemSize(the_geom)) As geomsizes,
sum(ST_MemSize(the_geom))*1.00/pg_total_relation_size('public.neighborhoods')*100 As pergeom
FROM neighborhoods;
fulltable_size geomsizes pergeom
-----
262144          96238          36.71188354492187500000
```

Veja também.

### 8.15.10 ST\_PointInsideCircle

`ST_PointInsideCircle` — Is the point geometry inside the circle defined by `center_x`, `center_y`, `radius`

#### Synopsis

boolean `ST_PointInsideCircle`(geometry a\_point, float center\_x, float center\_y, float radius);

## Descrição

A sintaxe para essa função é `ST_PointInsideCircle(<geometry>,<circle_center_x>,<circle_center_y>,<radius>)`. Retorna verdadeiro se a geometria for um ponto e estiver dentro do círculo. Retorna falso senão.



### Note

Isso só funciona para pontos como o nome sugere

Disponibilidade: 1.2

Alterações: 2.2.0 Em versões anteriores era chamada de `ST_Point_Inside_Circle`

## Exemplos

```
SELECT ST_PointInsideCircle(ST_Point(1,2), 0.5, 2, 3);
 st_pointinsidecircle
-----
t
```

Veja também.

[ST\\_DWithin](#)

## 8.16 Funções excepcionais

Estas funções são raramente utilizadas que devem somente serem usadas se seus estão corrompidos de alguma maneira. Elas são usadas para resolução de problemas e correção de algumas coisa, que em circunstâncias normais, nunca aconteceriam.

### 8.16.1 PostGIS\_AddBBox

`PostGIS_AddBBox` — Adicionar o retângulo envolvente a geometria.

#### Synopsis

```
geometry PostGIS_AddBBox(geometry geomA);
```

#### Descrição

Adiciona um retângulo envolvente a geometria. Isto faz pesquisas baseadas em retângulos envolventes mais rápidas, mas aumentará o tamanho da geometria.



### Note

Retângulos envolventes são geometrias adicionadas automaticamente, portanto, em geral, isto não é necessário, a menos que o retângulo envolvente foi corrompido de alguma forma ou você possui uma instalação antiga que não possui suporte a retângulos envolventes. Então é necessário dropar os retângulos antigos e readicioná-los.



This method supports Circular Strings and Curves

## Exemplos

```
UPDATE sometable
SET the_geom = PostGIS_AddBBox(the_geom)
WHERE PostGIS_HasBBox(the_geom) = false;
```

## Veja também

[PostGIS\\_DropBBox](#), [PostGIS\\_HasBBox](#)

### 8.16.2 PostGIS\_DropBBox

PostGIS\_DropBBox — Exclui o cache de retângulos envolventes da geometria.

## Synopsis

geometry **PostGIS\_DropBBox**(geometry geomA);

## Descrição

Exclui o cache de retângulos envolventes de uma geometria. Isto reduz o tamanho da geometria, mas torna as pesquisas baseadas em retângulos envolventes mais lentas. Um sinal claro de um cache corrompido é quando suas pesquisas de `ST_Intersects` e outras relações espaciais deixam de fora geometrias que deveriam retornar verdadeiro.

### Note



Retângulos envolventes são automaticamente adicionados as geometrias e melhoram a velocidade de pesquisas, então em geral, isto não é necessário, a menos que seu cache de retângulos envolventes tenha sido corrompido ou você possua uma instalação que não suportava esta funcionalidade. Este tipo de corrupção foi observado nas versões 8.3-8.3.6, onde os retângulos envolventes cacheados não eram sempre recalculados durante a alteração de uma geometria e o upgrade para uma nova versão, sem o uso de um processo de dump/reload não irá corrigir os retângulos corrompidos. Você pode corrigir isto manualmente, utilizando a função abaixo ou realizar um backup com processo de dump/reload.



This method supports Circular Strings and Curves

## Exemplos

```
--Este exemplo exclui os retângulos envolventes que não estão corretos.
--Aplique um ST_AsBinary antes de usar Box2D para forçar o cálculo de novo retângulo
--Retorna o retângulo cacheado
UPDATE sometable
SET the_geom = PostGIS_DropBBox(the_geom)
WHERE Not (Box2D(ST_AsBinary(the_geom)) = Box2D(the_geom));

UPDATE sometable
SET the_geom = PostGIS_AddBBox(the_geom)
WHERE Not PostGIS_HasBBOX(the_geom);
```

**Veja também**

[PostGIS\\_AddBBox](#), [PostGIS\\_HasBBox](#), [Caixa2D](#)

**8.16.3 PostGIS\_HasBBox**

PostGIS\_HasBBox — Retorna verdadeiro se o retângulo desta geometria está cacheado. FALSO caso contrário.

**Synopsis**

```
boolean PostGIS_HasBBox(geometry geomA);
```

**Descrição**

Retorna verdadeiro se o retângulo envolvendo desta geometria está cacheado, falso caso contrário. Use [PostGIS\\_AddBBox](#) e [PostGIS\\_DropBBox](#) para controlar o cache.



This method supports Circular Strings and Curves

**Exemplos**

```
SELECT the_geom
FROM sometable WHERE PostGIS_HasBBox(the_geom) = false;
```

**Veja também**

[PostGIS\\_AddBBox](#), [PostGIS\\_DropBBox](#)

## Chapter 9

# Referência Raster

As funções abaixo são as que um usuário do PostGIS Raster vai precisar e que estão disponíveis no momento no PostGIS Raster. Existem outras funções que precisam das funções suporte para os objetos raster que não são de uso geral.

`raster` é um novo tipo PostGIS para armazenar e analisar dados raster.

Para carregar rasters de arquivos raster, por favor, vá para [Section 5.1](#)

Para os exemplos nessa referência, nós usaremos uma table raster de rasters simulados - Formados com o seguinte código

```
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, ('01000003009A999999999999A93F9A9999999999A9BF000000E02B274A' ||
'410000000077195641000000000000000000000000000000000000000000000000 ←
FFFFFFF050005000400FDFFDFEFDFEFDFEFDFEFDF9FAFEF' ||
' ←
EFCF9FBFDFFDFEFDFCFAFEFEF04004E627AADD16076B4F9FE6370A9F5FE59637AB0E54F58617087040046566487A1506
')::raster);
```

## 9.1 Tipos de suporte de dados raster

### 9.1.1 geomval

**geomval** — Um tipo de dado espacial com dois campos - *geom* (possuindo objeto geométrico) e *val* (possuindo um valor de pixel de precisão dupla de uma banda raster).

#### Descrição

**geomval** é uma mistura de tipo de dados que consiste em um objeto de geometria referenciado pelo campo *.geom* e *val*, um valor de precisão dupla que representa o valor do pixel em uma localização específica de geometria em uma banda raster. É usado por `ST_DumpAsPolygon` e a família de interseção raster de funções como um tipo de saída para explodir uma banda raster em polígonos.

#### Veja também

Section [14.6](#)

### 9.1.2 addbandarg

**addbandarg** — Um tipo composto usado como entrada na função `ST_AddBand` definindo os atributos e valor inicial da nova banda.

#### Descrição

Um tipo composto usado como entrada na função `ST_AddBand` definindo os atributos e valor inicial da nova banda.

***index* integer** Valor de 1-base indicando a posição onde a nova banda será adicionada no meio das bandas do raster. Se NULO, a nova banda será adicionada no fim das bandas do raster.

***pixeltype* text** tipo do pixel da nova banda. Um dos tipos de pixel definidos como descrito em: [ST\\_BandPixelType](#).

***initialvalue* double precision** Valor inicial que todos os pixels da nova banda serão definidos.

***nodataval* double precision** Valor NODATA da nova banda. Se NULA, a nova banda terá uma valor NODATA assinado.

#### Veja também

[ST\\_AddBand](#)

### 9.1.3 rastbandarg

**rastbandarg** — Um tipo composto para usar quando for preciso expressar um raster e um índice de banda desse raster.

#### Descrição

Um tipo composto para usar quando for preciso expressar um raster e um índice de banda desse raster.

***rast* raster** O raster em questão/

***nband* integer** Valor 1-base indicando a banda do raster



**Veja também**

[Funções retorno de mapa algébrico embutido](#)

**9.1.4 raster**

raster — raster spatial data type.

**Descrição**

raster is a spatial data type used to represent raster data such as those imported from JPEGs, TIFFs, PNGs, digital elevation models. Each raster has 1 or more bands each having a set of pixel values. Rasters can be georeferenced.

**Note**

Requer que o PostGIS esteja compilado com o suporte GDAL. Os rasters, atualmente, podem ser convertidos implicitamente para geometria, mas a conversão retorna a `ST_ConvexHull` do raster. Este auto casting pode ser removido em futuro próximo, então não confie muito nisto.

**Comportamento Casting**

Essa seção lista os casts automáticos bem como os explícitos permitidos para esse tipo de dados

Cast To	Comportamento
geometria	automático

**Veja também**

[Chapter 9](#)

**9.1.5 reclassarg**

reclassarg — Um tipo composto usado como entrada dentro da função `ST_Reclass` definindo o comportamento da reclassificação.

**Descrição**

Um tipo composto usado como entrada dentro da função `ST_Reclass` definindo o comportamento da reclassificação.

***nband integer*** O número banda para banda para reclassificar.

***reclassexpr text*** expressão de variação consistindo em mapeamentos `range:map_range` delimitados por vírgulas. `:` para definir mapeamento que esclarece como mapear valores antigos de banda para novos. `(` means `>`, `)` significa menor que, `]` < ou igual, `[` significa `>` ou igual

1. `[a-b] = a <= x <= b`
2. `(a-b) = a < x <= b`
3. `[a-b) = a <= x < b`
4. `(a-b) = a < x < b`

( notação é opcional então `a-b` significa o mesmo que `(a-b)`)

**pixeltype text** Um dos tipos de pixel definidos como descrito em: [ST\\_BandPixelType](#)

**nodataval double precision** Valor para tratar como sem dados. Para saídas de imagens que suportam transparência, essas serão em branco.

#### Exemplo: Reclassificar banda 2 como um 8BUI onde 255 é o valor sem dados

```
SELECT ROW(2, '0-100:1-10, 101-500:11-150,501 - 10000: 151-254', '8BUI', 255)::reclassarg;
```

#### Exemplo: Reclassificar banda 1 como um 1BB e nenhum valor sem dados definido

```
SELECT ROW(1, '0-100]:0, (100-255:1', '1BB', NULL)::reclassarg;
```

#### Veja também

[ST\\_Reclass](#)

### 9.1.6 summarystats

**summarystats** — Um tipo composto retornado pelas funções [ST\\_SummaryStats](#) e [ST\\_SummaryStatsAgg](#).

#### Descrição

Um tipo composto retornado pelas funções [ST\\_SummaryStats](#) e [ST\\_SummaryStatsAgg](#).

**count integer** Número de pixels contados para as estatísticas resumo.

**sum double precision** Resumo de todos os valores contados de pixels.

**mean double precision** Significado aritmético de todos os valores de pixels.

**stddev double precision** Divergência padrão de todos os valores contados de pixels.

**min double precision** Valor mínimo dos valores dos pixels contados.

**max double precision** Valor máximo dos valores dos pixels contados.

#### Veja também

[ST\\_SummaryStats](#), [ST\\_SummaryStatsAgg](#)

### 9.1.7 unionarg

**unionarg** — Um tipo composto usado como entrada dentro da função [ST\\_Union](#) definindo as bandas a serem processadas e o comportamento da operação UNIÃO.

#### Descrição

Um tipo composto usado como entrada dentro da função [ST\\_Union](#) definindo as bandas a serem processadas e o comportamento da operação UNIÃO.

**nband integer** Valor 1-baseado indicando a banda de cada raster de entrada a ser processado.

**uniontype text** Tipo de operação de UNIÃO. Um dos tipos definidos como descritos em [ST\\_Union](#).

Veja também

[ST\\_Union](#)

## 9.2 Gerenciamento Raster

### 9.2.1 AddRasterConstraints

`AddRasterConstraints` — Adds raster constraints to a loaded raster table for a specific column that constrains spatial ref, scaling, blocksize, alignment, bands, band type and a flag to denote if raster column is regularly blocked. The table must be loaded with data for the constraints to be inferred. Returns true if the constraint setting was accomplished and issues a notice otherwise.

#### Synopsis

```
boolean AddRasterConstraints(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 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[] VARIADIC constraints);
boolean AddRasterConstraints(name rastschema, name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=true,
boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true, boolean same_alignment=true, boolean regular_blocking=true,
boolean num_bands=true, boolean pixel_types=true, boolean no_data_values=true, boolean out_db=true, boolean extent=true);
```

#### Descrição

Gera restrições em uma coluna raster que são usadas para expor informação no catálogo raster `raster_columns`. O `rastschema` é o nome da tabela esquema que a tabela está. O `srid` deve ser um valor inteiro referência a uma entrada na tabela `SPATIAL_REF_SYS`.

`raster2pgsql` o carregador usa esta função para registrar tabelas raster

Valida nomes restritos para passar: recorra a [Section 5.2.1](#) para mais detalhes.

- `blocksize` coloca X e Y blocksize
- `blocksize_x` coloca tile X (largura em pixels de cada tile)
- `blocksize_y` coloca tile Y (altura em pixels de cada tile)
- `extent` calcula a extensão da tabela toda e aplica restrições, todos os rasters devem estar dentro da extensão
- `num_bands` número de bandas
- `pixel_types` lê arranjo de tipos de pixels para cada banda garantir que todas as bandas n tenham o mesmo tipo de pixel
- `regular_blocking` espacialmente único (dois rasters não podem ser espacialmente iguais) e restrições de tile de cobertura (raster é alinhado a uma cobertura)
- `same_alignment` ensures they all have same alignment meaning any two tiles you compare will return true for. Refer to [ST\\_SameAlignment](#).
- `srid` assegura que todos tenham o mesmo srid
- Mais -- qualquer um listado como entrada dentro das funções acima





### 9.2.3 AddOverviewConstraints

AddOverviewConstraints — Marca uma coluna raster como sendo um resumo de outra.

#### 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);

#### Descrição

Adiciona restrições em uma coluna raster que são usadas para expor informações no catálogo `raster_overviews` raster.

O parâmetro `ovfactor` representa a escala multiplicadora na coluna resumo: fatores resumo mais altos possuem uma resolução menor.

Quando os parâmetros `ovschema` e `refschema` são omitidos, a primeira tabela encontrada escaneando o `search_path` será utilizada.

Disponibilidade: 2.0.0

#### Exemplos

```
CREATE TABLE res1 AS SELECT
ST_AddBand(
  ST_MakeEmptyRaster(1000, 1000, 0, 0, 2),
  1, '8BSI'::text, -129, NULL
) r1;

CREATE TABLE res2 AS SELECT
ST_AddBand(
  ST_MakeEmptyRaster(500, 500, 0, 0, 4),
  1, '8BSI'::text, -129, NULL
) r2;

SELECT AddOverviewConstraints('res2', 'r2', 'res1', 'r1', 2);

-- verify if registered correctly in the raster_overviews view --
SELECT o_table_name ot, o_raster_column oc,
       r_table_name rt, r_raster_column rc,
       overview_factor f
FROM raster_overviews WHERE o_table_name = 'res2';
  ot | oc | rt | rc | f
-----+-----+-----+-----+----
res2 | r2 | res1 | r1 | 2
(1 row)
```

#### Veja também

Section [5.2.2](#), [DropOverviewConstraints](#), [ST\\_CreateOverview](#), [AddRasterConstraints](#)

### 9.2.4 DropOverviewConstraints

DropOverviewConstraints — Desmarca uma coluna raster de ser um resumo de outra.

## Synopsis

boolean **DropOverviewConstraints**(name ovschema, name ovtable, name ovcolumn);  
boolean **DropOverviewConstraints**(name ovtable, name ovcolumn);

## Descrição

Remove as restrições de uma coluna raster usadas para apresentá-la como um resumo de outra no catálogo `raster_overviews` raster.

Quando o parâmetro `ovschema` é omitido, a primeira tabela encontradas escaneando o `search_path` será utilizada.

Disponibilidade: 2.0.0

## Veja também

Section [5.2.2](#), [AddOverviewConstraints](#), [DropRasterConstraints](#)

## 9.2.5 PostGIS\_GDAL\_Version

`PostGIS_GDAL_Version` — Relata a versão da biblioteca GDAL em uso pelo PostGIS

## Synopsis

text **PostGIS\_GDAL\_Version**();

## Descrição

Relata a versão da biblioteca em uso pelo PostGIS. Também irá verificar e reportar se GDAL pode encontrar os dados de seus arquivos.

## Exemplos

```
SELECT PostGIS_GDAL_Version();
       postgis_gdal_version
-----
GDAL 1.11dev, released 2013/04/13
```

## Veja também

[postgis.gdal\\_datapath](#)

## 9.2.6 PostGIS\_Raster\_Lib\_Build\_Date

`PostGIS_Raster_Lib_Build_Date` — Relata a data da biblioteca raster construída completa.

## Synopsis

text **PostGIS\_Raster\_Lib\_Build\_Date**();

### Descrição

Relata a data de construção raster

### Exemplos

```
SELECT PostGIS_Raster_Lib_Build_Date();
postgis_raster_lib_build_date
-----
2010-04-28 21:15:10
```

### Veja também

[PostGIS\\_Raster\\_Lib\\_Version](#)

## 9.2.7 PostGIS\_Raster\_Lib\_Version

PostGIS\_Raster\_Lib\_Version — Relata a versão raster completa e constrói informações de configuração.

### Synopsis

```
text PostGIS_Raster_Lib_Version();
```

### Descrição

Relata a versão raster completa e constrói informações de configuração.

### Exemplos

```
SELECT PostGIS_Raster_Lib_Version();
postgis_raster_lib_version
-----
2.0.0
```

### Veja também

[PostGIS\\_Lib\\_Version](#)

## 9.2.8 ST\_GDALDrivers

ST\_GDALDrivers — Returns a list of raster formats supported by PostGIS through GDAL. Only those formats with can\_write=True can be used by ST\_AsGDALRaster

### Synopsis

```
setof record ST_GDALDrivers(integer OUT idx, text OUT short_name, text OUT long_name, text OUT can_read, text OUT can_write, text OUT create_options);
```



## Descrição

Returns a list of raster formats `short_name`, `long_name` and creator options of each format supported by GDAL. Use the `short_name` as input in the `format` parameter of `ST_AsGDALRaster`. Options vary depending on what drivers your `libgdal` was compiled with. `create_options` returns an xml formatted set of `CreationOptionList/Option` consisting of name and optional type, description and set of `VALUE` for each creator option for the specific driver.

Changed: 2.5.0 - add `can_read` and `can_write` columns.

Alterações: 2.0.6, 2.1.3 - por padrão nenhum driver é ativado, a menos que GUC ou a variável ambiental `gdal_enabled_drivers` estejam colocadas.

Disponibilidade: 2.0.0 - requer GDAL  $\geq$  1.6.0.

## Exemplos: Lista de Dispositivos

```
SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SELECT short_name, long_name, can_write
FROM st_gdaldrivers()
ORDER BY short_name;
```

short_name	long_name	can_write
AAIGrid	Arc/Info ASCII Grid	t
ACE2	ACE2	f
ADRG	ARC Digitized Raster Graphics	f
AIG	Arc/Info Binary Grid	f
AirSAR	AirSAR Polarimetric Image	f
ARG	Azavea Raster Grid format	t
BAG	Bathymetry Attributed Grid	f
BIGGIF	Graphics Interchange Format (.gif)	f
BLX	Magellan topo (.blx)	t
BMP	MS Windows Device Independent Bitmap	f
BSB	Maptech BSB Nautical Charts	f
PAux	PCI .aux Labelled	f
PCIDSK	PCIDSK Database File	f
PCRaster	PCRaster Raster File	f
PDF	Geospatial PDF	f
PDS	NASA Planetary Data System	f
PDS4	NASA Planetary Data System 4	t
PLMOAIC	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 (.sdatt, .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

### Exemplo: Lista de opções para cada dispositivo

```
-- Output the create options XML column of JPEG as a table --
-- Note you can use these creator options in ST_AsGDALRaster options argument
SELECT (xpath('@name', g.opt))[1]::text As oname,
       (xpath('@type', g.opt))[1]::text As otype,
       (xpath('@description', g.opt))[1]::text As descrip
FROM (SELECT unnest(xpath('/CreationOptionList/Option', create_options::xml)) As opt
FROM st_gdaldrivers()
WHERE short_name = 'JPEG') As g;
```

oname	otype	descrip
PROGRESSIVE	boolean	whether to generate a progressive JPEG
QUALITY	int	good=100, bad=0, default=75
WORLDFILE	boolean	whether to generate a worldfile
INTERNAL_MASK	boolean	whether to generate a validity mask
COMMENT	string	Comment
SOURCE_ICC_PROFILE	string	ICC profile encoded in Base64
EXIF_THUMBNAIL	boolean	whether to generate an EXIF thumbnail(overview). By default its max dimension will be 128
THUMBNAIL_WIDTH	int	Forced thumbnail width
THUMBNAIL_HEIGHT	int	Forced thumbnail height

(9 rows)

```
-- raw xml output for creator options for GeoTiff --
SELECT create_options
FROM st_gdaldrivers()
WHERE short_name = 'GTiff';
```

```
<CreationOptionList>
  <Option name="COMPRESS" type="string-select">
    <Value>
>NONE</Value>
    <Value>
>LZW</Value>
    <Value>
>PACKBITS</Value>
    <Value>
>JPEG</Value>
    <Value>
>CCITTRLE</Value>
    <Value>
>CCITTFAX3</Value>
```

```

    <Value
>CCITTFAX4</Value>
    <Value
>DEFLATE</Value>
  </Option>
  <Option name="PREDICTOR" type="int" description="Predictor Type"/>
  <Option name="JPEG_QUALITY" type="int" description="JPEG quality 1-100" default="75"/>
  <Option name="ZLEVEL" type="int" description="DEFLATE compression level 1-9" default ←
    ="6"/>
  <Option name="NBITS" type="int" description="BITS for sub-byte files (1-7), sub-uint16 ←
    (9-15), sub-uint32 (17-31)"/>
  <Option name="INTERLEAVE" type="string-select" default="PIXEL">
    <Value
>BAND</Value>
    <Value
>PIXEL</Value>
  </Option>
  <Option name="TILED" type="boolean" description="Switch to tiled format"/>
  <Option name="TFW" type="boolean" description="Write out world file"/>
  <Option name="RPB" type="boolean" description="Write out .RPB (RPC) file"/>
  <Option name="BLOCKXSIZE" type="int" description="Tile Width"/>
  <Option name="BLOCKYSIZE" type="int" description="Tile/Strip Height"/>
  <Option name="PHOTOMETRIC" type="string-select">
    <Value
>MINISBLACK</Value>
    <Value
>MINISWHITE</Value>
    <Value
>PALETTE</Value>
    <Value
>RGB</Value>
    <Value
>CMYK</Value>
    <Value
>YCBCR</Value>
    <Value
>CIELAB</Value>
    <Value
>ICCLAB</Value>
    <Value
>ITULAB</Value>
  </Option>
  <Option name="SPARSE_OK" type="boolean" description="Can newly created files have ←
    missing blocks?" default="FALSE"/>
  <Option name="ALPHA" type="boolean" description="Mark first extrasample as being alpha ←
    "/>
  <Option name="PROFILE" type="string-select" default="GDALGeoTIFF">
    <Value
>GDALGeoTIFF</Value>
    <Value
>GeoTIFF</Value>
    <Value
>BASELINE</Value>
  </Option>
  <Option name="PIXELTYPE" type="string-select">
    <Value
>DEFAULT</Value>
    <Value
>SIGNEDBYTE</Value>
  </Option>
  <Option name="BIGTIFF" type="string-select" description="Force creation of BigTIFF file ←
    ">

```

```

        <Value
>YES</Value>
        <Value
>NO</Value>
        <Value
>IF_NEEDED</Value>
        <Value
>IF_SAFER</Value>
        </Option>
        <Option name="ENDIANNESS" type="string-select" default="NATIVE" description="Force ↵
        endianness of created file. For DEBUG purpose mostly">
        <Value
>NATIVE</Value>
        <Value
>INVERTED</Value>
        <Value
>LITTLE</Value>
        <Value
>BIG</Value>
        </Option>
        <Option name="COPY_SRC_OVERVIEWS" type="boolean" default="NO" description="Force copy ↵
        of overviews of source dataset (CreateCopy())"/>
</CreationOptionList
>

```

```

-- Output the create options XML column for GTiff as a table --
SELECT (xpath('@name', g.opt))[1]::text As oname,
       (xpath('@type', g.opt))[1]::text As otype,
       (xpath('@description', g.opt))[1]::text As descrip,
       array_to_string(xpath('Value/text()', g.opt),', ') As vals
FROM (SELECT unnest(xpath('/CreationOptionList/Option', create_options::xml)) As opt
FROM st_gdaldrivers()
WHERE short_name = 'GTiff') As g;

```

oname	otype	descrip	vals
COMPRESS	string-select		NONE, LZW, ↵
PACKBITS, JPEG, CCITTRLE, CCITTFAX3, CCITTFAX4, DEFLATE			
PREDICTOR	int	Predictor Type ↵	
JPEG_QUALITY	int	JPEG quality 1-100 ↵	
ZLEVEL	int	DEFLATE compression level 1-9 ↵	
NBITS	int	BITS for sub-byte files (1-7), sub-uint16 (9-15), sub ↵	
-uint32 (17-31)			
INTERLEAVE	string-select		BAND, PIXEL
TILED	boolean	Switch to tiled format ↵	
TFW	boolean	Write out world file ↵	
RPB	boolean	Write out .RPB (RPC) file ↵	
BLOCKXSIZE	int	Tile Width ↵	
BLOCKYSIZE	int	Tile/Strip Height ↵	
PHOTOMETRIC	string-select		

			MINISBLACK, ↔
MINISWHITE, PALETTE, RGB, CMYK, YCBCR, CIELAB, ICCLAB, ITULAB			
SPARSE_OK	boolean	Can newly created files have missing blocks?	↔
ALPHA	boolean	Mark first extrasample as being alpha	↔
PROFILE	string-select		↔
			GDALGeoTIFF, ↔
GeoTIFF, BASELINE			
PIXELTYPE	string-select		↔
			DEFAULT, ↔
SIGNEDBYTE			
BIGTIFF	string-select	Force creation of BigTIFF file	↔
		YES, NO, IF_NEEDED, IF_SAFER	
ENDIANNESS	string-select	Force endianness of created file. For DEBUG purpose	↔
mostly	NATIVE, INVERTED, LITTLE, BIG		
COPY_SRC_OVERVIEWS	boolean	Force copy of overviews of source dataset (CreateCopy	↔
( )			
(19 rows)			

### Veja também

[ST\\_AsGDALRaster](#), [ST\\_SRID](#), [postgis.gdal\\_enabled\\_drivers](#)

## 9.2.9 UpdateRasterSRID

UpdateRasterSRID — Altera o SRID de todos os rasters na coluna e tabela do usuário especificado.

### 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);
```

### Descrição

Altera o SRID de todos os rasters na coluna e tabela do usuário especificado. A função irá derrubar todas as restrições de colunas apropriadas (extensão, alinhamento e SRID) antes de modificar o SRID dos rasters específicos da coluna.



#### Note

Os dados (banda valores pixel) dos rasters não são mexidos por esta função. Somente os metadados do raster são alterados.

Disponibilidade: 2.1.0

### Veja também

[UpdateGeometrySRID](#)

## 9.2.10 ST\_CreateOverview

ST\_CreateOverview — Cria uma resolução de versão reduzida de uma dada cobertura raster.

## Synopsis

```
regclass ST_CreateOverview(regclass tab, name col, int factor, text algo='NearestNeighbor');
```

## Descrição

Cria uma tabela panorama com resampled tiles da tabela fonte. AS tiles de saída terão o mesmo tamanho das de entrada e cobrirão a mesma extensão espacial com uma resolução mais baixa (tamanho do pixel será  $1/\text{factor}$  do original em ambas direções).

A tabela panorama se tornará disponível no catálogo `raster_overviews` e terá restrições raster executadas.

As opções de algoritmo são: 'NearestNeighbor', 'Bilinear', 'Cubic', 'CubicSpline', e 'Lanczos'. Recorra a: [GDAL Warp resampling methods](#) para mais detalhes.

Disponibilidade: 2.2.0

## Exemplo

Output to generally better quality but slower to product format

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2, 'Lanczos');
```

Output to faster to process default nearest neighbor

```
SELECT ST_CreateOverview('mydata.mytable'::regclass, 'rast', 2);
```

## Veja também

[ST\\_Retile](#), [AddOverviewConstraints](#), [AddRasterConstraints](#), [Section 5.2.2](#)

## 9.3 Construtores Raster

### 9.3.1 ST\_AddBand

**ST\_AddBand** — Retorna um raster com nova banda(s) do tipo dado adicionado com o valor inicial com a localização do índice. Se nenhum índice for especificado, a banda é adicionada ao final.

## 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);

## Descrição

Retorna um raster com uma nova banda adicionada na posição (índice), do dado tipo, do valor inicial, e do dado valor nodata. Se nenhum índice for especificado, a banda é adicionada ao final. Se nenhum `fromband` for especificado, banda 1 é assumida. O tipo pixel é uma representação de string de um dos tipos de pixel especificados em `ST_BandPixelType`. Se um índice existente for especificado todas as bandas subsequentes  $\geq$  aquele índice é incrementado por 1. Se um valor inicial maior que o máximo do tipo pixel for especificado, então ele é estabelecido como o maior valor permitido pelo tipo pixel.

Para a variante que pega um arranjo de `addbandarg` (Variante 1), um valor de índice `addbandarg`'s específico é relativo ao raster no mesmo tempo que a banda é descrita por aquele `addbandarg`'s está sendo adicionada ao raster. Veja o exemplo abaixo.

Para a variante que pega um arranjo de rasters (Variante 5), se `torast` é NULO então a `fromband` banda de cada raster no arranjo está acumulada dentro de um novo raster.

Para as variantes que pegam `outdbfile` (Variantes 6 e 7), o valor deve incluir o caminho completo para o arquivo raster. O arquivo deve ser acessível também para o processo do servidor postgres.

Melhorias: 2.1.0 suporte para `addbandarg` adicionado.

Melhorias: 2.1.0 suporte para novas bandas out-db adicionado.

## Exemplos: Nova banda única

```
-- Add another band of type 8 bit unsigned integer with pixels initialized to 200
UPDATE dummy_rast
  SET rast = ST_AddBand(rast,'8BUI'::text,200)
WHERE rid = 1;
```

```
-- Create an empty raster 100x100 units, with upper left right at 0, add 2 bands (band 1 ←
  is 0/1 boolean bit switch, band2 allows values 0-15)
-- uses addbandargs
INSERT INTO dummy_rast(rid,rast)
  VALUES(10, ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 1, -1, 0, 0, 0),
    ARRAY[
      ROW(1, '1BB'::text, 0, NULL),
      ROW(2, '4BUI'::text, 0, NULL)
    ]::addbandarg[]
  )
);
```

```
-- output meta data of raster bands to verify all is right --
SELECT (bmd).*
FROM (SELECT ST_BandMetaData(rast,generate_series(1,2)) As bmd
  FROM dummy_rast WHERE rid = 10) AS foo;
--result --
pixeltype | nodatavalue | isoutdb | path
-----+-----+-----+-----
1BB      |              | f       |
4BUI     |              | f       |
```

```
-- output meta data of raster -
SELECT (rmd).width, (rmd).height, (rmd).numbands
FROM (SELECT ST_MetaData(rast) As rmd
  FROM dummy_rast WHERE rid = 10) AS foo;
-- result --
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | ←
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
0 | 0 | 100 | 100 | 1 | -1 | 0 | 0 | 0 | ←
2
```

**Exemplos: Várias bandas novas**

```

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)
    ]::adbandarg[]
  ),
  ARRAY[]::integer[]
);

```

bandnum	pixeltype	nodatavalue	isoutdb	path
1	8BUI	0	f	
2	32BF	-1	f	
3	32BUI	12	f	
4	16BUI	2	f	

```

-- Aggregate the 1st band of a table of like rasters into a single raster
-- with as many bands as there are test_types and as many rows (new rasters) as there are ←
  mice
-- NOTE: The ORDER BY test_type is only supported in PostgreSQL 9.0+
-- for 8.4 and below it usually works to order your data in a subselect (but not guaranteed ←
  )
-- The resulting raster will have a band for each test_type alphabetical by test_type
-- For mouse lovers: No mice were harmed in this exercise
SELECT
  mouse,
  ST_AddBand(NULL, array_agg(rast ORDER BY test_type), 1) As rast
FROM mice_studies
GROUP BY mouse;

```

**Exemplos: Nova banda out-db**

```

SELECT
  *
FROM ST_BandMetadata(
  ST_AddBand(
    ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1, 0, 0, 0),
    '/home/raster/mytestraster.tif'::text, NULL::int[]
  ),
  ARRAY[]::integer[]
);

```

bandnum	pixeltype	nodatavalue	isoutdb	path
1	8BUI		t	/home/raster/mytestraster.tif
2	8BUI		t	/home/raster/mytestraster.tif
3	8BUI		t	/home/raster/mytestraster.tif

**Veja também**

[ST\\_BandMetaData](#), [ST\\_BandPixelType](#), [ST\\_MakeEmptyRaster](#), [ST\\_MetaData](#), [ST\\_NumBands](#), [ST\\_Reclass](#)



### 9.3.2 ST\_AsRaster

ST\_AsRaster — Converte uma geometria PostGIS para um raster PostGIS.

#### Synopsis

```
raster ST_AsRaster(geometry geom, raster ref, text pixeltype, double precision value=1, double precision nodataval=0, boolean touched=false);
raster ST_AsRaster(geometry geom, raster ref, text[] pixeltype=ARRAY['8BIT'], 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['8BIT'], 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['8BIT'], 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);
```

#### Descrição

Converte uma geometria PostGIS para um raster PostGIS. As diversas variantes oferecem três grupos de possibilidades para configurar o alinhamento e o tamanho do pixel do raster resultante.

O primeiro grupo, composto pelas duas primeiras variantes, produz um raster tendo o mesmo alinhamento (*scalex*, *scaley*, *gridx* e *gridy*), tipo pixel e valor nodata como foi fornecido pelo raster referência. Você geralmente passa este raster referência unindo a tabela que contém a geometria com a que contém o raster referência.

O segundo grupo, composto por quatro variantes, permite que você fixe dimensões do raster fornecendo os parâmetros do tamanho de um pixel (*scalex* & *scaley* e *skewx* & *skewy*). O *width* & *height* do raster resultante será ajustado para caber na extensão da geometria. Na maioria dos casos, você deve cast *integer scalex* & *scaley* argumentos para dobrar a precisão para que o PostgreSQL escolha a variante correta.

O terceiro grupo, composto por quatro variantes, permite que você conserte dimensões do raster fornecendo elas (*width* & *height*). Os parâmetros do tamanho do pixel (*scalex* & *scaley* and *skewx* & *skewy*) do raster resultante será ajustado para caber na extensão da geometria.

As duas primeiras variantes de cada um destes dois últimos grupos permite que você especifique o alinhamento com um canto aleatório da rede de alinhamento (*gridx* & *gridy*) e as duas últimas variantes pegam o canto esquerdo mais alto (*upperleftx* & *upperlefty*).

Cada grupo de variantes permite a produção de uma ou várias bandas raster. Para produzir várias, você deve fornecer um arranjo de tipos pixel (`pixeltype[]`), um arranjo de valores iniciais (`value`) e um de valores nodata (`nodataval`). Se não fornecidos `pixeltype` torna-se 8BUI, valores para 1 e `nodataval` para 0.

O raster de saída terá a mesma referência espacial que a geometria fonte. A única exceção é para variantes com raster referência. Neste caso, o raster resultante terá o mesmo SRID do raster referência.

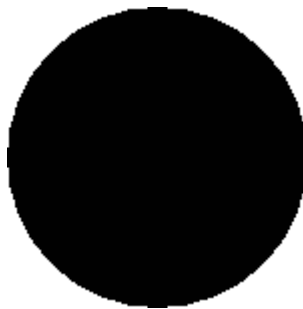
O parâmetro opcional `touched` é falso e mapeia a opção rasterização GDAL `ALL_TOUCHED`, a qual determina se pixels tocados por linhas ou polígonos serão queimados. Não apenas aqueles no caminho de renderização de linha, ou aqueles cujo ponto central está dentro do polígono.

Isto é particularmente útil para renderizar jpegs e pngs de geometrias diretamente do banco de dados quando usando em conjunto com `ST_AsPNG` e outra família `ST_AsGDALRaster` de funções.

Disponibilidade: 2.0.0 - requer GDAL  $\geq$  1.6.0.

**Note**

Ainda não é capaz de renderizar geometrias complexas como: curvas, TINS, e superfícies poliédricas, mas deveria ser, já que GDAL consegue.

**Exemplos: Gera geometrias como arquivos PNG**

*círculo preto*

```
-- this will output a black circle taking up 150 x 150 pixels --  
SELECT ST_AsPNG(ST_AsRaster(ST_Buffer(ST_Point(1,5),10),150, 150));
```



*exemplo de buffer renderizado só com o PostGIS*

```
-- the bands map to RGB bands - the value (118,154,118) - teal --
SELECT ST_AsPNG(
  ST_AsRaster(
    ST_Buffer(
      ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'), 10,'join=bevel ←
    '),
    200,200,ARRAY['8BUI', '8BUI', '8BUI'], ARRAY[118,154,118], ARRAY ←
    [0,0,0]));
```

## Veja também

[ST\\_BandPixelType](#), [ST\\_Buffer](#), [ST\\_GDALDrivers](#), [ST\\_AsGDALRaster](#), [ST\\_AsPNG](#), [ST\\_AsJPEG](#), [ST\\_SRID](#)

### 9.3.3 ST\_Band

**ST\_Band** — Retorna uma ou mais bandas de um raster existente como um novo raster. Útil para a construção de novos rasters a partir de rasters existentes.

#### 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=,);
```

#### Descrição

Returns one or more bands of an existing raster as a new raster. Useful for building new rasters from existing rasters or export of only selected bands of a raster or rearranging the order of bands in a raster. If no band is specified or any of specified bands does not exist in the raster, then all bands are returned. Used as a helper function in various functions such as for deleting a band.



#### Warning

Para as `nbands` como variantes de textos de função, o delimitador padrão é `,` que significa que você pode pedir por `'1,2,3'` e se quiser usar um delimitador diferente você poderia fazer `ST_Band(rast, '1@2@3', '@')`. Para pedir por várias bandas, sugerimos que use a forma de arranjo desta função ex.: `ST_Band(rast, '{1,2,3}'::int[])`; já que a forma de lista de banda `text` pode ser removida em versões futuras do PostGIS.

Disponibilidade: 2.0.0

#### Exemplos

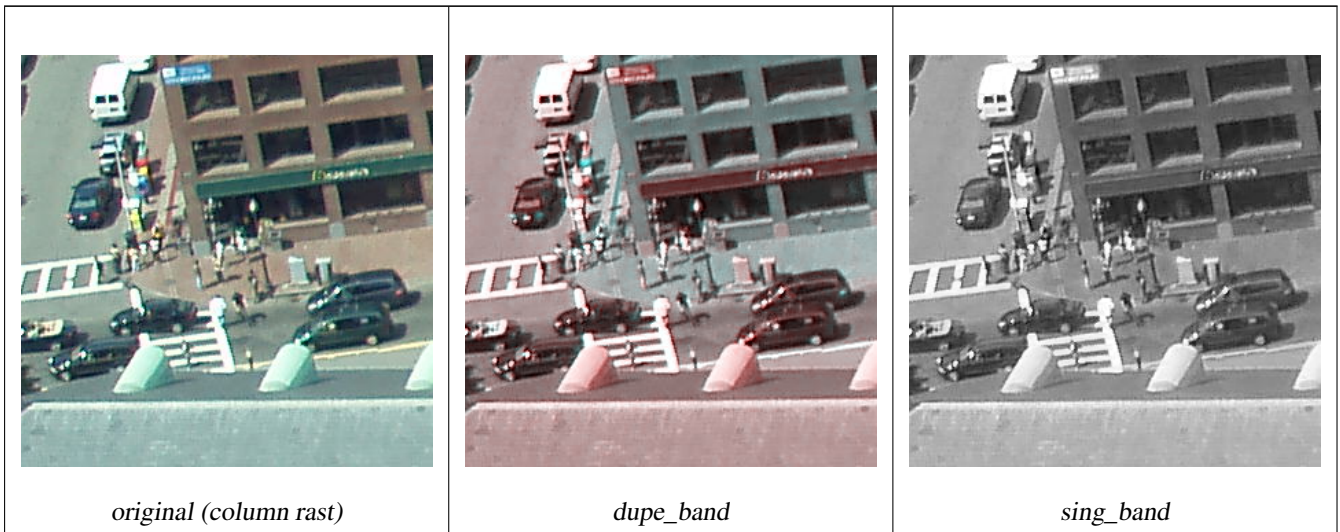
```
-- Make 2 new rasters: 1 containing band 1 of dummy, second containing band 2 of dummy and ←
  then reclassified as a 2BUI
SELECT ST_NumBands(rast1) As numb1, ST_BandPixelType(rast1) As pix1,
  ST_NumBands(rast2) As numb2, ST_BandPixelType(rast2) As pix2
FROM (
  SELECT ST_Band(rast) As rast1, ST_Reclass(ST_Band(rast,3), '100-200):1, [200-254:2', '2 ←
    BUI') As rast2
  FROM dummy_rast
  WHERE rid = 2) As foo;
```

```
numb1 | pix1 | numb2 | pix2
-----+-----+-----+-----
      1 | 8BUI |       1 | 2BUI
```

```
-- Return bands 2 and 3. Using array cast syntax
SELECT ST_NumBands(ST_Band(rast, '{2,3}'::int[])) As num_bands
FROM dummy_rast WHERE rid=2;
```

```
num_bands
-----
2
```

```
-- Return bands 2 and 3. Use array to define bands
SELECT ST_NumBands(ST_Band(rast, ARRAY[2,3])) As num_bands
FROM dummy_rast
WHERE rid=2;
```



```
--Make a new raster with 2nd band of original and 1st band repeated twice,
and another with just the third band
SELECT rast, ST_Band(rast, ARRAY[2,1,1]) As dupe_band,
ST_Band(rast, 3) As sing_band
FROM samples.than_chunked
WHERE rid=35;
```

## Veja também

[ST\\_AddBand](#), [ST\\_NumBands](#), [ST\\_Reclass](#), [Chapter 9](#)

### 9.3.4 ST\_MakeEmptyCoverage

`ST_MakeEmptyCoverage` — Cover georeferenced area with a grid of empty raster tiles.

#### 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);

### Descrição

Create a set of raster tiles with **ST\_MakeEmptyRaster**. Grid dimension is `width` & `height`. Tile dimension is `tilewidth` & `tileheight`. The covered georeferenced area is from upper left corner (`upperleftx`, `upperlefty`) to lower right corner (`upperleftx + width * scalex`, `upperlefty + height * scaley`).



**Note**

Note that `scaley` is generally negative for rasters and `scalex` is generally positive. So lower right corner will have a lower `y` value and higher `x` value than the upper left corner.

Availability: 2.4.0

### Exemplos básicos

Create 16 tiles in a 4x4 grid to cover the WGS84 area from upper left corner (22, 77) to lower right corner (55, 33).

```
SELECT (ST_MetaData(tile)).* FROM ST_MakeEmptyCoverage(1, 1, 4, 4, 22, 33, (55 - 22)/(4)::float, (33 - 77)/(4)::float, 0., 0., 4326) tile;
```

upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	numbands
22	77	1	1	8.25	-11	0	0	4326	1
30.25	77	1	1	8.25	-11	0	0	4326	1
38.5	77	1	1	8.25	-11	0	0	4326	1
46.75	77	1	1	8.25	-11	0	0	4326	1
22	73	1	1	8.25	-11	0	0	4326	1
30.25	73	1	1	8.25	-11	0	0	4326	1
38.5	73	1	1	8.25	-11	0	0	4326	1
46.75	73	1	1	8.25	-11	0	0	4326	1
22	69	1	1	8.25	-11	0	0	4326	1
30.25	69	1	1	8.25	-11	0	0	4326	1
38.5	69	1	1	8.25	-11	0	0	4326	1
46.75	69	1	1	8.25	-11	0	0	4326	1
22	65	1	1	8.25	-11	0	0	4326	1
30.25	65	1	1	8.25	-11	0	0	4326	1
38.5	65	1	1	8.25	-11	0	0	4326	1
46.75	65	1	1	8.25	-11	0	0	4326	1

### Veja também

[ST\\_MakeEmptyRaster](#)

### 9.3.5 ST\_MakeEmptyRaster

`ST_MakeEmptyRaster` — Retorna um raster vazio (sem bandas) das dimensões dadas (`width & height`), o `X` e `Y` do superior esquerdo, tamanho de pixel e rotação (`scalex`, `scaley`, `skewx` & `skewy`) e sistema de referência (`srid`). Se um raster passar, retorna um novo raster com o mesmo tamanho, alinhamento e `SRID`. Se o `srid` é deixado de fora, a referência espacial se torna desconhecida (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);
```

#### Descrição

Retorna um raster vazio (sem bandas) das dimensões dadas (`width & height`) e georeferenciado nas coordenadas espaciais (ou mundo) com o `X` esquerdo superior (`upperleftx`), `Y` superior esquerdo (`upperlefty`), tamanho de pixel e rotação (`scalex`, `scaley`, `skewx` & `skewy`) e sistema de referência (`srid`).

A última versão usa um único parâmetro para especificar o tamanho do pixel (`pixelsize`). `scalex` é estabelecida neste argumento e `scaley` é estabelecida no valor negativo deste argumento. `skewx` e `skewy` são 0.

Se um raster existente passar, ele retorna um novo raster com as mesmas configurações de meta dados (sem as bandas).

Se nenhum `srid` é especificado, é 0. Depois que você criou um raster vazio talvez queira adicionar bandas a ele e editá-lo. Recorra a [ST\\_AddBand](#) para definir bandas e [ST\\_SetValue](#) para valores iniciais de pixel.

#### Exemplos

```
INSERT INTO dummy_rast(rid,rast)
VALUES(3, ST_MakeEmptyRaster( 100, 100, 0.0005, 0.0005, 1, 1, 0, 0, 4326) );

--use an existing raster as template for new raster
INSERT INTO dummy_rast(rid,rast)
SELECT 4, ST_MakeEmptyRaster(rast)
FROM dummy_rast WHERE rid = 3;

-- output meta data of rasters we just added
SELECT rid, (md).*
FROM (SELECT rid, ST_MetaData(rast) As md
      FROM dummy_rast
      WHERE rid IN(3,4)) As foo;

-- output --
rid | upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands |
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
  3 |    0.0005 |    0.0005 |   100 |   100 |      1 |      1 |      0 |      0 |    0 |         0 |
    4326 |          0 |
  4 |    0.0005 |    0.0005 |   100 |   100 |      1 |      1 |      0 |      0 |    0 |         0 |
    4326 |          0 |
```

#### Veja também

[ST\\_AddBand](#), [ST\\_MetaData](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#), [ST\\_SetValue](#), [ST\\_SkewX](#), [ST\\_SkewY](#)

### 9.3.6 ST\_Tile

**ST\_Tile** — Retorna um conjunto de rasters resultante de uma divisão do raster de entrada baseado nas dimensões desejadas nos rasters de saída.

#### 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);
```

#### Descrição

Retorna um conjunto de rasters resultante de uma divisão do raster de entrada baseado nas dimensões desejadas nos rasters de saída.

Se `padwithnodata = FALSO`, tiles limite no lado direito e inferior do raster podem ter dimensões diferentes do resto das tiles. Se `padwithnodata = VERDADEIRO`, todas as tiles terão a mesma dimensão com a possibilidade das tiles limites serem preenchidas com valores NODATA. Se a banda(s) não possui valor(es) NODATA especificado, pode ser especificado por `nodataval`.



#### Note

Se uma banda especificada do raster de entrada estiver out-of-db, a banda correspondente nos rasters de saída também estará.

Disponibilidade: 2.1.0

#### Exemplos

```
WITH foo AS (
  SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8 ←
    BUI', 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, '8 ←
    BUI', 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, '8 ←
    BUI', 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, '8 ←
    BUI', 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, '8 ←
    BUI', 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, '8 ←
    BUI', 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, '8 ←
    BUI', 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, '8 ←
    BUI', 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, '8 ←
    BUI', 9, 0), 2, '8BUI', 90, 0) AS rast
), bar AS (
  SELECT ST_Union(rast) AS rast FROM foo
), baz AS (
```

```

        SELECT ST_Tile(rast, 3, 3, TRUE) AS rast FROM bar
    )
    SELECT
        ST_DumpValues(rast)
    FROM baz;

```

```

        st_dumpvalues
-----

```

```

(1, "{{1,1,1},{1,1,1},{1,1,1}}")
(2, "{{10,10,10},{10,10,10},{10,10,10}}")
(1, "{{2,2,2},{2,2,2},{2,2,2}}")
(2, "{{20,20,20},{20,20,20},{20,20,20}}")
(1, "{{3,3,3},{3,3,3},{3,3,3}}")
(2, "{{30,30,30},{30,30,30},{30,30,30}}")
(1, "{{4,4,4},{4,4,4},{4,4,4}}")
(2, "{{40,40,40},{40,40,40},{40,40,40}}")
(1, "{{5,5,5},{5,5,5},{5,5,5}}")
(2, "{{50,50,50},{50,50,50},{50,50,50}}")
(1, "{{6,6,6},{6,6,6},{6,6,6}}")
(2, "{{60,60,60},{60,60,60},{60,60,60}}")
(1, "{{7,7,7},{7,7,7},{7,7,7}}")
(2, "{{70,70,70},{70,70,70},{70,70,70}}")
(1, "{{8,8,8},{8,8,8},{8,8,8}}")
(2, "{{80,80,80},{80,80,80},{80,80,80}}")
(1, "{{9,9,9},{9,9,9},{9,9,9}}")
(2, "{{90,90,90},{90,90,90},{90,90,90}}")
(18 rows)

```

```

WITH foo AS (
    SELECT ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8 ←
        BUI', 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, '8 ←
        BUI', 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, '8 ←
        BUI', 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, '8 ←
        BUI', 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, '8 ←
        BUI', 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, '8 ←
        BUI', 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, '8 ←
        BUI', 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, '8 ←
        BUI', 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, '8 ←
        BUI', 9, 0), 2, '8BUI', 90, 0) AS rast
), bar AS (
    SELECT ST_Union(rast) AS rast FROM foo
), baz AS (
    SELECT ST_Tile(rast, 3, 3, 2) AS rast FROM bar
)
SELECT
    ST_DumpValues(rast)
FROM baz;

```

```

        st_dumpvalues
-----

```

```

(1, "{{10,10,10},{10,10,10},{10,10,10}}")
(1, "{{20,20,20},{20,20,20},{20,20,20}}")

```



```
(1, "{30,30,30},{30,30,30},{30,30,30}")
(1, "{40,40,40},{40,40,40},{40,40,40}")
(1, "{50,50,50},{50,50,50},{50,50,50}")
(1, "{60,60,60},{60,60,60},{60,60,60}")
(1, "{70,70,70},{70,70,70},{70,70,70}")
(1, "{80,80,80},{80,80,80},{80,80,80}")
(1, "{90,90,90},{90,90,90},{90,90,90}")
(9 rows)
```

### Veja também

[ST\\_Union](#), [ST\\_Retile](#)

### 9.3.7 ST\_Retile

`ST_Retile` — Retorna um conjunto de tiles configuradas de uma cobertura raster aleatória.

#### Synopsis

```
SETOF raster ST_Retile(regclass tab, name col, geometry ext, float8 sfx, float8 sfy, int tw, int th, text algo='NearestNeighbor');
```

#### Descrição

Retorna um conjunto de tiles tendo a escala especificada (`sfx`, `sfy`) e tamanho máximo (`tw`, `th`) e cobrindo a extensão especificada (`ext`) com os dados vindos da cobertura raster especificada (`tab`, `col`).

As opções de algoritmo são: 'NearestNeighbor', 'Bilinear', 'Cubic', 'CubicSpline', e 'Lanczos'. Recorra a: [GDAL Warp resampling methods](#) para mais detalhes.

Disponibilidade: 2.2.0

### Veja também

[ST\\_CreateOverview](#)

### 9.3.8 ST\_FromGDALRaster

`ST_FromGDALRaster` — Retorna um raster de um arquivo raster GDAL suportado.

#### Synopsis

```
raster ST_FromGDALRaster(bytea gdaldata, integer srid=NULL);
```

#### Descrição

Retorna um raster de uma arquivo raster GDAL suportado. `gdaldata` é do tipo `bytea` e deve ser o conteúdo do arquivo raster GDAL.

Se `srid` for NULO, a função tentará designar automaticamente o SRID do raster GDAL. Se `srid` for fornecido, o valor fornecido irá exceder qualquer SRID designado automaticamente.

Disponibilidade: 2.1.0

## Exemplos

```
WITH foo AS (
  SELECT ST_AsPNG(ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, ←
    0.1, -0.1, 0, 0, 4326), 1, '8BUI', 1, 0), 2, '8BUI', 2, 0), 3, '8BUI', 3, 0)) AS ←
    png
),
bar AS (
  SELECT 1 AS rid, ST_FromGDALRaster(png) AS rast FROM foo
  UNION ALL
  SELECT 2 AS rid, ST_FromGDALRaster(png, 3310) AS rast FROM foo
)
SELECT
  rid,
  ST_Metadata(rast) AS metadata,
  ST_SummaryStats(rast, 1) AS stats1,
  ST_SummaryStats(rast, 2) AS stats2,
  ST_SummaryStats(rast, 3) AS stats3
FROM bar
ORDER BY rid;
```

rid	metadata	stats1	stats2	stats3
1	(0,0,2,2,1,-1,0,0,0,3)	(4,4,1,0,1,1)	(4,8,2,0,2,2)	(4,12,3,0,3,3)
2	(0,0,2,2,1,-1,0,0,3310,3)	(4,4,1,0,1,1)	(4,8,2,0,2,2)	(4,12,3,0,3,3)

(2 rows)

## Veja também

[ST\\_AsGDALRaster](#)

## 9.4 Assessores Raster

### 9.4.1 ST\_GeoReference

`ST_GeoReference` — Retorna os metadados georreferenciados no formato GDAL ou ESRI como é comumente visto em um arquivo mundo. O padrão é GDAL.

#### Synopsis

```
text ST_GeoReference(raster rast, text format=GDAL);
```

#### Descrição

Retorna os metadados georreferenciados, incluindo transporte, no formato GDAL ou ESRI como visto no `srid`. O padrão é GDAL se nenhum tipo for especificado. O tipo é string 'GDAL' ou 'ESRI'.

A diferença entre representações de formatos é a seguinte:

GDAL:

```
scalex
skewy
skewx
scaley
upperleftx
upperlefty
```

ESRI:

```
scalex
skewy
skewx
scaley
upperleftx + scalex*0.5
upperlefty + scaley*0.5
```

## Exemplos

```
SELECT ST_GeoReference(rast, 'ESRI') As esri_ref, ST_GeoReference(rast, 'GDAL') As gdal_ref
FROM dummy_rast WHERE rid=1;
```

esri_ref	gdal_ref
2.0000000000	2.0000000000
0.0000000000	: 0.0000000000
0.0000000000	: 0.0000000000
3.0000000000	: 3.0000000000
1.5000000000	: 0.5000000000
2.0000000000	: 0.5000000000

## Veja também

[ST\\_SetGeoReference](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#)

## 9.4.2 ST\_Height

`ST_Height` — Retorna a altura do raster em pixels.

### Synopsis

```
integer ST_Height(raster rast);
```

### Descrição

Retorna a altura do raster.

### Exemplos

```
SELECT rid, ST_Height(rast) As rastheight
FROM dummy_rast;
```

rid	rastheight
1	20
2	5

## Veja também

[ST\\_Width](#)

### 9.4.3 ST\_IsEmpty

ST\_IsEmpty — Retorna verdadeiro se o raster estiver vazio (largura = 0 e altura = 0). Senão, retorna falso.

#### Synopsis

boolean **ST\_IsEmpty**(raster rast);

#### Descrição

Retorna verdadeiro se o raster estiver vazio (largura = 0 e altura = 0). Senão, retorna falso.

Disponibilidade: 2.0.0

#### Exemplos

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(100, 100, 0, 0, 0, 0, 0, 0))
st_isempty |
-----+
f          |

SELECT ST_IsEmpty(ST_MakeEmptyRaster(0, 0, 0, 0, 0, 0, 0, 0))
st_isempty |
-----+
t          |
```

#### Veja também

[ST\\_HasNoBand](#)

### 9.4.4 ST\_MemSize

ST\_MemSize — Retorna a quantidade de espaço (em bytes) que o raster pega.

#### Synopsis

integer **ST\_MemSize**(raster rast);

#### Descrição

Retorna a quantidade de espaço (em bytes) que o raster pega.

Isso é um ótimo elogio para o PostgreSQL construído nas funções `pg_column_size`, `pg_size_pretty`, `pg_relation_size`, `pg_total_relation_size`.



#### Note

`pg_relation_size` que fornece o tamanho em bytes de uma tabela, talvez retorne com o tamanho em byte menor que `ST_MemSize`. Isso acontece porque o `pg_relation_size` não adiciona contribuição de tabela toasted e grandes geometrias são guardadas em tabelas TOAST. `g_column_size` pode retornar menor porque retorna o tamanho comprimido. `pg_total_relation_size` - inclui, a tabela, as tabelas toasted, e os índices.

Disponibilidade: 2.2.0

**Exemplos**

```
SELECT ST_MemSize(ST_AsRaster(ST_Buffer(ST_Point(1,5),10,1000),150, 150, '8BUI')) As rast_mem;
--
rast_mem
-----
22568
```

**Veja também**

**9.4.5 ST\_MetaData**

ST\_MetaData — Retorna metadados básicos sobre um objeto raster como um tanho pixel, rotação (skew), esquerda superior, inferior etc.

**Synopsis**

record **ST\_MetaData**(raster rast);

**Descrição**

Retorna metadados básicos sobre um objeto raster como um tanho pixel, rotação (skew), esquerda superior, inferior etc. Colunas retornadas: upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands

**Exemplos**

```
SELECT rid, (foo.md).*
FROM (SELECT rid, ST_MetaData(rast) As md
FROM dummy_rast) As foo;
```

rid	upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	numbands
1	0	0	10	20	2	3			0	
2	3427927.75	5793244	5	5	0.05	-0.05			0	3

**Veja também**

[ST\\_BandMetaData](#), [ST\\_NumBands](#)

**9.4.6 ST\_NumBands**

ST\_NumBands — Retorna o número de bandas no objeto raster.

**Synopsis**

integer **ST\_NumBands**(raster rast);

**Descrição**

Retorna o número de bandas no objeto raster.

**Exemplos**

```
SELECT rid, ST_NumBands(rast) As numbands
FROM dummy_rast;
```

rid	numbands
1	0
2	3

**Veja também**

[ST\\_Value](#)

**9.4.7 ST\_PixelHeight**

**ST\_PixelHeight** — Retorna a altura do pixel em unidades geométricas do sistema de referência espacial.

**Synopsis**

double precision **ST\_PixelHeight**(raster rast);

**Descrição**

Retorna a altura do pixel em unidades geométricas do sistema de referência espacial. No caso comum onde não existem desvios, a altura do pixel é somente a escala de proporção entre coordenadas geométricas e pixels raster.

Recorra a [ST\\_PixelWidth](#) para uma visualização diagramática da relação.

**Exemplos: Rasters sem desvio**

```
SELECT ST_Height(rast) As rastheight, ST_PixelHeight(rast) As pixheight,
       ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
       ST_SkewY(rast) As skewy
FROM dummy_rast;
```

rastheight	pixheight	scalex	scaley	skewx	skewy
20	3	2	3	0	0
5	0.05	0.05	-0.05	0	0

**Exemplos: Rasters com desvio diferente de 0**

```
SELECT ST_Height(rast) As rastheight, ST_PixelHeight(rast) As pixheight,
       ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
       ST_SkewY(rast) As skewy
FROM (SELECT ST_SetSKew(rast,0.5,0.5) As rast
      FROM dummy_rast) As skewed;
```

rastheight	pixheight	scalex	scaley	skewx	skewy
20	3.04138126514911	2	3	0.5	0.5
5	0.502493781056044	0.05	-0.05	0.5	0.5

### Veja também

[ST\\_PixelWidth](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#), [ST\\_SkewX](#), [ST\\_SkewY](#)

## 9.4.8 ST\_PixelWidth

`ST_PixelWidth` — Retorna a largura do pixel em unidades geométricas do sistema de referência espacial.

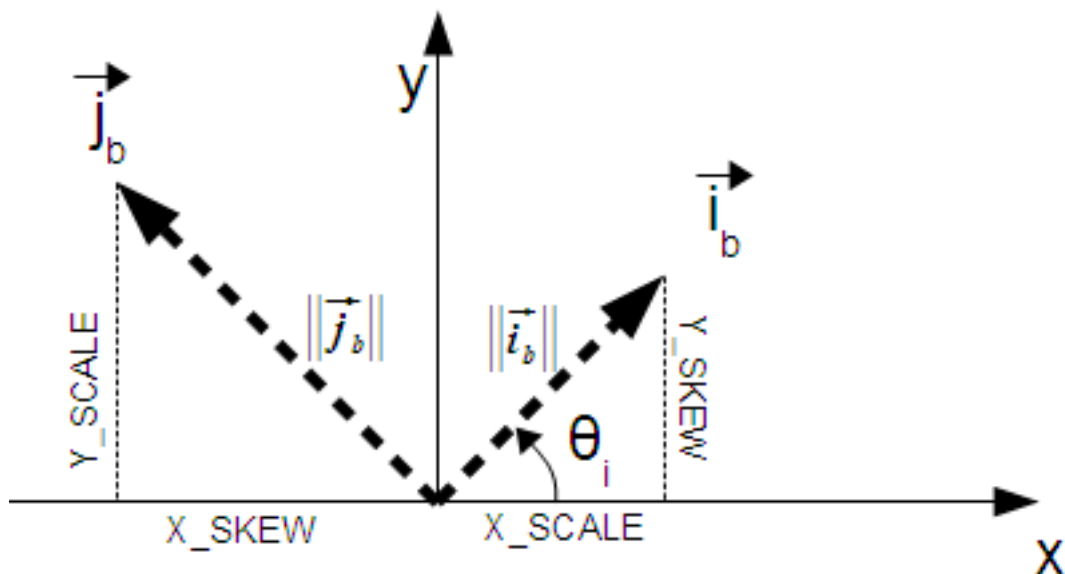
### Synopsis

double precision `ST_PixelWidth`(raster rast);

### Descrição

Retorna a largura do pixel em unidades geométricas do sistema de referência espacial. No caso comum onde não existem desvios, a largura do pixel é somente a escala de proporção entre coordenadas geométricas e pixels raster.

O diagrama a seguir demonstra a relação:



*Largura do Pixel: tamanho do pixel na direção  $i$*   
*Altura do Pixel: tamanho do pixel na direção  $j$*

### Exemplos: Rasters sem desvio

```
SELECT ST_Width(rast) As rastwidth, ST_PixelWidth(rast) As pixwidth,
       ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
       ST_SkewY(rast) As skewy
FROM dummy_rast;
```

rastwidth	pixwidth	scalex	scaley	skewx	skewy
10	2	2	3	0	0
5	0.05	0.05	-0.05	0	0

### Exemplos: Rasters com desvio diferente de 0

```
SELECT ST_Width(rast) As rastwidth, ST_PixelWidth(rast) As pixwidth,
       ST_ScaleX(rast) As scalex, ST_ScaleY(rast) As scaley, ST_SkewX(rast) As skewx,
       ST_SkewY(rast) As skewy
FROM (SELECT ST_SetSkew(rast,0.5,0.5) As rast
      FROM dummy_rast) As skewed;
```

rastwidth	pixwidth	scalex	scaley	skewx	skewy
10	2.06155281280883	2	3	0.5	0.5
5	0.502493781056044	0.05	-0.05	0.5	0.5

### Veja também

[ST\\_PixelHeight](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#), [ST\\_SkewX](#), [ST\\_SkewY](#)

## 9.4.9 ST\_ScaleX

**ST\_ScaleX** — Retorna o componente X da largura do pixel em unidades do sistema de referência coordenadas.

### Synopsis

```
float8 ST_ScaleX(raster rast);
```

### Descrição

Retorna o componente X da largura do pixel em unidades do sistema de referência coordenadas. Recorra a [World File](#) para mais detalhes.

Alterações: 2.0.0. Nas versões WKTRaster era chamado de `ST_PixelSizeX`.

### Exemplos

```
SELECT rid, ST_ScaleX(rast) As rastpixwidth
FROM dummy_rast;
```

rid	rastpixwidth
1	2
2	0.05

### Veja também

[ST\\_Width](#)



### 9.4.10 ST\_ScaleY

ST\_ScaleY — Retorna o componente Y da altura do pixel em unidades do sistema de referência coordenadas.

#### Synopsis

```
float8 ST_ScaleY(raster rast);
```

#### Descrição

Retorna o componente Y da altura do pixel em unidades do sistema de referência coordenadas. Recorra a [World File](#) para mais detalhes.

Alterações: 2.0.0. Nas versões WKTRaster era chamado de ST\_PixelSizeY.

#### Exemplos

```
SELECT rid, ST_ScaleY(rast) As rastpixheight
FROM dummy_rast;
```

rid	rastpixheight
1	3
2	-0.05

#### Veja também

[ST\\_Height](#)

### 9.4.11 ST\_RasterToWorldCoord

ST\_RasterToWorldCoord — Retorna o canto superior esquerdo do raster como X e Y geométricos (longitude e latitude) dada a coluna e linha. Coluna e linha começam em 1.

#### Synopsis

```
record ST_RasterToWorldCoord(raster rast, integer xcolumn, integer yrow);
```

#### Descrição

Retorna o canto esquerdo superior como X e Y geométricos (longitude e latitude) dada a coluna e linha. o X e Y estão em unidades geométricas do raster georreferenciado. A numeração da coluna e da linha começa no 1, mas se algum passar como zero, um número negativo ou um maior que a respectiva dimensão do raster, retornará as coordenadas fora do raster assumindo que a rede raster é aplicável fora dos limites do raster.

Disponibilidade: 2.1.0

## Exemplos

```
-- non-skewed raster
SELECT
    rid,
    (ST_RasterToWorldCoord(rast, 1, 1)).*,
    (ST_RasterToWorldCoord(rast, 2, 2)).*
FROM dummy_rast
```

rid	longitude	latitude	longitude	latitude
1	0.5	0.5	2.5	3.5
2	3427927.75	5793244	3427927.8	5793243.95

```
-- skewed raster
SELECT
    rid,
    (ST_RasterToWorldCoord(rast, 1, 1)).*,
    (ST_RasterToWorldCoord(rast, 2, 3)).*
FROM (
    SELECT
        rid,
        ST_SetSkew(rast, 100.5, 0) As rast
    FROM dummy_rast
) As foo
```

rid	longitude	latitude	longitude	latitude
1	0.5	0.5	203.5	6.5
2	3427927.75	5793244	3428128.8	5793243.9

## Veja também

[ST\\_RasterToWorldCoordX](#), [ST\\_RasterToWorldCoordY](#), [ST\\_SetSkew](#)

### 9.4.12 ST\_RasterToWorldCoordX

`ST_RasterToWorldCoordX` — Retorna a coordenada geométrica X superior esquerda de um raster, coluna ou linha. A numeração das colunas e linhas começam no 1.

#### Synopsis

```
float8 ST_RasterToWorldCoordX(raster rast, integer xcolumn);
float8 ST_RasterToWorldCoordX(raster rast, integer xcolumn, integer yrow);
```

#### Descrição

Retorna a coordenada X superior esquerda de uma coluna raster em unidades geométricas do raster georreferenciado. A numeração da coluna e da linha começa no 1, mas se algum passar como zero, um número negativo ou um maior que a respectiva dimensão do raster, retornará as coordenadas fora do raster para esquerda ou direita, assumindo que a skew e tamanhos do pixel são os mesmo que o raster selecionado.



#### Note

Para rasters sem desvio, fornecer a coluna X é suficiente. Para rasters com desvio, a coordenada georreferenciada é uma função da `ST_ScaleX` e `ST_SkewX` e linha e coluna. Um erro aparecerá se você der somente a coluna X para um raster desviado.

Alterações: 2.1.0 Em versões anteriores, era chamado de `ST_Raster2WorldCoordX`

### Exemplos

```
-- non-skewed raster providing column is sufficient
SELECT rid, ST_RasterToWorldCoordX(rast,1) As x1coord,
         ST_RasterToWorldCoordX(rast,2) As x2coord,
         ST_ScaleX(rast) As pixelx
FROM dummy_rast;
```

rid	x1coord	x2coord	pixelx
1	0.5	2.5	2
2	3427927.75	3427927.8	0.05

```
-- for fun lets skew it
SELECT rid, ST_RasterToWorldCoordX(rast, 1, 1) As x1coord,
         ST_RasterToWorldCoordX(rast, 2, 3) As x2coord,
         ST_ScaleX(rast) As pixelx
FROM (SELECT rid, ST_SetSkew(rast, 100.5, 0) As rast FROM dummy_rast) As foo;
```

rid	x1coord	x2coord	pixelx
1	0.5	203.5	2
2	3427927.75	3428128.8	0.05

### Veja também

[ST\\_ScaleX](#), [ST\\_RasterToWorldCoordY](#), [ST\\_SetSkew](#), [ST\\_SkewX](#)

### 9.4.13 ST\_RasterToWorldCoordY

`ST_RasterToWorldCoordY` — Retorna a coordenada geométrica Y superior esquerda de um raster, coluna e linha. A numeração das colunas e linhas começam no 1.

#### Synopsis

```
float8 ST_RasterToWorldCoordY(raster rast, integer yrow);
float8 ST_RasterToWorldCoordY(raster rast, integer xcolumn, integer yrow);
```

#### Descrição

Retorna a coordenada Y superior esquerda de uma coluna raster em unidades geométricas do raster georreferenciado. A numeração da coluna e da linha começa no 1, mas se algum passar como zero, um número negativo ou um maior que a respectiva dimensão do raster, retornará as coordenadas fora do raster para esquerda ou direita, assumindo que o desvio e tamanhos do pixel são os mesmos que o raster selecionado.



#### Note

Para rasters sem desvio, fornecer a coluna Y é suficiente. Para rasters com desvio, a coordenada georreferenciada é uma função da `ST_ScaleY` e `ST_SkewY` e linha e coluna. Um erro aparecerá se você der somente a linha Y para um raster desviado.

Alterações: 2.1.0 Em versões anteriores, era chamado de `ST_Raster2WorldCoordY`

## Exemplos

```
-- non-skewed raster providing row is sufficient
SELECT rid, ST_RasterToWorldCoordY(rast,1) As y1coord,
        ST_RasterToWorldCoordY(rast,3) As y2coord,
        ST_ScaleY(rast) As pixely
FROM dummy_rast;
```

rid	y1coord	y2coord	pixely
1	0.5	6.5	3
2	5793244	5793243.9	-0.05

```
-- for fun lets skew it
SELECT rid, ST_RasterToWorldCoordY(rast,1,1) As y1coord,
        ST_RasterToWorldCoordY(rast,2,3) As y2coord,
        ST_ScaleY(rast) As pixely
FROM (SELECT rid, ST_SetSkew(rast,0,100.5) As rast FROM dummy_rast) As foo;
```

rid	y1coord	y2coord	pixely
1	0.5	107	3
2	5793244	5793344.4	-0.05

## Veja também

[ST\\_ScaleY](#), [ST\\_RasterToWorldCoordX](#), [ST\\_SetSkew](#), [ST\\_SkewY](#)

### 9.4.14 ST\_Rotation

`ST_Rotation` — Retorna a rotação do raster em radianos.

#### Synopsis

```
float8 ST_Rotation(raster rast);
```

#### Descrição

Retorna a rotação uniforme do raster em radianos. Se um raster não tiver uma rotação uniforme, NaN retorna. Recorra a [World File](#) para mais detalhes.

#### Exemplos

```
SELECT rid, ST_Rotation(ST_SetScale(ST_SetSkew(rast, sqrt(2)), sqrt(2))) as rot FROM ↵
dummy_rast;
```

rid	rot
1	0.785398163397448
2	0.785398163397448

## Veja também

[ST\\_SetRotation](#), [ST\\_SetScale](#), [ST\\_SetSkew](#)

### 9.4.15 ST\_SkewX

ST\_SkewX — Retorna o desvio X georreferência (ou parâmetro e rotação).

#### Synopsis

```
float8 ST_SkewX(raster rast);
```

#### Descrição

Retorna o desvio X georreferência (ou parâmetro de rotação). Recorra a [World File](#) para mais detalhes.

#### Exemplos

```
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
       ST_GeoReference(rast) as georef
FROM dummy_rast;
```

rid	skewx	skewy	georef
1	0	0	2.0000000000 : 0.0000000000 : 0.0000000000 : 3.0000000000 : 0.5000000000 : 0.5000000000 :
2	0	0	0.0500000000 : 0.0000000000 : 0.0000000000 : -0.0500000000 : 3427927.7500000000 : 5793244.0000000000

#### Veja também

[ST\\_GeoReference](#), [ST\\_SkewY](#), [ST\\_SetSkew](#)

### 9.4.16 ST\_SkewY

ST\_SkewY — Retorna o desvio Y georreferência (ou parâmetro e rotação).

#### Synopsis

```
float8 ST_SkewY(raster rast);
```

#### Descrição

Retorna o desvio Y georreferência (ou parâmetro de rotação). Recorra a [World File](#) para mais detalhes.

## Exemplos

```
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
       ST_GeoReference(rast) as georef
FROM dummy_rast;
```

rid	skewx	skewy	georef
1	0	0	2.0000000000 : 0.0000000000 : 0.0000000000 : 3.0000000000 : 0.5000000000 : 0.5000000000 :
2	0	0	0.0500000000 : 0.0000000000 : 0.0000000000 : -0.0500000000 : 3427927.7500000000 : 5793244.0000000000

## Veja também

[ST\\_GeoReference](#), [ST\\_SkewX](#), [ST\\_SetSkew](#)

### 9.4.17 ST\_SRID

**ST\_SRID** — Retorna o identificador de referência espacial como definido na tabela `spatial_ref_sys`.

#### Synopsis

```
integer ST_SRID(raster rast);
```

#### Descrição

Retorna o identificador de referência espacial do objeto raster como definido na tabela `spatial_ref_sys`.



#### Note

A partir do PostGIS 2.0+ o srid de raster/geometria não georreferenciado é 0 em vez de -1.

## Exemplos

```
SELECT ST_SRID(rast) As srid
FROM dummy_rast WHERE rid=1;
```

srid
0

**Veja também**

Section [4.3.1](#), [ST\\_SRID](#)

**9.4.18 ST\_Summary**

`ST_Summary` — Retorna um texto resumo dos conteúdos do raster.

**Synopsis**

```
text ST_Summary(raster rast);
```

**Descrição**

Retorna um texto resumo dos conteúdos do raster.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT ST_Summary(
  ST_AddBand(
    ST_AddBand(
      ST_AddBand(
        ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1, 0, 0, 0)
        , 1, '8BUI', 1, 0
      )
      , 2, '32BF', 0, -9999
    )
    , 3, '16BSI', 0, NULL
  )
);

-----
st_summary
-----
Raster of 10x10 pixels has 3 bands and extent of BOX(0 -10,10 0)+
band 1 of pixtype 8BUI is in-db with NODATA value of 0      +
band 2 of pixtype 32BF is in-db with NODATA value of -9999 +
band 3 of pixtype 16BSI is in-db with no NODATA value
(1 row)
```

**Veja também**

[ST\\_MetaData](#), [ST\\_BandMetaData](#), [ST\\_Summary](#) [ST\\_Extent](#)

**9.4.19 ST\_UpperLeftX**

`ST_UpperLeftX` — Retorna a coordenada X superior esquerda na ref. espacial projetada.

**Synopsis**

```
float8 ST_UpperLeftX(raster rast);
```

**Descrição**

Retorna a coordenada X superior esquerda na ref. espacial projetada.

**Exemplos**

```
SELECT rid, ST_UpperLeftX(rast) As ulx
FROM dummy_rast;
```

rid	ulx
1	0.5
2	3427927.75

**Veja também**

[ST\\_UpperLeftY](#), [ST\\_GeoReference](#), [Caixa3D](#)

**9.4.20 ST\_UpperLeftY**

ST\_UpperLeftY — Retorna a coordenada Y superior esquerda na ref. espacial projetada.

**Synopsis**

```
float8 ST_UpperLeftY(raster rast);
```

**Descrição**

Retorna a coordenada Y superior esquerda na ref. espacial projetada.

**Exemplos**

```
SELECT rid, ST_UpperLeftY(rast) As uly
FROM dummy_rast;
```

rid	uly
1	0.5
2	5793244

**Veja também**

[ST\\_UpperLeftX](#), [ST\\_GeoReference](#), [Caixa3D](#)

**9.4.21 ST\_Width**

ST\_Width — Retorna a largura do raster em pixels.

**Synopsis**

```
integer ST_Width(raster rast);
```



**Descrição**

Retorna a largura do raster em pixels.

**Exemplos**

```
SELECT ST_Width(rast) As rastwidth
FROM dummy_rast WHERE rid=1;
```

```
rastwidth
-----
10
```

**Veja também**

[ST\\_Height](#)

**9.4.22 ST\_WorldToRasterCoord**

`ST_WorldToRasterCoord` — Retorna o canto superior esquerdo como coluna e linha dados os X e Y geométricos (longitude e latitude) ou um ponto expressado na coordenada do sistema de referência espacial do raster.

**Synopsis**

```
record ST_WorldToRasterCoord(raster rast, geometry pt);
record ST_WorldToRasterCoord(raster rast, double precision longitude, double precision latitude);
```

**Descrição**

Retorna o canto superior esquerdo como coluna e linha dados os X e Y geométricos (longitude e latitude) ou um ponto. Esta função funciona se o X e Y geométricos ou ponto estiver fora da extensão do raster ou não. O X e Y geométricos devem ser expressados na coordenada do sistema de referência espacial do raster.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT
    rid,
    (ST_WorldToRasterCoord(rast, 3427927.8, 20.5)).*,
    (ST_WorldToRasterCoord(rast, ST_GeomFromText('POINT(3427927.8 20.5)', ST_SRID(rast))) ↔
    ).*
FROM dummy_rast;
```

rid	columnx	rowy	columnx	rowy
1	1713964	7	1713964	7
2	2	115864471	2	115864471

**Veja também**

[ST\\_WorldToRasterCoordX](#), [ST\\_WorldToRasterCoordY](#), [ST\\_RasterToWorldCoordX](#), [ST\\_RasterToWorldCoordY](#), [ST\\_SRID](#)

### 9.4.23 ST\_WorldToRasterCoordX

`ST_WorldToRasterCoordX` — Retorna a coluna no raster do ponto (pt) ou uma coordenada X e Y (xw, yw) representada no sistema de referência espacial mundial de raster.

#### Synopsis

```
integer ST_WorldToRasterCoordX(raster rast, geometry pt);
integer ST_WorldToRasterCoordX(raster rast, double precision xw);
integer ST_WorldToRasterCoordX(raster rast, double precision xw, double precision yw);
```

#### Descrição

Retorna a coluna no raster do ponto (pt) ou uma coordenada X e Y (xw, yw). Um ponto, ou (ambas as coordenadas xw e yw são requeridas se um raster estiver desviado). Se um raster não estiver desviado, então xw é o suficiente. Coordenadas globais estão no sistema de coordenadas de referência espacial do raster.

Alterações: 2.1.0 Em versões anteriores, era chamado de `ST_World2RasterCoordX`

#### Exemplos

```
SELECT rid, ST_WorldToRasterCoordX(rast,3427927.8) As xcoord,
          ST_WorldToRasterCoordX(rast,3427927.8,20.5) As xcoord_xwyw,
          ST_WorldToRasterCoordX(rast,ST_GeomFromText('POINT(3427927.8 20.5)',ST_SRID ←
          (rast))) As ptxcoord
FROM dummy_rast;
```

rid	xcoord	xcoord_xwyw	ptxcoord
1	1713964	1713964	1713964
2	1	1	1

#### Veja também

[ST\\_RasterToWorldCoordX](#), [ST\\_RasterToWorldCoordY](#), [ST\\_SRID](#)

### 9.4.24 ST\_WorldToRasterCoordY

`ST_WorldToRasterCoordY` — Retorna a linha no raster do ponto (pt) ou uma coordenada X e Y (xw, yw) representada no sistema de referência espacial global de raster.

#### Synopsis

```
integer ST_WorldToRasterCoordY(raster rast, geometry pt);
integer ST_WorldToRasterCoordY(raster rast, double precision xw);
integer ST_WorldToRasterCoordY(raster rast, double precision xw, double precision yw);
```

#### Descrição

Retorna a linha no raster do ponto (pt) ou uma coordenada X e Y (xw, yw). Um ponto, ou (ambas as coordenadas xw e yw são requeridas se um raster estiver desviado). Se um raster não estiver desviado, então xw é o suficiente. Coordenadas globais estão no sistema de coordenadas de referência espacial do raster.

Alterações: 2.1.0 Em versões anteriores, era chamado de `ST_World2RasterCoordY`

## Exemplos

```
SELECT rid, ST_WorldToRasterCoordY(rast,20.5) As ycoord,
        ST_WorldToRasterCoordY(rast,3427927.8,20.5) As ycoord_xwyw,
        ST_WorldToRasterCoordY(rast,ST_GeomFromText('POINT(3427927.8 20.5)',ST_SRID ←
        (rast))) As ptycoord
FROM dummy_rast;
```

rid	ycoord	ycoord_xwyw	ptycoord
1	7	7	7
2	115864471	115864471	115864471

## Veja também

[ST\\_RasterToWorldCoordX](#), [ST\\_RasterToWorldCoordY](#), [ST\\_SRID](#)

## 9.5 Assessores de banda raster

### 9.5.1 ST\_BandMetaData

`ST_BandMetaData` — Retorna os metadados básicos para uma banda raster especificada. banda número 1 é assumida se nenhuma for especificada.

#### Synopsis

- (1) record `ST_BandMetaData`(raster rast, integer band=1);
- (2) record `ST_BandMetaData`(raster rast, integer[] band);

#### Descrição

Returns basic meta data about a raster band. Columns returned: `pixeltype`, `nodatavalue`, `isoutdb`, `path`, `outdbbandnum`, `filesize`, `filetimestamp`.



#### Note

Se o raster não contém nenhuma banda, então surge um erro.



#### Note

If band has no NODATA value, `nodatavalue` are NULL.



#### Note

If `isoutdb` is False, `path`, `outdbbandnum`, `filesize` and `filetimestamp` are NULL. If `outdb` access is disabled, `filesize` and `filetimestamp` will also be NULL.

Enhanced: 2.5.0 to include `outdbbandnum`, `filesize` and `filetimestamp` for `outdb` rasters.

**Exemplos: Variante 1**

```

SELECT
    rid,
    (foo.md).*
FROM (
    SELECT
        rid,
        ST_BandMetaData(rast, 1) AS md
    FROM dummy_rast
    WHERE rid=2
) As foo;

```

rid	pixeltype	nodatavalue	isoutdb	path	outdbbandnum
2	8BUI		0	f	

**Exemplos: Variant 2**

```

WITH foo AS (
    SELECT
        ST_AddBand(NULL::raster, '/home/pele/devel/geo/postgis-git/raster/test/ ↵
        regress/loader/Projected.tif', NULL::int[]) AS rast
)
SELECT
    *
FROM ST_BandMetadata(
    (SELECT rast FROM foo),
    ARRAY[1,3,2]::int[]
);

```

bandnum	pixeltype	nodatavalue	isoutdb	path	outdbbandnum	filesize	filetimestamp
1	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test ↵ /regress/loader/Projected.tif	1	12345	1521807257
3	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test ↵ /regress/loader/Projected.tif	3	12345	1521807257
2	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test ↵ /regress/loader/Projected.tif	2	12345	1521807257

**Veja também**

[ST\\_MetaData](#), [ST\\_BandPixelType](#)

**9.5.2 ST\_BandNoDataValue**

**ST\_BandNoDataValue** — Retorna o valor em uma dada banda que não representa nenhum valor. Se nenhuma banda número 1 for assumida.

**Synopsis**

```
double precision ST_BandNoDataValue(raster rast, integer bandnum=1);
```

## Descrição

Retorna o valor que não representa nenhum dado para a banda

## Exemplos

```
SELECT ST_BandNoDataValue(rast,1) As bnval1,
       ST_BandNoDataValue(rast,2) As bnval2, ST_BandNoDataValue(rast,3) As bnval3
FROM dummy_rast
WHERE rid = 2;
```

bnval1	bnval2	bnval3
0	0	0

## Veja também

[ST\\_NumBands](#)

### 9.5.3 ST\_BandIsNoData

ST\_BandIsNoData — Retorna verdadeiro se a banda estiver repleta somente de valores nodata.

## Synopsis

```
boolean ST_BandIsNoData(raster rast, integer band, boolean forceChecking=true);
boolean ST_BandIsNoData(raster rast, boolean forceChecking=true);
```

## Descrição

Retorna verdadeiro se a banda estiver repleta apenas de valores nodata. Banda 1 é assumida se não especificada. Se o último argumento for VERDADEIRO, a banda toda é verificada pixel por pixel. Caso contrário, a função simplesmente retorna o valor da bandeira isnodata para a banda. O valor padrão para este parâmetro é FALSO, se não especificado.

Disponibilidade: 2.0.0



### Note

Se a bandeira for suja (ou seja, o resultado for diferente usando VERDADEIRO como último parâmetro e não usando ele) você deveria atualizar o raster para tornar esta bandeira verdadeira, usando a [ST\\_SetBandIsNodata\(\)](#), ou [ST\\_SetBandNodataValue\(\)](#) com VERDADEIRO como último argumento. Veja [ST\\_SetBandIsNoData](#).

## Exemplos

```
-- Create dummy table with one raster column
create table dummy_rast (rid integer, rast raster);

-- Add raster with two bands, one pixel/band. In the first band, nodatavalue = pixel value ←
= 3.
-- In the second band, nodatavalue = 13, pixel value = 4
insert into dummy_rast values(1,
(
'01' -- little endian (uint8 ndr)
```

```

||
'0000' -- version (uint16 0)
||
'0200' -- nBands (uint16 0)
||
'17263529ED684A3F' -- scaleX (float64 0.000805965234044584)
||
'F9253529ED684ABF' -- scaleY (float64 -0.00080596523404458)
||
'1C9F33CE69E352C0' -- ipX (float64 -75.5533328537098)
||
'718F0E9A27A44840' -- ipY (float64 49.2824585505576)
||
'ED50EB853EC32B3F' -- skewX (float64 0.000211812383858707)
||
'7550EB853EC32B3F' -- skewY (float64 0.000211812383858704)
||
'E6100000' -- SRID (int32 4326)
||
'0100' -- width (uint16 1)
||
'0100' -- height (uint16 1)
||
'6' -- hasnodatavalue and isnodata value set to true.
||
'2' -- first band type (4BUI)
||
'03' -- novalue==3
||
'03' -- pixel(0,0)==3 (same that nodata)
||
'0' -- hasnodatavalue set to false
||
'5' -- second band type (16BSI)
||
'0D00' -- novalue==13
||
'0400' -- pixel(0,0)==4
)::raster
);

select st_bandisnodata(rast, 1) from dummy_rast where rid = 1; -- Expected true
select st_bandisnodata(rast, 2) from dummy_rast where rid = 1; -- Expected false

```

**Veja também**

[ST\\_BandNoDataValue](#), [ST\\_NumBands](#), [ST\\_SetBandNoDataValue](#), [ST\\_SetBandIsNoData](#)

**9.5.4 ST\_BandPath**

**ST\_BandPath** — Retorna o caminho do arquivo do sistema para uma banda armazenada em um sistema de arquivos. Se nenhum número de banda for especificado, usa-se 1.

**Synopsis**

```
text ST_BandPath(raster rast, integer bandnum=1);
```

## Descrição

Retorna o caminho do arquivo do sistema para uma banda. Surge um erro se for chamado com uma banda no banco de dados.

## Exemplos

### Veja também

## 9.5.5 ST\_BandFileSize

`ST_BandFileSize` — Returns the file size of a band stored in file system. If no bandnum specified, 1 is assumed.

### Synopsis

```
bigint ST_BandFileSize(raster rast, integer bandnum=1);
```

### Descrição

Returns the file size of a band stored in file system. Throws an error if called with an in db band, or if outdb access is not enabled.

This function is typically used in conjunction with `ST_BandPath()` and `ST_BandFileTimestamp()` so a client can determine if the filename of a outdb raster as seen by it is the same as the one seen by the server.

Availability: 2.5.0

### Exemplos

```
SELECT ST_BandFileSize(rast,1) FROM dummy_rast WHERE rid = 1;
```

```
st_bandfilesize
-----
          240574
```

## 9.5.6 ST\_BandFileTimestamp

`ST_BandFileTimestamp` — Returns the file timestamp of a band stored in file system. If no bandnum specified, 1 is assumed.

### Synopsis

```
bigint ST_BandFileTimestamp(raster rast, integer bandnum=1);
```

### Descrição

Returns the file timestamp (number of seconds since Jan 1st 1970 00:00:00 UTC) of a band stored in file system. Throws an error if called with an in db band, or if outdb access is not enabled.

This function is typically used in conjunction with `ST_BandPath()` and `ST_BandFileSize()` so a client can determine if the filename of a outdb raster as seen by it is the same as the one seen by the server.

Availability: 2.5.0

## Exemplos

```
SELECT ST_BandFileTimestamp(rast,1) FROM dummy_rast WHERE rid = 1;

 st_bandfiletimestamp
-----
          1521807257
```

## 9.5.7 ST\_BandPixelType

ST\_BandPixelType — Retorna o tipo pixel para uma dada banda. Se nenhum número de banda for especificado, usa-se 1.

### Synopsis

text **ST\_BandPixelType**(raster rast, integer bandnum=1);

### Descrição

Retorna o valor que não representa nenhum dado para a banda

Existem 11 tipos de pixel. Os tipos suportados são os seguintes:

- 1BB - 1-bit boolean
- 2BUI - 2-bit inteiro não assinado
- 4BUI - 4-bit inteiro não assinado
- 8BSI - 8-bit inteiro assinado
- 8BUI - 8-bit inteiro não assinado
- 16BSI - 16-bit inteiro assinado
- 16BUI - 16-bit inteiro não assinado
- 32BSI - 32-bit inteiro assinado
- 32BUI - 32-bit inteiro não assinado
- 32BF - 32-bit float
- 64BF - 64-bit float

## Exemplos

```
SELECT ST_BandPixelType(rast,1) As btype1,
       ST_BandPixelType(rast,2) As btype2, ST_BandPixelType(rast,3) As btype3
FROM dummy_rast
WHERE rid = 2;

 btype1 | btype2 | btype3
-----+-----+-----
  8BUI  |  8BUI  |  8BUI
```

## Veja também

[ST\\_NumBands](#)



## 9.5.8 ST\_HasNoBand

`ST_HasNoBand` — Retorna verdade se não existirem bandas com números dados. Se nenhum número de banda for especificado, então assume-se a banda 1.

### Synopsis

boolean `ST_HasNoBand`(raster rast, integer bandnum=1);

### Descrição

Retorna verdade se não existirem bandas com números dados. Se nenhum número de banda for especificado, então assume-se a banda 1.

Disponibilidade: 2.0.0

### Exemplos

```
SELECT rid, ST_HasNoBand(rast) As hb1, ST_HasNoBand(rast,2) as hb2,
ST_HasNoBand(rast,4) as hb4, ST_NumBands(rast) As numbands
FROM dummy_rast;
```

rid	hb1	hb2	hb4	numbands
1	t	t	t	0
2	f	f	t	3

### Veja também

[ST\\_NumBands](#)

## 9.6 Assessores e Setters de Pixel Raster

### 9.6.1 ST\_PixelAsPolygon

`ST_PixelAsPolygon` — Retorna o polígono que limita o pixel para uma linha e coluna específicas.

### Synopsis

geometry `ST_PixelAsPolygon`(raster rast, integer columnx, integer rowy);

### Descrição

Retorna o polígono que limita o pixel para uma linha e coluna específicas.

Disponibilidade: 2.0.0

## Exemplos

```
-- get raster pixel polygon
SELECT i, j, ST_AsText(ST_PixelAsPolygon(foo.rast, i, j)) As blpgeom
FROM dummy_rast As foo
      CROSS JOIN generate_series(1,2) As i
      CROSS JOIN generate_series(1,1) As j
WHERE rid=2;
```

```
 i | j | blpgeom
---+---+-----
 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, ..
```

## Veja também

[ST\\_DumpAsPolygons](#), [ST\\_PixelAsPolygons](#), [ST\\_PixelAsPoint](#), [ST\\_PixelAsPoints](#), [ST\\_PixelAsCentroid](#), [ST\\_PixelAsCentroids](#), [ST\\_Intersection](#), [ST\\_AsText](#)

### 9.6.2 ST\_PixelAsPolygons

`ST_PixelAsPolygons` — Retorna o polígono que limita cada pixel de uma banda raster ao longo do valor, as coordenadas raster X e Y de cada pixel.

#### Synopsis

setof record `ST_PixelAsPolygons`(raster rast, integer band=1, boolean exclude\_nodata\_value=TRUE);

#### Descrição

Retorna o polígono que limita cada pixel de uma banda raster ao longo do valor (precisão dobrada), as coordenadas (inteiras) raster X e Y de cada pixel.

Return record format: *geom* **geometry**, *val* double precision, *x* integer, *y* integers.



#### Note

When `exclude_nodata_value = TRUE`, only those pixels whose values are not NODATA are returned as points.



#### Note

`ST_PixelAsPolygons` retorna um polígono para cada pixel. Isto é diferente da `ST_DumpAsPolygons`, onde cada geometria representa um ou mais pixels com o mesmo valor.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 o argumento opcional `exclude_nodata_value` foi adicionado.

Alterações: 2.1.1 Mudança no comportamento do `exclude_nodata_value`.

## Exemplos

```
-- get raster pixel polygon
SELECT (gv).x, (gv).y, (gv).val, ST_AsText((gv).geom) geom
FROM (SELECT ST_PixelAsPolygons(
         ST_SetValue(ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 0.001, ←
         -0.001, 0.001, 0.001, 4269),
         '8BUI'::text, 1, 0),
         2, 2, 10),
         1, 1, NULL)
) gv
) foo;
```

x	y	val	geom
1	1		POLYGON((0 0,0.001 0.001,0.002 0,0.001 -0.001,0 0))
1	2	1	POLYGON((0.001 -0.001,0.002 0,0.003 -0.001,0.002 -0.002,0.001 -0.001))
2	1	1	POLYGON((0.001 0.001,0.002 0.002,0.003 0.001,0.002 0,0.001 0.001))
2	2	10	POLYGON((0.002 0,0.003 0.001,0.004 0,0.003 -0.001,0.002 0))

## Veja também

[ST\\_DumpAsPolygons](#), [ST\\_PixelAsPolygon](#), [ST\\_PixelAsPoint](#), [ST\\_PixelAsPoints](#), [ST\\_PixelAsCentroid](#), [ST\\_PixelAsCentroids](#), [ST\\_AsText](#)

### 9.6.3 ST\_PixelAsPoint

`ST_PixelAsPoint` — Retorna um ponto geométrico do canto superior esquerdo do pixel.

#### Synopsis

geometry `ST_PixelAsPoint`(raster rast, integer columnx, integer rowy);

#### Descrição

Retorna um ponto geométrico do canto superior esquerdo do pixel.

Disponibilidade: 2.1.0

#### Exemplos

```
SELECT ST_AsText(ST_PixelAsPoint(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;
```

```
st_astext
-----
POINT(0.5 0.5)
```

## Veja também

[ST\\_DumpAsPolygons](#), [ST\\_PixelAsPolygon](#), [ST\\_PixelAsPolygons](#), [ST\\_PixelAsPoints](#), [ST\\_PixelAsCentroid](#), [ST\\_PixelAsCentroids](#)

## 9.6.4 ST\_PixelAsPoints

`ST_PixelAsPoints` — Retorna um ponto geométrico para cada pixel de uma banda raster junto com o valor, as coordenadas raster X e Y de cada pixel. As coordenadas do ponto são do ponto esquerdo superior do pixel.

### Synopsis

```
setof record ST_PixelAsPoints(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);
```

### Descrição

Retorna um ponto geométrico para cada pixel de uma banda raster junto com o valor, as coordenadas raster X e Y de cada pixel. As coordenadas do ponto são do ponto esquerdo superior do pixel.

Return record format: *geom* **geometry**, *val* double precision, *x* integer, *y* integers.



#### Note

When `exclude_nodata_value = TRUE`, only those pixels whose values are not NODATA are returned as points.

Disponibilidade: 2.1.0

Alterações: 2.1.1 Mudança no comportamento do `exclude_nodata_value`.

### Exemplos

```
SELECT x, y, val, ST_AsText(geom) FROM (SELECT (ST_PixelAsPoints(rast, 1)).* FROM ↵
    dummy_rast WHERE rid = 2) foo;
```

x	y	val	st_astext
1	1	253	POINT(3427927.75 5793244)
2	1	254	POINT(3427927.8 5793244)
3	1	253	POINT(3427927.85 5793244)
4	1	254	POINT(3427927.9 5793244)
5	1	254	POINT(3427927.95 5793244)
1	2	253	POINT(3427927.75 5793243.95)
2	2	254	POINT(3427927.8 5793243.95)
3	2	254	POINT(3427927.85 5793243.95)
4	2	253	POINT(3427927.9 5793243.95)
5	2	249	POINT(3427927.95 5793243.95)
1	3	250	POINT(3427927.75 5793243.9)
2	3	254	POINT(3427927.8 5793243.9)
3	3	254	POINT(3427927.85 5793243.9)
4	3	252	POINT(3427927.9 5793243.9)
5	3	249	POINT(3427927.95 5793243.9)
1	4	251	POINT(3427927.75 5793243.85)
2	4	253	POINT(3427927.8 5793243.85)
3	4	254	POINT(3427927.85 5793243.85)
4	4	254	POINT(3427927.9 5793243.85)
5	4	253	POINT(3427927.95 5793243.85)
1	5	252	POINT(3427927.75 5793243.8)
2	5	250	POINT(3427927.8 5793243.8)
3	5	254	POINT(3427927.85 5793243.8)
4	5	254	POINT(3427927.9 5793243.8)
5	5	254	POINT(3427927.95 5793243.8)

**Veja também**

[ST\\_DumpAsPolygons](#), [ST\\_PixelAsPolygon](#), [ST\\_PixelAsPolygons](#), [ST\\_PixelAsPoint](#), [ST\\_PixelAsCentroid](#), [ST\\_PixelAsCentroids](#)

**9.6.5 ST\_PixelAsCentroid**

`ST_PixelAsCentroid` — Retorna o centroide (ponto) da área representada por um pixel.

**Synopsis**

geometry `ST_PixelAsCentroid`(raster rast, integer x, integer y);

**Descrição**

Retorna o centroide (ponto) da área representada por um pixel.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT ST_AsText(ST_PixelAsCentroid(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;

 st_astext
-----
POINT(1.5 2)
```

**Veja também**

[ST\\_DumpAsPolygons](#), [ST\\_PixelAsPolygon](#), [ST\\_PixelAsPolygons](#), [ST\\_PixelAsPoint](#), [ST\\_PixelAsPoints](#), [ST\\_PixelAsCentroids](#)

**9.6.6 ST\_PixelAsCentroids**

`ST_PixelAsCentroids` — Retorna o centroide (ponto geométrico) para cada pixel de uma banda raster junto com o valor, as coordenadas raster X e Y de cada pixel. O ponto é o centroide da área representada por um pixel.

**Synopsis**

setof record `ST_PixelAsCentroids`(raster rast, integer band=1, boolean exclude\_nodata\_value=TRUE);

**Descrição**

Retorna o centroide (ponto geométrico) para cada pixel de uma banda raster junto com o valor, as coordenadas raster X e Y de cada pixel. O ponto é o centroide da área representada por um pixel.

Return record format: *geom geometry*, *val* double precision, *x* integer, *y* integers.

**Note**

When `exclude_nodata_value = TRUE`, only those pixels whose values are not NODATA are returned as points.

Disponibilidade: 2.1.0

Alterações: 2.1.1 Mudança no comportamento do `exclude_nodata_value`.

## Exemplos

```
--LATERAL syntax requires PostgreSQL 9.3+
SELECT x, y, val, ST_AsText(geom)
      FROM (SELECT dp.* FROM dummy_rast, LATERAL ST_PixelAsCentroids(rast, 1) AS dp WHERE ←
            rid = 2) foo;
```

x	y	val	st_astext
1	1	253	POINT(3427927.775 5793243.975)
2	1	254	POINT(3427927.825 5793243.975)
3	1	253	POINT(3427927.875 5793243.975)
4	1	254	POINT(3427927.925 5793243.975)
5	1	254	POINT(3427927.975 5793243.975)
1	2	253	POINT(3427927.775 5793243.925)
2	2	254	POINT(3427927.825 5793243.925)
3	2	254	POINT(3427927.875 5793243.925)
4	2	253	POINT(3427927.925 5793243.925)
5	2	249	POINT(3427927.975 5793243.925)
1	3	250	POINT(3427927.775 5793243.875)
2	3	254	POINT(3427927.825 5793243.875)
3	3	254	POINT(3427927.875 5793243.875)
4	3	252	POINT(3427927.925 5793243.875)
5	3	249	POINT(3427927.975 5793243.875)
1	4	251	POINT(3427927.775 5793243.825)
2	4	253	POINT(3427927.825 5793243.825)
3	4	254	POINT(3427927.875 5793243.825)
4	4	254	POINT(3427927.925 5793243.825)
5	4	253	POINT(3427927.975 5793243.825)
1	5	252	POINT(3427927.775 5793243.775)
2	5	250	POINT(3427927.825 5793243.775)
3	5	254	POINT(3427927.875 5793243.775)
4	5	254	POINT(3427927.925 5793243.775)
5	5	254	POINT(3427927.975 5793243.775)

## Veja também

[ST\\_DumpAsPolygons](#), [ST\\_PixelAsPolygon](#), [ST\\_PixelAsPolygons](#), [ST\\_PixelAsPoint](#), [ST\\_PixelAsPoints](#), [ST\\_PixelAsCentroid](#)

## 9.6.7 ST\_Value

**ST\_Value** — Retorna o valor da banda dada com a coluna *x*, linha *y* pixel ou em um ponto específico. Os números de banda começam em 1 e assumem-se 1 se não especificados. Se `exclude_nodata_value` for falso, então todos os pixels, inclusive os `nodata`, são considerados para intersectar e retornar valor. Se `exclude_nodata_value` não passar então lê dos metadados do raster.

### Synopsis

```
double precision ST_Value(raster rast, geometry pt, boolean exclude_nodata_value=true);
double precision ST_Value(raster rast, integer band, geometry pt, boolean exclude_nodata_value=true);
double precision ST_Value(raster rast, integer x, integer y, boolean exclude_nodata_value=true);
double precision ST_Value(raster rast, integer band, integer x, integer y, boolean exclude_nodata_value=true);
```

### Descrição

Retorna o valor da banda dada com a coluna *x*, linha *y* pixel ou em um ponto específico. Os números de banda começam em 1 e assumem-se 1 se não especificados. Se `exclude_nodata_value` for verdadeiro, então somente pixels não `nodata` são considerados. Se `exclude_nodata_value` for falso, então todos os pixels são considerados.

Melhorias: 2.0.0 o argumento opcional `excluye_nodata_value` foi adicionado.

## Exemplos

```
-- get raster values at particular postgis geometry points
-- the srid of your geometry should be same as for your raster
SELECT rid, ST_Value(rast, foo.pt_geom) As b1pval, ST_Value(rast, 2, foo.pt_geom) As b2pval
FROM dummy_rast CROSS JOIN (SELECT ST_SetSRID(ST_Point(3427927.77, 5793243.76), 0) As ←
    pt_geom) As foo
WHERE rid=2;
```

rid	b1pval	b2pval
2	252	79

```
-- general fictitious example using a real table
SELECT rid, ST_Value(rast, 3, sometable.geom) As b3pval
FROM sometable
WHERE ST_Intersects(rast, sometable.geom);
```

```
SELECT rid, ST_Value(rast, 1, 1, 1) As b1pval,
    ST_Value(rast, 2, 1, 1) As b2pval, ST_Value(rast, 3, 1, 1) As b3pval
FROM dummy_rast
WHERE rid=2;
```

rid	b1pval	b2pval	b3pval
2	253	78	70

```
--- Get all values in bands 1,2,3 of each pixel ---
SELECT x, y, ST_Value(rast, 1, x, y) As b1val,
    ST_Value(rast, 2, x, y) As b2val, ST_Value(rast, 3, x, y) As b3val
FROM dummy_rast CROSS JOIN
generate_series(1, 1000) As x CROSS JOIN generate_series(1, 1000) As y
WHERE rid = 2 AND x <= ST_Width(rast) AND y <= ST_Height(rast);
```

x	y	b1val	b2val	b3val
1	1	253	78	70
1	2	253	96	80
1	3	250	99	90
1	4	251	89	77
1	5	252	79	62
2	1	254	98	86
2	2	254	118	108
:				
:				

```
--- Get all values in bands 1,2,3 of each pixel same as above but returning the upper left ←
    point point of each pixel ---
SELECT ST_AsText(ST_SetSRID(
    ST_Point(ST_UpperLeftX(rast) + ST_ScaleX(rast)*x,
        ST_UpperLeftY(rast) + ST_ScaleY(rast)*y),
    ST_SRID(rast))) As uplpt
    , ST_Value(rast, 1, x, y) As b1val,
    ST_Value(rast, 2, x, y) As b2val, ST_Value(rast, 3, x, y) As b3val
FROM dummy_rast CROSS JOIN
generate_series(1,1000) As x CROSS JOIN generate_series(1,1000) As y
WHERE rid = 2 AND x <= ST_Width(rast) AND y <= ST_Height(rast);
```

uplpt	b1val	b2val	b3val
POINT(3427929.25 5793245.5)	253	78	70
POINT(3427929.25 5793247)	253	96	80
POINT(3427929.25 5793248.5)	250	99	90

```

--- Get a polygon formed by union of all pixels
    that fall in a particular value range and intersect particular polygon --
SELECT ST_AsText(ST_Union(pixpolyg)) As shadow
FROM (SELECT ST_Translate(ST_MakeEnvelope(
    ST_UpperLeftX(rast), ST_UpperLeftY(rast),
    ST_UpperLeftX(rast) + ST_ScaleX(rast),
    ST_UpperLeftY(rast) + ST_ScaleY(rast), 0
    ), ST_ScaleX(rast)*x, ST_ScaleY(rast)*y
    ) As pixpolyg, ST_Value(rast, 2, x, y) As b2val
    FROM dummy_rast CROSS JOIN
generate_series(1,1000) As x CROSS JOIN generate_series(1,1000) As y
WHERE rid = 2
    AND x <= ST_Width(rast) AND y <= ST_Height(rast)) As foo
WHERE
    ST_Intersects(
        pixpolyg,
        ST_GeomFromText('POLYGON((3427928 5793244,3427927.75 5793243.75,3427928
        5793243.75,3427928 5793244))',0)
    ) AND b2val != 254;

-----
MULTIPOLYGON(((3427928 5793243.9,3427928 5793243.85,3427927.95 5793243.85,3427927.95
5793243.9,
3427927.95 5793243.95,3427928 5793243.95,3427928.05 5793243.95,3427928.05
5793243.9,3427928 5793243.9)),((3427927.95 5793243.9,3427927.95 579324
3.85,3427927.9 5793243.85,3427927.85 5793243.85,3427927.85 5793243.9,3427927.9
5793243.9,3427927.9 5793243.95,
3427927.95 5793243.95,3427927.95 5793243.9)),((3427927.85 5793243.75,3427927.85
5793243.7,3427927.8 5793243.7,3427927.8 5793243.75
,3427927.8 5793243.8,3427927.8 5793243.85,3427927.85 5793243.85,3427927.85
5793243.8,3427927.85 5793243.75)),
((3427928.05 5793243.75,3427928.05 5793243.7,3427928 5793243.7,3427927.95
5793243.7,3427927.95 5793243.75,3427927.95 5793243.8,3427
927.95 5793243.85,3427928 5793243.85,3427928 5793243.8,3427928.05 5793243.8,
3427928.05 5793243.75)),((3427927.95 5793243.75,3427927.95 5793243.7,3427927.9
5793243.7,3427927.85 5793243.7,
3427927.85 5793243.75,3427927.85 5793243.8,3427927.85 5793243.85,3427927.9 5793243.85,
3427927.95 5793243.85,3427927.95 5793243.8,3427927.95 5793243.75)))

```

```

--- Checking all the pixels of a large raster tile can take a long time.
--- You can dramatically improve speed at some lose of precision by orders of magnitude
-- by sampling pixels using the step optional parameter of generate_series.
-- This next example does the same as previous but by checking 1 for every 4 (2x2) pixels
-- and putting in the last checked
-- putting in the checked pixel as the value for subsequent 4

```

```

SELECT ST_AsText(ST_Union(pixpolyg)) As shadow
FROM (SELECT ST_Translate(ST_MakeEnvelope(
    ST_UpperLeftX(rast), ST_UpperLeftY(rast),
    ST_UpperLeftX(rast) + ST_ScaleX(rast)*2,
    ST_UpperLeftY(rast) + ST_ScaleY(rast)*2, 0

```



```

        ), ST_ScaleX(rast)*x, ST_ScaleY(rast)*y
        ) As pixpolyg, ST_Value(rast, 2, x, y) As b2val
    FROM dummy_rast CROSS JOIN
generate_series(1,1000,2) As x CROSS JOIN generate_series(1,1000,2) As y
WHERE rid = 2
    AND x <= ST_Width(rast) AND y <= ST_Height(rast) ) As foo
WHERE
    ST_Intersects(
        pixpolyg,
        ST_GeomFromText('POLYGON((3427928 5793244,3427927.75 5793243.75,3427928
        5793243.75,3427928 5793244))',0)
    ) AND b2val != 254;

-----
shadow
MULTIPOLYGON(((3427927.9 5793243.85,3427927.8 5793243.85,3427927.8 5793243.95,
3427927.9 5793243.95,3427928 5793243.95,3427928.1 5793243.95,3427928.1 5793243.85,3427928
5793243.85,3427927.9 5793243.85)),
((3427927.9 5793243.65,3427927.8 5793243.65,3427927.8 5793243.75,3427927.8
5793243.85,3427927.9 5793243.85,
3427928 5793243.85,3427928 5793243.75,3427928.1 5793243.75,3427928.1 5793243.65,3427928
5793243.65,3427927.9 5793243.65)))

```

## Veja também

[ST\\_SetValue](#), [ST\\_DumpAsPolygons](#), [ST\\_NumBands](#), [ST\\_PixelAsPolygon](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#), [ST\\_UpperLeftX](#), [ST\\_UpperLeftY](#), [ST\\_SRID](#), [ST\\_AsText](#), [ST\\_Point](#), [ST\\_MakeEnvelope](#), [ST\\_Intersects](#), [ST\\_Intersection](#)

## 9.6.8 ST\_NearestValue

**ST\_NearestValue** — Retorna o valor não-NODATA mais próximo de um dado pixel de banda especificado por uma coluna x e linha ou um ponto geométrico expressado no mesmo sistema de coordenada referência do raster.

### Synopsis

```

double precision ST_NearestValue(raster rast, integer bandnum, geometry pt, boolean exclude_nodata_value=true);
double precision ST_NearestValue(raster rast, geometry pt, boolean exclude_nodata_value=true);
double precision ST_NearestValue(raster rast, integer bandnum, integer columnx, integer rowy, boolean exclude_nodata_value=true);
double precision ST_NearestValue(raster rast, integer columnx, integer rowy, boolean exclude_nodata_value=true);

```

### Descrição

Returns the nearest non-NODATA value of a given band in a given columnx, rowy pixel or at a specific geometric point. If the columnx, rowy pixel or the pixel at the specified geometric point is NODATA, the function will find the nearest pixel to the columnx, rowy pixel or geometric point whose value is not NODATA.

O número de banda começa no 1 e `bandnum` é assumido a ser 1 se não estiver especificado. Se `exclude_nodata_value` for falso, então todos os pixels inclusive os pixels nodata são considerados para intersectar e retornar valor. Se `exclude_nodata_value` não passar então lê dos metadados do raster.

Disponibilidade: 2.1.0



#### Note

`ST_NearestValue` é uma substituição drop-in para `ST_Value`.

**Exemplos**

```
-- pixel 2x2 has value
SELECT
    ST_Value(rast, 2, 2) AS value,
    ST_NearestValue(rast, 2, 2) AS nearestvalue
FROM (
    SELECT
        ST_SetValue(
            ST_SetValue(
                ST_SetValue(
                    ST_SetValue(
                        ST_SetValue(
                            ST_AddBand(
                                ST_MakeEmptyRaster(5, 5, ←
                                    -2, 2, 1, -1, 0, 0, 0),
                                '8BUI'::text, 1, 0
                            ),
                            1, 1, 0.
                        ),
                        2, 3, 0.
                    ),
                    3, 5, 0.
                ),
                4, 2, 0.
            ),
            5, 4, 0.
        ) AS rast
    ) AS foo

value | nearestvalue
-----+-----
1 | 1
```

```
-- pixel 2x3 is NODATA
SELECT
    ST_Value(rast, 2, 3) AS value,
    ST_NearestValue(rast, 2, 3) AS nearestvalue
FROM (
    SELECT
        ST_SetValue(
            ST_SetValue(
                ST_SetValue(
                    ST_SetValue(
                        ST_SetValue(
                            ST_AddBand(
                                ST_MakeEmptyRaster(5, 5, ←
                                    -2, 2, 1, -1, 0, 0, 0),
                                '8BUI'::text, 1, 0
                            ),
                            1, 1, 0.
                        ),
                        2, 3, 0.
                    ),
                    3, 5, 0.
                ),
                4, 2, 0.
            ),
            5, 4, 0.
        ) AS rast
    ) AS foo
```

value	nearestvalue
	1

## Veja também

[ST\\_Neighborhood](#), [ST\\_Value](#)

### 9.6.9 ST\_Neighborhood

**ST\_Neighborhood** — Retorna um arranjo de precisão 2-D dobrada dos valores não-NODATA em torno da banda de pixel especificada ou por uma colunaX e linhaY ou um ponto geométrico expressado no mesmo sistema de coordenada de referência especial como o raster.

#### Synopsis

```
double precision[][] ST_Neighborhood(raster rast, integer bandnum, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
```

```
double precision[][] ST_Neighborhood(raster rast, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
```

```
double precision[][] ST_Neighborhood(raster rast, integer bandnum, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
```

```
double precision[][] ST_Neighborhood(raster rast, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);
```

#### Descrição

Retorna um arranjo de precisão 2-D dobrada dos valores não-NODATA em torno da banda de pixel especificada ou por uma colunaX e linhaY ou um ponto geométrico expressado no mesmo sistema de coordenada de referência especial como o raster. Os parâmetros `distanceX` e `distanceY` definem o número de pixels em torno do pixel especificado nos eixos X e Y, ex.: Quero todos os valores dentro de uma distância de 3 pixels no eixo X e 2 pixels de distância no eixo Y ao redor do meu pixel de interesse. O valor central do arranjo 2-D será o valor no pixel especificado pela colunaX e linhaY ou ponto geométrico.

O número de banda começa no 1 e `bandnum` é assumido a ser 1 se não estiver especificado. Se `exclude_nodata_value` for falso, então todos os pixels inclusive os pixels `nodata` são considerados para intersectar e retornar valor. Se `exclude_nodata_value` não passar então lê dos metadados do raster.



#### Note

O número de elementos ao longo de cada eixo do arranjo que está retornando 2-D é  $2 * (\text{distanceX}|\text{distanceY}) + 1$ . Então, para uma `distanceX` e `distanceY` de 1, o arranjo que retorna será 3x3.



#### Note

A saída do arranjo 2-D pode ser passado para qualquer um dos processos raster de funções built-in, ex.: `ST_Min4ma`, `ST_Sum4ma`, `ST_Mean4ma`.

Disponibilidade: 2.1.0

**Exemplos**

```
-- pixel 2x2 has value
SELECT
  ST_Neighborhood(rast, 2, 2, 1, 1)
FROM (
  SELECT
    ST_SetValues(
      ST_AddBand(
        ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
        '8BUI'::text, 1, 0
      ),
      1, 1, 1, ARRAY[
        [0, 1, 1, 1, 1],
        [1, 1, 1, 0, 1],
        [1, 0, 1, 1, 1],
        [1, 1, 1, 1, 0],
        [1, 1, 0, 1, 1]
      ]::double precision[],
      1
    ) AS rast
) AS foo

      st_neighborhood
-----
{{NULL,1,1},{1,1,NULL},{1,1,1}}
```

```
-- pixel 2x3 is NODATA
SELECT
  ST_Neighborhood(rast, 2, 3, 1, 1)
FROM (
  SELECT
    ST_SetValues(
      ST_AddBand(
        ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
        '8BUI'::text, 1, 0
      ),
      1, 1, 1, ARRAY[
        [0, 1, 1, 1, 1],
        [1, 1, 1, 0, 1],
        [1, 0, 1, 1, 1],
        [1, 1, 1, 1, 0],
        [1, 1, 0, 1, 1]
      ]::double precision[],
      1
    ) AS rast
) AS foo

      st_neighborhood
-----
{{1,1,1},{1,NULL,1},{1,1,1}}
```

```
-- pixel 3x3 has value
-- exclude_nodata_value = FALSE
SELECT
  ST_Neighborhood(rast, 3, 3, 1, 1, false)
FROM (
  SELECT
    ST_SetValues(
      ST_AddBand(
        ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
        '8BUI'::text, 1, 0
      )
    ) AS rast
) AS foo
```

```

        ),
        1, 1, 1, ARRAY[
            [0, 1, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 1, 1, 0],
            [1, 1, 0, 1, 1]
        ]::double precision[],
        1
    ) AS rast
) AS foo

-----
st_neighborhood
-----
{{1,0,1},{1,1,1},{0,1,1}}
```

### Veja também

[ST\\_NearestValue](#), [ST\\_Min4ma](#), [ST\\_Max4ma](#), [ST\\_Sum4ma](#), [ST\\_Mean4ma](#), [ST\\_Range4ma](#), [ST\\_Distinct4ma](#), [ST\\_StdDev4ma](#)

## 9.6.10 ST\_SetValue

**ST\_SetValue** — Retorna o raster modificado resultante do valor de uma banda em uma dada colunax, linha y pixel ou os pixels que intersectam uma geometria específica. Os números de banda começam no 1 e são assumidos como 1 se não estiverem especificados.

### 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);
```

### Descrição

Returns modified raster resulting from setting the specified pixels' values to new value for the designated band given the raster's row and column or a geometry. If no band is specified, then band 1 is assumed.

Melhorias: 2.1.0 Variante geométrica **ST\_SetValue()** agora suporta qualquer tipo de geometria, não apenas ponto. A variante geométrica é um envoltório em torno da variante **geomval[]** da **ST\_SetValues()**

### Exemplos

```

-- Geometry example
SELECT (foo.geomval).val, ST_AsText(ST_Union((foo.geomval).geom))
FROM (SELECT ST_DumpAsPolygons(
        ST_SetValue(rast,1,
                    ST_Point(3427927.75, 5793243.95),
                    50)
        ) As geomval
FROM dummy_rast
where rid = 2) As foo
WHERE (foo.geomval).val < 250
GROUP BY (foo.geomval).val;
```

```

val | st_astext
-----+-----
 50 | POLYGON((3427927.75 5793244,3427927.75 5793243.95,3427927.8 579324 ...
249 | POLYGON((3427927.95 5793243.95,3427927.95 5793243.85,3427928 57932 ...

-- Store the changed raster --
UPDATE dummy_rast SET rast = ST_SetValue(rast,1, ST_Point(3427927.75, 5793243.95) ↔
,100)
WHERE rid = 2 ;

```

## Veja também

[ST\\_Value](#), [ST\\_DumpAsPolygons](#)

### 9.6.11 ST\_SetValues

**ST\_SetValues** — Retorna o raster modificado resultante dos valores de uma dada banda.

#### 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);

```

#### Descrição

Retorna o raster modificado resultante dos valores especificados do pixel para novo valor(es) para a banda designada.

Se `keepnodata` for VERDADE, aqueles pixels cujos valores são NODATA não terão o valor correspondente em `newvalueset`.

Para Variante 1, os pixels específicos são determinados pela `columnx`, `rowy` coordenadas pixel e as dimensões do arranjo `newvalueset`. `noset` pode ser usado para prevenir pixels com valores presentes no `newvalueset` de serem estabelecidos ( PostgreSQL não permitindo arranjos `ragged/jagged`). Veja o exemplo de Variante 1.

Variante 2 é como a Variante 1, mas com uma precisão dupla simples `nosetvalue` em vez de um arranjo booleano `noset`. Elementos no `newvalueset` com o valor `nosetvalue` são pulados. Veja o exemplo da Variante 2.

Para Variante 3, os pixels a serem estabelecidos são determinados pelas `columnx`, `rowy` coordenadas pixel, `width` e `height`. Veja o exemplo da Variante 3.

A Variante 4 é a mesma que a Variante 3, com a exceção de que ela assume que a primeira banda do pixel de `rast` será estabelecida.

Para a Variante 5, um arranjo de `geomval` é usado para determinar os pixels específicos. Se todas as geometrias no arranjo forem do tipo PONTO ou MULTIPONTO, a função usa um atalho onde a longitude e latitude de cada ponto é usada para pôr um pixel diretamente. Caso contrário, as geometrias são convertidas para rasters e então iteradas através de um passo. Veja o exemplo de Variante 5.

Disponibilidade: 2.1.0

**Exemplos: Variante 1**

```

/*
The ST_SetValues() does the following...

+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 1 | 1 | 1 |
+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |   =
>   | 1 | 9 | 9 |
+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 1 | 9 | 9 |
+ - + - + - +           + - + - + - +
*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_AddBand(
                ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                1, '8BUI', 1, 0
            ),
            1, 2, 2, ARRAY[[9, 9], [9, 9]]::double precision[][]
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 x | y | val
---+---+---
 1 | 1 |   1
 1 | 2 |   1
 1 | 3 |   1
 2 | 1 |   1
 2 | 2 |   9
 2 | 3 |   9
 3 | 1 |   1
 3 | 2 |   9
 3 | 3 |   9

```

```

/*
The ST_SetValues() does the following...

+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 9 | 9 | 9 |
+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |   =
>   | 9 |   | 9 |
+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 9 | 9 | 9 |
+ - + - + - +           + - + - + - +
*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (

```

```

SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_AddBand(
                ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double ←
                precision[][]
        )
    ) AS poly
) foo
ORDER BY 1, 2;

```

x	y	val
1	1	9
1	2	9
1	3	9
2	1	9
2	2	
2	3	9
3	1	9
3	2	9
3	3	9

```

/*
The ST_SetValues() does the following...

```

+ - + - + - +		+ - + - + - +
1   1   1		9   9   9
+ - + - + - +		+ - + - + - +
1   1   1	=	
>   1		9
+ - + - + - +		+ - + - + - +
1   1   1		9   9   9
+ - + - + - +		+ - + - + - +

```

*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_AddBand(
                ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1,
            ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision ←
                [],
            ARRAY[[false], [true]]::boolean[][]
        )
    ) AS poly
) foo
ORDER BY 1, 2;

```

x	y	val
1	1	9



```

1 | 2 | 1
1 | 3 | 9
2 | 1 | 9
2 | 2 |
2 | 3 | 9
3 | 1 | 9
3 | 2 | 9
3 | 3 | 9

```

```

/*
The ST_SetValues() does the following...

+ - + - + - +      + - + - + - +
|   | 1 | 1 |      |   | 9 | 9 |
+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      =
>   | 1 |   | 9 |
+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      | 9 | 9 | 9 |
+ - + - + - +      + - + - + - +
*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_SetValue(
                ST_AddBand(
                    ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                    1, '8BUI', 1, 0
                ),
                1, 1, 1, NULL
            ),
            1, 1, 1,
            ARRAY[[9, 9, 9], [9, NULL, 9], [9, 9, 9]]::double precision ←
                [],
            ARRAY[[false], [true]]::boolean[[]],
            TRUE
        )
    ) AS poly
) foo
ORDER BY 1, 2;

x | y | val
---+---+-----
1 | 1 |
1 | 2 | 1
1 | 3 | 9
2 | 1 | 9
2 | 2 |
2 | 3 | 9
3 | 1 | 9
3 | 2 | 9
3 | 3 | 9

```

### Exemplos: Variant 2

```

/*
The ST_SetValues() does the following...

+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      | 1 | 1 | 1 |
+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      =
>   | 1 | 9 | 9 |
+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      | 1 | 9 | 9 |
+ - + - + - +      + - + - + - +
*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_AddBand(
                ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[-1, -1, -1], [-1, 9, 9], [-1, 9, 9]]::double ←
                precision[[]], -1
        )
    ) AS poly
) foo
ORDER BY 1, 2;

 x | y | val
---+---+---
 1 | 1 |  1
 1 | 2 |  1
 1 | 3 |  1
 2 | 1 |  1
 2 | 2 |  9
 2 | 3 |  9
 3 | 1 |  1
 3 | 2 |  9
 3 | 3 |  9

```

```

/*
This example is like the previous one. Instead of nosetvalue = -1, nosetvalue = NULL

The ST_SetValues() does the following...

+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      | 1 | 1 | 1 |
+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      =
>   | 1 | 9 | 9 |
+ - + - + - +      + - + - + - +
| 1 | 1 | 1 |      | 1 | 9 | 9 |
+ - + - + - +      + - + - + - +
*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val

```

```

FROM (
SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_AddBand(
                ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                1, '8BUI', 1, 0
            ),
            1, 1, 1, ARRAY[[NULL, NULL, NULL], [NULL, 9, 9], [NULL, 9, 9]]:: ←
                double precision[][], NULL::double precision
        )
    ) AS poly
) foo
ORDER BY 1, 2;

```

x	y	val
1	1	1
1	2	1
1	3	1
2	1	1
2	2	9
2	3	9
3	1	1
3	2	9
3	3	9

### Exemplos: Variante 3

```

/*
The ST_SetValues() does the following...

+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 1 | 1 | 1 |
+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           =
> | 1 | 9 | 9 |
+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 1 | 9 | 9 |
+ - + - + - +           + - + - + - +
*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_AddBand(
                ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                1, '8BUI', 1, 0
            ),
            1, 2, 2, 2, 2, 2, 9
        )
    ) AS poly
) foo
ORDER BY 1, 2;

x | y | val

```

```

-----+-----+-----
1 | 1 | 1
1 | 2 | 1
1 | 3 | 1
2 | 1 | 1
2 | 2 | 9
2 | 3 | 9
3 | 1 | 1
3 | 2 | 9
3 | 3 | 9

```

```

/*
The ST_SetValues() does the following...

+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 1 | 1 | 1 |
+ - + - + - +           + - + - + - +
| 1 |   | 1 |   =
>   | 1 |   | 9 |
+ - + - + - +           + - + - + - +
| 1 | 1 | 1 |           | 1 | 9 | 9 |
+ - + - + - +           + - + - + - +
*/
SELECT
    (poly).x,
    (poly).y,
    (poly).val
FROM (
SELECT
    ST_PixelAsPolygons(
        ST_SetValues(
            ST_SetValue(
                ST_AddBand(
                    ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0),
                    1, '8BUI', 1, 0
                ),
                1, 2, 2, NULL
            ),
            1, 2, 2, 2, 2, 9, TRUE
        )
    ) AS poly
) foo
ORDER BY 1, 2;

x | y | val
-----+-----+-----
1 | 1 | 1
1 | 2 | 1
1 | 3 | 1
2 | 1 | 1
2 | 2 |
2 | 3 | 9
3 | 1 | 1
3 | 2 | 9
3 | 3 | 9

```

### Exemplos: Variante 5

```
WITH foo AS (
```

```

SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8 ←
BUI', 0, 0) AS rast
), bar AS (
SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)::geometry geom UNION ALL
SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1, 4 -1, 4 -4, 1 -4, 1 -1))::geometry geom ←
UNION ALL
SELECT 3 AS gid, 'SRID=0;POLYGON((0 0, 5 0, 5 -1, 1 -1, 1 -4, 0 -4, 0 0)):: ←
geometry geom UNION ALL
SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0, 4 4, 4 -4)::geometry
)
SELECT
    rid, gid, ST_DumpValues(ST_SetValue(rast, 1, geom, gid))
FROM foo t1
CROSS JOIN bar t2
ORDER BY rid, gid;

```

rid	gid	st_dumpvalues
1	1	(1, "{NULL, NULL, NULL, NULL, NULL}, {NULL, NULL, NULL, NULL, NULL}, {NULL, NULL, 1, NULL, NULL}, {NULL, NULL, NULL, NULL, NULL}, {NULL, NULL, NULL, NULL, NULL}}")
1	2	(1, "{NULL, NULL, NULL, NULL, NULL}, {NULL, 2, 2, 2, NULL}, {NULL, 2, 2, 2, NULL}, {NULL, 2, 2, 2, NULL}, {NULL, NULL, NULL, NULL, NULL}}")
1	3	(1, "{3, 3, 3, 3, 3}, {3, NULL, NULL, NULL, NULL}, {3, NULL, NULL, NULL, NULL}, {3, NULL, NULL, NULL, NULL}, {NULL, NULL, NULL, NULL, NULL}}")
1	4	(1, "{4, NULL, NULL, NULL, NULL}, {NULL, NULL, NULL, NULL, NULL}, {NULL, NULL, NULL, NULL, NULL}, {NULL, NULL, NULL, NULL, 4}}")

(4 rows)

A seguir está demonstrado que geomvals podem, mais tarde, sobrescrever no arranjo geomvals anteriores

```

WITH foo AS (
SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8 ←
BUI', 0, 0) AS rast
), bar AS (
SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)::geometry geom UNION ALL
SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1, 4 -1, 4 -4, 1 -4, 1 -1))::geometry geom ←
UNION ALL
SELECT 3 AS gid, 'SRID=0;POLYGON((0 0, 5 0, 5 -1, 1 -1, 1 -4, 0 -4, 0 0)):: ←
geometry geom UNION ALL
SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0, 4 4, 4 -4)::geometry
)
SELECT
    t1.rid, t2.gid, t3.gid, ST_DumpValues(ST_SetValues(rast, 1, ARRAY[ROW(t2.geom, t2. ←
gid), ROW(t3.geom, t3.gid)]::geomval[]))
FROM foo t1
CROSS JOIN bar t2
CROSS JOIN bar t3
WHERE t2.gid = 1
AND t3.gid = 2
ORDER BY t1.rid, t2.gid, t3.gid;

```

rid	gid	gid	st_dumpvalues
1	1	2	(1, "{NULL, NULL, NULL, NULL, NULL}, {NULL, 2, 2, 2, NULL}, {NULL, 2, 2, 2, NULL}, {NULL, 2, 2, 2, NULL}, {NULL, NULL, NULL, NULL, NULL}}")

(1 row)

Este exemplo é o oposto do exemplo anterior

```

WITH foo AS (

```

```

SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '8 ←
BUI', 0, 0) AS rast
), bar AS (
SELECT 1 AS gid, 'SRID=0;POINT(2.5 -2.5)::geometry geom UNION ALL
SELECT 2 AS gid, 'SRID=0;POLYGON((1 -1, 4 -1, 4 -4, 1 -4, 1 -1))::geometry geom ←
UNION ALL
SELECT 3 AS gid, 'SRID=0;POLYGON((0 0, 5 0, 5 -1, 1 -1, 1 -4, 0 -4, 0 0)):: ←
geometry geom UNION ALL
SELECT 4 AS gid, 'SRID=0;MULTIPOINT(0 0, 4 4, 4 -4)::geometry
)
SELECT
t1.rid, t2.gid, t3.gid, ST_DumpValues(ST_SetValues(rast, 1, ARRAY[ROW(t2.geom, t2. ←
gid), ROW(t3.geom, t3.gid)]::geomval[]))
FROM foo t1
CROSS JOIN bar t2
CROSS JOIN bar t3
WHERE t2.gid = 2
AND t3.gid = 1
ORDER BY t1.rid, t2.gid, t3.gid;

```

```

rid | gid | gid | st_dumpvalues
-----+-----+-----+-----
1 | 2 | 1 | (1, "{NULL,NULL,NULL,NULL,NULL},{NULL,2,2,2,NULL},{NULL,2,1,2,NULL},{ ←
NULL,2,2,2,NULL},{NULL,NULL,NULL,NULL,NULL}")
(1 row)

```

## Veja também

[ST\\_Value](#), [ST\\_SetValue](#), [ST\\_PixelAsPolygons](#)

## 9.6.12 ST\_DumpValues

**ST\_DumpValues** — Obtenha os valores da banda específica como um arranjo 2-dimensional.

### 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 );

```

### Descrição

Obtenha os valores da banda especificada como um arranjo 2-dimensional (o primeiro índice é linha, o segundo é coluna). Se `nband` for NULO ou não for fornecido, todas as bandas raster serão processadas.

Disponibilidade: 2.1.0

### Exemplos

```

WITH foo AS (
SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, ←
0), 1, '8BUI'::text, 1, 0), 2, '32BF'::text, 3, -9999), 3, '16BSI', 0, 0) AS ←
rast
)
SELECT
(ST_DumpValues(rast)).*

```

```
FROM foo;
```

nband	valarray
1	{{1,1,1},{1,1,1},{1,1,1}}
2	{{3,3,3},{3,3,3},{3,3,3}}
3	{{NULL,NULL,NULL},{NULL,NULL,NULL},{NULL,NULL,NULL}}

(3 rows)

```
WITH foo AS (
  SELECT ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI'::text, 1, 0), 2, '32BF'::text, 3, -9999), 3, '16BSI', 0, 0) AS rast
)
SELECT
  (ST_DumpValues(rast, ARRAY[3, 1])).*
FROM foo;
```

nband	valarray
3	{{NULL,NULL,NULL},{NULL,NULL,NULL},{NULL,NULL,NULL}}
1	{{1,1,1},{1,1,1},{1,1,1}}

(2 rows)

```
WITH foo AS (
  SELECT ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8BUI', 1, 0), 1, 2, 5) AS rast
)
SELECT
  (ST_DumpValues(rast, 1))[2][1]
FROM foo;
```

st_dumpvalues
5

(1 row)

## Veja também

[ST\\_Value](#), [ST\\_SetValue](#), [ST\\_SetValues](#)

### 9.6.13 ST\_PixelOfValue

`ST_PixelOfValue` — Obtenha as coordenadas colunax, linhay do pixel cujos valores são iguais ao valor de pesquisa.

#### 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 );
```

#### Descrição

Obtenha as coordenadas colunax, linhay do pixel cujos valores são iguais ao valor de pesquisa. Se nenhuma banda for especificada, então a banda 1 é assumida.

Disponibilidade: 2.1.0

**Exemplos**

```

SELECT
  (pixels).*
FROM (
  SELECT
    ST_PixelOfValue(
      ST_SetValue(
        ST_SetValue(
          ST_SetValue(
            ST_SetValue(
              ST_SetValue(
                ST_AddBand(
                  ST_MakeEmptyRaster ←
                    (5, 5, -2, 2, 1, ←
                      -1, 0, 0, 0),
                  '8BUI'::text, 1, 0
                ),
              1, 1, 0
            ),
          2, 3, 0
        ),
      3, 5, 0
    ),
    4, 2, 0
  ),
  5, 4, 255
) AS foo
, 1, ARRAY[1, 255]) AS pixels

```

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



## 9.7 Editores Raster

### 9.7.1 ST\_SetGeoReference

`ST_SetGeoReference` — Coloque os parâmetros Georeference 6 em uma única chamada. Os números deverão ser separados por espaço branco. Aceita entrar no formato GDAL ou ESRI. O padrão é 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);

#### Descrição

Coloca os parâmetros georreferência 6 em uma única chamada. Aceita entrar no formato GDAL ou ESRI. O padrão é GDAL. Se 6 coordenadas não forem fornecidas, retornará null.

A diferença entre representações de formatos é a seguinte:

GDAL:

```
scalex skewy skewx scaley upperleftx upperlefty
```

ESRI:

```
scalex skewy skewx scaley upperleftx + scalex*0.5 upperlefty + scaley*0.5
```



#### Note

Se o raster tiver bandas fora do banco de dados, alterar a georreferência pode resultar em acesso incorreto dos dados de banda armazenados externamente.

Melhorias: 2.1.0 Adição da variante `ST_SetGeoReference(raster, double precision, ...)`

#### Exemplos

```
WITH foo AS (
  SELECT ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0) AS rast
)
SELECT
  0 AS rid, (ST_Metadata(rast)).*
FROM foo
UNION ALL
SELECT
  1, (ST_Metadata(ST_SetGeoReference(rast, '10 0 0 -10 0.1 0.1', 'GDAL'))).*
FROM foo
UNION ALL
SELECT
  2, (ST_Metadata(ST_SetGeoReference(rast, '10 0 0 -10 5.1 -4.9', 'ESRI'))).*
FROM foo
UNION ALL
SELECT
  3, (ST_Metadata(ST_SetGeoReference(rast, 1, 1, 10, -10, 0.001, 0.001))).*
FROM foo
```

rid	upperleftx skewy	srid	numbands	upperlefty	width	height	scalex	scaley	skewx	↔
0	0	0	0	0	5	5	1	-1	0	↔
1	0	0	0.1	0	5	5	10	-10	0	↔
2	0.09999999999999996	0	0	0.09999999999999996	5	5	10	-10	0	↔
3	0.001	0	1	0	5	5	10	-10	0.001	↔

**Veja também**

[ST\\_GeoReference](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#), [ST\\_UpperLeftX](#), [ST\\_UpperLeftY](#)

**9.7.2 ST\_SetRotation**

ST\_SetRotation — Põe a rotação do raster em radianos.

**Synopsis**

float8 ST\_SetRotation(raster rast, float8 rotation);

**Descrição**

Gira o raster uniformemente. A rotação é em radianos. Recorra a [World File](#) para mais detalhes.

**Exemplos**

```
SELECT
  ST_ScaleX(rast1), ST_ScaleY(rast1), ST_SkewX(rast1), ST_SkewY(rast1),
  ST_ScaleX(rast2), ST_ScaleY(rast2), ST_SkewX(rast2), ST_SkewY(rast2)
FROM (
  SELECT ST_SetRotation(rast, 15) AS rast1, rast as rast2 FROM dummy_rast
) AS foo;
```

st_scalex	st_scaley	st_skewx	st_skewy	↔
-1.51937582571764	-2.27906373857646	1.95086352047135	1.30057568031423	↔
2	3	0	0	
-0.0379843956429411	-0.0379843956429411	0.0325143920078558	0.0325143920078558	↔
0.05	-0.05	0	0	

**Veja também**

[ST\\_Rotation](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#), [ST\\_SkewX](#), [ST\\_SkewY](#)

### 9.7.3 ST\_SetScale

**ST\_SetScale** — Coloca os tamanhos X e Y dos pixels em unidades do sistema referencial de coordenadas. Número unidades/pixel largura/altura.

#### Synopsis

```
raster ST_SetScale(raster rast, float8 xy);
raster ST_SetScale(raster rast, float8 x, float8 y);
```

#### Descrição

Coloca os tamanhos X e Y dos pixels em unidades do sistema referencial de coordenadas. Número unidades/pixel largura/altura. Se apenas uma unidade passar, o X e Y assumido são o mesmo número.



#### Note

**ST\_SetScale** é diferente de **ST\_Rescale** onde a **ST\_SetScale** não resample o raster para combinar com a extensão. Apenas altera os metadados (ou georreferência) do raster, para corrigir uma escala originalmente mal especificada. A **ST\_SetScale** resulta em um raster tendo largura e altura diferentes, calculadas para caber na extensão geográfica do raster de entrada. A **ST\_SetScale** não modifica a largura, nem a altura do raster.

Alterações: 2.0.0 Nas versões WKTRaster era chamado de **ST\_SetPixelSize**. Foi modificado na 2.0.0.

#### Exemplos

```
UPDATE dummy_rast
   SET rast = ST_SetScale(rast, 1.5)
WHERE rid = 2;

SELECT ST_ScaleX(rast) As pixx, ST_ScaleY(rast) As pixy, Box3D(rast) As newbox
FROM dummy_rast
WHERE rid = 2;
```

pixx	pixy	newbox
1.5	1.5	BOX(3427927.75 5793244 0, 3427935.25 5793251.5 0)

```
UPDATE dummy_rast
   SET rast = ST_SetScale(rast, 1.5, 0.55)
WHERE rid = 2;

SELECT ST_ScaleX(rast) As pixx, ST_ScaleY(rast) As pixy, Box3D(rast) As newbox
FROM dummy_rast
WHERE rid = 2;
```

pixx	pixy	newbox
1.5	0.55	BOX(3427927.75 5793244 0,3427935.25 5793247 0)

#### Veja também

[ST\\_ScaleX](#), [ST\\_ScaleY](#), [Caixa3D](#)

## 9.7.4 ST\_SetSkew

`ST_SetSkew` — Coloca as georreferências X e Y distorcidas (ou parâmetro de rotação). Se somente um passar, coloca o X e o Y no mesmo valor.

### Synopsis

```
raster ST_SetSkew(raster rast, float8 skewxy);
raster ST_SetSkew(raster rast, float8 skewx, float8 skewy);
```

### Descrição

Coloca as georreferências X e Y distorcidas (ou parâmetro de rotação). Se somente um passar, coloca o X e o Y no mesmo valor. Recorra a [World File](#) para mais detalhes.

### Exemplos

```
-- Example 1
UPDATE dummy_rast SET rast = ST_SetSkew(rast,1,2) WHERE rid = 1;
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
       ST_GeoReference(rast) as georef
FROM dummy_rast WHERE rid = 1;
```

rid	skewx	skewy	georef
1	1	2	2.0000000000 : 2.0000000000 : 1.0000000000 : 3.0000000000 : 0.5000000000 : 0.5000000000

```
-- Example 2 set both to same number:
UPDATE dummy_rast SET rast = ST_SetSkew(rast,0) WHERE rid = 1;
SELECT rid, ST_SkewX(rast) As skewx, ST_SkewY(rast) As skewy,
       ST_GeoReference(rast) as georef
FROM dummy_rast WHERE rid = 1;
```

rid	skewx	skewy	georef
1	0	0	2.0000000000 : 0.0000000000 : 0.0000000000 : 3.0000000000 : 0.5000000000 : 0.5000000000

### Veja também

[ST\\_GeoReference](#), [ST\\_SetGeoReference](#), [ST\\_SkewX](#), [ST\\_SkewY](#)

## 9.7.5 ST\_SetSRID

`ST_SetSRID` — Coloca o SRID de um raster em um srid inteiro específico definido na tabela `spatial_ref_sys`.

## Synopsis

raster **ST\_SetSRID**(raster rast, integer srid);

## Descrição

Coloca o SRID em um raster para um valor inteiro específico.



### Note

Esta função não transforma o raster em forma alguma - simplesmente coloca metadados definindo a referência espacial do sistema de coordenadas referência que está sendo usado. É útil para futuras transformações.

## Veja também

Section [4.3.1](#), [ST\\_SRID](#)

## 9.7.6 ST\_SetUpperLeft

**ST\_SetUpperLeft** — Sets the value of the upper left corner of the pixel of the raster to projected X and Y coordinates.

## Synopsis

raster **ST\_SetUpperLeft**(raster rast, double precision x, double precision y);

## Descrição

Set the value of the upper left corner of raster to the projected X and Y coordinates

## Exemplos

```
SELECT ST_SetUpperLeft (rast, -71.01, 42.37)
FROM dummy_rast
WHERE rid = 2;
```

## Veja também

[ST\\_UpperLeftX](#), [ST\\_UpperLeftY](#)

## 9.7.7 ST\_Resample

**ST\_Resample** — Resample um raster usando um algoritmo específico, novas dimensões, um canto aleatório da grade e um conjunto de rasters georreferenciando atributos definidos ou emprestados de outro raster.

## Synopsis

raster **ST\_Resample**(raster rast, integer width, integer height, double precision gridx=NULL, double precision gridy=NULL, double precision skewx=0, double precision skewy=0, text algorithm=NearestNeighbour, 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=NearestNeighbour, double precision maxerr=0.125, boolean usescale=true);  
 raster **ST\_Resample**(raster rast, raster ref, boolean usescale, text algorithm=NearestNeighbour, double precision maxerr=0.125);

## Descrição

Resample um raster usando um algoritmo específico, novas dimensões (largura & altura), um canto de grade (gradex & gradey) e um conjunto de rasters georreferenciando atributos (scalex, scaley, skewx & skewy) definidos ou emprestados de outro raster. Se estiver utilizando uma referência raster, os dois rasters devem possuir o mesmo SRID.

Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (English or American spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor por ser mais rápido, porém produz a pior interpolação.

Uma porcentagem maxerror de 0.125 é usada se nenhum maxerr for especificado.



### Note

Recorra a: [GDAL Warp resampling methods](#) para mais detalhes.

Disponibilidade: 2.0.0 Requer GDAL 1.6.1+

Alterações: 2.1.0 O parâmetro srid foi removido. As variantes com um raster referência não aplica mais o SRID da referência raster. Use a ST\_Transform() para reprojeter o raster. Funciona em rasters sem SRID.

## Exemplos

```
SELECT
  ST_Width(orig) AS orig_width,
  ST_Width(reduce_100) AS new_width
FROM (
  SELECT
    rast AS orig,
    ST_Resample(rast,100,100) AS reduce_100
  FROM aerials.boston
  WHERE ST_Intersects(rast,
    ST_Transform(
      ST_MakeEnvelope(-71.128, 42.2392,-71.1277, 42.2397, 4326),26986)
    )
  )
LIMIT 1
) AS foo;
```

orig_width	new_width
200	100

## Veja também

[ST\\_Rescale](#), [ST\\_Resize](#), [ST\\_Transform](#)

## 9.7.8 ST\_Rescale

`ST_Rescale` — Resample um raster ajustando sua única escala (ou tamanho de pixel). Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor.

### Synopsis

```
raster ST_Rescale(raster rast, double precision scalexy, text algorithm=NearestNeighbour, double precision maxerr=0.125);
raster ST_Rescale(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeighbour, double precision maxerr=0.125);
```

### Descrição

Resample um raster ajustando sua única escala (ou tamanho de pixel). Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor pois é o mais rápido, mas resulta na pior interpolação.

`scalex` e `scaley` definem o tamanho do pixel. A `scaley` deve ser, geralmente, negativa para ser um raster bem orientado.

Quando a nova `scalex` ou `scaley` não é divisora da largura ou altura do raster, a extensão do raster resultante é expandido para encerrar a extensão do raster fornecido. Se quiser certificar-se de reter a entrada exata, veja [ST\\_Resize](#)

Uma porcentagem `maxerror` de 0.125 é usada se nenhum `maxerr` for especificado.



#### Note

Recorra a: [GDAL Warp resampling methods](#) para mais detalhes.



#### Note

`ST_Rescale` é diferente de `ST_SetScale` onde a `ST_SetScale` não resample o raster para combinar com a extensão. A `ST_SetScale` apenas altera os metadados (ou georreferência) do raster, para corrigir uma escala originalmente mal especificada. A `ST_Rescale` resulta em um raster tendo largura e altura diferentes, calculadas para caber na extensão geográfica do raster de entrada. A `ST_SetScale` não modifica a largura, nem a altura do raster.

Disponibilidade: 2.0.0 Requer GDAL 1.6.1+

Alterações: 2.1.0 Funciona em rasters sem SRID

### Exemplos

Um exemplo simples de reescalar um raster de um tamanho de pixel de 0.001 grau para um pixel de tamanho 0.0015 grau.

```
-- the original raster pixel size
SELECT ST_PixelWidth(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0, ↵
    4269), '8BUI'::text, 1, 0)) width

width
-----
0.001

-- the rescaled raster raster pixel size
SELECT ST_PixelWidth(ST_Rescale(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, ↵
    -0.001, 0, 0, 4269), '8BUI'::text, 1, 0), 0.0015)) width
```

```
width
-----
0.0015
```

### Veja também

[ST\\_Resize](#), [ST\\_Resample](#), [ST\\_SetScale](#), [ST\\_ScaleX](#), [ST\\_ScaleY](#), [ST\\_Transform](#)

## 9.7.9 ST\_Reskew

`ST_Reskew` — Resample um raster ajustando somente sua inclinação (ou tamanho de pixel). Novos valores de pixel são calculados usando o algoritmo `NearestNeighbor` (english or american spelling), `Bilinear`, `Cubic`, `CubicSpline` ou `Lanczos`. O padrão é `NearestNeighbor`.

### Synopsis

```
raster ST_Reskew(raster rast, double precision skewxy, text algorithm=NearestNeighbour, double precision maxerr=0.125);
raster ST_Reskew(raster rast, double precision skewx, double precision skewy, text algorithm=NearestNeighbour, double precision maxerr=0.125);
```

### Descrição

Resample um raster ajustando somente sua inclinação (ou tamanho de pixel). Novos valores de pixel são calculados usando o algoritmo `NearestNeighbor` (english or american spelling), `Bilinear`, `Cubic`, `CubicSpline` ou `Lanczos`. O padrão é `NearestNeighbor` pois é o mais rápido, mas resulta na pior interpolação.

`skewx` e `skewy` definem a nova distorção.

A extensão do novo raster irá encerrar a extensão do raster fornecido.

Uma porcentagem `maxerror` de 0.125 se nenhum `maxerr` for especificado.



#### Note

Recorra a: [GDAL Warp resampling methods](#) para mais detalhes.



#### Note

`ST_Reskew` é diferente de `ST_SetSkew` onde a `ST_SetSkew` não resample o raster para combinar com a extensão. A `ST_SetScale` apenas altera os metadados (ou georreferência) do raster, para corrigir uma escala originalmente mal especificada. A `ST_Reskew` resulta em um raster tendo largura e altura diferentes, calculadas para caber na extensão geográfica do raster de entrada. A `ST_SetSkew` não modifica a largura, nem a altura do raster.

Disponibilidade: 2.0.0 Requer GDAL 1.6.1+

Alterações: 2.1.0 Funciona em rasters sem SRID



## Exemplos

Um exemplo simples de reskewing um raster de uma inclinação de 0.0 para uma de 0.0015.

```
-- the original raster non-rotated
SELECT ST_Rotation(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, 0, 0, 4269) ←
, '8BUI'::text, 1, 0));

-- result
0

-- the reskewed raster raster rotation
SELECT ST_Rotation(ST_Reskew(ST_AddBand(ST_MakeEmptyRaster(100, 100, 0, 0, 0.001, -0.001, ←
0, 0, 4269), '8BUI'::text, 1, 0), 0.0015));

-- result
-0.982793723247329
```

## Veja também

[ST\\_Resample](#), [ST\\_Rescale](#), [ST\\_SetSkew](#), [ST\\_SetRotation](#), [ST\\_SkewX](#), [ST\\_SkewY](#), [ST\\_Transform](#)

### 9.7.10 ST\_SnapToGrid

`ST_SnapToGrid` — Resample um raster encaixando-o em uma grade. Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor.

#### Synopsis

```
raster ST_SnapToGrid(raster rast, double precision gridx, double precision gridy, text algorithm=NearestNeighbour, 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=NearestNeighbour, double precision maxerr=0.125);
raster ST_SnapToGrid(raster rast, double precision gridx, double precision gridy, double precision scalexy, text algorithm=NearestNeig
double precision maxerr=0.125);
```

#### Descrição

Resample um raster encaixando-o em uma grade definida por um pixel de canto (`gridx` & `gridy`) e opcionalmente um tamanho de pixel (`scalex` & `scaley`). Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor pois é o mais rápido, mas resulta na pior interpolação.

`gridx` e `gridy` definem um canto pixel aleatório da nova grade. Não é necessário o canto esquerdo superior do novo raster e não precisa estar dentro do limite da extensão do novo raster.

You can optionally define the pixel size of the new grid with `scalex` and `scaley`.

A extensão do novo raster irá encerrar a extensão do raster fornecido.

Uma porcentagem `maxerror` de 0.125 se nenhum `maxerr` for especificado.



#### Note

Recorra a: [GDAL Warp resampling methods](#) para mais detalhes.

**Note**

Use [ST\\_Resample](#) se precisar de mais controle sobre os parâmetros da grade.

Disponibilidade: 2.0.0 Requer GDAL 1.6.1+

Alterações: 2.1.0 Funciona em rasters sem SRID

**Exemplos**

Um exemplo simples movendo um raster para um grade um pouco diferente.

```
-- the original raster upper left X
SELECT ST_UpperLeftX(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 0.001, -0.001, 0, 0, 4269) ←
, '8BUI'::text, 1, 0));
-- result
0

-- the upper left of raster after snapping
SELECT ST_UpperLeftX(ST_SnapToGrid(ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 0.001, ←
-0.001, 0, 0, 4269), '8BUI'::text, 1, 0), 0.0002, 0.0002));

--result
-0.0008
```

**Veja também**

[ST\\_Resample](#), [ST\\_Rescale](#), [ST\\_UpperLeftX](#), [ST\\_UpperLeftY](#)

**9.7.11 ST\_Resize**

`ST_Resize` — Redimensiona largura/altura novas para um raster

**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);
```

**Descrição**

Redimensiona novas largura/altura para um raster. Elas podem ser especificadas no número exato de pixels ou uma porcentagem delas. A extensão do novo raster será a mesma da extensão do raster fornecido.

Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor por ser mais rápido, porém resulta na pior interpolação.

A Variante 1 espera a largura/altura atual do raster de saída.

A Variante 2 espera os valores decimais entre zero (0) e um (1) indicando a porcentagem da largura/altura do raster.

A Variante 3 pega qualquer largura/altura do raster de saída ou uma porcentagem textual ("20%") indicando a porcentagem da largura/altura do raster de entrada.

Disponibilidade: 2.1.0 Requer GDAL 1.6.1+

## Exemplos

```

WITH foo AS (
SELECT
  1 AS rid,
  ST_Resize(
    ST_AddBand(
      ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0)
      , 1, '8BUI', 255, 0
    )
    , '50%', '500') AS rast
UNION ALL
SELECT
  2 AS rid,
  ST_Resize(
    ST_AddBand(
      ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0)
      , 1, '8BUI', 255, 0
    )
    , 500, 100) AS rast
UNION ALL
SELECT
  3 AS rid,
  ST_Resize(
    ST_AddBand(
      ST_MakeEmptyRaster(1000, 1000, 0, 0, 1, -1, 0, 0, 0)
      , 1, '8BUI', 255, 0
    )
    , 0.25, 0.9) AS rast
), bar AS (
  SELECT rid, ST_Metadata(rast) AS meta, rast FROM foo
)
SELECT rid, (meta).* FROM bar

```

rid	upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	←
1	0	0	500	500	1	-1	0	0	0	←
2	0	0	500	100	1	-1	0	0	0	←
3	0	0	250	900	1	-1	0	0	0	←

(3 rows)

## Veja também

[ST\\_Resample](#), [ST\\_Rescale](#), [ST\\_Reskew](#), [ST\\_SnapToGrid](#)

### 9.7.12 ST\_Transform

**ST\_Transform** — Reprojeta um raster em um sistema de referência espacial conhecido para outro usando um algoritmo resampling especificado. As opções são NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos com o padrão sendo 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);
```

## Descrição

Reprojeta um raster em um sistema de referência espacial conhecido para outro usando um algoritmo pixel warping especificado. Usa "NearestNeighbor" se nenhum algoritmo for especificado e a porcentagem maxerror de 0.125 se nenhum maxerr for especificado.

As opções de algoritmo são: 'NearestNeighbor', 'Bilinear', 'Cubic', 'CubicSpline', e 'Lanczos'. Recorra a: [GDAL Warp resampling methods](#) para mais detalhes.

Geralmente, aST\_Transform confundida com a ST\_SetSRID(). Na verdade, a ST\_Transform modifica as coordenadas de um raster (e resample os valores do pixel) de um sistema de referência espacial para outro, enquanto a ST\_SetSRID() só altera o identificador de SRID do raster.

Diferente das outras variantes, a 3 requer um raster referência como alignto. O raster transformado será alterado para o sistema de referência espacial (SRID) do raster referência e será alinhado (ST\_SameAlignment = VERDADE) ao raster referência.

### Note



Se achar que seu suporte de transformação não estiver funcionando corretamente, talvez precise colocar a variável de ambiente PROJSO na biblioteca de projeção .so ou .dll que seu PostGIS está usando. Isto só precisará ter o mesmo nome do arquivo, Então, por exemplo, no Windows, você iria em Painel de Controle -> Sistema -> Variáveis de Ambiente adicionar um sistema variável chamado PROJSO e colocar em libproj.dll (se estiver usando proj 4.6.1). Você terá que reiniciar seu serviço/daemon PostgreSQL depois dessa alteração.

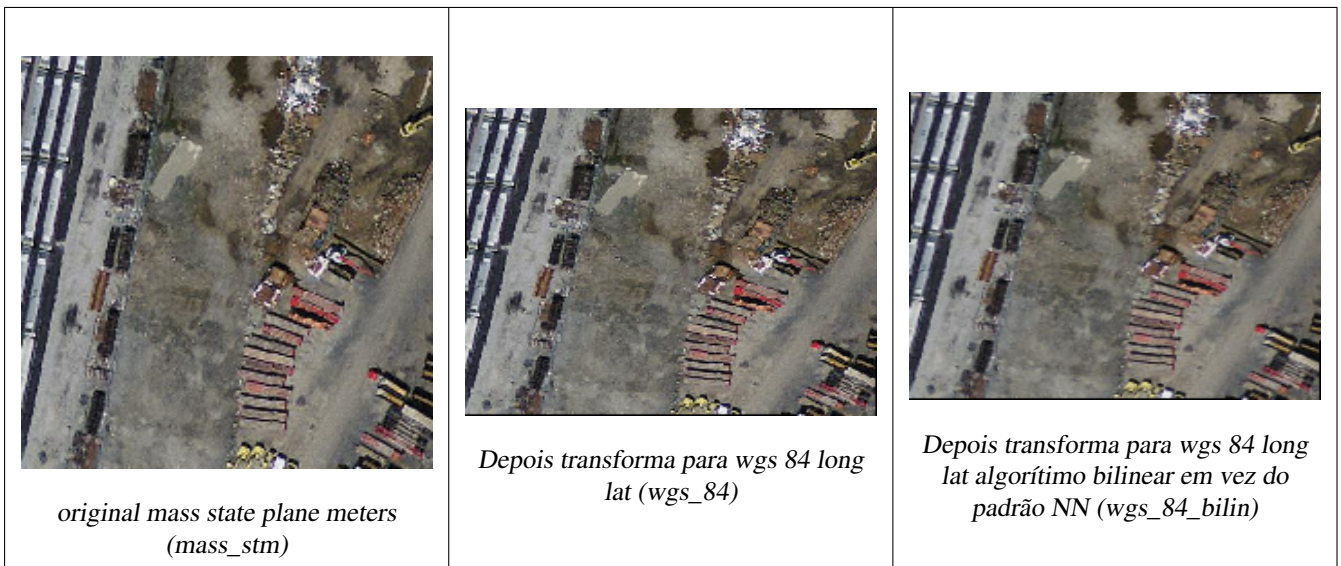
Disponibilidade: 2.0.0 Requer GDAL 1.6.1+

Melhorias: 2.1.0 Adição da variante ST\_Transform(rast, alignto)

## Exemplos

```
SELECT ST_Width(mass_stm) As w_before, ST_Width(wgs_84) As w_after,
       ST_Height(mass_stm) As h_before, ST_Height(wgs_84) As h_after
FROM
  ( SELECT rast As mass_stm, ST_Transform(rast,4326) As wgs_84
    , ST_Transform(rast,4326, 'Bilinear') AS wgs_84_bilin
  FROM aerials.o_2_boston
    WHERE ST_Intersects(rast,
                        ST_Transform(ST_MakeEnvelope(-71.128, 42.2392,-71.1277, ←
                        42.2397, 4326),26986) )
  LIMIT 1) As foo;
```

w_before	w_after	h_before	h_after
200	228	200	170



### Exemplos: Variante 3

A seguir está a diferença entre usar `ST_Transform(raster, srid)` e `ST_Transform(raster, alignto)`

```
WITH foo AS (
  SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 600000, 100, -100, 0, ←
    0, 2163), 1, '16BUI', 1, 0) AS rast UNION ALL
  SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 600000, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 2, 0) AS rast UNION ALL
  SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 600000, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 3, 0) AS rast UNION ALL

  SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 599800, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 10, 0) AS rast UNION ALL
  SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 599800, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 20, 0) AS rast UNION ALL
  SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 599800, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 30, 0) AS rast UNION ALL

  SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 599600, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 100, 0) AS rast UNION ALL
  SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 599600, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 200, 0) AS rast UNION ALL
  SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 599600, 100, -100, 0, 0, ←
    2163), 1, '16BUI', 300, 0) AS rast
), bar AS (
  SELECT
    ST_Transform(rast, 4269) AS alignto
  FROM foo
  LIMIT 1
), baz AS (
  SELECT
    rid,
    rast,
    ST_Transform(rast, 4269) AS not_aligned,
    ST_Transform(rast, alignto) AS aligned
  FROM foo
  CROSS JOIN bar
)
SELECT
```

```

    ST_SameAlignment(rast) AS rast,
    ST_SameAlignment(not_aligned) AS not_aligned,
    ST_SameAlignment(aligned) AS aligned
FROM baz

```

```

rast | not_aligned | aligned
-----+-----+-----
t   | f             | t

```

### Veja também

[ST\\_Transform](#), [ST\\_SetSRID](#)

## 9.8 Editores de Banda Raster

### 9.8.1 ST\_SetBandNoDataValue

`ST_SetBandNoDataValue` — Coloca o valor da banda que não representa nenhum dado. A banda 1 é assumida se nenhuma banda for especificada. Para marcar uma banda como tendo nenhum valor nodata, coloca ele = NULL.

#### Synopsis

```

raster ST_SetBandNoDataValue(raster rast, double precision nodatavalue);
raster ST_SetBandNoDataValue(raster rast, integer band, double precision nodatavalue, boolean forcechecking=false);

```

#### Descrição

Coloca o valor que não representa nenhum dado para a banda. A banda 1 é assumida se não especificada. Isso irá afetar os resultados de [ST\\_Polygon](#), [ST\\_DumpAsPolygons](#), e as funções `ST_PixelAs...()`.

#### Exemplos

```

-- change just first band no data value
UPDATE dummy_rast
   SET rast = ST_SetBandNoDataValue(rast,1, 254)
WHERE rid = 2;

-- change no data band value of bands 1,2,3
UPDATE dummy_rast
   SET rast =
       ST_SetBandNoDataValue(
           ST_SetBandNoDataValue(
               ST_SetBandNoDataValue(
                   rast,1, 254)
               ,2,99),
           ,3,108)
   WHERE rid = 2;

-- wipe out the nodata value this will ensure all pixels are considered for all processing ←
  functions
UPDATE dummy_rast
   SET rast = ST_SetBandNoDataValue(rast,1, NULL)
WHERE rid = 2;

```

**Veja também**

[ST\\_BandNoDataValue](#), [ST\\_NumBands](#)

**9.8.2 ST\_SetBandIsNoData**

ST\_SetBandIsNoData — Coloca a bandeira isnodata da banda como VERDADE.

**Synopsis**

```
raster ST_SetBandIsNoData(raster rast, integer band=1);
```

**Descrição**

Coloca a bandeira isnodata para a banda como verdade. A banda 1 é assumida se não especificada. Esta função deveria ser chamada apenas quando a bandeira for considerada suja. Isto é, quando a chamada resultado [ST\\_BandIsNoData](#) for diferente usando VERDADEIRO como último argumento e sem usá-lo.

Disponibilidade: 2.0.0

**Exemplos**

```
-- Create dummy table with one raster column
create table dummy_rast (rid integer, rast raster);

-- Add raster with two bands, one pixel/band. In the first band, nodatavalue = pixel value ←
= 3.
-- In the second band, nodatavalue = 13, pixel value = 4
insert into dummy_rast values(1,
(
'01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0200' -- nBands (uint16 0)
||
'17263529ED684A3F' -- scaleX (float64 0.000805965234044584)
||
'F9253529ED684ABF' -- scaleY (float64 -0.00080596523404458)
||
'1C9F33CE69E352C0' -- ipX (float64 -75.5533328537098)
||
'718F0E9A27A44840' -- ipY (float64 49.2824585505576)
||
'ED50EB853EC32B3F' -- skewX (float64 0.000211812383858707)
||
'7550EB853EC32B3F' -- skewY (float64 0.000211812383858704)
||
'E6100000' -- SRID (int32 4326)
||
'0100' -- width (uint16 1)
||
'0100' -- height (uint16 1)
||
'4' -- hasnodatavalue set to true, isnodata value set to false (when it should be true)
||
'2' -- first band type (4BUI)
```

```

||
'03' -- novalue==3
||
'03' -- pixel(0,0)==3 (same that nodata)
||
'0' -- hasnodatavalue set to false
||
'5' -- second band type (16BSI)
||
'0D00' -- novalue==13
||
'0400' -- pixel(0,0)==4
)::raster
);

select st_bandisnodata(rast, 1) from dummy_rast where rid = 1; -- Expected false
select st_bandisnodata(rast, 1, TRUE) from dummy_rast where rid = 1; -- Expected true

-- The isnodata flag is dirty. We are going to set it to true
update dummy_rast set rast = st_setbandisnodata(rast, 1) where rid = 1;

select st_bandisnodata(rast, 1) from dummy_rast where rid = 1; -- Expected true

```

**Veja também**

[ST\\_BandNoDataValue](#), [ST\\_NumBands](#), [ST\\_SetBandNoDataValue](#), [ST\\_BandIsNoData](#)

**9.8.3 ST\_SetBandPath**

`ST_SetBandPath` — Update the external path and band number of an out-db band

**Synopsis**

`raster` **ST\_SetBandPath**(`raster rast`, `integer band`, `text outdbpath`, `integer outdbindex`, `boolean force=false`);

**Descrição**

Updates an out-db band's external raster file path and external band number.

**Note**

If `force` is set to true, no tests are done to ensure compatibility (e.g. alignment, pixel support) between the external raster file and the PostGIS raster. This mode is intended for file system changes where the external raster resides.

---

**Note**

Internally, this method replaces the PostGIS raster's band at index `band` with a new band instead of updating the existing path information.

---

Availability: 2.5.0

---



**Exemplos**

```

WITH foo AS (
    SELECT
        ST_AddBand(NULL::raster, '/home/pele/devel/geo/postgis-git/raster/test/ ↵
        regress/loader/Projected.tif', NULL::int[]) AS rast
)
SELECT
    1 AS query,
    *
FROM ST_BandMetadata(
    (SELECT rast FROM foo),
    ARRAY[1,3,2]::int[]
)
UNION ALL
SELECT
    2,
    *
FROM ST_BandMetadata(
    (
        SELECT
            ST_SetBandPath(
                rast,
                2,
                '/home/pele/devel/geo/postgis-git/raster/test/regress/ ↵
                loader/Projected2.tif',
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1,3,2]::int[]
)
ORDER BY 1, 2;

```

query	bandnum	pixeltypes	nodatavalues	isoutdb	path
1	1	8BUI		t	/home/pele/devel/geo/postgis-git/ ↵ raster/test/regress/loader/Projected.tif   1
1	2	8BUI		t	/home/pele/devel/geo/postgis-git/ ↵ raster/test/regress/loader/Projected.tif   2
1	3	8BUI		t	/home/pele/devel/geo/postgis-git/ ↵ raster/test/regress/loader/Projected.tif   3
2	1	8BUI		t	/home/pele/devel/geo/postgis-git/ ↵ raster/test/regress/loader/Projected.tif   1
<b>2</b>	<b>2</b>	<b>8BUI</b>		<b>t</b>	<b>/home/pele/devel/geo/postgis-git/ ↵ raster/test/regress/loader/Projected2.tif   1</b>
2	3	8BUI		t	/home/pele/devel/geo/postgis-git/ ↵ raster/test/regress/loader/Projected.tif   3

**Veja também**

[ST\\_BandMetaData](#), [ST\\_SetBandIndex](#)

**9.8.4 ST\_SetBandIndex**

ST\_SetBandIndex — Update the external band number of an out-db band

## Synopsis

raster **ST\_SetBandIndex**(raster rast, integer band, integer outdbindex, boolean force=false);

## Descrição

Updates an out-db band's external band number. This does not touch the external raster file associated with the out-db band



### Note

If `force` is set to true, no tests are done to ensure compatibility (e.g. alignment, pixel support) between the external raster file and the PostGIS raster. This mode is intended for where bands are moved around in the external raster file.



### Note

Internally, this method replaces the PostGIS raster's band at index `band` with a new band instead of updating the existing path information.

Availability: 2.5.0

## Exemplos

```
WITH foo AS (
    SELECT
        ST_AddBand(NULL::raster, '/home/pele/devel/geo/postgis-git/raster/test/ ↵
        regress/loader/Projected.tif', NULL::int[]) AS rast
)
SELECT
    1 AS query,
    *
FROM ST_BandMetadata (
    (SELECT rast FROM foo),
    ARRAY[1,3,2]::int[]
)
UNION ALL
SELECT
    2,
    *
FROM ST_BandMetadata (
    (
        SELECT
            ST_SetBandIndex (
                rast,
                2,
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1,3,2]::int[]
)
ORDER BY 1, 2;

query | bandnum | pixeltype | nodatavalue | isoutdb | ↵
      |         |           |             |         | path | ↵
      |         |           |             |         | outdbbandnum
```



## Exemplos

```
--example will count all pixels not 249 and one will count all pixels. --
SELECT rid, ST_Count(ST_SetBandNoDataValue(rast,249)) As exclude_nodata,
       ST_Count(ST_SetBandNoDataValue(rast,249),false) As include_nodata
FROM dummy_rast WHERE rid=2;
```

rid	exclude_nodata	include_nodata
2	23	25

## Veja também

[ST\\_CountAgg](#), [ST\\_SummaryStats](#), [ST\\_SetBandNoDataValue](#)

### 9.9.2 ST\_CountAgg

**ST\_CountAgg** — Agregado. Retorna o número de pixels em uma banda dada de um raster ou cobertura raster. Se nenhuma banda for especificada, o padrão é usar a banda 1. Se `exclude_nodata_value` for verdade, contará somente pixels que são diferentes ao valor NODATA.

#### 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);
```

#### Descrição

Retorna o número de pixels em uma banda de um conjunto de rasters. Se nenhuma banda foi especificada `nband` usa-se 1.

Se `exclude_nodata_value` for verdade, contará apenas pixels com valor diferente do valor NODATA do raster. `exclude_nodata_value` é falso para contar todos os pixels.

Por padrão irá tomar todos os pixels. Para obter uma resposta mais rápida, coloque `sample_percent` no valor entre zero (0) e um (1)

Disponibilidade: 2.2.0

## Exemplos

```
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, 0)
            , 1, '64BF', 0, 0
          )
          , 1, 1, 1, -10
        )
      )
    , 1, 5, 4, 0
```

```

        )
        , 1, 5, 5, 3.14159
    ) AS rast
) AS rast
FULL JOIN (
    SELECT generate_series(1, 10) AS id
) AS id
    ON 1 = 1
)
SELECT
    ST_CountAgg(rast, 1, TRUE)
FROM foo;

 st_countagg
-----
            20
(1 row)

```

### Veja também

[ST\\_Count](#), [ST\\_SummaryStats](#), [ST\\_SetBandNoDataValue](#)

## 9.9.3 ST\_Histogram

**ST\_Histogram** — Retorna um conjunto de registros que resumem um raster ou distribuição de dados de cobertura raster intervalos bin separados. O número de bins é auto calculado.

### 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);
SETOF record ST_Histogram(text rastertable, text rastercolumn, integer nband, integer bins, boolean right);
SETOF record ST_Histogram(text rastertable, text rastercolumn, integer nband, boolean exclude_nodata_value, integer bins,
boolean right);
SETOF record ST_Histogram(text rastertable, text rastercolumn, integer nband=1, boolean exclude_nodata_value=true, integer
bins=autocomputed, double precision[] width=NULL, boolean right=false);
SETOF record ST_Histogram(text rastertable, text rastercolumn, integer nband=1, integer bins, double precision[] width=NULL,
boolean right=false);

```

### Descrição

retorna um conjunto de registros de porcentagens min, max, count, para uma banda raster dada para cada bin. Se nenhuma banda for especificada nband usa-se 1.



#### Note

Por padrão só considera valores de pixels diferentes do valor nodata. `exclude_nodata_value` é falso para contar todos os pixels.

**width double precision[]** largura: um arranjo indicando a largura de cada categoria/bin. Se o número de bins for maior que o número de larguras, elas são repetidas.

Exemplo: 9 bins, larguras são [a, b, c] terão a saída como [a, b, c, a, b, c, a, b, c]

**bins integer** Número de fugas -- este é o número de registros que terá de volta da função especificada. Se não especificado, o número de fugas é auto calculado.

**right boolean** calcula o histograma da direita ao invés do da esquerda (padrão). Isto altera o critério de avaliar um valor x de [a, b) para (a, b]

Disponibilidade: 2.0.0

#### Exemplo: Única raster tile - calcula histogramas para bandas 1, 2, 3 e auto calcula bins.

```
SELECT band, (stats).*
FROM (SELECT rid, band, ST_Histogram(rast, band) As stats
      FROM dummy_rast CROSS JOIN generate_series(1,3) As band
      WHERE rid=2) As foo;
```

band	min	max	count	percent
1	249	250	2	0.08
1	250	251	2	0.08
1	251	252	1	0.04
1	252	253	2	0.08
1	253	254	18	0.72
2	78	113.2	11	0.44
2	113.2	148.4	4	0.16
2	148.4	183.6	4	0.16
2	183.6	218.8	1	0.04
2	218.8	254	5	0.2
3	62	100.4	11	0.44
3	100.4	138.8	5	0.2
3	138.8	177.2	4	0.16
3	177.2	215.6	1	0.04
3	215.6	254	4	0.16

#### Exemplo: Apenas banda 2 mas para 6 bins

```
SELECT (stats).*
FROM (SELECT rid, ST_Histogram(rast, 2,6) As stats
      FROM dummy_rast
      WHERE rid=2) As foo;
```

min	max	count	percent
78	107.333333	9	0.36
107.333333	136.666667	6	0.24
136.666667	166	0	0
166	195.333333	4	0.16
195.333333	224.666667	1	0.04
224.666667	254	5	0.2

(6 rows)

-- Same as previous but we explicitly control the pixel value range of each bin.

```
SELECT (stats).*
FROM (SELECT rid, ST_Histogram(rast, 2,6,ARRAY[0.5,1,4,100,5]) As stats
      FROM dummy_rast
```

```
WHERE rid=2) As foo;
```

min	max	count	percent
78	78.5	1	0.08
78.5	79.5	1	0.04
79.5	83.5	0	0
83.5	183.5	17	0.0068
183.5	188.5	0	0
188.5	254	6	0.003664

(6 rows)

## Veja também

[ST\\_Count](#), [ST\\_SummaryStats](#), [ST\\_SummaryStatsAgg](#)

### 9.9.4 ST\_Quantile

**ST\_Quantile** — Calcula quantiles para um raster ou cobertura de tabela raster no contexto da amostra ou população. Assim, um valor poderia ser examinado para estar na porcentagem 25%, 50%, 75% do raster.

#### Synopsis

```
SETOF record ST_Quantile(raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] quantiles=NULL);
SETOF record ST_Quantile(raster rast, double precision[] quantiles);
SETOF record ST_Quantile(raster rast, integer nband, double precision[] quantiles);
double precision ST_Quantile(raster rast, double precision quantile);
double precision ST_Quantile(raster rast, boolean exclude_nodata_value, double precision quantile=NULL);
double precision ST_Quantile(raster rast, integer nband, double precision quantile);
double precision ST_Quantile(raster rast, integer nband, boolean exclude_nodata_value, double precision quantile);
double precision ST_Quantile(raster rast, integer nband, double precision quantile);
SETOF record ST_Quantile(text rastertable, text rastercolumn, integer nband=1, boolean exclude_nodata_value=true, double
precision[] quantiles=NULL);
SETOF record ST_Quantile(text rastertable, text rastercolumn, integer nband, double precision[] quantiles);
```

#### Descrição

Calcula quantiles para um raster ou cobertura de tabela raster no contexto da amostra ou população. Assim, um valor poderia ser examinado para estar na porcentagem 25%, 50%, 75% do raster.



#### Note

Se `exclude_nodata_value` for falso, contará também pixels sem dados.

Disponibilidade: 2.0.0

#### Exemplos

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast,249) WHERE rid=2;
--Example will consider only pixels of band 1 that are not 249 and in named quantiles --

SELECT (pvq).*
```

```
FROM (SELECT ST_Quantile(rast, ARRAY[0.25,0.75]) As pvq
      FROM dummy_rast WHERE rid=2) As foo
ORDER BY (pvq).quantile;
```

```
quantile | value
-----+-----
    0.25 |   253
    0.75 |   254
```

```
SELECT ST_Quantile(rast, 0.75) As value
FROM dummy_rast WHERE rid=2;
```

```
value
-----
   254
```

```
--real live example.  Quantile of all pixels in band 2 intersecting a geometry
SELECT rid, (ST_Quantile(rast,2)).* As pvc
FROM o_4_boston
WHERE ST_Intersects(rast,
                    ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706 892151,224486 892151))',26986)
                    )
ORDER BY value, quantile,rid
;
```

```
rid | quantile | value
-----+-----
  1 |         0 |     0
  2 |         0 |     0
 14 |         0 |     1
 15 |         0 |     2
 14 |        0.25 |    37
  1 |        0.25 |    42
 15 |        0.25 |    47
  2 |        0.25 |    50
 14 |        0.5 |    56
  1 |        0.5 |    64
 15 |        0.5 |    66
  2 |        0.5 |    77
 14 |        0.75 |    81
 15 |        0.75 |    87
  1 |        0.75 |    94
  2 |        0.75 |   106
 14 |         1 |   199
  1 |         1 |   244
  2 |         1 |   255
 15 |         1 |   255
```

## Veja também

[ST\\_Count](#), [ST\\_SummaryStats](#), [ST\\_SummaryStatsAgg](#), [ST\\_SetBandNoDataValue](#)

### 9.9.5 ST\_SummaryStats

**ST\_SummaryStats** — Retorna as estatísticas resumidas consistindo de count, sum, mean, stddev, min, max para uma dada banda raster de um raster ou cobertura raster. A banda 1 é assumida se nenhuma banda for especificada.



## Synopsis

```
summarystats ST_SummaryStats(raster rast, boolean exclude_nodata_value);
summarystats ST_SummaryStats(raster rast, integer nband, boolean exclude_nodata_value);
summarystats ST_SummaryStats(text rastertable, text rastercolumn, boolean exclude_nodata_value);
summarystats ST_SummaryStats(text rastertable, text rastercolumn, integer nband=1, boolean exclude_nodata_value=true);
```

## Descrição

Retorna `summarystats` consistindo de count, sum, mean, stddev, min, max para uma dada banda raster de um raster ou cobertura raster. Se nenhuma banda for especificada `nband` usa-se a 1.



### Note

Por padrão só considera valores de pixels diferentes do valor `nodata`. `exclude_nodata_value` é falso para contar todos os pixels.



### Note

Por padrão irá tomar todos os pixels. Para obter uma resposta mais rápida, use `sample_percent` para menor que 1

Disponibilidade: 2.0.0



### Warning

As variantes `ST_SummaryStats(rastertable, rastercolumn, ...)` são deprecadas como da 2.2.0. Ao contrário, use: `ST_SummaryStatsAgg`.

## Exemplo: Única tile raster

```
SELECT rid, band, (stats).*
FROM (SELECT rid, band, ST_SummaryStats(rast, band) As stats
      FROM dummy_rast CROSS JOIN generate_series(1,3) As band
      WHERE rid=2) As foo;
```

rid	band	count	sum	mean	stddev	min	max
2	1	23	5821	253.086957	1.248061	250	254
2	2	25	3682	147.28	59.862188	78	254
2	3	25	3290	131.6	61.647384	62	254

## Exemplo: Resuma pixels que intersectam construções de interesse

Este exemplo tomou 574ms no PostGIS windows 64-bit com todas as construções de Boston e tiles aéreas (cada uma com 150x150 pixels ~ 134,000 tiles), ~102,000 registros de construções

```
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(feats.geom,rast)
) As foo
)
-- finally summarize stats
SELECT building_id, SUM(count) As num_pixels
  , MIN(min) As min_pval
  , MAX(max) As max_pval
  , SUM(mean*count)/SUM(count) As avg_pval
  FROM b_stats
WHERE count
> 0
  GROUP BY building_id
  ORDER BY building_id;
building_id | num_pixels | min_pval | max_pval | avg_pval
-----+-----+-----+-----+-----
      100 |      1090 |         1 |        255 | 61.0697247706422
      103 |       655 |         7 |        182 | 70.5038167938931
      150 |       895 |         2 |        252 | 185.642458100559

```

### Exemplo: Cobertura raster

```

-- stats for each band --
SELECT band, (stats).*
FROM (SELECT band, ST_SummaryStats('o_4_boston','rast', band) As stats
      FROM generate_series(1,3) As band) As foo;

band | count | sum | mean | stddev | min | max
-----+-----+-----+-----+-----+-----+-----
    1 | 8450000 | 725799 | 82.7064349112426 | 45.6800222638537 | 0 | 255
    2 | 8450000 | 700487 | 81.4197705325444 | 44.2161184161765 | 0 | 255
    3 | 8450000 | 575943 | 74.682739408284 | 44.2143885481407 | 0 | 255

-- For a table -- will get better speed if set sampling to less than 100%
-- Here we set to 25% and get a much faster answer
SELECT band, (stats).*
FROM (SELECT band, ST_SummaryStats('o_4_boston','rast', band,true,0.25) As stats
      FROM generate_series(1,3) As band) As foo;

band | count | sum | mean | stddev | min | max
-----+-----+-----+-----+-----+-----+-----
    1 | 2112500 | 180686 | 82.6890480473373 | 45.6961043857248 | 0 | 255
    2 | 2112500 | 174571 | 81.448503668639 | 44.2252623171821 | 0 | 255
    3 | 2112500 | 144364 | 74.6765884023669 | 44.2014869384578 | 0 | 255

```

### Veja também

[summarystats](#), [ST\\_SummaryStatsAgg](#), [ST\\_Count](#), [ST\\_Clip](#)

## 9.9.6 ST\_SummaryStatsAgg

`ST_SummaryStatsAgg` — Agregado. Retorna as estatísticas resumidas consistindo de count, sum, mean, stddev, min, max para uma dada banda raster de um conjunto de rasters. A banda 1 é assumida se nenhuma banda for especificada.

### 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);
```

### Descrição

Retorna `summarystats` consistindo de count, sum, mean, stddev, min, max para uma dada banda raster de um raster ou cobertura raster. Se nenhuma banda for especificada `nband` usa-se a 1.



#### Note

Por padrão só considera valores de pixels diferentes do valor NODATA. `exclude_nodata_value` é falso para contar todos os pixels.



#### Note

Por padrão irá tomar todos os pixels. Para obter uma resposta mais rápida, coloque `sample_percent` no valor entre zero 0 e um 1

Disponibilidade: 2.2.0

### Exemplos

```
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,
              0,0)
            , 1, '64BF', 0, 0
          )
          , 1, 1, 1, -10
        )
        , 1, 5, 4, 0
      )
      , 1, 5, 5, 3.14159
    ) AS rast
  ) AS rast
  FULL JOIN (
    SELECT generate_series(1, 10) AS id
  ) AS id
  ON 1 = 1
)
SELECT
```

```

        (stats).count,
        round((stats).sum::numeric, 3),
        round((stats).mean::numeric, 3),
        round((stats).stddev::numeric, 3),
        round((stats).min::numeric, 3),
        round((stats).max::numeric, 3)
FROM (
    SELECT
        ST_SummaryStatsAgg(rast, 1, TRUE, 1) AS stats
    FROM foo
) bar;

```

```

count | round | round | round | round | round
-----+-----+-----+-----+-----+-----
      20 | -68.584 | -3.429 | 6.571 | -10.000 | 3.142
(1 row)

```

## Veja também

[summarystats](#), [ST\\_SummaryStats](#), [ST\\_Count](#), [ST\\_Clip](#)

## 9.9.7 ST\_ValueCount

**ST\_ValueCount** — Retorna o conjunto de registros contendo uma banda pixel de valor e conta do número de pixels em uma dada banda de um raster (ou uma cobertura raster) que tem um dado conjunto de valores. Usa-se a banda 1 se nenhuma for especificada. Por padrão pixels de valor nodata não são contados. Todos os outros valores no pixel são saída e os valores de pixels são arredondados para o inteiro mais próximo.

### 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);

```

### Descrição

Retorna um conjunto de registros com colunas `value count` que contêm o valor da banda pixel e soma de pixels na tile raster ou cobertura raster da banda selecionada.

Se nenhuma banda for especificada `nband` usa-se 1. Se nenhum `searchvalues` for especificado, retornarão pixels com valores encontrados no raster ou cobertura raster. Se um valor de pesquisa for dado, retornará um inteiro em vez de registros indicando a soma de pixels que têm aquele valor de banda pixel

**Note**

Se `excluye_nodata_value` for falso, contará também pixels sem dados.

Disponibilidade: 2.0.0

**Exemplos**

```
UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast,249) WHERE rid=2;
--Example will count only pixels of band 1 that are not 249. --
```

```
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast) As pvc
      FROM dummy_rast WHERE rid=2) As foo
      ORDER BY (pvc).value;
```

value	count
250	2
251	1
252	2
253	6
254	12

```
-- Example will count all pixels of band 1 including 249 --
```

```
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast,1,false) As pvc
      FROM dummy_rast WHERE rid=2) As foo
      ORDER BY (pvc).value;
```

value	count
249	2
250	2
251	1
252	2
253	6
254	12

```
-- Example will count only non-nodata value pixels of band 2
```

```
SELECT (pvc).*
FROM (SELECT ST_ValueCount(rast,2) As pvc
      FROM dummy_rast WHERE rid=2) As foo
      ORDER BY (pvc).value;
```

value	count
78	1
79	1
88	1
89	1
96	1
97	1
98	1
99	2

```
112 | 2
:
```

```
--real live example. Count all the pixels in an aerial raster tile band 2 intersecting a
geometry
-- and return only the pixel band values that have a count
> 500
SELECT (pvc).value, SUM((pvc).count) As total
FROM (SELECT ST_ValueCount(rast,2) As pvc
      FROM o_4_boston
      WHERE ST_Intersects(rast,
        ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706
          892151,224486 892151))',26986)
        ) As foo
      GROUP BY (pvc).value
      HAVING SUM((pvc).count)
> 500
      ORDER BY (pvc).value;

value | total
-----+-----
51 | 502
54 | 521
```

```
-- Just return count of pixels in each raster tile that have value of 100 of tiles that
intersect a specific geometry --
SELECT rid, ST_ValueCount(rast,2,100) As count
FROM o_4_boston
WHERE ST_Intersects(rast,
  ST_GeomFromText('POLYGON((224486 892151,224486 892200,224706 892200,224706
    892151,224486 892151))',26986)
  ) ;

rid | count
-----+-----
1 | 56
2 | 95
14 | 37
15 | 64
```

## Veja também

[ST\\_Count](#), [ST\\_SetBandNoDataValue](#)

## 9.10 Raster Inputs

### 9.10.1 ST\_RastFromWKB

`ST_RastFromWKB` — Return a raster value from a Well-Known Binary (WKB) raster.

#### Synopsis

raster `ST_RastFromWKB`(bytea wkb);









## Using PostgreSQL Large Object Support to export raster

One way to export raster into another format is using [PostgreSQL large object export functions](#). We'll repeat the prior example but also exporting. Note for this you'll need to have super user access to db since it uses server side lo 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.

```
DROP TABLE IF EXISTS tmp_out ;

CREATE TABLE tmp_out AS
SELECT lo_from_bytea(0,
    ST_AsGDALRaster(ST_Union(rast), 'JPEG', ARRAY['QUALITY=50'])
    ) AS loid
FROM dummy_rast
WHERE rast && ST_MakeEnvelope(10, 10, 11, 11);

SELECT lo_export(loid, '/tmp/dummy.jpg')
FROM tmp_out;

SELECT lo_unlink(loid)
FROM tmp_out;
```

## GTIFF Output Examples

```
SELECT ST_AsGDALRaster(rast, 'GTiff') As rastjpg
FROM dummy_rast WHERE rid=2;

-- Out GeoTiff with jpeg compression, 90% quality
SELECT ST_AsGDALRaster(rast, 'GTiff',
    ARRAY['COMPRESS=JPEG', 'JPEG_QUALITY=90'],
    4269) As rasttiff
FROM dummy_rast WHERE rid=2;
```

## Veja também

Section [5.3](#), [ST\\_GDALDrivers](#), [ST\\_SRID](#)

### 9.11.4 ST\_AsJPEG

**ST\_AsJPEG** — Retorna as bandas tile raster selecionadas como uma única Joint Photographic Exports Group (JPEG) image (byte arranjo). Se nenhuma banda for especificada e 1 ou mais que 3 bandas, então somente a primeira banda é usada. Se somente 3 bandas, então todas as 3 bandas serão usadas para mapear par 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);
```

## Descrição

Retorna as bandas selecionadas do raster como uma única Joint Photographic Exports Group Image (JPEG). Use [ST\\_AsGDALRaster](#) se precisar exportar como tipos raster menos comuns. Se nenhuma banda for especificada e 1 ou mais que 3 bandas, então somente a primeira é usada. Se 3 bandas, então 3 bandas são usadas. Existem muitas variantes da função com várias opções. Elas estão listadas abaixo:

- `nband` é para exportação de uma única banda.
- `nbands` é um arranjo para exportar (note que o máximo é 3 para JPEG) e a ordem das bandas é RGB. ex.: `ARRAY[3,2,1]` significa mapa banda 3 para Vermelho, banda 2 para verde e banda 1 para azul.
- `quality` número de 0 a 100. Quanto maior o número mais translúcida a imagem.
- `options` opções de textos Array of GDAL definidas para JPEG (veja em `create_options` para JPEG [ST\\_GDALDrivers](#)). Para JPEG válido eles são `PROGRESSIVE ON` or `OFF` e `QUALITY a range from 0 to 100 and default to 75`. Recorra a [GDAL Raster format options](#) para mais dealhes.

Disponibilidade: 2.0.0 - requer GDAL  $\geq$  1.6.0.

## Exemplos: Saída

```
-- output first 3 bands 75% quality
SELECT ST_AsJPEG(rast) As rastjpg
      FROM dummy_rast WHERE rid=2;

-- output only first band as 90% quality
SELECT ST_AsJPEG(rast,1,90) As rastjpg
      FROM dummy_rast WHERE rid=2;

-- output first 3 bands (but make band 2 Red, band 1 green, and band 3 blue, progressive ←
  and 90% quality
SELECT ST_AsJPEG(rast,ARRAY[2,1,3],ARRAY['QUALITY=90','PROGRESSIVE=ON']) As rastjpg
      FROM dummy_rast WHERE rid=2;
```

## Veja também

Section [5.3](#), [ST\\_GDALDrivers](#), [ST\\_AsGDALRaster](#), [ST\\_AsPNG](#), [ST\\_AsTIFF](#)

### 9.11.5 ST\_AsPNG

`ST_AsPNG` — Retorna as bandas tile raster selecionadas como um gráfico de rede portátil (PNG) imagem (byte array). Se as bandas raster 1, 3 ou 4 e nenhum banda for especificado, então todas as bandas são usadas. Se mais 2 ou mais que 4 bandas e nenhuma banda forem especificadas, então somente a banda 1 é usada. As bandas são mapeadas para espaço RGB ou RGBA.

## Synopsis

```
bytea ST_AsPNG(raster rast, text[] options=NULL);
bytea ST_AsPNG(raster rast, integer nband, integer compression);
bytea ST_AsPNG(raster rast, integer nband, text[] options=NULL);
bytea ST_AsPNG(raster rast, integer[] nbands, integer compression);
bytea ST_AsPNG(raster rast, integer[] nbands, text[] options=NULL);
```

## Descrição

Retorna as bandas selecionadas do raster como uma única Portable Network Graphics Image (PNG). Use [ST\\_AsGDALRaster](#) se precisar exportar como os tipo de raster menos comuns. Se nenhuma banda for especificada, então as 3 primeiras bandas serão exportadas. Existem muitas variantes da função com várias opções. Se nenhum `srid` for especificado, o `srid` do raster é usado. Eles estão listados abaixo:

- `nband` é para exportação de uma única banda.
- `nbands` é um arranjo para exportar (note que o máximo é 4 para JPEG) e a ordem das bandas é RGB. ex.: `ARRAY[3,2,1]` significa mapa banda 3 para Vermelho, banda 2 para verde e banda 1 para azul.
- `compression` número de 1 a 9. Quanto maior o número melhor a compressão.
- `options` opções de textos do Arranjo do GDAL como definidas para PNG (veja em `create_options` para PNG da [ST\\_GDALDrivers](#)). Para PNG válido é somente `ZLEVEL` (porção de tempo para gastar na compressão -- padrão 6) ex.: `ARRAY['ZLEVEL=9']`. `WORLDFILE` não é permitido já que a função teria que gerar duas saídas. Recorra a [GDAL Raster format options](#) para mais detalhes.

Disponibilidade: 2.0.0 - requer GDAL >= 1.6.0.

## Exemplos

```
SELECT ST_AsPNG(rast) As rastpng
FROM dummy_rast WHERE rid=2;

-- export the first 3 bands and map band 3 to Red, band 1 to Green, band 2 to blue
SELECT ST_AsPNG(rast, ARRAY[3,1,2]) As rastpng
FROM dummy_rast WHERE rid=2;
```

## Veja também

[ST\\_AsGDALRaster](#), [ST\\_ColorMap](#), [ST\\_GDALDrivers](#), [Section 5.3](#)

### 9.11.6 ST\_AsTIFF

`ST_AsTIFF` — Return the raster selected bands as a single TIFF image (byte array). If no band is specified or any of specified bands does not exist in the raster, then will try to use all bands.

## 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);
```

## Descrição

Retorna as bandas selecionadas do raster como um formato Tagged Image File Format (TIFF) único. Se nenhuma banda estiver especificada, tentaremos usar todas as bandas. Isto é uma envoltório em torno da [ST\\_AsGDALRaster](#). Use [ST\\_AsGDALRaster](#) se precisar exportar como tipos raster menos comuns. Existem muitas variantes da função com diversas opções. Se nenhuma texto de referência espacial SRS estiver presente, a referência espacial do raster é usada. Elas estão listadas abaixo:

- `nbands` é um arranjo de bandas para exportar (note que o máximo é 3 para PNG) e a ordem das bandas é RGB. ex.: `ARRAY[3,2,1]` significa mapear banda 3 para vermelho, banda 2 para verde e banda 1 para azul
- `compression` Expressão de compressão -- JPEG90 (ou algum outro percentual), LZW, JPEG, DEFLATE9.
- `options` text Array of GDAL create options as defined for GTiff (look at `create_options` for GTiff of [ST\\_GDALDrivers](#)). or refer to [GDAL Raster format options](#) for more details.
- `srid` srid do `spatial_ref_sys` do raster. É usado para popular a informação georreferência

Disponibilidade: 2.0.0 - requer GDAL >= 1.6.0.

#### Exemplo: Use jpeg compressão 90%

```
SELECT ST_AsTIFF(rast, 'JPEG90') As rasttiff
FROM dummy_rast WHERE rid=2;
```

#### Veja também

[ST\\_GDALDrivers](#), [ST\\_AsGDALRaster](#), [ST\\_SRID](#)

## 9.12 Processamento Raster

### 9.12.1 Mapa Algébrico

#### 9.12.1.1 ST\_Clip

`ST_Clip` — Retorna o raster suprimido pela geometria de entrada. Se o número de banda não for especificado, todas as bandas são processadas. Se `crop` não for especificado ou for VERDADE, o raster de saída é cortado.

#### Synopsis

```
raster ST_Clip(raster rast, integer[] nband, geometry geom, double precision[] nodataval=NULL, boolean crop=TRUE);
raster ST_Clip(raster rast, integer nband, geometry geom, double precision nodataval, boolean crop=TRUE);
raster ST_Clip(raster rast, integer nband, geometry geom, boolean crop);
raster ST_Clip(raster rast, geometry geom, double precision[] nodataval=NULL, boolean crop=TRUE);
raster ST_Clip(raster rast, geometry geom, double precision nodataval, boolean crop=TRUE);
raster ST_Clip(raster rast, geometry geom, boolean crop);
```

#### Descrição

Retorna um raster que é suprimido pela geometria de entrada `geom`. Se o índice de banda não for especificado, todas as bandas são processadas.

Os rasters resultantes da `ST_Clip` devem ter o valor nodata designado para as áreas suprimidas, um para cada banda. Se nenhum for promovido e o raster de entrada não tiver nenhum valor nodata definido, os valores nodata do raster resultante são `ST_MinPossibleValue(ST_BandPixelType(rast, band))`. Quando o número de valor nodata no arranjo é menor que o número de banda, o último no arranjo é usado para as bandas que sobraram. Se o número de valor nodata for maior que o número de banda, os valores extras serão ignorados. Todas as variantes que aceitam um arranjo de valores nodata também aceitam um valor único, que pode ser designado para cada banda.

Se `crop` não for especificado, é verdade, significando que o raster de saída é cortado para a intersecção das extensões `geom` e `rast`. Se `crop` for falso, o novo raster tem a mesma extensão que `rast`.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Reescrito em C

Os exemplos aqui utilizam os dados areais de Massachusetts disponíveis no site MassGIS [MassGIS Aerial Orthos](#). As coordenadas estão no Massachusetts State Plane Meters.

### Exemplos: 1 banda suprimindo

```
-- Clip the first band of an aerial tile by a 20 meter buffer.
SELECT ST_Clip(rast, 1,
              ST_Buffer(ST_Centroid(ST_Envelope(rast)),20)
              ) from aerials.boston
WHERE rid = 4;
```

```
-- Demonstrate effect of crop on final dimensions of raster
-- Note how final extent is clipped to that of the geometry
-- if crop = true
SELECT ST_XMax(ST_Envelope(ST_Clip(rast, 1, clipper, true))) As xmax_w_trim,
       ST_XMax(clipper) As xmax_clipper,
       ST_XMax(ST_Envelope(ST_Clip(rast, 1, clipper, false))) As xmax_wo_trim,
       ST_XMax(ST_Envelope(rast)) As xmax_rast_orig
FROM (SELECT rast, ST_Buffer(ST_Centroid(ST_Envelope(rast)),6) As clipper
      FROM aerials.boston
      WHERE rid = 6) As foo;
```

xmax_w_trim	xmax_clipper	xmax_wo_trim	xmax_rast_orig
230657.436173996	230657.436173996	230666.436173996	230666.436173996



*Tile raster completa antes se suprimir*



*Depois de suprimir*

### Exemplos: 1 banda suprimindo sem cortes e adiciona de volta outras bandas inalteradas

```
-- Same example as before, but we need to set crop to false to be able to use ST_AddBand
-- because ST_AddBand requires all bands be the same Width and height
SELECT ST_AddBand(ST_Clip(rast, 1,
                          ST_Buffer(ST_Centroid(ST_Envelope(rast)),20),false
```

```
), ARRAY[ST_Band(rast,2),ST_Band(rast,3)] ) from aerials.boston
WHERE rid = 6;
```



*Tile raster completa antes se suprimir*



*Depois de suprimir - surreal*

**Exemplos: Suprime todas as bandas**

```
-- Clip all bands of an aerial tile by a 20 meter buffer.
-- Only difference is we don't specify a specific band to clip
-- so all bands are clipped
SELECT ST_Clip(rast,
               ST_Buffer(ST_Centroid(ST_Envelope(rast)), 20),
               false
           ) from aerials.boston
WHERE rid = 4;
```



*Tile raster completa antes se suprimir*



*Depois de suprimir*



**Veja também**

[ST\\_AddBand](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Intersection](#)

**9.12.1.2 ST\_ColorMap**

`ST_ColorMap` — Cria um novo raster de até quatro bandas 8BUI (grayscale, RGB, RGBA) do raster fonte e uma banda específica. A banda 1 usada se não especificado.

**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);
```

**Descrição**

Aplica um `colormap` à banda na `nband` do `rast` resultando em um novo raster englobado com até quatro bandas 8BUI. O número de bandas 8BUI no novo raster é determinado pelo número de cores componentes definidas no `colormap`.

Se `nband` não for especificado, a banda 1 é assumida.

`colormap` pode ser uma palavra-chave de um colormap pré definido ou um conjunto de linhas definindo o valor e a cor dos componentes.

Palavra-chave válida do `colormap` pré definida:

- `grayscale` ou `greyscale` para uma banda raster 8BUI de tons de cinza.
- `pseudocolor` para quatro bandas raster 8BUI (RGBA) com cores indo de azul para verde e para vermelho.
- `fire` para quatro bandas raster 8BUI (RGBA) com cores indo de preto para vermelho para amarelo claro.
- `bluered` para quatro bandas raster 8BUI (RGBA) com cores indo de azul para branco para vermelho.

Os usuários podem passar um conjunto de entradas (uma por linha) para `colormap` para especificar colormaps personalizados. Cada entrada consiste de cinco valores: o valor de pixel e componentes Vermelho, Verde, Azul, Alfa correspondentes (entre 0 e 255). Valores de porcentagem podem ser usados em vez de valores de pixel onde 0% e 100% são os mínimos e os máximos encontrados na banda raster. Os valores podem ser separados por vírgulas (","), tabs, dois pontos (":") e/ou espaços. O valor do pixel pode ser `nv`, `null` ou `nodata` para o valor NODATA. Um exemplo é fornecido abaixo.

```
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
```

A sintaxe do `colormap` é parecida com com a do modo do auxílio de cor do GDAL [gdaldem](#).

Palavras-chave válidas para `method`:

- `INTERPOLATE` para usar interpolação linear para misturar suavemente as cores entre os valores do pixel
- `EXACT` para combinar estritamente somente aqueles valores de pixel encontrados no colormap. Os pixels cujos valores não combinarem com uma entrada do colormap serão 0 0 0 0 (RGBA)
- `NEAREST` para usar a entrada do colormap cujos valores são mais próximos ao valor do pixel



**Note**

Uma ótima referência para o colormap é [ColorBrewer](#).

**Warning**

As bandas resultantes do novo raster não terá nenhum valor NODATA. Use [ST\\_SetBandNoDataValue](#) se precisar de um valor NODATA.

Disponibilidade: 2.1.0

**Exemplos**

Esta não é uma boa tabela para desfrutar

```
-- setup test raster table --
DROP TABLE IF EXISTS funky_shapes;
CREATE TABLE funky_shapes(rast raster);

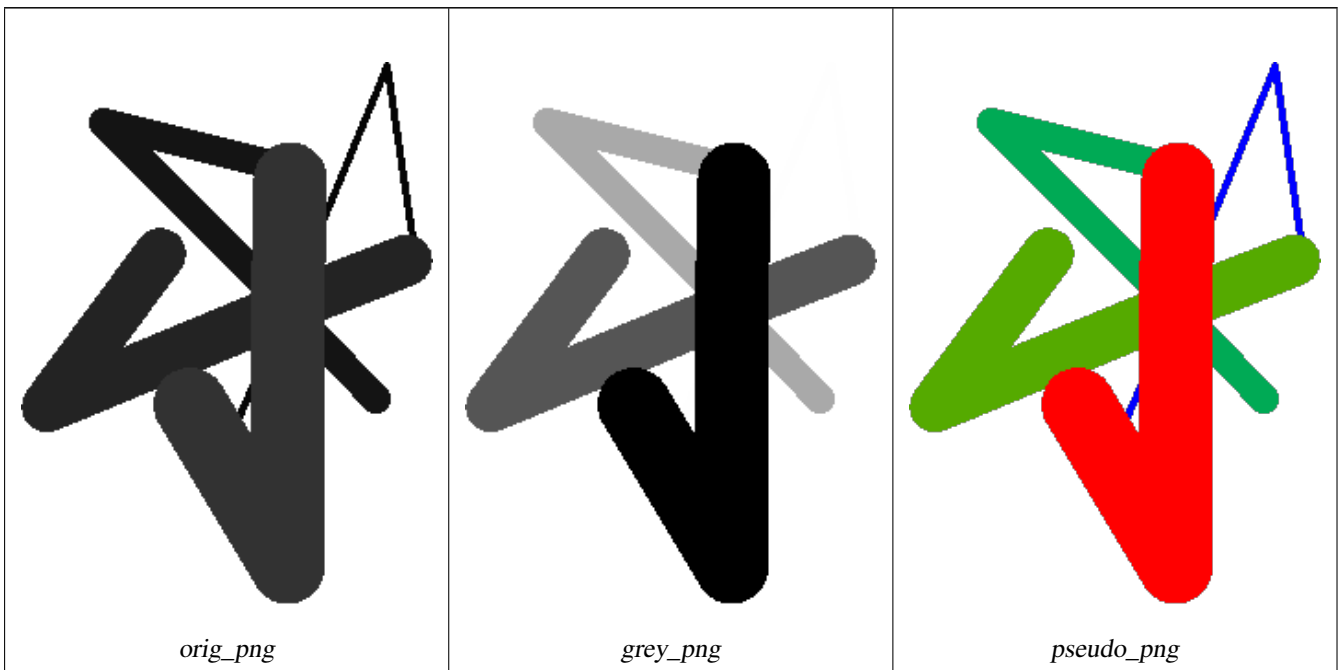
INSERT INTO funky_shapes(rast)
WITH ref AS (
    SELECT ST_MakeEmptyRaster( 200, 200, 0, 200, 1, -1, 0, 0) AS rast
)
SELECT
    ST_Union(rast)
FROM (
    SELECT
        ST_AsRaster(
            ST_Rotate(
                ST_Buffer(
                    ST_GeomFromText('LINESTRING(0 2,50 50,150 150,125 50)'),
                    i*2
                ),
                pi() * i * 0.125, ST_Point(50,50)
            ),
            ref.rast, '8BUI'::text, i * 5
        ) AS rast
    FROM ref
    CROSS JOIN generate_series(1, 10, 3) AS i
) AS shapes;
```

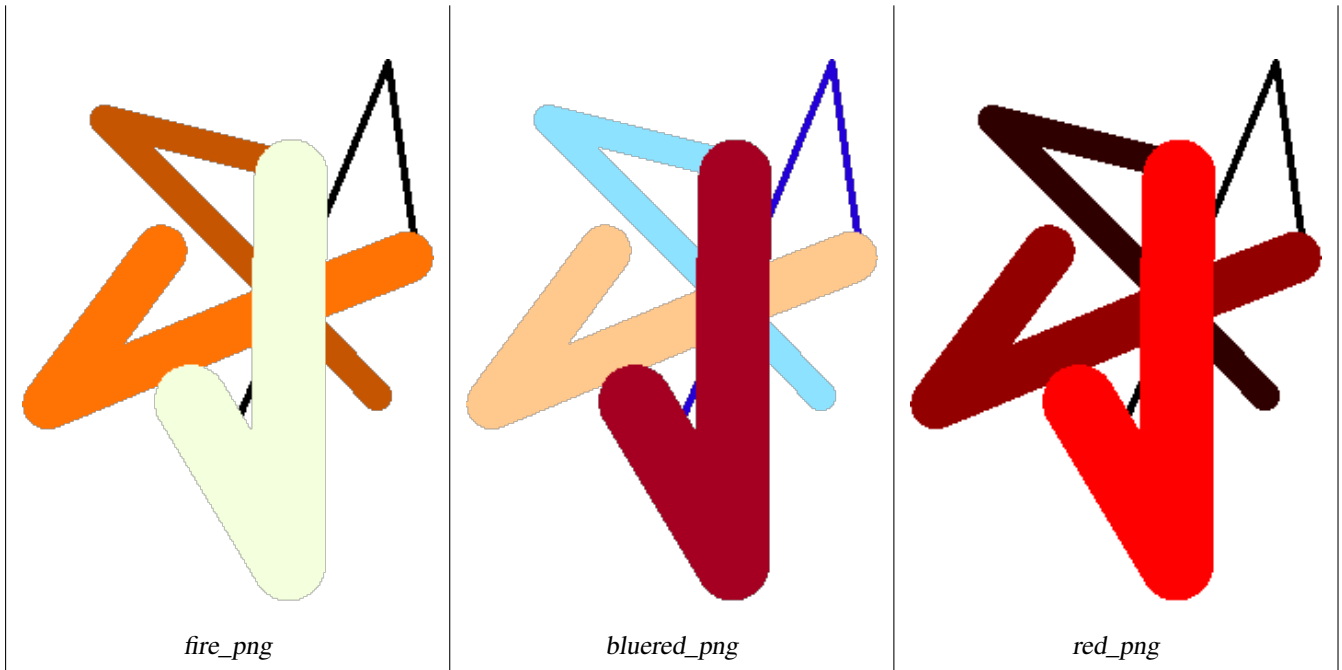
```
SELECT
    ST_NumBands(rast) As n_orig,
    ST_NumBands(ST_ColorMap(rast,1, 'greyscale')) As ngrey,
    ST_NumBands(ST_ColorMap(rast,1, 'pseudocolor')) As npseudo,
    ST_NumBands(ST_ColorMap(rast,1, 'fire')) As nfire,
    ST_NumBands(ST_ColorMap(rast,1, 'bluered')) As nbluered,
    ST_NumBands(ST_ColorMap(rast,1, '
100% 255 0 0
80% 160 0 0
50% 130 0 0
30% 30 0 0
20% 60 0 0
0% 0 0 0
nv 255 255 255
')) As nred
FROM funky_shapes;
```

```
n_orig | ngrey | npseudo | nfire | nbluered | nred
-----+-----+-----+-----+-----+-----
      1 |    1 |    4 |    4 |    4 |    3
```

### Exemplos: Compara cores diferentes no mapa usando ST\_AsPNG

```
SELECT
  ST_AsPNG(rast) As orig_png,
  ST_AsPNG(ST_ColorMap(rast,1,'greyscale')) As grey_png,
  ST_AsPNG(ST_ColorMap(rast,1,'pseudocolor')) As pseudo_png,
  ST_AsPNG(ST_ColorMap(rast,1,'nfire')) As fire_png,
  ST_AsPNG(ST_ColorMap(rast,1,'bluered')) As bluered_png,
  ST_AsPNG(ST_ColorMap(rast,1,'
100% 255 0 0
80% 160 0 0
50% 130 0 0
30% 30 0 0
20% 60 0 0
0% 0 0 0
nv 255 255 255
')) As red_png
FROM funky_shapes;
```





### Veja também

[ST\\_AsPNG](#), [ST\\_AsRaster](#) Funções retorno de mapa algébrico embutido, [ST\\_Grayscale](#) [ST\\_NumBands](#), [ST\\_Reclass](#), [ST\\_SetBandNoDataValue](#), [ST\\_Union](#)

#### 9.12.1.3 ST\_Grayscale

**ST\_Grayscale** — Creates a new one-8BUI band raster from the source raster and specified bands representing Red, Green and Blue

#### 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);

#### Descrição

Create a raster with one 8BUI band given three input bands (from one or more rasters). Any input band whose pixel type is not 8BUI will be reclassified using [ST\\_Reclass](#).



#### Note

This function is not like [ST\\_ColorMap](#) with the `grayscale` keyword as [ST\\_ColorMap](#) operates on only one band while this function expects three bands for RGB. This function applies the following equation for converting RGB to Grayscale:  $0.2989 * RED + 0.5870 * GREEN + 0.1140 * BLUE$

Availability: 2.5.0

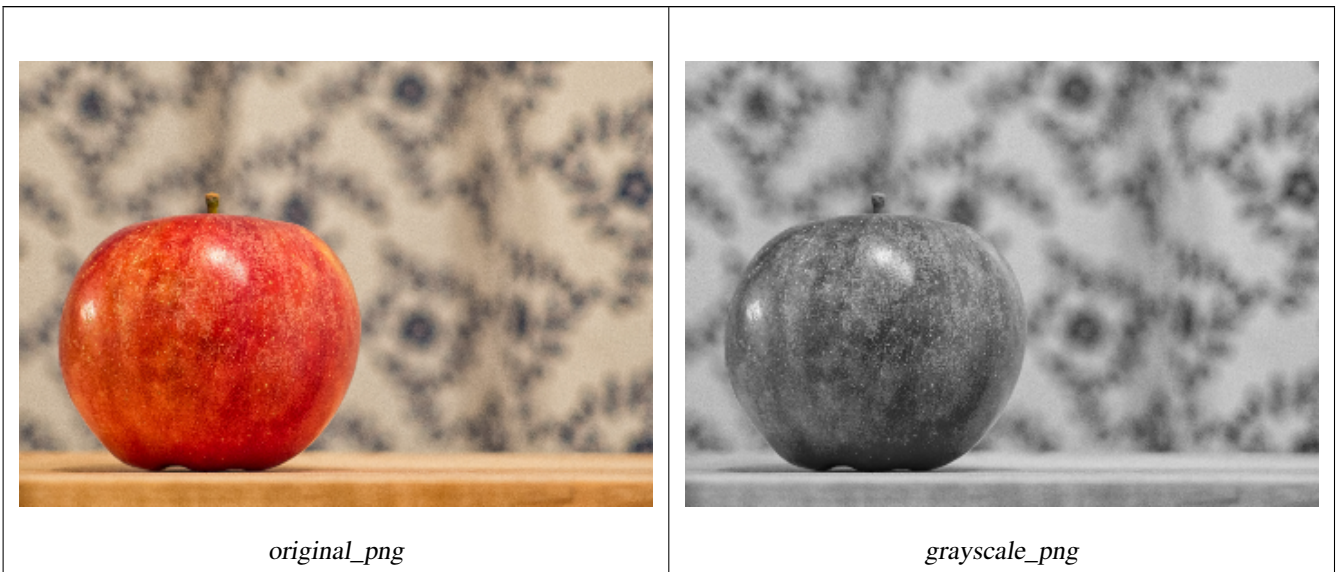
**Exemplos: Variante 1**

```

SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.enable_outdb_rasters = True;

WITH apple AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(350, 246, 0, 0, 1, -1, 0, 0, 0),
        '/tmp/apple.png'::text,
        NULL::int[]
    ) AS rast
)
SELECT
    ST_AsPNG(rast) AS original_png,
    ST_AsPNG(ST_Grayscale(rast)) AS grayscale_png
FROM apple;

```

**Exemplos: Variant 2**

```

SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.enable_outdb_rasters = True;

WITH apple AS (
    SELECT ST_AddBand(
        ST_MakeEmptyRaster(350, 246, 0, 0, 1, -1, 0, 0, 0),
        '/tmp/apple.png'::text,
        NULL::int[]
    ) AS rast
)
SELECT
    ST_AsPNG(rast) AS original_png,
    ST_AsPNG(ST_Grayscale(
        ARRAY[
            ROW(rast, 1)::rastbandarg, -- red
            ROW(rast, 2)::rastbandarg, -- green
            ROW(rast, 3)::rastbandarg, -- blue
        ]::rastbandarg[]
    )) AS grayscale_png
FROM apple;

```

**Veja também**

[ST\\_AsPNG](#), [ST\\_Reclass](#), [ST\\_ColorMap](#)

**9.12.1.4 ST\_Intersection**

**ST\_Intersection** — Retorna uma raster ou conjunto de pares de valores de pixels de geometria representando a porção dividida de dois rasters ou a interseção geométrica de uma vetorização do raster e uma geometria.

**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 geom);
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);
```

**Descrição**

Retorna uma raster ou conjunto de pares de valores de pixels de geometria representando a porção dividida de dois rasters ou a interseção geométrica de uma vetorização do raster e uma geometria.

As primeiras três variantes, retornando um conjunto de geomval, funciona no espaço vetor. Primeiramente, o raster é vetorizado (usando a `ST_DumpAsPolygon`) dentro de linhas geomval e elas intersectam com a geometria usando a função PostGIS `ST_Intersection(geometria, geometria)`. Somente as geometrias intersectando com uma área de valor nodata de um raster, retornam uma geometria vazia. Normalmente, elas são excluídas dos resultados pelo próprio uso da `ST_Intersection` na cláusula `ONDE`.

Você pode acessar a geometria e as partes do valor do conjunto geomval resultante colocando parênteses e adicionando `'geom'` ou `'val'` no fim da expressão. ex.: `(ST_Intersection(rast, geom)).geom`

As outras variantes, retornando um raster, funcionam no espaço raster. Elas estão usando a versão de dois raster da `ST_MapAlgebraExp` para representar a interseção.

A extensão do raster resultante corresponde à interseção geométrica das duas extensões raster. O raster resultante inclui `'BANDA1'`, `'BANDA2'` ou `'AMBAS'` as bandas, a seguir o que é passado como o parâmetro `returnband`. As áreas do valor nodata presentes em qualquer banda resultam áreas de valor nodata em todas as bandas do resultado. Em outras palavras, qualquer pixel intersectando com um pixel de valor nodata se torna um pixel de valor nodata no resultado.

Os rasters resultantes da `ST_Intersection` devem ter um valor nodata designado para áreas que não intersectam. Você pode definir ou substituir o valor nodata para qualquer banda resultante fornecendo um arranjo `nodataval[]` de um ou dois valores nodata, dependendo se solicitou `'BANDA1'`, `'BANDA2'` ou `'AMBAS'` as bandas. O primeiro valor no arranjo substitui o valor nodata na primeira banda e o segundo substitui o valor nodata na segunda banda. Se uma banda de entrada não possuir o valor nodata definido e nenhum for fornecido como arranjo, um é escolhido usando a função `ST_MinPossibleValue`. Todas as variantes que aceitam um arranjo com valor nodata também aceita um único valor que pode ser designado para cada banda pedida.

Em todas as variantes, se nenhum número de banda for especificado, a banda 1 é assumida. Se precisar de uma interseção entre um raster e uma geometria que retorna um raster, recorra a [ST\\_Clip](#).

**Note**

Para ter mais controle na extensão resultante ou no que retorna quando encontra um valor nodata, use a versão de dois raster da [ST\\_MapAlgebraExpr](#).

**Note**

Para calcular a interseção de uma banda raster com uma geometria em um espaço raster, use [ST\\_Clip](#). [ST\\_Clip](#) funciona em várias bandas rasters e não retorna uma banda correspondente para uma geometria rasterizada.

**Note**

A [ST\\_Intersection](#) deveria ser usada em conjunto com a [ST\\_Intersects](#) e um índice na coluna raster e/ou na coluna geométrica.

Melhorias: 2.0.0 - Interseção no espaço raster foi introduzida. Nas versões anteriores pre-2.0.0, somente a interseção apresentada no espaço do vetor era suportada.

**Exemplos: Geometria, Raster -- resultando em geometria vals**

```
SELECT
  foo.rid,
  foo.gid,
  ST_AsText((foo.geomval).geom) As geomwkt,
  (foo.geomval).val
FROM (
  SELECT
    A.rid,
    g.gid,
    ST_Intersection(A.rast, g.geom) As geomval
  FROM dummy_rast AS A
  CROSS JOIN (
    VALUES
      (1, ST_Point(3427928, 5793243.85) ),
      (2, ST_GeomFromText('LINESTRING(3427927.85 5793243.75,3427927.8 5793243.75,3427927.8 5793243.8)')),
      (3, ST_GeomFromText('LINESTRING(1 2, 3 4)'))
  ) As g(gid,geom)
  WHERE A.rid = 2
) As foo;
```

rid	gid	geomwkt	↔	val
2	1	POINT(3427928 5793243.85)	↔	249
2	1	POINT(3427928 5793243.85)	↔	253
2	2	POINT(3427927.85 5793243.75)	↔	254
2	2	POINT(3427927.8 5793243.8)	↔	251
2	2	POINT(3427927.8 5793243.8)	↔	253
2	2	LINESTRING(3427927.8 5793243.75,3427927.8 5793243.8)		252
2	2	MULTILINESTRING((3427927.8 5793243.8,3427927.8 5793243.75),...)		250
2	3	GEOMETRYCOLLECTION EMPTY		

**Veja também**

[geomval](#), [ST\\_Intersects](#), [ST\\_MapAlgebraExpr](#), [ST\\_Clip](#), [ST\\_AsText](#)

### 9.12.1.5 Funções retorno de mapa algébrico embutido

Funções retorno de mapa algébrico embutido — Versão função retorno - Retorna um raster de uma banda dado um ou mais rasters de entrada, os índices e uma função retorno de um usuário específico.

#### 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 user-
args=NULL);
raster ST_MapAlgebra(nband integer, regprocedure callbackfunc, float8[] mask, boolean weighted, text pixeltype=NULL, text
extenttype=INTERSECTION, raster customextent=NULL, text[] VARIADIC userargs=NULL);
```

#### Descrição

Retorna um raster de uma banda dado um ou mais rasters de entrada, os índices e uma função retorno de um usuário específico.

**rast,rast1,rast2, rastbandargset** Rasters onde o processo do mapa algébrico é avaliado.

`rastbandargset` permite o uso de uma operação do mapa algébrico em vários rasters e/ou bandas. Veja o exemplo da Variante 1.

**nband, nband1, nband2** Os números de banda do raster a ser avaliado. `nband` pode ser um inteiro ou inteiro [] indicando as bandas. `nband1` é uma banda no `rast1` e `nband2` é banda no `rast2` para caso hte 2 raster/2band.

**callbackfunc** The `callbackfunc` parameter must be the name and signature of an SQL or PL/pgSQL function, cast to a regprocedure. An example PL/pgSQL function example is:

```
CREATE OR REPLACE FUNCTION sample_callbackfunc(value double precision[][][], position integer[][], VARIADIC userargs text[])
    RETURNS double precision
    AS $$
    BEGIN
        RETURN 0;
    END;
    $$ LANGUAGE 'plpgsql' IMMUTABLE;
```

The `callbackfunc` must have three arguments: a 3-dimension double precision array, a 2-dimension integer array and 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 third argument `userargs` is for passing through any user-specified arguments.

Passing a regprocedure argument to a SQL function requires the full function signature to be passed, then cast to a regprocedure type. To pass the above example PL/pgSQL function as an argument, the SQL for the argument is:

```
'sample_callbackfunc(double precision[], integer[], text[])'::regprocedure
```

Note that the argument contains the name of the function, the types of the function arguments, quotes around the name and argument types, and a cast to a regprocedure.

**mask** An n-dimensional array (matrix) of numbers used to filter what cells get passed to map algebra call-back function. 0 means a neighbor cell value should be treated as no-data and 1 means value should be treated as data. If weight is set to true, then the values, are used as multipliers to multiple the pixel value of that value in the neighborhood position.

**weighted** booleano (verdadeiro/falso) para indicar se o valor da máscara deveria ser pesado (multiplicado pelo valor original) ou não (só se aplica para protocolo que usa máscara).

**pixeltype** Se um `pixeltype` for passado, a banda do novo raster será desse tipo de pixel. Se ele passar NULO ou for deixado, a nova banda raster terá o mesmo tipo de pixel da banda especificada do primeiro raster (para tipos de extensão: INTERSEÇÃO, UNIÃO, PRIMEIRO, CUSTOM) ou da banda específica do raster apropriado (para tipos de extensão: SEGUNDO, ÚLTIMO). Se estiver em dúvida, sempre especifique `pixeltype`.

O tipo de pixel resultante do raster de saída devem ser listados em `ST_BandPixelType` ou deixado de fora ou NULO.

**extenttype** Possíveis valores são INTERSEÇÃO (padrão), UNIÃO, PRIMEIRO (padrão para uma variante raster), SEGUNDO, ÚLTIMO, CUSTOM.

**customextent** Se `extenttype` for CUSTOM, um raster deve ser fornecido para `customextent`. Veja o exemplo 4 de Variante 1.

**distancex** The distance in pixels from the reference cell in x direction. So width of resulting matrix would be  $2 * distancex + 1$ . If not specified only the reference cell is considered (neighborhood of 0).

**distancey** A distância em pixels da célula de referência na direção y. A altura da matriz resultante seria  $2 * distancey + 1$ . Se não especificada, apenas a célula referência é considerada (vizinhança de 0).

**userargs** O terceiro argumento para a `callbackfunc` é um arranjo variadic text. Todos os argumentos de caminho de texto são passados pelo `callbackfunc` especificado, e são contados no argumento `userargs`.



#### Note

Para maiores informações sobre a palavra-chave VARIADIC, por favor recorra à documentação do PostgreSQL e a seção "SQL Functions with Variable Numbers of Arguments" do [Query Language \(SQL\) Functions](#).



#### Note

O argumento `text[]` para a `callbackfunc` é requerido, independente de onde você escolher passar qualquer argumento para a função retorno para processar ou não.

A Variante 1 aceita um arranjo de `rastbandarg` permitindo o uso da operação de mapa algébrico em vários rasters e/ou bandas. Veja o exemplo de Variante 1.

As Variantes 2 e 3 operam em uma ou mais bandas de um raster. Veja os exemplos das Variantes 2 e 3.

A Variante 4 opera em dois raster com uma banda por raster. Veja o exemplo da Variante 4.

Disponibilidade: 2.2.0: Habilidade de adicionar máscara

Disponibilidade: 2.1.0

### Exemplos: Variante 1

Um raster, uma banda

```
WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16 ←
        BUI', 1, 0) AS rast
)
SELECT
    ST_MapAlgebra (
```



```

        ARRAY[ROW(rast, 1)]::rastbandarg[],
        'sample_callbackfunc(double precision[], int[], text[])'::regprocedure
    ) AS rast
FROM foo

```

### Um raster, várias bandas

```

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

```

### Vários rasters, várias bandas

```

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

```

**Exemplo completo de tiles de uma cobertura com vizinhança.** Esta consulta funciona apenas com PostgreSQL 9.1 ou superior.

```

WITH foo AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16 ←
        BUI', 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

```

Exemplo como o anterior, mas funciona com o PostgreSQL 9.0.

```

WITH src AS (
    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16 ←
      BUI', 1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, 0, 1, -1, 0, 0, 0), 1, '16BUI', 2, ←
      0) AS rast UNION ALL
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, 0, 1, -1, 0, 0, 0), 1, '16BUI', 3, ←
      0) AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -2, 1, -1, 0, 0, 0), 1, '16BUI', ←
      10, 0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -2, 1, -1, 0, 0, 0), 1, '16BUI', ←
      20, 0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -2, 1, -1, 0, 0, 0), 1, '16BUI', ←
      30, 0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -4, 1, -1, 0, 0, 0), 1, '16BUI', ←
      100, 0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -4, 1, -1, 0, 0, 0), 1, '16BUI', ←
      200, 0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -4, 1, -1, 0, 0, 0), 1, '16BUI', ←
      300, 0) AS rast
)
WITH foo AS (
    SELECT
        t1.rid,
        ST_Union(t2.rast) AS rast
    FROM src t1
    JOIN src t2
        ON ST_Intersects(t1.rast, t2.rast)
        AND t2.rid BETWEEN 0 AND 8
    WHERE t1.rid = 4
    GROUP BY t1.rid
), bar AS (
    SELECT
        t1.rid,
        ST_MapAlgebra(
            ARRAY[ROW(t2.rast, 1)]::rastbandarg[],
            'raster_nmapalgebra_test(double precision[], int[], text[])':: ←
              regprocedure,
            '32BUI',
            'CUSTOM', t1.rast,
            1, 1
        ) AS rast

```

```

        FROM src t1
        JOIN foo t2
            ON t1.rid = t2.rid
    )
SELECT
    rid,
    (ST_Metadata(rast)),
    (ST_BandMetadata(rast, 1)),
    ST_Value(rast, 1, 1, 1)
FROM bar;

```

### Exemplos: Variantes 2 e 3

#### Um raster, várias bandas

```

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

```

#### Um raster, uma banda

```

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

```

### Exemplos: Variante 4

#### Dois rasters, duas bandas

```

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

```

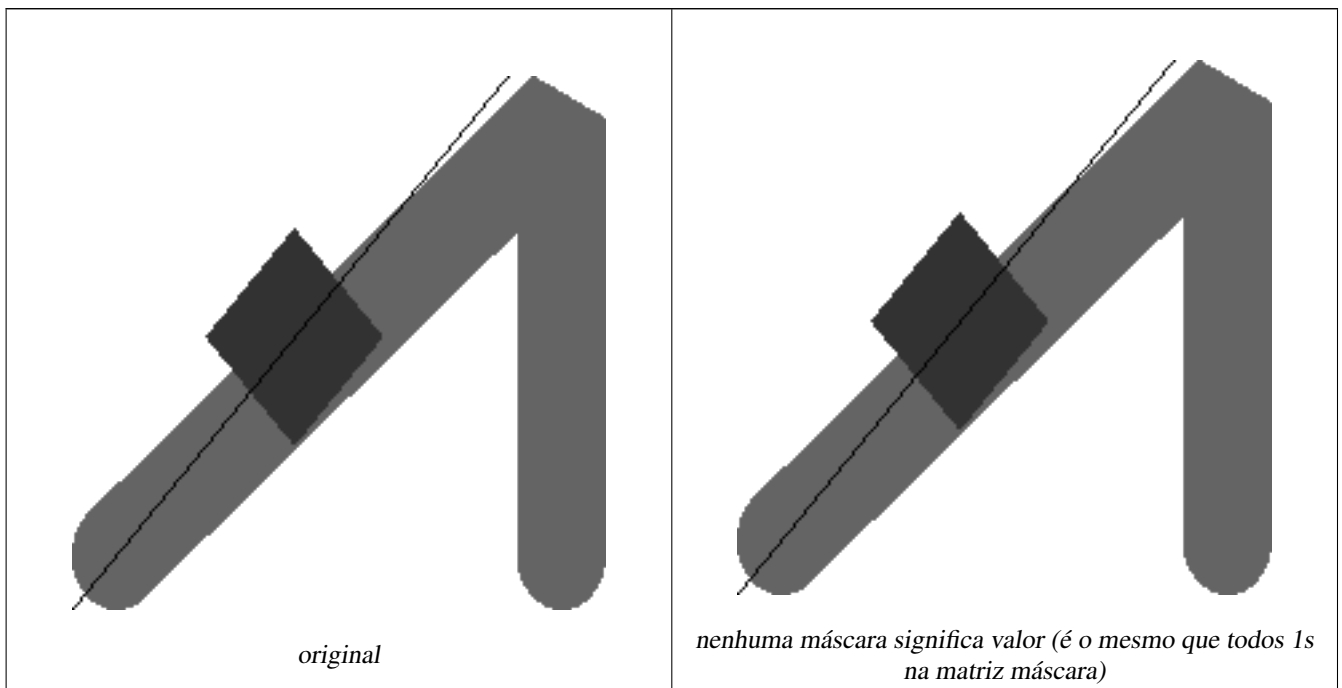
**Exemplos: Utilizando Máscaras**

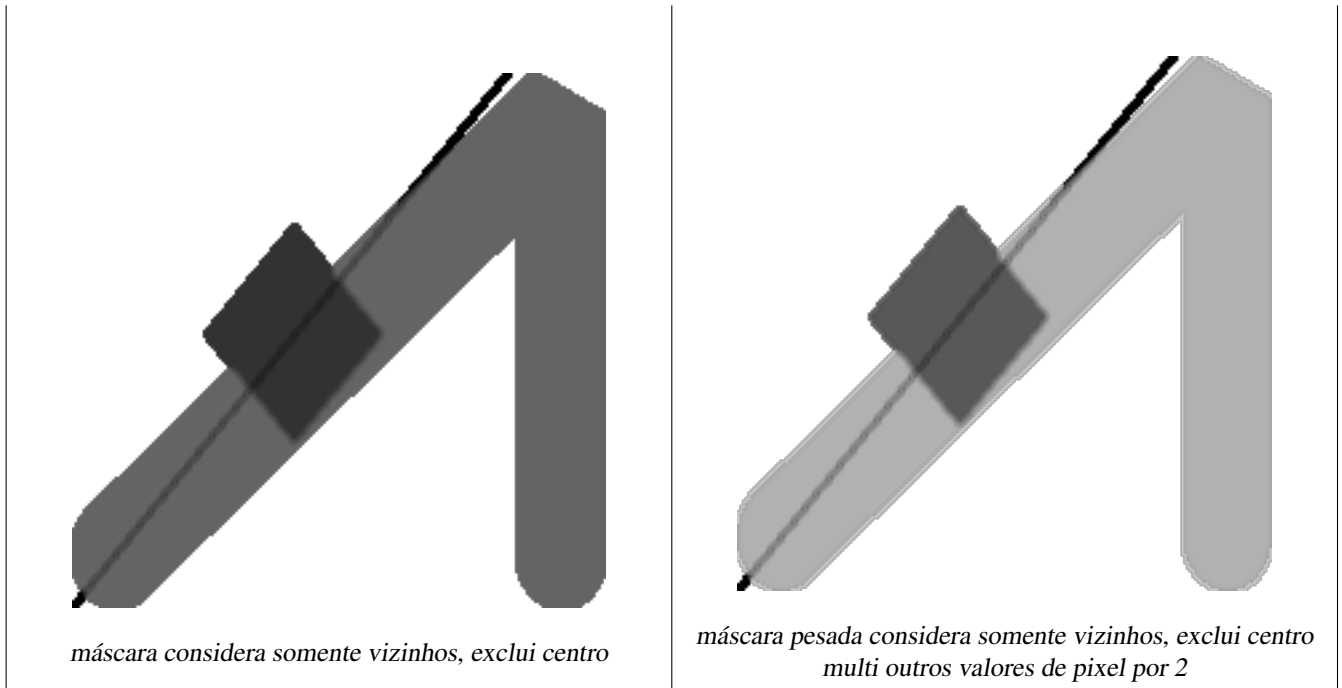
```

WITH foo AS (SELECT
  ST_SetBandNoDataValue(
  ST_SetValue(ST_AsRaster(
    ST_Buffer(
      ST_GeomFromText('LINESTRING(50 50,100 90,100 50)'), 5,'join=bevel') ←
      ,
      200,200,ARRAY['8BUI'], ARRAY[100], ARRAY[0]), ST_Buffer('POINT(70 ←
      70) '::geometry,10,'quad_segs=1') ,50),
    'LINESTRING(20 20, 100 100, 150 98) '::geometry,1),0) AS rast )
SELECT 'original' AS title, rast
FROM foo
UNION ALL
SELECT 'no mask mean value' AS title, ST_MapAlgebra(rast,1,'ST_mean4ma(double precision[], ←
  int[], text[]) '::regprocedure) AS rast
FROM foo
UNION ALL
SELECT 'mask only consider neighbors, exclude center' AS title, ST_MapAlgebra(rast,1,' ←
  ST_mean4ma(double precision[], int[], text[]) '::regprocedure,
  '{{1,1,1}, {1,0,1}, {1,1,1}}' ::double precision[], false) As rast
FROM foo

UNION ALL
SELECT 'mask weighted only consider neighbors, exclude center multi otehr pixel values by ←
  2' AS title, ST_MapAlgebra(rast,1,'ST_mean4ma(double precision[], int[], text[]) ':: ←
  regprocedure,
  '{{2,2,2}, {2,0,2}, {2,2,2}}' ::double precision[], true) As rast
FROM foo;

```





### Veja também

[rastbandarg](#), [ST\\_Union](#), [ST\\_MapAlgebraExpr](#)

#### 9.12.1.6 ST\_MapAlgebraExpr

`ST_MapAlgebraExpr` — Versão expressão - Retorna um raster de uma banda dado um ou mais rasters de entrada, índices de banda e uma ou mais expressões SQL de usuários específicos.

### 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 extnttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);
raster ST_MapAlgebra(raster rast1, raster rast2, text expression, text pixeltype=NULL, text extnttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);
```

### Descrição

Versão expressão - Retorna um raster de uma banda dado um ou mais rasters de entrada, índices de banda e uma ou mais expressões SQL de usuários específicos.

Disponibilidade: 2.1.0

#### Descrição: Variantes 1 e 2 (um raster)

Cria uma nova banda raster formada pela aplicação válida de uma operação algébrica PostgreSQL definida pela `expression` no raster de saída (`rast`). Se `nband` não for dado, a banda 1 é assumida. O novo raster terá a mesma georreferência, largura e altura que o raster original, mas só terá uma banda.

Se um `pixeltype` passar, então o novo raster terá a mesma banda dele. Se o tipo de pixel passar NULO, a nova banda raster terá o mesmo tipo de pixel que a banda de entrada `rast`.

- Palavras-chave permitidas para `expression`

1. `[rast]` - Valor do pixel de interesse
2. `[rast.val]` - Valor do pixel de interesse
3. `[rast.x]` - coluna pixel 1-baseada do pixel de interesse
4. `[rast.y]` - linha pixel 1-baseada do pixel de interesse

### Descrição: Variantes 3 e 4 (dois rasters)

Cria uma nova banda raster formada pela aplicação válida de uma operação algébrica PostgreSQL definida pela `expression` no raster de saída (`rast`). Se `nband`, `band2` não forem especificados, a banda 1 é assumida. O raster resultante será alinhado (escala, inclinação e cantos de pixel) na grade definida pelo primeiro raster. O raster resultante terá de ser definido pelo primeiro raster. O raster resultante terá a extensão definida pelo parâmetro `extenttype`.

**expressão** Uma expressão algébrica PostgreSQL envolvendo dois rasters e funções/operadores PostgreSQL definidos que irão elucidar o valor do pixel quando eles se intersectarem. ex.: `(([rast1] + [rast2])/2.0)::integer`

**pixeltype** O tipo de pixel resultante do raster de saída. Deve ser um listado em `ST_BandPixelType`, deixado de fora ou NULO. Se não passar ou for NULO, usa-se o tipo de pixel do primeiro raster.

**extenttype** Controla a extensão do raster resultante

1. `INTERSECTION` - A extensão do novo raster é a interseção de dois rasters. Este é o padrão.
2. `UNION` - A extensão do novo raster é a união dos dois raster.
3. `FIRST` - A extensão do novo raster é a mesma da do primeiro raster.
4. `SECOND` - A extensão do novo raster é a mesma da do segundo raster.

**nodata1expr** Uma expressão algébrica envolvendo somente `rast2` ou uma constante que define o que retornar quando pixels de `rast1` são valores `nodata` e os pixels `rast2` têm valores.

**nodata2expr** Uma expressão algébrica envolvendo somente `rast1` ou uma constante que define o que retornar quando pixels de `rast2` são valores `nodata` e os pixels `rast1` têm valores.

**nodatanodataval** Uma constante numérica para retornar quando os pixels `rast1` e `rast2` forem ambos valores `nodata`.

- Palavras-chave permitidas em `expression`, `nodata1expr` e `nodata2expr`

1. `[rast1]` - Valor do pixel de interesse do `rast1`
2. `[rast1.val]` - Valor do pixel de interesse do `rast1`
3. `[rast1.x]` - coluna pixel 1-based do pixel de interesse do `rast1`
4. `[rast1.y]` - linha pixel 1-based do pixel de interesse do `rast1`
5. `[rast2]` - Valor do pixel de interesse do `rast2`
6. `[rast2.val]` - Valor do pixel de interesse do `rast2`
7. `[rast2.x]` - coluna pixel 1-based do pixel de interesse do `rast2`
8. `[rast2.y]` - linha pixel 1-based do pixel de interesse do `rast2`

### Exemplos: Variantes 1 e 2

```
WITH foo AS (
    SELECT ST_AddBand(ST_MakeEmptyRaster(10, 10, 0, 0, 1, 1, 0, 0, 0), '32BF'::text, 1, ←
        -1) AS rast
)
SELECT
    ST_MapAlgebra(rast, 1, NULL, 'ceil([rast]*[rast.x]/[rast.y]+[rast.val])')
FROM foo;
```

**Exemplos: Variantes 3 e 4**

```

WITH foo AS (
    SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, ←
        -1, 0, 0, 0), 1, '16BUI', 1, 0), 2, '8BUI', 10, 0), 3, '32BUI'::text, 100, 0) ←
        AS rast UNION ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 1, 1, ←
        -1, 0, 0, 0), 1, '16BUI', 2, 0), 2, '8BUI', 20, 0), 3, '32BUI'::text, 300, 0) ←
        AS rast
)
SELECT
    ST_MapAlgebra(
        t1.rast, 2,
        t2.rast, 1,
        '([rast2] + [rast1.val]) / 2'
    ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 1
    AND t2.rid = 2;

```

**Veja também**

[rastbandarg](#), [ST\\_Union](#), [Funções retorno de mapa algébrico embutido](#)

**9.12.1.7 ST\_MapAlgebraExpr**

**ST\_MapAlgebraExpr** — Versão de banda raster 1: Cria uma nova banda raster formada pela aplicação de ma operação algébrica válida do PostgreSQL na banda raster de entrada de um tipo de pixel fornecido. A banda 1 é assumida se nenhuma banda for especificada.

**Synopsis**

raster **ST\_MapAlgebraExpr**(raster rast, integer band, text pixeltyp, text expression, double precision nodataval=NULL);  
raster **ST\_MapAlgebraExpr**(raster rast, text pixeltyp, text expression, double precision nodataval=NULL);

**Descrição****Warning**

**ST\_MapAlgebraExpr** é menosprezado como do 2.1.0. Use [ST\\_MapAlgebraExpr](#).

Cria uma nova banda raster formada pela aplicação válida de uma operação algébrica PostgreSQL definida pela *expression* no raster de entrada (*rast*). Se *band* não for dado, a banda 1 é assumida. O novo raster terá a mesma georreferência, largura e altura que o raster original, mas só terá uma banda.

Se um *pixeltyp* passar, então o novo raster terá a mesma banda dele. Se o tipo de pixel passar NULO, a nova banda raster terá o mesmo tipo de pixel que a banda de entrada *rast*.

Na expressão você pode usar o termo `[rast]` para referir o valor do pixel da banda original, `[rast.x]` para referir ao índice da coluna pixel 1-baseada, `[rast.y]` para referir ao índice da linha pixel 1-baseada.

Disponibilidade: 2.0.0

## Exemplos

Cria uma nova banda raster 1 a partir da nossa original que é uma função de módulo 2 da banda raster original.

```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
UPDATE dummy_rast SET map_rast = ST_MapAlgebraExpr(rast,NULL,'mod([rast]::numeric,2)') ↔
WHERE rid = 2;
```

```
SELECT
    ST_Value(rast,1,i,j) As origval,
    ST_Value(map_rast, 1, i, j) As mapval
FROM dummy_rast
CROSS JOIN generate_series(1, 3) AS i
CROSS JOIN generate_series(1,3) AS j
WHERE rid = 2;
```

origval	mapval
253	1
254	0
253	1
253	1
254	0
254	0
250	0
254	0
254	0

Cria uma nova banda raster 1 do tipo de pixel 2BUI a partir da nossa original que é reclassificada e obtém valor nodata 0.

```
ALTER TABLE dummy_rast ADD COLUMN map_rast2 raster;
UPDATE dummy_rast SET
    map_rast2 = ST_MapAlgebraExpr(rast,'2BUI'::text,'CASE WHEN [rast] BETWEEN 100 and ↔
    250 THEN 1 WHEN [rast] = 252 THEN 2 WHEN [rast] BETWEEN 253 and 254 THEN 3 ELSE ↔
    0 END'::text, '0')
WHERE rid = 2;
```

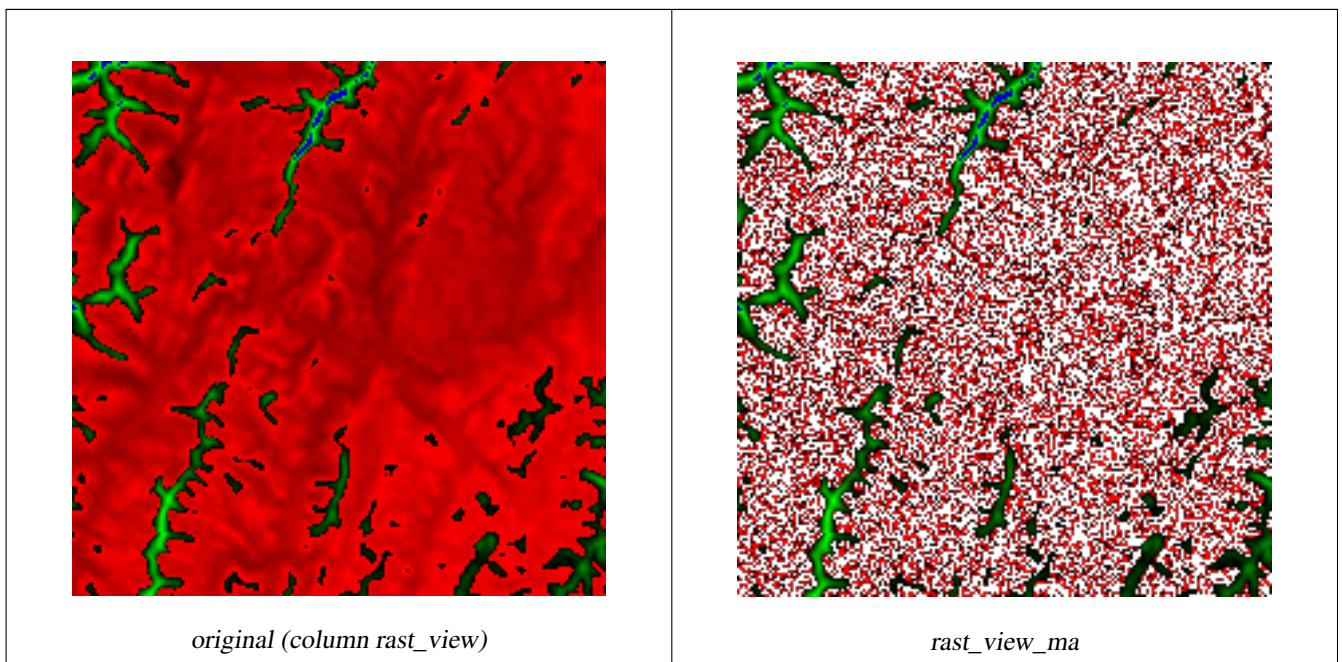
```
SELECT DISTINCT
    ST_Value(rast,1,i,j) As origval,
    ST_Value(map_rast2, 1, i, j) As mapval
FROM dummy_rast
CROSS JOIN generate_series(1, 5) AS i
CROSS JOIN generate_series(1,5) AS j
WHERE rid = 2;
```

origval	mapval
249	1
250	1
251	
252	2
253	3
254	3

```
SELECT
    ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast
WHERE rid = 2;
```

b1pixtyp
2BUI





Cria uma nova banda raster 3 do mesmo tipo de pixel da nossa banda 3 original, com a primeira banda alterada pelo mapa algébrico e 2 bandas permanecem inalteradas.

```
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;
```

### Veja também

[ST\\_MapAlgebraExpr](#), [ST\\_MapAlgebraFct](#), [ST\\_BandPixelType](#), [ST\\_GeoReference](#), [ST\\_Value](#)

#### 9.12.1.8 ST\_MapAlgebraExpr

**ST\_MapAlgebraExpr** — Versão de banda raster 2: Cria uma banda raster nova formada pela aplicação de uma operação algébrica válida PostgreSQL nas duas bandas raster de entrada e do tipo de pixel fornecido. A banda 1 de cada raster é assumida se nenhum número de bandas for especificado. O raster resultante será alinhado (escala, inclinação e cantos de pixel) na grade definida pelo primeiro raster e tem sua extensão definida pelo parâmetro "extenttype". O valores para "extenttype" pode ser: INTERSEÇÃO, UNIÃO, PRIMEIRO, SEGUNDO.

### 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);
```

## Descrição



### Warning

`ST_MapAlgebraExpr` é menosprezado como do 2.1.0. Use `ST_MapAlgebraExpr`.

Cria uma nova banda raster formada pela aplicação válida de uma operação algébrica PostgreSQL definida pela `expression` no raster de saída (`rast`). Se `nband`, `band2` não forem especificados, a banda 1 é assumida. O raster resultante será alinhado (escala, inclinação e cantos de pixel) na grade definida pelo primeiro raster. O raster resultante terá de ser definido pelo primeiro raster. O raster resultante terá a extensão definida pelo parâmetro `extenttype`.

**expressão** Uma expressão algébrica PostgreSQL envolvendo dois rasters e funções/operadores PostgreSQL definidos que irão elucidar o valor do pixel quando eles se intersectarem. ex.: `(([rast1] + [rast2])/2.0)::integer`

**pixeltype** O tipo de pixel resultante do raster de saída. Deve ser um listado em `ST_BandPixelType`, deixado de fora ou NULO. Se não passar ou for NULO, usa-se o tipo de pixel do primeiro raster.

**extenttype** Controla a extensão do raster resultante

1. INTERSECTION - A extensão do novo raster é a interseção de dois rasters. Este é o padrão.
2. UNION - A extensão do novo raster é a união dos dois raster.
3. FIRST - A extensão do novo raster é a mesma da do primeiro raster.
4. SECOND - A extensão do novo raster é a mesma da do segundo raster.

**nodata1expr** Uma expressão algébrica envolvendo somente `rast2` ou uma constante que define o que retornar quando pixels de `rast1` são valores nodata e os pixels `rast2` têm valores.

**nodata2expr** Uma expressão algébrica envolvendo somente `rast1` ou uma constante que define o que retornar quando pixels de `rast2` são valores nodata e os pixels `rast1` têm valores.

**nodatanodataval** Uma constante numérica para retornar quando os pixels `rast1` e `rast2` forem ambos valores nodata.

Se `pixeltype` passar, o novo raster terá uma banda desse tipo de pixel. Se o tipo de pixel passar NULO ou nenhum tipo for especificado, a nova banda raster terá o mesmo tipo de pixel da banda de entrada `rast1`.

Use o termo `[rast1.val] [rast2.val]` para referir-se ao valor de pixel das bandas rasters originais e `[rast1.x]`, `[rast1.y]` etc. para referir-se à posição da coluna/linha dos pixels.

Disponibilidade: 2.0.0

### Exemplo: 2 Interseção de Banda e União

Cria uma nova banda raster 1 a partir da nossa original que é uma função de módulo 2 da banda raster original.

```
--Create a cool set of rasters --
DROP TABLE IF EXISTS fun_shapes;
CREATE TABLE fun_shapes(rid serial PRIMARY KEY, fun_name text, rast raster);

-- Insert some cool shapes around Boston in Massachusetts state plane meters --
INSERT INTO fun_shapes(fun_name, rast)
VALUES ('ref', ST_AsRaster(ST_MakeEnvelope(235229, 899970, 237229, 901930,26986),200,200,'8 ←
    BUI',0,0));

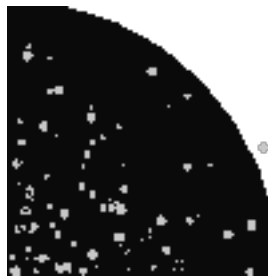
INSERT INTO fun_shapes(fun_name,rast)
WITH ref(rast) AS (SELECT rast FROM fun_shapes WHERE fun_name = 'ref' )
SELECT 'area' AS fun_name, ST_AsRaster(ST_Buffer(ST_SetSRID(ST_Point(236229, 900930),26986) ←
    , 1000),
    ref.rast,'8BUI', 10, 0) As rast
```

```

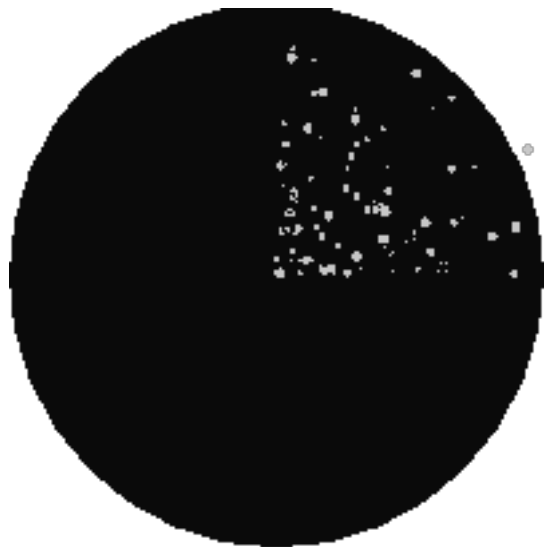
FROM ref
UNION ALL
SELECT 'rand bubbles',
      ST_AsRaster(
        (SELECT ST_Collect(geom)
         FROM (SELECT ST_Buffer(ST_SetSRID(ST_Point(236229 + i*random()*100, 900930 + j*
           random()*100),26986), random()*20) As geom
          FROM generate_series(1,10) As i, generate_series(1,10) As j
           ) As foo ), ref.rast,'8BUI', 200, 0)
FROM ref;

--map them -
SELECT ST_MapAlgebraExpr(
  area.rast, bub.rast, '[rast2.val]', '8BUI', 'INTERSECTION', '[rast2.val]', ←
  '[rast1.val]') As interrast,
  ST_MapAlgebraExpr(
  area.rast, bub.rast, '[rast2.val]', '8BUI', 'UNION', '[rast2.val]', ←
  '[rast1.val]') As unionrast
FROM
  (SELECT rast FROM fun_shapes WHERE
   fun_name = 'area') As area
CROSS JOIN (SELECT rast
FROM fun_shapes WHERE
fun_name = 'rand bubbles') As bub

```



*interseção de mapa algébrico*



*união de mapa algébrico*

### Exemplo: Revestindo rasters em um quadro como bandas separadas

```

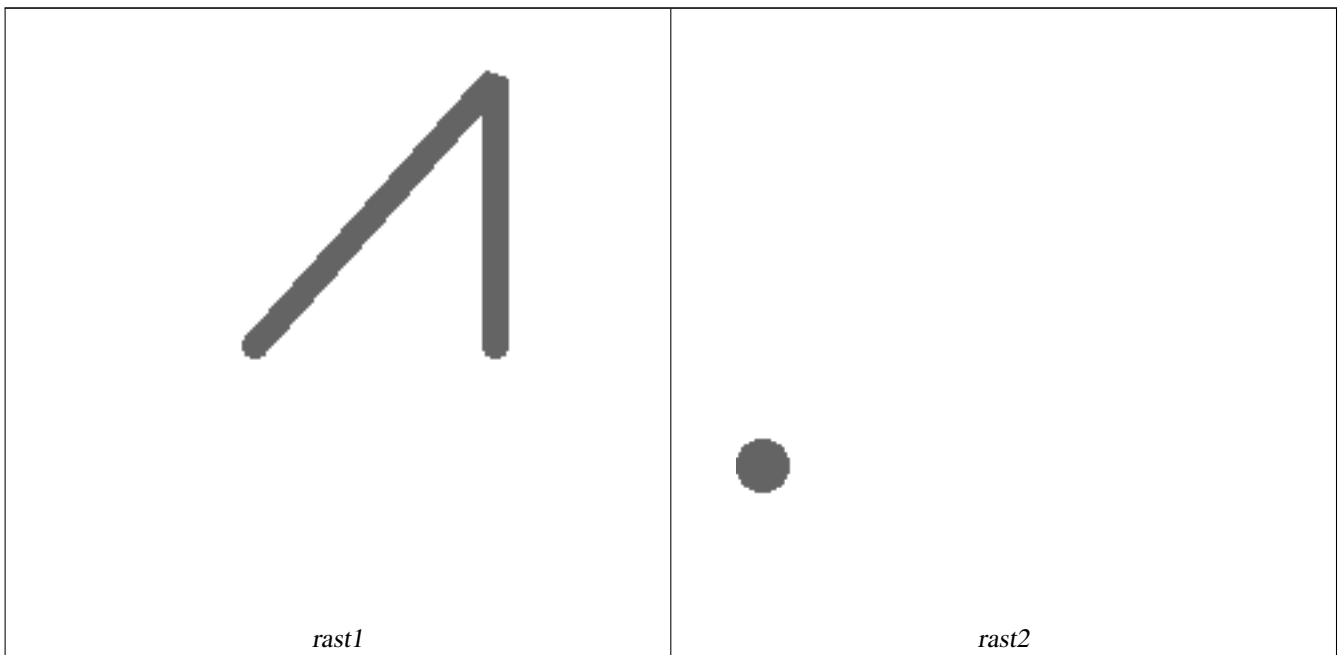
-- we use ST_AsPNG to render the image so all single band ones look grey --
WITH mygeoms
  AS ( SELECT 2 As bnum, ST_Buffer(ST_Point(1,5),10) As geom
      UNION ALL
      SELECT 3 AS bnum,
        ST_Buffer(ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'), 10,'join= ←
        bevel') As geom
      UNION ALL
      SELECT 1 As bnum,

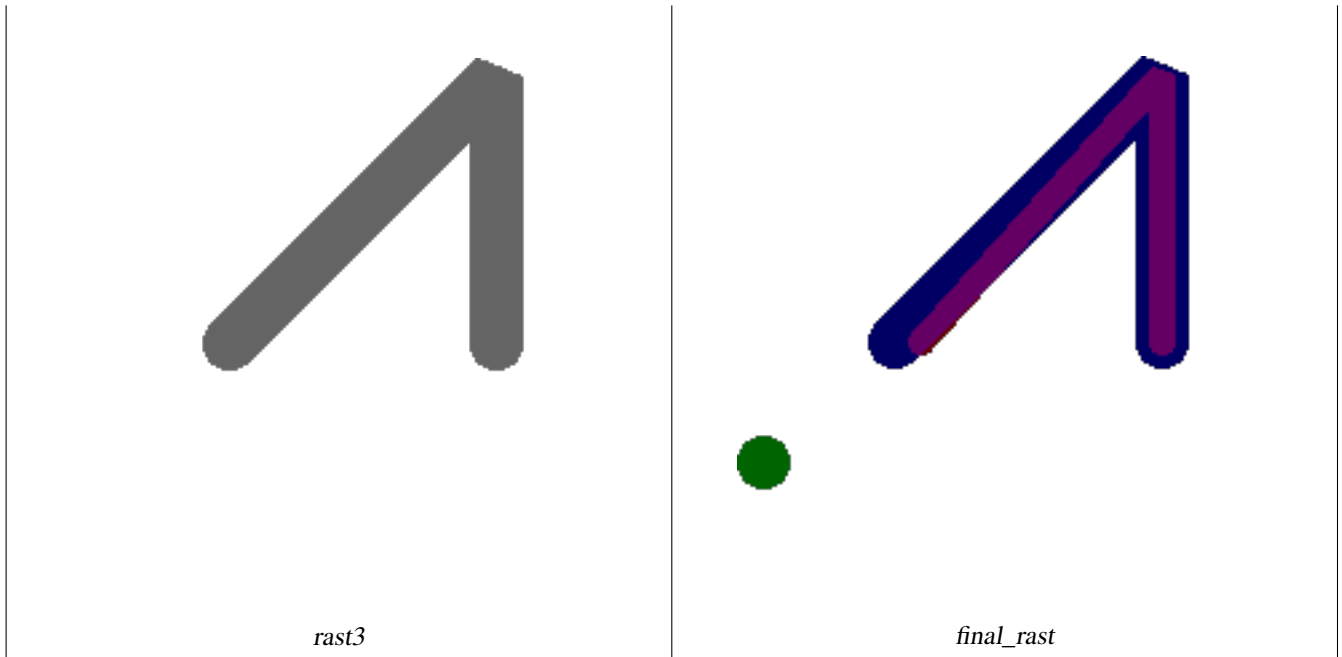
```

```

        ST_Buffer(ST_GeomFromText('LINESTRING(60 50,150 150,150 50)'), 5,'join= ↵
        bevel') As geom
    ),
-- define our canvas to be 1 to 1 pixel to geometry
canvas
AS (SELECT ST_AddBand(ST_MakeEmptyRaster(200,
    200,
    ST_XMin(e)::integer, ST_YMax(e)::integer, 1, -1, 0, 0) , '8BUI'::text,0) As rast
FROM (SELECT ST_Extent(geom) As e,
    Max(ST_SRID(geom)) As srid
    from mygeoms
    ) As foo
    ),
rbands AS (SELECT ARRAY(SELECT ST_MapAlgebraExpr(canvas.rast, ST_AsRaster(m.geom, canvas ↵
.rast, '8BUI', 100),
    '[rast2.val]', '8BUI', 'FIRST', '[rast2.val]', '[rast1.val]') As rast
FROM mygeoms AS m CROSS JOIN canvas
ORDER BY m.bnum) As rasts
    )
SELECT rasts[1] As rast1 , rasts[2] As rast2, rasts[3] As rast3, ST_AddBand(
    ST_AddBand(rasts[1],rasts[2]), rasts[3]) As final_rast
FROM rbands;

```





### Exemplo: Cobre 2 metros de limite das parcelas seleccionadas sobre uma área imaginária

```

-- Create new 3 band raster composed of first 2 clipped bands, and overlay of 3rd band with ←
  our geometry
-- This query took 3.6 seconds on PostGIS windows 64-bit install
WITH pr AS
-- Note the order of operation: we clip all the rasters to dimensions of our region
(SELECT ST_Clip(rast,ST_Expand(geom,50) ) As rast, g.geom
  FROM aerials.o_2_boston AS r INNER JOIN
-- union our parcels of interest so they form a single geometry we can later intersect with
  (SELECT ST_Union(ST_Transform(the_geom,26986)) AS geom
    FROM landparcels WHERE pid IN('0303890000', '0303900000')) As g
  ON ST_Intersects(rast::geometry, ST_Expand(g.geom,50))
),
-- we then union the raster shards together
-- ST_Union on raster is kinda of slow but much faster the smaller you can get the rasters
-- therefore we want to clip first and then union
prunion AS
(SELECT ST_AddBand(NULL, ARRAY[ST_Union(rast,1),ST_Union(rast,2),ST_Union(rast,3)] ) As ←
  clipped,geom
  FROM pr
  GROUP BY geom)
-- return our final raster which is the unioned shard with
-- with the overlay of our parcel boundaries
-- add first 2 bands, then mapalgebra of 3rd band + geometry
SELECT ST_AddBand(ST_Band(clipped,ARRAY[1,2])
  , ST_MapAlgebraExpr(ST_Band(clipped,3), ST_AsRaster(ST_Buffer(ST_Boundary(geom),2), ←
  clipped, '8BUI',250),
  '[rast2.val]', '8BUI', 'FIRST', '[rast2.val]', '[rast1.val]')) ) As rast
FROM prunion;

```



*As linhas azuis são os limites das parcelas seleccionadas*

#### Veja também

[ST\\_MapAlgebraExpr](#), [ST\\_AddBand](#), [ST\\_AsPNG](#), [ST\\_AsRaster](#), [ST\\_MapAlgebraFct](#), [ST\\_BandPixelType](#), [ST\\_GeoReference](#), [ST\\_Value](#), [ST\\_Union](#), [ST\\_Union](#)

#### 9.12.1.9 ST\_MapAlgebraFct

`ST_MapAlgebraFct` — Versão de banda raster 1: Cria uma nova banda raster formada pela aplicação de uma função válida do PostgreSQL na banda raster de entrada de um tipo de pixel fornecido. A banda 1 é assumida se nenhuma banda for especificada.

#### 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);
```

#### Descrição



#### Warning

`ST_MapAlgebraFct` é menosprezado como do 2.1.0. Use [Funções retorno de mapa algébrico embutido](#).

Cria uma nova banda raster formada pela aplicação válida de uma função PostgreSQL definida pela `onerasteruserfunc` no raster de entrada (`rast`). Se `band` não for dado, a banda 1 é assumida. O novo raster terá a mesma georreferência, largura e altura que o raster original, mas só terá uma banda.

Se um `pixeltype` passar, então o novo raster terá a mesma banda dele. Se o tipo de pixel passar NULO, a nova banda raster terá o mesmo tipo de pixel que a banda de entrada `rast`.

The `onerasteruserfunc` parameter must be the name and signature of a SQL or PL/pgSQL function, cast to a regprocedure. A very simple and quite useless PL/pgSQL function example is:

```
CREATE OR REPLACE FUNCTION simple_function(pixel FLOAT, pos INTEGER[], VARIADIC args TEXT ←
  [])
  RETURNS FLOAT
  AS $$ BEGIN
    RETURN 0.0;
  END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

The `userfunction` may accept two or three arguments: a float value, an optional integer array, and a variadic text array. The first argument is the value of an individual raster cell (regardless of the raster datatype). The second argument is the position of the current processing cell in the form '{x,y}'. The third argument indicates that all remaining parameters to `ST_MapAlgebraFct` shall be passed through to the `userfunction`.

Passing a regprocedure argument to a SQL function requires the full function signature to be passed, then cast to a regprocedure type. To pass the above example PL/pgSQL function as an argument, the SQL for the argument is:

```
'simple_function(float,integer[],text[])'::regprocedure
```

Note that the argument contains the name of the function, the types of the function arguments, quotes around the name and argument types, and a cast to a regprocedure.

O terceiro argumento para a `userfunction` é um variadic text arranjo. Todos os argumentos seguindo qualquer chamada `ST_MapAlgebraFct` passam pela `userfunction` especificada, e são contidos no argumento `args`.



#### Note

Para maiores informações sobre a palavra-chave VARIADIC, por favor recorra à documentação do PostgreSQL e a seção "SQL Functions with Variable Numbers of Arguments" do [Query Language \(SQL\) Functions](#).



#### Note

O argumento `text[]` para o `userfunction` é requerido, independente se escolher passar argumentos para sua função usuário processar ou não.

Disponibilidade: 2.0.0

## Exemplos

Cria uma nova banda raster 1 a partir da nossa original que é uma função de módulo 2 da banda raster original.

```
ALTER TABLE dummy_rast ADD COLUMN map_rast raster;
CREATE FUNCTION mod_fct(pixel float, pos integer[], variadic args text[])
  RETURNS float
  AS $$
  BEGIN
    RETURN pixel::integer % 2;
  END;
  $$
```

```
LANGUAGE 'plpgsql' IMMUTABLE;

UPDATE dummy_rast SET map_rast = ST_MapAlgebraFct(rast,NULL,'mod_fct(float,integer[],text ←
  [])'::regprocedure) WHERE rid = 2;

SELECT ST_Value(rast,1,i,j) As origval, ST_Value(map_rast, 1, i, j) As mapval
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1,3) AS j
WHERE rid = 2;
```

origval	mapval
253	1
254	0
253	1
253	1
254	0
254	0
250	0
254	0
254	0

Cria uma nova banda raster 1 de tipo pixel 2BUI da original que é reclassificada e adquire valor sem dados para uma parâmetro passado à função usuário (0).

```
ALTER TABLE dummy_rast ADD COLUMN map_rast2 raster;
CREATE FUNCTION classify_fct(pixel float, pos integer[], variadic args text[])
RETURNS float
AS
$$
DECLARE
  nodata float := 0;
BEGIN
  IF NOT args[1] IS NULL THEN
    nodata := args[1];
  END IF;
  IF pixel < 251 THEN
    RETURN 1;
  ELSIF pixel = 252 THEN
    RETURN 2;
  ELSIF pixel > 252 THEN
    RETURN 3;
  ELSE
    RETURN nodata;
  END IF;
END;
$$
LANGUAGE 'plpgsql';
UPDATE dummy_rast SET map_rast2 = ST_MapAlgebraFct(rast,'2BUI','classify_fct(float,integer ←
  [],text[])'::regprocedure, '0') WHERE rid = 2;

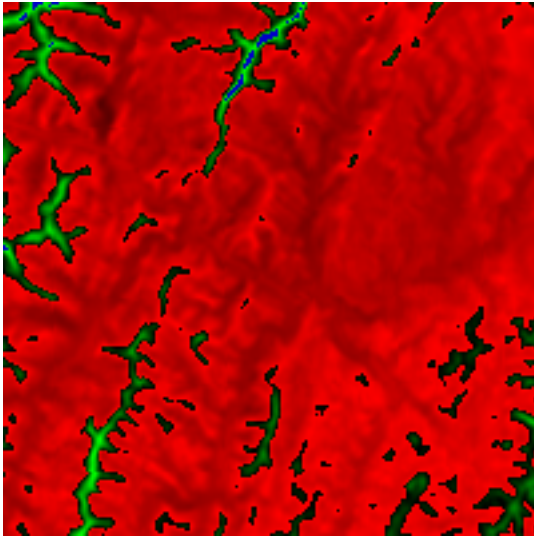
SELECT DISTINCT ST_Value(rast,1,i,j) As origval, ST_Value(map_rast2, 1, i, j) As mapval
FROM dummy_rast CROSS JOIN generate_series(1, 5) AS i CROSS JOIN generate_series(1,5) AS j
WHERE rid = 2;
```

origval	mapval
249	1
250	1
251	
252	2
253	3
254	3

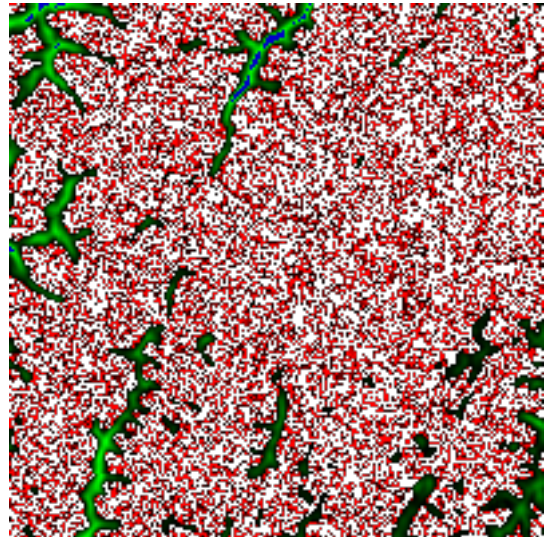


```
SELECT ST_BandPixelType(map_rast2) As b1pixtyp
FROM dummy_rast WHERE rid = 2;
```

```
b1pixtyp
-----
2BUI
```



*original (column rast-view)*



*rast\_view\_ma*

Cria uma nova banda raster 3 do mesmo tipo de pixel da nossa banda 3 original, com a primeira banda alterada pelo mapa algébrico e 2 bandas permanecem inalteradas.

```
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;
```

**Veja também**

[ST\\_MapAlgebraExpr](#), [ST\\_BandPixelType](#), [ST\\_GeoReference](#), [ST\\_SetValue](#)

**9.12.1.10 ST\_MapAlgebraFct**

**ST\_MapAlgebraFct** — Versão de banda 2 - Cria uma nova banda raster um formada pela aplicação de uma função PostgreSQL na 2 entrada de bandas raster e do tipo de pixel fornecido. A banda 1 é assumida se nenhuma banda for especificada. Tipo de extensão torna-se INTERSEÇÃO se não especificada.

**Synopsis**

raster **ST\_MapAlgebraFct**(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltypename=same\_as\_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

raster **ST\_MapAlgebraFct**(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltypename=same\_as\_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

**Descrição****Warning**

**ST\_MapAlgebraFct** é menosprezado como do 2.1.0. Use [Funções retorno de mapa algébrico embutido](#).

Cria uma nova banda raster um formada pela aplicação válida de uma função PostgreSQL definida pela `tworastuserfunc` no raster de entrada `rast1`, `rast1`. Se `band1` ou `band2` não forem especificadas, a banda 1 é assumida. O novo raster terá a mesma georreferência, largura e altura que o raster original, mas só terá uma banda.

Se um `pixeltypename` passar, então o novo raster terá a mesma banda dele. Se o tipo de pixel passar NULO, a nova banda raster terá o mesmo tipo de pixel que a banda de entrada `rast1`.

The `tworastuserfunc` parameter must be the name and signature of an SQL or PL/pgSQL function, cast to a regprocedure. An example PL/pgSQL function example is:

```
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;
```

The `tworastuserfunc` may accept three or four arguments: a double precision value, a double precision value, an optional integer array, and a variadic text array. The first argument is the value of an individual raster cell in `rast1` (regardless of the raster datatype). The second argument is an individual raster cell value in `rast2`. The third argument is the position of the current processing cell in the form '{x,y}'. The fourth argument indicates that all remaining parameters to **ST\_MapAlgebraFct** shall be passed through to the `tworastuserfunc`.

Passing a regprocedure argument to a SQL function requires the full function signature to be passed, then cast to a regprocedure type. To pass the above example PL/pgSQL function as an argument, the SQL for the argument is:

```
'simple_function(double precision, double precision, integer[], text[])::regprocedure
```

Note that the argument contains the name of the function, the types of the function arguments, quotes around the name and argument types, and a cast to a regprocedure.

The fourth argument to the `tworastuserfunc` is a variadic text array. All trailing text arguments to any **ST\_MapAlgebraFct** call are passed through to the specified `tworastuserfunc`, and are contained in the `userargs` argument.

**Note**

Para maiores informações sobre a palavra-chave VARIADIC, por favor recorra à documentação do PostgreSQL e a seção "SQL Functions with Variable Numbers of Arguments" do [Query Language \(SQL\) Functions](#).

**Note**

O argumento `text[]` para a `tworastuserfunc` é requerido, independente se escolher passar argumentos para sua função usuário processar ou não.

Disponibilidade: 2.0.0

**Exemplo: Revestindo rasters em um quadro como bandas separadas**

```
-- define our user defined function --
CREATE OR REPLACE FUNCTION raster_mapalgebra_union(
    rast1 double precision,
    rast2 double precision,
    pos integer[],
    VARIADIC userargs text[]
)
    RETURNS double precision
    AS $$
    DECLARE
    BEGIN
        CASE
            WHEN rast1 IS NOT NULL AND rast2 IS NOT NULL THEN
                RETURN ((rast1 + rast2)/2.);
            WHEN rast1 IS NULL AND rast2 IS NULL THEN
                RETURN NULL;
            WHEN rast1 IS NULL THEN
                RETURN rast2;
            ELSE
                RETURN rast1;
        END CASE;

        RETURN NULL;
    END;
    $$ LANGUAGE 'plpgsql' IMMUTABLE COST 1000;

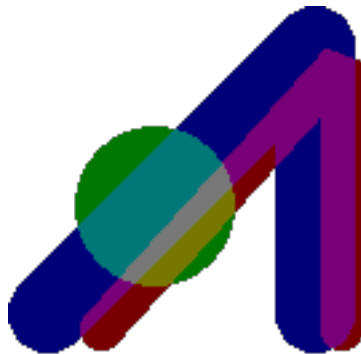
-- prep our test table of rasters
DROP TABLE IF EXISTS map_shapes;
CREATE TABLE map_shapes(rid serial PRIMARY KEY, rast raster, bnum integer, descrip text);
INSERT INTO map_shapes(rast,bnum, descrip)
WITH mygeoms
    AS ( SELECT 2 As bnum, ST_Buffer(ST_Point(90,90),30) As geom, 'circle' As descrip
        UNION ALL
        SELECT 3 AS bnum,
            ST_Buffer(ST_GeomFromText('LINESTRING(50 50,150 150,150 50)'), 15) As geom, ←
            'big road' As descrip
        UNION ALL
        SELECT 1 As bnum,
            ST_Translate(ST_Buffer(ST_GeomFromText('LINESTRING(60 50,150 150,150 50)'), ←
            8,'join=bevel'), 10,-6) As geom, 'small road' As descrip
    ),
-- define our canvas to be 1 to 1 pixel to geometry
canvas
    AS ( SELECT ST_AddBand(ST_MakeEmptyRaster(250,
```

```

250,
ST_XMin(e)::integer, ST_YMax(e)::integer, 1, -1, 0, 0 ) , '8BUI'::text,0) As rast
FROM (SELECT ST_Extent(geom) As e,
      Max(ST_SRID(geom)) As srid
      from mygeoms
      ) As foo
)
-- return our rasters aligned with our canvas
SELECT ST_AsRaster(m.geom, canvas.rast, '8BUI', 240) As rast, bnum, descrip
      FROM mygeoms AS m CROSS JOIN canvas
UNION ALL
SELECT canvas.rast, 4, 'canvas'
FROM canvas;

-- Map algebra on single band rasters and then collect with ST_AddBand
INSERT INTO map_shapes(rast,bnum,descrip)
SELECT ST_AddBand(ST_AddBand(rasts[1], rasts[2]),rasts[3]), 4, 'map bands overlay fct union ←
      (canvas) '
      FROM (SELECT ARRAY(SELECT ST_MapAlgebraFct(m1.rast, m2.rast,
          'raster_mapalgebra_union(double precision, double precision, ←
          integer[], text[])'::regprocedure, '8BUI', 'FIRST')
          FROM map_shapes As m1 CROSS JOIN map_shapes As m2
          WHERE m1.descrip = 'canvas' AND m2.descrip <> 'canvas' ORDER BY m2.bnum) As rasts) ←
      As foo;

```



*bandas mapa cobrem (quadro) (R: rua pequena, G: círculo, B: rua grande)*

### Função de usuário definido que toma argumentos extras

```

CREATE OR REPLACE FUNCTION raster_mapalgebra_userargs (
  rast1 double precision,
  rast2 double precision,
  pos integer[],
  VARIADIC userargs text[]

```

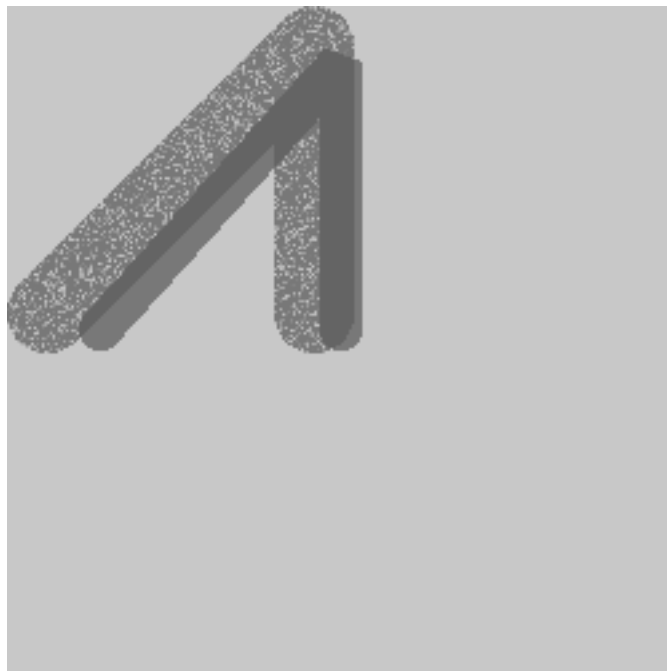
```

)
    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)';

```



*usuário definido com argumentos extras e bandas diferentes do mesmo raster*

#### **Veja também**

[ST\\_MapAlgebraExpr](#), [ST\\_BandPixelType](#), [ST\\_GeoReference](#), [ST\\_SetValue](#)

### 9.12.1.11 ST\_MapAlgebraFctNgb

ST\_MapAlgebraFctNgb — Versão 1-banda: o vizinho mais próximo no mapa algébrico usando a função de usuário definido PostgreSQL. Retorna um raster cujos valores são o resultado de uma função usuário PLPGSQL envolvendo uma vizinhança de valores da banda raster de entrada.

#### Synopsis

raster **ST\_MapAlgebraFctNgb**(raster rast, integer band, text pixeltype, integer ngbwidth, integer ngbheight, regprocedure onerastngbuserfunc, text nodatamode, text[] VARIADIC args);

#### Descrição



#### Warning

ST\_MapAlgebraFctNgb é menosprezado como do 2.1.0. Use [Funções retorno de mapa algébrico embutido](#).

(versão raster um) Retorna um raster cujos valores são o resultado de uma função usuário PLPGSQL envolvendo uma vizinhança de valores da banda raster de entrada. A função usuário toma a vizinhança de valores de pixel como um arranjo de números, para cada pixel, retorna o resultado da função usuário, substituindo o valor do pixel, inspecionado no momento, pelo resultado da função.

**rast** Raster no qual a função usuário é avaliada.

**banda** Número de banda do raster a ser avaliado. Padrão é 1.

**pixeltype** O tipo de pixel resultante do raster de saída. Deve estar listado em [ST\\_BandPixelType](#) ou ser deixado de fora ou ser NULO. Se não passar ou não for NULO, o padrão será o tipo de pixel do rast. Os resultados são cortados se eles forem maiores que o permitido para o tipo de pixel.

**ngbwidth** A largura da vizinhança, nas células.

**ngbheight** A altura da vizinhança, nas células.

**onerastngbuserfunc** A função usuário PLPGSQL/psq para aplicar uma vizinhança de pixels de uma única banda de um raster. O primeiro elemento é um arranjo 2-dimensional de números representando a vizinhança do pixel retangular

**nodatamode** Define qual valor passar para a função para uma vizinhança de pixel que é sem dados ou NULA

'ignore': quaisquer valores NODATA encontrados na vizinhança são ignorados pelo cálculo -- esta bandeira deve ser enviada para o retorno da função usuário, e ela decide como ignorar.

'NULL': quaisquer valores NODATA encontrados na vizinhança acusamos o pixel de ser NULL -- neste caso, o retorno da função usuário é pulado.

'value': quaisquer valores NODATA encontrados na vizinhança são substituídos pelo pixel referência (o no centro da vizinhança). Note que se este valor for NODATA, o comportamento é o mesmo ce 'NULL' (para a vizinhança afetada)

**args** Argumentos para passar dentro da função usuário.

Disponibilidade: 2.0.0

## Exemplos

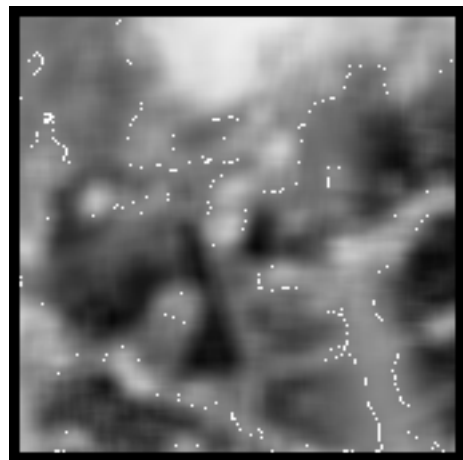
Exemplos utilizam o raster katrina carregado como única tile descrita em [http://trac.osgeo.org/gdal/wiki/frmts\\_wtkraster.html](http://trac.osgeo.org/gdal/wiki/frmts_wtkraster.html) e preparada nos exemplos [ST\\_Rescale](#)

```
--
-- A simple 'callback' user function that averages up all the values in a neighborhood.
--
CREATE OR REPLACE FUNCTION rast_avg(matrix float[][] , nodatamode text, variadic args text ←
[])
RETURNS float AS
$$
DECLARE
    _matrix float[][];
    x1 integer;
    x2 integer;
    y1 integer;
    y2 integer;
    sum float;
BEGIN
    _matrix := matrix;
    sum := 0;
    FOR x in array_lower(matrix, 1)..array_upper(matrix, 1) LOOP
        FOR y in array_lower(matrix, 2)..array_upper(matrix, 2) LOOP
            sum := sum + _matrix[x][y];
        END LOOP;
    END LOOP;
    RETURN (sum*1.0/(array_upper(matrix,1)*array_upper(matrix,2) ))::integer ;
END;
$$
LANGUAGE 'plpgsql' IMMUTABLE COST 1000;

-- now we apply to our raster averaging pixels within 2 pixels of each other in X and Y ←
direction --
SELECT ST_MapAlgebraFctNgb(rast, 1, '8BUI', 4,4,
'rast_avg(float[][] , text, text[])'::regprocedure, 'NULL', NULL) As ←
nn_with_border
FROM katrinas_rescaled
limit 1;
```



*Primeira banda do nosso raster*



*novo raster depois de calcular pixels atando 4x4 pixels de cada um*

**Veja também**

[ST\\_MapAlgebraFct](#), [ST\\_MapAlgebraExpr](#), [ST\\_Rescale](#)

**9.12.1.12 ST\_Reclass**

**ST\_Reclass** — Cria um novo raster composto por tipos de banda reclassificados do original. A nband pode ser alterada. Se nenhuma nband for especificada, usa-se a 1. Todas as outras bandas são retornadas inalteradas. Use caso: converta uma banda 16BUI para 8BUI e então adiante para uma renderização mais simples como formatos visíveis.

**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);
```

**Descrição**

Cria um novo raster formado pela aplicação de uma operação algébrica válida PostgreSQL definida pela *reclassexpr* no raster de entrada (*rast*). Se nenhum *band* for especificada, usa-se a banda 1. O novo raster terá a mesma georreferência, largura e altura do raster original. As bandar não designadas voltarão inalteradas. Recorra a [reclassarg](#) para descrição de expressões de reclassificação válidas.

As bandas do novo raster terão o mesmo tipo de pixel do *pixeltype*. Se *reclassargset* passar, então, cada argumento reclassificado define o comportamento de cada banda gerada.

Disponibilidade: 2.0.0

**Exemplos básicos**

Cria um novo raster a partir do original onde banda 2 é convertida de 8BUI para 4BUI e todos os valores de 101-254 são definidos para valor nodata.

```
ALTER TABLE dummy_rast ADD COLUMN reclass_rast raster;
UPDATE dummy_rast SET reclass_rast = ST_Reclass(rast,2,'0-87:1-10, 88-100:11-15, ←
    101-254:0-0', '4BUI',0) WHERE rid = 2;

SELECT i as col, j as row, ST_Value(rast,2,i,j) As origval,
       ST_Value(reclass_rast, 2, i, j) As reclassval,
       ST_Value(reclass_rast, 2, i, j, false) As reclassval_include_nodata
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1,3) AS j
WHERE rid = 2;
```

col	row	origval	reclassval	reclassval_include_nodata
1	1	78	9	9
2	1	98	14	14
3	1	122		0
1	2	96	14	14
2	2	118		0
3	2	180		0
1	3	99	15	15
2	3	112		0
3	3	169		0



**Exemplo: Uso avançado de múltiplos reclassargs**

Cria um novo raster do original onde banda 1,2,3 é convertida para 1BB, 4BUI, 4BUI respectivamente e reclassificada. Note que isto usa o argumento variado `reclassarg` que pode pegar como entrada e número indefinido de reclassargs (teoricamente quantas bandas tiver)

```
UPDATE dummy_rast SET reclass_rast =
  ST_Reclass(rast,
    ROW(2, '0-87]:1-10, (87-100]:11-15, (101-254):0-0', '4BUI', NULL)::reclassarg,
    ROW(1, '0-253]:1, 254:0', '1BB', NULL)::reclassarg,
    ROW(3, '0-70]:1, (70-86:2, [86-150]:3, [150-255:4', '4BUI', NULL)::reclassarg
  ) WHERE rid = 2;

SELECT i as col, j as row, ST_Value(rast,1,i,j) As ov1, ST_Value(reclass_rast, 1, i, j) As ←
  rv1,
  ST_Value(rast,2,i,j) As ov2, ST_Value(reclass_rast, 2, i, j) As rv2,
  ST_Value(rast,3,i,j) As ov3, ST_Value(reclass_rast, 3, i, j) As rv3
FROM dummy_rast CROSS JOIN generate_series(1, 3) AS i CROSS JOIN generate_series(1,3) AS j
WHERE rid = 2;
```

col	row	ov1	rv1	ov2	rv2	ov3	rv3
1	1	253	1	78	9	70	1
2	1	254	0	98	14	86	3
3	1	253	1	122	0	100	3
1	2	253	1	96	14	80	2
2	2	254	0	118	0	108	3
3	2	254	0	180	0	162	4
1	3	250	1	99	15	90	3
2	3	254	0	112	0	108	3
3	3	254	0	169	0	175	4

**Exemplo: Mapeamento avançado de uma única banda raster 32BF para multiplicar bandas visíveis**

Cria uma nova banda 3 (8BUI,8BUI,8BUI raster visível) a partir de um raster que tem apenas uma banda 32bf

```
ALTER TABLE wind ADD COLUMN rast_view raster;
UPDATE wind
  set rast_view = ST_AddBand( NULL,
  ARRAY[
    ST_Reclass(rast, 1, '0.1-10]:1-10, 9-10]:11, (11-33:0'::text, '8BUI'::text, 0),
    ST_Reclass(rast, 1, '11-33):0-255, [0-32:0, (34-1000:0'::text, '8BUI'::text, 0),
    ST_Reclass(rast, 1, '0-32]:0, (32-100:100-255'::text, '8BUI'::text, 0)
  ]
  );
```

**Veja também**

[ST\\_AddBand](#), [ST\\_Band](#), [ST\\_BandPixelType](#), [ST\\_MakeEmptyRaster](#), [reclassarg](#), [ST\\_Value](#)

**9.12.1.13 ST\_Union**

`ST_Union` — Retorna a união de um conjunto de tiles raster em um único raster composto de 1 ou mais bandas.

**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);
```

### Descrição

Retorna a união de um conjunto de tiles raster em um único raster composto de pelo menos uma banda. A extensão resultante do raster é a extensão do conjunto todo. No caso da interseção, o valor resultante é definido pelo `uniontype` que é um dos seguintes: LAST (default), FIRST, MIN, MAX, COUNT, SUM, MEAN, RANGE.



#### Note

In order for rasters to be unioned, they must all have the same alignment. Use [ST\\_SameAlignment](#) and [ST\\_NotSameAlignmentReason](#) for more details and help. One way to fix alignment issues is to use [ST\\_Resample](#) and use the same reference raster for alignment.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Velocidade aprimorada (fully C-Based)

Disponibilidade: 2.1.0 variante `ST_Union(rast, unionarg)` foi introduzida.

Melhorias: 2.1.0 uniões `ST_Union(rast)` (variante 1) todas as bandas de todos os rasters de entrada. As versões anteriores do PostGIS assumiam a primeira banda.

Melhorias: 2.1.0 `ST_Union(rast, uniontype)` (variante 4) uniões de todas as bandas de todos os rasters de entrada.

### Exemplos: Reconstitui uma única tile banda raster em pedaços

```
-- this creates a single band from first band of raster tiles
-- that form the original file system tile
SELECT filename, ST_Union(rast,1) As file_rast
FROM sometable WHERE filename IN('dem01', 'dem02') GROUP BY filename;
```

### Exemplos: Retorna uma multi banda raster que é a união de tiles intersectando geometrias

```
-- this creates a multi band raster collecting all the tiles that intersect a line
-- Note: In 2.0, this would have just returned a single band raster
-- , new union works on all bands by default
-- this is equivalent to unionarg:ARRAY[ROW(1, 'LAST'), ROW(2, 'LAST'), ROW(3, 'LAST')]:: ←
  unionarg[]
SELECT ST_Union(rast)
FROM aerials.boston
WHERE ST_Intersects(rast, ST_GeomFromText('LINESTRING(230486 887771, 230500 88772)',26986) ←
);
```

### Exemplos: Retorna uma multi banda raster que é a união de tiles intersectando geometrias

Aqui, usamos a sintaxe mais longa se só queremos uma subset de bandas ou queremos alterar a ordem das bandas

```
-- this creates a multi band raster collecting all the tiles that intersect a line
SELECT ST_Union(rast,ARRAY[ROW(2, 'LAST'), ROW(1, 'LAST'), ROW(3, 'LAST')]::unionarg[])
FROM aerials.boston
WHERE ST_Intersects(rast, ST_GeomFromText('LINESTRING(230486 887771, 230500 88772)',26986) ←
);
```

**Veja também**

[unionarg](#), [ST\\_Envelope](#), [ST\\_ConvexHull](#), [ST\\_Clip](#), [ST\\_Union](#)

**9.12.2 Funções retorno de mapa algébrico embutido****9.12.2.1 ST\_Distinct4ma**

ST\_Distinct4ma — Função de processamento raster que calcula o resumo de valores únicos de pixel em uma vizinhança.

**Synopsis**

float8 **ST\_Distinct4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);  
double precision **ST\_Distinct4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

**Descrição**

Calcula o número de valores únicos de pixel em uma vizinhança.

**Note**

A variante 1 é uma função retorno especializada em uso como um parâmetro de retorno para [ST\\_MapAlgebraFctNgb](#).

**Note**

A variante 2 é uma função retorno especializada em uso como um parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).

**Warning**

Uso da variante 1 é desencorajado desde que [ST\\_MapAlgebraFctNgb](#) foi menosprezada como de 2.1.0.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Adição da variante 2

**Exemplos**

```
SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, NULL, 1, 1, 'st_distinct4ma(float[][],text,text[])':: regprocedure, 'ignore', NULL), 2, 2
  )
FROM dummy_rast
WHERE rid = 2;
  rid | st_value
-----+-----
    2 |        3
(1 row)
```

**Veja também**

[ST\\_MapAlgebraFctNgb](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Min4ma](#), [ST\\_Max4ma](#), [ST\\_Sum4ma](#), [ST\\_Mean4ma](#), [ST\\_Distinct4ma](#), [ST\\_StdDev4ma](#)

**9.12.2.2 ST\_InvDistWeight4ma**

`ST_InvDistWeight4ma` — Função de processamento raster que interpola um valor de pixel de uma vizinhança.

**Synopsis**

double precision `ST_InvDistWeight4ma`(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

**Descrição**

Calcula um valor interpolado para um pixel usando o método do inverso da potência das distâncias.

Existem dois parâmetros opcionais que podem ser passados pelos `userargs`. O primeiro parâmetro é o fator de força (variável  $k$  na equação abaixo) entre 0 e 1 usado na equação do inverso da potência das distâncias. Se não especificado, usa-se 1. O segundo parâmetro é a porcentagem aplicada somente quando o valor do pixel de interesse estiver incluso no valor da vizinhança. Se não especificado e o pixel de interesse possuir um valor, o valor é retornado.

A equação do inverso da potência das distâncias é:

$$\hat{z}(x_o) = \frac{\sum_{j=1}^m z(x_j) d_{ij}^{-k}}{\sum_{j=1}^m d_{ij}^{-k}}$$

$k$  = fator força, um número real entre 0 e 1

**Note**

Esta função é uma função retorno especializada em uso como parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).

Disponibilidade: 2.1.0

**Exemplos**

```
-- PRECISA DE EXEMPLO
```

**Veja também**

[Funções retorno de mapa algébrico embutido](#), [ST\\_MinDist4ma](#)

**9.12.2.3 ST\_Max4ma**

`ST_Max4ma` — Função de processamento raster que calcula o valor máximo de pixel em uma vizinhança.

## Synopsis

float8 **ST\_Max4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);

double precision **ST\_Max4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

## Descrição

Calcula o valor de pixel máximo em uma vizinhança de pixels.

Para Variante 2, um valor de substituição para pixels NODATA podem ser especificados passando aquele valor para userargs.



### Note

A variante 1 é uma função retorno especializada em uso como um parâmetro de retorno para [ST\\_MapAlgebraFctNgb](#).



### Note

A variante 2 é uma função retorno especializada em uso como um parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).



### Warning

Uso da variante 1 é desencorajado desde que [ST\\_MapAlgebraFctNgb](#) foi menosprezada como de 2.1.0.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Adição da variante 2

## Exemplos

```
SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, NULL, 1, 1, 'st_max4ma(float [][] ,text ,text [])':: ↵
      regprocedure, 'ignore', NULL), 2, 2
  )
FROM dummy_rast
WHERE rid = 2;
  rid | st_value
-----+-----
    2 |      254
(1 row)
```

## Veja também

[ST\\_MapAlgebraFctNgb](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Min4ma](#), [ST\\_Sum4ma](#), [ST\\_Mean4ma](#), [ST\\_Range4ma](#), [ST\\_Distinct4ma](#), [ST\\_StdDev4ma](#)

### 9.12.2.4 ST\_Mean4ma

**ST\_Mean4ma** — Função de processamento raster que calcula o menor valor de pixel em uma vizinhança.

## Synopsis

float8 **ST\_Mean4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);

double precision **ST\_Mean4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

## Descrição

Calcula o menor valor de pixel em uma vizinhança de pixels.

Para Variante 2, um valor de substituição para pixels NODATA podem ser especificados passando aquele valor para userargs.



### Note

A variante 1 é uma função retorno especializada em uso como um parâmetro de retorno para [ST\\_MapAlgebraFctNgb](#).



### Note

A variante 2 é uma função retorno especializada em uso como um parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).



### Warning

Uso da variante 1 é desencorajado desde que [ST\\_MapAlgebraFctNgb](#) foi menosprezada como de 2.1.0.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Adição da variante 2

## Exemplos: Variante 1

```
SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, '32BF', 1, 1, 'st_mean4ma(float[][],text,text[])':: regprocedure, 'ignore', NULL), 2, 2
  )
FROM dummy_rast
WHERE rid = 2;
  rid |      st_value
-----+-----
    2 | 253.222229003906
(1 row)
```

## Exemplos: Variant 2

```
SELECT
  rid,
  st_value(
    ST_MapAlgebra(rast, 1, 'st_mean4ma(double precision[][][], integer[][], text [])'::regprocedure,'32BF', 'FIRST', NULL, 1, 1)
    , 2, 2)
FROM dummy_rast
```

```

WHERE rid = 2;
rid |      st_value
-----+-----
  2 | 253.222229003906
(1 row)

```

## Veja também

[ST\\_MapAlgebraFctNgb](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Min4ma](#), [ST\\_Max4ma](#), [ST\\_Sum4ma](#), [ST\\_Range4ma](#), [ST\\_StdDev4ma](#)

### 9.12.2.5 ST\_Min4ma

**ST\_Min4ma** — Função de processamento raster que calcula o valor mínimo de pixel em uma vizinhança.

#### Synopsis

float8 **ST\_Min4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);  
double precision **ST\_Min4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

#### Descrição

Calcula o valor de pixel mínimo em uma vizinhança de pixels.

Para Variante 2, um valor de substituição para pixels NODATA podem ser especificados passando aquele valor para userargs.



#### Note

A variante 1 é uma função retorno especializada em uso como um parâmetro de retorno para [ST\\_MapAlgebraFctNgb](#).



#### Note

A variante 2 é uma função retorno especializada em uso como um parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).



#### Warning

Uso da variante 1 é desencorajado desde que [ST\\_MapAlgebraFctNgb](#) foi menosprezada como de 2.1.0.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Adição da variante 2

## Exemplos

```

SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, NULL, 1, 1, 'st_min4ma(float [] [],text,text[])':: ←
      regprocedure, 'ignore', NULL), 2, 2

```

```

)
FROM dummy_rast
WHERE rid = 2;
  rid | st_value
-----+-----
    2 |      250
(1 row)

```

### Veja também

[ST\\_MapAlgebraFctNgb](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Max4ma](#), [ST\\_Sum4ma](#), [ST\\_Mean4ma](#), [ST\\_Range4ma](#), [ST\\_Distinct4ma](#), [ST\\_StdDev4ma](#)

#### 9.12.2.6 ST\_MinDist4ma

**ST\_MinDist4ma** — Função de processamento raster que retorna a distância mínima (em números de pixels) entre o pixel de interesse e um pixel vizinho de interesse com valor.

### Synopsis

double precision **ST\_MinDist4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

### Descrição

Retorna a menor função (em números de pixels) entre o pixel de interesse e o pixel mais próximo com valor na vizinhança.



#### Note

A intenção desta função é fornecer um ponto de dados informativos que ajude inferir a utilidade do valor interpolado do pixel de interesse da [ST\\_InvDistWeight4ma](#). Esta função é particularmente útil quando a vizinhança é esparsamente populada.



#### Note

Esta função é uma função retorno especializada em uso como parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).

Disponibilidade: 2.1.0

### Exemplos

```
-- PRECISA DE EXEMPLO
```

### Veja também

[Funções retorno de mapa algébrico embutido](#), [ST\\_InvDistWeight4ma](#)

#### 9.12.2.7 ST\_Range4ma

**ST\_Range4ma** — Função de processamento raster que calcula a variação de valores de pixel em uma vizinhança.



## Synopsis

float8 **ST\_Range4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);  
 double precision **ST\_Range4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

## Descrição

Calcula a variação de valores de pixel em uma vizinhança de pixels.

Para Variante 2, um valor de substituição para pixels NODATA podem ser especificados passando aquele valor para userargs.



### Note

A variante 1 é uma função retorno especializada em uso como um parâmetro de retorno para [ST\\_MapAlgebraFctNgb](#).



### Note

A variante 2 é uma função retorno especializada em uso como um parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).



### Warning

Uso da variante 1 é desencorajado desde que [ST\\_MapAlgebraFctNgb](#) foi menosprezada como de 2.1.0.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Adição da variante 2

## Exemplos

```
SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, NULL, 1, 1, 'st_range4ma(float[][],text,text[])':: ↵
      regprocedure, 'ignore', NULL), 2, 2
  )
FROM dummy_rast
WHERE rid = 2;
  rid | st_value
-----+-----
    2 |      4
(1 row)
```

## Veja também

[ST\\_MapAlgebraFctNgb](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Min4ma](#), [ST\\_Max4ma](#), [ST\\_Sum4ma](#), [ST\\_Mean4ma](#), [ST\\_Distinct4ma](#), [ST\\_StdDev4ma](#)

### 9.12.2.8 ST\_StdDev4ma

**ST\_StdDev4ma** — Função de processamento raster que calcula o padrão de divergência de valores de pixel em uma vizinhança.

## Synopsis

float8 **ST\_StdDev4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);  
 double precision **ST\_StdDev4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

## Descrição

Calcula o padrão de divergência de valores de pixel em uma vizinhança de pixels.



### Note

A variante 1 é uma função retorno especializada em uso como um parâmetro de retorno para [ST\\_MapAlgebraFctNgb](#).



### Note

A variante 2 é uma função retorno especializada em uso como um parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).



### Warning

Uso da variante 1 é desencorajado desde que [ST\\_MapAlgebraFctNgb](#) foi menosprezada como de 2.1.0.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Adição da variante 2

## Exemplos

```
SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, '32BF', 1, 1, 'st_stddev4ma(float[][],text,text[])':: ←
      regprocedure, 'ignore', NULL), 2, 2
  )
FROM dummy_rast
WHERE rid = 2;
  rid |      st_value
-----+-----
    2 | 1.30170822143555
(1 row)
```

## Veja também

[ST\\_MapAlgebraFctNgb](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Min4ma](#), [ST\\_Max4ma](#), [ST\\_Sum4ma](#), [ST\\_Mean4ma](#), [ST\\_Distinct4ma](#), [ST\\_StdDev4ma](#)

### 9.12.2.9 ST\_Sum4ma

**ST\_Sum4ma** — Função de processamento raster que calcula o resumo de todos os valores de pixel em uma vizinhança.

## Synopsis

float8 **ST\_Sum4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);  
 double precision **ST\_Sum4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

## Descrição

Calcula o resumo de todos os valores de pixel em uma vizinhança de pixels.

Para Variante 2, um valor de substituição para pixels NODATA podem ser especificados passando aquele valor para userargs.



### Note

A variante 1 é uma função retorno especializada em uso como um parâmetro de retorno para [ST\\_MapAlgebraFctNgb](#).



### Note

A variante 2 é uma função retorno especializada em uso como um parâmetro de retorno para [Funções retorno de mapa algébrico embutido](#).



### Warning

Uso da variante 1 é desencorajado desde que [ST\\_MapAlgebraFctNgb](#) foi menosprezada como de 2.1.0.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Adição da variante 2

## Exemplos

```
SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, '32BF', 1, 1, 'st_sum4ma(float[][] ,text,text[])':: ↵
      regprocedure, 'ignore', NULL), 2, 2
  )
FROM dummy_rast
WHERE rid = 2;
  rid | st_value
-----+-----
    2 |    2279
(1 row)
```

## Veja também

[ST\\_MapAlgebraFctNgb](#), [Funções retorno de mapa algébrico embutido](#), [ST\\_Min4ma](#), [ST\\_Max4ma](#), [ST\\_Mean4ma](#), [ST\\_Range4ma](#), [ST\\_Distinct4ma](#), [ST\\_StdDev4ma](#)

## 9.12.3 DEM (Elevação)

### 9.12.3.1 ST\_Aspect

**ST\_Aspect** — Retorna o aspecto (em graus) de uma banda raster de elevação. Útil para analisar terrenos.

## 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);

## Descrição

Retorna o aspecto (em graus) de uma banda raster de elevação. Utiliza mapa algébrico e aplica o aspecto de equação para pixels vizinhos.

`units` indica as unidade do aspecto. Possíveis valores são: RADIANOS, GRAUS (padrão).

Quando `units = RADIANOS`, valores são entre 0 e  $2 * \pi$  radianos medidos sentido horário a partir do Norte.

Quando `units = GRAUS`, valores são entre 0 e 360 graus medidos a partir do Norte.

Se o declive de pixel for zero, o aspecto do pixel é -1.



### Note

Para maiores informações sobre declive, aspecto e sombreado, por favor recorra a [ESRI - How hillshade works](#) e [ERDAS Field Guide - Aspect Images](#).

Disponibilidade: 2.0.0

melhorias: 2.1.0 Usa `ST_MapAlgebra()` e foi adicionado uma função parâmetro opcional `interpolate_nodata`

Alterações: 2.1.0 Nas versões anteriores, os valores retornados eram em radianos. Agora, eles retornam em graus

## Exemplos: Variante 1

```
WITH foo AS (
  SELECT ST_SetValues(
    ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF', 0, ←
      -9999),
    1, 1, 1, ARRAY[
      [1, 1, 1, 1, 1],
      [1, 2, 2, 2, 1],
      [1, 2, 3, 2, 1],
      [1, 2, 2, 2, 1],
      [1, 1, 1, 1, 1]
    ]::double precision[][]
  ) AS rast
)
SELECT
  ST_DumpValues(ST_Aspect(rast, 1, '32BF'))
FROM foo
```

---

```
(1, "{315,341.565063476562,0,18.4349479675293,45},{288.434936523438,315,0,45,71.5650482177734},{270,2227,180,161.565048217773,135}}")
(1 row)
```

**Exemplos: Variant 2**

Exemplo completo de tiles de uma cobertura. Esta consulta funciona apenas com PostgreSQL 9.1 ou superior.

```
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;
```

**Veja também**

[Funções retorno de mapa algébrico embutido](#), [ST\\_TRI](#), [ST\\_TPI](#), [ST\\_Roughness](#), [ST\\_HillShade](#), [ST\\_Slope](#)

**9.12.3.2 ST\_HillShade**

**ST\_HillShade** — Retorna a iluminação hipotética de uma banda raster de elevação usando as entradas de azimuth, altitude, claridade e escala fornecidas.

**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);

**Descrição**

Retorna a iluminação hipotética de uma banda raster de elevação usando as entradas de azimuth, altitude, claridade e escala fornecidas. Utiliza mapa algébrico e aplica a equação sombreada nos pixels vizinhos. Os valores de pixel retornados estão entre 0 e 255.

`azimuth` é um valor entre 0 e 360 graus medidos no sentido horário a partir do Norte.

`altitude` é um valor entre 0 e 90 graus onde 0 grau está no horizonte e 90 graus estão diretamente em cima.

`max_bright` é um valor entre 0 e 255 com 0 sendo nenhuma claridade e 255 sendo a claridade máxima.

scale is the ratio of vertical units to horizontal. For Feet:LatLon use scale=370400, for Meters:LatLon use scale=111120.

Se interpolate\_nodata for VERDADE, valores para pixels NODATA do raster de entrada serão interpolados usando [ST\\_InvDistWeight4ma](#) antes de calcular a iluminação sombreada.



#### Note

para maiores informações sobre sombreamento, por favor recorra a [How hillshade works](#).

Disponibilidade: 2.0.0

melhorias: 2.1.0 Usa ST\_MapAlgebra() e foi adicionado uma função parâmetro opcional interpolate\_nodata

Alterações: 2.1.0 Nas versões anteriores, o azimute e a altitude eram expressados em radianos. Agora, são representados em graus

#### Exemplos: Variante 1

```
WITH foo AS (
  SELECT ST_SetValues(
    ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF', 0, -9999),
    1, 1, 1, ARRAY[
      [1, 1, 1, 1, 1],
      [1, 2, 2, 2, 1],
      [1, 2, 3, 2, 1],
      [1, 2, 2, 2, 1],
      [1, 1, 1, 1, 1]
    ]::double precision[][]
  ) AS rast
)
SELECT
  ST_DumpValues(ST_Hillshade(rast, 1, '32BF'))
FROM foo
```

```
-----
(1, "{ {NULL,NULL,NULL,NULL,NULL}, {NULL,251.32763671875,220.749786376953,147.224319458008,
  NULL}, {NULL,220.749786376953,180.312225341797,67.7497863769531,NULL}, {NULL,
  147.224319458008
,67.7497863769531,43.1210060119629,NULL}, {NULL,NULL,NULL,NULL,NULL}}")
(1 row)
```

#### Exemplos: Variant 2

Exemplo completo de tiles de uma cobertura. Esta consulta funciona apenas com PostgreSQL 9.1 ou superior.

```
WITH foo AS (
  SELECT ST_Tile(
    ST_SetValues(
      ST_AddBand(
        ST_MakeEmptyRaster(6, 6, 0, 0, 1, -1, 0, 0, 0),
        1, '32BF', 0, -9999
      ),

```

```

        1, 1, 1, ARRAY[
            [1, 1, 1, 1, 1, 1],
            [1, 1, 1, 1, 2, 1],
            [1, 2, 2, 3, 3, 1],
            [1, 1, 3, 2, 1, 1],
            [1, 2, 2, 1, 2, 1],
            [1, 1, 1, 1, 1, 1]
        ]::double precision[]
    ),
    2, 2
) AS rast
)
SELECT
    t1.rast,
    ST_Hillshade(ST_Union(t2.rast), 1, t1.rast)
FROM foo t1
CROSS JOIN foo t2
WHERE ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rast;

```

### Veja também

Funções retorno de mapa algébrico embutido, [ST\\_TRI](#), [ST\\_TPI](#), [ST\\_Roughness](#), [ST\\_Aspect](#), [ST\\_Slope](#)

#### 9.12.3.3 ST\_Roughness

**ST\_Roughness** — Retorna um raster com a "robustez" calculada de um DEM.

#### Synopsis

```
raster ST_Roughness(raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);
```

#### Descrição

Calcula a "robustez" de um DEM, subtraindo o máximo do mínimo de uma dada área.

Disponibilidade: 2.1.0

#### Exemplos

```
-- precisa de exemplos
```

### Veja também

Funções retorno de mapa algébrico embutido, [ST\\_TRI](#), [ST\\_TPI](#), [ST\\_Slope](#), [ST\\_HillShade](#), [ST\\_Aspect](#)

#### 9.12.3.4 ST\_Slope

**ST\_Slope** — Retorna o declive (em graus) de uma banda raster de elevação. Útil para analisar terrenos.

## 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);

## Descrição

Retorna o declive (em graus) de uma banda raster de elevação. Utiliza mapa algébrico e aplica a equação de declive nos pixels vizinhos.

`units` indica as unidades do declive. Possíveis valores são: RADIANOS, GRAUS (padrão), PORCENTAGEM.

`scale` is the ratio of vertical units to horizontal. For Feet:LatLon use `scale=370400`, for Meters:LatLon use `scale=111120`.

Se `interpolate_nodata` for VERDADE, valores para pixels NODATA do raster de entrada serão interpolados usando **ST\_InvDistWeight4ma** antes de calcular a superfície inclinada.



### Note

Para maiores informações sobre declive, aspecto e sombreado, por favor recorra a [ESRI - How hillshade works](#) and [ERDAS Field Guide - Slope Images](#).

Disponibilidade: 2.0.0

Melhorias: 2.1.0 Usa `ST_MapAlgebra()` e foi adicionado a função parâmetros opcionais `units`, `scale`, `interpolate_nodata`

Alterações: 2.1.0 Nas versões anteriores, os valores retornados eram em radianos. Agora, eles retornam em graus

## Exemplos: Variante 1

```
WITH foo AS (
  SELECT ST_SetValues(
    ST_AddBand(ST_MakeEmptyRaster(5, 5, 0, 0, 1, -1, 0, 0, 0), 1, '32BF', 0, ←
      -9999),
    1, 1, 1, ARRAY[
      [1, 1, 1, 1, 1],
      [1, 2, 2, 2, 1],
      [1, 2, 3, 2, 1],
      [1, 2, 2, 2, 1],
      [1, 1, 1, 1, 1]
    ]::double precision[][]
  ) AS rast
)
SELECT
  ST_DumpValues(ST_Slope(rast, 1, '32BF'))
FROM foo

          st_dumpvalues
-----
-----
-----
(1, "{10.0249881744385,21.5681285858154,26.5650520324707,21.5681285858154,10.0249881744385},{21.5681285858154,26.5650520324707,36.8698959350586,0,36.8698959350586,26.5650520324707},{21.5681285858154,35.2643890
```



```
5681285858154,26.5650520324707,21.5681285858154,10.0249881744385}}")
(1 row)
```

### Exemplos: Variant 2

Exemplo completo de tiles de uma cobertura. Esta consulta funciona apenas com PostgreSQL 9.1 ou superior.

```
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;
```

### Veja também

[Funções retorno de mapa algébrico embutido](#), [ST\\_TRI](#), [ST\\_TPI](#), [ST\\_Roughness](#), [ST\\_HillShade](#), [ST\\_Aspect](#)

#### 9.12.3.5 ST\_TPI

ST\_TPI — Retorna um raster com o índice de posição topográfico calculado.

#### Synopsis

raster **ST\_TPI**(raster rast, integer nband, raster customextent, text pixeltype="32BF" , boolean interpolate\_nodata=FALSE );

#### Descrição

Calculates the Topographic Position Index, which is defined as the focal mean with radius of one minus the center cell.



#### Note

Esta função suporta apenas o raio mínimo central.

Disponibilidade: 2.1.0

## Exemplos

```
-- precisa de exemplos
```

## Veja também

Funções retorno de mapa algébrico embutido, [ST\\_TRI](#), [ST\\_Roughness](#), [ST\\_Slope](#), [ST\\_HillShade](#), [ST\\_Aspect](#)

### 9.12.3.6 ST\_TRI

`ST_TRI` — Retorna um raster com o índice de aspereza do terreno calculado.

## Synopsis

```
raster ST_TRI(raster rast, integer nband, raster customextent, text pixeltype="32BF" , boolean interpolate_nodata=FALSE );
```

## Descrição

O índice de aspereza do terreno é calculado pela comparação de um pixel central com seus vizinhos, pegando os valores absolutos das diferenças, e calculando o resultado.



### Note

Esta função suporta apenas o raio mínimo central.

Disponibilidade: 2.1.0

## Exemplos

```
-- precisa de exemplos
```

## Veja também

Funções retorno de mapa algébrico embutido, [ST\\_Roughness](#), [ST\\_TPI](#), [ST\\_Slope](#), [ST\\_HillShade](#), [ST\\_Aspect](#)

## 9.12.4 Raster para Geometria

### 9.12.4.1 Caixa3D

`Caixa3D` — Retorna a representação da caixa 3d da caixa encerrada do raster.

## Synopsis

```
box3d Box3D(raster rast);
```

## Descrição

Retorna a caixa representando a extensão do raster.

O polígono é definido pelos pontos de canto da caixa delimitadora ((MINX, MINY), (MAXX, MAXY))

Alterações: 2.0.0 Nas versões pre-2.0, costumava existir uma caixa2d em vez de uma caixa3d. Já que a caixa2d é um tipo inferior, foi alterado para caixa3d.

## Exemplos

```
SELECT
    rid,
    Box3D(rast) AS rastbox
FROM dummy_rast;
```

rid	rastbox
1	BOX3D(0.5 0.5 0,20.5 60.5 0)
2	BOX3D(3427927.75 5793243.5 0,3427928 5793244 0)

## Veja também

[ST\\_Envelope](#)

### 9.12.4.2 ST\_ConvexHull

**ST\_ConvexHull** — Retorna o casco convexo da geometria do raster incluindo valores iguais ao `BandNoDataValue`. Para rasters com formas normais e não desviadas, o resultado é o mesmo que `ST_Envelope`, então só é útil para rasters com formas irregulares ou desviados.

## Synopsis

```
geometry ST_ConvexHull(raster rast);
```

## Descrição

Retorna o casco convexo da geometria do raster incluindo valores iguais ao `NoDataBandValue` pixels banda. Para rasters com formas normais e não desviadas, o resultado é o mesmo que `ST_Envelope`, então só é útil para rasters com formas irregulares ou desviados.



### Note

`ST_Envelope` derruba as coordenadas e por isso adiciona um pequeno buffer em torno do raster, então a resposta é um pouco diferente da `ST_ConvexHull` que não derruba.

## Exemplos

Recorra a [PostGIS Raster Specification](#) para um diagrama.

```
-- Note envelope and convexhull are more or less the same
SELECT ST_AsText(ST_ConvexHull(rast)) As convhull,
       ST_AsText(ST_Envelope(rast)) As env
FROM dummy_rast WHERE rid=1;
```

```

                                convhull | env
-----+-----
POLYGON((0.5 0.5,20.5 0.5,20.5 60.5,0.5 60.5,0.5 0.5)) | POLYGON((0 0,20 0,20 60,0 60,0 0)
)

-- now we skew the raster
-- note how the convex hull and envelope are now different
SELECT ST_AsText(ST_ConvexHull(rast)) As convhull,
       ST_AsText(ST_Envelope(rast)) As env
FROM (SELECT ST_SetRotation(rast, 0.1, 0.1) As rast
      FROM dummy_rast WHERE rid=1) As foo;

                                convhull | env
-----+-----
POLYGON((0.5 0.5,20.5 1.5,22.5 61.5,2.5 60.5,0.5 0.5)) | POLYGON((0 0,22 0,22 61,0 61,0 0)
)

```

**Veja também**

[ST\\_Envelope](#), [ST\\_MinConvexHull](#), [ST\\_ConvexHull](#), [ST\\_AsText](#)

**9.12.4.3 ST\_DumpAsPolygons**

**ST\_DumpAsPolygons** — Retorna um conjunto de linhas geomval (geom,val), de uma dada banda raster. Se nenhum número de banda for especificado, o número de banda torna-se 1.

**Synopsis**

```
setof geomval ST_DumpAsPolygons(raster rast, integer band_num=1, boolean exclude_nodata_value=TRUE);
```

**Descrição**

Esta é uma função retorno (SRF). Ela retorna um conjunto de linhas geomval, formadas por uma geometria (geom) e uma banda pixel valor (val). Cada polígono é a união de todos os pixels para aquela banda que tem o mesmo valor de pixel indicado pelo val.

**ST\_DumpAsPolygon** é útil para poligonizar rasters. É o reverso de um GRUPO POR onde cria novas filas. Por exemplo, pode ser usada para expandir um único raster em POLÍGONOS/MULTIPOLÍGONOS.

Disponibilidade: Requer GDAL 1.7 ou superior.

**Note**

If there is a no data value set for a band, pixels with that value will not be returned except in the case of `exclude_nodata_value=false`.

**Note**

Se você se importa somente com pixels contados com um dado valor em um raster, é mais rápido usar: [ST\\_ValueCount](#).

**Note**

Isto é diferente da **ST\_PixelAsPolygons** onde uma geometria retorna para cada pixel independente do valor do pixel.

## Exemplos

```
-- this syntax requires PostgreSQL 9.3+
SELECT val, ST_AsText(geom) As geomwkt
FROM (
SELECT dp.*
FROM dummy_rast, LATERAL ST_DumpAsPolygons(rast) AS dp
WHERE rid = 2
) As foo
WHERE val BETWEEN 249 and 251
ORDER BY val;
```

val	geomwkt
249	POLYGON((3427927.95 5793243.95,3427927.95 5793243.85,3427928 5793243.85,3427928 5793243.95,3427927.95 5793243.95))
250	POLYGON((3427927.75 5793243.9,3427927.75 5793243.85,3427927.8 5793243.85,3427927.8 5793243.9,3427927.75 5793243.9))
250	POLYGON((3427927.8 5793243.8,3427927.8 5793243.75,3427927.85 5793243.75,3427927.85 5793243.8, 3427927.8 5793243.8))
251	POLYGON((3427927.75 5793243.85,3427927.75 5793243.8,3427927.8 5793243.8,3427927.8 5793243.85,3427927.75 5793243.85))

## Veja também

[geomval](#), [ST\\_Value](#), [ST\\_Polygon](#), [ST\\_ValueCount](#)

### 9.12.4.4 ST\_Envelope

`ST_Envelope` — Retorna a representação de polígono da extensão do raster.

#### Synopsis

geometry `ST_Envelope`(raster rast);

#### Descrição

Retorna a representação de polígono da extensão do raster em unidades de coordenadas espaciais definidas pelo srid. É uma caixa delimitadora float8 mínima representada como um polígono.

O polígono é definido pelos pontos do canto da caixa delimitadora ((MINX, MINY), (MINX, MAXY), (MAXX, MAXY), (MAXX, MINY), (MINX, MINY))

## Exemplos

```
SELECT rid, ST_AsText(ST_Envelope(rast)) As envgeomwkt
FROM dummy_rast;
```

rid	envgeomwkt
1	POLYGON((0 0,20 0,20 60,0 60,0 0))
2	POLYGON((3427927 5793243,3427928 5793243,3427928 5793244,3427927 5793244, 3427927 5793243))

**Veja também**

[ST\\_Envelope](#), [ST\\_AsText](#), [ST\\_SRID](#)

**9.12.4.5 ST\_MinConvexHull**

`ST_MinConvexHull` — Retorna a geometria de casco convexo do raster excluindo os pixels SEM DADOS.

**Synopsis**

```
geometry ST_MinConvexHull(raster rast, integer nband=NULL);
```

**Descrição**

Retorna a geometria de casco convexo do raster excluindo os pixels NODATA. Se `nband` for NULL, todas as bandas do raster serão consideradas.

Disponibilidade: 2.1.0

**Exemplos**

```
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
  ST_AsText(ST_ConvexHull(rast)) AS hull,
  ST_AsText(ST_MinConvexHull(rast)) AS mhull,
  ST_AsText(ST_MinConvexHull(rast, 1)) AS mhull_1,
```



```

MULTIPOLYGON(((3427927.9 5793243.95,3427927.85 5793243.95,3427927.85 5793244,3427927.9 ↵
5793244,3427927.9 5793243.95)),((3427928 5793243.85,3427928 5793243.8,3427927.95 ↵
5793243.8,3427927.95 5793243.85,3427927.9 5793243.85,3427927.9 5793243.9,3427927.9 ↵
5793243.95,3427927.95 5793243.95,3427928 5793243.95,3427928 5793243.85)),((3427927.8 ↵
5793243.75,3427927.75 5793243.75,3427927.75 5793243.8,3427927.75 5793243.85,3427927.75 ↵
5793243.9,3427927.75 5793244,3427927.8 5793244,3427927.8 5793243.9,3427927.8 ↵
5793243.85,3427927.85 5793243.85,3427927.85 5793243.8,3427927.85 5793243.75,3427927.8 ↵
5793243.75)))

-- Or if you want the no data value different for just one time

SELECT ST_AsText(
  ST_Polygon(
    ST_SetBandNoDataValue(rast,1,252)
  ) As geomwkt
FROM dummy_rast
WHERE rid =2;

geomwkt
-----
MULTIPOLYGON(((3427928 5793243.85,3427928 5793243.8,3427928 5793243.75,3427927.85 ↵
5793243.75,3427927.8 5793243.75,3427927.8 5793243.8,3427927.75 5793243.8,3427927.75 ↵
5793243.85,3427927.75 5793243.9,3427927.75 5793244,3427927.8 5793244,3427927.85 ↵
5793244,3427927.9 5793244,3427928 5793244,3427928 5793243.95,3427928 5793243.85) ↵
,(3427927.9 5793243.9,3427927.9 5793243.85,3427927.95 5793243.85,3427927.95 ↵
5793243.9,3427927.9 5793243.9)))

```

## Veja também

[ST\\_Value](#), [ST\\_DumpAsPolygons](#)

## 9.13 Operadores Raster

### 9.13.1 &&

**&&** — Retorna VERDADE se a caixa limitadora de A intersecta a caixa limitadora de B.

#### Synopsis

```

boolean &&( raster A , raster B );
boolean &&( raster A , geometry B );
boolean &&( geometry B , raster A );

```

#### Descrição

O operador **&&** retorna TRUE se a caixa limitadora da geometria/raster A intersecta a caixa limitadora da geometria/raster B.



#### Note

Este operador fará uso de qualquer índice que pode estar disponível nos rasters.

Disponibilidade: 2.0.0



## Exemplos

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast && B.rast As intersect
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B LIMIT 3;
```

a_rid	b_rid	intersect
2	2	t
2	3	f
2	1	f

### 9.13.2 &<

**&<** — Retorna VERDADE se uma caixa limitadora de A está à esquerda da de B.

#### Synopsis

boolean **&<**( raster A , raster B );

#### Descrição

O operador **&<** retorna VERDADE se a caixa limitadora da geometria A sobrepõe ou está à esquerda da caixa da geometria B, ou mais precisamente, sobrepõe ou NÃO está à direita da caixa limitadora da geometria B.



#### Note

Este operador fará uso de qualquer índice que pode estar disponível nos rasters.

## Exemplos

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast &< B.rast As overleft
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

a_rid	b_rid	overleft
2	2	t
2	3	f
2	1	f
3	2	t
3	3	t
3	1	f
1	2	t
1	3	t
1	1	t

### 9.13.3 &>

**&>** — Retorna VERDADE se uma caixa limitadora de A está à direita da de B.

#### Synopsis

boolean **&>**( raster A , raster B );

## Descrição

O operador `&>` retorna `TRUE` se a caixa delimitadora do raster A sobrepuser ou estiver à direita da do raster B, ou mais precisamente, sobrepuser ou **NÃO** estiver à esquerda da do raster B.



### Note

Este operador fará uso de qualquer um dos indexes que talvez estejam disponíveis nas geometrias.

## Exemplos

```
SELECT A.rid As a_rid, B.rid As b_rid, A.rast &> B.rast As overright
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;
```

a_rid	b_rid	overright
2	2	t
2	3	t
2	1	t
3	2	f
3	3	t
3	1	f
1	2	f
1	3	t
1	1	t

## 9.13.4 =

= — Retorna `VERDADE` se a caixa limitadora de A for a mesma de B. Utiliza precisão dupla de caixa limitadora.

### Synopsis

```
boolean =( raster A , raster B );
```

### Descrição

O operador `=` retorna `VERDADE` se a caixa limitadora da geometria/geografia A é a mesma da de B. O PostgreSQL usa o operadores `=`, `<`, e `>` definidos para geometrias para representar ordens e comparações internas de geometrias (ex. em um GRUPO ou ORDEM por oração).



### Caution

Este operador **NÃO** fará uso de nenhum índice que podem estar disponíveis nos rasters. Use `~=`. Este operador existe em sua maioria para poder ser agrupado pela coluna raster.

Disponibilidade: 2.1.0

### Veja também

`~=`

### 9.13.5 @

@ — Retorna VERDADE se a caixa limitadora de A estiver contida pela de B. Utiliza precisão dupla de caixa limitadora.

#### Synopsis

```
boolean @( raster A , raster B );  
boolean @( geometry A , raster B );  
boolean @( raster B , geometry A );
```

#### Descrição

O operador @ retorna TRUE se a caixa delimitadora do raster/geometria A estiver contida pela caixa delimitadora do raster/geometria B.



#### Note

Este operador usará índices espaciais nos rasters.

Disponibilidade: 2.0.0 raster @ raster, raster @ geometria introduzida

Disponibilidade: 2.0.5 geometria @ raster introduzida

#### Veja também

~

### 9.13.6 ~=

~= — Retorna VERDADE se a caixa limitadora de A é a mesma de B.

#### Synopsis

```
boolean ~= ( raster A , raster B );
```

#### Descrição

O operador ~= retorna TRUE se a caixa delimitadora do raster A for a mesma da do raster B.



#### Note

Este operador fará uso de qualquer índice que pode estar disponível nos rasters.

Disponibilidade: 2.0.0

#### Exemplos

Casos de uso muito úteis é pegar dois conjuntos de bandas raster únicas que são do mesmo pedaço, mas representam temas diferentes e criar uma multi banda raster

```
SELECT ST_AddBand(prec.rast, alt.rast) As new_rast  
FROM prec INNER JOIN alt ON (prec.rast ~= alt.rast);
```

**Veja também**

[ST\\_AddBand, =](#)

**9.13.7 ~**

~ — Retorna TRUE se a caixa delimitadora de A estiver contida na do B. Utiliza caixa delimitadora de precisão dupla.

**Synopsis**

```
boolean ~( raster A , raster B );  
boolean ~( geometry A , raster B );  
boolean ~( raster B , geometry A );
```

**Descrição**

O operador ~ retorna TRUE se a caixa delimitadora do raster/geometria A estiver contida na caixa delimitadora do raster/geometria B.

**Note**

Este operador usará índices espaciais nos rasters.

---

Disponibilidade: 2.0.0

**Veja também**

[@](#)

## 9.14 Relações raster e raster de banda espacial

### 9.14.1 ST\_Contains

ST\_Contains — Retorna verdade se nenhum ponto do raster rastB estiver no exterior do raster rastA e pelo menos um ponto do interior do rastB estiver no interior do rastA.

**Synopsis**

```
boolean ST_Contains( raster rastA , integer nbandA , raster rastB , integer nbandB );  
boolean ST_Contains( raster rastA , raster rastB );
```

**Descrição**

O raster rastA contém o rastB se e somente se nenhum ponto do rastB estiver no exterior do rastA. Se o número de banda não for fornecido (ou for NULL), apenas o casco convexo do raster será considerado no teste. Se o número de banda for fornecido, somente aqueles pixels com valor (não NODATA) são considerados no teste.

---

**Note**

Esta função fará uso de qualquer índice que possa estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, usa `ST_Polygon` no raster, ex.: `ST_Contains(ST_Polygon(raster), geometria)` ou `ST_Contains(geometria, ST_Polygon(raster))`.

**Note**

`ST_Contains()` é o inverso da `ST_Within()`. Logo, `ST_Contains(rastA, rastB)` implica `ST_Within(rastB, rastA)`.

Disponibilidade: 2.1.0

**Exemplos**

```
-- specified band numbers
SELECT r1.rid, r2.rid, ST_Contains(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↔
    dummy_rast r2 WHERE r1.rid = 1;
```

NOTICE: The first raster provided has no bands

```
rid | rid | st_contains
-----+-----+-----
 1 | 1 |
 1 | 2 | f
```

```
-- no band numbers specified
SELECT r1.rid, r2.rid, ST_Contains(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN ↔
    dummy_rast r2 WHERE r1.rid = 1;
```

```
rid | rid | st_contains
-----+-----+-----
 1 | 1 | t
 1 | 2 | f
```

**Veja também**

[ST\\_Intersects](#), [ST\\_Within](#)

**9.14.2 ST\_ContainsProperly**

`ST_ContainsProperly` — Retorna verdade se o `rastB` intersectar o interior do `rastA`, mas não o limite ou exterior do `rastA`.

**Synopsis**

boolean `ST_ContainsProperly`( raster `rastA` , integer `nbandA` , raster `rastB` , integer `nbandB` );

boolean `ST_ContainsProperly`( raster `rastA` , raster `rastB` );

**Descrição**

O raster `rastA` contém devidamente o `rastB` se ele intersectar o interior do `rastA`, mas não o limite ou exterior do `rastA`. Se o número de banda não for fornecido (ou for `NULL`), apenas o casco convexo do raster será considerado no teste. Se o número de banda for fornecido, somente aqueles pixels com valor (não `NODATA`) serão considerados no teste.

O `rastA` não se contém devidamente, mas se contém.

**Note**

Esta função fará uso de qualquer índice que possa estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, usa `ST_Polygon` no raster, ex.: `ST_ContainsProperly(ST_Polygon(raster), geometria)` ou `ST_ContainsProperly(geometria, ST_Polygon(raster))`.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT r1.rid, r2.rid, ST_ContainsProperly(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN dummy_rast r2 WHERE r1.rid = 2;
```

rid	rid	st_containsproperly
2	1	f
2	2	f

**Veja também**

[ST\\_Intersects](#), [ST\\_Contains](#)

**9.14.3 ST\_Covers**

`ST_Covers` — Retorna verdade se nenhum ponto do `rastB` estiver de fora do `rastA`.

**Synopsis**

```
boolean ST_Covers( raster rastA , integer nbandA , raster rastB , integer nbandB );
boolean ST_Covers( raster rastA , raster rastB );
```

**Descrição**

O `rastA` cobre `rastB` se e somente se nenhum ponto do `rastB` estiver no exterior do `rastA`. Se o número de banda não for fornecido (ou for `NULL`), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas aqueles pixels com valor (não `NODATA`) serão considerados no teste.

**Note**

Esta função fará uso de qualquer índice que possa estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, usa `ST_Polygon` no raster, ex.: `ST_Coveres(ST_Polygon(raster), geometria)` ou `ST_Coveres(geometria, ST_Polygon(raster))`.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT r1.rid, r2.rid, ST_Covers(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↔
dummy_rast r2 WHERE r1.rid = 2;
```

rid	rid	st_covers
2	1	f
2	2	t

**Veja também**

[ST\\_Intersects](#), [ST\\_CoveredBy](#)

**9.14.4 ST\_CoveredBy**

`ST_CoveredBy` — Retorna verdade se nenhum ponto do `rastA` estiver de fora do `rastB`.

**Synopsis**

boolean `ST_CoveredBy`( raster `rastA` , integer `nbandA` , raster `rastB` , integer `nbandB` );

boolean `ST_CoveredBy`( raster `rastA` , raster `rastB` );

**Descrição**

O `rastA` está coberto pelo `rastB` se e somente se nenhum ponto do `rastA` estiver no exterior do `rastB`. Se o número de banda não for fornecido ( ou for `NULO`), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas aqueles pixels com valor (não `NODATA`) serão considerados no teste.

**Note**

Esta função fará uso de qualquer índice que possa estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, usa `ST_Polygon` no raster, ex.: `ST_CoveredBy(ST_Polygon(raster), geometria)` ou `ST_CoveredBy(geometria, ST_Polygon(raster))`.

Disponibilidade: 2.1.0

## Exemplos

```
SELECT r1.rid, r2.rid, ST_CoveredBy(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↔
dummy_rast r2 WHERE r1.rid = 2;
```

rid	rid	st_coveredby
2	1	f
2	2	t

## Veja também

[ST\\_Intersects](#), [ST\\_Covers](#)

### 9.14.5 ST\_Disjoint

**ST\_Disjoint** — Retorna verdade se raster rastA não intersectar espacialmente com o rastB.

#### Synopsis

boolean **ST\_Disjoint**( raster rastA , integer nbandA , raster rastB , integer nbandB );

boolean **ST\_Disjoint**( raster rastA , raster rastB );

#### Descrição

O rastA e rastB estarão disjuntos se eles não dividirem nenhum espaço. Se o número de banda não for fornecido ( ou for NULL), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas aqueles pixels com valor (não NODATA) serão considerados no teste.



#### Note

Esta função NÃO usa nenhum índice.



#### Note

Para testar a relação espacial de um raster e uma geometria, use `ST_Polygon` no raster, ex.: `ST_Disjoint(ST_Polygon(raster), geometria)`.

Disponibilidade: 2.1.0

## Exemplos

```
-- rid = 1 has no bands, hence the NOTICE and the NULL value for st_disjoint
SELECT r1.rid, r2.rid, ST_Disjoint(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↔
dummy_rast r2 WHERE r1.rid = 2;
```

NOTICE: The second raster provided has no bands

rid	rid	st_disjoint
2	1	
2	2	f



```
-- this time, without specifying band numbers
SELECT r1.rid, r2.rid, ST_Disjoint(r1.rast, r2.rast) FROM dummy_rast r1 CROSS JOIN ↵
    dummy_rast r2 WHERE r1.rid = 2;
```

```
rid | rid | st_disjoint
-----+-----+-----
  2 |  1 | t
  2 |  2 | f
```

## Veja também

[ST\\_Intersects](#)

### 9.14.6 ST\_Intersects

**ST\_Intersects** — Retorna verdade se o raster *rastA* intersectar espacialmente com o raster *rastB*.

#### Synopsis

```
boolean ST_Intersects( raster rastA , integer nbandA , raster rastB , integer nbandB );
boolean ST_Intersects( raster rastA , raster rastB );
boolean ST_Intersects( raster rast , integer nband , geometry geommin );
boolean ST_Intersects( raster rast , geometry geommin , integer nband=NULL );
boolean ST_Intersects( geometry geommin , raster rast , integer nband=NULL );
```

#### Descrição

Retorna verdade se o *rastA* se intersectar espacialmente com o *rastB*. Se o número de banda não for fornecido ( ou for NULO), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas aqueles pixels com valor (não NODATA) serão considerados no teste.



#### Note

Esta função fará uso de qualquer índice que possa estar disponível nos rasters.

Melhorias: 2.0.0 suporte para interseções raster/raster foi introduzido.



#### Warning

Alterações: 2.1.0 O comportamento das variantes `ST_Intersects(raster, geometria)` foi alterado para combinar com `ST_Intersects(geometria, raster)`.

#### Exemplos

```
-- different bands of same raster
SELECT ST_Intersects(rast, 2, rast, 3) FROM dummy_rast WHERE rid = 2;
```

```
st_intersects
-----
t
```

**Veja também**

[ST\\_Intersection](#), [ST\\_Disjoint](#)

### 9.14.7 ST\_Overlaps

`ST_Overlaps` — Retorna verdade se o raster `rastA` e `rastB` se intersectam, mas um deles não contém o outro completamente.

**Synopsis**

boolean `ST_Overlaps`( raster `rastA` , integer `nbandA` , raster `rastB` , integer `nbandB` );

boolean `ST_Overlaps`( raster `rastA` , raster `rastB` );

**Descrição**

Retorna verdade se o raster `rastA` tocar espacialmente o raster `rastB`. Isso significa que eles se intersectam, mas um não contém o outro completamente. Se o número banda não for fornecido (ou for NULO), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas os pixels com valor (não NODATA) serão considerados no teste.

**Note**

Esta função fará uso de qualquer índice que possa estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, use `ST_Polygon` no raster, ex.: `ST_Overlaps(ST_Polygon(raster), geometria)`.

Disponibilidade: 2.1.0

**Exemplos**

```
-- comparing different bands of same raster
SELECT ST_Overlaps(rast, 1, rast, 2) FROM dummy_rast WHERE rid = 2;

 st_overlaps
-----
f
```

**Veja também**

[ST\\_Intersects](#)

### 9.14.8 ST\_Touches

`ST_Touches` — Retorna verdade se o raster `rastA` e `rastB` têm pelo menos um ponto em comum, mas seus interiores não se intersectarem.

## Synopsis

boolean **ST\_Touches**( raster rastA , integer nbandA , raster rastB , integer nbandB );  
 boolean **ST\_Touches**( raster rastA , raster rastB );

## Descrição

Retorna verdade se o raster rastA tocar espacialmente o raster rastB. Isso significa que eles têm pelo menos um ponto em comum, mas seus interiores não se intersectam. Se o número banda não for fornecido (ou for NULO), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas os pixels com valor (não NODATA) serão considerados no teste.



### Note

Esta função fará uso de qualquer índice que possa estar disponível nos rasters.



### Note

Para testar a relação espacial de um raster e uma geometria, use `ST_Polygon` no raster, ex.: `ST_Touches(ST_Polygon(raster), geometria)`.

Disponibilidade: 2.1.0

## Exemplos

```
SELECT r1.rid, r2.rid, ST_Touches(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↵
  dummy_rast r2 WHERE r1.rid = 2;
```

rid	rid	st_touches
2	1	f
2	2	f

## Veja também

[ST\\_Intersects](#)

### 9.14.9 ST\_SameAlignment

**ST\_SameAlignment** — Retorna verdade se os rasters têm a mesma inclinação, escala, referência espacial, e deslocamento (pixels podem ser colocados na mesma grade sem cortar eles) e falso se eles não notificarem problemas detalhados.

## 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 );

**Descrição**

Versão não agregada (variantes 1 e 2): Retorna verdade se dois rasters (fornecidos diretamente ou feitos usando os valores esquerdo superior, escala, inclinação ou srid) têm a mesma escala, inclinação, srid e pelo menos um de qualquer dos quatro cantos de pixel de um raster cair em algum canto da grade do outro raster. Retorna falso se eles não e um AVISO detalhando o problema de alinhamento.

Versão agregada (variante 3): De um conjunto de rasters, retorna verdade se todos os rasters no conjunto estiverem alinhados. A função `ST_SameAlignment()` é "agregada" na terminologia do PostgreSQL. Isso significa que ela opera nas linhas de dados, da mesma maneira que as funções `SUM()` e `AVG()` operam.

Disponibilidade: 2.0.0

Melhorias: 2.1.0 adição da variante agregada

**Exemplos: Rasters**

```
SELECT ST_SameAlignment(
    ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
    ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0)
) as sm;

sm
----
t
```

```
SELECT ST_SameAlignment(A.rast,b.rast)
FROM dummy_rast AS A CROSS JOIN dummy_rast AS B;

NOTICE: The two rasters provided have different SRIDs
NOTICE: The two rasters provided have different SRIDs
 st_samealignment
-----
t
f
f
f
```

**Veja também**

Section 5.1, [ST\\_NotSameAlignmentReason](#), [ST\\_MakeEmptyRaster](#)

**9.14.10 ST\_NotSameAlignmentReason**

`ST_NotSameAlignmentReason` — Retorna a declaração de texto se os rasters estiverem alinhados e se não tiverem, uma razão do porquê.

**Synopsis**

```
text ST_NotSameAlignmentReason(raster rastA, raster rastB);
```

**Descrição**

Retorna a declaração de texto se os rasters estiverem alinhados e se não tiverem, uma razão do porquê.

**Note**

Se existem várias razões do porquê os rasters não estão alinhados, apenas uma razão (o primeiro teste a falhar) retornará.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT
  ST_SameAlignment (
    ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
    ST_MakeEmptyRaster(1, 1, 0, 0, 1.1, 1.1, 0, 0)
  ),
  ST_NotSameAlignmentReason(
    ST_MakeEmptyRaster(1, 1, 0, 0, 1, 1, 0, 0),
    ST_MakeEmptyRaster(1, 1, 0, 0, 1.1, 1.1, 0, 0)
  )
;

st_samealignment |          st_otsamealignmentreason
-----+-----
f                | The rasters have different scales on the X axis
(1 row)
```

**Veja também**

Section [5.1](#), [ST\\_SameAlignment](#)

**9.14.11 ST\_Within**

**ST\_Within** — Retorna verdade se nenhum ponto do raster rastA estiver no exterior do raster rastB e pelo menos um ponto do interior do rastA estiver no interior do rastB.

**Synopsis**

boolean **ST\_Within**( raster rastA , integer nbandA , raster rastB , integer nbandB );  
 boolean **ST\_Within**( raster rastA , raster rastB );

**Descrição**

O raster rastA está dentro do rastB se e somente se nenhum ponto do rastA estiver no exterior do rastB e pelo menos um ponto do interior do rastA estiver no interior do rastB. Se o número de banda não for fornecido (ou for NULL), apenas o casco convexo do raster será considerado no teste. Se o número de banda for fornecido, somente aqueles pixels com valor (não NODATA) são considerados no teste.

**Note**

Este operador fará uso de qualquer índice que pode estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, usa `ST_Polygon` no raster, ex.: `ST_Within(ST_Polygon(raster), geometria)` ou `ST_Within(geometria, ST_Polygon(raster))`.

**Note**

`ST_Within()` é o inverso da `ST_Contains()`. Logo, `ST_Within(rastA, rastB)` implica `ST_Contains(rastB, rastA)`.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT r1.rid, r2.rid, ST_Within(r1.rast, 1, r2.rast, 1) FROM dummy_rast r1 CROSS JOIN ↔
dummy_rast r2 WHERE r1.rid = 2;
```

rid	rid	st_within
2	1	f
2	2	t

**Veja também**

[ST\\_Intersects](#), [ST\\_Contains](#), [ST\\_DWithin](#), [ST\\_DFullyWithin](#)

**9.14.12 ST\_DWithin**

`ST_DWithin` — Retorna verdade se os rasters `rastA` e `rastB` estiverem dentro da distância especificada de cada um.

**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 );
```

**Descrição**

Retorna verdade se os rasters `rastA` e `rastB` estiverem dentro da distância especificada de cada um. Se o número de banda não for fornecido ( ou for NULO), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas aqueles pixels com valor (não NODATA) serão considerados no teste.

A distância é especificada em unidades definidas pelo sistema de referência espacial dos rasters. Para esta função fazer sentido, os rasters fonte devem ser ambos da mesma projeção de coordenada, tendo o mesmo SRID.

**Note**

Este operador fará uso de qualquer índice que pode estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, use `ST_Polygon` no raster, ex.: `ST_DWithin(ST_Polygon(raster), geometria)`.

Disponibilidade: 2.1.0

**Exemplos**

```
SELECT r1.rid, r2.rid, ST_DWithin(r1.rast, 1, r2.rast, 1, 3.14) FROM dummy_rast r1 CROSS JOIN dummy_rast r2 WHERE r1.rid = 2;
```

rid	rid	st_dwithin
2	1	f
2	2	t

**Veja também**

[ST\\_Within](#), [ST\\_DFullyWithin](#)

**9.14.13 ST\_DFullyWithin**

`ST_DFullyWithin` — Retorna verdade se os rasters `rastA` e `rastB` estiverem completamente dentro da distância especificada de cada um.

**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` );

**Descrição**

Retorna verdade se os rasters `rastA` e `rastB` estiverem completamente dentro da distância especificada de cada um. Se o número de banda não for fornecido (ou for NULO), apenas o casco convexo do raster é considerado no teste. Se o número de banda for fornecido, apenas aqueles pixels com valor (não NODATA) serão considerados no teste.

A distância é especificada em unidades definidas pelo sistema de referência espacial dos rasters. Para esta função fazer sentido, os rasters fonte devem ser ambos da mesma projeção de coordenada, tendo o mesmo SRID.

**Note**

Este operador fará uso de qualquer índice que pode estar disponível nos rasters.

**Note**

Para testar a relação espacial de um raster e uma geometria, use `ST_Polygon` no raster, ex.: `ST_DFullyWithin(ST_Polygon(raster), geometria)`.

Disponibilidade: 2.1.0

## Exemplos

```
SELECT r1.rid, r2.rid, ST_DFullyWithin(r1.rast, 1, r2.rast, 1, 3.14) FROM dummy_rast r1 ←
  CROSS JOIN dummy_rast r2 WHERE r1.rid = 2;
```

```
rid | rid | st_dfullywithin
-----+-----+-----
  2 |  1 | f
  2 |  2 | t
```

## Veja também

[ST\\_Within](#), [ST\\_DWithin](#)

## 9.15 Raster Tips

### 9.15.1 Out-DB Rasters

#### 9.15.1.1 Directory containing many files

When GDAL opens a file, GDAL eagerly scans the directory of that file to build a catalog of other files. If this directory contains many files (e.g. thousands, millions), opening that file becomes extremely slow (especially if that file happens to be on a network drive such as NFS).

To control this behavior, GDAL provides the following environment variable: [GDAL\\_DISABLE\\_READDIR\\_ON\\_OPEN](#). Set `GDAL_DISABLE_READDIR_ON_OPEN` to `TRUE` to disable directory scanning.

In Ubuntu (and assuming you are using PostgreSQL's packages for Ubuntu), `GDAL_DISABLE_READDIR_ON_OPEN` can be set in `/etc/postgresql/POSTGRESQL_VERSION/CLUSTER_NAME/environment` (where `POSTGRESQL_VERSION` is the version of PostgreSQL, e.g. 9.6 and `CLUSTER_NAME` is the name of the cluster, e.g. maindb). You can also set PostGIS environment variables here as well.

```
# 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'
```

#### 9.15.1.2 Maximum Number of Open Files

The maximum number of open files permitted by Linux and PostgreSQL are typically conservative (typically 1024 open files per process) given the assumption that the system is consumed by human users. For Out-DB Rasters, a single valid query can easily exceed this limit (e.g. a dataset of 10 year's worth of rasters with one raster for each day containing minimum and maximum temperatures and we want to know the absolute min and max value for a pixel in that dataset).

The easiest change to make is the following PostgreSQL setting: [max\\_files\\_per\\_process](#). The default is set to 1000, which is far too low for Out-DB Rasters. A safe starting value could be 65536 but this really depends on your datasets and the queries run against those datasets. This setting can only be made on server start and probably only in the PostgreSQL configuration file (e.g. `/etc/postgresql/POSTGRESQL_VERSION/CLUSTER_NAME/postgresql.conf` in Ubuntu environments).



```
...
# - Kernel Resource Usage -

max_files_per_process = 65536          # min 25
                                       # (change requires restart)
...
```

The major change to make is the Linux kernel's open files limits. There are two parts to this:

- Maximum number of open files for the entire system
- Maximum number of open files per process

#### 9.15.1.2.1 Maximum number of open files for the entire system

You can inspect the current maximum number of open files for the entire system with the following example:

```
$ sysctl -a | grep fs.file-max
fs.file-max = 131072
```

If the value returned is not large enough, add a file to `/etc/sysctl.d/` as per the following example:

```
$ 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
```

#### 9.15.1.2.2 Maximum number of open files per process

We need to increase the maximum number of open files per process for the PostgreSQL server processes.

To see what the current PostgreSQL service processes are using for maximum number of open files, do as per the following example (make sure to have PostgreSQL running):

```
$ ps aux | grep postgres
postgres 31713  0.0  0.4 179012 17564 pts/0    S   Dec26   0:03 /home/dustymugs/devel/ ↔
    postgresql/sandbox/10/usr/local/bin/postgres -D /home/dustymugs/devel/postgresql/sandbox ↔
    /10/pgdata
postgres 31716  0.0  0.8 179776 33632 ?        Ss  Dec26   0:01 postgres: checkpointer ↔
    process
postgres 31717  0.0  0.2 179144  9416 ?        Ss  Dec26   0:05 postgres: writer process
postgres 31718  0.0  0.2 179012  8708 ?        Ss  Dec26   0:06 postgres: wal writer ↔
    process
postgres 31719  0.0  0.1 179568  7252 ?        Ss  Dec26   0:03 postgres: autovacuum ↔
    launcher process
postgres 31720  0.0  0.1  34228  4124 ?        Ss  Dec26   0:09 postgres: stats collector ↔
    process
postgres 31721  0.0  0.1 179308  6052 ?        Ss  Dec26   0:00 postgres: bgworker: ↔
    logical replication launcher

$ cat /proc/31718/limits
```

Limit	Soft Limit	Hard Limit	Units
Max cpu time	unlimited	unlimited	seconds
Max file size	unlimited	unlimited	bytes
Max data size	unlimited	unlimited	bytes
Max stack size	8388608	unlimited	bytes
Max core file size	0	unlimited	bytes
Max resident set	unlimited	unlimited	bytes
Max processes	15738	15738	processes
<b>Max open files</b>	<b>1024</b>	<b>4096</b>	<b>files</b>
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

In the example above, we inspected the open files limit for Process 31718. It doesn't matter which PostgreSQL process, any of them will do. The response we are interested in is *Max open files*.

We want to increase *Soft Limit* and *Hard Limit* of *Max open files* to be greater than the value we specified for the PostgreSQL setting `max_files_per_process`. In our example, we set `max_files_per_process` to 65536.

In Ubuntu (and assuming you are using PostgreSQL's packages for Ubuntu), the easiest way to change the *Soft Limit* and *Hard Limit* is to edit `/etc/init.d/postgresql` (SysV) or `/lib/systemd/system/postgresql*.service` (systemd).

Let's first address the SysV Ubuntu case where we add `ulimit -H -n 262144` and `ulimit -n 131072` to `/etc/init.d/postgresql`.

```
...
case "$1" in
  start|stop|restart|reload)
    if [ "$1" = "start" ]; then
      create_socket_directory
    fi
    if [ -z "`pg_lsclusters -h`" ]; then
      log_warning_msg 'No PostgreSQL clusters exist; see "man pg_createcluster"'
      exit 0
    fi

    ulimit -H -n 262144
    ulimit -n 131072

    for v in $versions; do
      $1 $v || EXIT=$?
    done
    exit ${EXIT:-0}
    ;;
status)
...

```

Now to address the systemd Ubuntu case. We will add `LimitNOFILE=131072` to every `/lib/systemd/system/postgresql*.service` file in the **[Service]** section.

```
...
[Service]

LimitNOFILE=131072

...

[Install]
WantedBy=multi-user.target

```

...

After making the necessary systemd changes, make sure to reload the daemon

```
systemctl daemon-reload
```

## Chapter 10

# Perguntas frequentes PostGIS Raster

1. *Onde posso saber mais sobre o projeto PostGIS Raster?*

Consulte o sítio [PostGIS Raster](#) .

2. *Existem livros ou tutoriais para aprender sobre esta maravilhosa invenção?*

Existe um tutorial completo para iniciantes [Interseção de buffers de vetores com grande cobertura de varredura usando PostGIS Raster](#). Jorge tem uma série de artigos no blog do PostGIS Raster que demonstram como carregar dados raster bem como comparar as mesmas tarefas no Oracle GeoRaster. Confira [Artigos de Jorge: PostGIS Raster / Oracle GeoRaster](#). Há um capítulo inteiro (mais de 35 páginas de conteúdo) dedicado a PostGIS Raster com código livre e dados para download em [PostGIS em Ação - Capítulo Raster](#). Você pode [comprar a cópia impressa PostGIS em Ação](#) agora de Manning (descontos significativos para compras em massa) ou apenas o formato E-book. Você também pode comprar na Amazon e outros distribuidores de livros. Todos os manuais impressos vêm com um cupom para baixar a versão E-book. Here is a review from a PostGIS Raster user [PostGIS raster applied to land classification urban forestry](#)

3. *Como instalar o suporte a raster no meu banco de dados PostGIS?*

Starting with PostGIS 2.0 PostGIS Raster is fully integrated, so it will be compiled when you compile PostGIS. Instruções para instalação no Windows estão disponíveis em [Como instalar e configurar o PostGIS Raster no Windows](#). Se você é um usuário Windows, você pode compilar ou utilizar [os binários pré-compilados](#). Se você é um usuário Mac OSX Leopard ou Snow Leopard, existem binários disponíveis em [binários Kyng Chaos Mac OSX PostgreSQL/GIS](#). For other platforms, install PostGIS from your software repository. For more details about compiling from source, please refer to [Installing PostGIS Raster from source](#)

4. *Recebo um erro que não foi possível carregar a biblioteca "C:/Program Files/PblostgreSQL/8.4/lib/rtpostgis.dll": O módulo especificado não pode ser encontrado ou não foi possível carregar a biblioteca no linux ao executar rtpostgis.sql*

rtpostgis.so/dll é construído com dependências da libgdal.so/dll. Tenha certeza no Windows que você tenha o arquivo libgdal-1.dll na pasta bin de sua instalação do PostgreSQL. Para Linux, a libgdal deve estar no PATH ou na pasta bin. Você também pode encontrar erros diferentes se você não tem o PostGIS instalado em seu banco de dados. Tenha certeza de instalar primeiro o PostGIS em seu banco de dados antes de tentar instalar o suporte a raster.

5. *Como carrego dados raster dentro de meu banco PostGIS?*

The latest version of PostGIS comes packaged with a `raster2pgsql` raster loader executable capable of loading many kinds of rasters and also generating lower resolution overviews without any additional software. Please refer to Section [5.1.1](#) for more details.

6. *Quais tipos de arquivos raster posso carregar em meu banco de dados?*

Qualquer raster que sua biblioteca GDAL suporte. Os formatos suportados pela GDAL estão documentados em [GDAL File Formats](#). Sua instalação específica da GDAL pode não suportar todos os formatos. Para verificar os formatos suportados em sua instalação, você pode usar

```
raster2pgsql -G
```

### 7. Posso exportar meus dados raster do banco de dados para outros formatos raster?

SimGDAL has a PostGIS raster driver, but is only compiled in if you choose to compile with PostgreSQL support. O driver não suporta rasters irregulares, apesar de ser possível de armazená-los no tipo de dados do PostGIS. Se você está compilando os fontes, você precisa incluir em sua configuração

```
--with-pg=caminho/para/pg_config
```

para habilitar o driver. Veja [GDAL Build Hints](#) para dicas sobre como compilar a GDAL em várias plataformas e sistemas operacionais. Se sua versão da GDAL for compilada com o driver PostGIS, você deve ver PostGIS Raster na lista quando executar

```
gdalinfo --formats
```

Para visualizar um sumário sobre seu raster via GDAL use gdalinfo:

```
gdalinfo "PG:host=localhost port=5432 dbname='mygisdb' user='postgres' password=' ←
whatever' schema='someschema' table=sometable"
```

Para exportar dados para outros formatos raster, use gdal\_translate. O comando abaixo irá exportar todos os dados de uma tabela para um arquivo PNG com 10% de seu tamanho. Dependendo do tipo de bandas, algumas conversões (via GDAL) podem não funcionar, se o formato não suportar este tipo de banda. Por exemplo, bandas de ponto flutuante e inteiros de 32 bits sem sinal não serão facilmente convertidas em JPGs e alguns outros. Aqui está um exemplo de uma simples conversão

```
gdal_translate -of PNG -outsize 10% 10% "PG:host=localhost port=5432 dbname='mygisdb' ←
user='postgres' password='whatever' schema='someschema' table=sometable" C:\ ←
somefile.png
```

Você também pode usar cláusulas SQL na sua exportação, com o parametro where=... em sua string de conexão. Abaixo estão alguns exemplos com a cláusula where

```
gdal_translate -of PNG -outsize 10% 10% "PG:host=localhost port=5432 dbname='mygisdb' ←
user='postgres' password='whatever' schema='someschema' table=sometable where=' ←
filename='abcd.sid\'' " C:\somefile.png
```

```
gdal_translate -of PNG -outsize 10% 10% "PG:host=localhost port=5432 dbname='mygisdb' ←
user='postgres' password='whatever' schema='someschema' table=sometable where=' ←
ST_Intersects(rast, ST_SetSRID(ST_Point(-71.032,42.3793),4326) )' " C:\ ←
intersectregion.png
```

Para visualizar mais exemplos e a sintaxe, confira a seção [Reading Raster Data of PostGIS Raster section](#)

### 8. Existem binários pré-compilados já com suporte ao PostGIS Raster?

Sim. Cheque a página [GDAL Binaries](#). Qualquer um destes deve ter suporte ao PostGIS Raster. O PostGIS Raster está passando por muitas mudanças. Se você desejar a versão diária para Windows, cheque as builds feitas por Tamas Szekeres com Visual Studio, que contém o trunk GDAL, suporte a Python, MapServer e o driver PostGIS embutido. Clique no bat SDK e rode seus comandos a partir daí. <http://www.gisinternals.com>. Também estão disponíveis os arquivos de projetos do Visual Studio. [Última versão estável para Windows é compilada com suporte ao PostGIS Raster](#)

### 9. Quais ferramentas posso usar para visualizar os dados do PostGIS Raster?

You can use MapServer compiled with GDAL 1.7+ and PostGIS Raster driver support to view Raster data. QGIS supports viewing of PostGIS Raster if you have PostGIS raster driver installed. Na teoria, qualquer ferramenta que consiga reproduzir dados utilizando GDAL pode suportar os dados PostGIS raster com pouco esforço. Novamente para Windows, os binários Tamas <http://www.gisinternals.com> são uma boa escolha se você não quiser o transtorno de configurar para compilar por conta própria.

### 10. Como posso adicionar uma camada PostGIS Raster em meu mapa do MapServer?

Primeiro você precisa da GDAL 1.7 ou maior, compilada com suporte ao PostGIS Raster. GDAL 1.8 ou maior é preferida, já que muitos problemas foram solucionados e muitas pendências com o PostGIS Raster resolvidas na versão trunk. Você pode gostar muito do que pode ser feito com qualquer outro raster. Referência [MapServer Raster processing options](#) para

listar várias funções de processamento que você pode utilizar com MapServer raster layers. O que torna o PostGIS Raster tão interessante, é que cada tile pode ocupar diversas colunas padrão e você pode segmentar sua fonte de dados. Abaixo está um exemplo de como você pode definir uma camada PostGIS Raster no MapServer.

**Note**

O `mode=2` é obrigatório para rasters divididos em tiles e este suporte foi adicionado no PostGIS 2.0 e nos drivers GDAL 1.8. Este suporte não existe na versão GDAL 1.7.

```
-- displaying raster with standard raster options
LAYER
    NAME coolwktraster
    TYPE raster
    STATUS ON
    DATA "PG:host=localhost port=5432 dbname='somedb' user='someuser' password=' ←
        whatever'
        schema='someschema' table='cooltable' mode='2'"
    PROCESSING "NODATA=0"
    PROCESSING "SCALE=AUTO"
    #... other standard raster processing functions here
    #... classes are optional but useful for 1 band data
    CLASS
        NAME "boring"
        EXPRESSION ([pixel] < 20)
        COLOR 250 250 250
    END
    CLASS
        NAME "mildly interesting"
        EXPRESSION ([pixel] > 20 AND [pixel] < 1000)
        COLOR 255 0 0
    END
    CLASS
        NAME "very interesting"
        EXPRESSION ([pixel] >= 1000)
        COLOR 0 255 0
    END
END
```

```
-- displaying raster with standard raster options and a where clause
LAYER
    NAME soil_survey2009
    TYPE raster
    STATUS ON
    DATA "PG:host=localhost port=5432 dbname='somedb' user='someuser' password=' ←
        whatever'
        schema='someschema' table='cooltable' where='survey_year=2009' mode ←
        ='2'"
    PROCESSING "NODATA=0"
    #... other standard raster processing functions here
    #... classes are optional but useful for 1 band data
END
```

11. *Quais funcionalidades posso usar atualmente em meus dados raster?*

Se refere a lista Chapter 9. Existem mais, mas ainda está trabalhando na melhoria. Se refere a [PostGIS Raster roadmap page](#) para detalhes do que você pode esperar para o futuro.

12. *Estou recebendo um erro ERROR: function st\_intersects(raster, unknow) is not unique or st\_union(geometry, text) is not unique. Como posso consertar este problema?*

The function is not unique error happens if one of your arguments is a textual representation of a geometry instead of a geometry. In these cases, PostgreSQL marks the textual representation as an unknown type, which means it can fall into the `st_intersects(raster, geometry)` or `st_intersects(raster,raster)` thus resulting in a non-unique case since both functions can in theory support your request. To prevent this, you need to cast the textual representation of the geometry to a geometry. Por exemplo, se seu código se parece com isto:

```
SELECT rast
FROM my_raster
WHERE ST_Intersects(rast, 'SRID=4326;POINT(-10 10)');
```

Converta a representação textual geométrica para uma geometria, alterando seu código assim:

```
SELECT rast
FROM my_raster
WHERE ST_Intersects(rast, 'SRID=4326;POINT(-10 10)::geometry');
```

### 13. Como o PostGIS Raster é diferente do tipo Oracle GeoRaster (SDO\_GEOASTER) e do tipo SDO\_RASTER?

Para uma discussão mais extensa sobre esse tópico, verifique Jorge Arévalo [Oracle GeoRaster e PostGIS Raster: Primeiras impressões](#) A maior vantagem do one-georeference-by-raster sobre one-georeference-by-layer é permitir: \* coberturas não são necessariamente retangulares (que é frequentemente o caso do raster coverage cobrindo grandes extensões. Veja a possibilidade de arranjos raster na documentação)\* rasters para sobreposição (que é necessário para implementar perda de menos vetores para conversões raster) Estes arranjos também são possíveis no Oracle, mas eles implicam no armazenamento de múltiplos objetos SDO\_GEOASTER conectados a muitas tabelas SDO\_RASTER. Uma cobertura complexa pode liderar para centenas de tabelas no banco de dados. Com PostGIS Raster você pode gravar um arranjo raster similar dentro de uma única tabela. É um pouco como se o PostGIS forçasse você a gravar apenas cobertura vetorial retangular cheia sem lacunas e sobreposições (uma perfeita camada topologica retangular). Isso é muito prático em algumas aplicações, mas na prática tem mostrado que não é realista ou desejável para a maioria das cobertura geográficas. Estruturas vetoriais necessitam da flexibilidade para gravar coberturas descontínuas e não retangulares. Nós acreditamos que uma grande vantagem que estruturas raster deveriam beneficiar também.

### 14. a carga de grandes arquivos com raster2pgsql falha com String de N bytes é muito longa para conversão de encoding?

O raster2pgsql não faz conexões com o banco de dados enquanto está gerando o arquivo para carga. Se seu banco de dados tem um encoding cliente explicitamente configurado, então enquanto estiver realizando a carga de arquivos raster grandes (acima de 30 MB em tamanho), você pode encontrar uma mensagem `bytes is too long for encoding conversion`. Isto geralmente acontece se seu banco de dados é UTF8, mas para suportar aplicações clientes Windows, você configurou o encoding cliente para WIN1252. Para resolver este problema durante a carga, tenha certeza que o encoding cliente é o mesmo do seu banco de dados. Você pode fazer isto explicitamente no script de carga. Exemplo, se você usa Windows:

```
set PGCLIENTENCODING=UTF8
```

Se você está no Unix/Linux

```
export PGCLIENTENCODING=UTF8
```

Os detalhes desta situação estão detalhados em: <http://trac.osgeo.org/postgis/ticket/2209>

### 15. Estou tendo erro ERRO: RASTER\_fromGDALRaster: Não pôde abrir bytea comGDAL. Certifique que o bytea é de um formato GDAL suportado. quando usando ST\_FromGDALRaster ou ERRO: rt\_raster Não pôde carregar a saída GDAL do dispositivo quando tentou usar ST\_AsPNG ou outras funções de entrada raster.

Assim como PostGIS 2.1.3 e 2.0.5, uma alteração de segurança foi feita em todos os drivers GDAL e db rasters. Essas notas de liberação estão em [PostGIS 2.0.6, 2.1.3 liberação de segurança](#). Com o propósito de reativar drivers específicos ou todos os drivers e reativar fora do suporte do banco de dados, consulte Section [2.1](#).

## Chapter 11

# Topologia

Os tipos e as funções de topologia do PostGIS são usados para administrar objetos como: faces, bordas e nodos.

Sandro Santilli's presentation at PostGIS Day Paris 2011 conference gives a good synopsis of PostGIS Topology and where it is headed [Topology with PostGIS 2.0 slide deck](#).

Vicent Picavet fornece uma boa sinopse e panorama do que é topologia, como é usada e várias ferramentas FOSS4G que a suportam em [PostGIS Topology PGConf EU 2012](#).

Um exemplo de um banco de dados GIS baseado topologicamente é o banco de dados [US Census Topologically Integrated Geographic Encoding and Referencing System \(TIGER\)](#). Se você quiser experimentar com a topologia PostGIS e precisa de alguns dados, confira [Topology\\_Load\\_Tiger](#).

O módulo PostGIS Topologia existiu em versões anteriores, mas nunca foi parte da documentação Oficial do PostGIS. A maior limpeza PostGIS 2.0.0, vai remover todas as funções menores, consertar problemas de usabilidade, vai documentar melhor as características e funções e melhorar a conformidade com os padrões SQL-MM.

Detalhes deste projeto podem ser encontrados em [PostGIS Topology Wiki](#)

Todas as funções e tables associadas com este módulo estão instaladas em um esquema nomeado `topology`.

Funções que são definidas no padrão SQL/MM estão prefixadas com `ST_` e funções específicas para o PostGIS não estão prefixadas.

Topology support is build by default starting with PostGIS 2.0, and can be disabled specifying `--without-topology` configure option at build time as described in [Chapter 2](#)

### 11.1 Tipos de topologia

#### 11.1.1 `getfaceedges_returntype`

`getfaceedges_returntype` — Um tipo composto que consiste em um número sequência e um número limite. Este é o tipo retorno para `ST_GetFaceEdges`

##### Descrição

Um tipo composto que consiste em um número sequência e um número limite. Este é o tipo retorno para a função `ST_GetFaceEdges`.

1. `sequence` é um inteiro: Refere-se a uma topologia definida na table `topology.topology` que define o esquema e srid da topologia.
2. `edge` é um inteiro: O identificador de um limite.



### 11.1.2 TopoGeometry

TopoGeometry — Um tipo composto representando uma geometria topologicamente definida

#### Descrição

Um tipo composto que refere-se a uma geometria de topologia em uma camada específica da topologia, tendo um tipo e id específicos. Os elementos de uma TopoGeometry são as propriedades: `topology_id`, `layer_id`, `inteira id`, `inteira do tipo`.

1. `topology_id` é um inteiro: Refere-se a uma topologia definida na table `topology.topology` que define o esquema e `srid` da topologia.
2. `layer_id` é um inteiro: A `layer_id` nas `layers` tables que a TopoGEometry pertence. A combinação de `topology_id`, `layer_id` fornece uma referência única na table `topology.layers`.
3. `id` é um inteiro: a identidade é o número sequência auto gerado que define a topogeometry na respectiva camada da topologia.
4. `type` inteiro entre 1 - 4 that define o tipo da geometria: 1:[multi]ponto, 2:[multi]linha, 3:[multi]poly, 4:coleção

#### Comportamento Casting

Esta seção lista os casts automáticos bem como os explícitos permitidos para esse tipo de dados

Cast To	Comportamento
geometria	automático

#### Veja também

[CreateTopoGeom](#)

### 11.1.3 validateTopology\_returnType

`validateTopology_returnType` — Um tipo composto que consiste em um mensagem de erro e `id1` e `id2` para indicar a localização do erro. Este é o tipo retorno para `ValidateTopology`

#### Descrição

Um tipo composto que consiste em uma mensagem de erro e dois inteiros. A função `ValidateTopology` retorna um conjunto para indicar erros de validação e a `id1` e `id2` para indicar as ids dos objetos da topologia envolvidas no erro.

1. `error` é varchar: Indica tipo de erro.  
A descrições de erro atuais são: nós coincidentes, limite cruza nó, limite não simples, geometria limite e nó que não combinam, limite começa e a geometria nó não combina, face sobrepõe face, face dentro de face,
2. `id1` é um inteiro: Indica identificador de limite / face / nós no erro.
3. `id2` é um inteiro: Para erros que envolvem limite / ou nó secundário

#### Veja também

[ValidateTopology](#)

## 11.2 Domínios de Topologia

### 11.2.1 TopoElement

TopoElement — Um arranjo de 2 inteiros geralmente usado para identificar um componente TopoGeometry.

#### Descrição

Um arranjo de 2 inteiros usados para representar um componente de um simples ou hierárquico **TopoGeometry**.

No caso de de uma TopoGeometria simples, o primeiro elemento do arranjo representa o identificador de um topológico primitivo, e o segundo elemento representa o tipo dele (1:nó, 2:limite, 3:face). No caso de uma TopoGeometria hierárquica o primeiro elemento do arranjo representa o identificador de uma TopoGeometria filha e o segundo elemento representa seu identificador de camada.



#### Note

Para qualquer uma das TopoGeometrias hierárquicas dadas, todos os elementos das TopoGeometrias filhas virão da mesma camada, assim com está especificado no relato topology.layer para a camada da TopoGeometria que está sendo definida.

#### Exemplos

```
SELECT te[1] AS id, te[2] AS type FROM
( SELECT ARRAY[1,2]::topology.topoelement AS te ) f;

id | type
----+-----
1  | 2
```

```
SELECT ARRAY[1,2]::topology.topoelement;

te
-----
{1,2}
```

```
--Example of what happens when you try to case a 3 element array to topoelement
-- NOTE: topoement has to be a 2 element array so fails dimension check
SELECT ARRAY[1,2,3]::topology.topoelement;
ERROR:  value for domain topology.topoelement violates check constraint "dimensions"
```

#### Veja também

[GetTopoGeomElements](#), [TopoElementArray](#), [TopoGeometry](#), [TopoGeom\\_addElement](#), [TopoGeom\\_remElement](#)

### 11.2.2 TopoElementArray

TopoElementArray — Um banco de dados de objetos TopoElement

**Descrição**

Um arranjo de 1 ou mais objetos TopoGeometria, geralmente usado para circular componentes dos objetos de TopoGeometrias.

**Exemplos**

```
SELECT '{{1,2},{4,3}}'::topology.topoelementarray As tea;

tea
-----
{{1,2},{4,3}}

-- more verbose equivalent --
SELECT ARRAY[ARRAY[1,2], ARRAY[4,3]]::topology.topoelementarray As tea;

tea
-----
{{1,2},{4,3}}

--using the array agg function packaged with topology --
SELECT topology.TopoElementArray_Agg(ARRAY[e,t]) As tea
FROM generate_series(1,4) As e CROSS JOIN generate_series(1,3) As t;

tea
-----
{{1,1},{1,2},{1,3},{2,1},{2,2},{2,3},{3,1},{3,2},{3,3},{4,1},{4,2},{4,3}}

SELECT '{{1,2,4},{3,4,5}}'::topology.topoelementarray As tea;
ERRO: valor para o domínio topology.topoelementarray viola a chave "dimensions"
```

**Veja também**

[TopoElement](#), [GetTopoGeomElementArray](#), [TopoElementArray\\_Agg](#)

**11.3 Gerenciamento de Topologia e TopoGeometria****11.3.1 AddTopoGeometryColumn**

AddTopoGeometryColumn — Adiciona uma coluna topogeometria a uma table, registra essa coluna nova como uma camada topology.layer e retorna a nova layer\_id.

## Synopsis

```
integer AddTopoGeometryColumn(varchar topology_name, varchar schema_name, varchar table_name, varchar column_name,
varchar feature_type);
integer AddTopoGeometryColumn(varchar topology_name, varchar schema_name, varchar table_name, varchar column_name,
varchar feature_type, integer child_layer);
```

## Descrição

Cada objeto TopoGeometria pertence à uma camada específica de uma Topologia específica. Antes de criar tal objeto, você precisa criar sua TopologyLayer. uma Camada de Topologia é uma associação de feature-table com a topologia. Também contém informações de tipo de hierárquicas. Nós criamos uma camada usando a função AddTopoGeometryColumn():

Esta função irá adicionar a coluna pedida e um relato para a table topology.layer com todas as informações dadas.

Se você não especificar [child\_layer] (ou configurar para NULO) essa camada irá conter TopoGeometrias Básicas (compostas por elementos de topologia primitivos). Senão essa camada conterá TopoGeometrias hierárquicas (compostas por TopoGeometrias da child\_layer).

Uma vez que a camada é criada (sua id retorna através da função AddTopoGeometryColumn) você pode construir objetos TopoGeometria nela.

feature\_types válidos são: PONTO, LINHA, POLÍGONO, COLEÇÃO

Disponibilidade: 1.?

## Exemplos

```
-- Note for this example we created our new table in the ma_topo schema
-- though we could have created it in a different schema -- in which case topology_name and ←
  schema_name would be different
CREATE SCHEMA ma;
CREATE TABLE ma.parcels(gid serial, parcel_id varchar(20) PRIMARY KEY, address text);
SELECT topology.AddTopoGeometryColumn('ma_topo', 'ma', 'parcels', 'topo', 'POLYGON');
```

```
CREATE SCHEMA ri;

CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);

SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

## Veja também

[Cria topologia](#), [CreateTopoGeom](#)

### 11.3.2 DropTopology

DropTopology — Cuidado ao usar: Derruba um esquema topologia e deleta sua referência da table topology.topology e referências para tables naquele esquema da table geometry\_columns.

## Synopsis

```
integer DropTopology(varchar topology_schema_name);
```

## Descrição

Derruba um esquema topologia e deleta sua referência da table `topology.topology` e referências para tables naquele esquema da table `geometry_columns`. Esta função deve ser USADA COM CUIDADO, ela pode destruir algum dado importante. Se o esquema não existir, ela só remove entradas de referência do esquema nomeado.

Disponibilidade: 1.?

## Exemplos

Cascata derruba o esquema `ma_topo` e remove todas as referências no `topology.topology` e `geometry_columns`.

```
SELECT topology.DropTopology('ma_topo');
```

## Veja também

### 11.3.3 DropTopoGeometryColumn

`DropTopoGeometryColumn` — Derruba a coluna topogeometria da table nomeada `table_name` no esquema `schema_name` e tira os registros da

## Synopsis

text **DropTopoGeometryColumn**(varchar `schema_name`, varchar `table_name`, varchar `column_name`);

## Descrição

Derruba a coluna topogeometria da table nomeada `table_name` no esquema `schema_name` e tira os registros da colunas da table `topology.layer`. Retorna um resumo do drop status. **NOTA:** ela primeiro configura todos os valores para NULO antes de derrubar checks de integridade referencial.

Disponibilidade: 1.?

## Exemplos

```
SELECT topology.DropTopoGeometryColumn('ma_topo', 'parcel_topo', 'topo');
```

## Veja também

[AddTopoGeometryColumn](#)

### 11.3.4 Populate\_Topology\_Layer

`Populate_Topology_Layer` — Adds missing entries to `topology.layer` table by reading metadata from topo tables.

## Synopsis

setof record **Populate\_Topology\_Layer**();

---

**Descrição**

Adds missing entries to the `topology.layer` table by inspecting topology constraints on tables. This function is useful for fixing up entries in topology catalog after restores of schemas with topo data.

It returns the list of entries created. Returned columns are `schema_name`, `table_name`, `feature_column`.

Disponibilidade: 2.3.0

**Exemplos**

```
SELECT CreateTopology('strk_topo');
CREATE SCHEMA strk;
CREATE TABLE strk.parcels(gid serial, parcel_id varchar(20) PRIMARY KEY, address text);
SELECT topology.AddTopoGeometryColumn('strk_topo', 'strk', 'parcels', 'topo', 'POLYGON');
-- this will return no records because this feature is already registered
SELECT *
  FROM topology.Populate_Topology_Layer();

-- let's rebuild
TRUNCATE TABLE topology.layer;

SELECT *
  FROM topology.Populate_Topology_Layer();

SELECT topology_id, layer_id, schema_name As sn, table_name As tn, feature_column As fc
FROM topology.layer;
```

```
schema_name | table_name | feature_column
-----+-----+-----
strk        | parcels    | topo
(1 row)

topology_id | layer_id | sn | tn | fc
-----+-----+-----+-----+-----
          2 |         2 | strk | parcels | topo
(1 row)
```

**Veja também**

[AddTopoGeometryColumn](#)

**11.3.5 TopologySummary**

**TopologySummary** — Pega um nome de topologia e fornece totais resumidos de tipos dos objetos na topologia

**Synopsis**

```
text TopologySummary(varchar topology_schema_name);
```

**Descrição**

Pega um nome de topologia e fornece totais resumidos de tipos dos objetos na topologia.

Disponibilidade: 2.0.0

## Exemplos

```
SELECT topology.topologysummary('city_data');
           topologysummary
-----
Topology city_data (329), SRID 4326, precision: 0
22 nodes, 24 edges, 10 faces, 29 topogeoms in 5 layers
Layer 1, type Polygonal (3), 9 topogeoms
  Deploy: features.land_parcels.feature
Layer 2, type Puntal (1), 8 topogeoms
  Deploy: features.traffic_signs.feature
Layer 3, type Lineal (2), 8 topogeoms
  Deploy: features.city_streets.feature
Layer 4, type Polygonal (3), 3 topogeoms
  Hierarchy level 1, child layer 1
  Deploy: features.big_parcels.feature
Layer 5, type Puntal (1), 1 topogeoms
  Hierarchy level 1, child layer 2
  Deploy: features.big_signs.feature
```

## Veja também

[Topology\\_Load\\_Tiger](#)

### 11.3.6 ValidateTopology

`ValidateTopology` — Retorna um conjunto de objetos `validatetopology_returntype` detalhando issues com topologia

#### Synopsis

```
setof validatetopology_returntype ValidateTopology(varchar topology_schema_name);
```

#### Descrição

Retorna um conjunto de objetos `validatetopology_returntype` detalhando issues com topologia. Lista de erros possíveis e do significado das ids que retornaram, está expressa abaixo:

Erro	id1	id2
borda cruza nodo	edge_id	node_id
borda inválida	edge_id	null
borda não simples	edge_id	null
borda cruza borda	edge_id	edge_id
edge start node geometry mis-match	edge_id	node_id
edge end node geometry mis-match	edge_id	node_id
face sem bordas	face_id	null
face sem anéis	face_id	null
face sobrepõe face	face_id	face_id
face dentro de face	inner face_id	outer face_id

Disponibilidade: 1.0.0

Melhorias: 2.0.0 limite mais eficiente cruzando detenção e consertos para falsos positivos que existiam em versões anteriores.

Alterações: 2.2.0 valores para id1 e id2 foram trocados para "limite cruza nó", para serem consistentes com a descrição do erro.

## Exemplos

```
SELECT * FROM topology.ValidateTopology('ma_topo');

error | id1 | id2
-----+-----+-----
face without edges | 0 |
```

## Veja também

[validatetopology\\_returntype](#), [Topology\\_Load\\_Tiger](#)

## 11.4 Construtores de topologia

### 11.4.1 Cria topologia

Cria topologia — Cria um novo esquema topologia e registra esse novo esquema na tabela topology.topology.

#### Synopsis

```
integer CreateTopology(varchar topology_schema_name);
integer CreateTopology(varchar topology_schema_name, integer srid);
integer CreateTopology(varchar topology_schema_name, integer srid, double precision prec);
integer CreateTopology(varchar topology_schema_name, integer srid, double precision prec, boolean hasz);
```

#### Descrição

Creates a new schema with name `topology_name` consisting of tables (`edge_data`, `face`, `node`, `relation` and registers this new topology in the `topology.topology` table. It returns the id of the topology in the topology table. The `srid` is the spatial reference identified as defined in `spatial_ref_sys` table for that topology. Topologies must be uniquely named. The tolerance is measured in the units of the spatial reference system. If the tolerance (`prec`) is not specified defaults to 0.

Isto é parecido com SQL/MM [ST\\_InitTopoGeo](#), mas um pouco mais funcional. `hasz` se torna falso se não estiver especificado.

Disponibilidade: 1.?

## Exemplos

Este exemplo cria um novo esquema chamado `ma_topo` que irá armazenar limites, faces e relações nos metros do plano do estado de Massachusetts. A tolerância representa 1/2 metros já que o sistema de referência espacial é baseado em metros.

```
SELECT topology.CreateTopology('ma_topo', 26986, 0.5);
```

Cria uma topologia Rhode Island no Estado Plano ft

```
SELECT topology.CreateTopology('ri_topo', 3438) As topoid;
```

```
topoid
```

```
-----
```

```
2
```



## Veja também

Section [4.3.1](#), [ST\\_InitTopoGeo](#), [Topology\\_Load\\_Tiger](#)

## 11.4.2 CopyTopology

**CopyTopology** — Faz uma cópia da estrutura de uma topologia (nós, limites, faces, camadas e TopoGeometrias).

### Synopsis

integer **CopyTopology**(varchar existing\_topology\_name, varchar new\_name);

### Descrição

Cria uma nova topologia com nome `new_topology_name` e SRID e precisão de `existing_topology_name`, copia todos os nós, limites e faces existentes lá, copia camadas e suas TopoGeometrias também.



#### Note

As novas fileiras na `topology.layer` irão conter valores sintetizados para `schema_name`, `table_name` and `feature_column`. Isto se dá, porque a TopoGeometria só existe como uma definição, mas ainda não estará disponível em nenhuma user-level table.

Disponibilidade: 2.0.0

### Exemplos

Este exemplo faz um backup de uma topologia chamada `ma_topo`

```
SELECT topology.CopyTopology('ma_topo', 'ma_topo_bakup');
```

## Veja também

Section [4.3.1](#), [Cria topologia](#)

## 11.4.3 ST\_InitTopoGeo

**ST\_InitTopoGeo** — Cria um novo esquema topologia e registra esse novo esquema na table `topology.topology` e detalha um resumo do processo.

### Synopsis

text **ST\_InitTopoGeo**(varchar topology\_schema\_name);

### Descrição

Este é um SQL-MM equivalente do `CreateTopology`, mas falta a referência espacial e as opções de tolerância do `CreateTopology` e gera uma descrição de texto da criação em vez da id da topologia.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.17

## Exemplos

```
SELECT topology.ST_InitTopoGeo('topo_schema_to_create') AS topocreation;

astopocreation

-----

Topologia-Geometria 'topo_schema_to_create' (id:7) criada.
```

## Veja também

[Cria topologia](#)

### 11.4.4 ST\_CreateTopoGeo

**ST\_CreateTopoGeo** — Adiciona uma coleção de geometrias para uma dada topologia vazia e retorna uma mensagem detalhando sucesso.

#### Synopsis

text **ST\_CreateTopoGeo**(varchar atopology, geometry acollection);

#### Descrição

Adiciona uma coleção de geometrias para uma dada topologia vazia e retorna uma mensagem detalhando sucesso.

Útil para popular uma topologia vazia.

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details -- X.3.18

## Exemplos

```
-- Populate topology --
SELECT topology.ST_CreateTopoGeo('ri_topo',
  ST_GeomFromText('MULTILINESTRING((384744 236928,384750 236923,384769 236911,384799 ↵
    236895,384811 236890,384833 236884,
    384844 236882,384866 236881,384879 236883,384954 236898,385087 236932,385117 236938,
    385167 236938,385203 236941,385224 236946,385233 236950,385241 236956,385254 236971,
    385260 236979,385268 236999,385273 237018,385273 237037,385271 237047,385267 237057,
    385225 237125,385210 237144,385192 237161,385167 237192,385162 237202,385159 237214,
    385159 237227,385162 237241,385166 237256,385196 237324,385209 237345,385234 237375,
    385237 237383,385238 237399,385236 237407,385227 237419,385213 237430,385193 237439,
    385174 237451,385170 237455,385169 237460,385171 237475,385181 237503,385190 237521,
    385200 237533,385206 237538,385213 237541,385221 237542,385235 237540,385242 237541,
    385249 237544,385260 237555,385270 237570,385289 237584,385292 237589,385291 ↵
    237596,385284 237630))',3438)
  );

      st_createtopogeo
-----
Topology ri_topo populated
```

```
-- create tables and topo geometries --
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);

SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

### Veja também

[AddTopoGeometryColumn](#), [Cria topologia](#), [DropTopology](#)

## 11.4.5 TopoGeo\_AddPoint

TopoGeo\_AddPoint — Adiciona um ponto a uma topologia usando uma tolerância e possivelmente dividindo um limite existente.

### Synopsis

integer **TopoGeo\_AddPoint**(varchar toponame, geometry apoint, float8 tolerance);

### Descrição

Adiciona um ponto a uma topologia existente e retorna como identificador. O ponto romperá para nós ou limites existentes com a tolerância. Um limite existente pode ser dividido pelo ponto.

Disponibilidade: 2.0.0

### Veja também

[TopoGeo\\_AddLineString](#), [TopoGeo\\_AddPolygon](#), [AddNode](#), [Cria topologia](#)

## 11.4.6 TopoGeo\_AddLineString

TopoGeo\_AddLineString — Adiciona uma linestring a ua topologia existente usando uma tolerância e possivelmente dividindo limites/faces existentes. Retorna identificadores de limites

### Synopsis

SETOF integer **TopoGeo\_AddLineString**(varchar toponame, geometry aline, float8 tolerance);

### Descrição

Adiciona uma linestring a uma topologia existente e retorna um conjunto de identificadores formando ela. A linha romperá para nós ou limites existentes com a tolerância. Limites e faces existentes podem ser divididos pela linha.

Disponibilidade: 2.0.0

### Veja também

[TopoGeo\\_AddPoint](#), [TopoGeo\\_AddPolygon](#), [AddEdge](#), [Cria topologia](#)

---

### 11.4.7 TopoGeo\_AddPolygon

TopoGeo\_AddPolygon — Adiciona um polígono a uma topologia usando uma tolerância e possivelmente dividindo limites/faces.

#### Synopsis

integer **TopoGeo\_AddPolygon**(varchar atopology, geometry apoly, float8 atolerance);

#### Descrição

Adiciona um polígono a uma topologia e retorna um conjunto de identificadores de faces formando ele. O limite do polígono romperá nós ou limites dentro da tolerância. Limites e faces existentes podem ser divididos pelo limite do novo polígono.

Disponibilidade: 2.0.0

#### Veja também

[TopoGeo\\_AddPoint](#), [TopoGeo\\_AddLineString](#), [AddFace](#), [Cria topologia](#)

## 11.5 Editores de Topologia

### 11.5.1 ST\_AddIsoNode

ST\_AddIsoNode — Adiciona um nó isolado a uma face em uma topologia e retorna a id do novo nó. Se a face é nula, o nó continua sendo criado.

#### Synopsis

integer **ST\_AddIsoNode**(varchar atopology, integer aface, geometry apoint);

#### Descrição

Adiciona um nó isolado com a localização do ponto `apoint` com uma face existente com `faceid aface` a uma topologia `atopology` e retorna a `nodeid` do novo nó.

O sistema de referência espacial (`srid`) da geometria pontual não é o mesmo que a topologia, o `apoint` não é uma geometria pontual, o ponto é nulo, ou o ponto intersecta um limite existente (mesmo nos limites), então uma exceção é aberta. Se o ponto já existe como um nó, uma exceção é aberta.

Se `aface` não é nula e o `apoint` não está dentro da face, então, uma exceção é aberta.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X+1.3.1

#### Exemplos

#### Veja também

[AddNode](#), [Cria topologia](#), [DropTopology](#), [ST\\_Intersects](#)

---

## 11.5.2 ST\_AddIsoEdge

`ST_AddIsoEdge` — Adiciona um limite isolado definido pela geometria `alinestring` a uma topologia conectando dois nós isolados `anode` e `anothernode` e retorna a nova id do novo limite.

### Synopsis

```
integer ST_AddIsoEdge(varchar atopology, integer anode, integer anothernode, geometry alinestring);
```

### Descrição

Adiciona um limite isolado definido pela geometria `alinestring` a uma topologia conectando dois nós isolados `anode` e `anothernode` e retorna a nova id do novo limite.

Se o sistema de referência espacial (`srid`) da geometria `alinestring` não for o mesmo da topologia, qualquer argumento de entrada é nulo, ou is nós estão contidos em mais de uma face, ou eles são o começo ou fim de um limite existente, então, uma exceção é aberta.

Se a `alinestring` não está dentro da face da face o `anode` e `anothernode` pertence, então, uma exceção é aberta.

Se o `anode` e `anothernode` não são os pontos de começo e fim da `alinestring` então, uma exceção é aberta.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.4

### Exemplos

#### Veja também

[ST\\_AddIsoNode](#), [ST\\_IsSimple](#), [ST\\_Within](#)

## 11.5.3 ST\_AddEdgeNewFaces

`ST_AddEdgeNewFaces` — Adiciona um novo limite e, se uma face for dividida, deleta a face original e substitui por duas novas faces.

### Synopsis

```
integer ST_AddEdgeNewFaces(varchar atopology, integer anode, integer anothernode, geometry acurve);
```

### Descrição

Adiciona um novo limite e, se uma face for dividida, deleta a face original e substitui por duas novas faces. Retorna a id do novo limite adicionado.

Atualiza todos os limites existentes e relacionamentos em conformidade.

Se algum argumento for nulo, os nós são desconhecidos (devem existir na table `node` do esquema de topologia), a `acurve` não é uma `LINESTRING`, o `anode` e `anothernode` não são os pontos de começo e fim da `acurve`, logo, um erro é lançado.

Se o sistema de referência espacial (`srid`) da geometria `acurve` não for o mesmo da topologia, uma exceção é lançada.

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.12

## Exemplos

### Veja também

[ST\\_RemEdgeNewFace](#)

[ST\\_AddEdgeModFace](#)

## 11.5.4 ST\_AddEdgeModFace

`ST_AddEdgeModFace` — Adiciona um novo limite e, se uma face for dividida, modifica a face original e adiciona uma nova face.

### Synopsis

integer `ST_AddEdgeModFace`(varchar atopology, integer anode, integer anothernode, geometry acurve);

### Descrição

Adiciona um novo limite e, se uma face for dividida, modifica a face original e adiciona uma nova.



#### Note

Se possível, a face nova será criada no lado esquerdo do novo limite. Isto não será possível se a face do lado esquerdo precisar ser a face universal (sem limites).

Retorna a id do novo limite adicionado.

Atualiza todos os limites existentes e relacionamentos em conformidade.

Se algum argumento for nulo, os nós são desconhecidos (devem existir na table `node` do esquema de topologia), a `acurve` não é uma `LINESTRING`, o `anode` e `anothernode` não são os pontos de começo e fim da `acurve`, logo, um erro é lançado.

Se o sistema de referência espacial (`srid`) da geometria `acurve` não for o mesmo da topologia, uma exceção é lançada.

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.13

## Exemplos

### Veja também

[ST\\_RemEdgeModFace](#)

[ST\\_AddEdgeNewFaces](#)

## 11.5.5 ST\_RemEdgeNewFace

`ST_RemEdgeNewFace` — Remove um limite e, se o limite removido separava duas faces, deleta as faces originais e as substitui por uma nova face.

### Synopsis

integer `ST_RemEdgeNewFace`(varchar atopology, integer anedge);

## Descrição

Remove um limite *e*, se o limite removido separava duas faces, deleta as faces originais e as substitui por uma nova face.

Retorna a id de uma face nova criada ou NULA, se nenhuma face nova for criada. Nenhuma face nova é criada quando o limite removido está pendurado, isolado ou confinado na face universal (possivelmente fazendo a inundação universal dentro da face no outro lado).

Atualiza todos os limites existentes e relacionamentos em conformidade.

Refuses to remove an edge participating in the definition of an existing TopoGeometry. Refuses to heal two faces if any TopoGeometry is defined by only one of them (and not the other).

Se qualquer argumento for nulo, o limite dado é desconhecido (deve existir na table *edge* do esquema de topologia), o nome da topologia é inválido, logo, um erro é lançado.

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.14

## Exemplos

### Veja também

[ST\\_RemEdgeModFace](#)

[ST\\_AddEdgeNewFaces](#)

## 11.5.6 ST\_RemEdgeModFace

*ST\_RemEdgeModFace* — Remove um limite *e*, se o limite removido separou duas faces, deleta uma das duas e modifica a outra para pegar o espaço delas.

## Synopsis

```
integer ST_RemEdgeModFace(varchar atopology, integer anedge);
```

## Descrição

Remove um limite *e*, se o limite removido separou duas faces, deleta uma das duas e modifica a outra para pegar o espaço delas. Mantém, preferencialmente, a face que está na direita, para ser simétrico com a *ST\_AddEdgeModFace*, também mantendo-a. Retorna a id da face remanescente no lugar do limite removido.

Atualiza todos os limites existentes e relacionamentos em conformidade.

Recusa remover um limite que participa da definição de uma TopoGeometria. Recusa fechar duas faces se qualquer TopoGeometria estiver definida por apenas uma delas (e não a outra).

Se qualquer argumento for nulo, o limite dado é desconhecido (deve existir na table *edge* do esquema de topologia), o nome da topologia é inválido, logo, um erro é lançado.

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.15

## Exemplos

### Veja também

[ST\\_AddEdgeModFace](#)

[ST\\_RemEdgeNewFace](#)

## 11.5.7 ST\_ChangeEdgeGeom

ST\_ChangeEdgeGeom — Modifica a forma de um limite sem afetar a estrutura da topologia.

### Synopsis

integer **ST\_ChangeEdgeGeom**(varchar atopology, integer anedge, geometry acurve);

### Descrição

Modifica a forma de um limite sem afetar a estrutura da topologia.

Se algum argumento for nulo, o limite dado não existe na table `edge` do esquema da topologia, a `acurve` não é uma `LINESTRING`, o `anode` e `anothernode` não são os pontos de início e fim de `acurve` ou a modificação iria mudar a topologia fundamental, então, um erro é lançado.

Se o sistema de referência espacial (`srid`) da geometria `acurve` não for o mesmo da topologia, uma exceção é lançada.

Se a nova `acurve` não for simples, um erro é lançado.

Se mover o limite de uma posição antiga acertar um obstáculo, um erro é lançado.

Disponibilidade: 1.1.0

Melhorias: 2.0.0 adiciona execução da consistência topológica



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details X.3.6

### Exemplos

```
SELECT topology.ST_ChangeEdgeGeom('ma_topo', 1,
    ST_GeomFromText('LINESTRING(227591.9 893900.4,227622.6 893844.3,227641.6
    893816.6, 227704.5 893778.5)', 26986) );
-----
Edge 1 changed
```

### Veja também

[ST\\_AddEdgeModFace](#)

[ST\\_RemEdgeModFace](#)

[ST\\_ModEdgeSplit](#)

## 11.5.8 ST\_ModEdgeSplit

ST\_ModEdgeSplit — Divide um limite criando um novo nó junto de um limite existente, modificando o limite original e adicionando um novo limite.



**Synopsis**

integer **ST\_ModEdgeSplit**(varchar atopology, integer anedge, geometry apoint);

**Descrição**

Divide um limite criando um novo nó junto de um limite existente, modificando o limite original e adicionando um novo limite. Atualiza todos os limites e relacionamentos em conformidade. Retorna o identificador do novo nó adicionado.

Disponibilidade: 1.?

Alterações: 2.0 - Nas versões anteriores, isto recebia o nome errado ST\_ModEdgesSplit



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

**Exemplos**

```
-- Add an edge --
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227592 893910, 227600 893910)', 26986) ) As edgeid;

-- edgeid-
3

-- Split the edge --
SELECT topology.ST_ModEdgeSplit('ma_topo', 3, ST_SetSRID(ST_Point(227594,893910),26986) ) As node_id;
-----
node_id
7
```

**Veja também**

[ST\\_NewEdgesSplit](#), [ST\\_ModEdgeHeal](#), [ST\\_NewEdgeHeal](#), [AddEdge](#)

**11.5.9 ST\_ModEdgeHeal**

**ST\_ModEdgeHeal** — Fecha dois limites deletando o nó que os conecta, modificando o primeiro limite e deletando o segundo. Retorna a id do nó deletado.

**Synopsis**

int **ST\_ModEdgeHeal**(varchar atopology, integer anedge, integer anotheredge);

**Descrição**

Fecha dois limites deletando o nó que os conecta, modificando o primeiro limite e deletando o segundo. Retorna a id do nó deletado. Atualiza todos o limites e relacionamentos em conformidade.

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

**Veja também**

[ST\\_ModEdgeSplit](#) [ST\\_NewEdgesSplit](#)

**11.5.10 ST\_NewEdgeHeal**

`ST_NewEdgeHeal` — Fecha dois limites deletando o nó que os conecta, deletando ambos limites, e substituindo-os com um limite cuja direção é a mesma do primeiro limite fornecido.

**Synopsis**

```
int ST_NewEdgeHeal(varchar atopology, integer anedge, integer anotheredge);
```

**Descrição**

Fecha dois limites deletando o nó que os conecta, deletando ambos limites, e substituindo-os com um limite cuja direção é a mesma do primeiro limite fornecido. Retorna a id do novo limite, substituindo os fechados. Atualiza todos os limites e relacionamentos em conformidade.

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

**Veja também**

[ST\\_ModEdgeHeal](#) [ST\\_ModEdgeSplit](#) [ST\\_NewEdgesSplit](#)

**11.5.11 ST\_MoveIsoNode**

`ST_MoveIsoNode` — Moves an isolated node in a topology from one point to another. If new `apoint` geometry exists as a node an error is thrown. Returns description of move.

**Synopsis**

```
text ST_MoveIsoNode(varchar atopology, integer anedge, geometry apoint);
```

**Descrição**

Move um nó isolado em uma topologia de um ponto para outro. Se nova geometria `apoint` existe como um nó, um erro é lançado.

Se qualquer argumento for nulo, o `apoint` não é um ponto, o nó existente não é isolado (é um ponto de início ou fim de um limite), uma nova localização de nó intersecta um limite existente (mesmo em pontos finais), então uma exceção é lançada.

Se o sistema de referência espacial (`srid`) da geometria pontual não for o mesmo da topologia, uma exceção é lançada.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X.3.2

## Exemplos

```
-- Add an isolated node with no face --
SELECT topology.ST_AddIsoNode('ma_topo', NULL, ST_GeomFromText('POINT(227579 893916)', ←
  26986) ) As nodeid;
  nodeid
-----
      7
-- Move the new node --
SELECT topology.ST_MoveIsoNode('ma_topo', 7, ST_GeomFromText('POINT(227579.5 893916.5)', ←
  26986) ) As descrip;
                descrip
-----
Isolated Node 7 moved to location 227579.5,893916.5
```

## Veja também

[ST\\_AddIsoNode](#)

### 11.5.12 ST\_NewEdgesSplit

**ST\_NewEdgesSplit** — Divide um limite criando um novo nó ao longo do limite existente, deletando o limite original e substituindo-o por dois novos. Retorna a id do novo nó criado que integra os novos limites.

#### Synopsis

integer **ST\_NewEdgesSplit**(varchar atopology, integer anedge, geometry apoint);

#### Descrição

Divide um limite com uma id limite *anedge* criando um novo nó com uma localização de ponto *apoint* junto co *i* limite atual, deletando o limite original e substituindo-o por dois novos. Retorna a id do novo nó criado que se une aos novos limites. Atualiza todos os limites unidos e relacionamentos em conformidade.

Se o sistema de referência espacial (srid) da geometria pontual não é o mesmo que a topologia, o *apoint* não é uma geometria pontual, o ponto é nulo, o ponto já existe como um nó, o limite não corresponde a um limite existente ou o ponto não está dentro do limite, então, uma exceção é aberta.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X.3.8

## Exemplos

```
-- Add an edge --
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575 893917,227592 893900) ←
  ', 26986) ) As edgeid;
-- result-
edgeid
-----
      2
-- Split the new edge --
SELECT topology.ST_NewEdgesSplit('ma_topo', 2, ST_GeomFromText('POINT(227578.5 893913.5)', ←
  26986) ) As newnodeid;
  newnodeid
-----
      6
```

**Veja também**

[ST\\_ModEdgeSplit](#) [ST\\_ModEdgeHeal](#) [ST\\_NewEdgeHeal](#) [AddEdge](#)

**11.5.13 ST\_RemoveIsoNode**

`ST_RemoveIsoNode` — Remove um nó isolado e retorna descrição de ação. Se o nó não for isolado (for começo ou fim de um limite), então, uma exceção é lançada.

**Synopsis**

text `ST_RemoveIsoNode`(varchar atopology, integer anode);

**Descrição**

Remove um nó isolado e retorna descrição de ação. Se o nó não for isolado (for começo ou fim de um limite), então, uma exceção é lançada.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3

**Exemplos**

```
-- Remove an isolated node with no face --
SELECT topology.ST_RemoveIsoNode('ma_topo', 7 ) As result;
           result
-----
Isolated node 7 removed
```

**Veja também**

[ST\\_AddIsoNode](#)

**11.5.14 ST\_RemoveIsoEdge**

`ST_RemoveIsoEdge` — Removes an isolated edge and returns description of action. If the edge is not isolated, then an exception is thrown.

**Synopsis**

text `ST_RemoveIsoEdge`(varchar atopology, integer anedge);

**Descrição**

Removes an isolated edge and returns description of action. If the edge is not isolated, then an exception is thrown.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3

## Exemplos

```
-- Remove an isolated node with no face --
SELECT topology.ST_RemoveIsoNode('ma_topo', 7) As result;
      result
-----
Isolated node 7 removed
```

## Veja também

[ST\\_AddIsoNode](#)

## 11.6 Assessores de Topologia

### 11.6.1 GetEdgeByPoint

GetEdgeByPoint — Encontra a edge-id de um limite que intercepta um dado ponto

#### Synopsis

integer **GetEdgeByPoint**(varchar atopology, geometry apoint, float8 tol);

#### Restaura a id de um limite que intersecta um ponto.

A função retorna uma inteireza (id-limite) dada uma topologia, um PONTO e uma tolerância. Se tolerância = 0, o ponto tem que intersectar o limite.

Se o ponto não intersecta um limite, retorna 0 (zero).

Se usa tolerância > 0 e não existe mais que um limite próximo ao ponto, uma exceção é lançada.



#### Note

Se tolerância = 0, a função usa ST\_Intersects, senão usa ST\_DWithin.

Disponibilidade: 2.0.0 - requer GEOS >= 3.3.0.

## Exemplos

Estes exemplos utilizam limites que criamos em

```
SELECT topology.GetEdgeByPoint('ma_topo',geom, 1) As withlmtol, topology.GetEdgeByPoint(' ←
      ma_topo',geom,0) As withnotol
FROM ST_GeomFromEWKT('SRID=26986;POINT(227622.6 893843)') As geom;
      withlmtol | withnotol
-----+-----
          2 |          0
```

```
SELECT topology.GetEdgeByPoint('ma_topo',geom, 1) As nearnode
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
```

```
-- get error --
ERROR:  Two or more edges found
```

**Veja também**

[AddEdge](#), [GetNodeByPoint](#), [GetFaceByPoint](#)

**11.6.2 GetFaceByPoint**

`GetFaceByPoint` — Encontra a face-id de uma face que intersecta um dado ponto

**Synopsis**

integer **GetFaceByPoint**(varchar atopology, geometry apoint, float8 tol);

**Descrição**

Restaura a id de uma face que intersecta um ponto.

A função retorna como inteireza (id-face) dado uma topologia, um PONTO e uma tolerância. Se tolerância = 0, o ponto tem que intersectar a face.

Se o ponto não intersecta uma face, retorna 0 (zero).

Se usa tolerância > 0 e não existe mais que uma face próxima ao ponto, uma exceção é lançada.

**Note**

Se tolerância = 0, a função usa `ST_Intersects`, senão usa `ST_DWithin`.

Disponibilidade: 2.0.0 - requer GEOS >= 3.3.0.

**Exemplos**

Estes exemplos utilizam faces criadas em [AddFace](#)

```
SELECT topology.GetFaceByPoint('ma_topo',geom, 10) As with1mtol, topology.GetFaceByPoint(' ←
  ma_topo',geom,0) As withnotol
  FROM ST_GeomFromEWKT('POINT(234604.6 899382.0)') As geom;
```

```
  with1mtol | withnotol
-----+-----
              1 |          0
```

```
SELECT topology.GetFaceByPoint('ma_topo',geom, 1) As nearnode
  FROM ST_GeomFromEWKT('POINT(227591.9 893900.4)') As geom;
```

```
-- get error --
ERROR:  Two or more faces found
```

**Veja também**

[AddFace](#), [GetNodeByPoint](#), [GetEdgeByPoint](#)

**11.6.3 GetNodeByPoint**

`GetNodeByPoint` — Encontra a id de um nó em uma localização de ponto

**Synopsis**

integer **GetNodeByPoint**(varchar atology, geometry point, float8 tol);

**Restaura a id de um nó em uma localização de ponto**

A função retorna um inteiro (id-nó) dada uma topologia, um PONTO e uma tolerância. Se tolerância = 0, significa exatamente intersecção, caso contrário restaura o nó de um intervalo.

Se existe um nó no ponto, ele retorna 0 (zero).

Se usa tolerância > 0 e próximo ao ponto existe mais que um nó, uma exceção é lançada.

**Note**

Se tolerância = 0, a função usa ST\_Intersects, senão irá usar ST\_DWithin.

Disponibilidade: 2.0.0 - requer GEOS >= 3.3.0.

**Exemplos**

Estes exemplos utilizam limites que criamos em

```
SELECT topology.GetNodeByPoint('ma_topo',geom, 1) As nearnode
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;
nearnode
-----
2
```

```
SELECT topology.GetNodeByPoint('ma_topo',geom, 1000) As too_much_tolerance
FROM ST_GeomFromEWKT('SRID=26986;POINT(227591.9 893900.4)') As geom;

----get error--
ERROR: Two or more nodes found
```

**Veja também**

[AddEdge](#), [GetEdgeByPoint](#), [GetFaceByPoint](#)

**11.6.4 GetTopologyID**

GetTopologyID — Retorna a id de uma topologia na table topology.topology dado o nome da topologia.

**Synopsis**

integer **GetTopologyID**(varchar toponame);

**Descrição**

Retorna a id de uma topologia na table topology.topology dado o nome da topologia.

Disponibilidade: 1.?

## Exemplos

```
SELECT topology.GetTopologyID('ma_topo') As topo_id;

topo_id
-----
1
```

## Veja também

[Cria topologia](#), [DropTopology](#), [GetTopologyName](#), [GetTopologySRID](#)

### 11.6.5 GetTopologySRID

`GetTopologySRID` — Retorna o SRID de uma topologia na table `topology.topology` dado o nome da topologia.

#### Synopsis

integer **GetTopologyID**(varchar toponame);

#### Descrição

Retorna a id de referência espacial de uma topologia na table `topology.topology` dado o nome da topologia.

Disponibilidade: 2.0.0

## Exemplos

```
SELECT topology.GetTopologySRID('ma_topo') As SRID;

SRID
-----
4326
```

## Veja também

[Cria topologia](#), [DropTopology](#), [GetTopologyName](#), [GetTopologyID](#)

### 11.6.6 GetTopologyName

`GetTopologyName` — Retorna o nome de uma topologia (esquema) dada a id da topologia.

#### Synopsis

varchar **GetTopologyName**(integer topology\_id);

---



**Descrição**

Retorna o nome da topologia (esquema) de uma table topology.topology dada a id topologia dela.

Disponibilidade: 1.?

**Exemplos**

```
SELECT topology.GetTopologyName(1) As topo_name;

topo_name
-----
ma_topo
```

**Veja também**

[Cria topologia](#), [DropTopology](#), [GetTopologyID](#), [GetTopologySRID](#)

**11.6.7 ST\_GetFaceEdges**

ST\_GetFaceEdges — Retorna um conjunto de limites ordenados que amarram aface.

**Synopsis**

```
getfaceedges_returntype ST_GetFaceEdges(varchar atopology, integer aface);
```

**Descrição**

Retorna um conjunto de limites ordenados que amarram aface. Cada saída consiste em uma sequência e uma limiteid. Os números das sequências começam com o valor 1.

A enumeração dos limites de cada anel começa do limite com o menos identificador. A ordem de limites segue uma regra da mão esquerda (a face amarrada está a esquerda de cada limite direito).

Disponibilidade: 2.0



This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.5

**Exemplos**

```
-- Returns the edges bounding face 1
SELECT (topology.ST_GetFaceEdges('tt', 1)).*;
-- result --
sequence | edge
-----+-----
         1 |   -4
         2 |    5
         3 |    7
         4 |   -6
         5 |    1
         6 |    2
         7 |    3
(7 rows)
```

```
-- Returns the sequence, edge id
-- and geometry of the edges that bound face 1
-- If you just need geom and seq, can use ST_GetFaceGeometry
SELECT t.seq, t.edge, geom
FROM topology.ST_GetFaceEdges('tt',1) As t(seq,edge)
      INNER JOIN tt.edge AS e ON abs(t.edge) = e.edge_id;
```

## Veja também

[GetRingEdges](#), [AddFace](#), [ST\\_GetFaceGeometry](#)

### 11.6.8 ST\_GetFaceGeometry

`ST_GetFaceGeometry` — Retorna o polígono na topologia dada com a id de face especificada.

#### Synopsis

geometry `ST_GetFaceGeometry`(varchar atopology, integer aface);

#### Descrição

Retorna o polígono na topologia dada com a id de face especificada. Constrói o polígono dos limites fazendo a face.

Disponibilidade: 1.?



This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.16

#### Exemplos

```
-- Returns the wkt of the polygon added with AddFace
SELECT ST_AsText(topology.ST_GetFaceGeometry('ma_topo', 1)) As facegeomwkt;
-- result --
      facegeomwkt
-----
POLYGON((234776.9 899563.7,234896.5 899456.7,234914 899436.4,234946.6 899356.9,
234872.5 899328.7,234891 899285.4,234992.5 899145,234890.6 899069,
234755.2 899255.4,234612.7 899379.4,234776.9 899563.7))
```

## Veja também

[AddFace](#)

### 11.6.9 GetRingEdges

`GetRingEdges` — Retorna o conjunto ordenado de identificadores de limites assinados, conhecidos caminhando em um lado da beirada.

#### Synopsis

getfaceedges\_returntype `GetRingEdges`(varchar atopology, integer aring, integer max\_edges=null);

## Descrição

Retorna o conjunto ordenado de identificadores de limites assinados, conhecidos caminhando em um lado da beirada. Cada saída consiste em uma sequência e uma id limite assinada. Números em sequência começam com o valor 1.

Se você passa uma id limite positiva, a caminhada começa no lado esquerdo do limite correspondente e segue sua direção. Se você passa uma id limite negativa, a caminhada começa no lado direito dele e orienta-se para trás.

Se `max_edges` não é nulo, não mais que aqueles relatos são retornados pela função. Isto foi feito para ser um parâmetro seguro ao lidar com topologias possivelmente inválidas.



### Note

Esta função utiliza anel limite vinculando metadados.

---

Disponibilidade: 2.0.0

## Veja também

[ST\\_GetFaceEdges](#), [GetNodeEdges](#)

### 11.6.10 GetNodeEdges

`GetNodeEdges` — Retorna um conjunto ordenado de limites incidentes no dado nó.

## Synopsis

```
getfaceedges_returntype GetNodeEdges(varchar atopology, integer anode);
```

## Descrição

Retorna um conjunto de limites incidentes no dado nó. Cada saída consiste em uma sequência e uma id limite assinada. Os números sequência começam com o valor 1. Um limite positivo começa no dado nó. Um limite negativo termina no dado nó. Limites fechado aparecerão duas vezes (com ambos sinais). A ordem é sentido horário, começando do norte.



### Note

Esta função computa ordenação em vez de derivação dos metadados e é, assim, útil para construir o vínculo do limite anel.

---

Disponibilidade: 2.0

## Veja também

[GetRingEdges](#), [ST\\_Azimuth](#)

## 11.7 Processamento de Topologia

### 11.7.1 Polygonize

`Polygonize` — Encontra e registra todas faces definidas pelos limites da topologia

---

## Synopsis

```
text Polygonize(varchar toponame);
```

## Descrição

Registra todas as faces que podem ser construídas de bordas primitivas das topologias.

A topologia alvo supostamente contém nenhuma borda que se auto intersecta.



### Note

Faces já conhecidas são reconhecidas, logo, é seguro chamar Polygonize várias vezes na mesma topologia.



### Note

Esta função não utiliza os campos set the next\_left\_edge e next\_right\_edge da table limite.

Disponibilidade: 2.0.0

## Veja também

[AddFace](#), [ST\\_Polygonize](#)

## 11.7.2 AddNode

AddNode — Adiciona um ponto nó na table nó no esquema topológico específico e retorna a nodeid do novo nó. Se o ponto já existe, a nodeid é retornada.

## Synopsis

```
integer AddNode(varchar toponame, geometry apoint, boolean allowEdgeSplitting=false, boolean computeContainingFace=false);
```

## Descrição

Adiciona um ponto nó na table nó no esquema topológico específico. A função [AddEdge](#) automaticamente adiciona pontos de início e fim de um limite quando chamado, não é necessário adicionar nós de um limite explicitamente.

Se qualquer limite cruzando o nó é encontrado, ou uma exceção surge ou a borda é dividida, dependendo do valor do parâmetro `allowEdgeSplitting`.

Se `computeContainingFace` for verdade, um novo nó adicionado irá corrigir a face computada.



### Note

Se a geometria `apoint` já existe como um nó, não se adiciona um nó, mas a nodeid existente retorna.

Disponibilidade: 2.0.0

## Exemplos

```
SELECT topology.AddNode('ma_topo', ST_GeomFromText('POINT(227641.6 893816.5)', 26986) ) As ←
    nodeid;

-- result --

nodeid
-----
4
```

## Veja também

[AddEdge](#), [Cria topologia](#)

### 11.7.3 AddEdge

**AddEdge** — Adiciona uma linestring limite à edge table e os pontos de início e fim associados à table ponto nó do esquema de topologia especificado usando a linestring geometria específica e retorna a bordaid da nova borda (ou da borda já existente).

#### Synopsis

integer **AddEdge**(varchar toponame, geometry aline);

#### Descrição

Adiciona uma borda à edge table e nós associados às nodes tables do esquema toponame especificado, usando a linestring geometria específica e retorna a bordaid do novo ou já existente relato. A nova borda adicionada tem a face "universal" nos dois lados e se conecta com si mesma.



#### Note

Se a `aline` geometria cruza, sobrepõe, contém ou é contida por uma borda linestring, um erro é lançado e a borda não é adicionada.



#### Note

A geometria da `aline` deve ter o mesmo `srid` definido para a topologia, senão um erro inválido é lançado no sistema de referência espacial.

Disponibilidade: 2.0.0. requer GEOS >= 3.3.0.

## Exemplos

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575.8 893917.2,227591.9 ←
    893900.4)', 26986) ) As edgeid;
-- result-
edgeid
-----
```

```

1
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.9 893900.4,227622.6 ↵
      893844.2,227641.6 893816.5,
      227704.5 893778.5)', 26986) ) As edgeid;
-- result --
edgeid
-----
2
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.2 893900, 227591.9 ↵
      893900.4,
      227704.5 893778.5)', 26986) ) As edgeid;
-- gives error --
ERROR:  Edge intersects (not on endpoints) with existing edge 1

```

### Veja também

[TopoGeo\\_AddLineString](#), [Cria topologia](#), [Section 4.3.1](#)

## 11.7.4 AddFace

AddFace — Registra uma face primitiva a uma topologia e pega seu identificador.

### Synopsis

integer **AddFace**(varchar toponame, geometry apolygon, boolean force\_new=false);

### Descrição

Registra uma face primitiva a uma topologia e pega seu identificador.

Para uma nova face adicionada, as bordas formando seus limites e as contidas na face, serão atualizadas para ter valores corretos nos campos `left_face` e `right_face`. Os nós isolados contidos na face também serão atualizados para ter um valor correto do campo `containing_face`.



#### Note

Esta função não utiliza os campos `set the next_left_edge` e `next_right_edge` da table limite.

A topologia alvo é supostamente válida (não contendo nenhuma borda auto intersectada). Uma exceção surge se: O limite do polígono não estiver completamente definido ou caso o polígono sobreponha uma face existente.

Se a `apolygon` geometria já existe como face, então: se `force_new` é falso (o padrão) a id da face da face existente retorna, se `force_new` é verdade uma nova id será assinada para a nova face registrada.



#### Note

Quando um nó de registro de uma face existente é representada (`force_new=true`), nenhuma ação será tomada para resolver referências pendentes a face existente na borda, nó e tables relacionadas, nem o relato do campo MBR será atualizado. Fica a critério do chamador lidar ou não com isso.

**Note**

A geometria da `apolygon` deve ter o mesmo `srid` definido para a topologia, senão um erro inválido é lançado no sistema de referência espacial.

Disponibilidade: 2.0.0

**Exemplos**

```
-- first add the edges we use generate_series as an iterator (the below
-- will only work for polygons with < 10000 points because of our max in gs)
SELECT topology.AddEdge('ma_topo', ST_MakeLine(ST_PointN(geom,i), ST_PointN(geom, i + 1) )) ←
  As edgeid
  FROM (SELECT ST_NPoints(geom) AS npt, geom
        FROM
          (SELECT ST_Boundary(ST_GeomFromText ('POLYGON((234896.5 899456.7,234914 ←
            899436.4,234946.6 899356.9,234872.5 899328.7,
            234891 899285.4,234992.5 899145, 234890.6 899069,234755.2 899255.4,
            234612.7 899379.4,234776.9 899563.7,234896.5 899456.7))', 26986) ) As geom
          ) As geoms) As facen CROSS JOIN generate_series(1,10000) As i
        WHERE i < npt;
-- result --
  edgeid
-----
      3
      4
      5
      6
      7
      8
      9
     10
     11
     12
(10 rows)
-- then add the face -

SELECT topology.AddFace('ma_topo',
  ST_GeomFromText ('POLYGON((234896.5 899456.7,234914 899436.4,234946.6 899356.9,234872.5 ←
    899328.7,
    234891 899285.4,234992.5 899145, 234890.6 899069,234755.2 899255.4,
    234612.7 899379.4,234776.9 899563.7,234896.5 899456.7))', 26986) ) As faceid;
-- result --
  faceid
-----
      1
```

**Veja também**

[AddEdge](#), [Cria topologia](#), [Section 4.3.1](#)

**11.7.5 ST\_Simplify**

`ST_Simplify` — Retorna uma versão "simplificada" da geometria da dada `TopoGeometria` usando o algoritmo Douglas-Peucker.

**Synopsis**

geometry **ST\_Simplify**(TopoGeometry geomA, float tolerance);

**Descrição**

Retorna uma versão "simplificada" da geometria da dada TopoGeometria usando o algoritmo Douglas-Peucker em cada borda componente.

**Note**

A geometria retornada pode ser não simples ou não válida.  
Dividir bordas componentes pode ajudar a manter simplicidade/validade.

Desempenhado pelo módulo GEOS.

Disponibilidade: 2.1.0

**Veja também**

Geometry [ST\\_Simplify](#), [ST\\_IsSimple](#), [ST\\_IsValid](#), [ST\\_ModEdgeSplit](#)

## 11.8 Construtores de TopoGeometria

### 11.8.1 CreateTopoGeom

CreateTopoGeom — Cria uma novo objeto de topo geometria de um arranjo topo elemento - tg\_type: 1:[multi]point, 2:[multi]line, 3:[multi]poly, 4:collection

**Synopsis**

topogeometry **CreateTopoGeom**(varchar toponame, integer tg\_type, integer layer\_id, topoelementarray tg\_objs);  
topogeometry **CreateTopoGeom**(varchar toponame, integer tg\_type, integer layer\_id);

**Descrição**

Cria um objeto topogeometria para camada indicada pela layer\_id e registra ela nas relations tables no esquema toponame.

tg\_type is an integer: 1:[multi]point (punctal), 2:[multi]line (lineal), 3:[multi]poly (areal), 4:collection. layer\_id is the layer id in the topology.layer table.

camadas pontuais são formadas a partir de um conjunto de nós, camadas lineares são formadas a partir de um conjunto de bordas, camadas areais são formadas a partir de um conjunto de faces e as coleções podem ser formadas a partir de uma mistura de nós, bordas e faces.

Omitir o arranjo de componentes gera um objeto TopoGeometria vazio.

Disponibilidade: 1.?

**Exemplos: Formados de bordas existentes**

Create a topogeom in ri\_topo schema for layer 2 (our ri\_roads), of type (2) LINE, for the first edge (we loaded in ST\_CreateTopoGeo)

```
INSERT INTO ri.ri_roads(road_name, topo) VALUES('Unknown', topology.CreateTopoGeom('ri_topo ←
',2,2, '{{1,2}}'::topology.topoelementarray);
```



### Exemplos: Converte uma geometria areal para uma topogeometria melhor

Digamos que tenhamos geometrias que deveriam ser formadas de uma coleção de faces. Nós temos, por exemplo, blockgroups tables e queremos saber a topo geometria de cada block group. Se seus dados foram perfeitamente alinhados, podemos fazer isto:

```
-- create our topo geometry column --
SELECT topology.AddTopoGeometryColumn(
    'topo_boston',
    'boston', 'blockgroups', 'topo', 'POLYGON');

-- addtopogeometrycolumn --
1

-- update our column assuming
-- everything is perfectly aligned with our edges
UPDATE boston.blockgroups AS bg
    SET topo = topology.CreateTopoGeom('topo_boston'
    ,3,1
    , foo.bfaces)
FROM (SELECT b.gid, topology.TopoElementArray_Agg(ARRAY[f.face_id,3]) As bfaces
    FROM boston.blockgroups As b
    INNER JOIN topo_boston.face As f ON b.geom && f.mbr
    WHERE ST_Covers(b.geom, topology.ST_GetFaceGeometry('topo_boston', f.face_id))
    GROUP BY b.gid) As foo
WHERE foo.gid = bg.gid;

--the world is rarely perfect allow for some error
--count the face if 50% of it falls
-- within what we think is our blockgroup boundary
UPDATE boston.blockgroups AS bg
    SET topo = topology.CreateTopoGeom('topo_boston'
    ,3,1
    , foo.bfaces)
FROM (SELECT b.gid, topology.TopoElementArray_Agg(ARRAY[f.face_id,3]) As bfaces
    FROM boston.blockgroups As b
    INNER JOIN topo_boston.face As f ON b.geom && f.mbr
    WHERE ST_Covers(b.geom, topology.ST_GetFaceGeometry('topo_boston', f.face_id))
    OR
    ( ST_Intersects(b.geom, topology.ST_GetFaceGeometry('topo_boston', f.face_id))
    AND ST_Area(ST_Intersection(b.geom, topology.ST_GetFaceGeometry('topo_boston', ←
    f.face_id) ) ) >
    ST_Area(topology.ST_GetFaceGeometry('topo_boston', f.face_id))*0.5
    )
    GROUP BY b.gid) As foo
WHERE foo.gid = bg.gid;

-- and if we wanted to convert our topogeometry back
-- to a denormalized geometry aligned with our faces and edges
-- cast the topo to a geometry
-- The really cool thing is my new geometries
-- are now aligned with my tiger street centerlines
UPDATE boston.blockgroups SET new_geom = topo::geometry;
```

### Veja também

[AddTopoGeometryColumn](#), [toTopoGeom](#) [ST\\_CreateTopoGeo](#), [ST\\_GetFaceGeometry](#), [TopoElementArray](#), [TopoElementArray\\_Agg](#)

## 11.8.2 toTopoGeom

toTopoGeom — Converte uma simples geometria em uma topo geometria

## Synopsis

```
topogeometry toTopoGeom(geometry geom, varchar toponame, integer layer_id, float8 tolerance);
topogeometry toTopoGeom(geometry geom, topogeometry topogeom, float8 tolerance);
```

## Descrição

Converte uma simples geometria em **TopoGeometry**.

Topológicos primitivos requeridos para representar a geometria de entrada será adicionada a topologia oculta, possivelmente dividindo as existentes, e elas serão associadas com a TopoGeometria de saída na table relation.

Objetos existentes de TopoGeometria (com a possível exceção de topogeom, se dada) manterão suas formas.

Quando `tolerance` é dada, será usada para quebrar a geometria de entrada para primitivas existentes.

Na primeira forma, uma nova TopoGeometria será criada para a dada camada (`layer_id`) da topologia (`toponame`)

Na segunda forma, as primitivos resultantes da conversão serão adicionadas a uma TopoGeometria pre existente (`topogeom`), adicionando, possivelmente, espaço à sua forma final. Para obter a nova forma completamente substituir a antiga, veja **clearTopoGeom**.

Disponibilidade: 2.0

Melhorias: 2.1.0 adiciona a versão pegando uma TopoGeometria existente.

## Exemplos

Este é um fluxo de trabalho auto contido completo

```
-- do this if you don't have a topology setup already
-- creates topology not allowing any tolerance
SELECT topology.CreateTopology('topo_boston_test', 2249);
-- create a new table
CREATE TABLE nei_topo(gid serial primary key, nei varchar(30));
--add a topogeometry column to it
SELECT topology.AddTopoGeometryColumn('topo_boston_test', 'public', 'nei_topo', 'topo', ' ←
MULTIPOLYGON') As new_layer_id;
new_layer_id
-----
1

--use new layer id in populating the new topogeometry column
-- we add the topogeoms to the new layer with 0 tolerance
INSERT INTO nei_topo(nei, topo)
SELECT nei, topology.toTopoGeom(geom, 'topo_boston_test', 1)
FROM neighborhoods
WHERE gid BETWEEN 1 and 15;

--use to verify what has happened --
SELECT * FROM
    topology.TopologySummary('topo_boston_test');

-- summary--
Topology topo_boston_test (5), SRID 2249, precision 0
61 nodes, 87 edges, 35 faces, 15 topogeoms in 1 layers
Layer 1, type Polygonal (3), 15 topogeoms
Deploy: public.nei_topo.topo
```

```
-- Shrink all TopoGeometry polygons by 10 meters
UPDATE nei_topo SET topo = ST_Buffer(clearTopoGeom(topo), -10);
```

```

-- Get the no-one-lands left by the above operation
-- I think GRASS calls this "polygon0 layer"
SELECT ST_GetFaceGeometry('topo_boston_test', f.face_id)
FROM topo_boston_test.face f
WHERE f.face_id
> 0 -- don't consider the universe face
AND NOT EXISTS ( -- check that no TopoGeometry references the face
  SELECT * FROM topo_boston_test.relation
  WHERE layer_id = 1 AND element_id = f.face_id
);

```

### Veja também

[Cria topologia](#), [AddTopoGeometryColumn](#), [CreateTopoGeom](#), [TopologySummary](#), [clearTopoGeom](#)

### 11.8.3 TopoElementArray\_Agg

`TopoElementArray_Agg` — Retorna um `topoelementarray` para um conjunto de arranjos `element_id`, `type` (topoelementos)

#### Synopsis

`topoelementarray` **TopoElementArray\_Agg**(topoelement set tefield);

#### Descrição

Usado para criar um **TopoElementArray** de um conjunto de **TopoElement**.

Disponibilidade: 2.0.0

#### Exemplos

```

SELECT topology.TopoElementArray_Agg(ARRAY[e,t]) As tea
FROM generate_series(1,3) As e CROSS JOIN generate_series(1,4) As t;
tea
-----
{{1,1},{1,2},{1,3},{1,4},{2,1},{2,2},{2,3},{2,4},{3,1},{3,2},{3,3},{3,4}}

```

### Veja também

[TopoElement](#), [TopoElementArray](#)

## 11.9 Editores de TopoGeometria

### 11.9.1 clearTopoGeom

`clearTopoGeom` — Limpa o conteúdo de uma topo geometria

## Synopsis

topogeometry **clearTopoGeom**(topogeometry topogeom);

## Descrição

Limpa o conteúdo de um **TopoGeometry** tornando-o vazio. Mais útil quando usado em conjunto com **toTopoGeom** para substituir o formato de objetos existentes e qualquer objeto dependente em níveis hierárquicos mais elevados.

Disponibilidade: 2.1

## Exemplos

```
-- Shrink all TopoGeometry polygons by 10 meters
UPDATE nei_topo SET topo = ST_Buffer(clearTopoGeom(topo), -10);
```

## Veja também

[toTopoGeom](#)

### 11.9.2 TopoGeom\_addElement

**TopoGeom\_addElement** — Adiciona um elemento à definição de uma TopoGeometria

## Synopsis

topogeometry **TopoGeom\_addElement**(topogeometry tg, topoelement el);

## Descrição

Adiciona um **TopoElement** à definição de um objeto de TopoGeometria. Não apresenta erro se o elemento já faz parte da definição.

Disponibilidade: 2.3

## Exemplos

```
-- Add edge 5 to TopoGeometry tg
UPDATE mylayer SET tg = TopoGeom_addElement(tg, '{5,2}');
```

## Veja também

[TopoGeom\\_remElement](#), [CreateTopoGeom](#)

### 11.9.3 TopoGeom\_remElement

**TopoGeom\_remElement** — Remove um elemento de uma definição de uma TopoGeometria

## Synopsis

topogeometry **TopoGeom\_remElement**(topogeometry tg, topoelement el);

---

## Descrição

Remove um [TopoElement](#) de uma definição de uma objeto de TopoGeometrias.

Disponibilidade: 2.3

## Exemplos

```
-- Remove face 43 from TopoGeometry tg
UPDATE mylayer SET tg = TopoGeom_remElement(tg, '{43,3}');
```

## Veja também

[TopoGeom\\_addElement](#), [CreateTopoGeom](#)

## 11.9.4 toTopoGeom

toTopoGeom — Adiciona um formato geométrico à uma topo geometria existente

## Descrição

Recorrer a [toTopoGeom](#)

## 11.10 Assessores de TopoGeometria

### 11.10.1 GetTopoGeomElementArray

GetTopoGeomElementArray — Retorna um `topoelementarray` (um arranjo de topoelementos) contendo os tipos e elementos topológicos da TopoGeometria (elementos primitivos)

## Synopsis

`topoelementarray` **GetTopoGeomElementArray**(varchar toponame, integer layer\_id, integer tg\_id);

`topoelementarray topoelement` **GetTopoGeomElementArray**(topogeometry tg);

## Descrição

Retorna um [TopoElementArray](#) contendo os tipos e elementos da dada TopoGeometria (elementos primitivos). Isto é parecido com [GetTopoGeomElements](#), mas em vez de retornar os elementos como dataset, retorna como um arranjo.

`tg_id` é a id topogeometria do objeto de topogeometria na camada indicada pela `layer_id` na `topology.layer` table.

Disponibilidade: 1.?

## Exemplos

## Veja também

[GetTopoGeomElements](#), [TopoElementArray](#)

---

## 11.10.2 GetTopoGeomElements

GetTopoGeomElements — Retorna um conjunto de objetos `topoelement` contendo os topological `element_id`, `element_type` da TopoGeometria (elementos primitivos)

### Synopsis

```
setof topoelement GetTopoGeomElements(varchar toponame, integer layer_id, integer tg_id);
setof topoelement GetTopoGeomElements(topogeometry tg);
```

### Descrição

Retorna um conjunto de `element_id`, `element_type` (topoelementos) para um dado objeto topogeométrico no esquema `toponame`. `tg_id` é a id topogeometria do objeto de topogeometria na camada indicada pela `layer_id` na `topology.layer` table.

Disponibilidade: 2.0.0

### Exemplos

#### Veja também

[GetTopoGeomElementArray](#), [TopoElement](#), [TopoGeom\\_addElement](#), [TopoGeom\\_remElement](#)

## 11.11 TopoGeometry Outputs

### 11.11.1 AsGML

AsGML — Retorna a representação GML de uma topogeometria.

### 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);
```

### Descrição

Retorna a representação GML de uma topogeometria na versão GML3 format. Se o `nsprefix_in` não for especificado, então `gml` é usado. Passa em uma string vazia para `nsprefix` para pegar um espaço não qualificado. A precisão (padrão: 15) e parâmetros (padrão 1) de opções, se dados, são passados inalterados para a chamada subjacente para `ST_AsGML`.

O parâmetro `visitedTable`, se dado, é usado para manter o caminho dos elementos nó e borda visitados, assim como para usar referências-cruzadas (`xlink:xref`) em vez de definições duplicadas. É esperado que a tabela tenha (pelo menos) dois campos inteiros: `'element_type'` e `'element_id'`. O usuário visitante deve ter os privilégios escritos e lidos na dada tabela. Para uma melhor apresentação, um `index` deve ser definido no `element_type` e `element_id`, nesta ordem. Tal `index` será adicionado automaticamente ao adicionar uma única limitação aos campos. Exemplo:

```
CREATE TABLE visited (
  element_type integer, element_id integer,
  unique(element_type, element_id)
);
```

O parâmetro `idprefix`, se dado, será antecipado aos identificadores tag de bordas e nós.

O parâmetro `gmlver`, se dado, será passado à `ST_AsGML` subjacente. Padrões para 3.

Disponibilidade: 2.0.0

## Exemplos

Isto usa a topo geometria que criamos em [CreateTopoGeom](#)

```
SELECT topology.AsGML(topo) As rdgml
FROM ri.roads
WHERE road_name = 'Unknown';

-- rdgml--
<gml:TopoCurve>
  <gml:directedEdge>
    <gml:Edge gml:id="E1">
      <gml:directedNode orientation="-">
        <gml:Node gml:id="N1"/>
      </gml:directedNode>
      <gml:directedNode>
    </gml:directedNode>
    <gml:curveProperty>
      <gml:Curve srsName="urn:ogc:def:crs:EPSG::3438">
        <gml:segments>
          <gml:LineStringSegment>
            <gml:posList srsDimension="2"
>384744 236928 384750 236923 384769 236911 384799 236895 384811 236890
          384833 236884 384844 236882 384866 236881 384879 236883 384954 ←
            236898 385087 236932 385117 236938
          385167 236938 385203 236941 385224 236946 385233 236950 385241 ←
            236956 385254 236971
          385260 236979 385268 236999 385273 237018 385273 237037 385271 ←
            237047 385267 237057 385225 237125
          385210 237144 385192 237161 385167 237192 385162 237202 385159 ←
            237214 385159 237227 385162 237241
          385166 237256 385196 237324 385209 237345 385234 237375 385237 ←
            237383 385238 237399 385236 237407
          385227 237419 385213 237430 385193 237439 385174 237451 385170 ←
            237455 385169 237460 385171 237475
          385181 237503 385190 237521 385200 237533 385206 237538 385213 ←
            237541 385221 237542 385235 237540 385242 237541
          385249 237544 385260 237555 385270 237570 385289 237584 385292 ←
            237589 385291 237596 385284 237630</gml:posList>
          </gml:LineStringSegment>
        </gml:segments>
      </gml:Curve>
    </gml:curveProperty>
  </gml:Edge>
</gml:directedEdge>
</gml:TopoCurve>
>
```

É o mesmo exercício do o anterior, mas sem o espaço para nome

```

SELECT topology.AsGML(topo, '') As rdgml
FROM ri.roads
WHERE road_name = 'Unknown';

-- rdgml--
<TopoCurve>
  <directedEdge>
    <Edge id="E1">
      <directedNode orientation="-">
        <Node id="N1"/>
      </directedNode>
      <directedNode
></directedNode>
      <curveProperty>
        <Curve srsName="urn:ogc:def:crs:EPSG::3438">
          <segments>
            <LineStringSegment>
              <posList srsDimension="2"
>384744 236928 384750 236923 384769 236911 384799 236895 384811 236890
              384833 236884 384844 236882 384866 236881 384879 236883 384954 ←
              236898 385087 236932 385117 236938
              385167 236938 385203 236941 385224 236946 385233 236950 385241 ←
              236956 385254 236971
              385260 236979 385268 236999 385273 237018 385273 237037 385271 ←
              237047 385267 237057 385225 237125
              385210 237144 385192 237161 385167 237192 385162 237202 385159 ←
              237214 385159 237227 385162 237241
              385166 237256 385196 237324 385209 237345 385234 237375 385237 ←
              237383 385238 237399 385236 237407
              385227 237419 385213 237430 385193 237439 385174 237451 385170 ←
              237455 385169 237460 385171 237475
              385181 237503 385190 237521 385200 237533 385206 237538 385213 ←
              237541 385221 237542 385235 237540 385242 237541
              385249 237544 385260 237555 385270 237570 385289 237584 385292 ←
              237589 385291 237596 385284 237630</posList>
            </LineStringSegment>
          </segments>
        </Curve>
      </curveProperty>
    </Edge>
  </directedEdge>
</TopoCurve
>

```

**Veja também**

[CreateTopoGeom](#), [ST\\_CreateTopoGeo](#)

**11.11.2 AsTopoJSON**

AsTopoJSON — Retorna a representação TopoJSON de uma topogeometria.

**Synopsis**

```
text AsTopoJSON(topogeometry tg, regclass edgeMapTable);
```



## Descrição

Retorna a representação TopoJSON de uma topogeometria. Se a `edgeMapTable` não for nula, será usada como um mapeamento de pesquisa/armazenamento de identificadores de borda para índices de arcos. Isto é para permitir um arranjo compacto de "arcos" no documento final.

É esperado que a tabela, se dada, tenha um campo "arc\_id" do tipo "serial" e uma "edge\_id" de tipo inteiro; o código irá consultar a tabela para "edge\_id", então é recomendado adicionar um index naquele campo.



### Note

Os índices de arcos na saída TopoJSON são 0-baseados mas são 1-baseados na tabela "edgeMapTable".

Um documento TopoJSON completo precisará conter, em soma com os fragmentos retornados por esta função, os arcos atuais mais alguns cabeçalhos. Veja a [TopoJSON specification](#).

Disponibilidade: 2.1.0

Melhorias: 2.2.1 suporte para entradas pontuais adicionado

## Veja também

[ST\\_AsGeoJSON](#)

## Exemplos

```
CREATE TEMP TABLE edgemap(arc_id serial, edge_id int unique);

-- header
SELECT '{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects ←
": {'

-- objects
UNION ALL SELECT '' || feature_name || ': ' || AsTopoJSON(feature, 'edgemap')
FROM features.big_parcelas WHERE feature_name = 'P3P4';

-- arcs
WITH edges AS (
  SELECT m.arc_id, e.geom FROM edgemap m, city_data.edge e
  WHERE e.edge_id = m.edge_id
), points AS (
  SELECT arc_id, (st_dumppoints(geom)).* FROM edges
), compare AS (
  SELECT p2.arc_id,
     CASE WHEN p1.path IS NULL THEN p2.geom
          ELSE ST_Translate(p2.geom, -ST_X(p1.geom), -ST_Y(p1.geom))
     END AS geom
  FROM points p2 LEFT OUTER JOIN points p1
  ON ( p1.arc_id = p2.arc_id AND p2.path[1] = p1.path[1]+1 )
  ORDER BY arc_id, p2.path
), arcsdump AS (
  SELECT arc_id, (regexp_matches( ST_AsGeoJSON(geom), '\[.*\]'))[1] as t
  FROM compare
), arcs AS (
  SELECT arc_id, '[' || array_to_string(array_agg(t), ',') || ']' as a FROM arcsdump
  GROUP BY arc_id
  ORDER BY arc_id
)
```

```

SELECT ', "arcs": [' UNION ALL
SELECT array_to_string(array_agg(a), E',\n') from arcs

-- footer
UNION ALL SELECT ']}':::text as t;

-- Result:
{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects": {
"P3P4": { "type": "MultiPolygon", "arcs": [[[-1]], [[6,5,-5,-4,-3,1]]]}
}, "arcs": [
[[25,30],[6,0],[0,10],[-14,0],[0,-10],[8,0]],
[[35,6],[0,8]],
[[35,6],[12,0]],
[[47,6],[0,8]],
[[47,14],[0,8]],
[[35,22],[12,0]],
[[35,14],[0,8]]
]]}

```

## 11.12 Relações de Topologia Espacial

### 11.12.1 Equivalentes

Equivalentes — Retorna verdade se duas topogeometrias forem compostas da mesma topologia primitiva

#### Synopsis

boolean **Equals**(topogeometry tg1, topogeometry tg2);

#### Descrição

Retorna verdade se duas topogeometrias forem compostas das mesmas topologias primitivas: faces, bordas e nós.



#### Note

Esta função não é suportada por geometrias que são coleções de geometrias. Também não pode comparar topogeometrias de topologias diferentes.

Disponibilidade: 1.1.0



This function supports 3d and will not drop the z-index.

#### Exemplos

#### Veja também

[GetTopoGeomElements](#), [ST\\_Equals](#)

## 11.12.2 Intersecta

Intersecta — Retorna verdade se algum par de primitivos das duas topologias se intersectar.

### Synopsis

```
boolean Intersects(topogeometry tg1, topogeometry tg2);
```

### Descrição

Retorna verdade se algum par de primitivos das duas topologias se intersectar.



#### Note

Esta função não é suportada por geometrias que são coleções de geometrias. Também não pode comparar topogeometrias de topologias diferentes. Além de não suportar topogeometrias hierárquicas (compostas de outras topogeometrias).

Disponibilidade: 1.1.0



This function supports 3d and will not drop the z-index.

### Exemplos

### Veja também

[ST\\_Intersects](#)

## Chapter 12

# Padronizador de endereço

Essa é uma forquilha do **padronizador PAGC** (código original para essa porção era **Padronizador de endereço PAGC PostgreSQL**).

O padronizador de endereços é uma única linha de análise sintática que pega um endereço de entrada e o normaliza baseado em um conjunto de regras armazenado em uma table e helper lex e gaz tables.

O código é construído em uma única biblioteca de extensão chamada `address_standardizer` a qual pode ser instalada com `CREATE EXTENSION address_standardizer;`. Juntamente com a extensão `address_standardizer`, uma extensão amostra de dados chamada `address_standardizer_data_us` é construída, a qual contém gaz, lex e regras tables para dados dos EUA. Essas extensões podem ser instaladas via: `CREATE EXTENSION address_standardizer_data_us;`

O código para esta extensão pode ser encontrado no PostGIS `extensions/address_standardizer` e está atualmente autocontido.

Para instruções de instalação consulte: [Section 2.7](#).

### 12.1 Como o analisador sintático funciona

O analisador sintático funciona da direita para a esquerda observando primeiramente os macro elementos para CEP, estado/província, cidade e depois observando os micro elementos para determinar se estamos lidando com uma casa numerada em uma rua ou intersecção ou ponto de referência. Ele normalmente não procura pelo código ou nome do país, mas isso poderia ser introduzido no futuro.

**Código do país** Suposto de ser EUA ou CA com base em: CEP como EUA ou estado/província do Canadá como EUA ou Canadá outro EUA

**Caixa postal/CEP** Esses são reconhecidos utilizando expressões Perl compatíveis. Esses regexs estão atualmente no `parseaddress-api.c` e são relativamente fáceis de alterar, caso seja necessário.

**Estado/província** Esses são reconhecidos utilizando expressões Perl compatíveis. Esses regexs estão atualmente no `parseaddress-api.c` e são relativamente fáceis de alterar, caso seja necessário.

### 12.2 Tipos de padronizador de endereço

#### 12.2.1 stdaddr

`stdaddr` — Um tipo composto que consiste nos elementos de um endereço. Este é o tipo de retorno para `standardize_address` função.

## Descrição

Um tipo composto dos elementos de um endereço. Este é o tipo de retorno para `standardize_address` função. Algumas descrições para elementos são emprestadas de [PAGC Postal Attributes](#).

Os números pegos denotam o número de referência da saída no [mesa de regras](#).



This method needs `address_standardizer` extension.

**construindo** é texto (token number 0): Refere ao número da construção ou nome. Identificadores e tipos de construções unparsed. Normalmente em branco para a maioria dos endereços.

**house\_num** é um texto (número token 1): Este é o número da rua em uma rua. Exemplo 75 em 75 Rua State.

**predir** é um texto (número token 2): NOME DA RUA PRE-DIRECTIONAL como Norte, Sul, Leste, Oeste etc.

**qual** é um texto (número token 3): NOME DA RUA PRE-MODIFIER Exemplo *VELHA* em 3715 ESTRADA VELHA 99.

**pré tipo** é um texto (número token 4): TIPO DE PREFIXO DA RUA

**nome** é um texto (número token 5): NOME DA RUA

**suftype** é um texto (número token 6): TIPO DE CORREIO DA RUA ex. R, Av, Cir. Um tipo de rua seguindo o nome raiz da rua. Exemplo *RUA* em 75 Rua State.

**sufdir** é um texto (número token 7): RUA POST-DIRECTIONAL Um modificador direcional que segue o nome da rua.. Exemplo *OESTE* em 3715 DÉDIMA AVENIDA OESTE.

**rota rural** is text (token number 8): RURAL ROUTE . Example 7 in RR 7.

**extra** é texto: informação extra como número de pisos.

**cidade** is text (token number 10): Exemplo Boston.

**estado** is text (token number 11): Exemplo MASSACHUSETTS

**país** is text (token number 12): Exemplo USA

**caixa postal** é texto CÓDIGO POSTAL (CÓDIGO ZIP) (número token 13): Exemplo 02109

**box** is text POSTAL BOX NUMBER (token number 14 and 15): Example 02109

**unidade** é texto Número do apartamento ou Número da suíte (número token 17): Exemplo *3B* em APTO 3B.

## 12.3 Mesas de padronizador de endereço

### 12.3.1 mesa de regras

mesa de regras — A mesa de regras contém um conjunto de regras que mapeia a sequência de tokens de entrada de endereço para a sequência de saída. Uma regra é definida como um conjunto de tokens de entrada seguido por -1 (terminator) seguido por conjunto de tokens de saída seguido por -1 seguido por um número que denota tipo de regra seguido por um ranking de regra.

## Descrição

Uma tabel regras deve ter pelo menos as colunas a seguir, embora você tenha permissão para adicionar mais para seus usos pessoais.

**id** Chave primária da tabela

**regra** campo de texto indicando a regra. Detalhes em [PAGC Registros da regra do padronizador de endereços](#).

Uma regra consiste em um conjunto de não negativos inteiros representando tokens de entrada, terminados por um -1, seguidos por um número igual de não negativos inteiros representando atributos postais, terminados por um -1, seguidos por um inteiro representando um tipo de regra, seguido por um inteiro representando o rank da regra. As regras são ranqueadas de 0 (menor) até 17 (maior).

Então por exemplo 2 0 2 22 3 -1 5 5 6 7 3 -1 2 6 mapeia para a sequência de tokens de saída *TYPE NUMBER TYPE DIRECT QUALIF* para a sequência de saída *STREET STREET SUFTYP SUFDIR QUALIF*. A regra é uma ARC\_C regra de rank 6.

Números para tokens da saída correspondentes estão listados em [stdaddr](#).

## Tokens de entrada

Cada regra começa com um conjunto de tokens de entrada seguidos por um terminator-1. Tokens de entrada extraídos de [PAGC Input Tokens](#) estão como segue:

### Tokens de entrada baseados na forma

**AMPERS** (13). O ampersand (&) é frequentemente utilizado para abreviar a palavra "e".

**DASH** (9). Um caractere de pontuação.

**DOBRO** (21). Uma sequência de duas letras. Normalmente utilizadas como identificadoras.

**FRACT** (25). Frações são usadas algumas vezes em números cívicos ou de unidade.

**MISTURADO** (23). Uma string alfanumérica que contém ambos: letras e dígitos. Usado por identificadores.

**NÚMERO** (0). Uma string de dígito.

**ORD** (15). Representações como Primeiro ou Iro. Normalmente usada em nomes de ruas.

**ORD** (18). Uma única letra.

**PALAVRA** (1). Uma palavra é uma string de letras de tamanho aleatório. Uma única letra pode ser os dois uma ÚNICA e uma PALAVRA.

### Tokens de entrada baseados na função

**BOXH** (14). Palavras usadas para indicar caixas do correio. Por exemplo *Caixa* ou *CO Caixa*.

**BUILDH** (19). Palavras usadas para indicar prédios ou condomínios, normalmente como um prefixo. Por exemplo: *Torre* em *Torre 7A*.

**BUILDT** (24). Palavras e abreviações usadas para indicar prédios ou complexos de prédios, normalmente como um sufixo. Por exemplo: *Shopping Center*.

**DIRETO** (22). Palavras usadas para indicar direções, por exemplo *Norte*.

**MILHA** (20). Palavras usadas para indicar endereços marco miliário.

**RUA** (6). Palavras e abreviações usadas para indicar estradas e ruas. Po exemplo: a *Interestadual* em *Interestadual 5*

**RR** (8). Palavras e abreviações usadas para indicar rotas rurais. *RR*.

**TIPO** (2). Palavras e abreviação usadas para indicar tipos de ruas. Por exemplo: *R* or *AV*.

**UNITH** (16). palavras e abreviação usada para indicar sub endereços. Por exemplo, *APTO* ou *UNIDADE*.

### Tokens de entrada de tipo postal

**QUÍNTUPLO** (28). Um número de 5 dígitos. Identifica um código Zip

**QUÁDRUPLO** (29). Um número de 4 dígitos. Identifica ZIP4.

**PCH** (27). Uma sequência de letra número letra de 3 caracteres. Identifica um FSA, os 3 primeiros caracteres de um código postal canadense.

**PCT** (26). Uma sequência de número letra número de 3 caracteres. Identifica um LDU, os 3 últimos caracteres de um código postal canadense.

### Palavras vazias

**PALAVRAS VAZIAS** combinadas com **PALAVRAS**. Uma string de múltiplas **PALAVRAS** e **PALAVRAS VAZIAS** será representada por uma única **PALAVRA** token.

**PALAVRA VAZIA** (7). Uma palavra com pouca significância lexical que pode ser omitida na análise sintática. Por exemplo: *O*.

### Tokens de saída

Depois do primeiro-1 (terminator), segue os tokens de saída e sua ordem, seguido por um terminator -1. Números para tokens de saída correspondentes estão listados em [stdaddr](#). Que estão permitidos é dependente em um tipo de regra. Tokens de saída válidos para cada tipo de regra estão listados em the section called “**Tipos de Regra e Classificação**”.

### Tipos de Regra e Classificação

A parte final da regra é o tipo de regra que é denotado por um dos seguintes, seguido por uma regra rank. As regras são classificadas de 0 (menor) até 17 (maior).

#### MACRO\_C

(token number = "0"). A classe de regras para as orações parsing MACRO como *PLACE STATE ZIP*

**MACRO\_C output tokens** (excerpted from <http://www.pgcgeo.org/docs/html/pagc-12.html#-r-tyt->).

**CIDADE** (número token "10"). Exemplo "Albany"

**ESTADO** (número token "11"). Exemplo "NY"

**NAÇÃO** (número token "12"). Este atributo não é usado na maioria dos arquivos de referência. Exemplo "USA"

**POSTAL** (número token "13"). (SADS elements "ZIP CODE" , "PLUS 4" ). Este atributo é usado para o US Zip e os códigos postais canadenses.

#### MICRO\_C

(número token = "1"). A classe de regras para orações parsing full MICRO (such as House, street, sufdir, predir, pretyp, suftype, qualif) (ie ARC\_C plus CIVIC\_C). Essas regras não são usadas na construção da frase.

**MICRO\_C output tokens** (excerpted from <http://www.pgcgeo.org/docs/html/pagc-12.html#-r-tyt->).

**CASA** é um texto (número token 1): Este é o número da rua em uma rua. Exemplo 75 em 75 Rua State.

**predir** é um texto (número token 2): NOME DA RUA PRE-DIRECTIONAL como Norte, Sul, Leste, Oeste etc.

**qual** é um texto (número token 3): NOME DA RUA PRE-MODIFIER Exemplo *VELHA* em 3715 ESTRADA VELHA 99.

**pré tipo** é um texto (número token 4): TIPO DE PREFIXO DA RUA

**rua** é um texto (número token 5): NOME DA RUA

**suftype** é um texto (número token 6): TIPO DE CORREIO DA RUA ex. R, Av, Cir. Um tipo de rua seguindo o nome raiz da rua. Exemplo *RUA* em 75 Rua State.

**sufdir** é um texto (número token 7): RUA POST-DIRECTIONAL Um modificador direcional que segue o nome da rua.. Exemplo *OESTE* em 3715 DÉDIMA AVENIDA OESTE.

### ARC\_C

(número token = "2"). A classe de regras para orações parsing MICRO, excluindo o atributo CASA. Como usa o mesmo conjunto de tokens de saída como MICRO\_C menos o token CASA.

### CIVIC\_C

(número token = "3"). A classe de regras para parsing o atributo da CASA.

### EXTRA\_C

(número token = "4"). A classe de regras para atributos parsing EXTRA - atributos excluídos do geocoding. Essas regras não são usadas na fase de construção.

**EXTRA\_C output tokens** (excerpted from <http://www.pgcgeo.org/docs/html/pagc-12.html#--r-typ-->).

**BLDNG** (token number 0): Unparsed identificadores e tipos de construção.

**BOXH** (token number 14): The **BOX** in BOX 3B

**BOXT** (token number 15): The **3B** in BOX 3B

**RR** (token number 8): The **RR** in RR 7

**UNITH** (token number 16): The **APT** in APT 3B

**UNITT** (token number 17): The **3B** in APT 3B

**DESCONHECIDO** (token number 9): Uma saída senão não classificada.

## 12.3.2 lex table

lex table — Uma gaz table é usada para classificar entrada e associado alfanumérico que entram com (a) tokens de entrada ( See the section called "**Tokens de entrada**") e (b) representações padronizadas.

### Descrição

Uma lex (diminutivo para léxico) table é usada para classificar entrada alfanumérica e associar que entra com the section called "**Tokens de entrada**" e (b) representações padronizadas. Coisas que você encontrará nessas tables são UM mapeado para stdword: 1.

Um lex tem pelo menos as colunas seguintes na table. Você talvez adicione

**id** Chave primária da tabela

**seq** inteiro: definição de número?

**palavra** texto: a palavra de entrada

**stdword** texto: a palavra substituta padronizada

**token** inteiro: o tipo de palavra ele é. Só se usado nesse contexto será substituído. Disponível em [PAGC Tokens](#).



### 12.3.3 gaz table

gaz table — Uma gaz table é usada para padronizar nomes de lugares e associações que entram com (a) tokens de entrada ( See the section called “[Tokens de entrada](#)”) e (b) representações padronizadas.

#### Descrição

A gaz (short for gazeteer) table is used to standardize place names and associate that input with the section called “[Tokens de entrada](#)” and (b) standardized representations. For example if you are in US, you may load these with State Names and associated abbreviations.

Uma gaz table tem pelo menos as colunas a seguir na table. Você talvez adicione mais colunas para seus próprios propósitos.

**id** Chave primária da tabela

**seq** inteiro: definição do número? - identificador usado para aquela ocasião da palavra

**palavra** texto: a palavra de entrada

**stdword** texto: a palavra substituta padronizada

**token** inteiro: o tipo de palavra ele é. Só se usado nesse contexto será substituído. Disponível em [PAGC Tokens](#).

## 12.4 Funções do padronizador de endereços

### 12.4.1 parse\_address

parse\_address — Pega um endereço linha 1 e quebra em partes

#### Synopsis

```
record parse_address(text address);
```

#### Descrição

Returns takes an address as input, and returns a record output consisting of fields *num*, *street*, *street2*, *address1*, *city*, *state*, *zip*, *zipplus*, *country*.

Disponibilidade: 2.2.0



This method needs address\_standardizer extension.

#### Exemplos

##### Endereços Únicos

```
SELECT num, street, city, zip, zipplus
       FROM parse_address('1 Devonshire Place, Boston, MA 02109-1234') AS a;
```

```
num | street | city | zip | zipplus
-----+-----+-----+-----+-----
  1 | Devonshire Place | Boston | 02109 | 1234
```

Table de endereços

```

-- basic table
CREATE TABLE places(addid serial PRIMARY KEY, address text);

INSERT INTO places(address)
VALUES ('529 Main Street, Boston MA, 02129'),
       ('77 Massachusetts Avenue, Cambridge, MA 02139'),
       ('25 Wizard of Oz, Walaford, KS 99912323'),
       ('26 Capen Street, Medford, MA'),
       ('124 Mount Auburn St, Cambridge, Massachusetts 02138'),
       ('950 Main Street, Worcester, MA 01610');

-- parse the addresses
-- if you want all fields you can use (a).*
SELECT addid, (a).num, (a).street, (a).city, (a).state, (a).zip, (a).zipplus
FROM (SELECT addid, parse_address(address) As a
      FROM places) AS p;

```

addid	num	street	city	state	zip	zipplus
1	529	Main Street	Boston	MA	02129	
2	77	Massachusetts Avenue	Cambridge	MA	02139	
3	25	Wizard of Oz	Walaford	KS	99912	323
4	26	Capen Street	Medford	MA		
5	124	Mount Auburn St	Cambridge	MA	02138	
6	950	Main Street	Worcester	MA	01610	

(6 rows)

## Veja também

### 12.4.2 standardize\_address

`standardize_address` — Retorna uma forma `stdaddr` de um endereço de entrada utilizando `lex`, `gaz` e `rule tables`.

#### Synopsis

```

stdaddr standardize_address(text lextab, text gaztab, text rultab, text address);
stdaddr standardize_address(text lextab, text gaztab, text rultab, text micro, text macro);

```

#### Descrição

Retorna a uma `stdaddr` forma de um endereço de entrada utilizando `lex table` table nome, `gaz table`, e `mesa de regras table` nomes e endereço.

Variante 1: Pega um endereço como uma única linha.

Variante 2: Pega o endereço em duas partes. Uma `micro` que consiste em padronizar a primeira linha do endereço postal ex. `house_num street`, e uma `macro` que consiste em adronizar a segunda linha de um endereço postal ex. `city, state postal_code country`.

Disponibilidade: 2.2.0



This method needs `address_standardizer` extension.

## Exemplos

### Using address\_standardizer\_data\_us extension

```
CREATE EXTENSION address_standardizer_data_us; -- only needs to be done once
```

### Variante 1: Única linha de endereço. Isso não funcionou bem com os endereços não-EUA

```
SELECT house_num, name, suftype, city, country, state, unit FROM standardize_address(' ←
us_lex',
                                     'us_gaz', 'us_rules', 'One Devonshire Place, PH 301, Boston, MA ←
                                     02109');
```

house_num	name	suftype	city	country	state	unit
1	DEVONSHIRE	PLACE	BOSTON	USA	MASSACHUSETTS	# PENTHOUSE 301

Utilizando tables compactadas com o geocoder tiger. Este exemplo só funciona se você instalou `postgis_tiger_geocoder`.

```
SELECT * FROM standardize_address('tiger.pagc_lex',
                                   'tiger.pagc_gaz', 'tiger.pagc_rules', 'One Devonshire Place, PH 301, Boston, MA ←
                                   02109-1234');
```

Para tornar a leitura mais fácil nós iremos abandonar a saída usando a extensão `hstore` `CREATE EXTENSION hstore`; você vai precisar instalar

```
SELECT (each(hstore(p))).*
FROM standardize_address('tiger.pagc_lex', 'tiger.pagc_gaz',
                         'tiger.pagc_rules', 'One Devonshire Place, PH 301, Boston, MA 02109') As p;
```

key	value
box	
city	BOSTON
name	DEVONSHIRE
qual	
unit	# PENTHOUSE 301
extra	
state	MA
predir	
sufdir	
country	USA
pretype	
suftype	PL
building	
postcode	02109
house_num	1
ruralroute	

(16 rows)

### variante 2: Como um endereço de duas partes

```
SELECT (each(hstore(p))).*
FROM standardize_address('tiger.pagc_lex', 'tiger.pagc_gaz',
                         'tiger.pagc_rules', 'One Devonshire Place, PH 301', 'Boston, MA 02109, US') As p;
```

key	value
box	
city	BOSTON
name	DEVONSHIRE

```
qual      |
unit      | # PENTHOUSE 301
extra     |
state     | MA
predir    |
sufdir    |
country   | USA
pretype   |
suftype   | PL
building  |
postcode  | 02109
house_num | 1
ruralroute |
(16 rows)
```

**Veja também**

[stdaddr](#), [mesa de regras](#), [lex table](#), [gaz table](#), [Pagc\\_Normalize\\_Address](#)

## Chapter 13

# PostGIS Extras

This chapter documents features found in the extras folder of the PostGIS source tarballs and source repository. These are not always packaged with PostGIS binary releases, but are usually plpgsql based or standard shell scripts that can be run as is.

### 13.1 Tiger Geocoder

Existem outras fontes abertas geocoder para o PostGIS, que, ao contrário do tiger geocoder, têm a vantagem do suporte geocoding para muitos países

- **Nominatim** usa dados OpenStreetMap gazeteer formatados. Requer o osm2pgsql para carregar os dados, PostgreSQL 8.4+ e PostGIS 1.5+ para funcionar. É compactado como uma interface de serviço da web e parece ter sido criado para ser chamado como webservice. Assim como o geocoder, ele tem dois componentes: o geocoder e o geocoder reverso. Na documentação não fica claro se ele tem uma interface SQL pura conforme o geocoder ou se tem um bom acordo da lógica implementado na interface da web.
- **GIS Graphy** também utiliza PostGIS e, como Nominatim, funciona com os dados OpenStreetMap (OSM). Ele possui um carregador para carregar dados OSM e, correspondente ao Nominatim, é capaz de geocoding não só nos EUA. Bem como Nominatim, ele executa como webservice e confia no Java 1.5, Servlet apps, Solr. O GisGraphy é uma multiplataforma e também possui um geocoder reverso juntamente com outros aspectos.

#### 13.1.1 Drop\_Indexes\_Generate\_Script

`Drop_Indexes_Generate_Script` — Gera uma script que derruba todas as chaves não primárias e indexes não únicos no esquema tiger e esquema especificado de usuário. Padroniza esquema para: `tiger_data` se nenhum esquema é especificado.

##### Synopsis

```
text Drop_Indexes_Generate_Script(text param_schema=tiger_data);
```

##### Descrição

Gera uma script que derruba todas as chaves não primárias e indexes não únicos no esquema tiger e esquema especificado de usuário. Padroniza esquema para: `tiger_data` se nenhum esquema é especificado.

Isso é útil para minimizar o excesso de indexes que pode confundir o organizador de pesquisas ou ocupar um espaço desnecessário. Use combinado com [Install\\_Missing\\_Indexes](#), para adicionar os indexes usados pelo geocoder.

Disponibilidade: 2.0.0

---

## Exemplos

```
SELECT drop_indexes_generate_script() As actionsql;
actionsql
-----
DROP INDEX tiger.idx_tiger_countysub_lookup_lower_name;
DROP INDEX tiger.idx_tiger_edges_countyfp;
DROP INDEX tiger.idx_tiger_faces_countyfp;
DROP INDEX tiger.tiger_place_the_geom_gist;
DROP INDEX tiger.tiger_edges_the_geom_gist;
DROP INDEX tiger.tiger_state_the_geom_gist;
DROP INDEX tiger.idx_tiger_addr_least_address;
DROP INDEX tiger.idx_tiger_addr_tlid;
DROP INDEX tiger.idx_tiger_addr_zip;
DROP INDEX tiger.idx_tiger_county_countyfp;
DROP INDEX tiger.idx_tiger_county_lookup_lower_name;
DROP INDEX tiger.idx_tiger_county_lookup_snd_name;
DROP INDEX tiger.idx_tiger_county_lower_name;
DROP INDEX tiger.idx_tiger_county_snd_name;
DROP INDEX tiger.idx_tiger_county_the_geom_gist;
DROP INDEX tiger.idx_tiger_countysub_lookup_snd_name;
DROP INDEX tiger.idx_tiger_cousub_countyfp;
DROP INDEX tiger.idx_tiger_cousub_cousubfp;
DROP INDEX tiger.idx_tiger_cousub_lower_name;
DROP INDEX tiger.idx_tiger_cousub_snd_name;
DROP INDEX tiger.idx_tiger_cousub_the_geom_gist;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_least_address;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_tlid;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_zip;
DROP INDEX tiger_data.idx_tiger_data_ma_county_countyfp;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lookup_lower_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lookup_snd_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lower_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_snd_name;
:
:
```

## Veja Também

[Install\\_Missing\\_Indexes](#), [Missing\\_Indexes\\_Generate\\_Script](#)

### 13.1.2 Drop\_Nation\_Tables\_Generate\_Script

`Drop_Nation_Tables_Generate_Script` — Gera uma script que derruba todas as tables no esquema específico que começa com `county_all`, `state_all` ou código de estado seguido por condado ou estado.

#### Synopsis

```
text Drop_Nation_Tables_Generate_Script(text param_schema=tiger_data);
```

#### Descrição

Gera uma script que derruba todas as tables no esquema específico que começa com `county_all`, `state_all` ou código de estado seguido por condado ou estado. Isso é necessário se você está atualizando os dados do `tiger_2010` para o `tiger_2011`.

Disponibilidade: 2.1.0

## Exemplos

```
SELECT drop_nation_tables_generate_script();
DROP TABLE tiger_data.county_all;
DROP TABLE tiger_data.county_all_lookup;
DROP TABLE tiger_data.state_all;
DROP TABLE tiger_data.ma_county;
DROP TABLE tiger_data.ma_state;
```

## Veja Também

[Loader\\_Generate\\_Nation\\_Script](#)

### 13.1.3 Drop\_State\_Tables\_Generate\_Script

**Drop\_State\_Tables\_Generate\_Script** — Gera uma script que derruba todas as tables no esquema específico que estão prefixados com abreviação do estado. Padroniza o esquema para `tiger_data` se nenhum esquema estiver especificado.

#### Synopsis

```
text Drop_State_Tables_Generate_Script(text param_state, text param_schema=tiger_data);
```

#### Descrição

Gera uma script que derruba todas as tables no esquema específico que estão prefixados com abreviação do estado. Padroniza o esquema para `tiger_data` se nenhum esquema estiver especificado. Essa função é útil para derrubar tables de um estado antes de recarregar um estado em caso de algo ter dado errado durante seu carregamento anterior.

Disponibilidade: 2.0.0

## Exemplos

```
SELECT drop_state_tables_generate_script('PA');
DROP TABLE tiger_data.pa_addr;
DROP TABLE tiger_data.pa_county;
DROP TABLE tiger_data.pa_county_lookup;
DROP TABLE tiger_data.pa_cousub;
DROP TABLE tiger_data.pa_edges;
DROP TABLE tiger_data.pa_faces;
DROP TABLE tiger_data.pa_featnames;
DROP TABLE tiger_data.pa_place;
DROP TABLE tiger_data.pa_state;
DROP TABLE tiger_data.pa_zip_lookup_base;
DROP TABLE tiger_data.pa_zip_state;
DROP TABLE tiger_data.pa_zip_state_loc;
```

## Veja Também

[Loader\\_Generate\\_Script](#)

### 13.1.4 Geocode

Geocode — Assimila um endereço como uma string (ou outro endereço normalizado) e gera um conjunto de localizações possíveis que inclui um ponto em NAD 83 long lat, um endereço normalizado para cada um e a avaliação. Quanto menor a avaliação, maior a chance de combinar. Os resultados são separados com menor avaliação em primeiro lugar. Pode passar no resultados máximos, até 10, e `restrict_region` (padrão NULO)

#### Synopsis

```
setof record geocode(varchar address, integer max_results=10, geometry restrict_region=NULL, norm_addy OUT addy, geometry OUT geomout, integer OUT rating);
```

```
setof record geocode(norm_addy in_addy, integer max_results=10, geometry restrict_region=NULL, norm_addy OUT addy, geometry OUT geomout, integer OUT rating);
```

#### Descrição

Assimila um endereço como uma string (ou endereço já normalizado) e gera uma série de possíveis localizações que inclui um ponto em NAD 83 long lat, um `normalized_address` (`addy`) para cada e a avaliação. Quanto menor a avaliação, maior a chance de combinar. Os resultados são separados com a menor avaliação em primeiro lugar. Usa os dados (limites, faces, `addr`) Tiger, uma string confusa PostgreSQL (`soundex,levenshtein`) linha de interpolação PostGIS para interpolar endereços ao longo dos limites do Tiger. Quanto maior a avaliação, menos o geocoder estará correto. O ponto geocodificado é padronizado para compensar 10 metros da linha central do lado (E/D) que o endereço da rua está localizado.

Melhorias: 2.0.0 para suportar o Tiger 2010, dados estruturados e lógica revisada para melhorar a velocidade, exatidão do geocoding e para compensar ponto da linha central para o lado do endereço que a rua está localizada. O novo parâmetro `max_results` é útil para especificar números dos melhores resultados ou apenas retornar o melhor resultado.

#### Exemplos: Básico

Os exemplos abaixo estão em um único processador 3.0 GHZ no Windows 7 com 2GB ram executando PostgreSQL 9.1rc1/PostGIS 2.0 carregados com todos os dados de estado Tiger MA,MN,CA, RI.

Combinações exatas são mais fáceis de computar (61ms)

```
SELECT g.rating, ST_X(g.geomout) As lon, ST_Y(g.geomout) As lat,
       (addy).address As stno, (addy).streetname As street,
       (addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st, ( ←
       addy).zip
FROM geocode('75 State Street, Boston MA 02109') As g;
rating |          lon          |          lat          | stno | street | styp | city | st | zip
-----+-----+-----+-----+-----+-----+-----+-----+-----+----- ←
      0 | -71.0556722990239 | 42.3589914927049 | 75 | State | St | Boston | MA | 02109
```

Mesmo se o zip não tiver passado no geocode pode estimar (demorou cerca de 122-150 ms)

```
SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktmlonlat,
       (addy).address As stno, (addy).streetname As street,
       (addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st, ( ←
       addy).zip
FROM geocode('226 Hanover Street, Boston, MA',1) As g;
rating |          wktmlonlat          | stno | street | styp | city | st | zip
-----+-----+-----+-----+-----+-----+-----+-----+----- ←
      1 | POINT(-71.05528 42.36316) | 226 | Hanover | St | Boston | MA | 02113
```

Sabe lidar com erros de ortografia e fornece mais de uma possibilidade de solução com avaliações e tomadas maiores (500ms).



```

SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktmlonlat,
      (addy).address As stno, (addy).streetname As street,
      (addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st, ( ←
      addy).zip
FROM geocode('31 - 37 Stewart Street, Boston, MA 02116') As g;
rating |          wktmlonlat          | stno | street | styp | city | st | zip
-----+-----+-----+-----+-----+-----+-----+-----
70 | POINT(-71.06459 42.35113) | 31 | Stuart | St  | Boston | MA | 02116

```

Utilizando para fazer um agrupamento geocode de endereços. Mais fácil para configurar `max_results=1`. Processa somente aqueles que ainda não foram geocodificados (não possuem avaliação).

```

CREATE TABLE addresses_to_geocode(addid serial PRIMARY KEY, address text,
      lon numeric, lat numeric, new_address text, rating integer);

INSERT INTO addresses_to_geocode(address)
VALUES ('529 Main Street, Boston MA, 02129'),
      ('77 Massachusetts Avenue, Cambridge, MA 02139'),
      ('25 Wizard of Oz, Walaford, KS 99912323'),
      ('26 Capen Street, Medford, MA'),
      ('124 Mount Auburn St, Cambridge, Massachusetts 02138'),
      ('950 Main Street, Worcester, MA 01610');

-- only update the first 3 addresses (323-704 ms - there are caching and shared memory ←
-- effects so first geocode you do is always slower) --
-- for large numbers of addresses you don't want to update all at once
-- since the whole geocode must commit at once
-- For this example we rejoin with LEFT JOIN
-- and set to rating to -1 rating if no match
-- to ensure we don't regeocode a bad address
UPDATE addresses_to_geocode
  SET (rating, new_address, lon, lat)
    = ( COALESCE((g.geo).rating,-1), pprint_addy((g.geo).addy),
        ST_X((g.geo).geomout)::numeric(8,5), ST_Y((g.geo).geomout)::numeric(8,5) )
FROM (SELECT addid
      FROM addresses_to_geocode
      WHERE rating IS NULL ORDER BY addid LIMIT 3) As a
LEFT JOIN (SELECT addid, (geocode(address,1)) As geo
      FROM addresses_to_geocode As ag
      WHERE ag.rating IS NULL ORDER BY addid LIMIT 3) As g ON a.addid = g.addid
WHERE a.addid = addresses_to_geocode.addid;

result
-----
Query returned successfully: 3 rows affected, 480 ms execution time.

SELECT * FROM addresses_to_geocode WHERE rating is not null;

addid |          address          | lon | lat | ←
-----+-----+-----+-----+-----+-----+-----+-----
1 | 529 Main Street, Boston MA, 02129 | -71.07181 | 42.38359 | 529 Main St, ←
  | Boston, MA 02129 | 0
2 | 77 Massachusetts Avenue, Cambridge, MA 02139 | -71.09428 | 42.35988 | 77 ←
  | Massachusetts Ave, Cambridge, MA 02139 | 0
3 | 25 Wizard of Oz, Walaford, KS 99912323 | | | ←
  | | -1

```

### Exemplos: Usando Filtros Geométricos

```
SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktlonlat,
       (addy).address As stno, (addy).streetname As street,
       (addy).streettypeabbrev As styp,
       (addy).location As city, (addy).stateabbrev As st, (addy).zip
FROM geocode('100 Federal Street, MA',
            3,
            (SELECT ST_Union(the_geom)
             FROM place WHERE statefp = '25' AND name = 'Lynn')::geometry
            ) As g;
```

rating	wktlonlat	stno	street	styp	city	st	zip
8	POINT(-70.96796 42.4659)	100	Federal	St	Lynn	MA	01905

Total query runtime: 245 ms.

### Veja Também

[Normalize\\_Address](#), [Pprint\\_Addy](#), [ST\\_AsText](#), [ST\\_SnapToGrid](#), [ST\\_X](#), [ST\\_Y](#)

### 13.1.5 Geocode\_Intersection

**Geocode\_Intersection** — Assimila 2 ruas que se intersectam e um estado, cidade, zip, e gera um conjunto de possíveis localizações no primeiro cruzamento que está na intersecção, também inclui um geomout como o ponto de localização em NAD 83 long lat, um `normalized_address (addy)` para cada localização, e a avaliação. Quanto menor a avaliação, maior a chance de combinar. Os resultados são separados com menor avaliação em primeiro lugar. Pode passar nos resultados máximos, até 10. Usa dados Tiger (limites, faces, addr), string confusa do PostgreSQL (soundex, evenshtein).

### Synopsis

```
setof record geocode_intersection(text roadway1, text roadway2, text in_state, text in_city, text in_zip, integer max_results=10,
norm_addy OUT addy, geometry OUT geomout, integer OUT rating);
```

### Descrição

Assimila 2 ruas que se intersectam e um estado, cidade, zip, e gera um conjunto de possíveis localizações no primeiro cruzamento que está na intersecção, também inclui um geomout como o ponto de localização em NAD 83 long lat para cada localização, e a avaliação. Quanto menor a avaliação, maior a chance de combinar. Os resultados são separados com menor avaliação em primeiro lugar. Pode passar nos resultados máximos, até 10. Retorna `normalized_address (addy)` para cada, geomout como o ponto da localização em nad 83 long lat, and the rating. Quanto menor a avaliação, maior a chance de combinar. Os resultados são separados com menor avaliação em primeiro lugar. Usa dados Tiger (limites, faces, addr), string confusa do PostgreSQL (soundex, evenshtein).

Disponibilidade: 2.0.0

### Exemplos: Básico

Os exemplos abaixo estão em um único processador 3.0 GHZ no Windows 7 com 2GB ram executando PostgreSQL 9.0/PostGIS 1.5 carregados com todos os dados de estado MA Tiger carregados. Atualmente um pouco devagar (3000 ms)

Testando no Windows 2003 64-bit 8GB on PostGIS 2.0 PostgreSQL 64-bit Tiger 2011 dados carregados -- (41ms)

```
SELECT pprint_addy(addy), st_astext(geomout),rating
      FROM geocode_intersection('Haverford St','Germania St','MA','Boston',↔
      '02130',1);
```

pprint_addy	st_astext	rating
98 Haverford St, Boston, MA 02130	POINT(-71.101375 42.31376)	0

Mesmo se o zip não passar no geocoder pode estimar (demorou cerca de 3500 ms na caixa do windows 7), no o windows 2003 64-bit 741 ms

```
SELECT pprint_addy(addy), st_astext(geomout),rating
      FROM geocode_intersection('Weld', 'School', 'MA', 'Boston');
```

pprint_addy	st_astext	rating
98 Weld Ave, Boston, MA 02119	POINT(-71.099 42.314234)	3
99 Weld Ave, Boston, MA 02119	POINT(-71.099 42.314234)	3

**Veja Também**

[Geocode](#), [Pprint\\_Addy](#), [ST\\_AsText](#)

**13.1.6 Get\_Geocode\_Setting**

Get\_Geocode\_Setting — Retorna a configuração de valor específico armazenada na table tiger.geocode\_settings.

**Synopsis**

text **Get\_Geocode\_Setting**(text setting\_name);

**Descrição**

Retorna valor da configuração específica armazenada na table tiger.geocode\_settings. As configurações te permitem comutar depuração de funções. Planos futuros serão para controlar a avaliação com as configurações. A seguir, a lista atual de configurações:

name	setting	unit	category	↔	short_desc
debug_geocode_address	false		boolean	debug	outputs debug information ↔ in notice log such as queries when geocode_address is called if true
debug_geocode_intersection	false		boolean	debug	outputs debug information ↔ in notice log such as queries when geocode_intersection is called if true
debug_normalize_address	false		boolean	debug	outputs debug information ↔ in notice log such as queries and intermediate expressions when normalize_address is ↔ called if true
debug_reverse_geocode	false		boolean	debug	if true, outputs debug ↔ information in notice log such as queries and intermediate expressions when ↔ reverse_geocode
reverse_geocode_numbered_roads_highways,	0		integer	rating	For state and county ↔ 0 - no preference in name, 1 - prefer the numbered ↔ highway name, 2 - ↔ prefer local state/ ↔ county name

```

use_pgc_address_parser      | false      | boolean | normalize | If set to true, will try ←
                           |            |         |           | to use the address_standardizer extension (via pgc_normalize_address)
                                                                    | instead of tiger ←
                                                                    | normalize_address built ←
                                                                    | one

```

Alterações: 2.2.0 : configurações padrão são guardadas em uma table chamada `geocode_settings_default`. As configurações personalizadas estão em `geocode_settings` e só contém aquelas que foram configuradas pelo usuário.

Disponibilidade: 2.1.0

### Exemplo da configuração de retornar depuração

```

SELECT get_geocode_setting('debug_geocode_address') As result;
result
-----
false

```

### Veja Também

[Set\\_Geocode\\_Setting](#)

## 13.1.7 Get\_Tract

`Get_Tract` — Retorna o trecho ou campo de uma `tract` table onde a geometria está localizada. Padrão para retornar um nome curto para o trecho.

### Synopsis

```
text get_tract(geometry loc_geom, text output_field=name);
```

### Descrição

Uma dada geometria irá retornar o trecho da localização do censo daquela geometria. NAD 83 long lat é assumida se nenhum spatial ref sys estiver especificado.

#### Note

This function uses the census `tract` which is not loaded by default. If you have already loaded your state table, you can load `tract` as well as `bg`, and `tabblock` using the [Loader\\_Generate\\_Census\\_Script](#) script.

If you have not loaded your state data yet and want these additional tables loaded, do the following

```

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

```

then they will be included by the [Loader\\_Generate\\_Script](#).

Disponibilidade: 2.0.0

## Exemplos: Básico

```
SELECT get_tract(ST_Point(-71.101375, 42.31376) ) As tract_name;
tract_name
-----
1203.01
```

```
--this one returns the tiger geoid
SELECT get_tract(ST_Point(-71.101375, 42.31376), 'tract_id' ) As tract_id;
tract_id
-----
25025120301
```

## Veja Também

[Geocode](#) >

### 13.1.8 Install\_Missing\_Indexes

`Install_Missing_Indexes` — Encontra todas as tables com colunas chave usadas no ingresso geocoder e condições de filtros que estão perdendo os indexes usados nessas colunas e irão adicionar elas.

#### Synopsis

boolean `Install_Missing_Indexes()`;

#### Descrição

Encontra todas as tables nos esquemas `tiger` e `tiger_data` com as colunas chave usadas no ingresso e filtros do geocoder que estão perdendo indexes nessas colunas e irão gerar o SQL DDL para definir o index para aquelas tables e, então, executar a script gerada. Essa é uma função ajudante, que adiciona novos indexes necessários para pesquisas mais rápidas que podem ter sido perdidas durante o carregamento. Essa função é uma acompanhante para [Missing\\_Indexes\\_Generate\\_Script](#), que somada à script que cria index, também a executa. Ela é uma parte da script de atualização `update_geocode.sql`.

Disponibilidade: 2.0.0

#### Exemplos

```
SELECT install_missing_indexes();
install_missing_indexes
-----
t
```

## Veja Também

[Loader\\_Generate\\_Script](#), [Missing\\_Indexes\\_Generate\\_Script](#)

### 13.1.9 Loader\_Generate\_Census\_Script

`Loader_Generate_Census_Script` — Gera uma shell script para a plataforma específica para os estados que irão baixar o trecho do censo de estado Tiger, bg e dados de tables `tabblocks`, arranjar e carregar dentro do esquema `tiger_data`. Cada state script retornou como um relato separado.

## Synopsis

```
setof text loader_generate_census_script(text[] param_states, text os);
```

## Descrição

Gera uma shell script para a plataforma específica para os estados que irão baixar `tract` do censo de estado Tiger, block groups `bg` e dados de tables `tabblocks`, arranja e carrega dentro do esquema `tiger_data`. Cada state script retornou como um relato separado.

Utiliza `unzip` no Linux (7-zip no Windows por padrão) e `wget` para fazer o download. Usa Section 4.4.2 para carregar nos dados. Note que a menor unidade que ele faz é um estado inteiro. Ele só irá processar os arquivos nas pastas representativas e temporárias.

Isso usa as seguintes tables de controle para controlar o processo e diferentes variações de sintaxe OS shell.

1. `loader_variables` armazena pistas de várias variáveis como o site do censo, ano, dados e esquemas representativos.
2. `loader_platform` perfis de numerosas plataformas e onde as várias executáveis estão localizadas. Está com o windows e linux. Mais pode ser adicionado.
3. `loader_lookuptables` cada relato define um tipo de table (estado, condado), quer para processar relatos nelas ou para carregar eles. Define os passos para importar dados, dados de representação, adicionar, remove colunas, indexes e restrições para cada um. Cada table é prefixada com o estado de uma table em um esquema tiger. ex: cria `tiger_data.ma_faces`, os quais herda das `tiger.faces`

Disponibilidade: 2.0.0



### Note

**Loader\_Generate\_Script** inclui essa lógica, mas se você instalou o geocoder tiger antes para o PostGIS 2.0.0 alpha5, você vai precisar executar esse nos estados que já fez para pegar essas tables adicionais.

## Exemplos

Gerar script para carregar dados para selecionar estados no formato script shell do Windows.

```
SELECT loader_generate_census_script(ARRAY['MA'], 'windows');
-- result --
set STATEDIR="\gisdata\www2.census.gov\geo\pvs\tiger2010st\25_Massachusetts"
set TMPDIR=\gisdata\temp\
set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"
set WGETTOOL="C:\wget\wget.exe"
set PGBIN=C:\projects\pg\pg91win\bin\
set PGPORT=5432
set PGHOST=localhost
set PGUSER=postgres
set PGPASSWORD=yourpasswordhere
set PGDATABASE=tiger_postgis20
set PSQL="%PGBIN%psql"
set SHP2PGSQL="%PGBIN%shp2pgsql"
cd \gisdata

%WGETTOOL% http://www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts/25/ --no-parent -- ←
    relative --accept=*bg10.zip,*tract10.zip,*tabblock10.zip --mirror --reject=html
del %TMPDIR%\*.* /Q
%PSQL% -c "DROP SCHEMA tiger_staging CASCADE;"
%PSQL% -c "CREATE SCHEMA tiger_staging;"
```

```

cd %STATEDIR%
for /r %%z in (*.zip) do %UNZIPTOOL% e %%z -o%TMPDIR%
cd %TMPDIR%
%PSQL% -c "CREATE TABLE tiger_data.MA_tract(CONSTRAINT pk_MA_tract PRIMARY KEY (tract_id) ) ←
    INHERITS(tiger.tract); "
%SHP2PGSQL% -c -s 4269 -g the_geom -W "latin1" tl_2010_25_tract10.dbf tiger_staging. ←
    ma_tract10 | %PSQL%
%PSQL% -c "ALTER TABLE tiger_staging.MA_tract10 RENAME geoid10 TO tract_id; SELECT ←
    loader_load_staged_data(lower('MA_tract10'), lower('MA_tract')); "
%PSQL% -c "CREATE INDEX tiger_data_MA_tract_the_geom_gist ON tiger_data.MA_tract USING gist ←
    (the_geom);"
%PSQL% -c "VACUUM ANALYZE tiger_data.MA_tract;"
%PSQL% -c "ALTER TABLE tiger_data.MA_tract ADD CONSTRAINT chk_statefp CHECK (statefp = ←
    '25');"
:

```

### Gerar script sh

```

STATEDIR="/gisdata/www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts"
TMPDIR="/gisdata/temp/"
UNZIPTOOL=unzip
WGETTOOL="/usr/bin/wget"
export PGBIN=/usr/pgsql-9.0/bin
export PGPORT=5432
export PGHOST=localhost
export PGUSER=postgres
export PGPASSWORD=yourpasswordhere
export PGDATABASE=geocoder
PSQL=${PGBIN}/psql
SHP2PGSQL=${PGBIN}/shp2pgsql
cd /gisdata

wget http://www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts/25/ --no-parent --relative ←
    --accept=*bg10.zip,*tract10.zip,*tabblock10.zip --mirror --reject=html
rm -f ${TMPDIR}/*. *
${PSQL} -c "DROP SCHEMA tiger_staging CASCADE;"
${PSQL} -c "CREATE SCHEMA tiger_staging;"
cd $STATEDIR
for z in *.zip; do $UNZIPTOOL -o -d $TMPDIR $z; done
:
:

```

### Veja Também

[Loader\\_Generate\\_Script](#)

#### 13.1.10 Loader\_Generate\_Script

**Loader\_Generate\_Script** — Gera uma shell script para a plataforma específica para os estados que irão baixar dados Tiger, arrumar e carregar dentro do esquema `tiger_data`. Cada state script retorna como um registro separado. A versão mais nova suporta mudanças estruturais do Tiger 2010 e também carrega trecho do censo, block groups, e block tables.

#### Synopsis

```
setof text loader_generate_script(text[] param_states, text os);
```

## Descrição

Gera uma shell script para a plataforma específica para os estados que irão baixar dados do Tiger, arranjar e carregar dentro do esquema `tiger_data`. Cada state script retorna como um registro separado.

Utiliza `unzip` no Linux (7-zip no Windows por padrão) e `wget` para fazer o download. Usa Section 4.4.2 para carregar nos dados. Note que a menor unidade que ele faz é um estado inteiro, mas você pode sobrescrever baixando os arquivos por conta própria. Ele só irá processar os arquivos nas pastas representativas e temporárias.

Isso usa as seguintes tables de controle para controlar o processo e diferentes variações de sintaxe OS shell.

1. `loader_variables` armazena pistas de várias variáveis como o site do censo, ano, dados e esquemas representativos.
2. `loader_platform` perfis de numerosas plataformas e onde as várias executáveis estão localizadas. Está com o `windows` e `linux`. Mais pode ser adicionado.
3. `loader_lookuptables` cada relato define um tipo de table (estado, condado), quer para processar relatos nelas ou para carregar eles. Define os passos para importar dados, dados de representação, adicionar, remove colunas, indexes e restrições para cada um. Cada table é prefixada com o estado de uma table em um esquema `tiger`. ex: cria `tiger_data.ma_faces`, os quais herda das `tiger.faces`

Disponibilidade: 2.0.0 para suportar tiger 2010 dados estruturados e carrega trecho (trecho) do censo , block groups (bg), e block (tabblocks) tables.



### Note

If you are using pgAdmin 3, be warned that by default pgAdmin 3 truncates long text. To fix, change *File -> Options -> Query Tool -> Query Editor -> Max. characters per column* to larger than 50000 characters.

## Exemplos

Using `psql` where `gistest` is your database and `/gisdata/data_load.sh` is the file to create with the shell commands to run.

```
psql -U postgres -h localhost -d gistest -A -t \
-c "SELECT Loader_Generate_Script (ARRAY['MA'], 'gistest') " > /gisdata/data_load.sh;
```

Gerar script para carregar dados para 2 estados na script de formato shell do Windows.

```
SELECT loader_generate_script (ARRAY['MA','RI'], 'windows') AS result;
-- result --
set TMPDIR=\gisdata\temp\
set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"
set WGETTOOL="C:\wget\wget.exe"
set PGBIN=C:\Program Files\PostgreSQL\9.4\bin\
set PGPORT=5432
set PGHOST=localhost
set PGUSER=postgres
set PGPASSWORD=yourpasswordhere
set PGDATABASE=geocoder
set PSQL="%PGBIN%psql"
set SHP2PGSQL="%PGBIN%shp2pgsql"
cd \gisdata

cd \gisdata
%WGETTOOL% ftp://ftp2.census.gov/geo/tiger/TIGER2015/PLACE/tl*_25_* --no-parent --relative ←
--recursive --level=2 --accept=zip --mirror --reject=html
cd \gisdata/ftp2.census.gov/geo/tiger/TIGER2015/PLACE
:
:
```



## Gerar script sh

```

STATEDIR="/gisdata/www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts"
TMPDIR="/gisdata/temp/"
UNZIPTOOL=unzip
WGETTOOL="/usr/bin/wget"
export PGBIN=/usr/pgsql-9.0/bin
export PGPORT=5432
export PGHOST=localhost
export PGUSER=postgres
export PGPASSWORD=yourpasswordhere
export PGDATABASE=geocoder
PSQL=${PGBIN}/psql
SHP2PGSQL=${PGBIN}/shp2pgsql
cd /gisdata

wget http://www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts/25/ --no-parent --relative ←
    --accept=*bg10.zip,*tract10.zip,*tabblock10.zip --mirror --reject=html
rm -f ${TMPDIR}/*. *
${PSQL} -c "DROP SCHEMA tiger_staging CASCADE;"
${PSQL} -c "CREATE SCHEMA tiger_staging;"
cd $STATEDIR
for z in *.zip; do $UNZIPTOOL -o -d $TMPDIR $z; done
:
:

```

## Veja Também

Section 2.8.1, [Loader\\_Generate\\_Nation\\_Script](#)

### 13.1.11 Loader\_Generate\_Nation\_Script

`Loader_Generate_Nation_Script` — Gerar uma script shell para a plataforma especificada que carrega as lookup tables de condado e estado.

#### Synopsis

```
text loader_generate_nation_script(text os);
```

#### Descrição

Gera uma script shell para a plataforma especificada que carrega as tables `county_all`, `county_all_lookup`, `state_all` dentro do esquema `tiger_data`. Elas herdam respectivamente das tables `county`, `county_lookup`, `state` no esquema `tiger`.

Utiliza `unzip` no Linux (7-zip no Windows por padrão) e `wget` para fazer o download. Usa Section 4.4.2 para carregar nos dados.

Utiliza as seguintes tables de controle: `tiger.loader_platform`, `tiger.loader_variables`, e `tiger.loader_lookup` para controlar o processo e diferentes variações de sintaxe OS shell.

1. `loader_variables` armazena pistas de várias variáveis como o site do censo, ano, dados e esquemas representativos.
2. `loader_platform` perfis de numerosas plataformas e onde as várias executáveis estão localizadas. Está com o windows e linux/unix. Mais pode ser adicionado.
3. `loader_lookuptables` cada relato define um tipo de table (estado, condado), quer para processar relatos nelas ou para carregar eles. Define os passos para importar dados, dados de representação, adicionar, remove colunas, indexes e restrições para cada um. Cada table é prefixada com o estado de uma table em um esquema `tiger`. ex: cria `tiger_data.ma_faces`, os quais herda das `tiger.faces`

Enhanced: 2.4.1 zip code 5 tabulation area (zcta5) load step was fixed and when enabled, zcta5 data is loaded as a single table called zcta5\_all as part of the nation script load.

Disponibilidade: 2.1.0

**Note**

If you want zip code 5 tabulation area (zcta5) to be included in your nation script load, do the following:

```
UPDATE tiger.loader_lookuptables SET load = true WHERE table_name = 'zcta510';
```

**Note**

Se você estiver executando a versão `tiger_2010` e quer recarregar como estado com `tiger_2011`, você vai precisar, para o primeiro carregamento, gerar e executar drop statements [Drop\\_Nation\\_Tables\\_Generate\\_Script](#) antes de executar essa script.

## Exemplos

Gerar script para carregar dados de uma nação no Windows.

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

Gerar script para carregar dados para os sistemas Linux/Unix.

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

## Veja Também

[Loader\\_Generate\\_Script](#)

### 13.1.12 Missing\_Indexes\_Generate\_Script

`Missing_Indexes_Generate_Script` — Encontra todas as tables com colunas chave usadas no ingresso geocoder que estão perdendo indexes nessas colunas e irão gerar o SQL DDL para definir o index para essas tables.

## Synopsis

```
text Missing_Indexes_Generate_Script();
```

## Descrição

Encontra todas as tables nos esquemas `tiger` e `tiger_data` com as colunas chave usadas no ingresso geocoder que está perdendo indexes nessas colunas e irão gerar SQL DDL para definir o index para essas tables. Essa é uma função ajudante que adiciona novos indexes necessários para pesquisas mais rápidas que podem ter sido perdidas no carregamento. Assim como o geocoder é melhorado, essa função será atualizada para acomodar novos indexes que estão sendo usados. Se essa função não gera nada, significa que suas tables possuem o que achamos ser os indexes chave no lugar certo.

Disponibilidade: 2.0.0

## Exemplos

```
SELECT missing_indexes_generate_script();
-- output: This was run on a database that was created before many corrections were made to ←
the loading script ---
CREATE INDEX idx_tiger_county_countyfp ON tiger.county USING btree(countyfp);
CREATE INDEX idx_tiger_cousub_countyfp ON tiger.cousub USING btree(countyfp);
CREATE INDEX idx_tiger_edges_tfidr ON tiger.edges USING btree(tfidr);
CREATE INDEX idx_tiger_edges_tfidl ON tiger.edges USING btree(tfidl);
CREATE INDEX idx_tiger_zip_lookup_all_zip ON tiger.zip_lookup_all USING btree(zip);
CREATE INDEX idx_tiger_data_ma_county_countyfp ON tiger_data.ma_county USING btree(countyfp ←
);
CREATE INDEX idx_tiger_data_ma_cousub_countyfp ON tiger_data.ma_cousub USING btree(countyfp ←
);
CREATE INDEX idx_tiger_data_ma_edges_countyfp ON tiger_data.ma_edges USING btree(countyfp);
CREATE INDEX idx_tiger_data_ma_faces_countyfp ON tiger_data.ma_faces USING btree(countyfp);
```

## Veja Também

[Loader\\_Generate\\_Script](#), [Install\\_Missing\\_Indexes](#)

### 13.1.13 Normalize\_Address

**Normalize\_Address** — Dado um endereço em texto de uma rua, retorna um tipo composto `norm_addy` que não tem um sufixo, prefixo e tipo padronizado, rua, nome de rua etc. quebrado e, campos separados. Essa função irá funcionar com os dados lookup compactados com o `tiger_geocoder` (dados do censo tiger não são necessários).

#### Synopsis

```
norm_addy normalize_address(varchar in_address);
```

#### Descrição

Dado um endereço em texto de uma rua, retorna um tipo composto `norm_addy` que não tem um sufixo, prefixo e tipo padronizado, rua, nome de rua etc. quebrado e, campos separados. Esse é o primeiro passo no processo de geocodificação para tornar todos os endereços normalizados no formato postal. Nenhum outro dado é requerido à parte do que está compactado com o geocoder.

Essa função utiliza as várias lookup tables direção/estado/sufixo pré carregadas com o `tiger_geocoder` e localizadas no esquema `tiger`, então, você não precisa baixar os dados do censo tiger ou qualquer outro tipo de dados para utilizá-la. Talvez você ache necessário adicionar mais abreviações ou nomes alternativos para as lookup tables no esquema `tiger`.

Utiliza várias tables lookup de controle localizadas no esquema `tiger` para normalizar o endereço de entrada.

Campos no tipo de objeto `norm_addy` retornou pela função nessa ordem, onde () indica um campo requerido pelo geocoder, [] indica um campo opcional:

```
(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip] [parsed]
[zip4] [address_alphanumeric]
```

Enhanced: 2.4.0 `norm_addy` object includes additional fields `zip4` and `address_alphanumeric`.

1. `address` é um inteiro: O número da rua
2. `predirAbbrev` is varchar: Prefixo direcional para rua como N, S, L, O etc. Esse são controlados usando a table `direction_lookup`.
3. `streetName` varchar

4. A versão varchar `streetTypeAbbrev` abreviada dos tipos de rua: ex: St., Av., Cir. Elas são controladas usando a table `street_type_lookup`.
5. As direções varchar `postdirAbbrev` abreviadas N, S, L, O etc. Elas são controladas utilizando a table `direction_lookup`.
6. Varchar interno endereço interno como um apartamento ou número de suíte.
7. Varchar localização normalmente uma cidade ou província governante.
8. Os estados varchar `stateAbbrev` dos EUA de dois caracteres. ex: MA, NY, MI. Estes são controlados pela table `state_lookup`.
9. zip varchar 5-digit zipcode. e.g. 02109.
10. `parsed` booleana - indica se um endereço foi formado pelo processo normalizador. A função `normalize_address` coloca isso como verdade antes de retornar o endereço.
11. `zip4` last 4 digits of a 9 digit zip code. Availability: PostGIS 2.4.0.
12. `address_alphanumeric` Full street number even if it has alpha characters like 17R. Parsing of this is better using [Pgcn\\_Normalize\\_Address](#) function. Availability: PostGIS 2.4.0.

## Exemplos

Gerar campos selecionados. Use [Pprint\\_Addy](#) se você quer uma saída textual.

```
SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev
FROM (SELECT address, normalize_address(address) As na
      FROM addresses_to_geocode) As g;
```

orig	streetname	streettypeabbrev
28 Capen Street, Medford, MA	Capen	St
124 Mount Auburn St, Cambridge, Massachusetts 02138	Mount Auburn	St
950 Main Street, Worcester, MA 01610	Main	St
529 Main Street, Boston MA, 02129	Main	St
77 Massachusetts Avenue, Cambridge, MA 02139	Massachusetts	Ave
25 Wizard of Oz, Walford, KS 99912323	Wizard of Oz	

## Veja Também

[Geocode](#), [Pprint\\_Addy](#)

### 13.1.14 Pgcn\_Normalize\_Address

`Pgcn_Normalize_Address` — Dado um endereço em texto de uma rua, retorna um tipo composto `norm_addy` que não tem um sufixo, prefixo e tipo padronizado, rua, nome de rua etc. quebrado e, campos separados. Essa função irá funcionar com os dados lookup compactados com o `tiger_geocoder` (dados do censo tiger não são necessários). Requer a extensão `address_standardizer`.

## Synopsis

```
norm_addy pgcn_normalize_address(varchar in_address);
```

## Descrição

Dado um endereço em texto de uma rua, retorna um tipo composto `norm_addy` que não tem um sufixo, prefixo e tipo padronizado, rua, nome de rua etc. quebrado e, campos separados. Esse é o primeiro passo no processo de geocodificação para tornar todos os endereços normalizados no formato postal. Nenhum outro dado é requerido à parte do que está compactado com o geocoder.

Essa função utiliza as várias lookup tables `pagc_*` pré carregadas com o `tiger_geocoder` e localizadas no esquema `tiger`, então, você não precisa baixar os dados do censo `tiger` ou qualquer outro tipo de dados para utilizá-la. Talvez você ache necessário adicionar mais abreviações ou nomes alternativos para as lookup tables no esquema `tiger`.

Utiliza várias tables lookup de controle localizadas no esquema `tiger` para normalizar o endereço de entrada.

Campos no tipo de objeto `norm_addy` retornou pela função nessa ordem, onde () indica um campo requerido pelo geocoder, [] indica um campo opcional:

Existem pequenas variações no revestimento e formatação do [Normalize\\_Address](#).

Disponibilidade: 2.1.0



This method needs `address_standardizer` extension.

(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip]

O `standardaddr` natural da extensão `address_standardizer` é um pouco mais rico que `norm_addy`, já que foi desenvolvido para suportar endereços internacionais (incluindo países). Os campos equivalentes do `standardaddr` são:

`house_num`, `predir`, `name`, `suftype`, `sufdir`, `unit`, `city`, `state`, `postcode`

Enhanced: 2.4.0 `norm_addy` object includes additional fields `zip4` and `address_alphanumeric`.

1. `address` é um inteiro: O número da rua
2. `predirAbbrev` is varchar: Prefixo direcional para rua como N, S, L, O etc. Esse são controlados usando a table `direction_lookup`.
3. `streetName` varchar
4. A versão varchar `streetTypeAbbrev` abreviada dos tipos de rua: ex: St., Av., Cir. Elas são controladas usando a tables `street_type_lookup`.
5. As direções varchar `postdirAbbrev` abreviadas N, S, L, O etc. Elas são controladas utilizando a table `direction_lookup`.
6. Varchar interno endereço interno como um apartamento ou número de suíte.
7. Varchar localização normalmente uma cidade ou província governante.
8. Os estados varchar `stateAbbrev` dos EUA de dois caracteres. ex: MA, NY, MI. Estes são controlados pela table `state_lookup`.
9. `zip` varchar 5-digit zipcode. e.g. 02109.
10. `parsed` booleana - indica se um endereço foi formado pelo processo normalizador. A função `normalize_address` coloca isso como verdade antes de retornar o endereço.
11. `zip4` last 4 digits of a 9 digit zip code. Availability: PostGIS 2.4.0.
12. `address_alphanumeric` Full street number even if it has alpha characters like 17R. Parsing of this is better using [Pagc\\_Normalize\\_Address](#) function. Availability: PostGIS 2.4.0.

## Exemplos

### Exemplo de chamada única

```
SELECT addy.*
FROM pagc_normalize_address('9000 E ROO ST STE 999, Springfield, CO') AS addy;
```

address	predirabbrev	streetname	streettypeabbrev	postdirabbrev	internal	location	stateabbrev	zip	parsed
9000	E	ROO	ST			SUITE 999	CO		t

Batch call. Existem issues de velocidade com a forma que o `postgis_tiger_geocoder` empacota o `address_standardizer`. Elas serão solucionadas em edições posteriores. Para funcionar em volta delas, se você precisa de velocidade para geocodificação agrupada para gerar um `normaddy` em modo agrupado, você é instigado a usar a função `address_standardizer` `standardize_address` diretamente, como é mostrado abaixo, que é similar ao exercício que fizemos em [Normalize\\_Address](#) que usa os dados criados em [Geocode](#).

```
WITH g AS (SELECT address, ROW((sa).house_num, (sa).predir, (sa).name
, (sa).suftype, (sa).sufdir, (sa).unit, (sa).city, (sa).state, (sa).postcode, true)::<
normaddy As na
FROM (SELECT address, standardize_address('tiger.pagc_lex'
, 'tiger.pagc_gaz'
, 'tiger.pagc_rules', address) As sa
FROM addresses_to_geocode) As g)
SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev
FROM g;
```

orig	streetname	streettypeabbrev
529 Main Street, Boston MA, 02129	MAIN	ST
77 Massachusetts Avenue, Cambridge, MA 02139	MASSACHUSETTS	AVE
25 Wizard of Oz, Walaford, KS 99912323	WIZARD OF	
26 Capen Street, Medford, MA	CAPEN	ST
124 Mount Auburn St, Cambridge, Massachusetts 02138	MOUNT AUBURN	ST
950 Main Street, Worcester, MA 01610	MAIN	ST

## Veja Também

[Normalize\\_Address](#), [Geocode](#)

### 13.1.15 Pprint\_Addy

`Pprint_Addy` — Dado um objeto de tipo composto `normaddy`, retorna uma representação impressa dele. Normalmente, usado em conjunto com o `normalize_address`.

#### Synopsis

```
varchar pprint_addy(normaddy in_addy);
```

#### Descrição

Dado um objeto de tipo composto `normaddy`, retorna uma representação impressa dele. Não é necessário nenhum outro tipo de dados além do que estão compactados com o geocoder.

Usado, normalmente, em conjunto com [Normalize\\_Address](#).

**Exemplos**

**Pretty print a single address**

```
SELECT pprint_addy(normalize_address('202 East Fremont Street, Las Vegas, Nevada 89101')) ←
  As pretty_address;
      pretty_address
-----
202 E Fremont St, Las Vegas, NV 89101
```

**Pretty print address a table of addresses**

```
SELECT address As orig, pprint_addy(normalize_address(address)) As pretty_address
  FROM addresses_to_geocode;
```

orig	pretty_address
529 Main Street, Boston MA, 02129	529 Main St, Boston MA, 02129
77 Massachusetts Avenue, Cambridge, MA 02139	77 Massachusetts Ave, Cambridge, MA ←
28 Capen Street, Medford, MA	28 Capen St, Medford, MA
124 Mount Auburn St, Cambridge, Massachusetts 02138	124 Mount Auburn St, Cambridge, MA ←
950 Main Street, Worcester, MA 01610	950 Main St, Worcester, MA 01610

**Veja Também**

[Normalize\\_Address](#)

**13.1.16 Reverse\_Geocode**

Reverse\_Geocode — Pega um ponto em um sistema de referência espacial conhecido e retorna um relato que contém um banco de dados de, teoricamente, possíveis endereços e um banco de dados de ruas cruzadas. Se `include_strnum_range = verdade`, inclui o alcance da rua nas ruas cruzadas.

**Synopsis**

record **Reverse\_Geocode**(geometry pt, boolean include\_strnum\_range=false, geometry[] OUT intpt, norm\_addy[] OUT addy, varchar[] OUT street);

**Descrição**

Pega um ponto em um sistema de referência espacial conhecido e retorna um relato que contém um banco de dados de, teoricamente, possíveis endereços e um banco de dados de ruas cruzadas. Se `include_strnum_range = verdade`, inclui o alcance da rua nas ruas cruzadas. `include_strnum_range` se torna falso se não passar. Os endereços são separados de acordo com qual rua um ponto é mais próximo, então, o primeiro endereço é o mais certo.

Porque dizemos endereços hipotéticos em vez de reais. Os dados Tiger não possuem endereços reais, somente variedades de ruas. O suposto endereço é interpolado baseado na variedade de ruas, por exemplo. Bem como interpolar um dos meus endereços de retorno em 26 Court St. and 26 Court Sq., sendo que não existem tais endereços. Isso se dá porque um ponto pode estar na esquina de 2 ruas e assim a lógica interpola as duas ruas. A lógica também presume que os endereços são espaçados igualmente ao longo de uma rua, o que está errado já que você pode ter um edifício municipal ocupando boa parte de uma rua, enquanto o resto das construções estão todas agrupadas no fim dela.

Nota: Esta função confia nos dados Tiger. Se você não carregou dados cobrindo a região deste ponto, então, você terá que ter um relato cheio de NULOS.

Elementos que retornaram do relato são como segue:





```

-----+-----+-----+-----
529 Main Street, Boston MA, 02129 | -71.07181 | 42.38359 | 527 Main St, ←
  Boston, MA 02129 | Medford St |
77 Massachusetts Avenue, Cambridge, MA 02139 | -71.09428 | 42.35988 | 77 ←
  Massachusetts Ave, Cambridge, MA 02139 | Vassar St |
26 Capen Street, Medford, MA | -71.12377 | 42.41101 | 9 Edison Ave, ←
  Medford, MA 02155 | Capen St | Tesla Ave
124 Mount Auburn St, Cambridge, Massachusetts 02138 | -71.12304 | 42.37328 | 3 University ←
  Rd, Cambridge, MA 02138 | Mount Auburn St |
950 Main Street, Worcester, MA 01610 | -71.82368 | 42.24956 | 3 Maywood St, ←
  Worcester, MA 01603 | Main St | Maywood Pl

```

## Veja Também

[Pprint\\_Addy](#), [Pprint\\_Addy](#), [ST\\_AsText](#)

### 13.1.17 Topology\_Load\_Tiger

`Topology_Load_Tiger` — Carrega uma região definida de dados tiger em uma Topologia PostGIS e transforma os dados tiger para referência espacial da topologia e rompe para a tolerância precisa da topologia.

#### Synopsis

```
text Topology_Load_Tiger(varchar topo_name, varchar region_type, varchar region_id);
```

#### Descrição

Carrega uma região definida de dados tiger em uma Topologia PostGIS. As faces, nós e limites são transformados em um sistema de referência espacial de uma topologia alvo e pontos são estalados à tolerância da topologia alvo. As faces, nós e limites criados, mantêm as mesmas identidades dos originais dos dados Tiger, para os datasets serem reconciliados mais facilmente no futuro com os dados tiger. Retorna detalhes resumidos do processo.

Seria útil, por exemplo, para dividir em novos distritos os dados onde você requer que os polígonos formados sigam as linhas centrais das ruas e para os polígonos resultantes não se sobreporem.



#### Note

Esta função confia nos dados tiger, bem como o módulo de instalação da topologia do PostGIS. para maiores informações, veja: [Chapter 11](#) e [Section 2.4.1](#). Se você não carregou os dados cobrindo a região de interesse, nenhum relato de topologia será criado, Esta função também falhará se você não tiver criado uma topologia usando as funções dela.



#### Note

A maior parte dos erros de validação de topologia são resultados de issues de tolerância, nas quais depois da transformação os pontos limites não se alinham ou sobrepõem. Para remediar esta situação, você talvez queira aumentar ou diminuir a precisão, se você tiver falhas na validação da topologia.

Argumentos obrigatórios:

1. `topo_name` O nome de uma topologia PostGIS para carregar dados.

2. `region_type` O tipo de região limitadora. Atualmente, somente `place` e `county` são suportados. O plano é ter muitas mais. Esta é a table para investigar para definir os limites da região. ex: `tiger.place`, `tiger.county`
3. `region_id` Isto é o que o Tiger chama de geoid. É o único identificador da região na table. Para lugar é a `plcidfp` coluna em `tiger.place`. Para condado é a `cntyidfp` coluna `tiger.county`

Disponibilidade: 2.0.0

### Exemplo: Topologia de Boston, Massachusetts

Criar uma topologia para Boston, Massachusetts em Mass State Plane Feet (2249), com tolerância 0.25 feet e então carregar na cidade de Boston faces, limites e nós tiger.

```
SELECT topology.CreateTopology('topo_boston', 2249, 0.25);
createtopology
-----
    15
-- 60,902 ms ~ 1 minute on windows 7 desktop running 9.1 (with 5 states tiger data loaded)
SELECT tiger.topology_load_tiger('topo_boston', 'place', '2507000');
-- topology_loader_tiger --
29722 edges holding in temporary. 11108 faces added. 1875 edges of faces added. 20576 ←
  nodes added.
19962 nodes contained in a face. 0 edge start end corrected. 31597 edges added.

-- 41 ms --
SELECT topology.TopologySummary('topo_boston');
-- topologysummary--
Topology topo_boston (15), SRID 2249, precision 0.25
20576 nodes, 31597 edges, 11109 faces, 0 topogeoms in 0 layers

-- 28,797 ms to validate yeh returned no errors --
SELECT * FROM
  topology.ValidateTopology('topo_boston');

  error      | id1  | id2
-----+-----+-----
```

### Exemplo: Suffolk, topologia de Massachusetts

Criar uma topologia para Suffolk, Massachusetts in Mass State Plane Meters (26986), com tolerância 0.25 metros e então carregar no condado de Suffolk faces, limites e nós tiger.

```
SELECT topology.CreateTopology('topo_suffolk', 26986, 0.25);
-- this took 56,275 ms ~ 1 minute on Windows 7 32-bit with 5 states of tiger loaded
-- must have been warmed up after loading boston
SELECT tiger.topology_load_tiger('topo_suffolk', 'county', '25025');
-- topology_loader_tiger --
36003 edges holding in temporary. 13518 faces added. 2172 edges of faces added.
24761 nodes added. 24075 nodes contained in a face. 0 edge start end corrected. 38175 ←
  edges added.
-- 31 ms --
SELECT topology.TopologySummary('topo_suffolk');
-- topologysummary--
Topology topo_suffolk (14), SRID 26986, precision 0.25
24761 nodes, 38175 edges, 13519 faces, 0 topogeoms in 0 layers

-- 33,606 ms to validate --
SELECT * FROM
  topology.ValidateTopology('topo_suffolk');
```

error	id1	id2
coincident nodes	81045651	81064553
edge crosses node	81045651	85737793
edge crosses node	81045651	85742215
edge crosses node	81045651	620628939
edge crosses node	81064553	85697815
edge crosses node	81064553	85728168
edge crosses node	81064553	85733413

## Veja Também

[Cria topologia](#), [CreateTopoGeom](#), [TopologySummary](#), [ValidateTopology](#)

### 13.1.18 Set\_Geocode\_Setting

`Set_Geocode_Setting` — Estabelece uma configuração que afeta comportamento das funções geocoder.

#### Synopsis

```
text Set_Geocode_Setting(text setting_name, text setting_value);
```

#### Descrição

Estabelece valor de uma configuração específica armazenada na table `tiger.geocode_settings`. Configurações permitem que você comute depuração de funções. Os planos futuros serão para controlar avaliação com configurações. A lista atual de configurações está em:

Disponibilidade: 2.1.0

#### Exemplo da configuração de retornar depuração

Se você executar `Geocode` quando esta função for verdade, o log NOTICE irá gerar horas e pesquisas.

```
SELECT set_geocode_setting('debug_geocode_address', 'true') As result;
result
-----
true
```

## Veja Também

[Get\\_Geocode\\_Setting](#)

## Chapter 14

# PostGIS Special Functions Index

### 14.1 PostGIS Aggregate Functions

The functions given below are spatial aggregate functions provided with PostGIS that can be used just like any other sql aggregate function such as sum, average.

- **ST\_3DExtent** - uma função agregada que retorna a caixa3D que estabelece linhas de geometria.
- **ST\_Accum** - Agregado. Constrói um banco de dados de geometrias.
- **ST\_AsGeobuf** - Return a Geobuf representation of a set of rows.
- **ST\_ClusterIntersecting** - Agregado. Retorna um arranjo com os componentes conectados de um conjunto de geometrias
- **ST\_ClusterWithin** - Agregado. Retorna um arranjo de coleções de geometrias, onde cada coleção representa um conjunto de geometrias separados por nada mais que a distância especificada.
- **ST\_Collect** - Retorna um valor ST\_Geometry específico de uma coleção de outras geometrias.
- **ST\_Extent** - uma função agregada que retorna a caixa limitadora que estabelece linhas de geometrias.
- **ST\_MakeLine** - Cria uma Linestring de ponto, multiponto ou linha das geometrias.
- **ST\_MemUnion** - O mesmo que ST\_Union, somente memória amigável (utiliza menos memória e mais tempo de processador).
- **ST\_Polygonize** - Agregado. Cria uma GeometryCollection contendo possíveis polígonos formados a partir da linework constituinte de um conjunto de geometrias.
- **ST\_SameAlignment** - Retorna verdade se os rasters têm a mesma inclinação, escala, referência espacial, e deslocamento (pixels podem ser colocados na mesma grade sem cortar eles) e falso se eles não notificarem problemas detalhados.
- **ST\_Union** - Retorna uma geometria que representa a união de pontos das Geometrias.
- **TopoElementArray\_Agg** - Retorna um topelementarray para um conjunto de arranjos element\_id, type (topoelementos)

### 14.2 PostGIS Window Functions

The functions given below are spatial window functions provided with PostGIS that can be used just like any other sql window function such as row\_number(), lead(), lag(). All these require an SQL OVER() clause.

- **ST\_ClusterDBSCAN** - Função do Windows que retorna id inteira para o grupo que cada geometria de entrada está, baseada na implementação 2D de agrupamento de densidade espacial de aplicações com algoritmo barulhento (DBSCAN).
  - **ST\_ClusterKMeans** - Função do Windows que retorna id inteira para o grupo que cada geometria de entrada está.
-

## 14.3 PostGIS SQL-MM Compliant Functions

The functions given below are PostGIS functions that conform to the SQL/MM 3 standard



### Note

SQL-MM defines the default SRID of all geometry constructors as 0. PostGIS uses a default SRID of -1.

- **ST\_3DDWithin** - Para tipo de geometria 3d (z) verdadeiro se a distância entre duas geometrias 3d está dentro de números de unidades. This method implements the SQL/MM specification. SQL-MM ?
- **ST\_3DDistance** - Para tipo geometria, retorna a menor distância cartesiana 3-dimensional (baseado no sistema de referência espacial) entre duas geometrias em unidades projetadas. This method implements the SQL/MM specification. SQL-MM ?
- **ST\_3DIntersects** - Retorna VERDADE se as geometrias "intersectadas espacialmente" em 3d - somente para pontos, linestrings, polígonos, superfícies poliédricas (área). Com o backend SFCGAL ativado, também suporta TINS This method implements the SQL/MM specification. SQL-MM 3: ?
- **ST\_AddEdgeModFace** - Adiciona um novo limite e, se uma face for dividida, modifica a face original e adiciona uma nova face. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.13
- **ST\_AddEdgeNewFaces** - Adiciona um novo limite e, se uma face for dividida, deleta a face original e substitui por duas novas faces. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.12
- **ST\_AddIsoEdge** - Adiciona um limite isolado definido pela geometria alinestring a uma topologia conectando dois nós isolados e retorna a nova id do novo limite. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.4
- **ST\_AddIsoNode** - Adiciona um nó isolado a uma face em uma topologia e retorna a id do novo nó. Se a face é nula, o nó continua sendo criado. This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X+1.3.1
- **ST\_Area** - Retorna a área da superfície se ela for um polígono ou multipolígono. Para geometria, uma área cartesiana 2D é determinada com unidades especificadas pelo SRID. Para geografia, a área é determinada em uma superfície com unidades em metros quadrados. This method implements the SQL/MM specification. SQL-MM 3: 8.1.2, 9.5.3
- **ST\_AsBinary** - Retorna a representação do binário bem conhecido (WKB) da geometria/geografia sem os meta dados SRID. This method implements the SQL/MM specification. SQL-MM 3: 5.1.37
- **ST\_AsText** - Retorna a representação de texto bem conhecida (WKT) da geometria/geografia sem os meta dados do SRID. This method implements the SQL/MM specification. SQL-MM 3: 5.1.25
- **ST\_Boundary** - Retorna o encerramento da borda combinatória dessa geometria. This method implements the SQL/MM specification. SQL-MM 3: 5.1.14
- **ST\_Buffer** - (T) Retorna uma geometria cobrindo todos os pontos com a dada distância da geometria de entrada. This method implements the SQL/MM specification. SQL-MM 3: 5.1.17
- **ST\_Centroid** - Retorna o centro geométrico de uma geometria. This method implements the SQL/MM specification. SQL-MM 3: 8.1.4, 9.5.5
- **ST\_ChangeEdgeGeom** - Modifica a forma de um limite sem afetar a estrutura da topologia. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details X.3.6
- **ST\_Contains** - Retorna verdade se nenhum ponto de B estiverem no exterior de A, e pelo menos um ponto do interior de B estiver no interior de A. This method implements the SQL/MM specification. SQL-MM 3: 5.1.31
- **ST\_ConvexHull** - O casco convexo de uma geometria representa a mínima geometria convexa que envolve todas as geometrias dentro do conjunto. This method implements the SQL/MM specification. SQL-MM 3: 5.1.16

- **ST\_CoordDim** - Retorna a dimensão da coordenada do valor ST\_Geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.3
  - **ST\_CreateTopoGeo** - Adiciona uma coleção de geometrias para uma dada topologia vazia e retorna uma mensagem detalhando sucesso. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details -- X.3.18
  - **ST\_Crosses** - Retorna TRUE se as geometrias fornecidas têm alguns, não todos, pontos em comum. This method implements the SQL/MM specification. SQL-MM 3: 5.1.29
  - **ST\_CurveToLine** - Converte uma CIRCULARSTRING/CURVEPOLYGON para uma LINestring/POLYGON This method implements the SQL/MM specification. SQL-MM 3: 7.1.7
  - **ST\_Difference** - Retorna uma geometria que representa aquela parte de geometria A que não intersecta com a geometria B. This method implements the SQL/MM specification. SQL-MM 3: 5.1.20
  - **ST\_Dimension** - A dimensão herdada desse objeto geométrico, o qual deve ser menor ou igual à dimensão coordenada. This method implements the SQL/MM specification. SQL-MM 3: 5.1.2
  - **ST\_Disjoint** - Retorna VERDADE se as geometrias não se "intersectam espacialmente" - se elas não dividem nenhum espaço. This method implements the SQL/MM specification. SQL-MM 3: 5.1.26
  - **ST\_Distance** - For geometry type returns the 2D Cartesian distance between two geometries in projected units (based on spatial reference system). For geography type defaults to return minimum geodesic distance between two geographies in meters. This method implements the SQL/MM specification. SQL-MM 3: 5.1.23
  - **ST\_EndPoint** - Retorna ao último ponto de uma LINestring ou CIRCULARLINestring geometria como um PONTO. This method implements the SQL/MM specification. SQL-MM 3: 7.1.4
  - **ST\_Envelope** - Retorna uma geometria representando a precisão da dobrada (float8) da caixa limitada da geometria fornecida. This method implements the SQL/MM specification. SQL-MM 3: 5.1.15
  - **ST\_Equals** - Retorna verdade se as geometrias representam a mesma geometria. A direcionalidade é ignorada. This method implements the SQL/MM specification. SQL-MM 3: 5.1.24
  - **ST\_ExteriorRing** - Retorna uma line string representando o anel exterior da geometria POLÍGONO. Retorna NULA se a geometria não for um polígono. Não funcionará com MULTIPOLÍGONO. This method implements the SQL/MM specification. SQL-MM 3: 8.2.3, 8.3.3
  - **ST\_GMLToSQL** - Retorna um valor ST\_Geometry específico da representação GML. Esse é um heterônimo para ST\_GeomFromGML This method implements the SQL/MM specification. SQL-MM 3: 5.1.50 (exceto para curvas suporte).
  - **ST\_GeomCollFromText** - Makes a collection Geometry from collection WKT with the given SRID. If SRID is not given, it defaults to 0. This method implements the SQL/MM specification.
  - **ST\_GeomFromText** - Retorna um valor ST\_Geometry específico da representação de texto bem conhecida (WKT). This method implements the SQL/MM specification. SQL-MM 3: 5.1.40
  - **ST\_GeomFromWKB** - Criar uma geometria exemplo de um representação bem conhecida de geometria binária (WKB) e SRID opcional. This method implements the SQL/MM specification. SQL-MM 3: 5.1.41
  - **ST\_GeometryFromText** - Retorna um valor ST\_Geometry específico da representação de texto estendida bem conhecida (EWKT). Isso é um heterônimo para ST\_GeomFromText This method implements the SQL/MM specification. SQL-MM 3: 5.1.40
  - **ST\_GeometryN** - Retorna a geometria de 1-base Nth se a geometria é uma GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINestring, MULTICURVE ou (MULTI)POLYGON, POLYHEDRALSURFACE. Senão, retorna NULA. This method implements the SQL/MM specification. SQL-MM 3: 9.1.5
  - **ST\_GeometryType** - Retorna o tipo de geometria de valor ST\_Geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.4
  - **ST\_GetFaceEdges** - Retorna um conjunto de limites ordenados que amarram aface. This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.5
-

- **ST\_GetFaceGeometry** - Retorna o polígono na topologia dada com a id de face especificada. This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.16
  - **ST\_InitTopoGeo** - Cria um novo esquema topologia e registra esse novo esquema na table topology.topology e detalha um resumo do processo. This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.17
  - **ST\_InteriorRingN** - Retorna o anel linestring Nth interior do polígono. Retorna NULO se a geometria não for um polígono ou o dado N está fora da extensão. This method implements the SQL/MM specification. SQL-MM 3: 8.2.6, 8.3.5
  - **ST\_Intersection** - (T) Retorna uma geometria que representa a porção dividida da geomA e geomB. This method implements the SQL/MM specification. SQL-MM 3: 5.1.18
  - **ST\_Intersects** - Retorna VERDADE se as geometrias/geografia 'intersectam espacialmente em 2D" - (dividem qualquer porção de espaço) e FALSO se elas não (estão disjuntas). Para geografia -- a tolerância é de 0.00001 metros (então quaisquer pontos que estão mais perto estão intersectando) This method implements the SQL/MM specification. SQL-MM 3: 5.1.27
  - **ST\_IsClosed** - Retorna VERDADEIRO se os pontos de começo e fim da LINESTRING são coincidentes. Para superfície poliédrica está fechada (volumétrica). This method implements the SQL/MM specification. SQL-MM 3: 7.1.5, 9.3.3
  - **ST\_IsEmpty** - Retorna verdadeiro se essa geometria é uma coleção vazia, polígono, ponto etc. This method implements the SQL/MM specification. SQL-MM 3: 5.1.7
  - **ST\_IsRing** - Retorna VERDADEIRO se essa LINESTRING for fechada e simples. This method implements the SQL/MM specification. SQL-MM 3: 7.1.6
  - **ST\_IsSimple** - Retorna (VERDADEIRA) se essa geometria não tem nenhum ponto irregular, como auto intersecção ou tangenciação. This method implements the SQL/MM specification. SQL-MM 3: 5.1.8
  - **ST\_IsValid** - Retorna verdadeira se a ST\_Geometry é bem formada. This method implements the SQL/MM specification. SQL-MM 3: 5.1.9
  - **ST\_Length** - Retorna o comprimento 2D da geometria se ela é uma LineString ou MultiLineString. A geometria está em unidades da referência espacial e geografia em metros (padrão esferoide) This method implements the SQL/MM specification. SQL-MM 3: 7.1.2, 9.3.4
  - **ST\_LineFromText** - Faz uma geometria de uma representação WKT com a SRID dada. Se a SRID não for dada, isso leva a 0. This method implements the SQL/MM specification. SQL-MM 3: 7.2.8
  - **ST\_LineFromWKB** - Faz uma LINESTRING de uma WKB com o SRID dado This method implements the SQL/MM specification. SQL-MM 3: 7.2.9
  - **ST\_LinestringFromWKB** - Faz uma geometria de uma WKB com o SRID dado. This method implements the SQL/MM specification. SQL-MM 3: 7.2.9
  - **ST\_M** - Retorna a coordenada M do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto. This method implements the SQL/MM specification.
  - **ST\_MLineFromText** - Retorna um valor específico ST\_MultiLineString de uma representação WKT. This method implements the SQL/MM specification. SQL-MM 3: 9.4.4
  - **ST\_MPointFromText** - Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0. This method implements the SQL/MM specification. SQL-MM 3: 9.2.4
  - **ST\_MPolyFromText** - Makes a MultiPolygon Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0. This method implements the SQL/MM specification. SQL-MM 3: 9.6.4
  - **ST\_ModEdgeHeal** - Fecha dois limites deletando o nó que os conecta, modificando o primeiro limite e deletando o segundo. Retorna a id do nó deletado. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9
  - **ST\_ModEdgeSplit** - Divide um limite criando um novo nó junto de um limite existente, modificando o limite original e adicionando um novo limite. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9
-



- **ST\_MoveIsoNode** - Moves an isolated node in a topology from one point to another. If new apoint geometry exists as a node an error is thrown. Returns description of move. This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X.3.2
  - **ST\_NewEdgeHeal** - Fecha dois limites deletando o nó que os conecta, deletando ambos limites, e substituindo-os com um limite cuja direção é a mesma do primeiro limite fornecido. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9
  - **ST\_NewEdgesSplit** - Divide um limite criando um novo nó ao longo do limite existente, deletando o limite original e substituindo-o por dois novos. Retorna a id do novo nó criado que integra os novos limites. This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X.3.8
  - **ST\_NumGeometries** - Se a geometria é uma GEOMETRYCOLLECTION (ou MULTI\*), retorna o número de geometria, para geometrias únicas retornará 1, senão retorna NULO. This method implements the SQL/MM specification. SQL-MM 3: 9.1.4
  - **ST\_NumInteriorRings** - Retorna o número de anéis interiores de um polígono. This method implements the SQL/MM specification. SQL-MM 3: 8.2.5
  - **ST\_NumPatches** - Retorna o número de faces em uma superfícies poliédrica. Retornará nulo para geometrias não poliédricas. This method implements the SQL/MM specification. SQL-MM 3: ?
  - **ST\_NumPoints** - Retorna o número de pontos em um valor ST\_LineString ou ST\_CircularString. This method implements the SQL/MM specification. SQL-MM 3: 7.2.4
  - **ST\_OrderingEquals** - Retorna verdade se as geometrias dadas representam a mesma geometrias e os pontos estão na mesma ordem direcional. This method implements the SQL/MM specification. SQL-MM 3: 5.1.43
  - **ST\_Overlaps** - Retorna VERDADE se as geometrias dividem espaço, são da mesma dimensão, mas não estão completamente contidas uma pela outra. This method implements the SQL/MM specification. SQL-MM 3: 5.1.32
  - **ST\_PatchN** - Retorna a geometria (face) de 1-base Nth se a geometria é uma POLYHEDRALSURFACE, POLYHEDRAL-SURFACEM. Senão, retorna NULA. This method implements the SQL/MM specification. SQL-MM 3: ?
  - **ST\_Perimeter** - Retorna o comprimento do limite de uma geometria ou geografia ST\_Surface ou ST\_MultiSurface. (Polígono, Multipolígono). A medição das unidades de geometria está na referência espacial e a da geografia em metros. This method implements the SQL/MM specification. SQL-MM 3: 8.1.3, 9.5.4
  - **ST\_Point** - Retorna uma ST\_Point com os valores de coordenada dados. Heterônimo OGC para ST\_MakePoint. This method implements the SQL/MM specification. SQL-MM 3: 6.1.2
  - **ST\_PointFromText** - Faz um ponto de um WKT com o SRID dado. Se o SRID não for dado, isso leva a desconhecido. This method implements the SQL/MM specification. SQL-MM 3: 6.1.8
  - **ST\_PointFromWKB** - Faz uma geometria a partir de um WKB com o SRID dado This method implements the SQL/MM specification. SQL-MM 3: 6.1.9
  - **ST\_PointN** - Retorna o ponto Nth na primeira linestring ou linestring circular na geometria. Valores negativos são contados tardiamente do fim da linestring. Retorna NULA se não há uma linestring na geometria. This method implements the SQL/MM specification. SQL-MM 3: 7.2.5, 7.3.5
  - **ST\_PointOnSurface** - Retorna um POINT garantido a ficar na superfície. This method implements the SQL/MM specification. SQL-MM 3: 8.1.5, 9.5.6. De acordo com as specs, a ST\_PointOnSurface funciona para superfícies (POLÍGONOS, MULTIPOLÍGONOS, POÍGONOS CURVOS). Então o PostGIS parece estar estendendo o que a spec permite aqui. A maioria dos banco de dados Oracle, DB II, ESRI SDE parecem suportar esta função para superfícies. Assim como o PostGIS o SQL Server 2008 suporta todas as geometrias comuns.
  - **ST\_Polygon** - Retorna um polígono construído de linestring e SRID especificados. This method implements the SQL/MM specification. SQL-MM 3: 8.3.2
  - **ST\_PolygonFromText** - Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0. This method implements the SQL/MM specification. SQL-MM 3: 8.3.6
-



- **ST\_Relate** - Retorna verdade se esta geometria estiver relacionada a outra geometria, testando interseções entre o interior, limite e exterior das duas geometrias como especificado pelos valores na intersectionMatrixPattern. Se nenhuma intersectionMatrixPattern passa, retorna a intersectionMatrixPattern máxima que relaciona as 2 geometrias. This method implements the SQL/MM specification. SQL-MM 3: 5.1.25
  - **ST\_RemEdgeModFace** - Remove um limite e, se o limite removido separou duas faces, deleta uma das duas e modifica a outra para pegar o espaço delas. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.15
  - **ST\_RemEdgeNewFace** - Remove um limite e, se o limite removido separava duas faces, deleta as faces originais e as substitui por uma nova face. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.14
  - **ST\_RemoveIsoEdge** - Removes an isolated edge and returns description of action. If the edge is not isolated, then an exception is thrown. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3
  - **ST\_RemoveIsoNode** - Remove um nó isolado e retorna descrição de ação. Se o nó não for isolado (for começo ou fim de um limite), então, uma exceção é lançada. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3
  - **ST\_SRID** - Retorna o identificador de referência espacial para a ST\_Geometry como definido na table spatial\_ref\_sys. This method implements the SQL/MM specification. SQL-MM 3: 5.1.5
  - **ST\_StartPoint** - Retorna ao último ponto de uma LINESTRING geometria como um PONTO. This method implements the SQL/MM specification. SQL-MM 3: 7.1.3
  - **ST\_SymDifference** - Retorna uma geometria que representa as porções de A e B que não intersectam. É chamado de diferença simétrica, porque  $ST\_SymDifference(A,B) = ST\_SymDifference(B,A)$ . This method implements the SQL/MM specification. SQL-MM 3: 5.1.21
  - **ST\_Touches** - Retorna TRUE se as geometrias têm pelo menos um ponto em comum, mas seus interiores não se intersectam. This method implements the SQL/MM specification. SQL-MM 3: 5.1.28
  - **ST\_Transform** - Retorna uma nova geometria com suas coordenadas transformadas em diferentes referências espaciais. This method implements the SQL/MM specification. SQL-MM 3: 5.1.6
  - **ST\_Union** - Retorna uma geometria que representa a união de pontos das Geometrias. This method implements the SQL/MM specification. SQL-MM 3: 5.1.19 o índice-z (elevação) quando polígonos não estão envolvidos.
  - **ST\_WKBToSQL** - Retorna um valor ST\_Geometry específico da representação de texto binário bem conhecida (WKB). Isso é um heterônimo para ST\_GeomFromWKB que não pega nenhum srid This method implements the SQL/MM specification. SQL-MM 3: 5.1.36
  - **ST\_WKTTToSQL** - Retorna um valor ST\_Geometry específico da representação de texto estendida bem conhecida (EWKT). Isso é um heterônimo para ST\_GeomFromText This method implements the SQL/MM specification. SQL-MM 3: 5.1.34
  - **ST\_Within** - Retorna verdade se a geometria A estiver completamente dentro da geometria B This method implements the SQL/MM specification. SQL-MM 3: 5.1.30
  - **ST\_X** - Retorna a coordenada X do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto. This method implements the SQL/MM specification. SQL-MM 3: 6.1.3
  - **ST\_Y** - Retorna a coordenada Y do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto. This method implements the SQL/MM specification. SQL-MM 3: 6.1.4
  - **ST\_Z** - Retorna a coordenada Z do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto. This method implements the SQL/MM specification.
-

## 14.4 PostGIS Geography Support Functions

The functions and operators given below are PostGIS functions/operators that take as input or return as output a **geography** data type object.



### Note

Functions with a (T) are not native geodetic functions, and use a ST\_Transform call to and from geometry to do the operation. As a result, they may not behave as expected when going over dateline, poles, and for large geometries or geometry pairs that cover more than one UTM zone. Basic transform - (favoring UTM, Lambert Azimuthal (North/South), and falling back on mercator in worst case scenario)

- **ST\_Area** - Retorna a área da superfície se ela for um polígono ou multipolígono. Para geometria, uma área cartesiana 2D é determinada com unidades especificadas pelo SRID. Para geografia, a área é determinada em uma superfície com unidades em metros quadrados.
- **ST\_AsBinary** - Retorna a representação do binário bem conhecido (WKB) da geometria/geografia sem os meta dados SRID.
- **ST\_AsEWKT** - Retorna a representação de texto bem conhecida (WKT) da geometria com os meta dados SRID.
- **ST\_AsGML** - Retorna a geometria como uma versão GML com 2 ou 3 elementos.
- **ST\_AsGeoJSON** - Retorna a geometria com um elemento GeoJSON.
- **ST\_AsKML** - Retorna a geometria como um elemento KML. Muitas variantes. Versão padrão=2, precisão padrão=15
- **ST\_AsMVT** - Return a Mapbox Vector Tile representation of a set of rows.
- **ST\_AsMVTGeom** - Transform a geometry into the coordinate space of a Mapbox Vector Tile.
- **ST\_AsSVG** - Retorna uma geometria em dados SVG path, dado um objeto de geometria ou geografia.
- **ST\_AsText** - Retorna a representação de texto bem conhecida (WKT) da geometria/geografia sem os meta dados do SRID.
- **ST\_Buffer** - (T) Retorna uma geometria cobrindo todos os pontos com a dada distância da geometria de entrada.
- **ST\_Centroid** - Retorna o centro geométrico de uma geometria.
- **ST\_CoveredBy** - Retorna 1 (VERDADE) se nenhum ponto na geometria/geografia A estiver fora da geometria/geografia B
- **ST\_Covers** - Retorna 1 (VERDADE) se nenhum ponto na geometria B estiver fora da geometria A
- **ST\_DWithin** - Retorna verdade se as geometrias estiverem dentro da distância especificada de outra. Para geometria, as unidades estão na referência espacial e para geografia, elas estão em metros e a medição é use\_spheroid=true (medida em volta do esferoide), para uma verificação mais rápida, use\_spheroid=false para medir ao longo da esfera.
- **ST\_Distance** - For geometry type returns the 2D Cartesian distance between two geometries in projected units (based on spatial reference system). For geography type defaults to return minimum geodesic distance between two geographies in meters.
- **=** - Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B.
- **ST\_Intersection** - (T) Retorna uma geometria que representa a porção dividida da geomA e geomB.
- **ST\_Intersects** - Retorna VERDADE se as geometrias/geografia 'intersectam espacialmente em 2D" - (dividem qualquer porção de espaço) e FALSO se elas não (estão disjuntas). Para geografia -- a tolerância é de 0.00001 metros (então quaisquer pontos que estão mais perto estão intersectando)
- **ST\_Length** - Retorna o comprimento 2D da geometria se ela é uma LineString ou MultiLineString. A geometria está em unidades da referência espacial e geografia em metros (padrão esferoide)
- **ST\_Perimeter** - Retorna o comprimento do limite de uma geometria ou geografia ST\_Surface ou ST\_MultiSurface. (Polígono, Multipolígono). A medição das unidades de geometria está na referência espacial e a da geografia em metros.

- **ST\_Project** - Retorna um POINT projetado de um ponto inicial usando uma distância em metros e suportando (azimute) em radianos.
- **ST\_Segmentize** - Retorna uma geometria/geografia alterada não tendo nenhum segmento maior que a distância dada.
- **<->** - Retorna a distância 2D entre A e B.
- **&&** - Retorna VERDADE se a caixa limitadora 2D de A intersecta a caixa limitadora 2D de B.

## 14.5 PostGIS Raster Support Functions

The functions and operators given below are PostGIS functions/operators that take as input or return as output a **raster** data type object. Listed in alphabetical order.

- **Caixa3D** - Retorna a representação da caixa 3d da caixa encerrada do raster.
- **@** - Retorna VERDADE se a caixa limitadora de A estiver contida pela de B. Utiliza precisão dupla de caixa limitadora.
- **~** - Retorna TRUE se a caixa delimitadora de A estiver contida na do B. Utiliza caixa delimitadora de precisão dupla.
- **=** - Retorna VERDADE se a caixa limitadora de A for a mesma de B. Utiliza precisão dupla de caixa limitadora.
- **&&** - Retorna VERDADE se a caixa limitadora de A intersecta a caixa limitadora de B.
- **&<** - Retorna VERDADE se uma caixa limitadora de A está à esquerda da de B.
- **&>** - Retorna VERDADE se uma caixa limitadora de A está à direita da de B.
- **~=** - Retorna VERDADE se a caixa limitadora de A é a mesma de B.
- **ST\_Retile** - Retorna um conjunto de tiles configuradas de uma cobertura raster aleatória.
- **ST\_AddBand** - Retorna um raster com nova banda(s) do tipo dado adicionado com o valor inicial com a localização do índice. Se nenhum índice for especificado, a banda é adicionada ao final.
- **ST\_AsBinary/ST\_AsWKB** - Return the Well-Known Binary (WKB) representation of the raster.
- **ST\_AsGDALRaster** - Return the raster tile in the designated GDAL Raster format. Raster formats are one of those supported by your compiled library. Use ST\_GDALDrivers() to get a list of formats supported by your library.
- **ST\_AsHexWKB** - Return the Well-Known Binary (WKB) in Hex representation of the raster.
- **ST\_AsJPEG** - Retorna as bandas tile raster selecionadas como uma única Joint Photographic Exports Group (JPEG) image (byte arranjo). Se nenhuma banda for especificada e 1 ou mais que 3 bandas, então somente a primeira banda é usada. Se somente 3 bandas, então todas as 3 bandas serão usadas para mapear par RGB.
- **ST\_AsPNG** - Retorna as bandas tile raster selecionadas como um gráfico de rede portátil (PNG) imagem (byte array). Se as bandas raster 1, 3 ou 4 e nenhuma banda for especificado, então todas as bandas são usadas. Se mais 2 ou mais que 4 bandas e nenhuma banda forem especificadas, então somente a banda 1 é usada. As bandas são mapeadas para espeço RGB ou RGBA.
- **ST\_AsRaster** - Converte uma geometria PostGIS para um raster PostGIS.
- **ST\_AsTIFF** - Return the raster selected bands as a single TIFF image (byte array). If no band is specified or any of specified bands does not exist in the raster, then will try to use all bands.
- **ST\_Aspect** - Retorna o aspecto (em graus) de uma banda raster de elevação. Útil para analisar terrenos.
- **ST\_Band** - Retorna uma ou mais bandas de um raster existente como um novo raster. Útil para a construção de novos rasters a partir de rasters existentes.
- **ST\_BandFileSize** - Returns the file size of a band stored in file system. If no bandnum specified, 1 is assumed.
- **ST\_BandFileTimestamp** - Returns the file timestamp of a band stored in file system. If no bandnum specified, 1 is assumed.

- **ST\_BandIsNoData** - Retorna verdadeiro se a banda estiver repleta somente de valores nodata.
  - **ST\_BandMetaData** - Retorna os metadados básicos para uma banda raster especificada. banda número 1 é assumida se nenhuma for especificada.
  - **ST\_BandNoDataValue** - Retorna o valor em uma dada banda que não representa nenhum valor. Se nenhuma banda número 1 for assumida.
  - **ST\_BandPath** - Retorna o caminho do arquivo do sistema para uma banda armazenada em um sistema de arquivos. Se nenhum número de banda for especificado, usa-se 1.
  - **ST\_BandPixelType** - Retorna o tipo pixel para uma dada banda. Se nenhum número de banda for especificado, usa-se 1.
  - **ST\_Clip** - Retorna o raster suprimido pela geometria de entrada. Se o número de banda não for especificado, todas as bandas são processadas. Se crop não for especificado ou for VERDADE, o raster de saída é cortado.
  - **ST\_ColorMap** - Cria um novo raster de até quatro bandas 8BUI (grayscale, RGB, RGBA) do raster fonte e uma banda específica. A banda 1 usada se não especificado.
  - **ST\_Contains** - Retorna verdade se nenhum ponto do raster rastB estiver no exterior do raster rastA e pelo menos um ponto do interior do rastB estiver no interior do rastA.
  - **ST\_ContainsProperly** - Retorna verdade se o rastB intersectar o interior do rastA, mas não o limite ou exterior do rastA.
  - **ST\_ConvexHull** - Retorna o casco convexo da geometria do raster incluindo valores iguais ao BandNoDataValue. Para rasters com formas normais e não desviadas, o resultado é o mesmo que ST\_Envelope, então só é útil para rasters com formas irregulares ou desviados.
  - **ST\_Count** - Retorna o número de pixels em uma banda dada de um raster ou cobertura raster. Se nenhuma banda for especificada, o padrão é usar a banda 1. Se exclude\_nodata\_value for verdade, contará somente pixels que não são iguais ao valor nodata.
  - **ST\_CountAgg** - Agregado. Retorna o número de pixels em uma banda dada de um raster ou cobertura raster. Se nenhuma banda for especificada, o padrão é usar a banda 1. Se exclude\_nodata\_value for verdade, contará somente pixels que são diferentes ao valor NODATA.
  - **ST\_CoveredBy** - Retorna verdade se nenhum ponto do rastA estiver de fora do rastB.
  - **ST\_Covers** - Retorna verdade se nenhum ponto do rastB estiver de fora do rastA.
  - **ST\_DFullyWithin** - Retorna verdade se os rasters rastA e rastB estiverem completamente dentro da distância especificada de cada um.
  - **ST\_DWithin** - Retorna verdade se os rasters rastA e rastB estiverem dentro da distância especificada de cada um.
  - **ST\_Disjoint** - Retorna verdade se raster rastA não intersectar espacialmente com o rastB.
  - **ST\_DumpAsPolygons** - Retorna um conjunto de linhas geomval (geom,val), de uma dada banda raster. Se nenhum número de banda for especificado, o número de banda torna-se 1.
  - **ST\_DumpValues** - Obtenha os valores da banda específica como um arranjo 2-dimensional.
  - **ST\_Envelope** - Retorna a representação de polígono da extensão do raster.
  - **ST\_FromGDALRaster** - Retorna um raster de um arquivo raster GDAL suportado.
  - **ST\_GeoReference** - Retorna os metadados georreferenciados no formato GDAL ou ESRI como é comumente visto em um arquivo mundo. O padrão é GDAL.
  - **ST\_Grayscale** - Cria uma nova banda raster de uma única banda 8BUI a partir do raster fonte e bandas especificadas representando Vermelho, Verde e Azul.
  - **ST\_HasNoBand** - Retorna verdade se não existirem bandas com números dados. Se nenhum número de banda for especificado, então assume-se a banda 1.
-

- **ST\_Height** - Retorna a altura do raster em pixels.
  - **ST\_HillShade** - Retorna a iluminação hipotética de uma banda raster de elevação usando as entradas de azimute, altitude, claridade e escala fornecidas.
  - **ST\_Histogram** - Retorna um conjunto de registros que resumem um raster ou distribuição de dados de cobertura raster intervalos bin separados. O número de bins é auto calculado.
  - **ST\_Intersection** - Retorna uma raster ou conjunto de pares de valores de pixels de geometria representando a porção dividida de dois rasters ou a interseção geométrica de uma vetorização do raster e uma geometria.
  - **ST\_Intersects** - Retorna verdade se o raster rastA intersectar espacialmente com o raster rastB.
  - **ST\_IsEmpty** - Retorna verdadeiro se o raster estiver vazio (largura = 0 e altura = 0). Senão, retorna falso.
  - **ST\_MakeEmptyCoverage** - Cover georeferenced area with a grid of empty raster tiles.
  - **ST\_MakeEmptyRaster** - Retorna um raster vazio (sem bandas) das dimensões dadas (width & height), o X e Y do superior esquerdo, tamanho de pixel e rotação (scalex, scaley, skewx & skewy) e sistema de referência (srid). Se um raster passar, retorna um novo raster com o mesmo tamanho, alinhamento e SRID. Se o srid é deixado de fora, a referência espacial se torna desconhecida (0).
  - **Funções retorno de mapa algébrico embutido** - Versão função retorno - Retorna um raster de uma banda dado um ou mais rasters de entrada, os índices e uma função retorno de um usuário específico.
  - **ST\_MapAlgebraExpr** - Versão de banda raster 1: Cria uma nova banda raster formada pela aplicação de ma operação algébrica válida do PostgreSQL na banda raster de entrada de um tipo de pixel fornecido. A banda 1 é assumida se nenhuma banda for especificada.
  - **ST\_MapAlgebraExpr** - Versão de banda raster 2: Cria uma banda raster nova formada pela aplicação de uma operação algébrica válida PostgreSQL nas duas bandas raster de entrada e do tipo de pixel fornecido. A banda 1 de cada raster é assumida se nenhum número de bandas for especificado. O raster resultante será alinhado (escala, inclinação e cantos de pixel) na grade definida pelo primeiro raster e tem sua extensão definida pelo parâmetro "extenttype". O valores para "extenttype" pode ser: INTERSEÇÃO, UNIÃO, PRIMEIRO, SEGUNDO.
  - **ST\_MapAlgebraFct** - Versão de banda raster 1: Cria uma nova banda raster formada pela aplicação de uma função válida do PostgreSQL na banda raster de entrada de um tipo de pixel fornecido. A banda 1 é assumida se nenhuma banda for especificada.
  - **ST\_MapAlgebraFct** - Versão de banda 2 - Cria uma nova banda raster um formada pela aplicação de uma função PostgreSQL na 2 entrada de bandas raster e do tipo de pixel fornecido. A banda 1 é assumida se nenhuma banda for especificada. Tipo de extensão torna-se INTERSEÇÃO se não especificada.
  - **ST\_MapAlgebraFctNgb** - Versão 1-banda: o vizinho mais próximo no mapa algébrico usando a função de usuário definido PostgreSQL. Retorna um raster cujos valores são o resultado de uma função usuário PLPGSQL envolvendo uma vizinhança de valores da banda raster de entrada.
  - **ST\_MapAlgebraExpr** - Versão expressão - Retorna um raster de uma banda dado um ou mais rasters de entrada, índices de banda e uma ou mais expressões SQL de usuários específicos.
  - **ST\_MemSize** - Retorna a quantidade de espaço (em bytes) que o raster pega.
  - **ST\_MetaData** - Retorna metadados básicos sobre um objeto raster como um tanho pixel, rotação (skew), esquerda superior, inferior etc.
  - **ST\_MinConvexHull** - Retorna a geometria de casco convexo do raster excluindo os pixels SEM DADOS.
  - **ST\_NearestValue** - Retorna o valor não-NODATA mais próximo de um dado pixel de banda especificado por uma colunax e linhay ou um ponto geométrico expressado no mesmo sistema de coordenada referência do raster.
  - **ST\_Neighborhood** - Retorna um arranjo de precisão 2-D dobrada dos valores não-NODATA em torno da banda de pixel especificada ou por uma colunaX e linhaY ou um ponto geométrico expressado no mesmo sistema de coordenada de referência especial como o raster.
-

- **ST\_NotSameAlignmentReason** - Retorna a declaração de texto se os rasters estiverem alinhados e se não tiverem, uma razão do porquê.
  - **ST\_NumBands** - Retorna o número de bandas no objeto raster.
  - **ST\_Overlaps** - Retorna verdade se o raster rastA e rastB se intersectam, mas um deles não contém o outro completamente.
  - **ST\_PixelAsCentroid** - Retorna o centroide (ponto) da área representada por um pixel.
  - **ST\_PixelAsCentroids** - Retorna o centroide (ponto geométrico) para cada pixel de uma banda raster junto com o valor, as coordenadas raster X e Y de cada pixel. O ponto é o centroide da área representada por um pixel.
  - **ST\_PixelAsPoint** - Retorna um ponto geométrico do canto superior esquerdo do pixel.
  - **ST\_PixelAsPoints** - Retorna um ponto geométrico para cada pixel de uma banda raster junto com o valor, as coordenadas raster X e Y de cada pixel. As coordenadas do ponto são do ponto esquerdo superior do pixel.
  - **ST\_PixelAsPolygon** - Retorna o polígono que limita o pixel para uma linha e coluna específicas.
  - **ST\_PixelAsPolygons** - Retorna o polígono que limita cada pixel de uma banda raster ao longo do valor, as coordenadas raster X e Y de cada pixel.
  - **ST\_PixelHeight** - Retorna a altura do pixel em unidades geométricas do sistema de referência espacial.
  - **ST\_PixelOfValue** - Obtenha as coordenadas colunax, linhay do pixel cujos valores são iguais ao valor de pesquisa.
  - **ST\_PixelWidth** - Retorna a largura do pixel em unidades geométricas do sistema de referência espacial.
  - **ST\_Polygon** - Retorna um multipolígono formado pela união de pixels que têm um valor que não é um valor sem dados. Se um número de banda for especificado, usa-se 1.
  - **ST\_Quantile** - Calcula quantiles para um raster ou cobertura de tabela raster no contexto da amostra ou população. Assim, um valor poderia ser examinado para estar na porcentagem 25%, 50%, 75% do raster.
  - **ST\_RastFromHexWKB** - Return a raster value from a Hex representation of Well-Known Binary (WKB) raster.
  - **ST\_RastFromWKB** - Return a raster value from a Well-Known Binary (WKB) raster.
  - **ST\_RasterToWorldCoord** - Retorna o canto superior esquerdo do raster como X e Y geométricos (longitude e latitude) dada a coluna e linha. Coluna e linha começam em 1.
  - **ST\_RasterToWorldCoordX** - Retorna a coordenada geométrica X superior esquerda de um raster, coluna ou linha. A numeração das colunas e linhas começam no 1.
  - **ST\_RasterToWorldCoordY** - Retorna a coordenada geométrica Y superior esquerda de um raster, coluna e linha. A numeração das colunas e linhas começam no 1.
  - **ST\_Reclass** - Cria um novo raster composto por tipos de banda reclassificados do original. A nband pode ser alterada. Se nenhuma nband for especificada, usa-se a 1. Todas as outras bandas são retornadas inalteradas. Use caso: converta uma banda 16BUI para 8BUI e então adiante para uma renderização mais simples como formatos visíveis.
  - **ST\_Resample** - Resample um raster usando um algoritmo específico, novas dimensões, um canto aleatório da grade e um conjunto de rasters georreferenciando atributos definidos ou emprestados de outro raster.
  - **ST\_Rescale** - Resample um raster ajustando sua única escala (ou tamanho de pixel). Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor.
  - **ST\_Resize** - Redimensiona largura/altura novas para um raster
  - **ST\_Reskew** - Resample um raster ajustando somente sua inclinação (ou tamanho de pixel). Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor.
  - **ST\_Rotation** - Retorna a rotação do raster em radianos.
-



- **ST\_Roughness** - Retorna um raster com a "robustez" calculada de um DEM.
  - **ST\_SRID** - Retorna o identificador de referência espacial como definido na tabela spatial\_ref\_sys.
  - **ST\_SameAlignment** - Retorna verdade se os rasters têm a mesma inclinação, escala, referência espacial, e deslocamento (pixels podem ser colocados na mesma grade sem cortar eles) e falso se eles não notificarem problemas detalhados.
  - **ST\_ScaleX** - Retorna o componente X da largura do pixel em unidades do sistema de referência coordenadas.
  - **ST\_ScaleY** - Retorna o componente Y da altura do pixel em unidades do sistema de referência coordenadas.
  - **ST\_SetBandIndex** - Update the external band number of an out-db band
  - **ST\_SetBandIsNoData** - Coloca a bandeira isnodata da banda como VERDADE.
  - **ST\_SetBandNoDataValue** - Coloca o valor da banda que não representa nenhum dado. A banda 1 é assumida se nenhuma banda for especificada. Para marcar uma banda como tendo nenhum valor nodata, coloca ele = NULL.
  - **ST\_SetBandPath** - Update the external path and band number of an out-db band
  - **ST\_SetGeoReference** - Coloque os parâmetros Georeference 6 em uma única chamada. Os números deverão ser separados por espaço branco. Aceita entrar no formato GDAL ou ESRI. O padrão é GDAL.
  - **ST\_SetRotation** - Põe a rotação do raster em radianos.
  - **ST\_SetSRID** - Coloca o SRID de um raster em um srid inteiro específico definido na tabela spatial\_ref\_sys.
  - **ST\_SetScale** - Coloca os tamanhos X e Y dos pixels em unidades do sistema referencial de coordenadas. Número unidades/pixel largura/altura.
  - **ST\_SetSkew** - Coloca as georreferências X e Y distorcidas (ou parâmetro de rotação). Se somente um passar, coloca o X e o Y no mesmo valor.
  - **ST\_SetUpperLeft** - Sets the value of the upper left corner of the pixel of the raster to projected X and Y coordinates.
  - **ST\_SetValue** - Retorna o raster modificado resultante do valor de uma banda em uma dada colunax, linha pixel ou os pixels que intersectam uma geometria específica. Os números de banda começam no 1 e são assumidos como 1 se não estiverem especificados.
  - **ST\_SetValues** - Retorna o raster modificado resultante dos valores de uma dada banda.
  - **ST\_SkewX** - Retorna o desvio X georreferência (ou parâmetro e rotação).
  - **ST\_SkewY** - Retorna o desvio Y georreferência (ou parâmetro e rotação).
  - **ST\_Slope** - Retorna o declive (em graus) de uma banda raster de elevação. Útil para analisar terrenos.
  - **ST\_SnapToGrid** - Resample um raster encaixando-o em uma grade. Novos valores de pixel são calculados usando o algoritmo NearestNeighbor (english or american spelling), Bilinear, Cubic, CubicSpline ou Lanczos. O padrão é NearestNeighbor.
  - **ST\_Summary** - Retorna um texto resumo dos conteúdos do raster.
  - **ST\_SummaryStats** - Retorna as estatísticas resumidas consistindo de count, sum, mean, stddev, min, max para uma dada banda raster de um raster ou cobertura raster. A banda 1 é assumida se nenhuma banda for especificada.
  - **ST\_SummaryStatsAgg** - Agregado. Retorna as estatísticas resumidas consistindo de count, sum, mean, stddev, min, max para uma dada banda raster de um conjunto de rasters. A banda 1 é assumida se nenhuma banda for especificada.
  - **ST\_TPI** - Retorna um raster com o índice de posição topográfico calculado.
  - **ST\_TRI** - Retorna um raster com o índice de aspereza do terreno calculado.
  - **ST\_Tile** - Retorna um conjunto de rasters resultante de uma divisão do raster de entrada baseado nas dimensões desejadas nos rasters de saída.
  - **ST\_Touches** - Retorna verdade se o raster rastA e rastB têm pelo menos um ponto em comum, mas seus interiores não se intersectarem.
-

- **ST\_Transform** - Reprojeta um raster em um sistema de referência espacial conhecido para outro usando um algoritmo resampling especificado. As opções são NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos com o padrão sendo NearestNeighbor.
- **ST\_Union** - Retorna a união de um conjunto de tiles raster em um único raster composto de 1 ou mais bandas.
- **ST\_UpperLeftX** - Retorna a coordenada X superior esquerda na ref. espacial projetada.
- **ST\_UpperLeftY** - Retorna a coordenada Y superior esquerda na ref. espacial projetada.
- **ST\_Value** - Retorna o valor da banda dada com a colunax, linha e pixel ou em um ponto específico. Os números de banda começam em 1 e assumem-se 1 se não especificados. Se `exclude_nodata_value` for falso, então todos os pixels, inclusive os nodata, são considerados para intersectar e retornar valor. Se `exclude_nodata_value` não passar então lê dos metadados do raster.
- **ST\_ValueCount** - Retorna o conjunto de registros contendo uma banda pixel de valor e conta do número de pixels em uma dada banda de um raster (ou uma cobertura raster) que tem um dado conjunto de valores. Usa-se a banda 1 se nenhuma for especificada. Por padrão pixels de valor nodata não são contados. Todos os outros valores no pixel são saída e os valores de pixels são arredondados para o inteiro mais próximo.
- **ST\_Width** - Retorna a largura do raster em pixels.
- **ST\_Within** - Retorna verdade se nenhum ponto do raster `rastA` estiver no exterior do raster `rastB` e pelo menos um ponto do interior do `rastA` estiver no interior do `rastB`.
- **ST\_WorldToRasterCoord** - Retorna o canto superior esquerdo como coluna e linha dados os X e Y geométricos (longitude e latitude) ou um ponto expressado na coordenada do sistema de referência espacial do raster.
- **ST\_WorldToRasterCoordX** - Retorna a coluna no raster do ponto (pt) ou uma coordenada X e Y (xw, yw) representada no sistema de referência espacial mundial de raster.
- **ST\_WorldToRasterCoordY** - Retorna a linha no raster do ponto (pt) ou uma coordenada X e Y (xw, yw) representada no sistema de referência espacial global de raster.
- **UpdateRasterSRID** - Altera o SRID de todos os rasters na coluna e tabela do usuário especificado.

## 14.6 PostGIS Geometry / Geography / Raster Dump Functions

The functions given below are PostGIS functions that take as input or return as output a set of or single **geometry\_dump** or **geomval** data type object.

- **ST\_DumpAsPolygons** - Retorna um conjunto de linhas **geomval** (geom,val), de uma dada banda raster. Se nenhum número de banda for especificado, o número de banda torna-se 1.
- **ST\_Intersection** - Retorna um raster ou conjunto de pares de valores de pixels de geometria representando a porção dividida de dois rasters ou a interseção geométrica de uma vetorização do raster e uma geometria.
- **ST\_Dump** - Retorna um conjunto de filas **geometry\_dump** (geom,path), que fazem uma geometria g1.
- **ST\_DumpPoints** - Retorna um conjunto de filas **geometry\_dump** (geom,path) de todos os pontos que fazem uma geometria.
- **ST\_DumpRings** - Retorna um conjunto de filas **geometry\_dump**, representando os anéis interiores e exteriores de um polígono.



## 14.7 PostGIS Box Functions

The functions given below are PostGIS functions that take as input or return as output the box\* family of PostGIS spatial types. The box family of types consists of **box2d**, and **box3d**

- **Caixa2D** - Retorna uma CAIXA2D representando a extensão máxima da geometria.
- **Caixa3D** - Retorna uma CAIXA3D representando a extensão máxima da geometria.
- **Caixa3D** - Retorna a representação da caixa 3d da caixa encerrada do raster.
- **ST\_3DExtent** - uma função agregada que retorna a caixa3D que estabelece linhas de geometria.
- **ST\_3DMakeBox** - Cria uma CAIXA2D definida pelos pontos 3d dados das geometrias.
- **ST\_AsTWKB** - Retorna a geometria como TWKB, também conhecido como "Tiny Well-Known Binary"
- **ST\_Box2dFromGeoHash** - Retorna uma CAIXA2D de uma string GeoHash.
- **ST\_ClipByBox2D** - Retorna a porção de uma geometria caindo dentro de um retângulo.
- **ST\_EstimatedExtent** - Retorna a extensão "estimada" da table espacial dada. O estimada é tirado das estatísticas da coluna geométrica. O esquema atual será usado se não especificado.
- **ST\_Expand** - A caixa limitadora retorna expandida em todas as direções da caixa da geometria de entrada. Utiliza precisão dobrada
- **ST\_Extent** - uma função agregada que retorna a caixa limitadora que estabelece linhas de geometrias.
- **ST\_MakeBox2D** - Cria uma CAIXA2D definida pelos pontos dados das geometrias.
- **ST\_XMax** - Retorna o X máximo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_XMin** - Retorna o X mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_YMax** - Retorna o Y máximo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_YMin** - Retorna o Y mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_ZMax** - Retorna o Z mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_ZMin** - Retorna o Z mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
- **~(box2df,box2df)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).
- **~(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains a geometry's 2D bonding box.
- **~(geometry,box2df)** - Returns TRUE if a geometry's 2D bonding box contains a 2D float precision bounding box (BOX2DF).
- **@(box2df,box2df)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into another 2D float precision bounding box.
- **@(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into a geometry's 2D bounding box.
- **@(geometry,box2df)** - Returns TRUE if a geometry's 2D bounding box is contained into a 2D float precision bounding box (BOX2DF).
- **&&(box2df,box2df)** - Returns TRUE if two 2D float precision bounding boxes (BOX2DF) intersect each other.
- **&&(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) intersects a geometry's (cached) 2D bounding box.
- **&&(geometry,box2df)** - Returns TRUE if a geometry's (cached) 2D bounding box intersects a 2D float precision bounding box (BOX2DF).

## 14.8 PostGIS Functions that support 3D

The functions given below are PostGIS functions that do not throw away the Z-Index.

- **AddGeometryColumn** - Adiciona uma coluna geometria à uma table de atributos existente. Por padrão usa um tipo modificador em vez de restrições. Passa em falso por usar use\_typmod para obter uma restrição antiga baseada em comportamento.
  - **Box3D** - Retorna uma CAIXA3D representando a extensão máxima da geometria.
  - **DropGeometryColumn** - Remove uma coluna geometria de uma spatial table.
  - **GeometryType** - Retorna o tipo de geometria como uma string. Exemplos: 'LINESTRING', 'POLÍGONO', 'MULTIPOINT', etc.
  - **ST\_3DArea** - Computa a área de geometrias de superfície 3D. Irá retornar 0 para sólidos.
  - **ST\_3DClosestPoint** - Retorna o ponto 3 dimensional em g1 que é o mais próximo de g2. Este é o primeiro ponto da linha mais curta em três dimensões.
  - **ST\_3DDFullyWithin** - Retorna verdade se todas as geometrias 3D estiverem dentro da distância especificada de um outro.
  - **ST\_3DDWithin** - Para tipo de geometria 3d (z) verdadeiro se a distância entre duas geometrias 3d está dentro de números de unidades.
  - **ST\_3DDifference** - Representar diferença 3D
  - **ST\_3DDistance** - Para tipo geometria, retorna a menor distância cartesiana 3-dimensional (baseado no sistema de referência espacial) entre duas geometrias em unidades projetadas.
  - **ST\_3DExtent** - uma função agregada que retorna a caixa3D que estabelece linhas de geometria.
  - **ST\_3DIntersection** - Representar intersecção 3D
  - **ST\_3DIntersects** - Retorna VERDADE se as geometrias "intersectadas espacialmente" em 3d - somente para pontos, linestrings, polígonos, superfícies poliédricas (área). Com o backend SFCGAL ativado, também suporta TINS
  - **ST\_3DLength** - Retorna o comprimento 3-dimensional ou 2-dimensional da geometria se for uma linestring ou multi-linestring.
  - **ST\_3DLongestLine** - Retorna a linha 3-dimensional mais longa entre duas geometrias
  - **ST\_3DMaxDistance** - Para tipo de geometria retorna a maior distância 3-dimensional cartesiana (baseada na referência espacial) entre duas geometrias em unidade projetadas.
  - **ST\_3DPerimeter** - Retorna o perímetro 3-dimensional da geometria, se for uma polígono ou multi-polígono.
  - **ST\_3DShortestLine** - Retorna a menor linha 3-dimensional entre duas geometrias
  - **ST\_3DUnion** - Representar união 3D
  - **ST\_Accum** - Agregado. Constrói um banco de dados de geometrias.
  - **ST\_AddMeasure** - Retorna uma geometria derivada com elementos de medida interpolados linearmente entre os pontos de início e de fim.
  - **ST\_AddPoint** - Adicione um ponto para uma LineString.
  - **ST\_Affine** - Aplique uma 3a transformação afin em uma geometria.
  - **ST\_ApproximateMedialAxis** - Computa o eixo mediano aproximado de uma geometria territorial.
  - **ST\_AsBinary** - Retorna a representação do binário bem conhecido (WKB) da geometria/geografia sem os meta dados SRID.
  - **ST\_AsEWKB** - Retorna a representação binária bem conhecida (WKB) da geometria com os meta dados SRID.
  - **ST\_AsEWKT** - Retorna a representação de texto bem conhecida (WKT) da geometria com os meta dados SRID.
-

- **ST\_AsGML** - Retorna a geometria como uma versão GML com 2 ou 3 elementos.
  - **ST\_AsGeoJSON** - Retorna a geometria com um elemento GeoJSON.
  - **ST\_AsHEXEWKB** - Retorna uma geometria no formato HEXEWKB (como texto) usando little-endian (NDR) ou big-endian (XDR) encoding.
  - **ST\_AsKML** - Retorna a geometria como um elemento KML. Muitas variantes. Versão padrão=2, precisão padrão=15
  - **ST\_AsX3D** - Retorna uma geometria em X3D nó xml formato do elemento: ISO-IEC-19776-1.2-X3DEncodings-XML
  - **ST\_Boundary** - Retorna o encerramento da borda combinatória dessa geometria.
  - **ST\_BoundingDiagonal** - Retorna a diagonal da geometria fornecida da caixa limitada.
  - **ST\_CPAWithin** - Retorna verdadeiro se os pontos mais próximos da trajetória estão até a distância especificada.
  - **ST\_ClosestPointOfApproach** - Retorna a medida em que pontos interpolados na linha são mais próximos.
  - **ST\_Collect** - Retorna um valor ST\_Geometry específico de uma coleção de outras geometrias.
  - **ST\_ConvexHull** - O casco convexo de uma geometria representa a mínima geometria convexa que envolve todas as geometrias dentro do conjunto.
  - **ST\_CoordDim** - Retorna a dimensão da coordenada do valor ST\_Geometry.
  - **ST\_CurveToLine** - Converte uma CIRCULARSTRING/CURVEPOLYGON para uma LINESTRING/POLYGON
  - **ST\_DelaunayTriangles** - Retorna uma triangulação de Delaunay em volta dos pontos de entrada.
  - **ST\_Difference** - Retorna uma geometria que representa aquela parte de geometria A que não intersecta com a geometria B.
  - **ST\_DistanceCPA** - Retorna a distância entre os pontos mais próximos de uma aproximação em duas trajetórias.
  - **ST\_Dump** - Retorna um conjunto de filas geometry\_dump (geom,path), que fazem uma geometria g1.
  - **ST\_DumpPoints** - Retorna um conjunto de filas geometry\_dump (geom,path) de todos os pontos que fazem uma geometria.
  - **ST\_DumpRings** - Retorna um conjunto de filas geometry\_dump, representando os anéis interiores e exteriores de um polígono.
  - **ST\_EndPoint** - Retorna ao último ponto de uma LINESTRING ou CIRCULARLINESTRING geometria como um PONTO.
  - **ST\_ExteriorRing** - Retorna uma line string representando o anel exterior da geometria POLÍGONO. Retorna NULA se a geometria não for um polígono. Não funcionará com MULTIPOLÍGONO.
  - **ST\_Extrude** - Extrude uma superfície a um volume relacionado
  - **ST\_FlipCoordinates** - Retorna uma versão da geometria dada com os eixos X e Y virados. Útil para pessoas que construíram características latitude/longitude e precisam consertá-las.
  - **ST\_Force2D** - Força a geometria para o modo de 2 dimensões.
  - **ST\_ForceCurve** - Converte para cima uma geometria para seu tipo curvo, se aplicável.
  - **ST\_ForceLHR** - Orientação força LHR
  - **ST\_ForcePolygonCCW** - Orients all exterior rings counter-clockwise and all interior rings clockwise.
  - **ST\_ForcePolygonCW** - Orients all exterior rings clockwise and all interior rings counter-clockwise.
  - **ST\_ForceRHR** - Força a orientação dos vértices em um polígono a seguir a regra da mão direita.
  - **ST\_ForceSFS** - Força as geometrias a utilizarem os tipos disponíveis na especificação SFS 1.1.
  - **ST\_Force\_3D** - Força a geometria para um modo XYZ. Este é um apelido para a função ST\_Force\_3DZ.
  - **ST\_Force\_3DZ** - Força as geometrias para o modo XYZ.
-

- **ST\_Force\_4D** - Força as geometrias para o modo XYZM.
  - **ST\_Force\_Collection** - Converte a geometria para um GEOMETRYCOLLECTION.
  - **ST\_GeomFromEWKB** - Retorna um valor ST\_Geometry específico da representação binária estendida bem conhecida (EWKB).
  - **ST\_GeomFromEWKT** - Retorna um valor ST\_Geometry específico da representação de texto estendida bem conhecida (EWKT).
  - **ST\_GeomFromGML** - Utiliza como entrada uma representação GML de geometria e como saída um objeto de geometria PostGIS
  - **ST\_GeomFromGeoJSON** - Utiliza como entrada uma representação geojson de uma geometria e como saída um objeto de geometria PostGIS
  - **ST\_GeomFromKML** - Utiliza como entrada uma representação KML de geometria e como saída um objeto de geometria PostGIS
  - **ST\_GeometricMedian** - Retorna a mediana de um MultiPonto.
  - **ST\_GeometryN** - Retorna a geometria de 1-base Nth se a geometria é uma GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINestring, MULTICURVE ou (MULTI)POLYGON, POLYHEDRALSURFACE. Senão, retorna NULA.
  - **ST\_GeometryType** - Retorna o tipo de geometria de valor ST\_Geometry.
  - **ST\_HasArc** - Retorna verdade se uma geometria ou coleção de geometria contém uma string circular
  - **ST\_InteriorRingN** - Retorna o anel linestring Nth interior do polígono. Retorna NULO se a geometria não for um polígono ou o dado N está fora da extensão.
  - **ST\_InterpolatePoint** - Retorna o valor da dimensão de medida da geometria no ponto fechado para o ponto fornecido.
  - **ST\_IsClosed** - Retorna VERDADEIRO se os pontos de começo e fim da LINestring são coincidentes. Para superfície poliédrica está fechada (volumétrica).
  - **ST\_IsCollection** - Retorna VERDADEIRO se o argumento é uma coleção (MULTI\*, GEOMETRYCOLLECTION, ...)
  - **ST\_IsPlanar** - Verifique se a superfície é ou não planar
  - **ST\_IsPolygonCCW** - Returns true if all exterior rings are oriented counter-clockwise and all interior rings are oriented clockwise.
  - **ST\_IsPolygonCW** - Returns true if all exterior rings are oriented clockwise and all interior rings are oriented counter-clockwise.
  - **ST\_IsSimple** - Retorna (VERDADEIRA) se essa geometria não tem nenhum ponto irregular, como auto intersecção ou tangenciação.
  - **ST\_IsSolid** - teste se a geometria é um sólido. Nenhuma verificação de validade é representada.
  - **ST\_IsValidTrajectory** - Retorna true se a geometria possui uma trajetória válida.
  - **ST\_Length\_Spheroid** - Calcula o comprimento/perímetro 2D ou 3D de uma geometria em um elipsoide. É útil se as coordenadas da geometria estão em longitude/latitude e um comprimento é desejado sem reprojeção.
  - **ST\_LineFromMultiPoint** - Cria uma linestring de um multiponto geométrico.
  - **ST\_LineInterpolatePoint** - Retorna um ponto interpolar ao longo de uma linha. Segundo argumento é um float8 entre 0 e 1 representando fração do comprimento total da linestring do ponto tem que ser localizado.
  - **ST\_LineInterpolatePoints** - Returns one or more points interpolated along a line.
  - **ST\_LineSubstring** - Retorna uma linestring sendo uma substring da de entrada começando e finalizando nas frações dadas do total 2d de comprimento. Segundo e terceiro argumentos são valores float8 entre 0 e 1.
  - **ST\_LineToCurve** - Converte uma LINestring/POLYGON para um CIRCULARSTRING, CURVEPOLYGON
  - **ST\_LocateBetweenElevations** - Retorna um valor de geometria derivada (coleção) com elementos que inserem a variação específica de elevações. Apenas 3D, 4D LINestrings e MULTILINestrings são suportadas.
-

- **ST\_M** - Retorna a coordenada M do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.
  - **ST\_MakeLine** - Cria uma Linestring de ponto, multiponto ou linha das geometrias.
  - **ST\_MakePoint** - Creates a 2D, 3DZ or 4D point geometry.
  - **ST\_MakePolygon** - Cria uma polígono formado pela dada shell. As geometrias de entrada devem ser LINESTRINGS fechadas.
  - **ST\_MakeSolid** - Molde a geometria para um sólido. Nenhuma verificação é apresentada. Para obter um sólido válido, a geometria de entrada deve ser uma superfície poliédrica fechada ou um TIN fechado.
  - **ST\_MakeValid** - Tenta tornar uma geometria inválida válida sem perder vértices.
  - **ST\_MemSize** - Retorna a quantidade de espaço (em bytes) que a geometria pega.
  - **ST\_MemUnion** - O mesmo que ST\_Union, somente memória amigável (utiliza menos memória e mais tempo de processador).
  - **ST\_NDims** - Retorna a dimensão coordenada da geometria como uma small int. Os valores são: 2, 3 ou 4.
  - **ST\_NPoints** - Retorna o número de pontos (vértices) em uma geometria.
  - **ST\_NRings** - Se a geometria for um polígono ou multi polígono, retorna o número de anéis.
  - **ST\_Node** - Nodar um conjunto de linestrings.
  - **ST\_NumGeometries** - Se a geometria é uma GEOMETRYCOLLECTION (ou MULTI\*), retorna o número de geometria, para geometrias únicas retornará 1, senão retorna NULO.
  - **ST\_NumPatches** - Retorna o número de faces em uma superfícies poliédrica. Retornará nulo para geometrias não poliédricas.
  - **ST\_Orientation** - Determine orientação da superfície
  - **ST\_PatchN** - Retorna a geometria (face) de 1-base Nth se a geometria é uma POLYHEDRALSURFACE, POLYHEDRAL-SURFACEM. Senão, retorna NULA.
  - **ST\_PointFromWKB** - Faz uma geometria a partir de um WKB com o SRID dado
  - **ST\_PointN** - Retorna o ponto Nth na primeira linestring ou linestring circular na geometria. Valores negativos são contados tardiamente do fim da linestring. Retorna NULA se não há uma linestring na geometria.
  - **ST\_PointOnSurface** - Retorna um POINT garantido a ficar na superfície.
  - **ST\_Points** - Retorna uma multilinestring contendo todas as coordenadas de uma geometria.
  - **ST\_Polygon** - Retorna um polígono construído de linestring e SRID especificados.
  - **ST\_RemovePoint** - Remove um ponto de uma linestring.
  - **ST\_RemoveRepeatedPoints** - Retorna uma versão da geometria dada com pontos removidos duplicados.
  - **ST\_Reverse** - Retorna a geometria com a ordem dos vértices revertida.
  - **ST\_Rotate** - Rotaciona uma geometria em rotRadians em sentido anti-horário de sua origem.
  - **ST\_RotateX** - Rotaciona uma geometria rotRadians em cima do eixo X.
  - **ST\_RotateY** - Rotaciona uma geometria rotRadians em cima do eixo Y.
  - **ST\_RotateZ** - Rotaciona uma geometria rotRadians em cima do eixo Z.
  - **ST\_Scale** - Escala uma geometria pelos fatores dados.
  - **ST\_SetPoint** - Substitui ponto de uma linestring com um dado ponto.
  - **ST\_Shift\_Longitude** - Coordenadas de geometria fechada entre -180...180 e 0...360 extensões.
  - **ST\_SnapToGrid** - Rompe todos os pontos da geometria de entrada para uma rede regular.
-

- **ST\_StartPoint** - Retorna ao último ponto de uma LINESTRING geometria como um PONTO.
  - **ST\_StraightSkeleton** - Calcule um esqueleto em linha reta de uma geometria
  - **ST\_SwapOrdinates** - Retorna uma versão da geometria dada com os valores ordenados dados trocados.
  - **ST\_SymDifference** - Retorna uma geometria que representa as porções de A e B que não intersectam. É chamado de diferença simétrica, porque  $ST\_SymDifference(A,B) = ST\_SymDifference(B,A)$ .
  - **ST\_Tessellate** - Representa superfície tesselação de um polígono ou superfície poliédrica e retorna como uma TIN ou coleção de TINS
  - **ST\_TransScale** - Translada uma geometria dando coeficientes e deslocamentos.
  - **ST\_Translate** - Translação de uma geometria pelos dados deslocamentos.
  - **ST\_UnaryUnion** - Parecido com ST\_Union, mas funcionando no nível do componente da geometria.
  - **ST\_Volume** - Computa o volume de um sólido 3D. Se aplicado a geometrias com superfícies (mesmo fechadas), irão retornar 0.
  - **ST\_WrapX** - Envolve uma geometria em torno de um valor X.
  - **ST\_X** - Retorna a coordenada X do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.
  - **ST\_XMax** - Retorna o X máximo de uma caixa limitante 2d ou 3d ou uma geometria.
  - **ST\_XMin** - Retorna o X mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
  - **ST\_Y** - Retorna a coordenada Y do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.
  - **ST\_YMax** - Retorna o Y máximo de uma caixa limitante 2d ou 3d ou uma geometria.
  - **ST\_YMin** - Retorna o Y mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
  - **ST\_Z** - Retorna a coordenada Z do ponto, ou NULA se não estiver disponível. Entrada deve ser um ponto.
  - **ST\_ZMax** - Retorna o Z máximo de uma caixa limitante 2d ou 3d ou uma geometria.
  - **ST\_ZMin** - Retorna o Z mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
  - **ST\_Zmflag** - Retorna a bandeira ZM (dimensão semântica) das geometrias como uma small int. Os valores são: 0=2d, 1=3dm, 2=3dz, 3=4d.
  - **TG\_Equals** - Retorna verdade se duas topogeometrias forem compostas da mesma topologia primitiva
  - **TG\_Intersects** - Retorna verdade se algum par de primitivos das duas topologias se intersectar.
  - **UpdateGeometrySRID** - Atualiza a SRID de todas as características em uma coluna geométrica, geometry\_columns metadados e srid. Se foi executado com restrições, elas serão atualizadas com a nova restrição srid. Se a antiga foi executada pelo definição de tipo, ela será alterada.
  - **geometry\_overlaps\_nd** - Retorna VERDADE se a caixa limitadora n-D de A intersecta a caixa limitadora n-D de B.
  - **overlaps\_nd\_geometry\_gidx** - Returns TRUE if a geometry's (cached) n-D bounding box intersects a n-D float precision bounding box (GIDX).
  - **overlaps\_nd\_gidx\_geometry** - Returns TRUE if a n-D float precision bounding box (GIDX) intersects a geometry's (cached) n-D bounding box.
  - **overlaps\_nd\_gidx\_gidx** - Returns TRUE if two n-D float precision bounding boxes (GIDX) intersect each other.
  - **postgis\_sfcgal\_version** - retorna a versão do SFCGAL em uso
-

## 14.9 PostGIS Curved Geometry Support Functions

The functions given below are PostGIS functions that can use CIRCULARSTRING, CURVEPOLYGON, and other curved geometry types

- **AddGeometryColumn** - Adiciona uma coluna geometria à uma table de atributos existente. Por padrão usa um tipo modificador em vez de restrições. Passa em falso por usar use\_typmod para obter uma restrição antiga baseada em comportamento.
- **Caixa2D** - Retorna uma CAIXA2D representando a extensão máxima da geometria.
- **Caixa3D** - Retorna uma CAIXA3D representando a extensão máxima da geometria.
- **DropGeometryColumn** - Remove uma coluna geometria de uma spatial table.
- **Tipo de geometria** - Retorna o tipo de geometria como uma string. Exemplos: 'LINESTRING', 'POLÍGONO', 'MULTI-POINT', etc.
- **PostGIS\_AddBBBox** - Adicionar o retângulo envolvente a geometria.
- **PostGIS\_DropBBBox** - Exclui o cache de retângulos envolventes da geometria.
- **PostGIS\_HasBBBox** - Retorna verdadeiro se o retângulo desta geometria está cacheado. FALSO caso contrário.
- **ST\_3DExtent** - uma função agregada que retorna a caixa3D que estabelece linhas de geometria.
- **ST\_Accum** - Agregado. Constrói um banco de dados de geometrias.
- **ST\_Affine** - Aplique uma 3a transformação afim em uma geometria.
- **ST\_AsBinary** - Retorna a representação do binário bem conhecido (WKB) da geometria/geografia sem os meta dados SRID.
- **ST\_AsEWKB** - Retorna a representação binária bem conhecida (WKB) da geometria com os meta dados SRID.
- **ST\_AsEWKT** - Retorna a representação de texto bem conhecida (WKT) da geometria com os meta dados SRID.
- **ST\_AsHEXEWKB** - Retorna uma geometria no formato HEXEWKB (como texto) usando little-endian (NDR) ou big-endian (XDR) encoding.
- **ST\_AsText** - Retorna a representação de texto bem conhecida (WKT) da geometria/geografia sem os meta dados do SRID.
- **ST\_Collect** - Retorna um valor ST\_Geometry específico de uma coleção de outras geometrias.
- **ST\_CoordDim** - Retorna a dimensão da coordenada do valor ST\_Geometry.
- **ST\_CurveToLine** - Converte uma CIRCULARSTRING/CURVEPOLYGON para uma LINESTRING/POLYGON
- **ST\_Distance** - For geometry type returns the 2D Cartesian distance between two geometries in projected units (based on spatial reference system). For geography type defaults to return minimum geodesic distance between two geographies in meters.
- **ST\_Dump** - Retorna um conjunto de filas geometry\_dump (geom,path), que fazem uma geometria g1.
- **ST\_DumpPoints** - Retorna um conjunto de filas geometry\_dump (geom,path) de todos os pontos que fazem uma geometria.
- **ST\_EndPoint** - Retorna ao último ponto de uma LINESTRING ou CIRCULARLINESTRING geometria como um PONTO.
- **ST\_EstimatedExtent** - Retorna a extensão "estimada" da table espacial dada. O estimada é tirado das estatísticas da coluna geométrica. O esquema atual será usado se não especificado.
- **ST\_FlipCoordinates** - Retorna uma versão da geometria dada com os eixos X e Y virados. Útil para pessoas que construíram características latitude/longitude e precisam consertá-las.
- **ST\_Force2D** - Força a geometria para o modo de 2 dimensões.
- **ST\_ForceCurve** - Converte para cima uma geometria para seu tipo curvo, se aplicável.
- **ST\_ForceSFS** - Força as geometrias a utilizarem os tipos disponíveis na especificação SFS 1.1.



- **ST\_Force3D** - Força a geometria para um modo XYZ. Este é um apelido para a função ST\_Force\_3DZ.
  - **ST\_Force3DM** - Força as geometrias para o modo XYM.
  - **ST\_Force3DZ** - Força as geometrias para o modo XYZ.
  - **ST\_Force4D** - Força as geometrias para o modo XYZM.
  - **ST\_ForceCollection** - Converte a geometria para um GEOMETRYCOLLECTION.
  - **ST\_GeoHash** - Retorna uma representação GeoHash da geometria.
  - **ST\_GeogFromWKB** - Cria uma ocasião geografia de uma geometria binária bem conhecida (WKB) ou binário estendido bem conhecido (EWKB).
  - **ST\_GeomFromEWKB** - Retorna um valor ST\_Geometry específico da representação binária estendida bem conhecida (EWKB).
  - **ST\_GeomFromEWKT** - Retorna um valor ST\_Geometry específico da representação de texto estendida bem conhecida (EWKT).
  - **ST\_GeomFromText** - Retorna um valor ST\_Geometry específico da representação de texto bem conhecida (WKT).
  - **ST\_GeomFromWKB** - Criar uma geometria exemplo de um representação bem conhecida de geometria binária (WKB) e SRID opcional.
  - **ST\_GeometryN** - Retorna a geometria de 1-base Nth se a geometria é uma GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINestring, MULTICURVE ou (MULTI)POLYGON, POLYHEDRALSURFACE. Senão, retorna NULA.
  - **=** - Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B.
  - **&<|** - Retorna VERDADE se a caixa limitadora de A sobrepõe ou está abaixo de B.
  - **ST\_HasArc** - Retorna verdade se uma geometria ou coleção de geometria contém uma string circular
  - **ST\_IsClosed** - Retorna VERDADEIRO se os pontos de começo e fim da LINestring são coincidentes. Para superfície poliédrica está fechada (volumétrica).
  - **ST\_IsCollection** - Retorna VERDADEIRO se o argumento é uma coleção (MULTI\*, GEOMETRYCOLLECTION, ...)
  - **ST\_IsEmpty** - Retorna verdadeiro se essa geometria é uma coleção vazia, polígono, ponto etc.
  - **ST\_LineToCurve** - Converte uma LINestring/POLYGON para um CIRCULARSTRING, CURVEPOLYGON
  - **ST\_MemSize** - Retorna a quantidade de espaço (em bytes) que a geometria pega.
  - **ST\_NPoints** - Retorna o número de pontos (vértices) em uma geometria.
  - **ST\_NRings** - Se a geometria for um polígono ou multi polígono, retorna o número de anéis.
  - **ST\_PointFromWKB** - Faz uma geometria a partir de um WKB com o SRID dado
  - **ST\_PointN** - Retorna o ponto Nth na primeira linestring ou linestring circular na geometria. Valores negativos são contados tardiamente do fim da linestring. Retorna NULA se não há uma linestring na geometria.
  - **ST\_Points** - Retorna uma multilinestring contendo todas as coordenadas de uma geometria.
  - **ST\_Rotate** - Rotaciona uma geometria em rotRadians em sentido anti-horário de sua origem.
  - **ST\_RotateZ** - Rotaciona uma geometria rotRadians em cima do eixo Z.
  - **ST\_SRID** - Retorna o identificador de referência espacial para a ST\_Geometry como definido na table spatial\_ref\_sys.
  - **ST\_Scale** - Escala uma geometria pelos fatores dados.
  - **ST\_SetSRID** - Configure SRID em uma geometria para um valor inteiro específico.
  - **ST\_StartPoint** - Retorna ao último ponto de uma LINestring geometria como um PONTO.
-



- **ST\_Summary** - Retorna um texto resumo dos conteúdos da geometria.
- **ST\_SwapOrdinates** - Retorna uma versão da geometria dada com os valores ordenados dados trocados.
- **ST\_TransScale** - Translada uma geometria dando coeficientes e deslocamentos.
- **ST\_Transform** - Retorna uma nova geometria com suas coordenadas transformadas em diferentes referências espaciais.
- **ST\_Translate** - Translação de uma geometria pelos dados deslocamentos.
- **ST\_XMax** - Retorna o X máximo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_XMin** - Retorna o X mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_YMax** - Retorna o Y máximo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_YMin** - Retorna o Y mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_ZMax** - Retorna o Z máximo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_ZMin** - Retorna o Z mínimo de uma caixa limitante 2d ou 3d ou uma geometria.
- **ST\_Zmflag** - Retorna a bandeira ZM (dimensão semântica) das geometrias como uma small int. Os valores são: 0=2d, 1=3dm, 2=3dz, 3=4d.
- **UpdateGeometrySRID** - Atualiza a SRID de todas as características em uma coluna geométrica, geometry\_columns metadados e srid. Se foi executado com restrições, elas serão atualizadas com a nova restrição srid. Se a antiga foi executada pelo definição de tipo, ela será alterada.
- **~(box2df,box2df)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).
- **~(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains a geometry's 2D bonding box.
- **~(geometry,box2df)** - Returns TRUE if a geometry's 2D bonding box contains a 2D float precision bounding box (GIDX).
- **&&** - Retorna VERDADE se a caixa limitadora 2D de A intersecta a caixa limitadora 2D de B.
- **&&&** - Retorna VERDADE se a caixa limitadora n-D de A intersecta a caixa limitadora n-D de B.
- **@(box2df,box2df)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into another 2D float precision bounding box.
- **@(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into a geometry's 2D bounding box.
- **@(geometry,box2df)** - Returns TRUE if a geometry's 2D bounding box is contained into a 2D float precision bounding box (BOX2DF).
- **&&(box2df,box2df)** - Returns TRUE if two 2D float precision bounding boxes (BOX2DF) intersect each other.
- **&&(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) intersects a geometry's (cached) 2D bounding box.
- **&&(geometry,box2df)** - Returns TRUE if a geometry's (cached) 2D bounding box intersects a 2D float precision bounding box (BOX2DF).
- **&&&(geometry,gidx)** - Returns TRUE if a geometry's (cached) n-D bounding box intersects a n-D float precision bounding box (GIDX).
- **&&&(gidx,geometry)** - Returns TRUE if a n-D float precision bounding box (GIDX) intersects a geometry's (cached) n-D bounding box.
- **&&&(gidx,gidx)** - Returns TRUE if two n-D float precision bounding boxes (GIDX) intersect each other.

## 14.10 PostGIS Polyhedral Surface Support Functions

The functions given below are PostGIS functions that can use POLYHEDRALSURFACE, POLYHEDRALSURFACEM geometries






- **Caixa2D** - Retorna uma CAIXA2D representando a extensão máxima da geometria.
  - **Caixa3D** - Retorna uma CAIXA3D representando a extensão máxima da geometria.
  - **Tipo de geometria** - Retorna o tipo de geometria como uma string. Exemplos: 'LINESTRING', 'POLÍGONO', 'MULTIPOINT', etc.
  - **ST\_3DArea** - Computa a área de geometrias de superfície 3D. Irá retornar 0 para sólidos.
  - **ST\_3DClosestPoint** - Retorna o ponto 3 dimensional em g1 que é o mais próximo de g2. Este é o primeiro ponto da linha mais curta em três dimensões.
  - **ST\_3DDFullyWithin** - Retorna verdade se todas as geometrias 3D estiverem dentro da distância especificada de um outro.
  - **ST\_3DDWithin** - Para tipo de geometria 3d (z) verdadeiro se a distância entre duas geometrias 3d está dentro de números de unidades.
  - **ST\_3DDifference** - Representar diferença 3D
  - **ST\_3DDistance** - Para tipo geometria, retorna a menor distância cartesiana 3-dimensional (baseado no sistema de referência espacial) entre duas geometrias em unidades projetadas.
  - **ST\_3DExtent** - uma função agregada que retorna a caixa3D que estabelece linhas de geometria.
  - **ST\_3DIntersection** - Representar intersecção 3D
  - **ST\_3DIntersects** - Retorna VERDADE se as geometrias "intersectadas espacialmente" em 3d - somente para pontos, linestrings, polígonos, superfícies poliédricas (área). Com o backend SFCGAL ativado, também suporta TINS
  - **ST\_3DLongestLine** - Retorna a linha 3-dimensional mais longa entre duas geometrias
  - **ST\_3DMaxDistance** - Para tipo de geometria retorna a maior distância 3-dimensional cartesiana (baseada na referência espacial) entre duas geometrias em unidade projetadas.
  - **ST\_3DShortestLine** - Retorna a menor linha 3-dimensional entre duas geometrias
  - **ST\_3DUnion** - Representar união 3D
  - **ST\_Accum** - Agregado. Constrói um banco de dados de geometrias.
  - **ST\_Affine** - Aplique uma 3a transformação afim em uma geometria.
  - **ST\_ApproximateMedialAxis** - Computa o eixo mediano aproximado de uma geometria territorial.
  - **ST\_Area** - Retorna a área da superfície se ela for um polígono ou multipolígono. Para geometria, uma área cartesiana 2D é determinada com unidades especificadas pelo SRID. Para geografia, a área é determinada em uma superfície com unidades em metros quadrados.
  - **ST\_AsBinary** - Retorna a representação do binário bem conhecido (WKB) da geometria/geografia sem os meta dados SRID.
  - **ST\_AsEWKB** - Retorna a representação binária bem conhecida (WKB) da geometria com os meta dados SRID.
  - **ST\_AsEWKT** - Retorna a representação de texto bem conhecida (WKT) da geometria com os meta dados SRID.
  - **ST\_AsGML** - Retorna a geometria como uma versão GML com 2 ou 3 elementos.
  - **ST\_AsX3D** - Retorna uma geometria em X3D nó xml formato do elemento: ISO-IEC-19776-1.2-X3DEncodings-XML
  - **ST\_CoordDim** - Retorna a dimensão da coordenada do valor ST\_Geometry.
-


















































- **ST\_Dimension** - A dimensão herdada desse objeto geométrico, o qual deve ser menor ou igual à dimensão coordenada.
  - **ST\_Dump** - Retorna um conjunto de filas geometry\_dump (geom,path), que fazem uma geometria g1.
  - **ST\_DumpPoints** - Retorna um conjunto de filas geometry\_dump (geom,path) de todos os pontos que fazem uma geometria.
  - **ST\_Expand** - A caixa limitadora retorna expandida em todas as direções da caixa da geometria de entrada. Utiliza precisão dobrada
  - **ST\_Extent** - uma função agregada que retorna a caixa limitadora que estabelece linhas de geometrias.
  - **ST\_Extrude** - Extrude uma superfície a um volume relacionado
  - **ST\_FlipCoordinates** - Retorna uma versão da geometria dada com os eixos X e Y virados. Útil para pessoas que construíram características latitude/longitude e precisam consertá-las.
  - **ST\_Force2D** - Força a geometria para o modo de 2 dimensões.
  - **ST\_ForceLHR** - Orientação força LHR
  - **ST\_ForceRHR** - Força a orientação dos vértices em um polígono a seguir a regra da mão direita.
  - **ST\_ForceSFS** - Força as geometrias a utilizarem os tipos disponíveis na especificação SFS 1.1.
  - **ST\_Force3D** - Força a geometria para um modo XYZ. Este é um apelido para a função ST\_Force\_3DZ.
  - **ST\_Force3DZ** - Força as geometrias para o modo XYZ.
  - **ST\_ForceCollection** - Converte a geometria para um GEOMETRYCOLLECTION.
  - **ST\_GeomFromEWKB** - Retorna um valor ST\_Geometry específico da representação binária estendida bem conhecida (EWKB).
  - **ST\_GeomFromEWKT** - Retorna um valor ST\_Geometry específico da representação de texto estendida bem conhecida (EWKT).
  - **ST\_GeomFromGML** - Utiliza como entrada uma representação GML de geometria e como saída um objeto de geometria PostGIS
  - **ST\_GeometryN** - Retorna a geometria de 1-base Nth se a geometria é uma GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINestring, MULTICURVE ou (MULTI)POLYGON, POLYHEDRALSURFACE. Senão, retorna NULA.
  - **ST\_GeometryType** - Retorna o tipo de geometria de valor ST\_Geometry.
  - **=** - Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B.
  - **&<l** - Retorna VERDADE se a caixa limitadora de A sobrepõe ou está abaixo de B.
  - **~=** - Retorna VERDADE se a caixa limitadora de A é a mesma de B.
  - **ST\_IsClosed** - Retorna VERDADEIRO se os pontos de começo e fim da LINestring são coincidentes. Para superfície poliédrica está fechada (volumétrica).
  - **ST\_IsPlanar** - Verifique se a superfície é ou não planar
  - **ST\_IsSolid** - teste se a geometria é um sólido. Nenhuma verificação de validade é representada.
  - **ST\_MakeSolid** - Molde a geometria para um sólido. Nenhuma verificação é apresentada. Para obter um sólido válido, a geometria de entrada deve ser uma superfície poliédrica fechada ou um TIN fechado.
  - **ST\_MemSize** - Retorna a quantidade de espaço (em bytes) que a geometria pega.
  - **ST\_NPoints** - Retorna o número de pontos (vértices) em uma geometria.
  - **ST\_NumGeometries** - Se a geometria é uma GEOMETRYCOLLECTION (ou MULTI\*), retorna o número de geometria, para geometrias únicas retornará 1, senão retorna NULO.
  - **ST\_NumPatches** - Retorna o número de faces em uma superfícies poliédrica. Retornará nulo para geometrias não poliédricas.
-










- **ST\_PatchN** - Retorna a geometria (face) de 1-base Nth se a geometria é uma POLYHEDRALSURFACE, POLYHEDRAL-SURFACEM. Senão, retorna NULA.
  - **ST\_RemoveRepeatedPoints** - Retorna uma versão da geometria dada com pontos removidos duplicados.
  - **ST\_Reverse** - Retorna a geometria com a ordem dos vértices revertida.
  - **ST\_Rotate** - Rotaciona uma geometria em rotRadians em sentido anti-horário de sua origem.
  - **ST\_RotateX** - Rotaciona uma geometria rotRadians em cima do eixo X.
  - **ST\_RotateY** - Rotaciona uma geometria rotRadians em cima do eixo Y.
  - **ST\_RotateZ** - Rotaciona uma geometria rotRadians em cima do eixo Z.
  - **ST\_Scale** - Escala uma geometria pelos fatores dados.
  - **ST\_ShiftLongitude** - Coordenadas de geometria fechada entre -180...180 e 0...360 extensões.
  - **ST\_StraightSkeleton** - Calcule um esqueleto em linha reta de uma geometria
  - **ST\_Summary** - Retorna um texto resumo dos conteúdos da geometria.
  - **ST\_SwapOrdinates** - Retorna uma versão da geometria dada com os valores ordenados dados trocados.
  - **ST\_Tessellate** - Representa superfície tesselação de um polígono ou superfície poliédrica e retorna como uma TIN ou coleção de TINS
  - **ST\_Transform** - Retorna uma nova geometria com suas coordenadas transformadas em diferentes referências espaciais.
  - **ST\_Volume** - Computa o volume de um sólido 3D. Se aplicado a geometrias com superfícies (mesmo fechadas), irão retornar 0.
  - **~(box2df,box2df)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).
  - **~(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains a geometry's 2D bonding box.
  - **~(geometry,box2df)** - Returns TRUE if a geometry's 2D bonding box contains a 2D float precision bounding box (GIDX).
  - **&&** - Retorna VERDADE se a caixa limitadora 2D de A intersecta a caixa limitadora 2D de B.
  - **&&&** - Retorna VERDADE se a caixa limitadora n-D de A intersecta a caixa limitadora n-D de B.
  - **@(box2df,box2df)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into another 2D float precision bounding box.
  - **@(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into a geometry's 2D bounding box.
  - **@(geometry,box2df)** - Returns TRUE if a geometry's 2D bounding box is contained into a 2D float precision bounding box (BOX2DF).
  - **&&(box2df,box2df)** - Returns TRUE if two 2D float precision bounding boxes (BOX2DF) intersect each other.
  - **&&(box2df,geometry)** - Returns TRUE if a 2D float precision bounding box (BOX2DF) intersects a geometry's (cached) 2D bounding box.
  - **&&(geometry,box2df)** - Returns TRUE if a geometry's (cached) 2D bounding box intersects a 2D float precision bounding box (BOX2DF).
  - **&&&(geometry,gidx)** - Returns TRUE if a geometry's (cached) n-D bounding box intersects a n-D float precision bounding box (GIDX).
  - **&&&(gidx,geometry)** - Returns TRUE if a n-D float precision bounding box (GIDX) intersects a geometry's (cached) n-D bounding box.
  - **&&&(gidx,gidx)** - Returns TRUE if two n-D float precision bounding boxes (GIDX) intersect each other.
  - **postgis\_sfcgal\_version** - retorna a versão do SFCGAL em uso
-

### 14.11 PostGIS Function Support Matrix









Below is an alphabetical listing of spatial specific functions in PostGIS and the kinds of spatial types they work with or OGC/SQL compliance they try to conform to.

- A  means the function works with the type or subtype natively.
- A  means it works but with a transform cast built-in using cast to geometry, transform to a "best srid" spatial ref and then cast back. Results may not be as expected for large areas or areas at poles and may accumulate floating point junk.
- A  means the function works with the type because of a auto-cast to another such as to box3d rather than direct type support.
- A  means the function only available if PostGIS compiled with SFCGAL support.
- A  means the function support is provided by SFCGAL if PostGIS compiled with SFCGAL support, otherwise GEOS/built-in support.
- geom - Basic 2D geometry support (x,y).
- geog - Basic 2D geography support (x,y).
- 2.5D - basic 2D geometries in 3 D/4D space (has Z or M coord).
- PS - Polyhedral surfaces
- T - Triangles and Triangulated Irregular Network surfaces (TIN)

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
Caixa2D							
Caixa3D							
Find_SRID							
Tipo de geometria							
ST_3DArea							
ST_3DClosestPoint							
ST_3DDFullyWithin							
ST_3DDWithin							
ST_3DDifference							
ST_3DDistance							
ST_3DExtent							
ST_3DIntersection							
ST_3DIntersects							

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_3DLength	✓		✓				
ST_3DLongestLine	✓		✓			✓	
ST_3DMakeBox	✓						
ST_3DMaxDistance	✓		✓			✓	
ST_3DPerímetro	✓		✓				
ST_3DShortestLine	✓		✓			✓	
ST_3DUnion							
ST_Accum	✓		✓	✓		✓	✓
ST_AddMeasure			✓				
ST_AddPoint	✓		✓				
ST_Affine	✓		✓	✓		✓	✓
ST_Angle	✓						
ST_ApproximateMedialAxis							
ST_Area	✓	✓					
ST_AsBinary	✓	✓	✓	✓	✓	✓	✓
ST_AsEWKB	✓		✓	✓		✓	✓
ST_AsEWKT	✓	✓	✓	✓		✓	✓
ST_AsEncodedPoly	✓						
ST_AsGML	✓	✓	✓			✓	✓
ST_AsGeoJSON	✓	✓	✓				
ST_AsGeobuf							
ST_AsHEXEWKB	✓		✓	✓			
ST_AsKML	✓	✓	✓				
ST_AsLatLonText	✓						
ST_AsMVT	✓	✓					
ST_AsMVTGeom	✓	✓					
ST_AsSVG	✓	✓					
ST_AsTWKB	✓						
ST_AsText	✓	✓		✓	✓		
ST_AsX3D	✓		✓			✓	✓
ST_Azimuth							
ST_BdMPolyFromText							
ST_BdPolyFromText							
ST_Boundary			✓		✓		

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_BoundingDiagonal			✓				
ST_Box2dFromGeoPolyh	✓						
ST_Buffer	✓	😄			✓		
ST_BuildArea	✓						
ST_CPAWithin	✓		✓				
ST_Centroid	✓	✓			✓		
ST_ChaikinSmooth	✓						
ST_ClipByBox2D	✓						
ST_ClosestPoint							
ST_ClosestPointOfFace	✓		✓				
ST_ClusterDBSCAN							
ST_ClusterIntersect	✓						
ST_ClusterKMeans							
ST_ClusterWithin	✓						
ST_Collect	✓		✓	✓			
ST_CollectionExtract	✓						
ST_CollectionHomogenize	✓						
ST_ConcaveHull	✓						
ST_Contains					✓		
ST_ContainsProperly	✓						
ST_ConvexHull	✓		✓		✓		
ST_CoordDim			✓	✓	✓	✓	✓
ST_CoveredBy	✓	✓					
ST_Covers	✓	✓					
ST_Crosses	✓				✓		
ST_CurveToLine	✓		✓	✓	✓		
ST_DFullyWithin	✓						
ST_DWithin	✓	✓					
ST_DelaunayTriangulation	✓		✓				✓
ST_Difference	✓		✓		✓		
ST_Dimension					✓	✓	✓
ST_Disjoint	✓				✓		
ST_Distance	✓	✓		✓	📏		
ST_DistanceCPA	✓		✓				

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_DistanceSphere	✓						
ST_DistanceSpheroid	✓						
ST_Dump	✓		✓	✓		✓	✓
ST_DumpPoints	✓		✓	✓		✓	✓
ST_DumpRings	✓		✓				
ST_EndPoint			✓	✓	✓		
ST_Envelope					✓		
ST_Equals	✓				✓		
ST_EstimatedExtent	✗			✓			
ST_Expand	✓					✓	✓
ST_Extent	✓					✓	✓
ST_ExteriorRing	✓		✓		✓		
ST_Extrude							
ST_FilterByM	✓						
ST_FlipCoordinates	✓		✓	✓		✓	✓
ST_Force2D	✓		✓	✓		✓	
ST_ForceCurve	✓		✓	✓			
ST_ForceLHR							
ST_ForcePolygonCCW	✓		✓				
ST_ForcePolygonCW	✓		✓				
ST_ForceRHR	✓		✓			✓	
ST_ForceSFS	✓		✓	✓		✓	✓
ST_Force3D	✓		✓	✓		✓	
ST_Force3DM	✓			✓			
ST_Force3DZ	✓		✓	✓		✓	
ST_Force4D	✓		✓	✓			
ST_ForceCollection	✓		✓	✓		✓	
ST_FrechetDistance	✓						
ST_GMLToSQL	✓				✓		
ST_GeneratePoints	✓						
ST_GeoHash	✓			✓			
ST_GeogFromText							
ST_GeogFromWKB				✓			
ST_GeographyFromText							



Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_GeomCollFromText					✓		
ST_GeomFromEWKB			✓	✓		✓	✓
ST_GeomFromEWKT			✓	✓		✓	✓
ST_GeomFromGML			✓			✓	✓
ST_GeomFromGeoHash							
ST_GeomFromGeo✓ N			✓				
ST_GeomFromKML			✓				
ST_GeomFromTWKB							
ST_GeomFromText				✓	✓		
ST_GeomFromWK✓	✓			✓	✓		
ST_GeometricMedi✓	✓		✓				
ST_GeometryFromText					✓		
ST_GeometryN			✓	✓	✓	✓	✓
ST_GeometryType			✓		✓	✓	
>>	✓						
<<	✓						
~	✓						
@	✓						
=	✓	✓		✓		✓	
<<	✓						
&>	✓						
&<	✓			✓		✓	
&<	✓						
&>	✓						
>>	✓						
~=	✓					✓	
ST_HasArc	✓		✓	✓			
ST_HausdorffDistar✓	✓						
ST_InteriorRingN			✓		✓		
ST_InterpolatePoint✓	✓		✓				
ST_Intersection	✓	😄			📄		
ST_Intersects	✓	✓			📄		
ST_IsClosed			✓	✓	✓	✓	
ST_IsCollection			✓	✓			

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_IsEmpty				✓	✓		
ST_IsPlanar							
ST_IsPolygonCCW	✓		✓				
ST_IsPolygonCW	✓		✓				
ST_IsRing					✓		
ST_IsSimple			✓		✓		
ST_IsSolid							
ST_IsValid					✓		
ST_IsValidDetail	✓						
ST_IsValidReason							
ST_IsValidTrajectory	✓		✓				
ST_Length	✓	✓					
ST_Length2D	✓						
ST_Length2D_Spheroid	✓						
ST_LengthSpheroid	✓		✓				
ST_LineCrossingDirection	✓						
ST_LineFromEncodedPolyline							
ST_LineFromMultiPoint			✓				
ST_LineFromText					✓		
ST_LineFromWKB					✓		
ST_LineInterpolatePoint			✓				
ST_LineInterpolatePoints	✓		✓				
ST_LineLocatePoint							
ST_LineMerge	✓						
ST_LineSubstring			✓				
ST_LineToCurve	✓		✓	✓			
ST_LinestringFromWKB	✓				✓		
ST_LocateAlong							
ST_LocateBetween							
ST_LocateBetweenElevations			✓				
ST_LongestLine	✓						
ST_M			✓		✓		
ST_MLineFromText					✓		
ST_MPointFromText	✓				✓		
ST_MPolyFromText					✓		

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_MakeBox2D							
ST_MakeEnvelope	✓						
ST_MakeLine	✓		✓				
ST_MakePoint	✓		✓				
ST_MakePointM	✓						
ST_MakePolygon	✓		✓				
ST_MakeSolid							
ST_MakeValid	✓		✓				
ST_MaxDistance	✓						
ST_MemSize	✓		✓	✓		✓	✓
ST_MemUnion	✓		✓				
ST_MinimumBoundingCircle	✓	Circle					
ST_MinimumBoundingRadius	✓	Radius					
ST_MinimumClearance	✓						
ST_MinimumClearanceLine	✓	Line					
ST_MinkowskiSum							
ST_Multi	✓						
ST_NDims			✓				
ST_NPoints			✓	✓		✓	
ST_NRings			✓	✓			
ST_Node	✓		✓				
ST_Normalize	✓						
ST_NumGeometries			✓		✓	✓	✓
ST_NumInteriorRing							
ST_NumInteriorRings					✓		
ST_NumPatches			✓		✓	✓	
ST_NumPoints					✓		
ST_OffsetCurve	✓						
ST_OrderingEquals	✓				✓		
ST_Orientation							
ST_OrientedEnvelope	✓						
ST_Overlaps	✓				✓		
ST_PatchN			✓		✓	✓	

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Perimeter	✓	✓			✓		
ST_Perimeter2D	✓						
ST_Point					✓		
ST_PointFromGeoHash							
ST_PointFromText	✓				✓		
ST_PointFromWKB	✓		✓	✓	✓		
ST_PointN			✓	✓	✓		
ST_PointOnSurface	✓		✓		✓		
ST_PointInsideCircle	✓						
ST_Points			✓	✓			
ST_Polygon	✓		✓		✓		
ST_PolygonFromText	✓				✓		
ST_Polygonize	✓						
ST_Project		✓					
ST_QuantizeCoordinates	✓						
ST_Relate	✓				✓		
ST_RelateMatch							
ST_RemovePoint	✓		✓				
ST_RemoveRepeatedPoints	✓		✓			✓	
ST_Reverse	✓		✓			✓	
ST_Rotate	✓		✓	✓		✓	✓
ST_RotateX	✓		✓			✓	✓
ST_RotateY	✓		✓			✓	✓
ST_RotateZ	✓		✓	✓		✓	✓
ST_SRID				✓	✓		
ST_Scale	✓		✓	✓		✓	✓
ST_Segmentize	✓	✓					
ST_SetEffectiveArea	✓						
ST_SetPoint	✓		✓				
ST_SetSRID	✓			✓			
ST_SharedPaths	✓						
ST_ShiftLongitude	✓		✓			✓	✓
ST_ShortestLine	✓						
ST_Simplify	✓						
ST_SimplifyPreserveTopology	✓						

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_SimplifyVW	✓						
ST_Snap	✓						
ST_SnapToGrid	✓		✓				
ST_Split	✓						
ST_StartPoint			✓	✓	✓		
ST_StraightSkeleton							
ST_Subdivide	✓						
ST_Summary				✓		✓	✓
ST_SwapOrdinates	✓		✓	✓		✓	✓
ST_SymDifference	✓		✓		✓		
ST_Tessellate							
ST_Touches	✓				✓		
ST_TransScale	✓		✓	✓			
ST_Transform	✓			✓	✓	✓	
ST_Translate	✓		✓	✓			
ST_UnaryUnion	✓		✓				
ST_Union	✓				✓		
ST_Volume							
ST_Voronoi	✓						
ST_Voronoi	✓						
ST_WKBToSQL	✓				✓		
ST_WKTToSQL					✓		
ST_Within	✓				✓		
ST_WrapX	✓		✓				
ST_X			✓		✓		
ST_XMax			✓	✓			
ST_XMin			✓	✓			
ST_Y			✓		✓		
ST_YMax			✓	✓			
ST_YMin			✓	✓			
ST_Z			✓		✓		
ST_ZMax			✓	✓			
ST_ZMin			✓	✓			

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Zmflag			✓	✓			
~(box2df,box2df)				✓		✓	
~(box2df,geometry)	✓			✓		✓	
~(geometry,box2df)	✓			✓		✓	
<#>	✓						
<<#>>	✓						
<<->>	✓						
=	✓						
<->	✓	✓					
&&	✓	✓		✓		✓	
&&&	✓		✓	✓		✓	✓
@(box2df,box2df)				✓		✓	
@(box2df,geometry)	✓			✓		✓	
@(geometry,box2df)	✓			✓		✓	
&&(box2df,box2df)				✓		✓	
&&(box2df,geomet)	✓			✓		✓	
&&(geometry,box2d	✓			✓		✓	
&&&(geometry,gid	✓		✓	✓		✓	✓
&&&(gidx,geometr	✓		✓	✓		✓	✓
&&&(gidx,gidx)			✓	✓		✓	✓
postgis.backend							
postgis.enable_outdb_rasters							
postgis.gdal_datapath							
postgis.gdal_enabled_drivers							
postgis_sfcgal_version							

## 14.12 New, Enhanced or changed PostGIS Functions

### 14.12.1 PostGIS Functions new or enhanced in 2.5

The functions given below are PostGIS functions that were added or enhanced.

Functions new in PostGIS 2.5

- **PostGIS\_Extensions\_Upgrade** - Availability: 2.5.0 Upgrades installed postgis packaged extensions (e.g. postgis\_sfcgal, postgis\_topology, postgis\_sfcgal) to latest installed version. Reports full postgis version and build configuration infos after.
- **ST\_Angle** - Availability: 2.5.0 Returns the angle between 3 points, or between 2 vectors (4 points or 2 lines).
- **ST\_AsHexWKB** - Availability: 2.5.0 Return the Well-Known Binary (WKB) in Hex representation of the raster.

- **ST\_BandFileSize** - Availability: 2.5.0 Returns the file size of a band stored in file system. If no bandnum specified, 1 is assumed.
- **ST\_BandFileTimestamp** - Availability: 2.5.0 Returns the file timestamp of a band stored in file system. If no bandnum specified, 1 is assumed.
- **ST\_Grayscale** - Availability: 2.5.0 Creates a new one-8BUI band raster from the source raster and specified bands representing Red, Green and Blue
- **ST\_LineInterpolatePoints** - Availability: 2.5.0 Returns one or more points interpolated along a line.
- **ST\_QuantizeCoordinates** - Availability: 2.5.0 Sets least significant bits of coordinates to zero
- **ST\_RastFromHexWKB** - Availability: 2.5.0 Return a raster value from a Hex representation of Well-Known Binary (WKB) raster.
- **ST\_RastFromWKB** - Availability: 2.5.0 Return a raster value from a Well-Known Binary (WKB) raster.
- **ST\_SetBandIndex** - Availability: 2.5.0 Update the external band number of an out-db band
- **ST\_SetBandPath** - Availability: 2.5.0 Update the external path and band number of an out-db band

#### Functions enhanced in PostGIS 2.5

- **ST\_AsBinary/ST\_AsWKB** - Enhanced: 2.5.0 Addition of ST\_AsWKB Return the Well-Known Binary (WKB) representation of the raster.
- **ST\_BandMetaData** - Enhanced: 2.5.0 to include outdbbandnum, filesize and filetimestamp for outdb rasters. Retorna os metadados básicos para uma banda raster especificada. banda número 1 é assumida se nenhuma for especificada.
- **ST\_Buffer** - Enhanced: 2.5.0 - ST\_Buffer geometry support was enhanced to allow for side buffering specification side=both|left|right. (T) Retorna uma geometria cobrindo todos os pontos com a dada distância da geometria de entrada.
- **ST\_GeomFromGeoJSON** - Enhanced: 2.5.0 can now accept json and jsonb as inputs. Utiliza como entrada uma representação geojson de uma geometria e como saída um objeto de geometria PostGIS
- **ST\_GeometricMedian** - Enhanced: 2.5.0 Added support for M as weight of points. Retorna a mediana de um MultiPonto.
- **ST\_Intersects** - Enhanced: 2.5.0 Supports GEOMETRYCOLLECTION. Retorna VERDADE se as geometrias/geografia 'intersectam espacialmente em 2D" - (dividem qualquer porção de espaço) e FALSO se elas não (estão disjuntas). Para geografia -- a tolerância é de 0.00001 metros (então quaisquer pontos que estão mais perto estão intersectando)
- **ST\_Subdivide** - Enhanced: 2.5.0 reuses existing points on polygon split, vertex count is lowered from 8 to 5. Retorna um conjunto de geometrias, no qual nenhuma geometria no conjunto tem mais vértices que o número especificado.

#### Functions changed in PostGIS 2.5

- **ST\_GDALDrivers** - Changed: 2.5.0 - add can\_read and can\_write columns. Returns a list of raster formats supported by PostGIS through GDAL. Only those formats with can\_write=True can be used by ST\_AsGDALRaster

### 14.12.2 PostGIS Functions new or enhanced in 2.4

The functions given below are PostGIS functions that were added or enhanced.

#### Functions new in PostGIS 2.4

- **ST\_AsGeobuf** - Availability: 2.4.0 Return a Geobuf representation of a set of rows.
- **ST\_AsMVT** - Availability: 2.4.0 Return a Mapbox Vector Tile representation of a set of rows.
- **ST\_AsMVTGeom** - Availability: 2.4.0 Transform a geometry into the coordinate space of a Mapbox Vector Tile.

- **ST\_Centroid** - Availability: 2.4.0 support for geography was introduced. Retorna o centro geométrico de uma geometria.
- **ST\_FrechetDistance** - Availability: 2.4.0 - requires GEOS >= 3.7.0 Returns the Fréchet distance between two geometries. This is a measure of similarity between curves that takes into account the location and ordering of the points along the curves. Units are in the units of the spatial reference system of the geometries.
- **ST\_MakeEmptyCoverage** - Availability: 2.4.0 Cover georeferenced area with a grid of empty raster tiles.

#### Functions enhanced in PostGIS 2.4

All aggregates now marked as parallel safe which should allow them to be used in plans that can employ parallelism.

PostGIS 2.4.1 `postgis_tiger_geocoder` set to load Tiger 2017 data. Can optionally load zip code 5-digit tabulation (zcta) as part of the **Loader\_Generate\_Nation\_Script**.

- **Loader\_Generate\_Nation\_Script** - Enhanced: 2.4.1 zip code 5 tabulation area (zcta5) load step was fixed and when enabled, zcta5 data is loaded as a single table called `zcta5_all` as part of the nation script load. Gerar uma script shell para a plataforma especificada que carrega as lookup tables de condado e estado.
- **Normalize\_Address** - Enhanced: 2.4.0 `norm_addy` object includes additional fields `zip4` and `address_alphanumeric`. Dado um endereço em texto de uma rua, retorna um tipo composto `norm_addy` que não tem um sufixo, prefixo e tipo padronizado, rua, nome de rua etc. quebrado e, campos separados. Essa função irá funcionar com os dados lookup compactados com o `tiger_geocoder` (dados do censo tiger não são necessários).
- **Page\_Normalize\_Address** - Enhanced: 2.4.0 `norm_addy` object includes additional fields `zip4` and `address_alphanumeric`. Dado um endereço em texto de uma rua, retorna um tipo composto `norm_addy` que não tem um sufixo, prefixo e tipo padronizado, rua, nome de rua etc. quebrado e, campos separados. Essa função irá funcionar com os dados lookup compactados com o `tiger_geocoder` (dados do censo tiger não são necessários). Requer a extensão `address_standardizer`.
- **Reverse\_Geocode** - Enhanced: 2.4.1 if optional zcta5 dataset is loaded, the `reverse_geocode` function can resolve to state and zip even if the specific state data is not loaded. Refer to for details on loading zcta5 data. Pega um ponto em um sistema de referência espacial conhecido e retorna um relato que contém um banco de dados de, teoricamente, possíveis endereços e um banco de dados de ruas cruzadas. Se `include_strnum_range = verdade`, inclui o alcance da rua nas ruas cruzadas.
- **ST\_AsTWKB** - Enhanced: 2.4.0 memory and speed improvements. Retorna a geometria como TWKB, também conhecido como "Tiny Well-Known Binary"
- **ST\_Covers** - Enhanced: 2.4.0 Support for polygon in polygon and line in polygon added for geography type Retorna 1 (VERDADE) se nenhum ponto na geometria B estiver fora da geometria A
- **ST\_CurveToLine** - Enhanced: 2.4.0 added support for max-deviation and max-angle tolerance, and for symmetric output. Converte uma CIRCULARSTRING/CURVEPOLYGON para uma LINESTRING/POLYGON
- **ST\_Project** - Enhanced: 2.4.0 Allow negative distance and non-normalized azimuth. Retorna um POINT projetado de um ponto inicial usando uma distância em metros e suportando (azimute) em radianos.
- **ST\_Reverse** - Enhanced: 2.4.0 support for curves was introduced. Retorna a geometria com a ordem dos vértices revertida.

#### Functions changed in PostGIS 2.4

All PostGIS aggregates now marked as parallel safe. This will force a drop and recreate of aggregates during upgrade which may fail if any user views or sql functions rely on PostGIS aggregates.

- **=** - Changed: 2.4.0, in prior versions this was bounding box equality not a geometric equality. If you need bounding box equality, use `ST_Equals` instead. Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B.
- **ST\_Node** - Changed: 2.4.0 this function uses `GEOSNode` internally instead of `GEOSUnaryUnion`. This may cause the resulting linestrings to have a different order and direction compared to Postgis < 2.4. Nodar um conjunto de linestrings.



### 14.12.3 PostGIS Functions new or enhanced in 2.3

The functions given below are PostGIS functions that were added or enhanced.



**Note**

PostGIS 2.3.0: PostgreSQL 9.6+ support for parallel queries.



**Note**

PostGIS 2.3.0: PostGIS extension, all functions schema qualified to reduce issues in database restore.



**Note**

PostGIS 2.3.0: PostgreSQL 9.4+ support for BRIN indexes. Refer to Section [4.6.2](#).



**Note**

PostGIS 2.3.0: Tiger Geocoder upgraded to work with TIGER 2016 data.

Functions new in PostGIS 2.3

- [&&\(geometry,gidx\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a geometry's (cached) n-D bounding box intersects a n-D float precision bounding box (GIDX).
- [&&\(gidx,geometry\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a n-D float precision bounding box (GIDX) intersects a geometry's (cached) n-D bounding box.
- [&&\(gidx,gidx\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if two n-D float precision bounding boxes (GIDX) intersect each other.
- [&&\(box2df,box2df\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if two 2D float precision bounding boxes (BOX2DF) intersect each other.
- [&&\(box2df,geometry\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a 2D float precision bounding box (BOX2DF) intersects a geometry's (cached) 2D bounding box.
- [&&\(geometry,box2df\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a geometry's (cached) 2D bounding box intersects a 2D float precision bounding box (BOX2DF).
- [@\(box2df,box2df\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into another 2D float precision bounding box.
- [@\(box2df,geometry\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into a geometry's 2D bounding box.
- [@\(geometry,box2df\)](#) - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a geometry's 2D bounding box is contained into a 2D float precision bounding box (BOX2DF).
- [ST\\_MakeLine](#) - Availability: 2.3.0 - Support for multipoint input elements was introduced. Cria uma Linestring de ponto, multiponto ou linha das geometrias.

- `~(box2df,box2df)` - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).
- `~(box2df,geometry)` - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a 2D float precision bounding box (BOX2DF) contains a geometry's 2D bonding box.
- `~(geometry,box2df)` - Availability: 2.3.0 support for Block Range INdexes (BRIN) was introduced. Requires PostgreSQL 9.5+. Returns TRUE if a geometry's 2D bonding box contains a 2D float precision bounding box (GIDX).

The functions given below are PostGIS functions that are enhanced in PostGIS 2.3.

- `ST_Segmentize` - Enhanced: 2.3.0 Segmentize geography now uses equal length segments

#### 14.12.4 PostGIS Functions new or enhanced in 2.2

The functions given below are PostGIS functions that were added or enhanced.



##### Note

`postgis_sfcgal` now can be installed as an extension using `CREATE EXTENSION postgis_sfcgal;`



##### Note

PostGIS 2.2.0: Tiger Geocoder upgraded to work with TIGER 2015 data.



##### Note

`address_standardizer`, `address_standardizer_data_us` extensions for standardizing address data refer to Chapter 12 for details.



##### Note

Many functions in topology rewritten as C functions for increased performance.

#### 14.12.5 PostGIS functions breaking changes in 2.2

The functions given below are PostGIS functions that have possibly breaking changes in PostGIS 2.2. If you use any of these, you may need to check your existing code.

- `ST_Equals` - Changed: 2.2.0 Returns true even for invalid geometries if they are binary equal

#### 14.12.6 PostGIS Functions new or enhanced in 2.1

The functions given below are PostGIS functions that were added or enhanced.



##### Note

More Topology performance Improvements. Please refer to Chapter 11 for more details.

**Note**

Bug fixes (particularly with handling of out-of-band rasters), many new functions (often shortening code you have to write to accomplish a common task) and massive speed improvements to raster functionality. Refer to Chapter 9 for more details.

**Note**

PostGIS 2.1.0: Tiger Geocoder upgraded to work with TIGER 2012 census data. `geocode_settings` added for debugging and tweaking rating preferences, loader made less greedy, now only downloads tables to be loaded. PostGIS 2.1.1: Tiger Geocoder upgraded to work with TIGER 2013 data. Please refer to Section 13.1 for more details.

### 14.12.7 PostGIS Functions new, behavior changed, or enhanced in 2.0

The functions given below are PostGIS functions that were added, enhanced, or have Section 14.12.8 breaking changes in 2.0 releases.

New geometry types: TIN and Polyhedral surfaces was introduced in 2.0

**Note**

Greatly improved support for Topology. Please refer to Chapter 11 for more details.

**Note**

In PostGIS 2.0, raster type and raster functionality has been integrated. There are way too many new raster functions to list here and all are new so please refer to Chapter 9 for more details of the raster functions available. Earlier pre-2.0 versions had `raster_columns/raster_overviews` as real tables. These were changed to views before release. Functions such as `ST_AddRasterColumn` were removed and replaced with `AddRasterConstraints`, `DropRasterConstraints` as a result some apps that created raster tables may need changing.

**Note**

Tiger Geocoder upgraded to work with TIGER 2010 census data and now included in the core PostGIS documentation. A reverse geocoder function was also added. Please refer to Section 13.1 for more details.

### 14.12.8 PostGIS Functions changed behavior in 2.0

The functions given below are PostGIS functions that have changed behavior in PostGIS 2.0 and may require application changes.

**Note**

Most deprecated functions have been removed. These are functions that haven't been documented since 1.2 or some internal functions that were never documented. If you are using a function that you don't see documented, it's probably deprecated, about to be deprecated, or internal and should be avoided. If you have applications or tools that rely on deprecated functions, please refer to [?qandaentry] for more details.

**Note**

Bounding boxes of geometries have been changed from float4 to double precision (float8). This has an impact on answers you get using bounding box operators and casting of bounding boxes to geometries. E.g `ST_SetSRID(abbox)` will often return a different more accurate answer in PostGIS 2.0+ than it did in prior versions which may very well slightly change answers to view port queries.

**Note**

The arguments `hasnodata` was replaced with `exclude_nodata_value` which has the same meaning as the older `hasnodata` but clearer in purpose.

---

### **14.12.9 PostGIS Functions new, behavior changed, or enhanced in 1.5**

#### **14.12.10 PostGIS Functions new, behavior changed, or enhanced in 1.4**

The functions given below are PostGIS functions that were introduced or enhanced in the 1.4 release.

#### **14.12.11 PostGIS Functions new in 1.3**

The functions given below are PostGIS functions that were introduced in the 1.3 release.

---

## Chapter 15

# Reporting Problems

### 15.1 Reporting Software Bugs

Reporting bugs effectively is a fundamental way to help PostGIS development. The most effective bug report is that enabling PostGIS developers to reproduce it, so it would ideally contain a script triggering it and every information regarding the environment in which it was detected. Good enough info can be extracted running `SELECT postgis_full_version()` [for postgis] and `SELECT version()` [for postgresql].

If you aren't using the latest release, it's worth taking a look at its [release changelog](#) first, to find out if your bug has already been fixed.

Using the [PostGIS bug tracker](#) will ensure your reports are not discarded, and will keep you informed on its handling process. Before reporting a new bug please query the database to see if it is a known one, and if it is please add any new information you have about it.

You might want to read Simon Tatham's paper about [How to Report Bugs Effectively](#) before filing a new report.

### 15.2 Reporting Documentation Issues

The documentation should accurately reflect the features and behavior of the software. If it doesn't, it could be because of a software bug or because the documentation is in error or deficient.

Documentation issues can also be reported to the [PostGIS bug tracker](#).

If your revision is trivial, just describe it in a new bug tracker issue, being specific about its location in the documentation.

If your changes are more extensive, a Subversion patch is definitely preferred. This is a four step process on Unix (assuming you already have [Subversion](#) installed):

1. Check out a copy of PostGIS' Subversion trunk. On Unix, type:

```
svn checkout http://svn.osgeo.org/postgis/trunk/
```

This will be stored in the directory `./trunk`

2. Make your changes to the documentation with your favorite text editor. On Unix, type (for example):

```
vim trunk/doc/postgis.xml
```

Note that the documentation is written in DocBook XML rather than HTML, so if you are not familiar with it please follow the example of the rest of the documentation.

3. Make a patch file containing the differences from the master copy of the documentation. On Unix, type:

```
svn diff trunk/doc/postgis.xml > doc.patch
```

4. Attach the patch to a new issue in bug tracker.

## Appendix A

# Apêndice

### A.1 Release 2.5.0

Release date: 2018/09/23

If compiling with PostgreSQL+JIT, LLVM  $\geq$  6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.4 - PostgreSQL 12 (in development) GEOS  $\geq$  3.5

#### A.1.1 Novos Recursos

#1847, sfgist 2d and 3d support for PG 11+ (Esteban Zimányi and Arthur Lesuisse from Université Libre de Bruxelles (ULB), Darafei Praliaskouski)

#4056, ST\_FilterByM (Nicklas Avén)

#4050, ST\_ChaikinSmoothing (Nicklas Avén)

#3989, ST\_Buffer single sided option (Stephen Knox)

#3876, ST\_Angle function (Rémi Cura)

#3564, ST\_LineInterpolatePoints (Dan Baston)

#3896, PostGIS\_Extensions\_Upgrade() (Regina Obe)

#3913, Upgrade when creating extension from unpackaged (Sandro Santilli)

#2256, \_postgis\_index\_extent() for extent from index (Paul Ramsey)

#3176, Add ST\_OrientedEnvelope (Dan Baston)

#4029, Add ST\_QuantizeCoordinates (Dan Baston)

#4063, Optional false origin point for ST\_Scale (Paul Ramsey)

#4082, Add ST\_BandFileSize and ST\_BandFileTimestamp, extend ST\_BandMetadata (Even Rouault)

#2597, Add ST\_Grayscale (Bborie Park)

#4007, Add ST\_SetBandPath (Bborie Park)

#4008, Add ST\_SetBandIndex (Bborie Park)

## A.1.2 Breaking Changes

Upgrade scripts from multiple old versions are now all symlinks to a single upgrade script (Sandro Santilli)

#3944, Update to EPSG register v9.2 (Even Rouault)

#3927, Parallel implementation of ST\_AsMVT

#3925, Simplify geometry using map grid cell size before generating MVT

#3899, BTree sort order is now defined on collections of EMPTY and same-prefix geometries (Darafei Praliaskouski)

#3864, Performance improvement for sorting POINT geometries (Darafei Praliaskouski)

#3900, GCC warnings fixed, make -j is now working (Darafei Praliaskouski) - TopoGeo\_addLinestring robustness improvements (Sandro Santilli) #1855, #1946, #3718, #3838

#3234, Do not accept EMPTY points as topology nodes (Sandro Santilli)

#1014, Hashable geometry, allowing direct use in CTE signatures (Paul Ramsey)

#3097, Really allow MULTILINESTRING blades in ST\_Split() (Paul Ramsey)

#3942, geojson: Do not include private header for json-c >= 0.13 (Björn Esser)

#3954, ST\_GeometricMedian now supports point weights (Darafei Praliaskouski)

#3965, #3971, #3977, #4071 ST\_ClusterKMeans rewritten: better initialization, faster convergence, K=2 even faster (Darafei Praliaskouski)

#3982, ST\_AsEncodedPolyline supports LINESTRING EMPTY and MULTIPOINT EMPTY (Darafei Praliaskouski)

#3986, ST\_AsText now has second argument to limit decimal digits (Marc Ducobu, Darafei Praliaskouski)

#4020, Casting from box3d to geometry now returns correctly connected PolyhedralSurface (Matthias Bay)

#2508, ST\_OffsetCurve now works with collections (Darafei Praliaskouski)

#4006, ST\_GeomFromGeoJSON support for json and jsonb as input (Paul Ramsey, Regina Obe)

#4038, ST\_Subdivide now selects pivot for geometry split that reuses input vertices. (Darafei Praliaskouski)

#4025, #4032 Fixed precision issue in ST\_ClosestPointOfApproach, ST\_DistanceCPA, and ST\_CPAWithin (Paul Ramsey, Darafei Praliaskouski)

#4076, Reduce use of GEOS in topology implementation (Björn Harrtell)

#4080, Add external raster band index to ST\_BandMetaData - Add Raster Tips section to Documentation for information about Raster behavior (e.g. Out-DB performance, maximum open files)

#4084: Fixed wrong code-comment regarding front/back of BOX3D (Matthias Bay)

#4060, #4094, PostgreSQL JIT support (Raúl Marín, Laurenz Albe)

#3960, ST\_Centroid now uses lwgeom\_centroid (Darafei Praliaskouski)

#4027, Remove duplicated code in lwgeom\_geos (Darafei Praliaskouski, Daniel Baston)

#4115, Fix a bug that created MVTs with incorrect property values under parallel plans (Raúl Marín).

#4120, ST\_AsMVTGeom: Clip using tile coordinates (Raúl Marín).

#4132, ST\_Intersection on Raster now works without throwing TopologyException (Vinícius A.B. Schmidt, Darafei Praliaskouski)

#4177, #4180 Support for PostgreSQL 12 dev branch (Laurenz Albe, Raúl Marín)

#4156, ST\_ChaikinSmoothing: also smooth start/end point of polygon by default (Darafei Praliaskouski)

## A.2 Release 2.4.4

Release date: 2018/04/08

Esta é uma versão de correção de bugs e melhoria de desempenho.

## A.2.1 Correção de Erros

- #3055, [raster] ST\_Clip() on a raster without band crashes the server (Regina Obe)
- #3942, geojson: Do not include private header for json-c >= 0.13 (Björn Esser)
- #3952, ST\_Transform fails in parallel mode (Paul Ramsey)
- #3978, Fix KNN when upgrading from 2.1 or older (Sandro Santilli)
- #4003, lwpoly\_construct\_circle: Avoid division by zero (Raúl Marín Rodríguez)
- #4004, Avoid memory exhaustion when building a btree index (Edmund Horner)
- #4016, proj 5.0.0 support (Raúl Marín Rodríguez)
- #4017, lwgeom lexer memory corruption (Peter E)
- #4020, Casting from box3d to geometry now returns correctly connected PolyhedralSurface (Matthias Bay)
- #4025, #4032 Incorrect answers for temporally "almost overlapping" ranges (Paul Ramsey, Darafei Praliaskouski)
- #4052, schema qualify several functions in geography (Regina Obe)
- #4055, ST\_ClusterIntersecting drops SRID (Daniel Baston)

## A.2.2 Melhorias

- #3946, Compile support for PostgreSQL 11 (Paul Ramsey)
- #3992, Use PKG\_PROG\_PKG\_CONFIG macro from pkg.m4 to detect pkg-config (Bas Couwenberg)
- #4044, Upgrade support for PostgreSQL 11 (Regina Obe)

## A.3 Release 2.4.3

Release date: 2018/01/17

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.3.1 Bug Fixes and Enhancements

- #3713, Support encodings that happen to output a '\ ' character
- #3827, Set configure default to not do interrupt testing, was causing false negatives for many people. (Regina Obe) revised to be standards compliant in #3988 (Greg Troxel)
- #3930, Minimum bounding circle issues on 32-bit platforms
- #3965, ST\_ClusterKMeans used to lose some clusters on initialization (Darafei Praliaskouski)
- #3956, Brin opclass object does not upgrade properly (Sandro Santilli)
- #3982, ST\_AsEncodedPolyline supports LINestring EMPTY and MULTIPOINT EMPTY (Darafei Praliaskouski)
- #3975, ST\_Transform runs query on spatial\_ref\_sys without schema qualification. Was causing restore issues. (Paul Ramsey)

## A.4 Release 2.4.2

Release date: 2017/11/15

Esta é uma versão de correção de bugs e melhoria de desempenho.

---



## A.4.1 Bug Fixes and Enhancements

#3917, Fix zcta5 load

#3667, Fix for bug in geography ST\_Segmentize

#3926, Add missing 2.2.6 and 2.3.4 upgrade paths (Muhammad Usama)

## A.5 Release 2.4.1

Release date: 2017/10/18

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.5.1 Bug Fixes and Enhancements

#3864, Fix memory leaks in BTREE operators

#3869, Fix build with "gold" linker

#3845, Gracefully handle short-measure issue

#3871, Performance tweak for geometry cmp function

#3879, Division by zero in some arc cases

#3878, Single defn of signum in header

#3880, Undefined behaviour in TYPMOD\_GET\_SRID

#3875, Fix undefined behaviour in shift operation

#3864, Performance improvements for b-tree geometry sorts

#3874, lw\_dist2d\_pt\_arc division by zero

#3882, undefined behaviour in zigzag with negative inputs

#3891, undefined behaviour in pointarray\_to\_encoded\_polyline

#3895, throw error on malformed WKB input

#3886, fix rare missing boxes in geometry subdivision

#3907, Allocate enough space for all possible GBOX string outputs (Raúl Marín Rodríguez)

## A.6 Release 2.4.0

Release date: 2017/09/30

### A.6.1 Novos Recursos

#3822, Have postgis\_full\_version() also show and check version of PostgreSQL the scripts were built against (Sandro Santilli)

#2411, curves support in ST\_Reverse (Sandro Santilli)

#2951, ST\_Centroid for geography (Danny Götte)

#3788, Allow postgis\_restore.pl to work on directory-style (-Fd) dumps (Roger Crew)

#3772, Direction agnostic ST\_CurveToLine output (Sandro Santilli / KKGeo)

#2464, ST\_CurveToLine with MaxError tolerance (Sandro Santilli / KKGeo)

---

- #3599, Geobuf output support via ST\_AsGeobuf (Björn Harrtell)
- #3661, Mapbox vector tile output support via ST\_AsMVT (Björn Harrtell / CartoDB)
- #3689, Add orientation checking and forcing functions (Dan Baston)
- #3753, Gist penalty speed improvements for 2D and ND points (Darafei Praliaskouski, Andrey Borodin)
- #3677, ST\_FrechetDistance (Shinichi Sugiyama)
- Most aggregates (raster and geometry), and all stable / immutable (raster and geometry) marked as parallel safe
- #2249, ST\_MakeEmptyCoverage for raster (David Zwarg, ainomieli)
- #3709, Allow signed distance for ST\_Project (Darafei Praliaskouski)
- #524, Covers support for polygon on polygon, line on line, point on line for geography (Danny Götte)

## A.6.2 Enhancements and Fixes

Many corrections to docs and several translations almost complete. Andreas Schild who provided many corrections to core docs. PostGIS Japanese translation team first to reach completion of translation.

Support for PostgreSQL 10

Preliminary support for PostgreSQL 11

- #3645, Avoid loading logically deleted records from shapefiles
- #3747, Add zip4 and address\_alphanumeric as attributes to norm\_addy tiger\_geocoder type.
- #3748, address\_standardizer lookup tables update so page\_normalize\_address better standardizes abbreviations
- #3647, better handling of nodding in ST\_Node using GEOSNode (Wouter Geraedts)
- #3684, Update to EPSG register v9 (Even Rouault)
- #3830, Fix initialization of incompatible type ( >=9.6) address\_standardizer
- #3662, Make shp2pgsql work in debug mode by sending debug to stderr
- #3405, Fixed memory leak in lwgeom\_to\_points
- #3832, Support wide integer fields as int8 in shp2pgsql
- #3841, Deterministic sorting support for empty geometries in btree geography
- #3844, Make = operator a strict equality test, and < > to rough "spatial sorting"
- #3855, ST\_AsTWKB memory and speed improvements

## A.6.3 Breaking Changes

Dropped support for PostgreSQL 9.2.

#3810, GEOS 3.4.0 or above minimum required to compile

Most aggregates now marked as parallel safe, which means most aggs have to be dropped / recreated. If you have views that utilize PostGIS aggs, you'll need to drop before upgrade and recreate after upgrade

#3578, ST\_NumInteriorRings(POLYGON EMPTY) now returns 0 instead of NULL

\_ST\_DumpPoints removed, was no longer needed after PostGIS 2.1.0 when ST\_DumpPoints got reimplemented in C

B-Tree index operators < = > changed to provide better spatial locality on sorting and have expected behavior on GROUP BY. If you have btree index for geometry or geography, you need to REINDEX it, or review if it was created by accident and needs to be replaced with GiST index. If your code relies on old left-to-right box compare ordering, update it to use << >> operators.

## A.7 Release 2.3.3

Release date: 2017/07/01

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.7.1 Bug Fixes and Enhancements

#3777, GROUP BY anomaly with empty geometries

#3711, Azimuth error upon adding 2.5D edges to topology

#3726, PDF manual from dblatex renders fancy quotes for programlisting (Mike Toews)

#3738, raster: Using -s without -Y in raster2pgsql transforms raster data instead of setting srid

#3744, ST\_Subdivide loses subparts of inverted geometries (Darafei Praliaskouski Komzpa)

#3750, @ and ~ operator not always schema qualified in geometry and raster functions. Causes restore issues. (Shane StClair of Axiom Data Science)

#3682, Strange fieldlength for boolean in result of pgsq2shp

#3701, Escape double quotes issue in pgsq2shp

#3704, ST\_AsX3D crashes on empty geometry

#3730, Change ST\_Clip from Error to Notice when ST\_Clip can't compute a band

## A.8 Release 2.3.2

Release date: 2017/01/31

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.8.1 Bug Fixes and Enhancements

#3418, KNN recheck in 9.5+ fails with index returned tuples in wrong order

#3675, Relationship functions not using an index in some cases

#3680, PostGIS upgrade scripts missing GRANT for views

#3683, Unable to update postgis after postgres pg\_upgrade going from < 9.5 to pg > 9.4

#3688, ST\_AsLatLonText: round minutes

## A.9 Release 2.3.1

Release date: 2016/11/28

Esta é uma versão de correção de bugs e melhoria de desempenho.

---

## A.9.1 Bug Fixes and Enhancements

- #1973, st\_concavehull() returns sometimes empty geometry collection Fix from gde
- #3501, add raster constraint max extent exceeds array size limit for large tables
- #3643, PostGIS not building on latest OSX XCode
- #3644, Deadlock on interrupt
- #3650, Mark ST\_Extent, ST\_3DExtent and ST\_Mem\* agg functions as parallel safe so they can be parallelized
- #3652, Crash on Collection(MultiCurve())
- #3656, Fix upgrade of aggregates from 2.2 or lower version
- #3659, Crash caused by raster GUC define after CREATE EXTENSION using wrong memory context. (manaem)
- #3665, Index corruption and memory leak in BRIN indexes patch from Julien Rouhaud (Dalibo)
- #3667, geography ST\_Segmentize bug patch from Hugo Mercier (Oslandia)

## A.10 Release 2.3.0

Release date: 2016/09/26

This is a new feature release, with new functions, improved performance, all relevant bug fixes from PostGIS 2.2.3, and other goodies.

### A.10.1 Importante / Mudanças Críticas

- #3466, Casting from box3d to geometry now returns a 3D geometry (Julien Rouhaud of Dalibo)
- #3396, ST\_EstimatedExtent, throw WARNING instead of ERROR (Regina Obe)

### A.10.2 Novos Recursos

- Add support for custom TOC in postgis\_restore.pl (Christoph Moench-Tegeder)
- Add support for negative indexing in ST\_PointN and ST\_SetPoint (Rémi Cura)
- Add parameters for geography ST\_Buffer (Thomas Bonfort)
- TopoGeom\_addElement, TopoGeom\_remElement (Sandro Santilli)
- populate\_topology\_layer (Sandro Santilli)
- #454, ST\_WrapX and lwgeom\_wrapx (Sandro Santilli)
- #1758, ST\_Normalize (Sandro Santilli)
- #2236, shp2pgsql -d now emits "DROP TABLE IF EXISTS"
- #2259, ST\_VoronoiPolygons and ST\_VoronoiLines (Dan Baston)
- #2841 and #2996, ST\_MinimumBoundingRadius and new ST\_MinimumBoundingCircle implementation using Welzl's algorithm (Dan Baston)
- #2991, Enable ST\_Transform to use PROJ.4 text (Mike Toews)
- #3059, Allow passing per-dimension parameters in ST\_Expand (Dan Baston)
- #3339, ST\_GeneratePoints (Paul Ramsey)
- #3362, ST\_ClusterDBSCAN (Dan Baston)
- #3364, ST\_GeometricMedian (Dan Baston)

- #3391, Add table inheritance support in ST\_EstimatedExtent (Alessandro Pasotti)
- #3424, ST\_MinimumClearance (Dan Baston)
- #3428, ST\_Points (Dan Baston)
- #3465, ST\_ClusterKMeans (Paul Ramsey)
- #3469, ST\_MakeLine with MULTIPOINTS (Paul Norman)
- #3549, Support PostgreSQL 9.6 parallel query mode, as far as possible (Paul Ramsey, Regina Obe)
- #3557, Geometry function costs based on query stats (Paul Norman)
- #3591, Add support for BRIN indexes. PostgreSQL 9.4+ required. (Giuseppe Broccolo of 2nd Quadrant, Julien Rouhaud and Ronan Dunklau of Dalibo)
- #3496, Make postgis non-relocateable for extension install, schema qualify calls in functions (Regina Obe) Should resolve once and for all for extensions #3494, #3486, #3076
- #3547, Update tiger geocoder to support TIGER 2016 and to support both http and ftp.
- #3613, Segmentize geography using equal length segments (Hugo Mercier of Oslandia)

### A.10.3 Correção de Erros

All relevant bug fixes from PostGIS 2.2.3

- #2841, ST\_MinimumBoundingCircle not covering original
- #3604, pgcommon/Makefile.in orders CFLAGS incorrectly leading to wrong liblwgeom.h (Greg Troxel)

### A.10.4 Melhorias de Desempenho

- #75, Enhancement to PIP short circuit (Dan Baston)
- #3383, Avoid deserializing small geometries during index operations (Dan Baston)
- #3400, Minor optimization of PIP routines (Dan Baston)
- Make adding a line to topology interruptible (Sandro Santilli)
- Documentation updates from Mike Toews

## A.11 Release 2.2.2

Release date: 2016/03/22

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.11.1 Novos Recursos

- #3463, Fix crash on face-collapsing edge change
  - #3422, Improve ST\_Split robustness on standard precision double systems (arm64, ppc64el, s390c, powerpc, ...)
  - #3427, Update spatial\_ref\_sys to EPSG version 8.8
  - #3433, ST\_ClusterIntersecting incorrect for MultiPoints
  - #3435, ST\_AsX3D fix rendering of concave geometries
  - #3436, memory handling mistake in ptdarray\_clone\_deep
  - #3437, ST\_Intersects incorrect for MultiPoints
-

- #3461, ST\_GeomFromKML crashes Postgres when there are innerBoundaryIs and no outerBoundaryIs
- #3429, upgrading to 2.3 or from 2.1 can cause loop/hang on some platforms
- #3460, ST\_ClusterWithin 'Tolerance not defined' error after upgrade
- #3490, Raster data restore issues, materialized views. Scripts postgis\_proc\_set\_search\_path.sql, rtpostgis\_proc\_set\_search\_path.sql refer to [http://postgis.net/docs/manual-2.2/RT\\_FAQ.html#faq\\_raster\\_data\\_not\\_restore](http://postgis.net/docs/manual-2.2/RT_FAQ.html#faq_raster_data_not_restore)
- #3426, failing POINT EMPTY tests on fun architectures

## A.12 Release 2.2.1

Data de lançamento: 06/01/2016

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.12.1 Novos Recursos

- #2232, evitar erros acumulados no arredondamento em SVG
  - #3321, correção de uma regressão de desempenho na carga de topologia
  - #3329, correção em uma regressão de robustez em TopoGeo\_addPoint
  - #3349, Conserta caminho de instalação das scripts postgis\_topology
  - #3351, isolamento de nós finais na ST\_RemoveIsoEdge (e lwt\_RemIsoEdge)
  - #3355, geografia ST\_Segmentize tem bbox geometria
  - #3359, Correção na perda toTopoGeom de baixa-id da definição TopoGeometry
  - #3360, \_raster\_constraint\_info\_scale sintaxe de entrada inválida
  - #3375, quebra em pontos repetidos remoção para coleção(ponto)
  - #3378, Correção na superação de TopoGeometries hierárquicas na presença de várias topologias
  - #3380, #3402, Dizima linhas no carregamento da topologia
  - #3388, #3410, Correção nos pontos finais faltado na ST\_Removepoints
  - #3389, Buffer overflow em lwgeom\_to\_geojson
  - #3390, Erro na compilação no Alpine Linux 3.2, no momento de compilar o postgis e a extensão postgis\_topology
  - #3393, ST\_Area como NaN para alguns polígonos
  - #3401, Melhora a robustez ST\_Split em sistemas 32bit
  - #3404, ST\_ClusterWithin quebra o backend
  - #3407, Correção na divisão ou quebra de uma face ou um limite definindo vários objetos TopoGeometry
  - #3411, Funções clustering não usando o índice espacial
  - #3412, Melhora a robustez do passo snapping na TopoGeo\_addLinestring
  - #3415, Correção OSX 10.9 construído de baixo de pkgsrc
- Correção no vazamento de memória em lwt\_ChangeEdgeGeom [liblwgeom]

## A.13 Versão 2.2.0

Data de lançamento: 2015/10/07

Esta é uma nova característica lançada, com novas funções, desenvolvimento melhorado e outras melhorias.

### A.13.1 Novos Recursos

Topologia API in liblwgeom (Sandro Santilli / Regione Toscana - SITA)

Novo método lwgeom\_unaryunion em liblwgeom

Novo método lwgeom\_linemerge em liblwgeom

Novo método lwgeom\_is\_simple em liblwgeom

**#3169**, Add SFCGAL 1.1 support: add ST\_3DDifference, ST\_3DUnion, ST\_Volume, ST\_MakeSolid, ST\_IsSolid (Vincent Mora / Oslandia)

**#3169**, ST\_ApproximateMedialAxis (Sandro Santilli)

ST\_CPAWithin (Sandro Santilli / Boundless)

Add |=| operator with CPA semantic and KNN support with PostgreSQL 9.5+ (Sandro Santilli / Boundless)

**#3131**, KNN support for the geography type (Paul Ramsey / CartoDB)

**#3023**, ST\_ClusterIntersecting / ST\_ClusterWithin (Dan Baston)

**#2703**, Exact KNN results for all geometry types, aka "KNN re-check" (Paul Ramsey / CartoDB)

**#1137**, Allow a tolerance value in ST\_RemoveRepeatedPoints (Paul Ramsey / CartoDB)

**#3062**, Allow passing M factor to ST\_Scale (Sandro Santilli / Boundless)

**#3139**, ST\_BoundingDiagonal (Sandro Santilli / Boundless)

**#3129**, ST\_IsValidTrajectory (Sandro Santilli / Boundless)

**#3128**, ST\_ClosestPointOfApproach (Sandro Santilli / Boundless)

**#3152**, ST\_DistanceCPA (Sandro Santilli / Boundless)

Saída canônica para índice de tipos de chaves

ST\_SwapOrdinates (Sandro Santilli / Boundless)

**#2918**, Use GeographicLib functions for geodetics (Mike Toews)

**#3074**, ST\_Subdivide to break up large geometry (Paul Ramsey / CartoDB)

**#3040**, KNN GiST index based centroid (<<<->>) n-D distance operators (Sandro Santilli / Boundless)

Interruptibilidade API for liblwgeom (Sandro Santilli / CartoDB)

**#2939**, ST\_ClipByBox2D (Sandro Santilli / CartoDB)

**#2247**, ST\_Retile and ST\_CreateOverview: in-db raster overviews creation (Sandro Santilli / Vizzuality)

**#899**, -m shp2pgsql attribute names mapping -m switch (Regina Obe / Sandro Santilli)

**#1678**, Adicionado GUC postgres.gdal\_datapath para especificar a variável GDAL de configuração GDAL\_DATA

**#2843**, Suporta reprojeção na importação raster (Sandro Santilli / Vizzuality)

**#2349**, Suporte para encoded\_polyline input/output (Kashif Rasul)

**#2159**, report libjson version from postgres\_full\_version()

**#2770**, ST\_MemSize(raster)

Adiciona postgres\_noop(raster)

Adicionadas as variantes faltando das ST\_TPI(), ST\_TRI() e ST\_Roughness()

Adicionado GUC postgres.gdal\_enabled\_drivers para especificar a variável GDAL de configuração GDAL\_SKIP

Adicionado GUC postgres.enable\_outdb\_rasters para ativar o acesso para rasters com out-db bands

**#2387**, address\_standardizer extension as part of PostGIS (Stephen Woodbridge / imaptools.com, Walter Sinclair, Regina Obe)

- #2816, address\_standardizer\_data\_us extension provides reference lex,gaz,rules for address\_standardizer (Stephen Woodbridge / imaptools.com, Walter Sinclair, Regina Obe)
- #2341, New mask parameter for ST\_MapAlgebra
- #2397, read encoding info automatically in shapefile loader
- #2430, ST\_ForceCurve
- #2565, ST\_SummaryStatsAgg()
- #2567, ST\_CountAgg()
- #2632, ST\_AsGML() suporte para características curvas
- #2652, Add --upgrade-path switch to run\_test.pl
- #2754, sfcgal envolvida como uma extensão
- #2227, Simplificação com o algoritmo Visvalingam-Whyatt ST\_SimplifyVW, ST\_SetEffectiveArea (Nicklas Avén)
- Functions to encode and decode TWKB ST\_AsTWKB, ST\_GeomFromTWKB (Paul Ramsey / Nicklas Avén / CartoDB)

### A.13.2 Melhorias

- #3223, Add memcmp short-circuit to ST\_Equals (Daniel Baston)
- #3227, Tiger geocoder atualizado para suportar o censo Tiger 2015
- #2278, Torna o liblwgeom compatível entre liberações menores
- #897, ST\_AsX3D support for GeoCoordinates and systems "GD" "WE" ability to flip x/y axis (use option = 2, 3)
- ST\_Split: permite dividir linhas por limites de multilinhas, multipontos e (multi)polígonos
- #3070, Simplifica a restrição de tipo de geometria
- #2839, Implementa estimador seletivo para índices funcionais, aumentando a velocidade de consultas espaciais em tabelas raster. (Sandro Santilli / Vizzuality)
- #2361, Added spatial\_index column to raster\_columns view
- #2390, Suíte de teste para pgsq2shp
- #2527, Bandeira -k adicionada a raster2pgsql para pular a verificação de que a banda é NODATA
- #2616, Reduz os casts de texto durante a construção e exportação de topologia
- #2717, suporta ponto inicial, ponto final, ponto n, numpoints para curvas compostas
- #2747, Adiciona suporte para GDAL 2.0
- #2754, SFCGAL agora pode ser instalado com CREATE EXTENSION (Vincent Mora @ Oslandia)
- #2828, Converte ST\_Envelope(raster) de SQL para C
- #2829, Shortcut ST\_Clip(raster) se a geometria contém o raster completamente e NODATA especificado
- #2906, Update tiger geocoder para lidar com dados tiger 2014
- #3048, Acelera a simplificação de geometria (J.Santana @ CartoDB)
- #3092, Diminui a performance das geometry\_columns com várias tabelas

## A.14 Versão 2.1.8

Data de lançamento: 2015-07-07

Esta é uma correção de bug decisiva.



### A.14.1 Correção de Erros

#3159, não força uma bbox salvar na ST\_Affine  
#3018, GROUP BY geografia algumas vezes retorna linhas duplicadas  
#3084, shp2pgsql - illegal number format when specific system locale set  
#3094, Malformado GeoJSON fornece backend quebrado  
#3104, st\_asgml introduz caracteres aleatórios no campo ID  
#3155, Remove liblwgeom.h em fazer desinstalação  
#3177, gserialized\_is\_empty não pode lidar com casos encaixados vazios  
Corrige crash em ST\_LineLocatePoint

## A.15 Versão 2.1.7

Data de lançamento: 2015-03-30

Esta é uma correção de bug decisiva.

### A.15.1 Correção de Erros

#3086, ST\_DumpValues() quebra backend na limpeza com índices de banda inválidos  
#3088, Não (re)define strcasestr em um liblwgeom.h  
#3094, Malformado GeoJSON fornece backend quebrado

## A.16 Versão 2.1.6

Data de lançamento: 2015-03-20

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.16.1 Melhorias

#3000, Ensure edge splitting and healing algorithms use indexes  
#3048, Acelera simplificação de geometria (J.Santana @ CartoDB)  
#3050, Acelera o leitor de tipo de geometria (J.Santana @ CartoDB)

### A.16.2 Correção de Erros

#2941, permite colunas de geografia com SRID em vez de 4326  
#3069, pequenos objetos tornando-se caixas inapropriadamente fluffed up  
#3068, Ter postgis\_typmod\_dims retorna NULL para dims sem restrições  
#3061, Permite pontos duplicados nas funções JSON, GML, GML ST\_GeomFrom\*  
#3058, Correção ND-GiST método picksplit para dividir no melhor plano  
#3052, Torna os operadores <-> and <#> disponíveis para o PostgreSQL < 9.1  
#3045, Correção da confusão de dimensionalidade no operador &&&

- #3016, Permite cancelar o registro de camadas de topologias corrompidas
- #3015, Evita exceções do TopologySummary
- #3020, ST\_AddBand out-db bug where height using width value
- #3031, Permite restauração das tabelas Geometry(Point) descartadas com vazias nelas

## A.17 Versão 2.1.5

Data de lançamento: 2014-12-18

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.17.1 Melhorias

- #2933, Acelera a construção de grandes objetos multi geometria

### A.17.2 Correção de Erros

- #2947, Correção no vazamento de memória em lwgeom\_make\_valid para entrada de coleção de um único componente
- #2949, Correção no vazamento de memória em lwgeom\_mindistance2d para entrada de curva
- #2931, Representação de CAIXA em caso sensível
- #2942, Suporte PostgreSQL 9.5
- #2953, Estatísticas 2D não geradas quando os valores Z/M forem extremos
- #3009, Geography cast may effect underlying tuple

## A.18 Versão 2.1.4

Data de lançamento: 2014-09-10

Esta é uma versão de correção de bugs e melhoria de desempenho.

### A.18.1 Melhorias

- #2745, Acelera as chamadas ST\_Simplify contra pontos
  - #2747, Suporte para GDAL 2.0
  - #2749, Faz rtpostgis\_upgrade\_20\_21.sql ACID
  - #2811, Não especifica nomes de índices no carregamento de shapefiles/rasters
  - #2829, Shortcut ST\_Clip(raster) se a geometria contém o raster completamente e NODATA especificado
  - #2895, Aumenta o custo da ST\_ConvexHull(raster) para 300 para planos melhores
-

## A.18.2 Correção de Erros

#2605, armel: `_ST_Covers()` retorna verdade para ponto no buraco

#2911, Correção na escala de saída `ST_Rescale/ST_Resample/ST_Resize` de rasters com escala 1/-1 e compensação 0/0.

Corrigido crash em `ST_Union(raster)`

#2704, `ST_GeomFromGML()` não funciona corretamente com arranjo de `gml:pos` (Even Roualt)

#2708, `updategeometrysrid` não atualiza `srid` check quando o esquema não está especificado. Caminho de Marc Jansen

#2720, `lwpoly_add_ring` should update maxrings after realloc

#2759, Fix `postgis_restore.pl` handling of multiline object comments embedding sql comments

#2774, fix undefined behavior in `ptarray_calculate_gbox_geodetic`

Correção de falta potencial de memória na `ST_MakeValid`

#2784, Fix handling of bogus argument to `--with-sfcgal`

#2772, Premature memory free in `RASTER_getBandPath (ST_BandPath)`

#2755, Fix regressions tests against all versions of SFCGAL

#2775, `lwline_from_lwmpoint` leaks memory

#2802, `ST_MapAlgebra` checks for valid callback function return value

#2803, `ST_MapAlgebra` handles no userarg and STRICT callback function

#2834, `ST_Estimated_Extent` and mixedCase table names (regression bug)

#2845, Bad geometry created from `ST_AddPoint`

#2870, Binary insert into geography column results geometry being inserted

#2872, make install builds documentation (Greg Troxell)

#2819, find isfinite or replacement on Centos5 / Solaris

#2899, geocode limit 1 not returning best answer (tiger geocoder)

#2903, Incapaz de compilar no FreeBSD

#2927 `reverse_geocode` not filling in direction prefix (tiger geocoder) get rid of deprecated `ST_Line_Locate_Point` called

## A.19 Versão 2.1.3

Data de lançamento: 2014/05/13

Esta é uma correção de bug e comunicado seguro.

### A.19.1 Mudanças importantes

Começando com este acesso de versão raster offline e uso dos drivers GDAL estão desativados por padrão.

Uma variável de ambiente é introduzida para permitir a ativação de drivers GDAL específicos: `POSTGIS_GDAL_ENABLED_DRIVERS`. Por padrão, todos os drivers GDAL estão desativados

Uma variável de ambiente é introduzida para permitir a ativação de bandas raster out-db: `POSTGIS_ENABLE_OUTDB_RASTERS`. Por padrão, bandas out-db raster são desativadas

As variáveis de ambiente devem ser configuradas para o processo PostgreSQL, e determina o comportamento do cluster inteiro.

## A.19.2 Correção de Erros

#2697, invalid GeoJSON Polygon input crashes server process

#2700, Fix dumping of higher-dimension datasets with null rows

#2706, ST\_DumpPoints of EMPTY geometries crashes server

## A.20 Versão 2.1.2

Data de Lançamento: 2014/03/31

Este é uma versão apenas de correção de erros, resolvendo questões que foram solicitadas desde a versão 2.1.1.

### A.20.1 Correção de Erros

#2666, Error out at configure time if no SQL preprocessor can be found

#2534, st\_distance returning incorrect results for large geographies

#2539, Check for json-c/json.h presence/usability before json/json.h

#2543, invalid join selectivity error from simple query

#2546, GeoJSON with string coordinates parses incorrectly

#2547, Fix ST\_Simplify(TopoGeometry) for hierarchical topogeoms

#2552, Fix NULL raster handling in ST\_AsPNG, ST\_AsTIFF and ST\_AsJPEG

#2555, Fix parsing issue of range arguments of ST\_Reclass

#2556, geography ST\_Intersects results depending on insert order

#2580, Do not allow installing postgis twice in the same database

#2589, Remove use of unnecessary void pointers

#2607, Cannot open more than 1024 out-db files in one process

#2610, Ensure face splitting algorithm uses the edge index

#2615, EstimatedExtent (and hence, underlying stats) gathering wrong bbox

#2619, Empty rings array in GeoJSON polygon causes crash

#2634, regression in sphere distance code

#2638, Geography distance on M geometries sometimes wrong

#2648, #2653, Fix topology functions when "topology" is not in search\_path

#2654, Drop deprecated calls from topology

#2655, Permite usuários sem privilégios de topologia chamar postgis\_full\_version()

#2674, Correção no operador ausente = e hash\_raster\_ops opclass no raster

#2675, #2534, #2636, #2634, #2638, Issues de distância geográfica com otimização de árvore

### A.20.2 Melhorias

#2494, evita cópia de memória no índice GiST (hayamiz)

#2560, soft upgrade: avoid drop/recreate of aggregates that hadn't changed

## A.21 Versão 2.1.1

Data de lançamento: 2013/11/06

Este é uma versão apenas de correção de erros, resolvendo questões que foram solicitadas desde a versão 2.1.0.

### A.21.1 Mudanças importantes

#2514, Change raster license from GPL v3+ to v2+, allowing distribution of PostGIS Extension as GPLv2.

### A.21.2 Correção de Erros

#2396, Make regression tests more endian-agnostic

#2434, Fix ST\_Intersection(geog,geog) regression in rare cases

#2454, Fix behavior of ST\_PixelAsXXX functions regarding exclude\_nodata\_value parameter

#2489, Fix upgrades from 2.0 leaving stale function signatures

#2525, Fix handling of SRID in nested collections

#2449, Fix potential infinite loop in index building

#2493, Fix behavior of ST\_DumpValues when passed an empty raster

#2502, Fix postgis\_topology\_scripts\_installed() install schema

#2504, Fix segfault on bogus pgsqldshp call

#2512, Support for foreign tables and materialized views in raster\_columns and raster\_overviews

### A.21.3 Melhorias

#2478, support for tiger 2013

#2463, support for exact length calculations on arc geometries

## A.22 Versão 2.1.0

Data de Lançamento: 2013/08/17

This is a minor release addressing both bug fixes and performance and functionality enhancements addressing issues since 2.0.3 release. If you are upgrading from 2.0+, only a soft upgrade is required. If you are upgrading from 1.5 or earlier, a hard upgrade is required.

### A.22.1 Importante / Mudanças Críticas

#1653, Removed srid parameter from ST\_Resample(raster) and variants with reference raster no longer apply reference raster's SRID.

#1962 ST\_Segmentize - As a result of the introduction of geography support, The construct: `SELECT ST_Segmentize('LINESTRING(2 3 4)', 0.5);` will result in ambiguous function error

#2026, ST\_Union(raster) now unions all bands of all rasters

#2089, liblwgeom: lwgeom\_set\_handlers replaces lwgeom\_init\_allocators.

#2150, regular\_blocking is no longer a constraint. column of same name in raster\_columns now checks for existence of spatially\_unique and coverage\_tile constraints

`ST_Intersects(raster, geometria)` se comporta da mesma maneira que `ST_Intersects(geometria, raster)`.

o ponto variante da `ST_SetValue(raster)` não verificava o SRID da geometria de entrada e raster.

Os parâmetros `ST_Hillshade` azimute e altitude agora estão em graus em vez de radianos.

`ST_Slope` e `ST_Aspect` retornam os valores de pixel em graus em vez de radianos.

**#2104**, `ST_World2RasterCoord`, `ST_World2RasterCoordX` and `ST_World2RasterCoordY` renamed to `ST_WorldToRasterCoord`, `ST_WorldToRasterCoordX` and `ST_WorldToRasterCoordY`. `ST_Raster2WorldCoord`, `ST_Raster2WorldCoordX` and `ST_Raster2WorldCoordY` renamed to `ST_RasterToWorldCoord`, `ST_RasterToWorldCoordX` and `ST_RasterToWorldCoordY`

`ST_Estimated_Extent` renomeado para `ST_EstimatedExtent`

`ST_Line_Interpolate_Point` renomeado para `ST_LineInterpolatePoint`

`ST_Line_Substring` renomeado para `ST_LineSubstring`

`ST_Line_Locate_Point` renomeado para `ST_LineLocatePoint`

`ST_Force_XXX` renomeado para `ST_ForceXXX`

`ST_MapAlgebraFctNgb` e 1 e 2 raster variantes de `ST_MapAlgebraFct`. Use `ST_MapAlgebra` ao contrário.

1 e 2 raster variantes de `ST_MapAlgebraExpr`. Use expressão variante de `ST_MapAlgebra` ao contrário

## A.22.2 Novos Recursos

- Refer to [http://postgis.net/docs/manual-2.1/PostGIS\\_Special\\_Functions\\_Index.html#NewFunctions\\_2\\_1](http://postgis.net/docs/manual-2.1/PostGIS_Special_Functions_Index.html#NewFunctions_2_1) for complete list of new functions

**#310**, `ST_DumpPoints` converted to a C function (Nathan Wagner) and much faster

**#739**, `UpdateRasterSRID()`

**#945**, improved join selectivity, N-D selectivity calculations, user accessible selectivity and stats reader functions for testing (Paul Ramsey / OpenGeo)

`toTopoGeom` with `TopoGeometry` sink (Sandro Santilli / Vizzuality)

`clearTopoGeom` (Sandro Santilli / Vizzuality)

`ST_Segmentize(geography)` (Paul Ramsey / OpenGeo)

`ST_DelaunayTriangles` (Sandro Santilli / Vizzuality)

`ST_NearestValue`, `ST_Neighborhood` (Bborie Park / UC Davis)

`ST_PixelAsPoint`, `ST_PixelAsPoints` (Bborie Park / UC Davis)

`ST_PixelAsCentroid`, `ST_PixelAsCentroids` (Bborie Park / UC Davis)

`ST_Raster2WorldCoord`, `ST_World2RasterCoord` (Bborie Park / UC Davis)

Additional raster/raster spatial relationship functions (`ST_Contains`, `ST_ContainsProperly`, `ST_Covers`, `ST_CoveredBy`, `ST_Disjoint`, `ST_Overlaps`, `ST_Touches`, `ST_Within`, `ST_DWithin`, `ST_DFullyWithin`) (Bborie Park / UC Davis)

Added array variants of `ST_SetValues()` to set many pixel values of a band in one call (Bborie Park / UC Davis)

**#1293**, `ST_Resize(raster)` to resize rasters based upon width/height

**#1627**, package `tiger_geocoder` as a PostgreSQL extension

**#1643**, **#2076**, Upgrade tiger geocoder to support loading tiger 2011 and 2012 (Regina Obe / Paragon Corporation) Funded by Hunter Systems Group

GEOMETRYCOLLECTION support for `ST_MakeValid` (Sandro Santilli / Vizzuality)

**#1709**, `ST_NotSameAlignmentReason(raster, raster)`

**#1818**, `ST_GeomFromGeoHash` and friends (Jason Smith (darkpanda))

#1856, reverse geocoder rating setting for prefer numbered highway name

ST\_PixelOfValue (Bborie Park / UC Davis)

Casts to/from PostgreSQL geotypes (point/path/polygon).

Added geomval array variant of ST\_SetValues() to set many pixel values of a band using a set of geometries and corresponding values in one call (Bborie Park / UC Davis)

ST\_Tile(raster) to break up a raster into tiles (Bborie Park / UC Davis)

#1895, new r-tree node splitting algorithm (Alex Korotkov)

#2011, ST\_DumpValues to output raster as array (Bborie Park / UC Davis)

#2018, ST\_Distance support for CircularString, CurvePolygon, MultiCurve, MultiSurface, CompoundCurve

#2030, n-raster (and n-band) ST\_MapAlgebra (Bborie Park / UC Davis)

#2193, Utilize PASC parser as drop in replacement for tiger normalizer (Steve Woodbridge, Regina Obe)

#2210, ST\_MinConvexHull(raster)

lwgeom\_from\_geojson in liblwgeom (Sandro Santilli / Vizzuality)

#1687, ST\_Simplify for TopoGeometry (Sandro Santilli / Vizzuality)

#2228, TopoJSON output for TopoGeometry (Sandro Santilli / Vizzuality)

#2123, ST\_FromGDALRaster

#613, ST\_SetGeoReference with numerical parameters instead of text

#2276, ST\_AddBand(raster) variant for out-db bands

#2280, ST\_Summary(raster)

#2163, ST\_TPI for raster (Nathaniel Clay)

#2164, ST\_TRI for raster (Nathaniel Clay)

#2302, ST\_Roughness for raster (Nathaniel Clay)

#2290, ST\_ColorMap(raster) to generate RGBA bands

#2254, Add SFCGAL backend support. (Backend selection through postgres.backend var) Functions available both through GEOS or SFCGAL: ST\_Intersects, ST\_3DIntersects, ST\_Intersection, ST\_Area, ST\_Distance, ST\_3DDistance New functions available only with SFCGAL backend: ST\_3DIntersection, ST\_Tesselate, ST\_3DArea, ST\_Extrude, ST\_ForceLHR ST\_Orientation, ST\_Minkowski, ST\_StraightSkeleton postgres\_sfcgal\_version New function available in PostGIS: ST\_ForceSFS (Olivier Courtin and Hugo Mercier / Oslandia)

### A.22.3 Melhorias

For detail of new functions and function improvements, please refer to Section 14.12.6.

Much faster raster ST\_Union, ST\_Clip and many more function additions operations

For geometry/geography better planner selectivity and a lot more functions.

#823, tiger geocoder: Make loader\_generate\_script download portion less greedy

#826, raster2pgsql no longer defaults to padding tiles. Flag -P can be used to pad tiles

#1363, ST\_AddBand(raster, ...) array version rewritten in C

#1364, ST\_Union(raster, ...) aggregate function rewritten in C

#1655, Additional default values for parameters of ST\_Slope

#1661, Add aggregate variant of ST\_SameAlignment

#1719, Add support for Point and GeometryCollection ST\_MakeValid inputs

- #1780, support ST\_GeoHash for geography
  - #1796, Big performance boost for distance calculations in geography
  - #1802, improved function interruptibility.
  - #1823, add parameter in ST\_AsGML to use id column for GML 3 output (become mandatory since GML 3.2.1)
  - #1856, tiger geocoder: reverse geocoder rating setting for prefer numbered highway name
  - #1938, Refactor basic ST\_AddBand to add multiple new bands in one call
  - #1978, wrong answer when calculating length of a closed circular arc (circle)
  - #1989, Preprocess input geometry to just intersection with raster to be clipped
  - #2021, Added multi-band support to ST\_Union(raster, ...) aggregate function
  - #2006, better support of ST\_Area(geography) over poles and dateline
  - #2065, ST\_Clip(raster, ...) now a C function
  - #2069, Added parameters to ST\_Tile(raster) to control padding of tiles
  - #2078, New variants of ST\_Slope, ST\_Aspect and ST\_HillShade to provide solution to handling tiles in a coverage
  - #2097, Added RANGE uniontype option for ST\_Union(raster)
  - #2105, Added ST\_Transform(raster) variant for aligning output to reference raster
  - #2119, Rasters passed to ST\_Resample(), ST\_Rescale(), ST\_Reskew(), and ST\_SnapToGrid() no longer require an SRID
  - #2141, More verbose output when constraints fail to be added to a raster column
  - #2143, Changed blocksize constraint of raster to allow multiple values
  - #2148, Addition of coverage\_tile constraint for raster
  - #2149, Addition of spatially\_unique constraint for raster
- TopologySummary output now includes unregistered layers and a count of missing TopoGeometry objects from their natural layer.
- ST\_HillShade(), ST\_Aspect() and ST\_Slope() have one new optional parameter to interpolate NODATA pixels before running the operation.
- Point variant of ST\_SetValue(raster) is now a wrapper around geomval variant of ST\_SetValues(rast).
- Proper support for raster band's isnodata flag in core API and loader.
- Additional default values for parameters of ST\_Aspect and ST\_HillShade
- #2178, ST\_Summary now advertises presence of known srid with an [S] flag
  - #2202, Make libjson-c optional (--without-json configure switch)
  - #2213, Add support libjson-c 0.10+
  - #2231, raster2pgsql supports user naming of filename column with -n
  - #2200, ST\_Union(raster, uniontype) unions all bands of all rasters
  - #2264, postgis\_restore.pl support for restoring into databases with postgis in a custom schema
  - #2244, emit warning when changing raster's georeference if raster has out-db bands
  - #2222, add parameter OutAsIn to flag whether ST\_AsBinary should return out-db bands as in-db bands
-



## A.22.4 Correções

- #1839, handling of subdatasets in GeoTIFF in raster2pgsql.
  - #1840, fix logic of when to compute # of tiles in raster2pgsql.
  - #1870, align the docs and actual behavior of raster's ST\_Intersects
  - #1872, fix ST\_ApproxSummarystats to prevent division by zero
  - #1875, ST\_SummaryStats returns NULL for all parameters except count when count is zero
  - #1932, fix raster2pgsql of syntax for index tablespaces
  - #1936, ST\_GeomFromGML on CurvePolygon causes server crash
  - #1939, remove custom data types: summarystats, histogram, quantile, valuecount
  - #1951, remove crash on zero-length linestrings
  - #1957, ST\_Distance to a one-point LineString returns NULL
  - #1976, Geography point-in-ring code overhauled for more reliability
  - #1981, cleanup of unused variables causing warnings with gcc 4.6+
  - #1996, support POINT EMPTY in GeoJSON output
  - #2062, improve performance of distance calculations
  - #2057, Fixed linking issue for raster2pgsql to libpq
  - #2077, Fixed incorrect values returning from ST\_Hillshade()
  - #2019, ST\_FlipCoordinates does not update bbox
  - #2100, ST\_AsRaster may not return raster with specified pixel type
  - #2126, Better handling of empty rasters from ST\_ConvexHull()
  - #2165, ST\_NumPoints regression failure with CircularString
  - #2168, ST\_Distance is not always commutative
  - #2182, Fix issue with outdb rasters with no SRID and ST\_Resize
  - #2188, Fix function parameter value overflow that caused problems when copying data from a GDAL dataset
  - #2198, Fix incorrect dimensions used when generating bands of out-db rasters in ST\_Tile()
  - #2201, ST\_GeoHash wrong on boundaries
  - #2203, Changed how rasters with unknown SRID and default geotransform are handled when passing to GDAL Warp API
  - #2215, Fixed raster exclusion constraint for conflicting name of implicit index
  - #2251, Fix bad dimensions when rescaling rasters with default geotransform matrix
  - #2133, Fix performance regression in expression variant of ST\_MapAlgebra
  - #2257, GBOX variables not initialized when testing with empty geometries
  - #2271, Prevent parallel make of raster
  - #2282, Fix call to undefined function nd\_stats\_to\_grid() in debug mode
  - #2307, ST\_MakeValid outputs invalid geometries
  - #2309, Remove confusing INFO message when trying to get SRS info
  - #2336, FIPS 20 (KS) causes wildcard expansion to wget all files
  - #2348, Provide raster upgrade path for 2.0 to 2.1
  - #2351, st\_distance between geographies wrong
  - #2359, Fix handling of schema name when adding overview constraints
  - #2371, Support GEOS versions with more than 1 digit in micro
  - #2383, Remove unsafe use of \ from raster warning message
  - #2384, Incorrect variable datatypes for ST\_Neighborhood
-

## A.22.5 Known Issues

#2111, Raster bands can only reference the first 256 bands of out-db rasters

## A.23 Versão 2.0.5

Data de Lançamento: 2014/03/31

This is a bug fix release, addressing issues that have been filed since the 2.0.4 release. If you are using PostGIS 2.0+ a soft upgrade is required. For users of PostGIS 1.5 or below, a hard upgrade is required.

### A.23.1 Correção de Erros

#2494, avoid memcpy in GIST index

#2502, Fix postgis\_topology\_scripts\_installed() install schema

#2504, Fix segfault on bogus pgsq2shp call

#2528, Fix memory leak in ST\_Split / lwline\_split\_by\_line

#2532, Add missing raster/geometry commutator operators

#2533, Remove duplicated signatures

#2552, Fix NULL raster handling in ST\_AsPNG, ST\_AsTIFF and ST\_AsJPEG

#2555, Fix parsing issue of range arguments of ST\_Reclass

#2589, Remove use of unnecessary void pointers

#2607, Cannot open more than 1024 out-db files in process

#2610, Ensure face splitting algorithm uses the edge index

#2619, Empty ring array in GeoJSON polygon causes crash

#2638, Geography distance on M geometries sometimes wrong

### A.23.2 Mudanças importantes

##2514, Change raster license from GPL v3+ to v2+, allowing distribution of PostGIS Extension as GPLv2.

## A.24 Versão 2.0.4

Data de lançamento: 2013/09/06

This is a bug fix release, addressing issues that have been filed since the 2.0.3 release. If you are using PostGIS 2.0+ a soft upgrade is required. For users of PostGIS 1.5 or below, a hard upgrade is required.

### A.24.1 Correção de Erros

#2110, Equality operator between EMPTY and point on origin

Allow adding points at precision distance with TopoGeo\_addPoint

#1968, Fix missing edge from toTopoGeom return

#2165, ST\_NumPoints regression failure with CircularString

- #2168, ST\_Distance is not always commutative
- #2186, gui progress bar updates too frequent
- #2201, ST\_GeoHash wrong on boundaries
- #2257, GBOX variables not initialized when testing with empty geometries
- #2271, Prevent parallel make of raster
- #2267, Server crash from analyze table
- #2277, potential segfault removed
- #2307, ST\_MakeValid outputs invalid geometries
- #2351, st\_distance between geographies wrong
- #2359, Incorrect handling of schema for overview constraints
- #2371, Support GEOS versions with more than 1 digit in micro
- #2372, Cannot parse space-padded KML coordinates
- Fix build with systemwide liblwgeom installed
- #2383, Fix unsafe use of \ in warning message
- #2410, Fix segmentize of collinear curve
- #2412, ST\_LineToCurve support for lines with less than 4 vertices
- #2415, ST\_Multi support for COMPOUNDCURVE and CURVEPOLYGON
- #2420, ST\_LineToCurve: require at least 8 edges to define a full circle
- #2423, ST\_LineToCurve: require all arc edges to form the same angle
- #2424, ST\_CurveToLine: add support for COMPOUNDCURVE in MULTICURVE
- #2427, Make sure to retain first point of curves on ST\_CurveToLine

## A.24.2 Melhorias

- #2269, Avoid uselessly detoasting full geometries on ANALYZE

## A.24.3 Known Issues

- #2111, Raster bands can only reference the first 256 bands of out-db rasters

## A.25 Versão 2.0.3

Data de Lançamento: 2013/03/01

This is a bug fix release, addressing issues that have been filed since the 2.0.2 release. If you are using PostGIS 2.0+ a soft upgrade is required. For users of PostGIS 1.5 or below, a hard upgrade is required.

### A.25.1 Correção de Erros

#2126, Better handling of empty rasters from ST\_ConvexHull()

#2134, Make sure to process SRS before passing it off to GDAL functions

Fix various memory leaks in liblwgeom

#2173, Fix robustness issue in splitting a line with own vertex also affecting topology building (#2172)

#2174, Fix usage of wrong function lwpoly\_free()

#2176, Fix robustness issue with ST\_ChangeEdgeGeom

#2184, Properly copy topologies with Z value

postgis\_restore.pl support for mixed case geometry column name in dumps

#2188, Fix function parameter value overflow that caused problems when copying data from a GDAL dataset

#2216, More memory errors in MultiPolygon GeoJSON parsing (with holes)

Fix Memory leak in GeoJSON parser

### A.25.2 Melhorias

#2141, More verbose output when constraints fail to be added to a raster column

Speedup ST\_ChangeEdgeGeom

## A.26 Versão 2.0.2

Data de Lançamento: 2012/12/03

This is a bug fix release, addressing issues that have been filed since the 2.0.1 release.

### A.26.1 Correção de Erros

#1287, Drop of "gist\_geometry\_ops" broke a few clients package of legacy\_gist.sql for these cases

#1391, Errors during upgrade from 1.5

#1828, Poor selectivity estimate on ST\_DWithin

#1838, error importing tiger/line data

#1869, ST\_AsBinary is not unique added to legacy\_minor/legacy.sql scripts

#1885, Missing field from tabblock table in tiger2010 census\_loader.sql

#1891, Use LDFFLAGS environment when building liblwgeom

#1900, Fix pgsq2shp for big-endian systems

#1932, Fix raster2pgsql for invalid syntax for setting index tablespace

#1936, ST\_GeomFromGML on CurvePolygon causes server crash

#1955, ST\_ModEdgeHeal and ST\_NewEdgeHeal for doubly connected edges

#1957, ST\_Distance to a one-point LineString returns NULL

#1976, Geography point-in-ring code overhauled for more reliability

#1978, wrong answer calculating length of closed circular arc (circle)

#1981, Remove unused but set variables as found with gcc 4.6+

- #1987, Restore 1.5.x behaviour of ST\_Simplify
  - #1989, Preprocess input geometry to just intersection with raster to be clipped
  - #1991, geocode really slow on PostgreSQL 9.2
  - #1996, support POINT EMPTY in GeoJSON output
  - #1998, Fix ST\_{Mod,New}EdgeHeal joining edges sharing both endpoints
  - #2001, ST\_CurveToLine has no effect if the geometry doesn't actually contain an arc
  - #2015, ST\_IsEmpty('POLYGON(EMPTY)') returns False
  - #2019, ST\_FlipCoordinates does not update bbox
  - #2025, Fix side location conflict at TopoGeo\_AddLineString
  - #2026, improve performance of distance calculations
  - #2033, Fix adding a splitting point into a 2.5d topology
  - #2051, Fix excess of precision in ST\_AsGeoJSON output
  - #2052, Fix buffer overflow in lwgeom\_to\_geojson
  - #2056, Fixed lack of SRID check of raster and geometry in ST\_SetValue()
  - #2057, Fixed linking issue for raster2psql to libpq
  - #2060, Fix "dimension" check violation by GetTopoGeomElementArray
  - #2072, Removed outdated checks preventing ST\_Intersects(raster) from working on out-db bands
  - #2077, Fixed incorrect answers from ST\_Hillshade(raster)
  - #2092, Namespace issue with ST\_GeomFromKML,ST\_GeomFromGML for libxml 2.8+
  - #2099, Fix double free on exception in ST\_OffsetCurve
  - #2100, ST\_AsRaster() may not return raster with specified pixel type
  - #2108, Ensure ST\_Line\_Interpolate\_Point always returns POINT
  - #2109, Ensure ST\_Centroid always returns POINT
  - #2117, Ensure ST\_PointOnSurface always returns POINT
  - #2129, Fix SRID in ST\_Homogenize output with collection input
  - #2130, Fix memory error in MultiPolygon GeoJson parsing
- Update URL of Maven jar

## A.26.2 Melhorias

- #1581, ST\_Clip(raster, ...) no longer imposes NODATA on a band if the corresponding band from the source raster did not have NODATA
- #1928, Accept array properties in GML input multi-geom input (Kashif Rasul and Shoaib Burq / SpacialDB)
- #2082, Add indices on start\_node and end\_node of topology edge tables
- #2087, Speedup topology.GetRingEdges using a recursive CTE

## A.27 Versão 2.0.1

Data de lançamento: 2012/06/22

Este é uma versão apenas de correção de erros, resolvendo questões que foram solicitadas desde a versão 2.0.0.

## A.27.1 Correção de Erros

- #1264, fix `st_dwithin(geog, geog, 0)`.
  - #1468 `shp2pgsql-gui` table column schema get shifted
  - #1694, fix building with clang. (vince)
  - #1708, improve restore of pre-PostGIS 2.0 backups.
  - #1714, more robust handling of high topology tolerance.
  - #1755, `ST_GeographyFromText` support for higher dimensions.
  - #1759, loading transformed shapefiles in raster enabled db.
  - #1761, handling of subdatasets in NetCDF, HDF4 and HDF5 in `raster2pgsql`.
  - #1763, `topology.toTopoGeom` use with custom `search_path`.
  - #1766, don't let `ST_RemEdge*` destroy peripheral `TopoGeometry` objects.
  - #1774, Clearer error on setting an edge geometry to an invalid one.
  - #1775, `ST_ChangeEdgeGeom` collision detection with 2-vertex target.
  - #1776, fix `ST_SymDifference(empty, geom)` to return `geom`.
  - #1779, install SQL comment files.
  - #1782, fix spatial reference string handling in raster.
  - #1789, fix false edge-node crossing report in `ValidateTopology`.
  - #1790, fix `toTopoGeom` handling of duplicated primitives.
  - #1791, fix `ST_Azimuth` with very close but distinct points.
  - #1797, fix `(ValidateTopology(xxx)).*` syntax calls.
  - #1805, put back the 900913 SRID entry.
  - #1813, Only show readable relations in metadata tables.
  - #1819, fix floating point issues with `ST_World2RasterCoord` and `ST_Raster2WorldCoord` variants.
  - #1820 compilation on 9.2beta1.
  - #1822, topology load on PostgreSQL 9.2beta1.
  - #1825, fix prepared geometry cache lookup
  - #1829, fix uninitialized read in GeoJSON parser
  - #1834, revise `postgis` extension to only backup user specified `spatial_ref_sys`
  - #1839, handling of subdatasets in GeoTIFF in `raster2pgsql`.
  - #1840, fix logic of when to compute # of tiles in `raster2pgsql`.
  - #1851, fix `spatial_ref_system` parameters for EPSG:3844
  - #1857, fix failure to detect endpoint mismatch in `ST_AddEdge*Face*`
  - #1865, data loss in `postgis_restore.pl` when data rows have leading dashes.
  - #1867, catch invalid topology name passed to `topogeo_add*`
  - #1872, fix `ST_ApproxSummaryStats` to prevent division by zero
  - #1873, fix `ptarray_locate_point` to return interpolated Z/M values for on-the-line case
  - #1875, `ST_SummaryStats` returns NULL for all parameters except count when count is zero
  - #1881, `shp2pgsql-gui -- editing` a field sometimes triggers removing row
  - #1883, Geocoder install fails trying to run `create_census_base_tables()` (Brian Panulla)
-

## A.27.2 Melhorias

Mensagem de exceção mais detalhada das funções de edição da topologia.

#1786, dependências de construção melhoradas

#1806, aceleração da ST\_BuildArea, ST\_MakeValid e ST\_GetFaceGeometry.

#1812, Add lwgeom\_normalize in LIBLWGEOM for more stable testing.

## A.28 Versão 2.0.0

Data de Lançamento: 2012/04/03

Esta é uma versão maior. Uma atualização hard é necessária. Sim, isto significa um carregamento de descarte completo e algumas preparações especiais se você estiver usando funções obsoletas. Recorra a Section 2.10.2 para detalhes na atualização. Recorra a Section 14.12.7 para mais detalhes e funções alteradas/novas.

### A.28.1 Verificadores - Nossos heróis não aclamados

Somos eternamente gratos aos numerosos membros na comunidade PostGIS que foram corajosos o suficiente para testar as novas características nesta versão. Nenhuma outra versão pode ser bem sucedida sem este pessoal.

Abaixo estão aqueles que foram os mais corajosos, forneceram bem detalhados e através de relatórios de bugs, e análises detalhadas.

Andrea Peri - vários testes em topologia, checando para a correção

Andreas Forø Tollefsen - testando raster

Chris English - a topologia estressa testando novas funções

Salvatore Larosa - testando a robustez do teste

Brian Hamlin - Benchmarking (também divisões experimentais antes que estejam completamente no centro), teste geral de várias partes

Mike Pease - Teste do tiger geocoder - relatório de issues bastante detalhado

Tom van Tilburg - testando raster

### A.28.2 Importante / Mudanças Críticas

#722, #302, Most deprecated functions removed (over 250 functions) (Regina Obe, Paul Ramsey)

Unknown SRID changed from -1 to 0. (Paul Ramsey)

-- (most deprecated in 1.2) removed non-ST variants buffer, length, intersects (and internal functions renamed) etc.

-- Se você esteve usando funções menosprezadas MUDE suas aplicações ou sofra as consequências. Se você não vê uma função documentada -- não é suportada ou é uma função interna. Algumas restrições, em tabelas mais antigas, foram construídas com funções desprezadas. Se você restaurar, talvez precise reconstruir as restrições de tabela com populate\_geometry\_columns(). Se você tem aplicações ou ferramentas que confiam em funções menosprezadas, por favor recorra a [?qandaentry] para mais detalhes.

#944 geometry\_columns é uma view agora em vez de uma tabela (Paul Ramsey, Regina Obe) para tabelas criadas com as formas antigas de ler (srid, type, dims) restrições para colunas geométricas criadas com modificadores de tipo da definição da coluna

#1081, #1082, #1084, #1088 - Funções de gerenciamento suportam a criação de funções de colunas geométricas typmod agora padrão a criação typmod (Regina Obe)

#1083 probe\_geometry\_columns(), rename\_geometry\_table\_constraints(), fix\_geometry\_columns(); removed - now obsolete with geometry\_column view (Regina Obe)

#817 Renomeando funções 3D para conversão de ST\_3D (Nicklas Avén)

#548 (sorta), ST\_NumGeometries, ST\_GeometryN agora retorna 1 (ou a geometria) em vez de nulo para geometrias únicas (Sandro Santilli, Maxime van Noppen)

### A.28.3 Novos Recursos

**KNN Gist index based centroid (<->) and box (<#>) distance operators (Paul Ramsey / funded by Vizzuality)**

Support for TIN and PolyHedralSurface and enhancement of many functions to support 3D (Olivier Courtin / Oslandia)

**Raster support integrated and documented** (Pierre Racine, Jorge Arévalo, Mateusz Loskot, Sandro Santilli, David Zwarg, Regina Obe, Bborie Park) (Company developer and funding: University Laval, Deimos Space, CadCorp, Michigan Tech Research Institute, Azavea, Paragon Corporation, UC Davis Center for Vectorborne Diseases)

Making spatial indexes 3D aware - in progress (Paul Ramsey, Mark Cave-Ayland)

Topology support improved (more functions), documented, testing (Sandro Santilli / Faunalia for RT-SIGTA), Andrea Peri, Regina Obe, Jose Carlos Martinez Llari

3D relationship and measurement support functions (Nicklas Avén)

ST\_3DDistance, ST\_3DClosestPoint, ST\_3DIntersects, ST\_3DShortestLine and more...

N-Dimensional spatial indexes (Paul Ramsey / OpenGeo)

ST\_Split (Sandro Santilli / Faunalia para RT-SIGTA)

ST\_IsValidDetail (Sandro Santilli / Faunalia para RT-SIGTA)

ST\_MakeValid (Sandro Santilli / Faunalia para RT-SIGTA)

ST\_RemoveRepeatedPoints (Sandro Santilli / Faunalia para RT-SIGTA)

ST\_GeometryN and ST\_NumGeometries support for non-collections (Sandro Santilli)

ST\_IsCollection (Sandro Santilli, Maxime van Noppen)

ST\_SharedPaths (Sandro Santilli / Faunalia para RT-SIGTA)

ST\_Snap (Sandro Santilli)

ST\_RelateMatch (Sandro Santilli / Faunalia para RT-SIGTA)

ST\_ConcaveHull (Regina Obe e Leo Hsu / Paragon Corporation)

ST\_UnaryUnion (Sandro Santilli / Faunalia para RT-SIGTA)

ST\_AsX3D (Regina Obe / Arrival 3D funding)

ST\_OffsetCurve (Sandro Santilli, Rafal Magda)

**ST\_GeomFromGeoJSON (Kashif Rasul, Paul Ramsey / Vizzuality funding)**

### A.28.4 Melhorias

Made shape file loader tolerant of truncated multibyte values found in some free worldwide shapefiles (Sandro Santilli)

Lots of bug fixes and enhancements to shp2pgsql Beefing up regression tests for loaders Reproject support for both geometry and geography during import (Jeff Adams / Azavea, Mark Cave-Ayland)

pgsql2shp conversion from predefined list (Loic Dachary / Mark Cave-Ayland)

Shp-pgsql GUI loader - support loading multiple files at a time. (Mark Leslie)

Extras - upgraded tiger\_geocoder from using old TIGER format to use new TIGER shp and file structure format (Stephen Frost)

Extras - revised tiger\_geocoder to work with TIGER census 2010 data, addition of reverse geocoder function, various bug fixes, accuracy enhancements, limit max result return, speed improvements, loading routines. (Regina Obe, Leo Hsu / Paragon Corporation / funding provided by Hunter Systems Group)

Overall Documentation proofreading and corrections. (Kasif Rasul)

Cleanup PostGIS JDBC classes, revise to use Maven build. (Maria Arias de Reyna, Sandro Santilli)



## A.28.5 Correção de Erros

#1335 ST\_AddPoint returns incorrect result on Linux (Even Rouault)

## A.28.6 Créditos específicos deste lançamento

We thank [U.S Department of State Human Information Unit \(HIU\)](#) and [Vizzuality](#) for general monetary support to get PostGIS 2.0 out the door.

## A.29 Versão 1.5.4

Data de lançamento: 2012/05/07

Este é uma versão contendo apenas correções, resolvendo questões relatadas desde o lançamento da versão 1.5.3.

### A.29.1 Correção de Erros

#547, ST\_Contains memory problems (Sandro Santilli)

#621, Problem finding intersections with geography (Paul Ramsey)

#627, PostGIS/PostgreSQL process die on invalid geometry (Paul Ramsey)

#810, Increase accuracy of area calculation (Paul Ramsey)

#852, improve spatial predicates robustness (Sandro Santilli, Nicklas Avén)

#877, ST\_Estimated\_Extent returns NULL on empty tables (Sandro Santilli)

#1028, ST\_AsSVG kills whole postgres server when fails (Paul Ramsey)

#1056, Fix boxes of arcs and circle stroking code (Paul Ramsey)

#1121, populate\_geometry\_columns using deprecated functions (Regin Obe, Paul Ramsey)

#1135, improve testsuite predictability (Andreas 'ads' Scherbaum)

#1146, images generator crashes (bronaugh)

#1170, North Pole intersection fails (Paul Ramsey)

#1179, ST\_AsText crash with bad value (kjurka)

#1184, honour DESTDIR in documentation Makefile (Bryce L Nordgren)

#1227, server crash on invalid GML

#1252, SRID appearing in WKT (Paul Ramsey)

#1264, st\_dwithin(g, g, 0) doesn't work (Paul Ramsey)

#1344, allow exporting tables with invalid geometries (Sandro Santilli)

#1389, wrong proj4text for SRID 31300 and 31370 (Paul Ramsey)

#1406, shp2pgsql crashes when loading into geography (Sandro Santilli)

#1595, fixed SRID redundancy in ST\_Line\_SubString (Sandro Santilli)

#1596, check SRID in UpdateGeometrySRID (Mike Toews, Sandro Santilli)

#1602, fix ST\_Polygonize to retain Z (Sandro Santilli)

#1697, fix crash with EMPTY entries in GiST index (Paul Ramsey)

#1772, fix ST\_Line\_Locate\_Point with collapsed input (Sandro Santilli)

#1799, Protect ST\_Segmentize from max\_length=0 (Sandro Santilli)

Alter parameter order in 900913 (Paul Ramsey)

Support builds with "gmake" (Greg Troxel)

## A.30 Versão 1.5.3

Data de Lançamento: 2011/06/25

This is a bug fix release, addressing issues that have been filed since the 1.5.2 release. If you are running PostGIS 1.3+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

### A.30.1 Correção de Erros

- #1056, produce correct bboxes for arc geometries, fixes index errors (Paul Ramsey)
- #1007, ST\_IsValid crash fix requires GEOS 3.3.0+ or 3.2.3+ (Sandro Santilli, reported by Birgit Laggner)
- #940, support for PostgreSQL 9.1 beta 1 (Regina Obe, Paul Ramsey, patch submitted by stl)
- #845, ST\_Intersects precision error (Sandro Santilli, Nicklas Avén) Reported by cdestigter
- #884, Unstable results with ST\_Within, ST\_Intersects (Chris Hodgson)
- #779, shp2pgsql -S option seems to fail on points (Jeff Adams)
- #666, ST\_DumpPoints is not null safe (Regina Obe)
- #631, Update NZ projections for grid transformation support (jpalmer)
- #630, Peculiar Null treatment in arrays in ST\_Collect (Chris Hodgson) Reported by David Bitner
- #624, Memory leak in ST\_GeogFromText (ryang, Paul Ramsey)
- #609, Bad source code in manual section 5.2 Java Clients (simoc, Regina Obe)
- #604, shp2pgsql usage touchups (Mike Toews, Paul Ramsey)
- #573 ST\_Union fails on a group of linestrings Not a PostGIS bug, fixed in GEOS 3.3.0
- #457 ST\_CollectionExtract returns non-requested type (Nicklas Avén, Paul Ramsey)
- #441 ST\_AsGeoJson Bbox on GeometryCollection error (Olivier Courtin)
- #411 Ability to backup invalid geometries (Sando Santilli) Reported by Regione Toscana
- #409 ST\_AsSVG - degraded (Olivier Courtin) Reported by Sdikiy
- #373 Documentation syntax error in hard upgrade (Paul Ramsey) Reported by psvensso

## A.31 Versão 1.5.2

Data de Lançamento: 2010/09/27

This is a bug fix release, addressing issues that have been filed since the 1.5.1 release. If you are running PostGIS 1.3+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

### A.31.1 Correção de Erros

Loader: fix handling of empty (0-verticed) geometries in shapefiles. (Sandro Santilli)

#536, Geography ST\_Intersects, ST\_Covers, ST\_CoveredBy and Geometry ST\_Equals not using spatial index (Regina Obe, Nicklas Aven)

#573, Improvement to ST\_Contains geography (Paul Ramsey)

Loader: Add support for command-q shutdown in Mac GTK build (Paul Ramsey)

#393, Loader: Add temporary patch for large DBF files (Maxime Guillaud, Paul Ramsey)

- #507, Fix wrong OGC URN in GeoJSON and GML output (Olivier Courtin)
- spatial\_ref\_sys.sql Add datum conversion for projection SRID 3021 (Paul Ramsey)
- Geography - remove crash for case when all geographies are out of the estimate (Paul Ramsey)
- #469, Fix for array\_aggregation error (Greg Stark, Paul Ramsey)
- #532, Temporary geography tables showing up in other user sessions (Paul Ramsey)
- #562, ST\_Dwithin errors for large geographies (Paul Ramsey)
- #513, shape loading GUI tries to make spatial index when loading DBF only mode (Paul Ramsey)
- #527, shape loading GUI should always append log messages (Mark Cave-Ayland)
- #504, shp2pgsql should rename xmin/xmax fields (Sandro Santilli)
- #458, postgis\_comments being installed in contrib instead of version folder (Mark Cave-Ayland)
- #474, Analyzing a table with geography column crashes server (Paul Ramsey)
- #581, LWGEOM-expand produces inconsistent results (Mark Cave-Ayland)
- #513, Add dbf filter to shp2pgsql-gui and allow uploading dbf only (Paul Ramsey)
- Fix further build issues against PostgreSQL 9.0 (Mark Cave-Ayland)
- #572, Password whitespace for Shape File (Mark Cave-Ayland)
- #603, shp2pgsql: "-w" produces invalid WKT for MULTI\* objects. (Mark Cave-Ayland)

## A.32 Versão 1.5.1

Data de Lançamento: 2010/03/11

This is a bug fix release, addressing issues that have been filed since the 1.4.1 release. If you are running PostGIS 1.3+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

### A.32.1 Correção de Erros

- #410, update embedded bbox when applying ST\_SetPoint, ST\_AddPoint ST\_RemovePoint to a linestring (Paul Ramsey)
  - #411, allow dumping tables with invalid geometries (Sandro Santilli, for Regione Toscana-SIGTA)
  - #414, include geography\_columns view when running upgrade scripts (Paul Ramsey)
  - #419, allow support for multilinestring in ST\_Line\_Substring (Paul Ramsey, for Lidwala Consulting Engineers)
  - #421, fix computed string length in ST\_AsGML() (Olivier Courtin)
  - #441, fix GML generation with heterogeneous collections (Olivier Courtin)
  - #443, incorrect coordinate reversal in GML 3 generation (Olivier Courtin)
  - #450, #451, wrong area calculation for geography features that cross the date line (Paul Ramsey)
- Ensure support for upcoming 9.0 PgSQL release (Paul Ramsey)

## A.33 Versão 1.5.0

Data de Lançamento: 2010/02/04

This release provides support for geographic coordinates (lat/lon) via a new GEOGRAPHY type. Also performance enhancements, new input format support (GML,KML) and general upkeep.

---

### A.33.1 API Stability

The public API of PostGIS will not change during minor (0.0.X) releases.

The definition of the `=~` operator has changed from an exact geometric equality check to a bounding box equality check.

### A.33.2 Compatibilidade

GEOS, Proj4, and LibXML2 are now mandatory dependencies

The library versions below are the minimum requirements for PostGIS 1.5

PostgreSQL 8.3 ou superior em todas as plataformas.

GEOS 3.1 and higher only (GEOS 3.2+ to take advantage of all features)

LibXML2 2.5+ related to new ST\_GeomFromGML/KML functionality

Proj4 4.5 ou superior apenas

### A.33.3 Novos Recursos

Section [14.12.9](#)

Added Hausdorff distance calculations ([#209](#)) (Vincent Picavet)

Added parameters argument to ST\_Buffer operation to support one-sided buffering and other buffering styles (Sandro Santilli)

Addition of other Distance related visualization and analysis functions (Nicklas Aven)

- ST\_ClosestPoint
- ST\_DFullyWithin
- ST\_LongestLine
- ST\_MaxDistance
- ST\_ShortestLine

ST\_DumpPoints (Maxime van Noppen)

KML, GML input via ST\_GeomFromGML and ST\_GeomFromKML (Olivier Courtin)

Extract homogeneous collection with ST\_CollectionExtract (Paul Ramsey)

Add measure values to an existing linestring with ST\_AddMeasure (Paul Ramsey)

History table implementation in utils (George Silva)

Geography type and supporting functions

- Spherical algorithms (Dave Skea)
  - Object/index implementation (Paul Ramsey)
  - Selectivity implementation (Mark Cave-Ayland)
  - Serializations to KML, GML and JSON (Olivier Courtin)
  - ST\_Area, ST\_Distance, ST\_DWithin, ST\_GeogFromText, ST\_GeogFromWKB, ST\_Intersects, ST\_Covers, ST\_Buffer (Paul Ramsey)
-

### A.33.4 Melhorias

Performance improvements to ST\_Distance (Nicklas Aven)  
Documentation updates and improvements (Regina Obe, Kevin Neufeld)  
Teste e controle de qualidade (Regina Obe)  
PostGIS 1.5 support PostgreSQL 8.5 trunk (Guillaume Lelarge)  
Win32 support and improvement of core shp2pgsql-gui (Mark Cave-Ayland)  
In place 'make check' support (Paul Ramsey)

### A.33.5 Correção de Erros

<http://trac.osgeo.org/postgis/query?status=closed&milestone=PostGIS+1.5.0&order=priority>

## A.34 Versão 1.4.0

Data de lançamento: 2009/07/24

This release provides performance enhancements, improved internal structures and testing, new features, and upgraded documentation. If you are running PostGIS 1.1+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

### A.34.1 API Stability

As of the 1.4 release series, the public API of PostGIS will not change during minor releases.

### A.34.2 Compatibilidade

The versions below are the \*minimum\* requirements for PostGIS 1.4

PostgreSQL 8.2 ou superior em todas as plataformas

GEOS 3.0 ou superior apenas

PROJ4 4.5 ou superior apenas

### A.34.3 Novos Recursos

ST\_Union() uses high-speed cascaded union when compiled against GEOS 3.1+ (Paul Ramsey)

ST\_ContainsProperly() necessita do GEOS 3.1+

ST\_Intersects(), ST\_Contains(), ST\_Within() use high-speed cached prepared geometry against GEOS 3.1+ (Paul Ramsey / funded by Zonar Systems)

Vastly improved documentation and reference manual (Regina Obe & Kevin Neufeld)

Figures and diagram examples in the reference manual (Kevin Neufeld)

ST\_IsValidReason() returns readable explanations for validity failures (Paul Ramsey)

ST\_GeoHash() returns a geohash.org signature for geometries (Paul Ramsey)

GTK+ multi-platform GUI for shape file loading (Paul Ramsey)

ST\_LineCrossingDirection() returns crossing directions (Paul Ramsey)

ST\_LocateBetweenElevations() returns sub-string based on Z-ordinate. (Paul Ramsey)

Geometry parser returns explicit error message about location of syntax errors (Mark Cave-Ayland)

ST\_AsGeoJSON() return JSON formatted geometry (Olivier Courtin)

Populate\_Geometry\_Columns() -- automatically add records to geometry\_columns for TABLES and VIEWS (Kevin Neufeld)

ST\_MinimumBoundingCircle() -- returns the smallest circle polygon that can encompass a geometry (Bruce Rindahl)

#### A.34.4 Melhorias

Core geometry system moved into independent library, liblwgeom. (Mark Cave-Ayland)

New build system uses PostgreSQL "pgxs" build bootstrapper. (Mark Cave-Ayland)

Debugging framework formalized and simplified. (Mark Cave-Ayland)

All build-time #defines generated at configure time and placed in headers for easier cross-platform support (Mark Cave-Ayland)

Logging framework formalized and simplified (Mark Cave-Ayland)

Expanded and more stable support for CIRCULARSTRING, COMPOUNDCURVE and CURVEPOLYGON, better parsing, wider support in functions (Mark Leslie & Mark Cave-Ayland)

Improved support for OpenSolaris builds (Paul Ramsey)

Improved support for MSVC builds (Mateusz Loskot)

Updated KML support (Olivier Courtin)

Unit testing framework for liblwgeom (Paul Ramsey)

New testing framework to comprehensively exercise every PostGIS function (Regine Obe)

Performance improvements to all geometry aggregate functions (Paul Ramsey)

Support for the upcoming PostgreSQL 8.4 (Mark Cave-Ayland, Talha Bin Rizwan)

Shp2pgsql and pgsq2shp re-worked to depend on the common parsing/unparsing code in liblwgeom (Mark Cave-Ayland)

Use of PDF DbLatex to build PDF docs and preliminary instructions for build (Jean David Techer)

Automated User documentation build (PDF and HTML) and Developer Doxygen Documentation (Kevin Neufeld)

Automated build of document images using ImageMagick from WKT geometry text files (Kevin Neufeld)

More attractive CSS for HTML documentation (Dane Springmeyer)

#### A.34.5 Correção de Erros

<http://trac.osgeo.org/postgis/query?status=closed&milestone=PostGIS+1.4.0&order=priority>

### A.35 Versão 1.3.6

Data de lançamento: 2009/05/04

If you are running PostGIS 1.1+, a soft upgrade is sufficient otherwise a hard upgrade is recommended. This release adds support for PostgreSQL 8.4, exporting prj files from the database with shape data, some crash fixes for shp2pgsql, and several small bug fixes in the handling of "curve" types, logical error importing dbf only files, improved error handling of AddGeometryColumns.

### A.36 Versão 1.3.5

Data de lançamento: 2008/12/15

If you are running PostGIS 1.1+, a soft upgrade is sufficient otherwise a hard upgrade is recommended. This release is a bug fix release to address a failure in ST\_Force\_Collection and related functions that critically affects using MapServer with LINE layers.

---

## A.37 Versão 1.3.4

Data de Lançamento: 2008/11/24

This release adds support for GeoJSON output, building with PostgreSQL 8.4, improves documentation quality and output aesthetics, adds function-level SQL documentation, and improves performance for some spatial predicates (point-in-polygon tests).

Bug fixes include removal of crashers in handling circular strings for many functions, some memory leaks removed, a linear referencing failure for measures on vertices, and more. See the NEWS file for details.

## A.38 Versão 1.3.3

Data de Lançamento: 2008/04/12

This release fixes bugs shp2pgsql, adds enhancements to SVG and KML support, adds a ST\_SimplifyPreserveTopology function, makes the build more sensitive to GEOS versions, and fixes a handful of severe but rare failure cases.

## A.39 Versão 1.3.2

Data de Lançamento: 2007/12/01

This release fixes bugs in ST\_EndPoint() and ST\_Envelope, improves support for JDBC building and OS/X, and adds better support for GML output with ST\_AsGML(), including GML3 output.

## A.40 Versão 1.3.1

Data de Lançamento: 2007/08/13

This release fixes some oversights in the previous release around version numbering, documentation, and tagging.

## A.41 Versão 1.3.0

Data de Lançamento: 2007/08/09

This release provides performance enhancements to the relational functions, adds new relational functions and begins the migration of our function names to the SQL-MM convention, using the spatial type (SP) prefix.

### A.41.1 Funcionalidade Adicionada

JDBC: Added Hibernate Dialect (thanks to Norman Barker)

Added ST\_Covers and ST\_CoveredBy relational functions. Description and justification of these functions can be found at <http://lin-ear-th-inking.blogspot.com/2007/06/subtleties-of-ogc-covers-spatial.html>

Added ST\_DWithin relational function.

### A.41.2 Melhorias de Desempenho

Added cached and indexed point-in-polygon short-circuits for the functions ST\_Contains, ST\_Intersects, ST\_Within and ST\_Disjoint

Added inline index support for relational functions (except ST\_Disjoint)

---

### A.41.3 Outras Mudanças

Extended curved geometry support into the geometry accessor and some processing functions

Began migration of functions to the SQL-MM naming convention; using a spatial type (ST) prefix.

Adicionado suporte inicial ao PostgreSQL 8.3

## A.42 Versão 1.2.1

Data de Lançamento: 2007/01/11

This release provides bug fixes in PostgreSQL 8.2 support and some small performance enhancements.

### A.42.1 Mudanças

Fixed point-in-polygon shortcut bug in `Within()`.

Fixed PostgreSQL 8.2 NULL handling for indexes.

Updated RPM spec files.

Added short-circuit for `Transform()` in no-op case.

JDBC: Fixed JTS handling for multi-dimensional geometries (thanks to Thomas Marti for hint and partial patch). Additionally, now JavaDoc is compiled and packaged. Fixed classpath problems with GCJ. Fixed pgjdbc 8.2 compatibility, losing support for jdk 1.3 and older.

## A.43 Versão 1.2.0

Data de Lançamento: 2006/12/08

This release provides type definitions along with serialization/deserialization capabilities for SQL-MM defined curved geometries, as well as performance enhancements.

### A.43.1 Mudanças

Added curved geometry type support for serialization/deserialization

Added point-in-polygon shortcircuit to the `Contains` and `Within` functions to improve performance for these cases.

## A.44 Versão 1.1.6

Data de lançamento: 2006/11/02

This is a bugfix release, in particular fixing a critical error with GEOS interface in 64bit systems. Includes an updated of the SRS parameters and an improvement in reprojections (take Z in consideration). Upgrade is *encouraged*.

### A.44.1 Atualizando

If you are upgrading from release 1.0.3 or later follow the **soft upgrade** procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the **upgrade section** of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an **hard upgrade**.

---



### A.44.2 Correção de Erros

fixed CAPI change that broke 64-bit platforms

loader/dumper: fixed regression tests and usage output

Fixed setSRID() bug in JDBC, thanks to Thomas Marti

### A.44.3 Outras mudanças

use Z ordinate in reprojections

spatial\_ref\_sys.sql updated to EPSG 6.11.1

Simplified Version.config infrastructure to use a single pack of version variables for everything.

Include the Version.config in loader/dumper USAGE messages

Replace hand-made, fragile JDBC version parser with Properties

## A.45 Versão 1.1.5

Data de Lançamento: 2006/10/13

This is an bugfix release, including a critical segfault on win32. Upgrade is *encouraged*.

### A.45.1 Atualizando

If you are upgrading from release 1.0.3 or later follow the [soft upgrade](#) procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the [upgrade section](#) of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an [hard upgrade](#).

### A.45.2 Correção de Erros

Fixed MingW link error that was causing pgsq2shp to segfault on Win32 when compiled for PostgreSQL 8.2

fixed nullpointer Exception in Geometry.equals() method in Java

Added EJB3Spatial.odt to fulfill the GPL requirement of distributing the "preferred form of modification"

Removed obsolete synchronization from JDBC Jts code.

Updated heavily outdated README files for shp2pgsql/pgsq2shp by merging them with the manpages.

Fixed version tag in jdbc code that still said "1.1.3" in the "1.1.4" release.

### A.45.3 Novos Recursos

Added -S option for non-multi geometries to shp2pgsql

## A.46 Versão 1.1.4

Data de lançamento: 2006/09/27

This is an bugfix release including some improvements in the Java interface. Upgrade is *encouraged*.

---

### A.46.1 Atualizando

If you are upgrading from release 1.0.3 or later follow the [soft upgrade](#) procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the [upgrade section](#) of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an [hard upgrade](#).

### A.46.2 Correção de Erros

Corrigido o suporte para o PostgreSQL 8.2

Fixed bug in collect() function discarding SRID of input

Added SRID match check in MakeBox2d and MakeBox3d

Fixed regress tests to pass with GEOS-3.0.0

Improved pgsq2shp run concurrency.

### A.46.3 Mudanças para o Java

reworked JTS support to reflect new upstream JTS developers' attitude to SRID handling. Simplifies code and drops build depend on GNU trove.

Added EJB2 support generously donated by the "Geodetix s.r.l. Company"

Added EJB3 tutorial / examples donated by Norman Barker <nbarker@ittvis.com>

Reorganized java directory layout a little.

## A.47 Versão 1.1.3

Data de Lançamento: 2006/06/30

This is an bugfix release including also some new functionalities (most notably long transaction support) and portability enhancements. Upgrade is *encouraged*.

### A.47.1 Atualizando

If you are upgrading from release 1.0.3 or later follow the [soft upgrade](#) procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the [upgrade section](#) of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an [hard upgrade](#).

### A.47.2 Bug fixes / correctness

BUGFIX in distance(poly,poly) giving wrong results.

BUGFIX in pgsq2shp successful return code.

BUGFIX in shp2pgsql handling of MultiLine WKT.

BUGFIX in affine() failing to update bounding box.

WKT parser: forbidden construction of multigeometries with EMPTY elements (still supported for GEOMETRYCOLLECTION).

### A.47.3 New functionalities

NEW Long Transactions support.

NEW DumpRings() function.

NEW AsHEXEWKB(geom, XDRINDR) function.

### A.47.4 Mudanças para o JDBC

Improved regression tests: MultiPoint and scientific ordinates

Fixed some minor bugs in jdbc code

Added proper accessor functions for all fields in preparation of making those fields private later

### A.47.5 Outras mudanças

NEW regress test support for loader/dumper.

Added --with-proj-libdir and --with-geos-libdir configure switches.

Support for build Tru64 build.

Use Jade for generating documentation.

Don't link postgres to more libs than required.

Suporte inicial para o PostgreSQL 8.2.

## A.48 Versão 1.1.2

Data de Lançamento: 2006/03/30

This is an bugfix release including some new functions and portability enhancements. Upgrade is *encouraged*.

### A.48.1 Atualizando

If you are upgrading from release 1.0.3 or later follow the **soft upgrade** procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the **upgrade section** of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an **hard upgrade**.

### A.48.2 Correção de Erros

BUGFIX in SnapToGrid() computation of output bounding box

BUGFIX in EnforceRHR()

jdbc2 SRID handling fixes in JTS code

Fixed support for 64bit archs

---

### A.48.3 New functionalities

Regress tests can now be run *before* postgis installation

New affine() matrix transformation functions

New rotate{,X,Y,Z}() function

Old translating and scaling functions now use affine() internally

Embedded access control in estimated\_extent() for builds against pgsqll >= 8.0.0

### A.48.4 Outras mudanças

More portable ./configure script

Changed ./run\_test script to have more sane default behaviour

## A.49 Versão 1.1.1

Data de Lançamento: 2006/01/23

This is an important Bugfix release, upgrade is *highly recommended*. Previous version contained a bug in postgis\_restore.pl preventing **hard upgrade** procedure to complete and a bug in GEOS-2.2+ connector preventing GeometryCollection objects to be used in topological operations.

### A.49.1 Atualizando

If you are upgrading from release 1.0.3 or later follow the **soft upgrade** procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the **upgrade section** of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an **hard upgrade**.

### A.49.2 Correção de Erros

Fixed a premature exit in postgis\_restore.pl

BUGFIX in geometrycollection handling of GEOS-CAPI connector

Solaris 2.7 and MingW support improvements

BUGFIX in line\_locate\_point()

Fixed handling of postgresql paths

BUGFIX in line\_substring()

Added support for localized cluster in regress tester

### A.49.3 New functionalities

New Z and M interpolation in line\_substring()

New Z and M interpolation in line\_interpolate\_point()

added NumInteriorRing() alias due to OpenGIS ambiguity

---

## A.50 Versão 1.1.0

Data de lançamento: 2005/12/21

This is a Minor release, containing many improvements and new things. Most notably: build procedure greatly simplified; transform() performance drastically improved; more stable GEOS connectivity (CAPI support); lots of new functions; draft topology support.

It is *highly recommended* that you upgrade to GEOS-2.2.x before installing PostGIS, this will ensure future GEOS upgrades won't require a rebuild of the PostGIS library.

### A.50.1 Credits

This release includes code from Mark Cave Ayland for caching of proj4 objects. Markus Schaber added many improvements in his JDBC2 code. Alex Bodnaru helped with PostgreSQL source dependency relief and provided Debian specfiles. Michael Fuhr tested new things on Solaris arch. David Techer and Gerald Fenoy helped testing GEOS C-API connector. Hartmut Tschauner provided code for the azimuth() function. Devrim GUNDUZ provided RPM specfiles. Carl Anderson helped with the new area building functions. See the [credits](#) section for more names.

### A.50.2 Atualizando

If you are upgrading from release 1.0.3 or later you *DO NOT* need a dump/reload. Simply sourcing the new lwpostgis\_upgrade.sql script in all your existing databases will work. See the [soft upgrade](#) chapter for more information.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the [upgrade section](#) of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an [hard upgrade](#).

### A.50.3 Novas funções

scale() and transscale() companion methods to translate()

line\_substring()

line\_locate\_point()

M(point)

LineMerge(geometry)

shift\_longitude(geometry)

azimuth(geometry)

locate\_along\_measure(geometry, float8)

locate\_between\_measures(geometry, float8, float8)

SnapToGrid by point offset (up to 4d support)

BuildArea(any\_geometry)

OGC BdPolyFromText(linestring\_wkt, srid)

OGC BdMPolyFromText(linestring\_wkt, srid)

RemovePoint(linestring, offset)

ReplacePoint(linestring, offset, point)

#### **A.50.4 Correção de Erros**

Fixed memory leak in polygonize()

Fixed bug in lwgeom\_as\_anytype cast functions

Fixed USE\_GEOS, USE\_PROJ and USE\_STATS elements of postgis\_version() output to always reflect library state.

#### **A.50.5 Function semantic changes**

SnapToGrid doesn't discard higher dimensions

Changed Z() function to return NULL if requested dimension is not available

#### **A.50.6 Performance improvements**

Much faster transform() function, caching proj4 objects

Removed automatic call to fix\_geometry\_columns() in AddGeometryColumns() and update\_geometry\_stats()

#### **A.50.7 JDBC2 works**

Makefile improvements

JTS support improvements

Improved regression test system

Basic consistency check method for geometry collections

Support for (Hex)(E)wkb

Autoprobing DriverWrapper for HexWKB / EWKT switching

fix compile problems in ValueSetter for ancient jdk releases.

fix EWKT constructors to accept SRID=4711; representation

added preliminary read-only support for java2d geometries

#### **A.50.8 Outras coisas novas**

Full autoconf-based configuration, with PostgreSQL source dependency relief

GEOS C-API support (2.2.0 and higher)

Initial support for topology modelling

Debian and RPM specfiles

New lwpostgis\_upgrade.sql script

#### **A.50.9 Outras mudanças**

JTS support improvements

Stricter mapping between DBF and SQL integer and string attributes

Wider and cleaner regression test suite

old jdbc code removed from release

obsoleted direct use of postgis\_proc\_upgrade.pl

scripts version unified with release version

## A.51 Versão 1.0.6

Data de lançamento: 2005/12/06

Contém algumas pequenas correções e melhorias.

### A.51.1 Atualizando

If you are upgrading from release 1.0.3 or later you *DO NOT* need a dump/reload.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the [upgrade section](#) of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an [hard upgrade](#).

### A.51.2 Correção de Erros

Fixed palloc(0) call in collection deserializer (only gives problem with --enable-cassert)

Fixed bbox cache handling bugs

Fixed geom\_accum(NULL, NULL) segfault

Fixed segfault in addPoint()

Fixed short-allocation in lwcollection\_clone()

Fixed bug in segmentize()

Fixed bbox computation of SnapToGrid output

### A.51.3 Melhorias

Suporte inicial para o postgresql 8.2.

Added missing SRID mismatch checks in GEOS ops

## A.52 Versão 1.0.5

Data de lançamento: 2005/11/25

Contains memory-alignment fixes in the library, a segfault fix in loader's handling of UTF8 attributes and a few improvements and cleanups.



#### Note

Return code of shp2pgsql changed from previous releases to conform to unix standards (return 0 on success).

---

### A.52.1 Atualizando

If you are upgrading from release 1.0.3 or later you *DO NOT* need a dump/reload.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the [upgrade section](#) of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an [hard upgrade](#).

---

### A.52.2 Library changes

Fixed memory alignment problems

Fixed computation of null values fraction in analyzer

Fixed a small bug in the `getPoint4d_p()` low-level function

Speedup of serializer functions

Fixed a bug in `force_3dm()`, `force_3dz()` and `force_4d()`

### A.52.3 Loader changes

Fixed return code of `shp2pgsql`

Fixed back-compatibility issue in loader (load of null shapefiles)

Fixed handling of trailing dots in dbf numerical attributes

Segfault fix in `shp2pgsql` (utf8 encoding)

### A.52.4 Outras mudanças

Schema aware `postgis_proc_upgrade.pl`, support for `pgsql 7.2+`

New "Reporting Bugs" chapter in manual

## A.53 Versão 1.0.4

Data de Lançamento: 2005/09/09

Contains important bug fixes and a few improvements. In particular, it fixes a memory leak preventing successful build of GiST indexes for large spatial tables.

### A.53.1 Atualizando

If you are upgrading from release 1.0.3 you *DO NOT* need a dump/reload.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the [upgrade section](#) of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an [hard upgrade](#).

### A.53.2 Correção de Erros

Memory leak plugged in GiST indexing

Segfault fix in `transform()` handling of proj4 errors

Fixed some proj4 texts in `spatial_ref_sys` (missing `+proj`)

Loader: fixed string functions usage, reworked NULL objects check, fixed segfault on MULTILINESTRING input.

Fixed bug in `MakeLine` dimension handling

Fixed bug in `translate()` corrupting output bounding box

---



### A.53.3 Melhorias

Documentation improvements  
More robust selectivity estimator  
Minor speedup in distance()  
Minor cleanups  
GiST indexing cleanup  
Looser syntax acceptance in box3d parser

## A.54 Versão 1.0.3

Data de lançamento: 2005/08/08

Contains some bug fixes - *including a severe one affecting correctness of stored geometries* - and a few improvements.

### A.54.1 Atualizando

Due to a bug in a bounding box computation routine, the upgrade procedure requires special attention, as bounding boxes cached in the database could be incorrect.

An **hard upgrade** procedure (dump/reload) will force recomputation of all bounding boxes (not included in dumps). This is *required* if upgrading from releases prior to 1.0.0RC6.

If you are upgrading from versions 1.0.0RC6 or up, this release includes a perl script (utils/rebuild\_bbox\_caches.pl) to force recomputation of geometries' bounding boxes and invoke all operations required to propagate eventual changes in them (geometry statistics update, reindexing). Invoke the script after a make install (run with no args for syntax help). Optionally run utils/postgis\_proc\_upgrade.pl to refresh postgis procedures and functions signatures (see **Soft upgrade**).

### A.54.2 Correção de Erros

Severe bugfix in lwgeom's 2d bounding box computation  
Bugfix in WKT (-w) POINT handling in loader  
Bugfix in dumper on 64bit machines  
Bugfix in dumper handling of user-defined queries  
Bugfix in create\_undef.pl script

### A.54.3 Melhorias

Small performance improvement in canonical input function  
Minor cleanups in loader  
Support for multibyte field names in loader  
Improvement in the postgis\_restore.pl script  
New rebuild\_bbox\_caches.pl util script

## A.55 Versão 1.0.2

Data de lançamento: 2005/07/04

Contém algumas pequenas correções e melhorias.

---

### A.55.1 Atualizando

If you are upgrading from release 1.0.0RC6 or up you *DO NOT* need a dump/reload.

Upgrading from older releases requires a dump/reload. See the [upgrading](#) chapter for more informations.

### A.55.2 Correção de Erros

Fault tolerant btree ops

Memory leak plugged in pg\_error

Rtree index fix

Cleaner build scripts (avoided mix of CFLAGS and CXXFLAGS)

### A.55.3 Melhorias

New index creation capabilities in loader (-I switch)

Initial support for postgresql 8.1dev

## A.56 Versão 1.0.1

Release date: 2005/05/24

Contains a few bug fixes and some improvements.

### A.56.1 Atualizando

If you are upgrading from release 1.0.0RC6 or up you *DO NOT* need a dump/reload.

Upgrading from older releases requires a dump/reload. See the [upgrading](#) chapter for more informations.

### A.56.2 Library changes

BUGFIX in 3d computation of length\_spheroid()

BUGFIX in join selectivity estimator

### A.56.3 Other changes/additions

BUGFIX in shp2pgsql escape functions

better support for concurrent postgis in multiple schemas

documentation fixes

jdbc2: compile with "-target 1.2 -source 1.2" by default

NEW -k switch for postgresql2shp

NEW support for custom createdb options in postgis\_restore.pl

BUGFIX in postgresql2shp attribute names unicity enforcement

BUGFIX in Paris projections definitions

postgis\_restore.pl cleanups

---

## A.57 Versão 1.0.0

Release date: 2005/04/19

Final 1.0.0 release. Contains a few bug fixes, some improvements in the loader (most notably support for older postgis versions), and more docs.

### A.57.1 Atualizando

If you are upgrading from release 1.0.0RC6 you *DO NOT* need a dump/reload.

Upgrading from any other precedent release requires a dump/reload. See the [upgrading](#) chapter for more informations.

### A.57.2 Library changes

BUGFIX in transform() releasing random memory address

BUGFIX in force\_3dm() allocating less memory then required

BUGFIX in join selectivity estimator (defaults, leaks, tuplecount, sd)

### A.57.3 Other changes/additions

BUGFIX in shp2pgsql escape of values starting with tab or single-quote

NEW manual pages for loader/dumper

NEW shp2pgsql support for old (HWGEOM) postgis versions

NEW -p (prepare) flag for shp2pgsql

NEW manual chapter about OGC compliancy enforcement

NEW autoconf support for JTS lib

BUGFIX in estimator testers (support for LWGEOM and schema parsing)

## A.58 Versão 1.0.0RC6

Release date: 2005/03/30

Sixth release candidate for 1.0.0. Contains a few bug fixes and cleanups.

### A.58.1 Atualizando

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

### A.58.2 Library changes

BUGFIX in multi()

early return [when noop] from multi()

### A.58.3 Scripts changes

dropped {x,y}{min,max}(box2d) functions

---

#### A.58.4 Outras mudanças

BUGFIX in postgis\_restore.pl scrip

BUGFIX in dumper's 64bit support

### A.59 Release 1.0.0RC5

Release date: 2005/03/25

Fifth release candidate for 1.0.0. Contains a few bug fixes and a improvements.

#### A.59.1 Atualizando

If you are upgrading from release 1.0.0RC4 you *DO NOT* need a dump/reload.

Upgrading from any other precedent release requires a dump/reload. See the [upgrading](#) chapter for more informations.

#### A.59.2 Library changes

BUGFIX (segfaulting) in box3d computation (yes, another!).

BUGFIX (segfaulting) in estimated\_extent().

#### A.59.3 Outras mudanças

Small build scripts and utilities refinements.

Additional performance tips documented.

### A.60 Versão 1.0.0RC4

Data de Lançamento: 2005/03/18

Fourth release candidate for 1.0.0. Contains bug fixes and a few improvements.

#### A.60.1 Atualizando

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

#### A.60.2 Library changes

BUGFIX (segfaulting) in geom\_accum().

BUGFIX in 64bit architectures support.

BUGFIX in box3d computation function with collections.

NEW subselects support in selectivity estimator.

Early return from force\_collection.

Consistency check fix in SnapToGrid().

Box2d output changed back to 15 significant digits.

---

### A.60.3 Scripts changes

NEW `distance_sphere()` function.

Changed `get_proj4_from_srid` implementation to use PL/PGSQL instead of SQL.

### A.60.4 Outras mudanças

BUGFIX in loader and dumper handling of MultiLine shapes

BUGFIX in loader, skipping all but first hole of polygons.

jdbc2: code cleanups, Makefile improvements

FLEX and YACC variables set `*after*` `pgsql` `Makefile.global` is included and only if the `pgsql` `*stripped*` version evaluates to the empty string

Added already generated parser in release

Build scripts refinements

improved version handling, central `Version.config`

improvements in `postgis_restore.pl`

## A.61 Versão 1.0.0RC3

Data de lançamento: 2005/02/24

Third release candidate for 1.0.0. Contains many bug fixes and improvements.

### A.61.1 Atualizando

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

### A.61.2 Library changes

BUGFIX in `transform()`: missing SRID, better error handling.

BUGFIX in memory alignment handling

BUGFIX in `force_collection()` causing mapserver connector failures on simple (single) geometry types.

BUGFIX in `GeometryFromText()` missing to add a bbox cache.

reduced precision of `box2d` output.

prefixed `DEBUG` macros with `PGIS_` to avoid clash with `pgsql` one

plugged a leak in `GEOS2POSTGIS` converter

Reduced memory usage by early releasing query-context pallocated one.

### A.61.3 Scripts changes

BUGFIX in 72 index bindings.

BUGFIX in `probe_geometry_columns()` to work with PG72 and support multiple geometry columns in a single table

NEW `bool::text` cast

Some functions made `IMMUTABLE` from `STABLE`, for performance improvement.

---

### A.61.4 Mudanças para o JDBC

jdbc2: small patches, box2d/3d tests, revised docs and license.

jdbc2: bug fix and testcase in for pgjdbc 8.0 type autoregistration

jdbc2: Removed use of jdk1.4 only features to enable build with older jdk releases.

jdbc2: Added support for building against pg72jdbc2.jar

jdbc2: updated and cleaned makefile

jdbc2: added BETA support for jts geometry classes

jdbc2: Skip known-to-fail tests against older PostGIS servers.

jdbc2: Fixed handling of measured geometries in EWKT.

### A.61.5 Outras mudanças

new performance tips chapter in manual

documentation updates: pgsq172 requirement, lwpostgis.sql

few changes in autoconf

BUILDDATE extraction made more portable

fixed spatial\_ref\_sys.sql to avoid vacuuming the whole database.

spatial\_ref\_sys: changed Paris entries to match the ones distributed with 0.x.

## A.62 Versão 1.0.0RC2

Data de lançamento: 2005/01/26

Second release candidate for 1.0.0 containing bug fixes and a few improvements.

### A.62.1 Atualizando

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

### A.62.2 Library changes

BUGFIX in pointarray box3d computation

BUGFIX in distance\_spheroid definition

BUGFIX in transform() missing to update bbox cache

NEW jdbc driver (jdbc2)

GEOMETRYCOLLECTION(EMPTY) syntax support for backward compatibility

Faster binary outputs

Stricter OGC WKB/WKT constructors

### A.62.3 Scripts changes

More correct STABLE, IMMUTABLE, STRICT uses in lwpostgis.sql

stricter OGC WKB/WKT constructors

---

### A.62.4 Outras mudanças

Faster and more robust loader (both i18n and not)

Initial autoconf script

## A.63 Versão 1.0.0RC1

Data de lançamento: 2005/01/13

This is the first candidate of a major postgis release, with internal storage of postgis types redesigned to be smaller and faster on indexed queries.

### A.63.1 Atualizando

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

### A.63.2 Mudanças

Faster canonical input parsing.

Lossless canonical output.

EWKB Canonical binary IO with PG>73.

Support for up to 4d coordinates, providing lossless shapefile->postgis->shapefile conversion.

New function: UpdateGeometrySRID(), AsGML(), SnapToGrid(), ForceRHR(), estimated\_extent(), accum().

Vertical positioning indexed operators.

JOIN selectivity function.

More geometry constructors / editors.

PostGIS extension API.

UTF8 support in loader.

---