

**PostGIS 3.3.9dev 사용자
지침서**

**DEV (Tue 18 Feb 2025 12:35:24 PM UTC rev.
3.3.7-17-gaaf02e1c9)**

Contents

1	Introduction	1
1.1	Getting started	1
1.2	Installing PostGIS	2
1.3	Using the command line	2
1.4	Using the GUI	3
2	PostGIS	6
2.1	Overview	6
2.2	Getting started	7
2.2.1	Installing PostGIS	8
2.2.2	Using the command line	8
2.2.3	Using the GUI	10
2.2.4	PostGIS Extensions	12
2.2.5	PostGIS Data	13
2.2.6	PostGIS Geocoding	15
2.2.7	PostGIS Routing	19
2.3	PostGIS Functions	20
2.3.1	Regex::Assemble	20
2.4	Tiger Geocoder	20
2.4.1	TIGER Data	21
2.4.1.1	TIGER Data	21
2.4.2	TIGER Data	24
2.4.3	TIGER Data	25
2.4.4	Tiger Data	25
2.4.5	Tiger Geocoder	26
2.5	PostGIS Routing	28

3	PostGIS Administration	29
3.1	Performance Tuning	29
3.1.1	Startup	29
3.1.2	Runtime	30
3.2	Configuring raster support	30
3.3	Enabling spatial support	31
3.3.1	Spatially enable database using EXTENSION	31
3.3.2	Spatially enable database without using EXTENSION (discouraged)	31
3.3.3	Create a spatially-enabled database from a template	32
3.4	Upgrading spatial databases	32
3.4.1	Soft upgrade	32
3.4.1.1	Soft Upgrade 9.1+ using extensions	32
3.4.1.2	Soft Upgrade Pre 9.1+ or without extensions	33
3.4.2	Hard upgrade	34
4	Data Management	36
4.1	GIS Data Types	36
4.1.1	OGC Geometry	36
4.1.1.1	Point	37
4.1.1.2	LineString	37
4.1.1.3	LinearRing	37
4.1.1.4	Polygon	37
4.1.1.5	MultiPoint	37
4.1.1.6	MultiLineString	37
4.1.1.7	MultiPolygon	38
4.1.1.8	GeometryCollection	38
4.1.1.9	PolyhedralSurface	38
4.1.1.10	Triangle	38
4.1.1.11	TIN	38
4.1.2	SQL-MM Part 3	38
4.1.2.1	CircularString	39
4.1.2.2	CompoundCurve	39
4.1.2.3	CurvePolygon	39
4.1.2.4	MultiCurve	40
4.1.2.5	MultiSurface	40
4.1.3	OpenGIS WKB and WKT	40
4.2	Geometry Data Type	41
4.2.1	OpenGIS WKB and WKT	42
4.3	PostGIS Raster	43

4.3.1		44
4.3.2	PostGIS	45
4.3.3		47
4.3.4		48
4.4	Geometry Validation	49
4.4.1	Simple Geometry	49
4.4.2	Valid Geometry	51
4.4.3	Managing Validity	53
4.5	SPATIAL_REF_SYS	
4.5.1	SPATIAL_REF_SYS Table	55
4.5.2	SPATIAL_REF_SYS	
4.6		57
4.6.1		57
4.6.2	The GEOMETRY_COLUMNS VIEW	58
4.6.3	geometry_columns	
4.7	GIS	61
4.7.1	SQL	
4.7.2	shp2pgsql: ESRI shapefile	62
4.8		65
4.8.1	SQL	
4.8.2		67
4.9		67
4.9.1	GiST	68
4.9.2	GiST	69
4.9.3	GiST	71
4.9.4		71
5	Spatial Queries	73
5.1	Determining Spatial Relationships	73
5.1.1	Dimensionally Extended 9-Intersection Model	73
5.1.2	Named Spatial Relationships	75
5.1.3	General Spatial Relationships	76
5.2	Using Spatial Indexes	78
5.3	Examples of Spatial SQL	78

6	Geometry Constructors	81
6.1	Geometry Constructors	81
6.1.1	Point Constructors	81
6.1.2	Line Constructors	82
6.2	Area Constructors	83
6.3	Other Constructors	83
7	PostGIS Functions	
7.1	MapServer Functions	85
7.1.1	MapServer Functions	85
7.1.2	FAQ	87
7.1.3	Other Functions	88
7.1.4	Other Functions	89
7.2	Java Functions (JDBC)	91
7.3	C Functions (libpq)	93
7.3.1	C Functions (libpq)	93
7.3.2	C Functions (libpq)	93
8	PostGIS Reference	94
8.1	PostgreSQL PostGIS Geometry/Geography/Box Functions	94
8.1.1	box2d	94
8.1.2	box3d	95
8.1.3	geometry	96
8.1.4	geometry_dump	96
8.1.5	geography	96
8.2	Geometry Functions	97
8.2.1	AddGeometryColumn	97
8.2.2	DropGeometryColumn	100
8.2.3	DropGeometryTable	101
8.2.4	Find_SRID	101
8.2.5	Populate_Geometry_Columns	102
8.2.6	UpdateGeometrySRID	104
8.3	Geometry Constructors (constructor)	105
8.3.1	ST_GeomCollFromText	105
8.3.2	ST_LineFromMultiPoint	107
8.3.3	ST_MakeEnvelope	108
8.3.4	ST_MakeLine	109
8.3.5	ST_MakePoint	110

8.3.6	ST_MakePointM	111
8.3.7	ST_MakePolygon	112
8.3.8	ST_Point	115
8.3.9	ST_Point	116
8.3.10	ST_Point	117
8.3.11	ST_Point	117
8.3.12	ST_Polygon	118
8.3.13	ST_MakeEnvelope	119
8.3.14	ST_HexagonGrid	119
8.3.15	ST_Hexagon	122
8.3.16	ST_SquareGrid	123
8.3.17	ST_Square	124
8.3.18	ST_Letters	125
8.4	ST_Geometry (accessor)	126
8.4.1	ST_Geometry (accessor)	126
8.4.2	ST_Boundary	127
8.4.3	ST_BoundingDiagonal	130
8.4.4	ST_CoordDim	131
8.4.5	ST_Dimension	132
8.4.6	ST_Dump	132
8.4.7	ST_NumPoints	134
8.4.8	ST_NumPoints	138
8.4.9	ST_NRings	140
8.4.10	ST_EndPoint	141
8.4.11	ST_Envelope	143
8.4.12	ST_ExteriorRing	144
8.4.13	ST_GeometryN	145
8.4.14	ST_GeometryType	148
8.4.15	ST_HasArc	149
8.4.16	ST_InteriorRingN	150
8.4.17	ST_IsClosed	150
8.4.18	ST_IsCollection	152
8.4.19	ST_IsEmpty	153
8.4.20	ST_IsPolygonCCW	155
8.4.21	ST_IsPolygonCW	155
8.4.22	ST_IsRing	156
8.4.23	ST_IsSimple	157
8.4.24	ST_M	158
8.4.25	ST_MemSize	159

8.4.26	ST_NDims	160
8.4.27	ST_NPoints	160
8.4.28	ST_NRings	161
8.4.29	ST_NumGeometries	162
8.4.30	ST_NumInteriorRings	163
8.4.31	ST_NumInteriorRing	164
8.4.32	ST_NumPatches	164
8.4.33	ST_NumPoints	165
8.4.34	ST_PatchN	166
8.4.35	ST_PointN	167
8.4.36	ST_Points	169
8.4.37	ST_StartPoint	169
8.4.38	ST_Summary	171
8.4.39	ST_X	172
8.4.40	ST_Y	173
8.4.41	ST_Z	173
8.4.42	ST_Zmflag	174
8.5	ST_AddPoint (editor)	175
8.5.1	ST_AddPoint	175
8.5.2	ST_CollectionExtract	176
8.5.3	ST_CollectionHomogenize	177
8.5.4	ST_CurveToLine	179
8.5.5	ST_Scroll	181
8.5.6	ST_FlipCoordinates	182
8.5.7	ST_Force2D	183
8.5.8	ST_Force3D	183
8.5.9	ST_Force3DZ	184
8.5.10	ST_Force3DM	185
8.5.11	ST_Force4D	186
8.5.12	ST_ForcePolygonCCW	187
8.5.13	ST_ForceCollection	187
8.5.14	ST_ForcePolygonCW	189
8.5.15	ST_ForceSFS	189
8.5.16	ST_ForceRHR	189
8.5.17	ST_ForceCurve	190
8.5.18	ST_LineToCurve	191
8.5.19	ST_Multi	193
8.5.20	ST_Normalize	193
8.5.21	ST_QuantizeCoordinates	194

8.5.22	ST_RemovePoint	196
8.5.23	ST_RemoveRepeatedPoints	197
8.5.24	ST_Reverse	197
8.5.25	ST_Segmentize	198
8.5.26	ST_SetPoint	199
8.5.27	ST_ShiftLongitude	200
8.5.28	ST_WrapX	201
8.5.29	ST_SnapToGrid	202
8.5.30	ST_Snap	204
8.5.31	ST_SwapOrdinates	208
8.6	Geometry Validation	208
8.6.1	ST_IsValid	208
8.6.2	ST_IsValidDetail	209
8.6.3	ST_IsValidReason	211
8.6.4	ST_MakeValid	212
8.7	Spatial Reference System Functions	217
8.7.1	ST_SetSRID	217
8.7.2	ST_SRID	218
8.7.3	ST_Transform	219
8.8	Geometry Input	221
8.8.1	Well-Known Text (WKT)	221
8.8.1.1	ST_BdPolyFromText	221
8.8.1.2	ST_BdMPolyFromText	222
8.8.1.3	ST_GeogFromText	222
8.8.1.4	ST_GeographyFromText	223
8.8.1.5	ST_GeomCollFromText	223
8.8.1.6	ST_GeomFromEWKT	224
8.8.1.7	ST_GeomFromMARC21	226
8.8.1.8	ST_GeometryFromText	228
8.8.1.9	ST_GeomFromText	228
8.8.1.10	ST_LineFromText	230
8.8.1.11	ST_MLineFromText	231
8.8.1.12	ST_MPointFromText	232
8.8.1.13	ST_MPolyFromText	232
8.8.1.14	ST_PointFromText	233
8.8.1.15	ST_PolygonFromText	234
8.8.1.16	ST_WKTToSQL	235
8.8.2	Well-Known Binary (WKB)	236
8.8.2.1	ST_GeogFromWKB	236

8.8.2.2	ST_GeomFromEWKB	237
8.8.2.3	ST_GeomFromWKB	238
8.8.2.4	ST_LineFromWKB	239
8.8.2.5	ST_LinestringFromWKB	240
8.8.2.6	ST_PointFromWKB	241
8.8.2.7	ST_WKBToSQL	242
8.8.3	Other Formats	243
8.8.3.1	ST_Box2dFromGeoHash	243
8.8.3.2	ST_GeomFromGeoHash	243
8.8.3.3	ST_GeomFromGML	244
8.8.3.4	ST_GeomFromGeoJSON	247
8.8.3.5	ST_GeomFromKML	248
8.8.3.6	ST_GeomFromTWKB	249
8.8.3.7	ST_GMLToSQL	250
8.8.3.8	ST_LineFromEncodedPolyline	250
8.8.3.9	ST_PointFromGeoHash	251
8.8.3.10	ST_FromFlatGeobufToTable	252
8.8.3.11	ST_FromFlatGeobuf	252
8.9	Geometry Output	253
8.9.1	Well-Known Text (WKT)	253
8.9.1.1	ST_AsEWKT	253
8.9.1.2	ST_AsText	254
8.9.2	Well-Known Binary (WKB)	255
8.9.2.1	ST_AsBinary	255
8.9.2.2	ST_AsEWKB	257
8.9.2.3	ST_AsHEXEWKB	258
8.9.3	Other Formats	259
8.9.3.1	ST_AsEncodedPolyline	259
8.9.3.2	ST_AsFlatGeobuf	260
8.9.3.3	ST_AsGeobuf	260
8.9.3.4	ST_AsGeoJSON	261
8.9.3.5	ST_AsGML	263
8.9.3.6	ST_AsKML	267
8.9.3.7	ST_AsLatLonText	269
8.9.3.8	ST_AsMARC21	270
8.9.3.9	ST_AsMVTGeom	272
8.9.3.10	ST_AsMVT	273
8.9.3.11	ST_AsSVG	274
8.9.3.12	ST_AsTWKB	275

8.9.3.13	ST_AsX3D	277
8.9.3.14	ST_GeoHash	281
8.10	(operator)	283
8.10.1	Bounding Box Operators	283
8.10.1.1	&&	283
8.10.1.2	&&(geometry,box2df)	284
8.10.1.3	&&(box2df,geometry)	284
8.10.1.4	&&(box2df,box2df)	285
8.10.1.5	&&&	286
8.10.1.6	&&&(geometry,gidx)	287
8.10.1.7	&&&(gidx,geometry)	288
8.10.1.8	&&&(gidx,gidx)	289
8.10.1.9	&<	289
8.10.1.10	&<	290
8.10.1.11	&>	291
8.10.1.12	<<	292
8.10.1.13	<<	293
8.10.1.14	=	294
8.10.1.15	>>	295
8.10.1.16	@	296
8.10.1.17	@(geometry,box2df)	297
8.10.1.18	@(box2df,geometry)	298
8.10.1.19	@(box2df,box2df)	298
8.10.1.20	&>	299
8.10.1.21	>>	300
8.10.1.22	~	301
8.10.1.23	~(geometry,box2df)	301
8.10.1.24	~(box2df,geometry)	302
8.10.1.25	~(box2df,box2df)	303
8.10.1.26	~=	304
8.10.2	(operator)	304
8.10.2.1	<->	304
8.10.2.2	=	307
8.10.2.3	<#>	308
8.10.2.4	<<->>	310
8.10.2.5	<<#>>	310
8.11	Spatial Relationships	311
8.11.1	Topological Relationships	311
8.11.1.1	ST_3DIntersects	311

8.11.1.2	ST_Contains	312
8.11.1.3	ST_ContainsProperly	316
8.11.1.4	ST_CoveredBy	317
8.11.1.5	ST_Covers	318
8.11.1.6	ST_Crosses	320
8.11.1.7	ST_Disjoint	322
8.11.1.8	ST_Equals	323
8.11.1.9	ST_Intersects	324
8.11.1.10	ST_LineCrossingDirection	325
8.11.1.11	ST_OrderingEquals	328
8.11.1.12	ST_Overlaps	329
8.11.1.13	ST_Relate	332
8.11.1.14	ST_RelateMatch	334
8.11.1.15	ST_Touches	335
8.11.1.16	ST_Within	337
8.11.2	Distance Relationships	338
8.11.2.1	ST_3DDWithin	338
8.11.2.2	ST_3DDFullyWithin	339
8.11.2.3	ST_DFullyWithin	340
8.11.2.4	ST_DWithin	341
8.11.2.5	ST_PointInsideCircle	342
8.12	Measurement Functions	343
8.12.1	ST_Area	343
8.12.2	ST_Azimuth	345
8.12.3	ST_Angle	346
8.12.4	ST_ClosestPoint	348
8.12.5	ST_3DClosestPoint	349
8.12.6	ST_Distance	351
8.12.7	ST_3DDistance	352
8.12.8	ST_DistanceSphere	354
8.12.9	ST_DistanceSpheroid	355
8.12.10	ST_FrechetDistance	356
8.12.11	ST_HausdorffDistance	357
8.12.12	ST_Length	358
8.12.13	ST_Length2D	360
8.12.14	ST_3DLength	361
8.12.15	ST_LengthSpheroid	361
8.12.16	ST_LongestLine	363
8.12.17	ST_3DLongestLine	365

8.12.18 ST_MaxDistance	366
8.12.19 ST_3DMaxDistance	367
8.12.20 ST_MinimumClearance	368
8.12.21 ST_MinimumClearanceLine	369
8.12.22 ST_Perimeter	370
8.12.23 ST_Perimeter2D	372
8.12.24 ST_3DPerimeter	372
8.12.25 ST_Project	373
8.12.26 ST_ShortestLine	374
8.12.27 ST_3DShortestLine	375
8.13 Overlay Functions	377
8.13.1 ST_ClipByBox2D	377
8.13.2 ST_Difference	377
8.13.3 ST_Intersection	379
8.13.4 ST_MemUnion	381
8.13.5 ST_Node	382
8.13.6 ST_Split	383
8.13.7 ST_Subdivide	385
8.13.8 ST_SymDifference	387
8.13.9 ST_UnaryUnion	389
8.13.10 ST_Union	389
8.14 도형 공간 처리	392
8.14.1 ST_Buffer	392
8.14.2 ST_BuildArea	396
8.14.3 ST_Centroid	397
8.14.4 ST_ChaikinSmoothing	399
8.14.5 ST_ConcaveHull	400
8.14.6 ST_ConvexHull	403
8.14.7 ST_DelaunayTriangles	404
8.14.8 ST_FilterByM	409
8.14.9 ST_GeneratePoints	410
8.14.10 ST_GeometricMedian	410
8.14.11 ST_LineMerge	412
8.14.12 ST_MaximumInscribedCircle	414
8.14.13 ST_MinimumBoundingCircle	416
8.14.14 ST_MinimumBoundingRadius	418
8.14.15 ST_OrientedEnvelope	418
8.14.16 ST_OffsetCurve	420
8.14.17 ST_PointOnSurface	423

8.14.18 ST_Polygonize	425
8.14.19 ST_ReducePrecision	426
8.14.20 ST_SharedPaths	428
8.14.21 ST_Simplify	430
8.14.22 ST_SimplifyPreserveTopology	431
8.14.23 ST_SimplifyPolygonHull	432
8.14.24 ST_SimplifyVW	434
8.14.25 ST_SetEffectiveArea	435
8.14.26 ST_TriangulatePolygon	437
8.14.27 ST_VoronoiLines	438
8.14.28 ST_VoronoiPolygons	439
8.15 Affine Transformations	442
8.15.1 ST_Affine	442
8.15.2 ST_Rotate	444
8.15.3 ST_RotateX	445
8.15.4 ST_RotateY	446
8.15.5 ST_RotateZ	446
8.15.6 ST_Scale	448
8.15.7 ST_Translate	449
8.15.8 ST_TransScale	450
8.16 Clustering Functions	451
8.16.1 ST_ClusterDBSCAN	451
8.16.2 ST_ClusterIntersecting	454
8.16.3 ST_ClusterKMeans	454
8.16.4 ST_ClusterWithin	456
8.17 Bounding Box Functions	457
8.17.1 Box2D	457
8.17.2 Box3D	458
8.17.3 ST_EstimatedExtent	459
8.17.4 ST_Expand	459
8.17.5 ST_Extent	461
8.17.6 ST_3DExtent	462
8.17.7 ST_MakeBox2D	463
8.17.8 ST_3DMakeBox	464
8.17.9 ST_XMax	464
8.17.10 ST_XMin	465
8.17.11 ST_YMax	466
8.17.12 ST_YMin	467
8.17.13 ST_ZMax	468

8.17.14 ST_ZMin	469
8.18 ST_LineInterpolatePoint (Linear Referencing)	470
8.18.1 ST_LineInterpolatePoint	470
8.18.2 ST_LineInterpolatePoint	472
8.18.3 ST_LineInterpolatePoints	473
8.18.4 ST_LineLocatePoint	474
8.18.5 ST_LineSubstring	475
8.18.6 ST_LocateAlong	477
8.18.7 ST_LocateBetween	478
8.18.8 ST_LocateBetweenElevations	480
8.18.9 ST_InterpolatePoint	480
8.18.10 ST_AddMeasure	481
8.19 Trajectory Functions	482
8.19.1 ST_IsValidTrajectory	482
8.19.2 ST_ClosestPointOfApproach	483
8.19.3 ST_DistanceCPA	484
8.19.4 ST_CPAWithin	485
8.20 SFCGAL	485
8.20.1 postgis_sfcgal_version	485
8.20.2 postgis_sfcgal_version	486
8.20.3 ST_3DArea	486
8.20.4 ST_3DConvexHull	487
8.20.5 ST_3DIntersection	489
8.20.6 ST_3DDifference	491
8.20.7 ST_3DUnion	492
8.20.8 ST_AlphaShape	493
8.20.9 ST_ApproximateMedialAxis	496
8.20.10 ST_ConstrainedDelaunayTriangles	497
8.20.11 ST_Extrude	498
8.20.12 ST_ForceLHR	500
8.20.13 ST_IsPlanar	500
8.20.14 ST_IsSolid	501
8.20.15 ST_MakeSolid	501
8.20.16 ST_MinkowskiSum	502
8.20.17 ST_OptimalAlphaShape	504
8.20.18 ST_Orientation	506
8.20.19 ST_StraightSkeleton	507
8.20.20 ST_Tessellate	508
8.20.21 ST_Volume	510

8.21	(Long Transaction)	511
8.21.1	AddAuth	511
8.21.2	CheckAuth	512
8.21.3	DisableLongTransactions	513
8.21.4	EnableLongTransactions	513
8.21.5	LockRow	514
8.21.6	UnlockRows	515
8.22	Version Functions	515
8.22.1	PostGIS_Extensions_Upgrade	515
8.22.2	PostGIS_Full_Version	516
8.22.3	PostGIS_GEOS_Version	517
8.22.4	PostGIS_Liblwgeom_Version	517
8.22.5	PostGIS_LibXML_Version	518
8.22.6	PostGIS_Lib_Build_Date	518
8.22.7	PostGIS_Lib_Version	519
8.22.8	PostGIS_PROJ_Version	519
8.22.9	PostGIS_Wagyu_Version	520
8.22.10	PostGIS_Scripts_Build_Date	520
8.22.11	PostGIS_Scripts_Installed	521
8.22.12	PostGIS_Scripts_Released	521
8.22.13	PostGIS_Version	522
8.23	PostGIS GUC(Grand Unified Custom Variable)	522
8.23.1	postgis.backend	522
8.23.2	postgis.gdal_datapath	523
8.23.3	postgis.gdal_enabled_drivers	524
8.23.4	postgis.enable_outdb_rasters	526
8.23.5	postgis.gdal_vsi_options	527
8.24	Troubleshooting Functions	527
8.24.1	PostGIS_AddBBox	527
8.24.2	PostGIS_DropBBox	528
8.24.3	PostGIS_HasBBox	529
9	PostGIS	530
10	(topology)	538
10.1		539
10.1.1	getfaceedges_returntype	539
10.1.2	TopoGeometry	539
10.1.3	validatetopology_returntype	540

10.2	TopoElement	540
10.2.1	TopoElement	540
10.2.2	TopoElementArray	541
10.3	TopoGeometry	542
10.3.1	AddTopoGeometryColumn	542
10.3.2	DropTopology	543
10.3.3	DropTopoGeometryColumn	544
10.3.4	Populate_Topology_Layer	545
10.3.5	TopologySummary	546
10.3.6	ValidateTopology	546
10.3.7	ValidateTopologyRelation	549
10.3.8	FindTopology	549
10.3.9	FindLayer	550
10.4	Topology Statistics Management	550
10.5	TopoGeo	550
10.5.1	CreateTopology	550
10.5.2	CopyTopology	551
10.5.3	ST_InitTopoGeo	552
10.5.4	ST_CreateTopoGeo	553
10.5.5	TopoGeo_AddPoint	554
10.5.6	TopoGeo_AddLineString	554
10.5.7	TopoGeo_AddPolygon	555
10.6	ST_Iso	555
10.6.1	ST_AddIsoNode	555
10.6.2	ST_AddIsoEdge	556
10.6.3	ST_AddEdgeNewFaces	557
10.6.4	ST_AddEdgeModFace	558
10.6.5	ST_RemEdgeNewFace	558
10.6.6	ST_RemEdgeModFace	559
10.6.7	ST_ChangeEdgeGeom	560
10.6.8	ST_ModEdgeSplit	561
10.6.9	ST_ModEdgeHeal	562
10.6.10	ST_NewEdgeHeal	562
10.6.11	ST_MoveIsoNode	563
10.6.12	ST_NewEdgesSplit	564
10.6.13	ST_RemoveIsoNode	565
10.6.14	ST_RemoveIsoEdge	565
10.7	ST_GetEdgeByPoint	566
10.7.1	GetEdgeByPoint	566

10.7.2	GetFaceByPoint	567
10.7.3	GetFaceContainingPoint	568
10.7.4	GetNodeByPoint	568
10.7.5	GetTopologyID	569
10.7.6	GetTopologySRID	569
10.7.7	GetTopologyName	570
10.7.8	ST_GetFaceEdges	571
10.7.9	ST_GetFaceGeometry	572
10.7.10	GetRingEdges	572
10.7.11	GetNodeEdges	573
10.8	ST_Polygonize	574
10.8.1	Polygonize	574
10.8.2	AddNode	575
10.8.3	AddEdge	576
10.8.4	AddFace	577
10.8.5	ST_Simplify	579
10.8.6	RemoveUnusedPrimitives	580
10.9	TopoGeometry	580
10.9.1	CreateTopoGeom	580
10.9.2	toTopoGeom	582
10.9.3	TopoElementArray_Agg	584
10.10	TopoGeometry	584
10.10.1	clearTopoGeom	584
10.10.2	TopoGeom_addElement	585
10.10.3	TopoGeom_remElement	585
10.10.4	TopoGeom_addTopoGeom	586
10.10.5	toTopoGeom	586
10.11	TopoGeometry	587
10.11.1	GetTopoGeomElementArray	587
10.11.2	GetTopoGeomElements	587
10.11.3	ST_SRID	588
10.12	TopoGeometry	588
10.12.1	AsGML	588
10.12.2	AsTopoJSON	591
10.13	ST_Equals	593
10.13.1	Equals	593
10.13.2	ST_Equals	594
10.14	Importing and exporting Topologies	594
10.14.1	Using the Topology exporter	595
10.14.2	Using the Topology importer	595

11	래스터 데이터의 관리 쿼리 및 응용	596
11.1	래스터 로드 및 생성	596
11.1.1	raster2pgsql을 이용해 래스터를 로드생성하기	602
11.1.2	PostGIS 래스터 함수를 이용해 래스생성하기	603
11.1.3	Using "out db" cloud rasters	604
11.2	래스터 󌹴탈로그	604
11.2.1	래스터 열 󌹴탈로그	604
11.2.2	래스터 오஄뷰	606
11.3	PostGIS 래스터를 이용하는 사용자 지정 응용 프로그램 빌드하기 608	
11.3.1	다른 래스터 함수와 함께 ST_AsPNG를 이용해서 PHP 예제를 출력 하기	609
11.3.2	다른 래스터 함수와 함께 ST_AsPNG를 이용해서 ASP.NET C# 예제를 출력 하기	612
11.3.3	래스터 쿼리를 이미지 파일력 하는 Java 콘솔 응용 프로그력 하기	612
11.3.4	PLPython을 이용해서 SQL을 통해 이미덤프하기	612
11.3.5	PSQL을 이용해서 래스터 출력하	
12	래스트 참조򻬸서	614
12.1	래스터 지원 데이터형	615
12.1.1	geomval	615
12.1.2	addbandarg	615
12.1.3	rastbandarg	616
12.1.4	raster	616
12.1.5	reclassarg	617
12.1.6	summarystats	618
12.1.7	unionarg	618
12.2	래스터 관리	619
12.2.1	AddRasterConstraints	619
12.2.2	DropRasterConstraints	621
12.2.3	AddOverviewConstraints	622
12.2.4	DropOverviewConstraints	623
12.2.5	PostGIS_GDAL_Version	624
12.2.6	PostGIS_Raster_Lib_Build_Date	624
12.2.7	PostGIS_Raster_Lib_Version	625
12.2.8	ST_GDALDrivers	625
12.2.9	ST_Count	630

12.2.10 ST_MakeEmptyRaster	631
12.2.11 UpdateRasterSRID	632
12.2.12 ST_CreateOverview	632
12.3 ST_Geometry (constructor)	633
12.3.1 ST_AddBand	633
12.3.2 ST_AsRaster	636
12.3.3 ST_Band	639
12.3.4 ST_MakeEmptyCoverage	641
12.3.5 ST_MakeEmptyRaster	643
12.3.6 ST_Tile	644
12.3.7 ST_Retile	646
12.3.8 ST_FromGDALRaster	647
12.4 ST_Geometry (accessor)	648
12.4.1 ST_GeoReference	648
12.4.2 ST_Height	649
12.4.3 ST_IsEmpty	649
12.4.4 ST_MemSize	650
12.4.5 ST_MetaData	651
12.4.6 ST_NumBands	652
12.4.7 ST_PixelHeight	652
12.4.8 ST_PixelWidth	653
12.4.9 ST_ScaleX	655
12.4.10 ST_ScaleY	655
12.4.11 ST_RasterToWorldCoord	656
12.4.12 ST_RasterToWorldCoordX	657
12.4.13 ST_RasterToWorldCoordY	658
12.4.14 ST_Rotation	659
12.4.15 ST_SkewX	660
12.4.16 ST_SkewY	661
12.4.17 ST_SRID	661
12.4.18 ST_Summary	662
12.4.19 ST_UpperLeftX	663
12.4.20 ST_UpperLeftY	663
12.4.21 ST_Width	664
12.4.22 ST_WorldToRasterCoord	665
12.4.23 ST_WorldToRasterCoordX	665
12.4.24 ST_WorldToRasterCoordY	666
12.5 ST_Geometry (array)	667
12.5.1 ST_BandMetaData	667

12.5.2	ST_BandNoDataValue	669
12.5.3	ST_BandIsNoData	669
12.5.4	ST_BandPath	671
12.5.5	ST_BandFileSize	672
12.5.6	ST_BandFileTimestamp	672
12.5.7	ST_BandPixelType	673
12.5.8	ST_PixelOfValue	673
12.5.9	ST_HasNoBand	674
12.6	ST_PixelAsPolygon	675
12.6.1	ST_PixelAsPolygons	676
12.6.2	ST_PixelAsPoint	677
12.6.3	ST_PixelAsPoints	677
12.6.4	ST_PixelAsCentroid	679
12.6.5	ST_PixelAsCentroids	679
12.6.6	ST_Value	680
12.6.7	ST_NearestValue	684
12.6.8	ST_SetSkew	686
12.6.9	ST_SetSkew	687
12.6.10	ST_Neighborhood	688
12.6.11	ST_SetValue	691
12.6.12	ST_SetValues	692
12.6.13	ST_DumpValues	700
12.6.14	ST_PixelOfValue	701
12.7	ST_SetGeoReference	703
12.7.1	ST_SetRotation	704
12.7.2	ST_SetScale	705
12.7.3	ST_SetSkew	706
12.7.4	ST_SetSRID	707
12.7.5	ST_SetUpperLeft	708
12.7.6	ST_Resample	708
12.7.7	ST_Rescale	710
12.7.8	ST_Reskew	711
12.7.9	ST_SnapToGrid	713
12.7.10	ST_Resize	714
12.7.11	ST_Transform	716
12.8	ST_SetBandNoDataValue	719
12.8.1		

12.8.2	ST_SetBandIsNoData	720
12.8.3	ST_SetBandPath	722
12.8.4	ST_SetBandIndex	723
12.9	ST_Band	725
12.9.1	ST_Count	725
12.9.2	ST_CountAgg	726
12.9.3	ST_Histogram	727
12.9.4	ST_Quantile	729
12.9.5	ST_SummaryStats	731
12.9.6	ST_SummaryStatsAgg	733
12.9.7	ST_ValueCount	735
12.10	Raster Inputs	738
12.10.1	ST_RastFromWKB	738
12.10.2	ST_RastFromHexWKB	739
12.11	ST_AsBinary/ST_AsWKB	739
12.11.1	ST_AsBinary/ST_AsWKB	739
12.11.2	ST_AsHexWKB	740
12.11.3	ST_AsGDALRaster	741
12.11.4	ST_AsJPEG	742
12.11.5	ST_AsPNG	744
12.11.6	ST_AsTIFF	745
12.12	ST_Clip	746
12.12.1	ST_Clip	746
12.12.2	ST_ColorMap	750
12.12.3	ST_Grayscale	754
12.12.4	ST_Intersection	756
12.12.5	ST_MapAlgebraExpr	758
12.12.6	ST_MapAlgebraExpr	766
12.12.7	ST_MapAlgebraExpr	769
12.12.8	ST_MapAlgebraExpr	773
12.12.9	ST_MapAlgebraFct	778
12.12.10	ST_MapAlgebraFct	783
12.12.11	ST_MapAlgebraFctNgb	787
12.12.12	ST_Reclass	790
12.12.13	ST_Union	792
12.13	ST_Distinct4ma	794
12.13.1	ST_Distinct4ma	794
12.13.2	ST_InvDistWeight4ma	795
12.13.3	ST_Max4ma	796

12.13.4 ST_Mean4ma	797
12.13.5 ST_Min4ma	799
12.13.6 ST_MinDist4ma	800
12.13.7 ST_Range4ma	800
12.13.8 ST_StdDev4ma	802
12.13.9 ST_Sum4ma	803
12.14 ST_Area	804
12.14.1 ST_Aspect	804
12.14.2 ST_HillShade	806
12.14.3 ST_Roughness	808
12.14.4 ST_Slope	809
12.14.5 ST_TPI	811
12.14.6 ST_TRI	811
12.15 ST_Box3D	812
12.15.1 Box3D	812
12.15.2 ST_ConvexHull	813
12.15.3 ST_DumpAsPolygons	814
12.15.4 ST_Envelope	816
12.15.5 ST_MinConvexHull	816
12.15.6 ST_Polygon	817
12.16 ST_Equals	819
12.16.1 &&	819
12.16.2 &<	820
12.16.3 &>	821
12.16.4 = ...	821
12.16.5 @	822
12.16.6 ~=	823
12.16.7 ~	823
12.17 ST_Contains	824
12.17.1 ST_Contains	824
12.17.2 ST_ContainsProperly	825
12.17.3 ST_Covers	826
12.17.4 ST_CoveredBy	827
12.17.5 ST_Disjoint	828
12.17.6 ST_Intersects	830
12.17.7 ST_Overlaps	831
12.17.8 ST_Touches	832
12.17.9 ST_SameAlignment	833

12.17.1	ST_NotSameAlignmentReason	834
12.17.1	ST_Within	835
12.17.1	ST_DWithin	836
12.17.1	ST_DFullyWithin	837
12.18	Raster Tips	838
12.18.1	Out-DB Rasters	838
12.18.1.1	Directory containing many files	838
12.18.1.2	Maximum Number of Open Files	838
12.18.1.2.1	Maximum number of open files for the entire system	839
12.18.1.2.2	Maximum number of open files per process	839
13	PostGIS	842
14	PostGIS Extras	849
14.1		849
14.1.1		849
14.1.2		850
14.1.2.1	stdaddr	850
14.1.3		851
14.1.3.1		851
14.1.3.2		856
14.1.3.3		856
14.1.4		857
14.1.4.1	parse_address	857
14.1.4.2	standardize_address	858
14.2	TIGER	860
14.2.1	Drop_Indexes_Generate_Script	860
14.2.2	Drop_Nation_Tables_Generate_Script	862
14.2.3	Drop_State_Tables_Generate_Script	862
14.2.4	Geocode	863
14.2.5	Geocode_Intersection	866
14.2.6	Get_Geocode_Setting	868
14.2.7	Get_Tract	869
14.2.8	Install_Missing_Indexes	870
14.2.9	Loader_Generate_Census_Script	871
14.2.10	Loader_Generate_Script	873
14.2.11	Loader_Generate_Nation_Script	876
14.2.12	Missing_Indexes_Generate_Script	877
14.2.13	Normalize_Address	878

14.2.14 Page_Normalize_Address	880
14.2.15 Pprint_Addy	883
14.2.16 Reverse_Geocode	884
14.2.17 Topology_Load_Tiger	887
14.2.18 Set_Geocode_Setting	889

15 PostGIS Special Functions Index 891

15.1 PostGIS Aggregate Functions	891
15.2 PostGIS Window Functions	892
15.3 PostGIS SQL-MM Compliant Functions	892
15.4 PostGIS Geography Support Functions	898
15.5 PostGIS Raster Support Functions	899
15.6 PostGIS Geometry / Geography / Raster Dump Functions	909
15.7 PostGIS Box Functions	909
15.8 PostGIS Functions that support 3D	911
15.9 PostGIS Curved Geometry Support Functions	917
15.10 PostGIS Polyhedral Surface Support Functions	920
15.11 PostGIS Function Support Matrix	924
15.12 New, Enhanced or changed PostGIS Functions	932
15.12.1 PostGIS Functions new or enhanced in 3.3	932
15.12.2 PostGIS Functions new or enhanced in 3.2	932
15.12.3 PostGIS Functions new or enhanced in 3.1	933
15.12.4 PostGIS Functions new or enhanced in 3.0	935
15.12.5 PostGIS Functions new or enhanced in 2.5	936
15.12.6 PostGIS Functions new or enhanced in 2.4	938
15.12.7 PostGIS Functions new or enhanced in 2.3	939
15.12.8 PostGIS Functions new or enhanced in 2.2	941
15.12.9 PostGIS functions breaking changes in 2.2	941
15.12.10 PostGIS Functions new or enhanced in 2.1	942
15.12.11 PostGIS functions breaking changes in 2.1	942
15.12.12 PostGIS Functions new, behavior changed, or enhanced in 2.0	942
15.12.13 PostGIS Functions changed behavior in 2.0	944
15.12.14 PostGIS Functions new, behavior changed, or enhanced in 1.5	944
15.12.15 PostGIS Functions new, behavior changed, or enhanced in 1.4	945
15.12.16 PostGIS Functions new in 1.3	945

16 Reporting Problems 946

16.1 Reporting Software Bugs	946
16.2 Reporting Documentation Issues	946

A	Appendix	947
A.1	PostGIS 3.3.8	947
A.1.1	Bug Fixes	947
A.2	PostGIS 3.3.7	947
A.2.1	Bug Fixes	947
A.3	PostGIS 3.3.6	948
A.3.1	Bug Fixes	948
A.4	PostGIS 3.3.5	948
A.4.1	Bug Fixes and Enhancements	948
A.5	PostGIS 3.3.4	949
A.5.1	Bug Fixes and Enhancements	949
A.6	PostGIS 3.3.3	949
A.6.1	Bug Fixes and Enhancements	949
A.7	PostGIS 3.3.2	950
A.7.1	Bug and Security Fixes	950
A.8	PostGIS 3.3.1	950
A.8.1	Bug Fixes	950
A.9	PostGIS 3.3.0	951
A.9.1	New features	951
A.9.2	Breaking Changes	951
A.9.3	Enhancements	951
A.9.4	Bug Fixes	952
A.10	PostGIS 3.3.0rc2	952
A.10.1	Bug Fixes	952
A.11	PostGIS 3.3.0rc1	953
A.11.1	Bug Fixes	953
A.12	PostGIS 3.3.0beta2	953
A.12.1	New Features	953
A.12.2	Enhancements	954
A.12.3	Bug Fixes	954
A.13	PostGIS 3.3.0beta1	954
A.13.1	Enhancements	954
A.13.2	New features	954
A.13.3	Bug Fix	954
A.14	PostGIS 3.3.0alpha1	955
A.14.1	Breaking changes	955
A.14.2	Enhancements	955
A.14.3	New features	955
A.14.4	Bug Fix	956

A.15 PostGIS 3.2.0 (Olivier Courtin Edition)	956
A.15.1 Breaking changes	956
A.15.2 Enhancements	956
A.15.3 New features	957
A.16 PostGIS 3.2.0beta3	958
A.16.1 Breaking changes / fixes	958
A.17 Release 3.2.0beta2	958
A.17.1 Breaking changes / fixes	959
A.17.2 Enhancements	959
A.18 Release 3.2.0beta1	959
A.18.1 Bug Fixes and Breaking Changes	959
A.18.2 Enhancements	959
A.19 Release 3.2.0alpha1	959
A.19.1 Breaking changes	959
A.19.2 Enhancements	960
A.19.3 New features	960
A.20 Release 3.1.0beta1	961
A.20.1 Breaking changes	961
A.20.2 Enhancements	961
A.21 Release 3.1.0alpha3	961
A.21.1 Breaking changes	961
A.21.2 New features	962
A.21.3 Enhancements	962
A.21.4 Bug Fixes	962
A.22 Release 3.1.0alpha2	962
A.22.1 New Features	963
A.22.2 Enhancements	963
A.22.3 Bug fixes	963
A.23 Release 3.1.0alpha1	963
A.23.1 Breaking Changes	963
A.23.2 New features	964
A.23.3 Enhancements	964
A.24 Release 3.0.0	964
A.24.1 New Features	964
A.24.2 Breaking Changes	965
A.24.3 Enhancements	965
A.25 Release 3.0.0rc2	966
A.25.1 Major highlights	967
A.26 Release 3.0.0rc1	967

A.26.1 Major highlights	967
A.27 Release 3.0.0beta1	967
A.27.1 Major highlights	967
A.28 Release 3.0.0alpha4	968
A.28.1 Major highlights	968
A.29 Release 3.0.0alpha3	968
A.29.1 Major highlights	968
A.30 Release 3.0.0alpha2	969
A.30.1 Major highlights	969
A.31 Release 3.0.0alpha1	969
A.31.1 New Features	969
A.32 Release 2.5.0	969
A.32.1 New Features	970
A.32.2 Breaking Changes	970
A.33 Release 2.4.5	971
A.33.1 Bug Fixes	971
A.34 Release 2.4.4	972
A.34.1 Bug Fixes	972
A.34.2 Enhancements	972
A.35 Release 2.4.3	972
A.35.1 Bug Fixes and Enhancements	972
A.36 Release 2.4.2	973
A.36.1 Bug Fixes and Enhancements	973
A.37 Release 2.4.1	973
A.37.1 Bug Fixes and Enhancements	973
A.38 Release 2.4.0	973
A.38.1 New Features	974
A.38.2 Enhancements and Fixes	974
A.38.3 Breaking Changes	975
A.39 Release 2.3.3	975
A.39.1 Bug Fixes and Enhancements	975
A.40 Release 2.3.2	975
A.40.1 Bug Fixes and Enhancements	975
A.41 Release 2.3.1	976
A.41.1 Bug Fixes and Enhancements	976
A.42 Release 2.3.0	976
A.42.1 Important / Breaking Changes	976
A.42.2 New Features	976
A.42.3 Bug Fixes	977

A.42.4 Performance Enhancements	977
A.43 Release 2.2.2	977
A.43.1 New Features	978
A.44 Release 2.2.1	978
A.44.1 New Features	978
A.45 Release 2.2.0	979
A.45.1 New Features	979
A.45.2 Enhancements	980
A.46 Release 2.1.8	981
A.46.1 Bug Fixes	981
A.47 Release 2.1.7	981
A.47.1 Bug Fixes	981
A.48 Release 2.1.6	981
A.48.1 Enhancements	981
A.48.2 Bug Fixes	982
A.49 Release 2.1.5	982
A.49.1 Enhancements	982
A.49.2 Bug Fixes	982
A.50 Release 2.1.4	982
A.50.1 Enhancements	982
A.50.2 Bug Fixes	983
A.51 Release 2.1.3	983
A.51.1 Important changes	983
A.51.2 Bug Fixes	984
A.52 Release 2.1.2	984
A.52.1 Bug Fixes	984
A.52.2 Enhancements	984
A.53 Release 2.1.1	985
A.53.1 Important Changes	985
A.53.2 Bug Fixes	985
A.53.3 Enhancements	985
A.54 Release 2.1.0	985
A.54.1 Important / Breaking Changes	985
A.54.2 New Features	986
A.54.3 Enhancements	987
A.54.4 Fixes	989
A.54.5 Known Issues	990
A.55 Release 2.0.5	990
A.55.1 Bug Fixes	990

A.55.2 Important Changes	990
A.56 Release 2.0.4	990
A.56.1 Bug Fixes	990
A.56.2 Enhancements	991
A.56.3 Known Issues	991
A.57 Release 2.0.3	991
A.57.1 Bug Fixes	992
A.57.2 Enhancements	992
A.58 Release 2.0.2	992
A.58.1 Bug Fixes	992
A.58.2 Enhancements	993
A.59 Release 2.0.1	993
A.59.1 Bug Fixes	994
A.59.2 Enhancements	995
A.60 Release 2.0.0	995
A.60.1 Testers - Our unsung heroes	995
A.60.2 Important / Breaking Changes	995
A.60.3 New Features	996
A.60.4 Enhancements	996
A.60.5 Bug Fixes	997
A.60.6 Release specific credits	997
A.61 Release 1.5.4	997
A.61.1 Bug Fixes	997
A.62 Release 1.5.3	998
A.62.1 Bug Fixes	998
A.63 Release 1.5.2	998
A.63.1 Bug Fixes	998
A.64 Release 1.5.1	999
A.64.1 Bug Fixes	999
A.65 Release 1.5.0	999
A.65.1 API Stability	1000
A.65.2 Compatibility	1000
A.65.3 New Features	1000
A.65.4 Enhancements	1001
A.65.5 Bug fixes	1001
A.66 Release 1.4.0	1001
A.66.1 API Stability	1001
A.66.2 Compatibility	1001
A.66.3 New Features	1001

A.66.4 Enhancements	1002
A.66.5 Bug fixes	1002
A.67 Release 1.3.6	1002
A.68 Release 1.3.5	1002
A.69 Release 1.3.4	1003
A.70 Release 1.3.3	1003
A.71 Release 1.3.2	1003
A.72 Release 1.3.1	1003
A.73 Release 1.3.0	1003
A.73.1 Added Functionality	1003
A.73.2 Performance Enhancements	1003
A.73.3 Other Changes	1004
A.74 Release 1.2.1	1004
A.74.1 Changes	1004
A.75 Release 1.2.0	1004
A.75.1 Changes	1004
A.76 Release 1.1.6	1004
A.76.1 업그레이드	1004
A.76.2 Bug fixes	1005
A.76.3 Other changes	1005
A.77 Release 1.1.5	1005
A.77.1 업그레이드	1005
A.77.2 Bug fixes	1005
A.77.3 New Features	1005
A.78 Release 1.1.4	1005
A.78.1 업그레이드	1006
A.78.2 Bug fixes	1006
A.78.3 Java changes	1006
A.79 Release 1.1.3	1006
A.79.1 업그레이드	1006
A.79.2 Bug fixes / correctness	1006
A.79.3 New functionalities	1007
A.79.4 JDBC changes	1007
A.79.5 Other changes	1007
A.80 Release 1.1.2	1007
A.80.1 업그레이드	1007
A.80.2 Bug fixes	1007
A.80.3 New functionalities	1008
A.80.4 Other changes	1008

A.81 Release 1.1.1	1008
A.81.1	1008
A.81.2 Bug fixes	1008
A.81.3 New functionalities	1008
A.82 Release 1.1.0	1009
A.82.1 Credits	1009
A.82.2	1009
A.82.3 New functions	1009
A.82.4 Bug fixes	1010
A.82.5 Function semantic changes	1010
A.82.6 Performance improvements	1010
A.82.7 JDBC2 works	1010
A.82.8 Other new things	1010
A.82.9 Other changes	1010
A.83 Release 1.0.6	1011
A.83.1	1011
A.83.2 Bug fixes	1011
A.83.3 Improvements	1011
A.84 Release 1.0.5	1011
A.84.1	1011
A.84.2 Library changes	1012
A.84.3 Loader changes	1012
A.84.4 Other changes	1012
A.85 Release 1.0.4	1012
A.85.1	1012
A.85.2 Bug fixes	1012
A.85.3 Improvements	1013
A.86 Release 1.0.3	1013
A.86.1	1013
A.86.2 Bug fixes	1013
A.86.3 Improvements	1013
A.87 Release 1.0.2	1013
A.87.1	1014
A.87.2 Bug fixes	1014
A.87.3 Improvements	1014
A.88 Release 1.0.1	1014
A.88.1	1014
A.88.2 Library changes	1014
A.88.3 Other changes/additions	1014

A.89 Release 1.0.0	1015
A.89.1	1015
A.89.2 Library changes	1015
A.89.3 Other changes/additions	1015
A.90 Release 1.0.0RC6	1015
A.90.1	1015
A.90.2 Library changes	1015
A.90.3 Scripts changes	1015
A.90.4 Other changes	1016
A.91 Release 1.0.0RC5	1016
A.91.1	1016
A.91.2 Library changes	1016
A.91.3 Other changes	1016
A.92 Release 1.0.0RC4	1016
A.92.1	1016
A.92.2 Library changes	1016
A.92.3 Scripts changes	1017
A.92.4 Other changes	1017
A.93 Release 1.0.0RC3	1017
A.93.1	1017
A.93.2 Library changes	1017
A.93.3 Scripts changes	1017
A.93.4 JDBC changes	1018
A.93.5 Other changes	1018
A.94 Release 1.0.0RC2	1018
A.94.1	1018
A.94.2 Library changes	1018
A.94.3 Scripts changes	1018
A.94.4 Other changes	1019
A.95 Release 1.0.0RC1	1019
A.95.1	1019
A.95.2 Changes	1019

Abstract

PostGIS is an extension to the PostgreSQL database management system that provides spatial data capabilities. It is a free and open-source software project that is part of the PostgreSQL ecosystem. PostGIS is designed to store, retrieve, and manipulate spatial data in a database. It supports a wide range of spatial data types and operations, including point, line, and polygon geometries. PostGIS is used by a variety of applications, including web mapping services, geographic information systems (GIS), and data analysis tools. It is a powerful tool for managing and analyzing spatial data in a database environment.



PostGIS 3.3.9dev is the latest development version of the software. It includes many new features and improvements over the previous versions. The development version is available for download from the PostGIS website.



PostGIS is licensed under the Creative Commons Attribution-ShareAlike (CC BY-SA) license. This means that you are free to use, modify, and distribute the software, as long as you give credit to the original author and share any modifications you make. The license is a type of copyleft license, which means that any derivative work must also be licensed under the same terms. The license is designed to encourage the sharing and reuse of software and data. For more information about the license, please visit the Creative Commons website.

1.2

Nicklas Avén Geometry clustering function additions, other geometry algorithm enhancements, GEOS enhancements and general user support

Loïc Bartoletti SFCGAL enhancements and maintenance and ci support

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

Martin Davis GEOS enhancements and documentation

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

Aliaksandr Kalenik Geometry Processing, PostgreSQL gist, general bug fixing

1.3

Borrie Park Prior PSC Member. Raster development, integration with GDAL, raster loader, user support, general bug fixing, testing on various OS (Slackware, Mac, Windows, and more)

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

Jorge Arévalo GDAL

Olivier Courtin XML(KML, GML)/GeoJSON

Chris Hodgson OSGeo

Mateusz Loskot PostGIS

Kevin Neufeld PostGIS

Dave Blasby PostGIS

Jeff Lounsbury shapefile

Mark Leslie shapefile GUI

** Pierre Racine** Architect of PostGIS raster implementation. Raster overall architecture, prototyping, programming support

** David Zwarg**

1.4

Alex Bodnaru
 Alex Mayrhofer
 Andrea Peri
 Andreas Forø Tollefsen
 Andreas Neumann
 Andrew Gierth
 Anne Ghisla
 Antoine Bajolet
 Arthur Lesuisse
 Artur Zakirov
 Barbara Phillipot
 Ben Jubb
 Bernhard Reiter
 Björn Esser
 Brian Hamlin
 Bruce Rindahl
 Bruno Wolff III
 Bryce L. Nordgren
 Carl Anderson
 Charlie Savage
 Christoph Berg
 Christoph Moench-Tegeder
 Dane Springmeyer
 Dave Fuhry
 (David Zwarg)
 (David Zwarg)
 (David Zwarg)
 Dian M Fay
 Dmitry Vasilyev
 Eduin Carrillo
 Eugene Antimirov
 Even Rouault
 Frank Warmerdam
 George Silva
 Gerald Fenoy

** **

- [Aiven](#)
- [Arrival 3D](#)
- [Associazione Italiana per l'Informazione Geografica Libera \(GFOSS.it\)](#)
- [AusVet](#)
- [Avencia](#)
- [Azavea](#)

- [Boundless](#)
- [Cadcorp](#)
- [Camptocamp](#)
- [Carto](#)
- [Crunchy Data](#)
- [City of Boston \(DND\)](#)
- [City of Helsinki](#)
- [Clever Elephant Solutions](#)
- [Cooperativa Alveo](#)
- [Deimos Space](#)
- [Faunalia](#)
- [Geographic Data BC](#)
- [Hunter Systems Group](#)
- [ISciences, LLC](#)
- [Kontur](#)
- [Lidwala Consulting Engineers](#)
- [LISAssoft](#)
- [Logical Tracking & Tracing International AG](#)
- [Maponics](#)
- [Michigan Tech Research Institute](#)
- [Natural Resources Canada](#)
- [Norwegian Forest and Landscape Institue](#)
- [Norwegian Institute of Bioeconomy Research \(NIBIO\)](#)
- [OSGeo](#)
- [Oslandia](#)
- [Palantir Technologies](#)
- [Paragon Corporation](#)
- [R3 GIS](#)
- [Refractions Research](#)
- [Regione Toscana - SITA](#)
- [Safe Software](#)
- [Sirius Corporation plc](#)
- [Stadt Uster](#)
- [UC Davis Center for Vectorborne Diseases](#)
- [Université Laval](#)
- [U.S. Department of State \(HIU\)](#)
- [Zonar Systems](#)

크라우드 펀딩 캠페인 크라우드 펀딩 캠페인 이란 수많은 사람ୌ서비스할 수 있는., 우리가 간절&#xd7원하는 기능들을 후원받기 위&#xd5진행하는 캠페인입니다.. 각 캠&#xd3특정 기능 또는 일련의 기능에 특화되어 있습니다.. 각 후원자&#xb2필요한 펀딩의 작은 일부분을

PostGIS 2.0.0 and PostGIS 2.1.0. The **PostGIS** project is a community-driven project that aims to provide a robust and reliable spatial database. The project is currently in the process of developing a new version, PostGIS 3.0.0. The new version will include many new features and improvements, including support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation. The new version will also include support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation. The new version will also include support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation.

PostGIS 2.0.0 and PostGIS 2.1.0. The **PostGIS** project is a community-driven project that aims to provide a robust and reliable spatial database. The project is currently in the process of developing a new version, PostGIS 3.0.0. The new version will include many new features and improvements, including support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation. The new version will also include support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation. The new version will also include support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation.

PostGIS 2.0.0 and PostGIS 2.1.0. The **PostGIS** project is a community-driven project that aims to provide a robust and reliable spatial database. The project is currently in the process of developing a new version, PostGIS 3.0.0. The new version will include many new features and improvements, including support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation. The new version will also include support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation. The new version will also include support for the **PostGIS** 3.0.0 schema, the **PostGIS** 3.0.0 API, and the **PostGIS** 3.0.0 documentation.

The **GEOS** geometry operations library

The **GDAL** Geospatial Data Abstraction Library used to power much of the raster functionality introduced in PostGIS 2. In kind, improvements needed in GDAL to support PostGIS are contributed back to the GDAL project.

The **PROJ** cartographic projection library

Last but not least, **PostgreSQL**, the giant that PostGIS stands on. Much of the speed and flexibility of PostGIS would not be possible without the extensibility, great query planner, GIST index, and plethora of SQL features provided by PostgreSQL.

Chapter 2

PostGIS

PostGIS is a spatial database engine that runs on PostgreSQL. It provides a rich set of spatial data types and functions, and is widely used in GIS applications.

2.1 Installing PostGIS

This section describes how to install PostGIS on a Linux system. The instructions are for the development version (3.3.9dev) and may vary slightly for the stable releases.

```
tar xvfz postgis-3.3.9dev.tar.gz
cd postgis-3.3.9dev
./configure
make
make install
```

PostGIS is installed as a PostgreSQL extension. To use it, you need to create a database and install the extension. The following commands show how to do this:

2.2 **소스를 컴파일하고 설상세 설명**

Note

```
&#xd604;&#xc7ac;      &#xb9ce;&#xc740;    OS   &#xc2dc;&#xc2a4;&#xd15c;&#xb4e4;&#xc774;    PostgreSQL-  
PostGIS&#xc6a9;      &#xc0ac;&#xc804;      &#xad6c;&#xcd95;&#xb41c;      &#xd328;&#xd0a4;&#xc9c0;&#xb97c;  
&#xd3ec;&#xd568;&#xd558;&#xace0; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.  &#xb9ce;&#xc740; &#xacbd;&#xc6b0;,  
&#xcd5c;&#xcca8;&#xb2e8;      &#xbc84;&#xc804;&#xc774;      &#xd544;&#xc694;&#xd558;&#xac70;&#xb098;  
&#xd328;&#xd0a4;&#xc9c0;      &#xad00;&#xb9ac;&#xc790;&#xc778;      &#xacbd;&#xc6b0;&#xac00;  
&#xc544;&#xb2c8;&#xb77c;&#xba74;      &#xcef4;&#xd30c;&#xc77c;&#xd560;      &#xd544;&#xc694;&#xac00;  
&#xc5c6;&#xc2b5;&#xb2c8;&#xb2e4;.  &#xc774;      &#xb2e8;&#xc6d0;&#xc5d0;&#xc11c;&#xb294;      &#xc77c;&#xbc18;&#xc801;&#xc778;  
&#xcef4;&#xd30c;&#xc77c;&#xc5d0;      &#xb300;&#xd574;      &#xc124;&#xba85;&#xd558;&#xace0;  
&#xc788;&#xc73c;&#xbbc0;&#xb85c;.  &#xb354;      &#xc0c1;&#xc138;&#xd55c;      &#xc124;&#xba85;&#xc744;  
&#xc6d0;&#xd560; &#xacbd;&#xc6b0; PostGIS User contributed compile guides &#xc640; PostGIS Dev Wiki  
&#xb97c; &#xcc38;&#xc870;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

PostGIS Pre-built Packages 에서 다양한 OS를
지원하는 사전 빌드된
패키지들을 확인할 수
있습니다.
만약 윈도우 사용자라면 Stack-
builder 또는 PostGIS Windows download site 를 통해
안정적인 버전을 받을 수
있습니다., 또 1~2주에 한 번 혹은
뭔가 특이한 일이 일어날 경우
빌드를 하는 very bleeding-edge windows experimental builds
가 있습니다., 사용자는 이를
통해 진행중인 PostGIS 배포본을
실행해본 수 있습니다.

The PostGIS module is an extension to the PostgreSQL backend server. As such, PostGIS 3.3.9dev *requires* full PostgreSQL server headers access in order to compile. It can be built against PostgreSQL versions 11 - 17. Earlier versions of PostgreSQL are *not* supported.

아직 PostgreSQL을 설엨하지 않았다면 PostgreSQL설엨 지౨서를 참조하십시오.
<http://www.postgresql.org> .

Note

GEOS 기능성을 위해 PostgreSQL설치
시 표준 C++ 라이브러리를
명확하게 링크해야 할 수도
있습니다.

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

이ಃ은 구 버전 개발 도구
 이용시 거짓 C++ 예외 상호작용을
 피하기 위한 방법입니다.
 만약 사용자가 이상한
 문제(백엔드가 갑자기
 끊어진다거나 또는 비슷한
 증상)를 ૊는다면 PostgreSQL을 다시
 컴파일해야 할 수도 있습니다.

다음은 PostGIS 소스를 설정하고 췈합한단계를 설명합니다. 리눅스 사용위한 설명이므로 윈도우 또는 맥 사용자에겐 해당되지 않습니다.

2.2.1 $\frac{1}{2} \frac{d}{dt} \left(\frac{1}{2} \frac{d}{dt} \right)$

PostGIS ƌʤՕ축파일을다운로드웹&#xc
http://postgis.net/stuff/postgis-3.3.9dev.tar.gz 에서다운로드하십&#xc2dc

```
wget http://postgis.net/stuff/postgis-3.3.9dev.tar.gz
tar -xvzf postgis-3.3.9dev.tar.gz
cd postgis-3.3.9dev
```

```
&#xc774; &#xba85;&#xb839;&#xc5b4;&#xb97c; &#xc2e4;&#xd589;&#xd558;&#xba74; &#xd604;&#xc7ac; &#xc791;&#xc5c5;
&#xb514;&#xb809;&#xd1a0;&#xb9ac;&#xc5d0; postgres-3.3.9dev (&#xc774;)&#xb77c;&#xb294; &#xba85;&#xce6d;&#xc75
&#xb514;&#xb809;&#xd130;&#xb9ac;&#xac00; &#xc0dd;&#xae38; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
&#xb2e4;&#xb978; &#xbc29;&#xbc95;&#xc73c;&#xb85c;&#xb294;, svn &#xc800;&#xc7a5;&#xc18c; http://svn.osgeo.org/-  
postgis/trunk/ &#xc5d0;&#xc11c; &#xc18c;&#xc2a4;&#xb97c; &#xccb4;&#xd06c;&#xc544;&#xc6c3;(checkout)&#xd560; &#xc218  
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

```
&#xc124;&#xce58;&#xb97c;&#xacc4;&#xc18d;&#xd558;&#xb824;&#xba74;&#xc0c8;&#xb85c;&#xb9cc;&#xb4e0; postgres-3
&#xb514;&#xb809;&#xd130;&#xb9ac;&#xb85c;&#xc774;&#xb3d9;&#xd569;&#xb2c8;&#xb2e4;.
```

```
./configure
```

2.2.2 $\frac{1}{2} \frac{d}{dt} \left(\frac{1}{2} \frac{d}{dt} \right)$

PostGIS를 빌드하고 사용하기 위해서&#xb
다음과 같은 요구사항들을 만족&#xc
합니다.

 $\&\#xd544;\&\#xc218;\&\#xc0ac;\&\#xd56d;$

- PostgreSQL 11 - 17. A complete installation of PostgreSQL (including server headers) is required. PostgreSQL is available from <http://www.postgresql.org>.
- GNU C ఄ체 PostgreSQL/PostGIS 지원 매트릭스 및 PostGIS/-GEOS 지원 매트릭스는 <http://trac.osgeo.org/postgis/wiki/UsersWikiPostgreSQL> 를 참조하십시오.
- GNU C ఄ파일러(gcc). PostGIS를 ఄ파일하기위해 그 밖에 다른 ANSI C ఄ파일러들사용할 수 있으나 gcc 로 ఄ파일했&#xc7경우 오류가 훨씬 적게 발생합니&
- GNU Make(gmake또는 make). 많은 시스템들에서 GNU make 는 make의 기본 버전입니다. make -v를 통해 버전을 확인하십시오. 다른 버전의 make는 PostGIS Makefile을 제대로 처리&#xc못할 수도 있습니다.
- Proj reprojection library. Proj 4.9 or above is required. The Proj library is used to provide coordinate reprojection support within PostGIS. Proj is available for download from <https://proj.org>.

- GEOS geometry library, version 3.6 or greater, but GEOS 3.11+ is required to take full advantage of all the new functions and features. GEOS is available for download from <https://libgeos.org/>.
- LibXML2, version 2.5.x or higher. LibXML2 is currently used in some imports functions (ST_GeomFromGML and ST_GeomFromKML). LibXML2 is available for download from <https://gitlab.gnome.org/GNOME/libxml2/-/releases>.
- JSON-C 0.9. JSON-C and ST_GeomFromGeoJSON. JSON-C is available for download from <https://github.com/json-c/json-c/releases/>.
- GDAL, version 2+ is required 3+ is preferred. This is required for raster support. <https://gdal.org/download.html>.
- PostgreSQL. PostgreSQL is required for the database. PostgreSQL 10.4 or higher is recommended. PostgreSQL is available for download from <http://trac.osgeo.org/postgis/ticket/635>.

Dependencies

- Section 2.1. PostgreSQL. PostgreSQL is required for the database. PostgreSQL 10.4 or higher is recommended. PostgreSQL is available for download from <http://trac.osgeo.org/postgis/ticket/635>.
- shapefile. shp2pgsql-gui. shp2pgsql-gui is a GUI for shp2pgsql. It is available for download from <http://www.gtk.org/>.
- SFCGAL, version 1.3.1 (or higher), 1.4.1 or higher is recommended. SFCGAL can be used to provide additional 2D and 3D advanced analysis functions to PostGIS cf Section 8.20. And also allow to use SFCGAL rather than GEOS for some 2D functions provided by both backends (like ST_Intersection or ST_Area, for instance). A PostgreSQL configuration variable `postgis.backend` allow end user to control which backend he want to use if SFCGAL is installed (GEOS by default). Nota: SFCGAL 1.2 require at least CGAL 4.3 and Boost 1.54 (cf: <https://sfcgal.org> <https://gitlab.com/sfcgal/SFCGAL/>).
- Section 14.1. PCRE. PCRE is a library for Perl Compatible Regular Expressions. It is available for download from <http://www.pcre.org/>.
- To enable ST_AsMVT protobuf-c library 1.1.0 or higher (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](https://github.com/dimand/protobuf-c). By default, Postgis will use Wagyu to validate MVT polygons faster which requires a c++11 compiler. It will use CXXFLAGS and the same compiler as the PostgreSQL installation. To disable this and use GEOS instead use the `--without-wagyu` during the configure step.
- CUnit(CUnit). CUnit is a unit testing framework for C. It is available for download from <http://cunit.sourceforge.net/>.
- DocBook(xsltproc). DocBook is a standard for technical documentation. It is available for download from <http://www.docbook.org/>.
- DBLatex(dblatex). DBLatex is a LaTeX package for database diagrams. It is available for download from <http://dblatex.sourceforge.net/>.
- ImageMagick(convert). ImageMagick is a software library to process images. It is available for download from <http://www.imagemagick.org/>.

2.2.3

PostGIS 3.3.9dev

./configure

PostGIS 3.3.9dev

PostGIS 3.3.9dev

--with-library-minor-version Starting with PostGIS 3.0, the library files generated by default will no longer have the minor version as part of the file name. This means all PostGIS 3 libs will end in `postgis-3`. This was done to make `pg_upgrade` easier, with downside that you can only install one version PostGIS 3 series in your server. To get the old behavior of file including the minor version: e.g. `postgis-3.0` add this switch to your configure statement.

--prefix=PREFIX PostGIS

Caution



PostGIS

--with-pgconfig=FILE PostgreSQL

--with-gdalconfig=FILE GDAL

--with-geosconfig=FILE GEOS는 필수 도형 라이브러리 GEOS 설치 디렉터리의 위치를 확인 소프트웨어 설치를 활성화하&#xae3 위한 **geos-config**라는 유틸리티를 제공&#xd5 PostGIS 빌드 시 대상이 될 특정 GEOS 설치 디렉터리를 사용자가 직접 설&#xc81 이 파라미터(**--with-geosconfig=/path/to/geos-config**)를 사용&#xd5

--with-xml2config=FILE LibXML은 GeomFromKML/GML 프로세스를 진&#xc 위해 필요한 라이브러리입니&#xb2e 일򸰘적으로 libxml을 설치하면 찾을 수 있지만 설치하지 않았거나 특정 ஄전을 사용하기 ଔ랄 경&#xc6 LibXML 설치 디렉터리의 위치를 확&#xc77 PostGIS 소프트웨어 설치를 활성화하&#xae3 위해 **xml2-config**라는 설정 파일에 PostGIS의 위치를 지정해야 합니다 PostGIS 빌&#xb4dc 시 대상이 될 특정 LibXML 설치 디렉&#xd1 사용자가 직접 설정하려면 이 파라미터(**--with-xml2config=/path/to/xml2-config**)를 사용하&#xc

--with-projdir=DIR Proj4는 PostGIS 필수 재투영 라이브러 PostGIS 빌드 시 대상이 될 특정 Proj4 설치 디렉터리를 사용자가 직접 설&#xc81 이 파라미터(**--with-projdir=/path/to/projdir**)를 사용하&#xc

--with-libiconv=DIR iconv 설치 경로

--with-jsondir=DIR **JSON-C** 는 MIT-라이선스의 JSON 라이브&#xc PostGIS의 ST_GeomFromJSON 지원에 필요합니다 PostGIS 빌드 시 대상이 될 특정 JSON-C 설&#xce58 디렉터리를 사용자가 직접 설&#xc81 이 파라미터(**--with-jsondir=/path/to/jsondir**)를 사용하&#xc

--with-pcredir=DIR **PCRE** 는 BSD-라이선스의 펄 호환 가능 정귞 표현식 라이브러리&#xb85 address_standardizer 확장 프로그램이 필요합&#xb2 PostGIS 빌드 시 대상이 될 특정 PCRE 설치 디렉터리를 사용자가 직접 설&#xc81 이 파라미터(**--with-pcredir=/path/to/pcredir**)를 사용하&#xc

--with-gui 데이터 임포트 GUI 󌻴파일(GTK+2.0 필&#xc6 shp2pgsql-gui의 shp2pgsql에 대한 그래픽 인테페&#xc 생성합니다.

--without-raster 래스터 지원 설치

--without-topology Disable topology support. There is no corresponding library as all logic needed for topology is in postgis-3.3.9dev library.

--with-gettext=no 기본적으로 PostGIS는 gettext 지원을 &#xac10 함૨ 컴파일하지만 로더의 파&#xc19 야기하는 비호환성 문제가 ଜ&#xc0c 경우 이 명령어로 gettext 지원을 완전 비활성화시킬 수 있습니다 이&#xb7 ଩க으로 설정을 변경해서 문&#xc81 해결하는 예는 ஄그 티ఓ <http://trac.osgeo.org/postgis/ticket/748> 을 참조하십시오 주의 gettext 지끈다고 해서 별다른 문제는 없&#xc2 gettext 지원은 아직 문서화되지도 않

검증중에 있는 GUI 로더 용 국제 도지원에 사용됩니다.

--with-sfcgal=PATH 기본적으로 PostGIS는 이 스위치 없이는 sfcgal 지원과 함께 설치되지 않습니다. PATH 는 sfcgal-config를 가리키는 대체 경로를 지정하도록 해주는 선택적인 인자입니다.

--without-phony-revision Disable updating postgis_revision.h to match current HEAD of the git repository.

Note

PostGIS를 SVN 저장소 에서 얻었다면,먼저 다음 스크립트를 실행하십시오.

./autogen.sh

이 스크립트는 **configure** 스크립트를 생성하는데,이 스크립트는 PostGIS의 사용자 지정 설치를 위해 이용됩니다.만약 tar 파일 형태로 PostGIS를 얻었다면 이미 **configure** 가 생성되었기 때문에 ./autogen.sh 를 실행할 필요는 없습니다.

2.2.4 빌드

일단 Makefile이 생성되면 PostGIS 빌드 작업지실행만큼이나 쉽습니다.

make

산출물의 마지막 줄에 "PostGIS was built successfully. Ready to install."이란 문장이 보여야 합니다.

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.

make comments

PostGIS 2.0 ஄전부터 소개되었습니다. 빠참조 또는 학습용 유인물에 적합지 참조 자료(cheat sheet) html 파일을 생성합니다 파일 생성에 xsltproc가 필요하며,폴더 안에 다음 topology_cheatsheet.html, tiger_geocoder_cheatsheet.html, raster_cheatsheet.html, postgis_cheatsheet.html 4개의 파일을 생성지 것입니다.

html 렏 pdf 형식으로 &#bbf8;리 만들어진 파지 **PostGIS / PostgreSQL Study Guides** 에서 다운로드렛을 수 있습니다.

make cheatsheets


```
share/extension &#xd3f4;&#xb354;&#xc758;&#xb8e8;&#xd2b8;&#xc5d0;&#xbcf5;&#xc0ac;&#xb418;&#xc5b4;&#xc57c;
&#xd568;&#xc5d0;&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;,. extensions/postgis/sql/*.sql,
extensions/postgis_topology/sql/*.sql
```

```
&#xc774;&#xb807;&#xac8c;&#xd55c;&#xb2e4;&#xc74c;,&#xc0ac;&#xc6a9;&#xc790;&#xb294;&#xc0ac;&#xc6a9;&#xac00;&#xc774;
extensions&#xc73c;&#xb85c;postgis,postgis_topology&#xc744;PgAdmin->extensions&#xc5d0;&#xc11c;&#xbcf5;
&#xc218;&#xc788;&#xc744;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
&#xb9cc;&#xc57d;psql&#xc744;&#xc774;&#xc6a9;&#xc911;&#xc774;&#xb77c;&#xba74;&#xb2e4;&#xc74c;&#xc758;
&#xcffc;&#xb9ac;&#xb97c;&#xc2e4;&#xd589;&#xd568;&#xc73c;&#xb85c;&#xc368;&#xd655;&#xc778;&#xd560;&#xc218;
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

name	default_version	installed_version
address_standardizer	3.3.9dev	3.3.9dev
address_standardizer_data_us	3.3.9dev	3.3.9dev
postgis	3.3.9dev	3.3.9dev
postgis_sfcgal	3.3.9dev	
postgis_tiger_geocoder	3.3.9dev	3.3.9dev
postgis_topology	3.3.9dev	

(6 rows)

```
&#xb9cc;&#xc57d;&#xc0ac;&#xc6a9;&#xc790;&#xac00;&#xcffc;&#xb9ac;&#xd558;&#xb294;&#xb370;&#xc774;&#xd130;&#xc774;
extension&#xc774;&#xc124;&#xc58;&#xb418;&#xc5b4;&#xc788;&#xb2e4;&#xba74;,&#xc0ac;&#xc6a9;&#xc790;&#xb294;
installed_version&#xc5cc;&#xb7fc;&#xc5d0;&#xc11c;&#xc774;&#xb984;&#xc744;&#xbcf5;&#xc218;&#xc788;&#xc2b5;
&#xb9cc;&#xc57d;&#xc544;&#xbb34;&#xb808;&#xc5f4;&#xb4dc;&#xb3c4;&#xc5c6;&#xb2e4;&#xba74;&#xc11c;&#xbcf5;&#xc774;
postgis extension&#xc774;&#xc804;&#xd600;&#xc124;&#xc58;&#xb418;&#xc5b4;&#xc788;&#xc9c0;&#xc54a;&#xc74c;&#xc774;
&#xb73b;&#xd569;&#xb2c8;&#xb2e4;PgAdmin III 1.14&#xc774;&#xc0c1;&#xbcf5;&#xc804;&#xc5d0;&#xc11c;&#xb294;
&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xd0d0;&#xc0c9;&#xd2b8;&#xb9ac;&#xc758;extensions&#xc5d0;
&#xb9c8;&#xc6b0;&#xc2a4;&#xc624;&#xb978;&#xcabd;&#xbcf5;&#xd2bc;&#xd074;&#xb9ad;&#xc744;&#xd1b5;&#xd574;
&#xc5c5;&#xadf8;&#xb808;&#xc774;&#xb4dc;&#xb610;&#xb294;&#xc0ad;&#xc81c;&#xb97c;&#xd5c8;&#xc6a9;&#xd569;&#xc774;
extension&#xc774;&#xc774;&#xc6a9;&#xac00;&#xb2a5;&#xd55c;&#xc0c1;&#xd0dc;&#xb77c;&#xba74;pgAdmin exten-
sion&#xc778;&#xd130;&#xd398;&#xc774;&#xc2a4;&#xb610;&#xb294;&#xb2e4;&#xc74c;&#xc758;sql&#xb85;&#xb839;&#xc774;
&#xc2e4;&#xd589;&#xd568;&#xc73c;&#xb85c;&#xc368;&#xc120;&#xd0dd;&#xd55c;&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;
&#xc548;&#xc5d0;postgis extension&#xc744;&#xc124;&#xc58;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
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;
```

```
PSQL&#xc5d0;&#xc11c;&#xb2e4;&#xc74c;&#xba85;&#xb839;&#xc5b4;&#xb97c;&#xc0ac;&#xc6a9;&#xd558;&#xba74;
&#xc5b4;&#xb5a4;&#xbcf5;&#xc804;&#xc744;&#xc124;&#xc58;&#xd588;&#xb294;&#xc9c0;,&#xc5b4;&#xb5a4;&#xc2a4;&#xc774;
&#xc124;&#xc58;&#xd588;&#xb294;&#xc9c0;&#xc54c;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

```
List of installed extensions
```

```
-[ RECORD 1 ]-----
Name          | postgis
```


**Note**

PostgreSQL, GEOS, Proj4, LD_LIBRARY_PATH
 –with-pgconfig, PATH, PGPOR

**Caution**

–with-pgconfig, PATH, PGPOR
 –with-pgconfig, PATH, PGPOR

If successful, make check will produce the output of almost 500 tests. The results will look similar to the following (numerous lines omitted below):

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

.
.
.

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

.
.
.

Running tests

.
.
.

Run tests: 134
Failed: 0

-- if you build with SFCGAL

.
.
.

Running tests

.
.
.
```

```

Run tests: 13
Failed: 0

-- if you built with raster support

.
.
.

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

.
.
.

Running tests

.
.
.

Run tests: 101
Failed: 0

-- topology regress

.
.
.

Running tests

.
.
.

Run tests: 51
Failed: 0

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

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

.
.
.

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

```

```
postgis_tiger_geocoder &#xc640; address_standardizer &#xd655;&#xc7a5; &#xd504;&#xb85c;&#xadf8;&#xb7a8;
&#xd604;&#xc7ac; &#xd45c;&#xc900; PostgreSQL &#xc124;&#xce58;&#xac80;&#xc0ac;(installcheck)&#xb9cc;&#xc744;
&#xc9c0;&#xc6d0;&#xd569;&#xb2c8;&#xb2e4;. &#xc774; &#xd655;&#xc7a5; &#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xb4e4;&#
&#xd14c;&#xc2a4;&#xd2b8;&#xd558;&#xb824;&#xba74; &#xb2e4;&#xc74c;&#xc744; &#xc2e4;&#xd589;&#xd558;&#xc2ed;&#x
```

주의; 이미 PostGIS 코드 폴더의 루트에&#x
make install을 실행했다면 다시 실행할 필&
없습니다.

```
address_standardizer&#xc758; &#xacbd;&#xc6b0;:
```

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

다음과 같은 결과가 나와야 합니&#xl

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

TIGER 지오코딩 도구의 경우, 사용자&#xc7
PostgreSQL 인스턴스 안에서 PostGIS 및 fuzzystmatch 확&#xc7a
프로그램을 이용할 수 있는지 확&#
address_standardizer 지원이 되도록 PostGIS를 빌드했&#
address_standardizer 테스트도 함께 실행될 것입&#

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

다음과 같은 결과가 나와야 합니&#xl

```
===== dropping database "contrib_regression" =====
DROP DATABASE
===== creating database "contrib_regression" =====
CREATE DATABASE
ALTER DATABASE
===== installing fuzzystmatch =====
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-page_normalize_address ... ok

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

2.2.7 $\&\#xc124;\&\#xce58;$

make install

이것은 --**prefix** 설정 파라미터에 정의&#xb
하위 경로에 PostGIS 설치 파일을 복사&#xc
것입니다.

- `로더(loader)와덤퍼ଔ이너리들은[prefix]/bin/에설치됩니다.`
- `postgis.sql와같은SQL파일들은[prefix]/share/contrib/에설치됩니다.`
- `PostGIS라이브러리들은[prefix]/lib/에설치됩니다.`

```
&#xb9cc;&#xc57d;&#xae30;&#xc874;&#xc5d0;postgis_comments.sql,raster_comments.sql&#xd30c;&#xc77c;&#x
&#xc0dd;&#xc131;&#xd558;&#xae30;&#xc704;&#xd574;make comments&#xba85;&#xb839;&#xc5b4;&#xb97c;&#xc2e4;&#xd
&#xc801;&#xc774;&#xc788;&#xc73c;&#xc2dc;&#xb2e4;&#xba74;,&#xb2e4;&#xc74c;&#xc744;&#xc2e4;&#xd589;&#xd574;
sql&#xd30c;&#xc77c;&#xc744;&#xc124;&#xce58;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

make comments-install



Note

```
xsltproc&#xc758;      &#xc801;&#xc6a9;      &#xc774;&#xd6c4;      &#xc77c;&#xc18;&#xc801;&#xc778;
&#xc124;&#xc58;&#xb85c;&#xbd80;&#xd130;      postgis_comments.sql,      raster_comments.sql,
topology_comments.sql&#xb294;&#xbd84;&#xb9ac;&#xb418;&#xc5c8;&#xc2b5;&#xb2c8;&#xb2e4;
```

2.3 **주소 표준화 도구 설&#xce5
및 활용**

address_standardizer 확장프로그램은별도로다운로드해야하는별도의패키이확장프로그램이଴슨일을하사용자의필요에따라어떻게설대한자세한정보는 Section 14.1 를참조하이표준화도구는 Normalize_Address 대신 PostGIS 용패키징된 TIGER 지오코딩도구(geocoder)와함께쓰일수있습니다.이렇게대사용하는଩க은 Section 2.4.3 을참조하십주소표준화도구를사용자의다지오코딩도구를위한구성요소(building block)로사용하거나.주소를더쉽게비교하기위해사용자주소를표데사용할수도있습니다.

주소 표준화 도구는 PCRE에 의존성&#xc74
갖습니다. PCRE는 많은 유닉스 파생
시스템에 일반적으로 이미 설치&#xb
있지만 <http://www.pcre.org> 에서 최신 버전을 다&#
수 있습니다. Section **2.2.3** 과정에서 PCRE를 찾&#xb
주소 표준화 도구 확장 프로그램&#xb

```
&#xc790;&#xb3d9;&#xc801;&#xc73c;&#xb85c;&#xbe4c;&#xb4dc;&#xd560;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.&#xc0ac;&#xc0ac;&#xc6a9;&#xd558;&#xace0;&#xc790;&#xd558;&#xb294; PCRE&#xb97c;&#xb530;&#xb85c;&#xc124;&#xce58;&#xd558;&#xacbd;&#xc6b0;.&#xc124;&#xc815;&#xd30c;&#xb77c;&#xbbf8;&#xd130;--with-pcredir=/path/to/pcre&#xc758;/path/to/pcre&#xbd80;&#xbd84;&#xc5d0;&#xc0ac;&#xc6a9;&#xc790;&#xc758; PCRE include&#xbc0f; lib&#xb514;&#xb8e8;&#xd2b8;&#xd3f4;&#xb354;&#xb97c;&#xc785;&#xb825;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

```
&#xc708;&#xb3c4;&#xc6b0;&#xc0ac;&#xc6a9;&#xc790;&#xc758;&#xacbd;&#xc6b0; PostGIS 2.1&#xc774;&#xc0c1;&#xbc84;&#xbc88;&#xb4e4;&#xc740;&#xc774;&#xbbf8;address_standardizer&#xc640;&#xd568;&#xae8;&#xd328;&#xd0a4;&#xc9d5;&#xc788;&#xc73c;&#xbbc0;&#xb85c;&#xcef4;&#xd30c;&#xc77c;&#xd560;&#xd544;&#xc694;&#xc5c6;&#xc774;&#xbc14;&#xc774;CREATE EXTENSION&#xb2e8;&#xacc4;&#xb85c;&#xac74;&#xb108;&#xb6f8;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;&#xc124;&#xce58;&#xb97c;&#xc644;&#xb8cc;&#xd588;&#xb2e4;&#xba74;&#xc0ac;&#xc6a9;&#xc790;&#xb370;&#xc774;&#xc5f0;&#xacb0;&#xd574;&#xc11c; SQL&#xc744;&#xc2e4;&#xd589;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;
```

```
CREATE EXTENSION address_standardizer;
```

```
&#xb2e4;&#xc74c;&#xd14c;&#xc2a4;&#xd2b8;&#xc5d0;&#xb294;&#xc5b4;&#xb5a4; rules, gaz, &#xb610;&#xb294; lex&#xd14c;&#xc774;&#xbe14;&#xb3c4;&#xd544;&#xc694;&#xc5c6;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

```
&#xb2e4;&#xc74c;&#xacfc;&#xac19;&#xc740;&#xacb0;&#xacfc;&#xac00;&#xb098;&#xc640;&#xc57c;&#xd569;&#xb2c8;&#xb2e4;
```

num	street	city	state	zip
1	Devonshire Place PH301	Boston	MA	02109

2.3.1 Regexp::Assemble

```
address_standardizer&#xd655;&#xc7a5;&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc744;&#xcef4;&#xd30c;&#xc77c;&#xd558;&#xb370;&#xd384; Regexp:Assemble&#xc740;&#xb354;&#xc774;&#xc0c1;&#xd544;&#xc694;&#xc5c6;&#xc2b5;&#xb2c8;&#xb2e4;&#xd384; Regexp:Assemble&#xc774;&#xc0dd;&#xc131;&#xd558;&#xb294;&#xd30c;&#xc77c;&#xb4e4;&#xc774;&#xc18c;&#xc2b8;&#xb9ac;&#xc758;&#xc77c;&#xbd80;&#xb85c;&#xd1b5;&#xd569;&#xb418;&#xc5c8;&#xae30;&#xb54c;&#xbb38;&#xd558;&#xc9c0;&#xb9cc; usps-st-city-orig.txt&#xb610;&#xb294; usps-st-city-orig.txt usps-st-city&#xd30c;&#xc77c;&#xc744;&#xd3b8;&#xc9d1;&#xd574;&#xc57c;&#xd560;&#xacbd;&#xc6b0; parseaddress-stcity&#xb97c;&#xb2e4;&#xc2dc;&#xbe4c;&#xb4dc;&#xd574;&#xc57c;&#xd558;&#xb294;&#xb370;&#xc774;&#xb54c; Regexp:Assemble&#xc774;&#xd544;&#xc694;&#xd569;&#xb2c8;&#xb2e4;.
```

```
cpan Regexp::Assemble
```

```
&#xb610;&#xb294;&#xc6b0;&#xbd84;&#xd22c;/&#xb370;&#be44;&#xc548;&#xc2dc;&#xc2a4;&#xd15c;&#xc758;&#xacbd;&#xb2e4;&#xc74c;&#xc791;&#xc5c5;&#xc744;&#xd574;&#xc57c;&#xd560;&#xc218;&#xb3c4;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;
```

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

2.4 Tiger Geocoder

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.

2.4.1 TIGER 지오코딩 도구로 사용&#xc79
PostGIS 데이터베이스 활성화::
확장 프로그램 이용

PostgreSQL 9.1 이상 ಄전쫼 PostGIS 2.1 이상 ಄전을
사용중이라면, TIGER 지오󏱔딩 도구&#xb9
설치하는 데 새로운 확장 프로그&#
모델을 활용할 수 있습니다. 그 ఩&
다음쫼 ఙ습니다.

1. 㨼저 PostGIS 2.1 이상 버전의 바이너리다운로드하거나 컴파일해서 일방법으로 설치하십시오. TIGER 지오도구는 򻬼론 필수 확장 프로그&#xb7함께 설치될 것입니다.
2. PSQL, pgAdmin 또는 다른 도구를 통해 사&#xc6a9데이터베이스에 연결해서 다&#xc74명령어를 실행하십시오. 이򻯸 PostGIS가 설치된 데이터베이스에 &#xc124경우,, 첫 번째 단계를 수행할 필&#xc6없다는 사실을 주의하십시오. 이򻯸 fuzzystmatch 확장 프로그램이 설&#xce5있다면 이 두 번째 단계도 수행&#xd5필요가 없습니다.

```
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;
```

```
&#xc774;&#xbfb8; postgis_tiger_geocoder &#xd655;&#xc7a5; &#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc744; &#xc124;&#x
&#xcd5c;&#xc2e0; &#xbc84;&#xc804;&#xc73c;&#xb85c; &#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xb9cc; &#xd558;&#xace
&#xd560; &#xacbd;&#xc6b0;:
```

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

```
tiger.loader_platform&#x0ac;tiger.loader_variables &#x5d0; &#x0ac;&#xc6a9;&#xc790; &#xc9c0;&#  
&#xd56d;&#xbaa9;&#xc744; &#xb9cc;&#xb4e4;&#xc5c8;&#xac70;&#xb098; &#xbcc0;&#xacbd;&#xd55c; &#xacbd;&#xc6b  
&#xc774; &#xd14c;&#xc774;&#xbe14;&#xb4e4;&#xb3c4; &#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xd574;&#xc57c;  
&#xd560; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

3. 제대로설치되었는지확인하렭사용자데이터베이스에다음을실행하십시&#xc624::

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

다음&#xacfc: 같&#xc740: 결&#xacfc:&#xac00: 나와&#xc57c: 합&#xb2c8

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

4. `tiger.loader_platform` table, `os`, `declare_sect`, `pgbin`, `wget`, `unzip_command`, `psql`, `path_sep`, `loader`, `environ_set_command`, `county_process_command`

```
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';
```

```
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';
```

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.

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 = 'zcta520';
```

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. `tiger.loader_variables` table, `os`, `declare_sect`, `pgbin`, `wget`, `unzip_command`, `psql`, `path_sep`, `loader`, `environ_set_command`, `county_process_command`
7. `gisdata` table, `os`, `declare_sect`, `pgbin`, `wget`, `unzip_command`, `psql`, `path_sep`, `loader`, `environ_set_command`, `county_process_command`

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.

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

15.

```
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;
```

2.4.1.1 TIGER

확장 프로그램 모델을 사용하지 않고 TIGER 지오코딩 도구를 설치했다음과 같이 확장 프로그램 모델변환시킬 수 있습니다:

1. 확장 프로그램 제외(non-extension) 모델을 업그레이드하려면 Section 2.4.5 의 지침따라해보십시오.
2. PSQL 또는 pgAdmin을 통해 사용자 데이터 연결하고 다음 명령어를 실행하

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

2.4.2 TIGER

우선 앞의 설명에 따라 PostGIS를 설치다운로드하십시오.

tar xvfz postgis-3.3.9dev.tar.gz

cd postgis-3.3.9dev/extras/tiger_geocoder

tiger_loader_2015.sql (또는 다른 년도를 로드하 할 경우 사용자가 찾을 수 있는 최로더)를 사용자의 실행 가능한 서등의 경로로 편집하거나 그 대신 설치 완료 후 loader_platform 테이블을 업데 수도 있습니다 이 파일 또는 loader_platform 테이블을 편집지 않는다면 각 항흔히 쓰이는(common case) 위치만 담게 되어 Loader_Generate_Nation_Script 볰 Loader_Generate_Script SQL 함수를 실행해 생성된 스크립트를 직접 편집해 할 것입니다.

TIGER 지오코딩 도구를 처음 설치하다경우 윈도우 시스템이라면 create_geocode.bat 스크립트를 Linux/Unix/Mac OSX 시스템이라면 create_geocode.sh 를 사용자의 PostgreSQL에 특화된 설정으로 편집한 다음 명령 프롬 각각 상응하는 스크립트를 실행다.

tiger 스키마가 데이터베이스에 있다지와 사용제 데이터베이스 search_path에 해당되는 지를 확인하십시오 안있다면 아래 명령을 함께 추가하다.

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

```
&#xd45c;&#xc900;&#xd654;&#xc8fc;&#xc18c;&#xae30;&#xb2a5;&#xc740;&#xae4c;&#xb2e4;&#xb85c;&#xc6b4;&#xc8fc;&#xc81c;&#xc678;&#xd558;&#xace0;&#xb294;&#xb3d9;&#xc791;&#xd569;&#xb2c8;&#xb2e4;.&#xc544;&#xb798;&#xc640;&#xbe44;&#xc2b7;&#xd558;&#xac8c;&#xb098;&#xc624;&#xb294;&#xc9c0;&#xd14c;&#xc2a4;&#xd2b8;&#xd574;&#xbcf4;&#xc
```

```
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
```

2.4.3 주소표준화도구를 TIGER 지오도구와함께사용

```
&#xc0ac;&#xc6a9;&#xc790;&#xb4e4;&#xc758;&#xb9ce;&#xc740;&#xbd88;&#xd3c9;&#xac00;&#xc6b4;&#xb370;&#xd558;&#xc8fc;&#xc18c;&#xc815;&#xad6c;&#xd654;&#xb3c4;&#xad6c; Normalize_Address &#xd568;&#xc218;&#xac00;&#xc9c0;&#xc791;&#xc5c5;&#xc804;&#xc900;&#xbe44;&#xacfc;&#xc815;&#xc5d0;&#xc11c;&#xc8fc;&#xc18c;&#xb97c;&#xc815;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.&#xc815;&#xad6c;&#xd654;&#xb3c4;&#xad6c;&#xb294;&#xc644;&#xbcb4;&#xd558;&#xc54a;&#xc544;&#xadf8;&#xbd88;&#xc644;&#xc804;&#xd568;&#xc744;&#xc218;&#xc815;&#xd558;&#xb824;&#xba74;&#xb9c9;&#xb300;&#xd55c;&#xb178;&#xb825;&#xc774;&#xd544;&#xc694;&#xd569;&#xb2c8;&#xb2e4;.&#xadf8;&#xb798;&#xc6b0;&#xb9ac;&#xb294;&#xd6e8;&#xc52c;&#xb098;&#xc740;&#xc8fc;&#xc18c;&#xd45c;&#xc900;&#xd654;&#xb3c4;&#xc5d4;&#xc9c4;&#xc744;&#xac00;&#xc9c4;&#xb610;&#xb2e4;&#xb978;&#xd504;&#xb85c;&#xc81d;&#xd2b8;&#xc640;&#xd1b5;&#xd569;&#xc2dc;&#xcf30;&#xc2b5;&#xb2c8;&#xb2e4;.&#xc774;&#xc0c8;&#xb85c;&#xc6b4; address_standardizer&#xc774;&#xc6a9;&#xd558;&#xb824;&#xba74;., Section 2.3 &#xc5d0;&#xc124;&#xba85;&#xb41c;&#xb300;&#xb85c;&#xd655;&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc744;&#xcef4;&#xd30c;&#xc77c;&#xd574;&#xc11c;&#xc0ac;&#xc6a9;&#xc790;&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xc5d0;&#xd655;&#xc7a5;&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc124;&#xce58;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

```
postgis_tiger_geocoder &#xb97c;&#xc124;&#xce58;&#xd588;&#xb358;&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc774;&#xd655;&#xc7a5;&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc744;&#xc124;&#xce58;&#xd588;&#xb2e4;&#xba74;., Normalize_Address &#xb300;&#xc2e0; Page_Normalize_Address &#xb97c;&#xc774;&#xc6a9;&#xd560;&#xc218;&#xc788;&#xc774;&#xd655;&#xc7a5;&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc740; TIGER &#xc720;&#xb34;&#xc640;&#xc0c1;&#xa&#xb3d9;&#xc791;&#xd558;&#xbbc0;&#xb85c;., &#xad6d;&#xc81c;&#xc8fc;&#xc18c;&#xc640;&#xac19;&#xc740;&#xb2e4;&#xb370;&#xc774;&#xd130;&#xc18c;&#xc2a4;&#xc640;&#xd568;&#xaed8;&#xc0ac;&#xc6a9;&#xd560;&#xc218;&#xc788;&#xc2e4;&#xc81c;&#xb85c; TIGER &#xc9c0;&#xc624;&#xcf54;&#xb529;&#xb3c4;&#xad6c;&#xd655;&#xc7a5;&#xd504;&#xb&#xad6c;&#xce59;&#xd14c;&#xc774;&#xbe14; (tiger.pagc_rules), &#xc9c0;&#xba85;&#xc0c9;&#xc778;&#xd14c;&#xc (tiger.pagc_gaz), &#xadf8;&#xb9ac;&#xace0; &#xc5b4;&#xd718;&#xbaa9;&#xb85d;&#xd14c;&#xc774;&#xbe14; (tiger.&#xc790;&#xc6b4;&#xc218;&#xc815;&#xb84;&#xc804;&#xacfc;&#xd568;&#xaed8;&#xd328;&#xd0a4;&#xc9d5;&#xb418;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.&#xc0ac;&#xc6a9;&#xc790;&#xc790;&#xc2e0;&#xc758;&#xd544;&#xc694;&#xc5d0;&#xb530;&#xb77c;&#xc774;&#xb4e4;&#xc744;&#xcd94;&#xac00;&#xd558;&#xace0;&#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xd45c;&#xc900;&#xd654;&#xc791;&#xc5c5;&#xacfc;&#xc815;&#xc744;&#xd5a5;&#xc0c1;&#xc2dc;&#xd0ac;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

2.4.4 Tiger Data 불러오기

```
Tiger &#xb370;&#xc774;&#xd130;&#xb97c;&#xb85c;&#xb529;&#xd558;&#xae30;&#xc704;&#xd55c;&#xbcf4;&#xb2e4;&#xc790;&#xc138;&#xd55c;&#xc124;&#xba85;&#xc740; extras/tiger_geocoder/tiger_2011/README&#xc5d0;&#xc774;&#xc6a9;&#xac00;&#xb2a5;&#xd569;&#xb2c8;&#xb2e4;.&#xc5ec;&#xae30;&#xc11c;&#xb294;&#xc77c;&#xb818;&#xacfc;&#xc815;&#xb9cc;&#xc548;&#xb0b4;&#xd574;&#xb4dc;&#xb9bd;&#xb2c8;&#xb2e4;.
```

```
&#xc778;&#xad6c;&#xc870;&#xc0ac;&#xc6f9;&#xc0ac;&#xc774;&#xd2b8;&#xc5d0;&#xc11c;&#xd544;&#xc694;&#xd55c;&#xc8fc;&#xc758;&#xb370;&#xc774;&#xd130;&#xb97c;&#xb2e4;&#xc6b4;&#xb85c;&#xb4dc;&#xb81b;&#xc2b5;&#xb2c8;&#xc555;&#xcd95;&#xc744;&#xd480;&#xba74;&#xc8fc;&#xb2e8;&#xc704;&#xb85c;&#xbcc4;&#xac1c;&#xc758;&#xc138;&#xc774;&#xb904;&#xc838;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.&#xac01;&#xc8fc;&#xd14c;&#xc774;&#xbe14;&#xc740;
```

tiger_drop_state_tables_generate_script.shxz

Drop State Tables Generate Script

- Drop State Tables Generate Script

Unix

7zip

- PostGIS

- 7zip

gnuw32.sourceforge.net/packages/wget.htm

tiger_loader_generate_nation_script.shxz

Loader Generate Nation Script

install_missing_indexes.shxz

```
SELECT install_missing_indexes();
```

Geocode

2.4.5 Tiger Geocoder

PostGIS 2.0

```
&#xd568;&#xaed8; &#xc124;&#xce58;&#xb418;&#xc9c0; &#xc54a;&#xc740; TIGER &#xc9c0;&#xc624;&#xcf54;&#xb529;
&#xb3c4;&#xad6c;&#xb77c;&#xc57c;&#xb9cc; &#xac00;&#xb2a5;&#xd569;&#xb2c8;&#xb2e4;.
```

extras 폴더가 없을 경우,, <http://postgis.net/stuff/postgis-3.3.9dev.tar.gz> 에다운로드하십시오.

```
tar xvfz postgres-3.3.9dev.tar.gz
```

```
cd postgis-3.3.9dev/extras/tiger_geocoder/tiger_2011
```

윈도우시스템이라면 upgrade_geocoder.bat 스&#xLinux/Unix/Mac OSX 시스템이라면 upgrade_geocoder.sh 스크&#찾으십시오. 사용자의 PostGIS 데이터&#xb사양에 맞춰 파일을 편집하십시&#x

2010이나 2011을 업그레이드하는 경우,
로더 스크립트의 해당 라인을 주&#
처리(unremark out)해야 2012 데이터를 로드하&#
위한 최신 스크립트를 얻을 수 있&#
점을 기억하십시오.

અ령 프롬프트에서 각 플랫폼에
상응하는 스크립트를 실행하십&#x

```
&#xb2e4;&#xc74c;&#xc73c;&#xb85c;&#xbaa8;&#xb4e0; nation &#xd14c;&#xc774;&#xbe14;&#xc744; drop &#xd558;&#xace0;
&#xc0c8;&#xb85c;&#xbd88;&#xb7ec;&#xc635;&#xb2c8;&#xb2e4;. &#xc774; SQL &#xbb38;&#xc7a5;&#xc73c;&#xb85c;
drop &#xc2a4;&#xd06c;&#xb9bd;&#xd2b8;&#xb97c; &#xb9cc;&#xb4ed;&#xb2c8;&#xb2e4;. &#xc790;&#xc138;&#xd55c;
&#xb0b4;&#xc6a9;&#xc740; &#xb2e4;&#xc74c;&#xc744; &#xcc38;&#xace0;&#xd558;&#xc2ed;&#xc2dc;&#xc624;. Drop_Nation
```

```
SELECT drop_nation_tables_generate_script();
```

생성된 drop SQL 문장을 실행하십시오

```
&#xc774; SELECT &#xad6c;&#xb38;&#xc73c;&#xb85c; nation load &#xc2a4;&#xd06c;&#xb9bd;&#xd2b8;&#xb97c; &#xc0dd;&#xc790;&#xc138;&#xd55c; &#xb0b4;&#xc6a9;&#xc744; &#xb2e4;&#xc74c;&#xc744; &#xcc38;&#xace0;&#xd558;&#xc2ed;&#xa
```

Loader_Generate_Nation_Script

8

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

unix/linux 용

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

생성된스크립트를어떻게실행ର우시려면 Section 2.4.4을 참고하십시오.이것은 단지 한번만하면 됩니다.

Note



Note!

```
&#xc0ac;&#xc6a9;&#xc790;&#xb294;      2010/2011      state      &#xd14c;&#xc774;&#xbe14;&#xc744;
&#xd569;&#xce60;      &#xc218;&#xb3c4;      &#xc788;&#xace0;      &#xac01;      state      &#xbcc4;&#xb85c;
&#xc5c5;&#xadf8;&#xb808;&#xc774;&#xb4dc; &#xd560; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;
state&#xb97c;      2011&#xb85c;      &#xc5c5;&#xb370;&#xc774;&#xd2b8;      &#xd558;&#xae30;
&#xc804;&#xc5d0;,      Drop_State_Tables_Generate_Script&#xc744;      &#xc774;&#xc6a9;&#xd574;      2010
&#xd14c;&#xc774;&#xbe14;&#xb4e4;&#xc744; drop &#xd574;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;.
```

2.5 설치과정에서 흔히 &#xbc1
문제들

인스톨이나 업그레이드 시 잘 되않을 경우확인해야 할 몇 가지 사아래와 같습니다.

1. PostgreSQL 11 ؐ는그보다최신버전을설체크하십시오. 현재작동하고계시는 PostgreSQL의버전곏동일한 PostgreSQL소스버전을&#xbcl4;탕으로컴파일한확인해주십시오. (Linux) 배포본이이 PostgreSQL이설치된경우중복설치가될수있고.또는이전에설치한사실을잊어버렸을수도있습닄 PostgreSQL는 PostgreSQL 11 또는그이상버전에서작동되므로구버전을이용시예&#xc0c1.하지못한에러들이&#xbcl;생핬수있습니다.어떤 PostgreSQL 버전이작있는지확인하기위해서는 psql을통하여데이터베이스에접속해다음의쿼리를실행하십시오:

```
SELECT version();
```

```
RPM &#xbc29;&#xc2dd;&#xc774;&#xb77c;&#xba74; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc740; rpm &#xcee4;&#xb9e
&#xc774;&#xc804; &#xc124;&#xce58; &#xd328;&#xd0a4;&#xc9c0;&#xc758; &#xc874;&#xc7ac;&#xb97c; &#xd655;&#xc7
&#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.: rpm -qa | grep postgresql
```

2. œ ս ׅ 그 레 이 드 가 실 패 하 면 기 존 PostGIS 설 치 버 전 으 로 복 원 하 셔 합 니 다.

```
SELECT postgis_full_version();
```

또한 PostreSQL, Proj4 라이브러리 및 GEOS 라이브대한 버전과 경로에 대한 설정이 올바른지 확인하십시오.

1. 설정은postgis_config.hh파일을생성하기
위해사용됩니다. POSTGIS_PGSQL_VERSION,POSTGIS_PROJ_VERS
andPOSTGIS GEOS VERSION변수가정확한지확인&#

Chapter 3

PostGIS Administration

3.1 Performance Tuning

Tuning for PostGIS performance is much like tuning for any PostgreSQL workload. The only additional consideration is that geometries and rasters are usually large, 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+ configuration 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 forces non-startup configs to take effect for new connections
SELECT pg_reload_conf();
-- show current setting value
-- use SHOW ALL to see all settings
SHOW work_mem;
```

In addition to the Postgres settings, PostGIS has some custom settings which are listed in [Section 8.23](#).

3.1.1 Startup

These settings are configured in `postgresql.conf`:

`constraint_exclusion`

- Default: partition
- This is generally used for table partitioning. The default for this is set to "partition" which is ideal for PostgreSQL 8.4 and above since it will force the planner to only analyze tables for constraint consideration if they are in an inherited hierarchy and not pay the planner penalty otherwise.

`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.

`max_worker_processes` This setting is only available for PostgreSQL 9.4+. For PostgreSQL 9.6+ this setting has additional importance in that it controls the max number of processes you can have for parallel queries.

- Default: 8
 - Sets the maximum number of background processes that the system can support. This parameter can only be set at server start.
-

3.1.2 Runtime

work_mem - sets the size of memory used for sort operations and complex queries

- Default: 1-4MB
- Adjust up for large dbs, complex queries, lots of RAM
- Adjust down for many concurrent users or low RAM.
- If you have lots of RAM and few developers:

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

maintenance_work_mem - the memory size used for VACUUM, CREATE INDEX, etc.

- Default: 16-64MB
- Generally too low - ties up I/O, locks objects while swapping memory
- Recommend 32MB to 1GB on production servers w/lots of RAM, but depends on the # of concurrent users. If you have lots of RAM and few developers:

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

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.

- Default: 0
- 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.

3.2 Configuring raster support

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

As of PostGIS 2.1.3, out-of-db rasters and all raster drivers are disabled by default. In order to re-enable these, you need to set the following environment variables `POSTGIS_GDAL_ENABLED_DRIVERS` and `POSTGIS_ENABLE_OUTDB_RASTERS` in the server environment. For PostGIS 2.2, you can use the more cross-platform approach of setting the corresponding Section 8.23.

If you want to enable offline raster:

```
POSTGIS_ENABLE_OUTDB_RASTERS=1
```

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

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

```
POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL
```

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

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

**Note**

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

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

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

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

3.3 **PostGIS Installation**

3.3.1 Spatially enable database using EXTENSION

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

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

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

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

3.3.2 Spatially enable database without using EXTENSION (discouraged)

**Note**

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

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

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

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

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

Run the following Shell commands in your terminal:


```
DB=[yourdatabase]
SCRIPTSDIR=`pg_config --sharedir`/contrib/postgis-3.2/

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

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

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

3.3.3 Create a spatially-enabled database from a template

Some packaged distributions of PostGIS (in particular the Win32 installers for PostGIS >= 1.1.5) load the PostGIS functions into a template database called `template_postgis`. If the `template_postgis` database exists in your PostgreSQL installation then it is possible for users and/or applications to create spatially-enabled databases using a single command. Note that in both cases, the database user must have been granted the privilege to create new databases.

From the shell:

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

From SQL:

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

3.4 Upgrading spatial databases

Upgrading existing spatial databases can be tricky as it requires replacement or introduction of new PostGIS object definitions. Unfortunately not all definitions can be easily replaced in a live database, so sometimes your best bet is a dump/reload process. PostGIS provides a **SOFT UPGRADE** procedure for minor or bugfix releases, and a **HARD UPGRADE** procedure for major releases.

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

3.4.1 Soft upgrade

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

3.4.1.1 Soft Upgrade 9.1+ using extensions

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

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

```
SELECT postgis_extensions_upgrade();
```

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

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

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

```
ALTER EXTENSION postgis UPDATE TO "3.3.9dev";
ALTER EXTENSION postgis_topology UPDATE TO "3.3.9dev";
```

If you get an error notice something like:

```
No migration path defined for ... to 3.3.9dev
```

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

If you get a notice message like:

```
Version "3.3.9dev" of extension "postgis" is already installed
```

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

```
ALTER EXTENSION postgis UPDATE TO "3.3.9devnext";
ALTER EXTENSION postgis_topology UPDATE TO "3.3.9devnext";
```



Note

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



Note

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

```
DROP EXTENSION postgis_raster;
```

3.4.1.2 Soft Upgrade Pre 9.1+ or without extensions

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

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

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

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

```
ls `pg_config --sharedir`/contrib/postgis-3.3.9dev/*_upgrade.sql
```

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

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

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

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

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

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

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

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



Note

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

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

3.4.2 Hard upgrade

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

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

Supplementary instructions for windows users are available at [Windows Hard upgrade](#).

The Procedure is as follows:

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

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

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

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

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

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

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

Errors may arise in the following cases:

1. Some of your views or functions make use of deprecated PostGIS objects. In order to fix this you may try loading `legacy.sql` script prior to restore or you'll have to restore to a version of PostGIS which still contains those objects and try a migration again after porting your code. If the `legacy.sql` way works for you, don't forget to fix your code to stop using deprecated functions and drop them loading `uninstall_legacy.sql`.
2. Some custom records of `spatial_ref_sys` in dump file have an invalid SRID value. Valid SRID values are bigger than 0 and smaller than 999000. Values in the 999000.999999 range are reserved for internal use while values > 999999 can't be used at all. All your custom records with invalid SRIDs will be retained, with those > 999999 moved into the reserved range, but the `spatial_ref_sys` table would lose a check constraint guarding for that invariant to hold and possibly also its primary key (when multiple invalid SRIDS get converted to the same reserved SRID value).

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

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

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

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

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

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

```
perl utils/postgis_restore.pl "/somepath/olddb.backup" > olddb.sql
```

run this command :

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

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

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

Chapter 4

Data Management

4.1 GIS

4.1.1 OGC Geometry

The Open Geospatial Consortium (OGC) developed the *Simple Features Access* standard (SFA) to provide a model for geospatial data. It defines the fundamental spatial type of **Geometry**, along with operations which manipulate and transform geometry values to perform spatial analysis tasks. PostGIS implements the OGC Geometry model as the PostgreSQL data types **geometry** and **geography**.

Geometry is an *abstract* type. Geometry values belong to one of its *concrete* subtypes which represent various kinds and dimensions of geometric shapes. These include the **atomic** types **Point**, **LineString**, **LinearRing** and **Polygon**, and the **collection** types **MultiPoint**, **MultiLineString**, **MultiPolygon** and **GeometryCollection**. The *Simple Features Access - Part 1: Common architecture v1.2.1* adds subtypes for the structures **PolyhedralSurface**, **Triangle** and **TIN**.

Geometry models shapes in the 2-dimensional Cartesian plane. The PolyhedralSurface, Triangle, and TIN types can also represent shapes in 3-dimensional space. The size and location of shapes are specified by their **coordinates**. Each coordinate has a X and Y **ordinate** value determining its location in the plane. Shapes are constructed from points or line segments, with points specified by a single coordinate, and line segments by two coordinates.

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

Geometry values are associated with a **spatial reference system** indicating the coordinate system in which it is embedded. The spatial reference system is identified by the geometry SRID number. The units of the X and Y axes are determined by the spatial reference system. In **planar** reference systems the X and Y coordinates typically represent easting and northing, while in **geodetic** systems they represent longitude and latitude. SRID 0 represents an infinite Cartesian plane with no units assigned to its axes. See Section 4.5.

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

A geometry value may be **empty**. Empty values contain no vertices (for atomic geometry types) or no elements (for collections).

An important property of geometry values is their spatial **extent** or **bounding box**, which the OGC model calls **envelope**. This is the 2 or 3-dimensional box which encloses the coordinates of a geometry. It is an efficient way to represent a geometry's extent in coordinate space and to check whether two geometries interact.

The geometry model allows evaluating topological spatial relationships as described in Section 5.1.1. To support this the concepts of **interior**, **boundary** and **exterior** are defined for each geometry type. Geometries are topologically closed, so they always contain their boundary. The boundary is a geometry of dimension one less than that of the geometry itself.

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

4.1.1.1 Point

A Point is a 0-dimensional geometry that represents a single location in coordinate space.

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

4.1.1.2 LineString

A LineString is a 1-dimensional line formed by a contiguous sequence of line segments. Each line segment is defined by two points, with the end point of one segment forming the start point of the next segment. An OGC-valid LineString has either zero or two or more points, but PostGIS also allows single-point LineStrings. LineStrings may cross themselves (self-intersect). A LineString is **closed** if the start and end points are the same. A LineString is **simple** if it does not self-intersect.

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

4.1.1.3 LinearRing

A LinearRing is a LineString which is both closed and simple. The first and last points must be equal, and the line must not self-intersect.

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

4.1.1.4 Polygon

A Polygon is a 2-dimensional planar region, delimited by an exterior boundary (the shell) and zero or more interior boundaries (holes). Each boundary is a [LinearRing](#).

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

4.1.1.5 MultiPoint

A MultiPoint is a collection of Points.

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

4.1.1.6 MultiLineString

A MultiLineString is a collection of LineStrings. A MultiLineString is closed if each of its elements is closed.

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


4.1.2.1 CircularString

CIRCULARSTRING은 기본 곡선 유형으로, 선형&
LINESTRING과 비슷합니다. 단일 분절에&#xb29
시작점과 종단점(첫 번째 및 세 번&#
그리고 곡선 위의 다른 한 점 이렇&#
포인트 세 개가 필요합니다. 예외&#
닫힌 원으로, 이 경우 시작점과 종&#
동일합니다. 이 경우 두 번째 포인&#
원호의 중심, 즉 원의 반대편이 되&
합니다. 원호를 함께 묶으려면, LINESTRI
마찬가지로 한 원호의 마지막 포&#
다음 원호의 첫 번째 포인트가 되&#
합니다. 즉 유효한 원형 스트링은
1을 초과하는 홀수 개수의 포인트&#
가져야만 한다는 뜻입니다.

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

4.1.2.2 CompoundCurve

복심곡선(compound curve)이란곡선(원호) 분선형 분절이 함૨있는 연속적인 단일 곡선을 말합니다. 즉 구성 요잘 형성되어야 함은 물론. (마지막제외한) 모든 구성 요소의 종단점다음 구성 요소의 시작점과 일치한다는 &#b73b.입니다.

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

4.1.2.3 CurvePolygon

CURVEPOLYGON은 일 반 폴 리 곤 입 니 다. 외 곽 &#x함 께 0개 이 상 의 내 곽 선 을 가 지 고 있 을 뿐 입 니 다. 차 이 라 면 내 외 곽 &#원 형 스 트 링 , 선 형 스 트 링 , 또 는 복 스 트 링 형 태 를 할 수 있 다 는 점 입 &

PostGIS->1.4->8080->130->9cc->ace1->d3f4->9ac->ace4->xc5d0->300->bcf5->xc2ec->ace1->xc120->xc744->xc9c0->xc6d0->d569->b2c8->b2e4..

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

Example: A CurvePolygon with the shell defined by a CompoundCurve containing a CircularString and a LineString, and a hole defined by a CircularString

```
&#xb9cc;&#xace1; &#xd3f4;&#xb9ac;&#xace4; &#xb0b4;&#xbd80;&#xc758;  ←  
  &#xbcf5;&#xc2ec;&#xace1;&#xc120;&#xc758; &#xc608;: CURVEPOLYGON(COMPOUNDCURVE(  ←  
    CIRCULARSTRING(0 2 0, 2 1, 2 3, 4 3),(4 3, 4 5, 1 4, 0 0)), CIRCULARSTRING(1.7 1, 1.4  ←  
    0.4, 1.6 0.4, 1.6 0.5, 1.7 1) )
```


4.1.2.4 MultiCurve

MULTICURVE는 원형 스트링, 선형 스트링, 복합 스트링을 포함할 수 있는 곡집합입니다.

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

4.1.2.5 MultiSurface

MULTISURFACE는 면 집합으로, (선형) 폴리곤수도 만곡 폴리곤일 수도 있습니만

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

4.1.3 OpenGIS WKB &#x0f; WKT

OpenGIS사양서에는 공간 오브젝트들삘타내는 두 가지표준 방법이 정있습니다. Well-Known Text (WKT) 형태와 Well-Known Binary (WKB) 형태. WKT와 WKB 모두 오브젝트 타입과 오브젝트를 구성하는 좌표들에 대한 정보를 포함하고 있습니다.

공간 참조 시스템의 WKT(Well-Known Text) 표현&#xc2dd
다음은 WKT SRS 표현식의 예입니다:

- POINT(0 0)
- POINT(0 0)
- POINT(0 0)
- POINT EMPTY
- LINESTRING(0 0,1 1,1 2)
- LINESTRING
- POLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1)))
- MULTIPOINT((0 0),(1 2))
- MULTIPOINT((0 0),(1 2))
- MULTIPOINT
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON((((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1))))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))
- GEOMETRYCOLLECTION

Input and output of WKT is provided by the functions `ST_AsText` and `ST_GeomFromText`:

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

```
&#xc608;&#xb97c;&#xb4e4;&#xc5b4; OGC &#xacf5;&#xac04; &#xac1d;&#xccb4;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xace
&#xc0bd;&#xc785;&#xd558;&#xae30; &#xc704;&#xd55c; &#xc720;&#xd6a8;&#xd55c; &#xc0bd;&#xc785; &#xad6c;&#xbb38;&#
&#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc744; &#xac83;&#xc785;&#xb2c8;&#xb2e4;;
```

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

Well-Known Binary (WKB) provides a portable, full-precision representation of spatial data as binary data (arrays of bytes). Examples of the WKB representations of spatial objects are:

- POINT(0 0)
WKB: 010100000000000000000000F03F000000000000F03
- LINESTRING(0 0,1 1,1 2)
WKB: 010200000002000000000000000000400000000000000040000000000000224000000000000002240

Input and output of WKB is provided by the functions [ST_AsBinary](#) and [ST_GeomFromWKB](#):

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

```
&#xc608;&#xb97c;&#xb4e4;&#xc5b4; OGC &#xacf5;&#xac04; &#xac1d;&#xccb4;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xace
&#xc0bd;&#xc785;&#xd558;&#xae30; &#xc704;&#xd55c; &#xc720;&#xd6a8;&#xd55c; &#xc0bd;&#xc785; &#xad6c;&#xbb38;&#
&#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc744; &#xac83;&#xc785;&#xb2c8;&#xb2e4;;
```

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

4.2 Geometry Data Type

PostGIS implements the OGC Simple Features model by defining a PostgreSQL data type called `geometry`. It represents all of the geometry subtypes by using an internal type code (see [도형](#) [유형](#) and [ST_GeometryType](#)). This allows modelling spatial features as rows of tables defined with a column of type `geometry`.

The `geometry` data type is *opaque*, which means that all access is done via invoking functions on geometry values. Functions allow creating geometry objects, accessing or updating all internal fields, and compute new geometry values. PostGIS supports all the functions specified in the OGC [Simple feature access - Part 2: SQL option](#) (SFS) specification, as well many others. See Chapter 8 for the full list of functions.



Note

PostGIS follows the SFA standard by prefixing spatial functions with "ST_". This was intended to stand for "Spatial and Temporal", but the temporal part of the standard was never developed. Instead it can be interpreted as "Spatial Type".

```
OpenGIS &#xc0ac;&#xc591;&#xc11c;&#xb294; &#xacf5;&#xac04; &#xac1d;&#xccb4;&#xc758; &#xb0b4;&#xbd80; &#xc800;&#
&#xd615;&#xc2dd;&#xc774; &#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c; &#xc2dd;&#xbcc4;&#xc790;(SR
&#xd3ec;&#xd568;&#xd558;&#xb3c4;&#xb85d; &#xc694;&#xad6c;&#xd569;&#xb2c8;&#xb2e4;. &#xb370;&#xc774;&#xd130;&#
&#xc0bd;&#xc785;&#xb420; &#xacf5;&#xac04; &#xac1d;&#xccb4; &#xc0dd;&#xc131;&#xc2dc; SRID&#xac00; &#xd544;&#xc69
```

To make querying geometry efficient PostGIS defines various kinds of spatial indexes, and spatial operators to use them. See Section 4.9 and Section 5.2 for details.


```
&#xd53c;&#xccd0;&#xb4e4;&#xc758; &#xacf5;&#xac04; &#xc624;&#xbe0c;&#xc81d;&#xd2b8;&#xb4e4;&#xc758; &#xd14d;&#xb38;&#xc790;&#xc5f4;&#xd45c;&#xd604;&#xb4e4;(WKT) &#xc758; &#xc608;&#xb4e4;&#xb85c;&#xb294; &#xb2e4;&#xc758; &#xc19;&#xc740; &#xc83;&#xb4e4;&#xc774; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;;
```

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

```
&#xb2e4;&#xc74c;&#xacfc; &#xc19;&#xc740; &#xc778;&#xd130;&#xd398;&#xc774;&#xc2a4;&#xb97c; &#xc774;&#xc6a9;&#xc774; &#xd615;&#xc2dd;&#xc744; &#xc785;&#xb825;/&#xcd9c;&#xb825;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&
```

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

```
&#xc608;&#xb97c; &#xb4e4;&#xc5b4; PostGIS &#xacf5;&#xac04; &#xac1d;&#xccb4;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xc0bd;&#xc785;&#xd558;&#xae30; &#xc704;&#xd55c; &#xc720;&#xd6a8;&#xd55c; &#xc0bd;&#xc785; &#xad6c;&#xb38;&#xb2e4;&#xc74c;&#xacfc; &#xc19;&#xc744; &#xc83;&#xc785;&#xb2c8;&#xb2e4;;
```

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

4.3 PostGIS 지리형 유형

```
&#xc9c0;&#xb9ac;&#xd615; &#xc720;&#xd615;&#xc740; (&#xc885;&#xc885; "&#xce21;&#xc9c0;" &#xc88c;&#xd45c;, &#xb610;&#xb294; "&#xc704;&#xb3c4;/&#xacbd;&#xb3c4;"&#xb098; "&#xacbd;&#xb3c4;/&#xc704;&#xb3c4;"&#xb77c;&#xacef;&#xbd88;&#xb9ac;&#xb294;)"&#xc9c0;&#xb9ac;" &#xc88c;&#xd45c;&#xb85c; &#xd45c;&#xd604;&#xb418;&#xb294; &#xacf5;&#xd53c;&#xc98;&#xb97c; &#xc790;&#xccb4;&#xc801;&#xc73c;&#xb85c; &#xc9c0;&#xc6d0;&#xd569;&#xb2c8;&#xb2e4;. &#xc9c0;&#xb9ac; &#xc88c;&#xd45c;&#xb294; &#xc01;&#xb3c4; &#xb2e8;&#xc704;(&#xb3c4;)&#xb97c; &#xc0ac;&#xc6a9;&#xad6c;&#xba74;(&#x7403;&#x9762;) &#xc88c;&#xd45c;&#xc785;&#xb2c8;&#xb2e4;.
```

```
PostGIS &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc740; &#xd3c9;&#xba74;&#xc744; &#xae30;&#xbc18;&#xc73c;&#xb85c; &#xd569;&#xb2c8;&#xb2e4;. &#xd3c9;&#xba74;&#xc0c1;&#xc5d0;&#xc11c; &#xb450; &#xd3ec;&#xc778;&#xd2b8; &#xc0ac;&#xc00;&#xc7a5; &#xc9e7;&#xc740; &#xacbd;&#xb85c;&#xb294; &#xc9c1;&#xc120;&#xc785;&#xb2c8;&#xb2e4;. &#xc989; &#xb370;&#xce74;&#xb974;&#xd2b8; &#xc218;&#xd559;&#xacfc; &#xc9c1;&#xc120; &#xbca1;&#xd130;&#xb97c; &#xc774;&#xc774;&
```

도형에 대해 계산(면적., 거리., 길이., 교차점 등)한다는 뜻입니다.

PostGIS ী리형유형은구면을기반으다합니다. 구면상에서두포인트사다가장짧은경로는대권(&x5927;&x5708;; great circle arc)입니다. 즉지리형에대한계산(면거리,길이,교차점등)은더복잡한수학을이용해서구면상에서이한다는뜻입니다. 더정확하게측지구의실제회전타원운(spheroidal shape)를고려해서계산해야하는데,수학아주복잡해질수밖에없습니다.

기저 수학이 훨씬 더 복잡하기 때지 리형 유형을 위해 정의된 함수도형 유형의 함수보다 더 적습니시간이 지날수록 새 알고리즘이 추가되어 지리형 유형의 역량은 확장될 것입니다.

Like the geometry data type, geography data is associated with a spatial reference system via a spatial reference system identifier (SRID). Any geodetic (long/lat based) spatial reference system defined in the `spatial_ref_sys` table can be used. (Prior to PostGIS 2.2, the geography type supported only WGS 84 geodetic (SRID:4326)). You can add your own custom geodetic spatial reference system as described in [Section 4.5.2](#).

For all spatial reference systems the units returned by measurement functions (e.g. `ST_Distance`, `ST_Length`, `ST_Perimeter`, `ST_Area`) and for the distance argument of `ST_DWithin` are in meters.

4.3.1 공간 테이블 생성

You can create a table to store geography data using the **CREATE TABLE** SQL statement with a column of type `geography`. The following example creates a table with a geography column storing 2D LineStrings in the WGS84 geodetic coordinate system (SRID 4326):

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

The geography type supports two optional type modifiers:

- `유형변경자가허용하는값은다같습니다.` POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON. `또이변경자는Z,M볰ZM이라는접미사를통해차원수제약도지따라서,예를들자면'LINESTRINGM'의변경Ã차원이하의라인스트링만을허것이며,세벸째차원을기준으로췈급할것입니다.마찬가지로'POINTZM'네가지차원의데이터를입력해할것입니다.`
- the SRID modifier restricts the spatial reference system SRID to a particular number. If omitted, the SRID defaults to 4326 (WGS84 geodetic), and all calculations are performed using WGS84.

Examples of creating tables with geography columns:

- POINT: 2D

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

- POINT: 2D

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

- Create a table with 3D (XYZ) POINTs and an explicit SRID of 4326:

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINTZ,4326) );
```

- Create a table with 2D LINESTRING geography with the default SRID 4326:

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

- POINT: 2D

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

Geography fields are registered in the `geography_columns` system view. You can query the `geography_columns` view and see that the table is listed:

```
-- &#xb54;&#xd0c0;&#xb370;&#xc774;&#xd130; &#xdf0;&#xc758; &#xb0b4;&#xc6a9;&#xc744;  ←
    &#xc0b4;&#xd3b4;&#xbd05;&#xc2dc;&#xb2e4;
SELECT * FROM geography_columns;
```

```
&#xb3c4;&#xd615;&#xacfc; &#xb3d9;&#xc77c;&#xd55c; &#xbc29;&#xbc95;&#xc73c;&#xb85c; &#xc778;&#xb371;&#xc2a4;&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;. PostGIS&#xac00; &#xc5f4;&#xc720;&#xd615;&#xc774; &#xc9c0;&#xb9ac;&#xc83;&#xc744; &#xac10;&#xc9c0;&#xd558;&#xace0; &#xc77c;&#xbc18;&#xc801;&#xc778; &#xb3c4;&#xd615; &#xc6a9;&#xd3c9;&#xba74; &#xc778;&#xb371;&#xc2a4; &#xb300;&#xc2e0; &#xc801;&#xc808;&#xd55c; &#xad6c;&#xba74; &#xae30;&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc0dd;&#xc131;&#xd560; &#xc83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
-- &#xd14c;&#xc2a4;&#xd2b8; &#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xad6c;&#xba74;  ←
    &#xc778;&#xb371;&#xc2a4; &#xc0dd;&#xc131;
CREATE INDEX global_points_gix ON global_points USING GIST ( location );
```

4.3.2 PostGIS

You can insert data into geography tables in the same way as geometry. Geometry data will autocast to the geography type if it has SRID 4326. The **EWKT** and **EWKB** formats can also be used to specify geography values.

```
-- &#xd14c;&#xc2a4;&#xd2b8;&#xc6a9; &#xd14c;&#xc774;&#xbe14;&#xc5d0;  ←
    &#xb370;&#xc774;&#xd130;&#xb97c; &#xcd94;&#xac00;&#xd574;&#xbd05;&#xc2dc;&#xb2e4;
INSERT INTO global_points (name, location) VALUES ('Town', ST_GeographyFromText('SRID=4326; ←
POINT(-110 30)')) );
INSERT INTO global_points (name, location) VALUES ('Forest', ST_GeographyFromText('SRID ←
=4326;POINT(-109 29)')) );
INSERT INTO global_points (name, location) VALUES ('London', ST_GeographyFromText('SRID ←
=4326;POINT(0 49)')) );
```

Any geodetic (long/lat) spatial reference system listed in `spatial_ref_sys` table may be specified as a geography SRID. Non-geodetic coordinate systems raise an error if used.

대권 매퍼(Great Circle mapper) 도형 유형은 평면 세계지도 상에서 시애틀곏 런던직선으로 잇는 경로와 레이캬비사이의 아무 의뮿도 없는 데카르󊱰리를 계산합니다 결곏값의 명단위를 "도(degree)"라고 할 수도 있겠지결곏값은 세 포인트 사이의 어떤 실제 각도 차이도 반영하지 않기 때문에 "도"라고 하는 것조차 부정 일이 됩니다.


```
-- <#b3c4;#d615;#c744; #c774;#c6a9;#d55c; #ac70;#b9ac;  ←
  &#acc4;#c0b0; (13.3"<#b3c4;")
SELECT ST_Distance('LINESTRING(-122.33 47.606, 0.0 51.5) '::geometry, 'POINT(-21.96 64.15) ←
  ':: geometry);
```

4.3.3 <#b3c4;#d615;#b370;#c774;#d130;#c720;#d615;#acfc;#c9c0;#b9ac;#b370;#c774;#d130;#c720;#d615;#c744;#c911;#cca9;#d574;#c11c;#c774;#c6a9;#d558;#b294;#acbd;#c6b0;

<#c0c8;#b85c;#c6b4; #c9c0;#b9ac;#d615; #c720;#d615;#c740; #b370;#c774;#d130;#b97c; #acbd;#b3c4;/#c704;#b3c4; #c88c;#d45c;#b85c; #c800;#c7a5;#d560; #c218; #c788;#b3c4;#d574;#c8fc;#c9c0;#b9ac; #b2e8;#c810;#b3c4; #c788;#c2b5;#b2c8;#b2e4; #b3c4;#d615;#b300;#c0c1;#c73c;#b85c; #c815;#c758;#b41c; #d568;#c218;#bcb4;#b2e4; #c9c0;#b9ac;#b300;#c0c1; #d568;#c218;#ac00; #b354; #c801;#ace0; #xadf8; #c815;#c758;#b41c; #d568;#c2e4;#d589;#d558;#b294; #b370; CPU #c2dc;#ac04;#c744; #b354; #b9ce;#c774; #c7a1;#c5<#c0ac;#c6a9;#c790;#ac00; #c120;#d0dd;#d55c; #c720;#d615;#c740; #c0ac;#c6a9;#c790;#b<#e4c;#b4dc;#d558;#b294; #c751;#c6a9; #d504;#b85c;#xadf8;#b7a8; #c601;#c5ed;#c5d0;#xae30;#b300;#d55c;#b300;#b85c; #b3d9;#c791;#d558;#b3c4;#b85d; #c801;#d569;#d55c; #c870;#ac74;#c744; #ac16;#cdb0;#c57c; #d569;#b2c8;#b2e4; #c0ac;#c6a9;#c790; #b370;#c804;#c138;#acc4; #b610;#b294; #xad11;#b300;#d55c; #b300;#b959; #c9c0;#c5ed;#c744; #d3ec;#xad04;#d560; #c608;#c815;#c785;#b2c8;#xae4c; #c544;#b2c8;#ba74; #c2dc; #b3c4; #xad70; #b610;#b294; #xadf8; #c774;#d558;#c758; #c9c0;#c790;#ccb4;#c5d0; #xad6d;#d55c;#c608;#c815;#c785;#b2c8;#xae4c;?

- <#c0ac;#c6a9;#c790; #b370;#c774;#d130;#ac00; #c881;#c740; #c9c0;#c5ed;#c5d0; #xad6d;#c0ac;#c6a9; #ac00;#b2a5;#d55c; #c2e4;#d589;#c131; #bcbf; #xae30;#b2a5;#c131; #xad00;#c801;#d569;#d55c; #d22c;#c601;#c744; #c120;#d0dd;#d558;#ace0; #b3c4;#d615;#c744; #c774;#c6a9;#d558;#b294; #ac83;#c774; #cd5c;#c120;#c758; #d574;#acbd;#cc45;#c774; #b420; #c218;#b3c4; #c788;#c2b5;#b2c8;#b2e4;.
- <#c0ac;#c6a9;#c790; #b370;#c774;#d130;#ac00; #c804;#c138;#acc4; #b610;#b294; #b300;#ac78;#ccd0; #c788;#c744; #acbd;#c6b0; #c9c0;#b9ac;#d615;#c744; #c774;#c6a9;#d558;#c77c;#c77c;#c774; #c5b4;#b5a4; #d22c;#c601;#bcbf;#c744; #c774;#c6a9;#d560;#c9c0; #ace0;#bcbf;#d558;#c9c0; #c54a;#ace0; #c2dc;#c2a4;#d15c;#c744; #be4c;#b4dc;#d560; #c218;#b3c4; #c788;#c2b5;#b2c8;#b2e4; #c0ac;#c6a9;#c790; #b370;#c774;#d130;#b97c; #acbd;#b3c4;/#c704;#b3c4;#b85c; #c800;#c7a5;#d558;#ace0; #c9c0;#b9ac;#d615;#c744; #b300;#c0c1;#c73c;#b85c; #c815;#c758;#b41c; #d568;#c218;#b97c; #c774;#c6a9;#d558;#<#d22c;#c601;#c5d0; #b300;#d55c; #c774;#d574;#ac00; #bd80;#c871;#d558;#ace0; #b530; #acf5;#bd80;#d558;#ace0; #c2f6;#c9c0;#b3c4; #c54a;#c73c;#ba70; #c9c0;#b9ac;#d615; #c0ac;#c6a9;#c2dc; #xae30;#b2a5;#c131;#c774; #c81c;#d55c;#b41c;#b2e4;#b294; #c0ac;#bcb1b;#c544;#b4e4;#c77c; #c900;#be44;#ac00; #b418;#c5b4; #c788;#b2e4;#ba74; #b3c4; #c9c0;#b9ac;#d615;#c744; #c0ac;#c6a9;#d558;#b294; #d3b8;#c774; #b354; #c26c;#c6b8; #c218; #c788;#c2b5;#b2c8;#b2e4; #xadf8;#b0e5; #c0ac;#c6a9;#c790; #b370;#c774;#d130;#acbd;#b3c4;/#c704;#b3c4;#b85c; #b85c;#b4dc;#d55c; #b2e4;#c74c; #c791;#c5c5;#c744; #c2dc;#c791;#d558;#c2ed;#c2dc;#c624;.

<#c9c0;#b9ac;#d615;#acfc; #b3c4;#d615;#c744; #ac01;#ac01; #c9c0;#c6d0;#d558;#b294; #d568;#c218;#b97c; #be44;#xad50;#d574;#bcbf4;#b824;#ba74; Section 15.11 #b97c; #cc38;#c870; #c9c0;#b9ac;#d615; #d568;#c218;#c758; #baa9;#b85d; #bcbf; #c124;#ba85;#c744; #ac04;#c0b4;#d3b4;#bcbf4;#b824;#ba74; Section 15.4 #b97c; #cc38;#c870;#d558;#c2ed;#c2dc;#c624;.

4.3.4 FAQ

- `계산작업시구체상에서계산하됩니까회전타원체상에서하까됩니까?`

`기본적으로모든거리௏면적계산은회전타원체상에서이எ좁은지역을대상으로한계산౵결과와해당지역에적절한투ౠ적용한평&#ba74;상계산결과는일Å것입니다더넓은지역이라&#ba74;투영볹을적용한평&#ba74;상계산௏회전타원체상계산이언제나더정확할것입니다최종불파&#xb;'FALSE'를설정하&#ba74;&#baa8;&#b4e0;지리형함డ구체상계산을할수있습니다이렇게하&#ba74;계산속도가조금빨라질것입니다특히도형들Ç매우단순한경우에말입니다.`
- `날짜변경선과남극/북극은어୛&#baa8;든계산은날짜변경선이나양극을고려하지않고이루어ಝ좌표가회전타원체(경도/위도)Ç때문에날짜변경선을지나는형상이라도계산이라는관점౜보&#ba74;다른어떤형상과도다를보가없습니다.`
- `공간처리할수있는가장긴원Ð&#bb34;엇인가요?`

`두포인트사이의보간라인으&#xb;대권호(great circle arc)를이용합니다즉대따라어느볹향으로이동하느°따라두포인트가실제로는두가지볹식으로만난다는뜻입ବ&#baa8;&#b4e0;코드는포인트들이대권౴따라가는두경로가운데짧은'경로로만난다고가정합니다결과적으로이상의원호를가진다&#ba74;정확히&#baa8;델링된형ఌ아니게됩니다.`
- `유럽이나러시아의&#ba74;적을계ఋ또는광대한지역을삽입하는작업이이렇게느린이유가뭐É폴리곤이너무나크기&#b54c;&#bb38;이ಜ광대한지역은두가지이유로좋지않습니다&#ba3c;저경계가워°길기&#b54c;&#bb38;에어떤쿼리를실행ൕ인덱스가피처전체를읽어오଩경향이있습니다그리고꼭짓ಁ개수도너&#bb34;만아서거리볰폐여부등의테스트를할&#b54c;적어³한본때로는&n;&#bcb8;이상(이때&n;은다&#후&#bcbf4;피처의꼭짓점개수)꼭짓ಁ&#baa9;&#b85d;전체&#b97c;훑어야하기&#b54c;&#bb38;&#xc7`

```

&#xb3c4;&#xd615;&#xc758; &#xacbd;&#xc6b0;, &#xb300;&#xc6a9;&#xb7c9; &#xd3f4;&#xb9ac;&#xace4;&#xc744;
&#xb300;&#xc0c1;&#xc73c;&#xb85c; &#xc881;&#xc740; &#xc9c0;&#xc5ed;&#xc5d0; &#xb300;&#xd55c; &#xcffc;&#xb9a
&#xd560; &#xb54c; &#xc0ac;&#xc6a9;&#xc790; &#xb3c4;&#xd615; &#xb370;&#xc774;&#xd130;&#xb97c; &#xb354;
&#xc791;&#xc740; &#xb369;&#xc5b4;&#xb9ac;&#xb4e4;&#xb85c; "&#xbe44;&#xc815;&#xaddc;&#xd654;"&#xd574;&#xc
&#xc778;&#xb371;&#xc2a4;&#xac00; &#xd6a8;&#xc728;&#xc801;&#xc73c;&#xb85c; &#xac1d;&#xccb4;&#xc758;
&#xc77c;&#xbd80;&#xbd84;&#xc744; &#xd558;&#xc704; &#xcffc;&#xb9ac;(subquery)&#xd560; &#xc218; &#xc788;&#xb3
&#xb9cc;&#xb4e4;&#xc5b4; &#xcffc;&#xb9ac; &#xc2dc; &#xb9e4;&#xbc88; &#xc804;&#xccb4; &#xac1d;&#xccb4;&#xb97
&#xc77d;&#xc5b4;&#xc62c; &#xd544;&#xc694;&#xac00; &#xc5c6;&#xb3c4;&#xb85d; &#xd558;&#xb294; &#xd3b8;&#xc7
&#xc88b;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc720;&#xb7fd; &#xc804;&#xccb4;&#xb97c; &#xd3f4;&#xb9ac;&#xace4;
&#xd55c; &#xac1c;&#xb85c; &#xc800;&#xc7a5;&#xd560; &#xc218; &#xc788;&#xb2e4;&#xace0; &#xd574;&#xc11c;
&#xaf2d; &#xadf8;&#xb807;&#xac8c; &#xd574;&#xc57c; &#xd55c;&#xb2e4;&#xb294; &#xb73b;&#xc740; &#xc544;&#xb2

```

4.4 Geometry Validation

PostGIS is compliant with the Open Geospatial Consortium's (OGC) Simple Features specification. That standard defines the concepts of geometry being *simple* and *valid*. These definitions allow the Simple Features geometry model to represent spatial objects in a consistent and unambiguous way that supports efficient computation. (Note: the OGC SF and SQL/MM have the same definitions for simple and valid.)

4.4.1 Simple Geometry

A *simple* geometry is one that has no anomalous geometric points, such as self intersection or self tangency.

```

POINT &#xb780; 0&#xcc28;&#xc6d0; &#xb3c4;&#xd615; &#xac1d;&#xccb4;&#xb85c;&#xc11c; &#xc0c1;&#xc18d;&#xc801;&#xc
&#xb2e8;&#xc21c;&#xd615; &#xc785;&#xb2c8;&#xb2e4;.

```

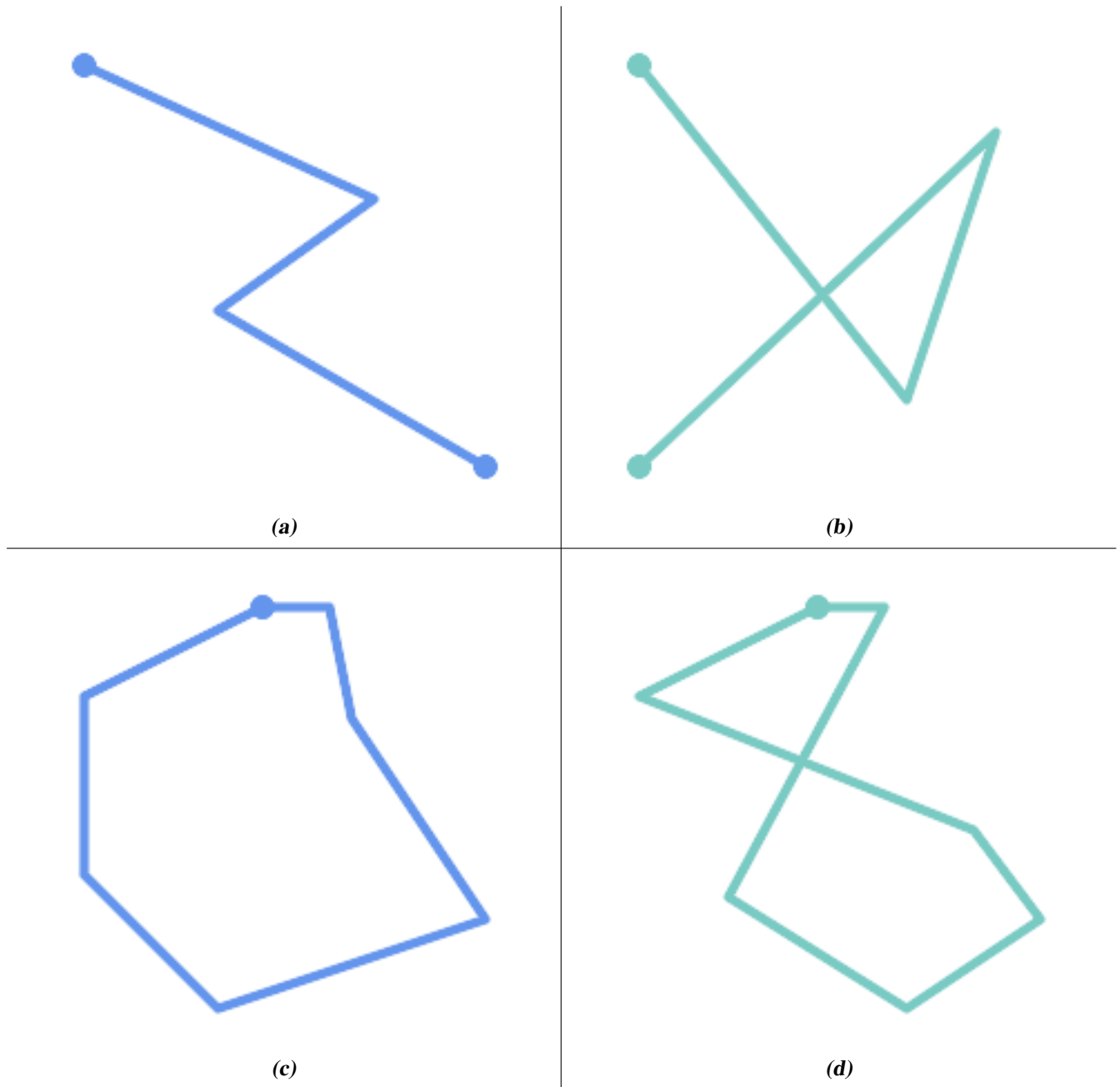
```

MULTIPOINT &#xb294; &#xc5b4;&#xb5a4; &#xb450; &#xc88c;&#xd45c;(POINT)&#xb3c4; &#xb3d9;&#xc77c;&#xd558;&#xc9c
&#xc54a;&#xc740; (&#xb3d9;&#xc77c;&#xd55c; &#xc88c;&#xd45c;&#xb97c; &#xacf5;&#xc720;&#xd558;&#xc9c0; &#xc54a;&#xc
&#xb2e8;&#xc21c;&#xd615; &#xc785;&#xb2c8;&#xb2e4;.

```

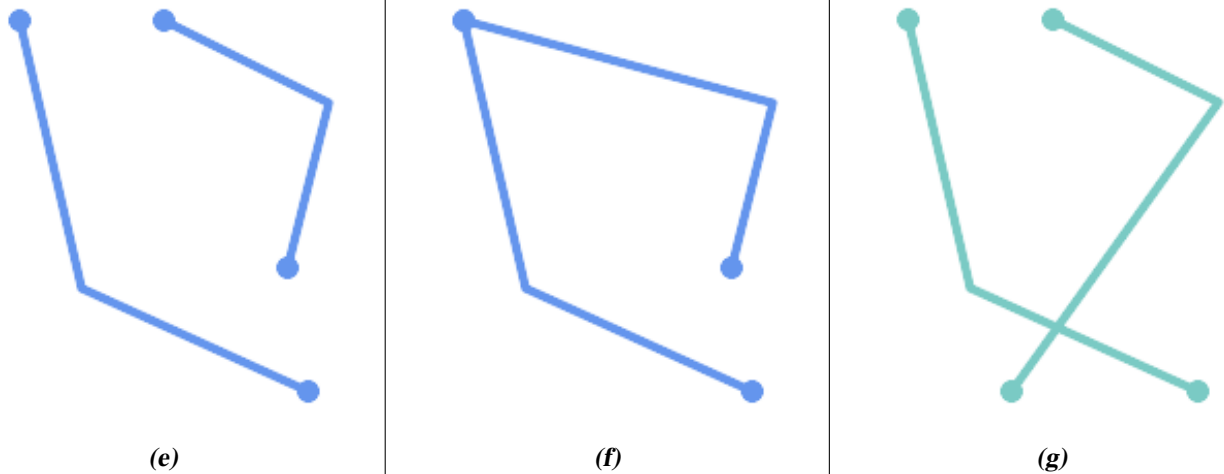
A **LINestring** is *simple* if it does not pass through the same point twice, except for the endpoints. If the endpoints of a simple LineString are identical it is called *closed* and referred to as a Linear Ring.

(a) and (c) are simple LINESTRINGS. (b) and (d) are not simple. (c) is a closed Linear Ring.



A `MULTILINESTRING` is *simple* only if all of its elements are simple and the only intersection between any two elements occurs at points that are on the boundaries of both elements.

(e) and (f) are *simple* `MULTILINESTRING`s. (g) is *not* simple.



POLYGONS are formed from linear rings, so valid polygonal geometry is always *simple*.

To test if a geometry is simple use the **ST_IsSimple** function:

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

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

Generally, PostGIS functions do not require geometric arguments to be simple. Simplicity is primarily used as a basis for defining geometric validity. It is also a requirement for some kinds of spatial data models (for example, linear networks often disallow lines that cross). Multipoint and linear geometry can be made simple using **ST_UnaryUnion**.

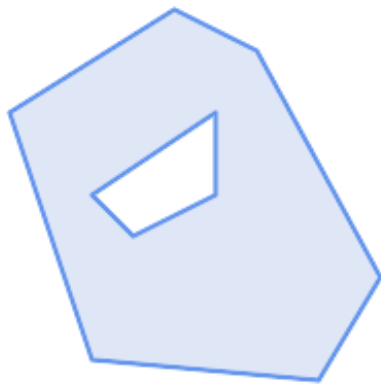
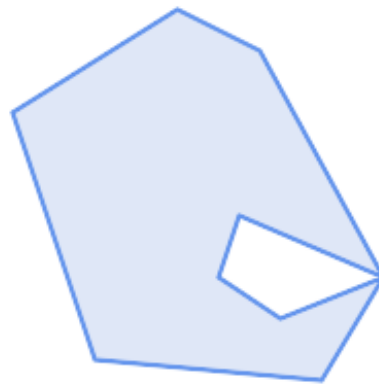
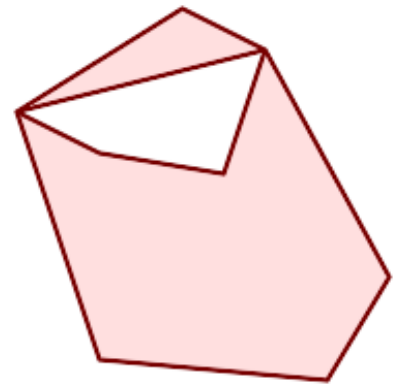
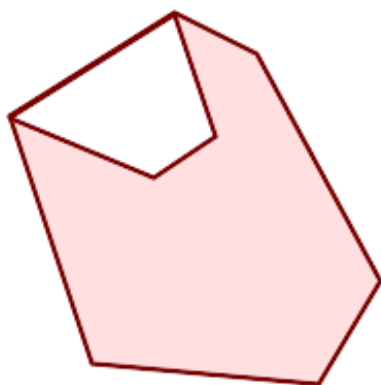
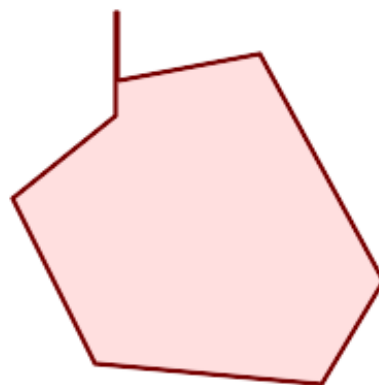
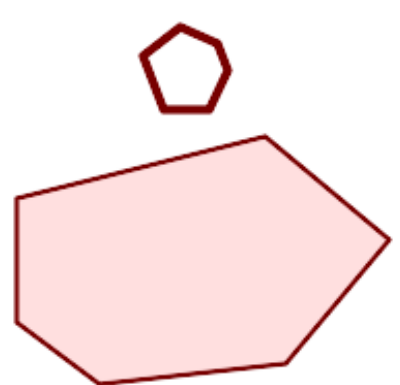
4.4.2 Valid Geometry

Geometry validity primarily applies to 2-dimensional geometries (POLYGONS and MULTIPOLYGONS). Validity is defined by rules that allow polygonal geometry to model planar areas unambiguously.

A POLYGON is *valid* if:

1. the polygon boundary rings (the exterior shell ring and interior hole rings) are *simple* (do not cross or self-touch). Because of this a polygon cannot have cut lines, spikes or loops. This implies that polygon holes must be represented as interior rings, rather than by the exterior ring self-touching (a so-called "inverted hole").
2. boundary rings do not cross
3. boundary rings may touch at points but only as a tangent (i.e. not in a line)
4. interior rings are contained in the exterior ring
5. the polygon interior is simply connected (i.e. the rings must not touch in a way that splits the polygon into more than one part)

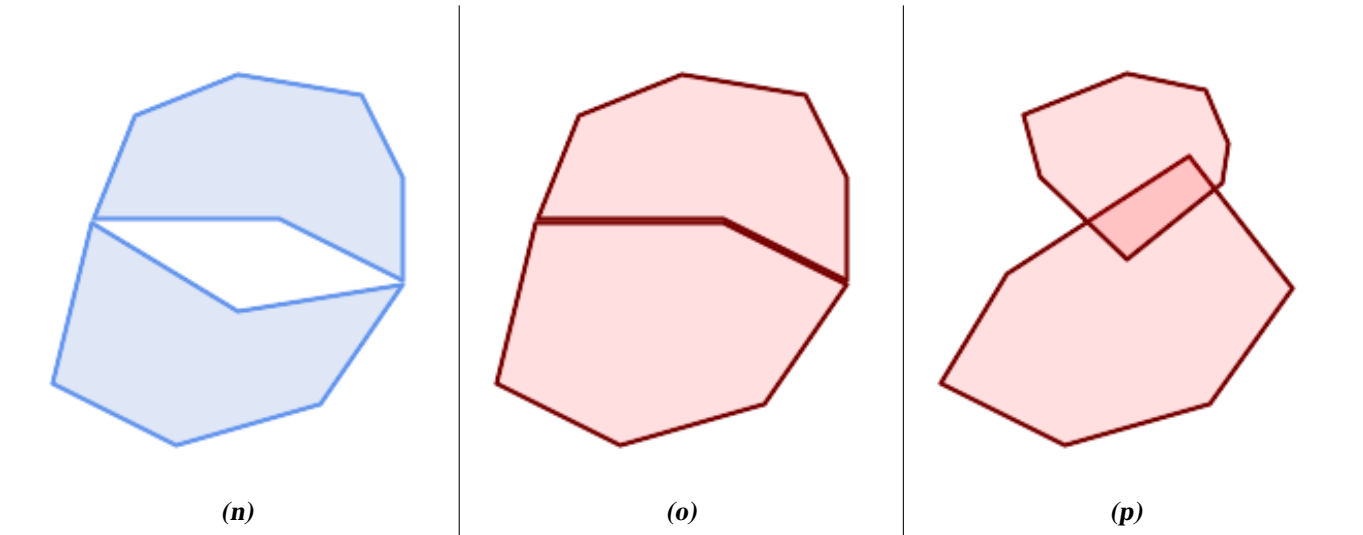
(h) and (i) are valid POLYGONS. (j-m) are invalid. (j) can be represented as a valid MULTIPOLYGON.

**(h)****(i)****(j)****(k)****(l)****(m)**

A MULTIPOLYGON is *valid* if:

1. its element POLYGONS are valid
2. elements do not overlap (i.e. their interiors must not intersect)
3. elements touch only at points (i.e. not along a line)

(n) is a valid MULTIPOLYGON. **(o)** and **(p)** are invalid.



These rules mean that valid polygonal geometry is also *simple*.

For linear geometry the only validity rule is that `LINESTRINGS` must have at least two points and have non-zero length (or equivalently, have at least two distinct points.) Note that non-simple (self-intersecting) lines are valid.

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

len_nonzero	len_zero	self_int
t	f	t

`POINT` and `MULTIPOINT` geometries have no validity rules.

4.4.3 Managing Validity

PostGIS allows creating and storing both valid and invalid Geometry. This allows invalid geometry to be detected and flagged or fixed. There are also situations where the OGC validity rules are stricter than desired (examples of this are zero-length linestrings and polygons with inverted holes.)

Many of the functions provided by PostGIS rely on the assumption that geometry arguments are valid. For example, it does not make sense to calculate the area of a polygon that has a hole defined outside of the polygon, or to construct a polygon from a non-simple boundary line. Assuming valid geometric inputs allows functions to operate more efficiently, since they do not need to check for topological correctness. (Notable exceptions are that zero-length lines and polygons with inversions are generally handled correctly.) Also, most PostGIS functions produce valid geometry output if the inputs are valid. This allows PostGIS functions to be chained together safely.

If you encounter unexpected error messages when calling PostGIS functions (such as "GEOS Intersection() threw an error!"), you should first confirm that the function arguments are valid. If they are not, then consider using one of the techniques below to ensure the data you are processing is valid.



Note If a function reports an error with valid inputs, then you may have found an error in either PostGIS or one of the libraries it uses, and you should report this to the PostGIS project. The same is true if a PostGIS function returns an invalid geometry for valid input.

To test if a geometry is valid use the `ST_IsValid` function:

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

Information about the nature and location of an geometry invalidity are provided by the [ST_IsValidDetail](#) function:

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

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

In some situations it is desirable to correct invalid geometry automatically. Use the [ST_MakeValid](#) function to do this. ([ST_MakeValid](#) is a case of a spatial function that *does* allow invalid input!)

By default, PostGIS does not check for validity when loading geometry, because validity testing can take a lot of CPU time for complex geometries. If you do not trust your data sources, you can enforce a validity check on your tables by adding a check constraint:

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

4.5 SPATIAL_REF_SYS

A [Spatial Reference System](#) (SRS) (also called a Coordinate Reference System (CRS)) defines how geometry is referenced to locations on the Earth's surface. There are three types of SRS:

- A **geodetic** SRS uses angular coordinates (longitude and latitude) which map directly to the surface of the earth.
- A **projected** SRS uses a mathematical projection transformation to "flatten" the surface of the spheroidal earth onto a plane. It assigns location coordinates in a way that allows direct measurement of quantities such as distance, area, and angle. The coordinate system is Cartesian, which means it has a defined origin point and two perpendicular axes (usually oriented North and East). Each projected SRS uses a stated length unit (usually metres or feet). A projected SRS may be limited in its area of applicability to avoid distortion and fit within the defined coordinate bounds.
- A **local** SRS is a Cartesian coordinate system which is not referenced to the earth's surface. In PostGIS this is specified by a SRID value of 0.

There are many different spatial reference systems in use. Common SRSEs are standardized in the European Petroleum Survey Group [EPSG database](#). For convenience PostGIS (and many other spatial systems) refers to SRS definitions using an integer identifier called a SRID.

A geometry is associated with a Spatial Reference System by its SRID value, which is accessed by [ST_SRID](#). The SRID for a geometry can be assigned using [ST_SetSRID](#). Some geometry constructor functions allow supplying a SRID (such as [ST_Point](#) and [ST_MakeEnvelope](#)). The [EWKT](#) format supports SRIDs with the `SRID=n;` prefix.

Spatial functions processing pairs of geometries (such as [overlay](#) and [relationship](#) functions) require that the input geometries are in the same spatial reference system (have the same SRID). Geometry data can be transformed into a different spatial reference system using [ST_Transform](#). Geometry returned from functions has the same SRS as the input geometries.

4.5.1 SPATIAL_REF_SYS Table

The `SPATIAL_REF_SYS` table used by PostGIS is an OGC-compliant database table that defines the available spatial reference systems. It holds the numeric SRIDs and textual descriptions of the coordinate systems.

`SPATIAL_REF_SYS`

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

srid The SRID value. For example, 4326 is the SRID for the WGS 1984 datum. The SRID is a numeric value that identifies the coordinate system. The SRID is a numeric value that identifies the coordinate system. The SRID is a numeric value that identifies the coordinate system.

auth_name The name of the authority that defines the coordinate system. For example, "EPSG" is the authority for the EPSG database. The auth_name is a text value that identifies the authority that defines the coordinate system. The auth_name is a text value that identifies the authority that defines the coordinate system.

auth_srid The ID of the Spatial Reference System as defined by the Authority cited in the `auth_name`. In the case of EPSG, this is the EPSG code.

srtext The Well-Known Text (WKT) representation of the coordinate system. The srtext is a text value that represents the coordinate system in WKT format. The srtext is a text value that represents the coordinate system in WKT format.

```
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]
]
```

For a discussion of SRS WKT, see the OGC standard [Well-known text representation of coordinate reference systems](#).

proj4text The Proj4 text representation of the coordinate system. The proj4text is a text value that represents the coordinate system in Proj4 format. The proj4text is a text value that represents the coordinate system in Proj4 format.

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

The `spatial_ref_sys` table is used to store the spatial reference systems. The `spatial_ref_sys` table is used to store the spatial reference systems. The `spatial_ref_sys` table is used to store the spatial reference systems.

- If `auth_name` and `auth_srid` are present (non-NULL) use the PROJ SRS based on those entries (if one exists).
- If `srttext` is present create a SRS using it, if possible.
- If `proj4text` is present create a SRS using it, if possible.

```
PostGIS&#xc758; SPATIAL_REF_SYS &#xd14c;&#xc774;&#xbe14;&#xc774; proj &#xb77c;&#xc774;&#xbe0c;&#xb7ec;&#xb9ac
&#xcc98;&#xb9ac;&#xd560; &#xc218; &#xc788;&#xb294;,&#xc880; &#xb354; &#xb110;&#xb9ac; &#xc0ac;&#xc6a9;&#xb418;&
&#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c; 3000&#xc5ec; &#xac1c;&#xb97c; &#xb2f4;&#xace0;
&#xc788;&#xae34; &#xd558;&#xc9c0;&#xb9cc; &#xd604;&#xc7ac;&#xae4c;&#xc9c0; &#xc54c;&#xb824;&#xc9c4; &#xbaa8;&#x
&#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c;&#xc744; &#xb2e4; &#xb2f4;&#xace0; &#xc788;&#xc9c0;&#x
&#xc54a;&#xc73c;&#xba70;,&#xc0ac;&#xc6a9;&#xc790;&#xac00; proj4&#xc758; &#xad6c;&#xc870;&#xb97c; &#xc798;
&#xc54c;&#xace0; &#xc788;&#xb2e4;&#xba74; &#xc790;&#xae30;&#xb9cc;&#xc758; &#xc0ac;&#xc6a9;&#xc790; &#xc9c0;&#x
&#xd22c;&#xc601;&#xc744; &#xc815;&#xc758;&#xd560; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.&#xacf5;&#
&#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c; &#xb300;&#xbd80;&#xbd84;&#xc740; &#xd2b9;&#xc815; &#xc9c0;&#xc5ed;&#
&#xd2b9;&#xd654;&#xb418;&#xc5b4; &#xc788;&#xc73c;&#xba70;,&#xd2b9;&#xd654;&#xb41c; &#xc9c0;&#xc5ed; &#xbc94;&#
&#xbc14;&#xae65;&#xc5d0;&#xc11c; &#xc0ac;&#xc6a9;&#xd560; &#xacbd;&#xc6b0; &#xc5b4;&#xb5a4; &#xc758;&#xbbf8;&#x
&#xc5c6;&#xb2e4;&#xb294; &#xc810;&#xc744; &#xba85;&#xc2ec;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

좀더널리사용되는공간참조시&#x4326 - WGS 84 Long Lat, 4269 - NAD 83 Long Lat, 3395 - WGS 84 World Mercator, 2163 - US National Atlas Equal Area, 그리고 NAD 83 및 WGS 84 UTM 대(ึ zone)의공간참조시스템이있습니다. 각 UTM 대&#xb2డ정에가장이상적인공간참조시스템이지만, 6도범위의지역에특화되어있습니다.

You can even define non-Earth-based coordinate systems, such as **Mars 2000**. This Mars coordinate system is non-planar (it's in degrees spheroidal), but you can use it with the `geography` type to obtain length and proximity measurements in meters instead of degrees.

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

4.6

4.6.1

You can create a table to store geometry data using the **CREATE TABLE** SQL statement with a column of type `geometry`. The following example creates a table with a geometry column storing 2D (XY) LineStrings in the BC-Albers coordinate system (SRID 3005):

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

The `geometry` type supports two optional **type modifiers**:

- the **geometry type** restricts the `geometry` type to a specific geometry type. The supported types are: `POINT`, `LINESTRING`, `POLYGON`, `MULTIPOINT`, `MULTILINESTRING`, `MULTIPOLYGON`, `Z`, `M`, `ZM`, `LINESTRINGM`, `POLYGONM`, `MULTIPOLYGONM`, `POINTZ`, `POINTM`, `LINESTRINGZ`, `LINESTRINGM`, `POLYGONZ`, `POLYGONM`, `MULTIPOLYGONZ`, `MULTIPOLYGONM`, `POINTZM`, `POINTM`, `LINESTRINGZM`, `LINESTRINGM`, `POLYGONZM`, `POLYGONM`, `MULTIPOLYGONZM`, `MULTIPOLYGONM`. The `geometry` type also supports the `SRID` modifier.
- the **SRID modifier** restricts the **spatial reference system** SRID to a particular number. If omitted, the SRID defaults to 0.

Examples of creating tables with geometry columns:

- Create a table holding any kind of geometry with the default SRID:

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

- Create a table with 2D POINT geometry with the default SRID:

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

- Create a table with 3D (XYZ) POINTs and an explicit SRID of 3005:

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

- Create a table with 4D (XYZM) LINESTRING geometry with the default SRID:

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```

- Create a table with 2D POLYGON geometry with the SRID 4267 (NAD 1927 long lat):

```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINTZ,4326) );
```

It is possible to have more than one geometry column in a table. This can be specified when the table is created, or a column can be added using the **ALTER TABLE** SQL statement. This example adds a column that can hold 3D LineStrings:

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



```
&#xc788;&#xb294;&#xb370; SQL &#xbdf0; &#xadf8;&#xb9ac;&#xace0; &#xb300;&#xaddc;&#xbaa8; &#xc0bd;&#xc785;(bulk
insert)&#xc758; &#xacbd;&#xc6b0;&#xc785;&#xb2c8;&#xb2e4;. &#xc774;&#xb7f0; &#xacbd;&#xc6b0;,&#xd574;&#xb2f9;
&#xc5f4;&#xc5d0; &#xc81c;&#xc57d; &#xc870;&#xac74;&#xc744; &#xac78;&#xc5b4;&#xc11c; geometry_columns &#xd14c;&#x
&#xb4f1;&#xb85d;&#xc744; &#xb14;&#xb85c;&#xc7a1;&#xc744; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. Post-
GIS 2.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc5d0;&#xc11c;&#xb294;,&#xc0ac;&#xc6a9;&#xc790; &#xc5f4;&#xc774;
typmod &#xae30;&#xb18;&#xc774;&#xb77c;&#xba74; &#xc0dd;&#xc131; &#xacfc;&#xc815; &#xc911;&#xc5d0; &#xc815;&#xd
&#xb4f1;&#xb85d;&#xd560; &#xac83;&#xc774;&#xae30; &#xb54c;&#xbb38;&#xc5d0; &#xc544;&#xbb34;&#xac83;&#xb3c4;
&#xd560; &#xd544;&#xc694;&#xac00; &#xc5c6;&#xb2e4;&#xb294; &#xc810;&#xc744; &#xae30;&#xc5b5;&#xd558;&#xc2ed;&#x
```

```
-- &#xc774;&#xb807;&#xac8c; &#xc0dd;&#xc131;&#xb41c; &#xbdf0;&#xac00; ←
&#xc788;&#xb2e4;&#xace0; &#xd569;&#xc2dc;&#xb2e4;.
CREATE VIEW public.vwmytablemercator AS
    SELECT gid, ST_Transform(geom,3395) As geom, f_name
    FROM public.mytable;

-- PostGIS 2.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc5d0; &#xc815;&#xd655;&#xd558;&#xac8c; ←
&#xb4f1;&#xb85d;&#xd558;&#xb824;&#xba74;
-- &#xb3c4;&#xd615;&#xc744; &#xd615;&#xbcc0;&#xd658;&#xd574;&#xc57c; ←
&#xd569;&#xb2c8;&#xb2e4;.
--
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;

-- &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc774; 2D &#xd3f4;&#xb9ac;&#xace4;&#xc774;&#xb780; ←
&#xc0ac;&#xc2e4;&#xc744; &#xd655;&#xc2e4;&#xd788; &#xc54c;&#xace0; &#xc788;&#xc744; ←
&#xacbd;&#xc6b0; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc774; &#xd560; &#xc218; ←
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
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;
```

```
-- &#xb300;&#xaddc;&#xbaa8; &#xc0bd;&#xc785; &#xc791;&#xc5c5;&#xc744; &#xd1b5;&#xd574; ←
&#xd30c;&#xc0dd; &#xd14c;&#xc774;&#xbel4;&#xc744; ←
&#xc0dd;&#xc131;&#xd588;&#xb2e4;&#xace0; &#xd569;&#xc2dc;&#xb2e4;.
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);

-- &#xc0c8; &#xd14c;&#xc774;&#xbel4;&#xc5d0; 2D &#xc778;&#xb371;&#xc2a4;&#xb97c; ←
&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
CREATE INDEX idx_myschema_myspecialpois_geom_gist
ON myschema.my_special_pois USING gist(geom);

-- &#xc0ac;&#xc6a9;&#xc790; &#xd3ec;&#xc778;&#xd2b8;&#xac00; 3D &#xb610;&#xb294; 3M ←
&#xd3ec;&#xc778;&#xd2b8;&#xc77c; &#xacbd;&#xc6b0;,
-- 2D &#xc778;&#xb371;&#xc2a4;&#xac00; &#xc544;&#xb2c8;&#xb77c; nD ←
&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xb294; &#xd3b8;&#xc774; ←
&#xc88b;&#xc744; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
-- &#xb2e4;&#xc74c;&#xcc98;&#xb7fc; &#xb9d0;&#xc774;&#xc8e0;.
CREATE INDEX my_special_pois_geom_gist_nd
ON my_special_pois USING gist(geom gist_geometry_ops_nd);

-- &#xc774; &#xc0c8; &#xd14c;&#xc774;&#xbel4;&#xc758; &#xb3c4;&#xd615; &#xc5f4;&#xc744; ←
geometry_columns &#xd14c;&#xc774;&#xbel4;&#xc5d0; &#xc9c1;&#xc811; ←
&#xb4f1;&#xb85d;&#xd558;&#xb294;
-- &#xb2e4;&#xc74c; &#xb29;&#xb95;&#xc740; PostGIS 2.0 &#xc774;&#xd6c4; &#xbc84;&#xc804; ←
&#xb0f; 1.4 &#xc774;&#xd6c4; &#xbc84;&#xc804; &#xbaa8;&#xb450;&#xc5d0;&#xc11c; ←
&#xb3d9;&#xc791;&#xd569;&#xb2c8;&#xb2e4;.
```

```
-- PostGIS 2.0 &#xabc84;&#xc804;&#xc758; &#xacbd;&#xc6b0; &#xc5f4;&#xc744; typmod ←
&#xae30;&#xabc18;&#xc73c;&#xb85c; &#xb9cc;&#xb4e4;&#xae30; &#xc704;&#xd574;
-- &#xd14c;&#xc774;&#xbe14;&#xc758; &#xae30;&#xc800; &#xad6c;&#xc870;&#xb97c; ←
&#xbcc0;&#xacbd;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
-- PostGIS 2.0 &#xc774;&#xc804; &#xabc84;&#xc804;&#xc758; &#xacbd;&#xc6b0;, ←
&#xb3d9;&#xc77c;&#xd55c; &#xabc29;&#xabc95;&#xc73c;&#xb85c; &#xbdf0;&#xb97c; ←
&#xb4f1;&#xb85d;&#xd560; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
SELECT populate_geometry_columns('myschema.my_special_pois'::regclass);

-- PostGIS 2.0 &#xabc84;&#xc804;&#xc744; &#xc0ac;&#xc6a9;&#xc911;&#xc774;&#xace0; ←
&#xc5b4;&#xb5a4; &#xc774;&#xc720;&#xc5d0;&#xc11c;&#xb4e0;
-- &#xad6c;&#xc2dd; &#xc81c;&#xc57d;&#xc870;&#xac74; &#xae30;&#xabc18; &#xc815;&#xc758; ←
&#xb3d9;&#xc791;&#xc774; &#xd544;&#xc694;&#xd55c; &#xacbd;&#xc6b0;
-- (&#xbaa8;&#xb4e0; &#xc790;&#xc2dd; &#xac1d;&#xccb4;&#xac00; &#xb3d9;&#xc77c;&#xd55c; ←
&#xc720;&#xd615;&#xacfc; SRID&#xac00; &#xc544;&#xb2cc; &#xc0c1;&#xc18d; ←
&#xd14c;&#xc774;&#xbe14;&#xc758; &#xacbd;&#xc6b0; &#xb4f1;)
-- &#xc120;&#xd0dd;&#xc801;&#xc778; &#xc0c8; use_typmod argument ←
&#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb97c; &#xac70;&#xc9d3;&#xc73c;&#xb85c; ←
&#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
SELECT populate_geometry_columns('myschema.my_special_pois'::regclass, false);
```

```
&#xad6c;&#xc2dd; &#xc81c;&#xc57d;&#xc870;&#xac74; &#xae30;&#xabc18; &#xabc29;&#xabc95;&#xc744; &#xc5ec;&#xc804;&#xc9c0;&#xc6d0;&#xd558;&#xae34; &#xd558;&#xc9c0;&#xb9cc; &#xbdf0;&#xc5d0;&#xc11c; &#xc9c1;&#xc811;&#xc801;&#xc0ac;&#xc6a9;&#xb418;&#xb294; &#xc81c;&#xc57d;&#xc870;&#xac74; &#xae30;&#xabc18; &#xb3c4;&#xd615; &#xc5f4;&#xae30;&#xabc18; &#xc5f4;&#xacfc;&#xb294; &#xb2ec;&#xb9ac; geometry_columns &#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc815;&#xd655;&#xd558;&#xac8c; &#xb4f1;&#xb85d;&#xb418;&#xc9c0; &#xc54a;&#xc744; &#xac81;&#xb2c8;&#xb2e4; &#xb2e4;&#xc74c;&#xc740; typmod&#xb97c; &#xc774;&#xc6a9;&#xd558;&#xb294; &#xc5f4;&#xacfc; &#xc81c;&#xc57d;&#xc870;&#xc774;&#xc6a9;&#xd558;&#xb294; &#xb610;&#xb2e4;&#xb978; &#xc5f4;&#xc744; &#xc815;&#xc758;&#xd558;&#xb294; &#xc608;&#xc2dc;&#xc785;&#xb2c8;&#xb2e4;.
```

```
CREATE TABLE pois_ny(gid SERIAL PRIMARY KEY
, poi_name text, cat varchar(20)
, geom geometry(POINT,4326) );
SELECT AddGeometryColumn('pois_ny', 'geom_2160', 2160, 'POINT', 2, false);
```

```
PSQL&#xc5d0;&#xc11c; &#xc2e4;&#xd589;&#xd560; &#xacbd;&#xc6b0;
```

```
\d pois_ny;
```

```
&#xb450; &#xc5f4;&#xc774; &#xc11c;&#xb85c; &#xb2e4;&#xb974;&#xac8c; &#xc815;&#xc758;&#xb418;&#xc5c8;&#xb2e4;&#xc0ac;&#xc2e4;&#xc744; &#xc54c; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xd558;&#xb098;&#xb294; typmod, &#xb2e4;&#xb978; &#xd558;&#xb098;&#xb294; &#xc81c;&#xc57d;&#xc870;&#xac74;&#xc73c;&#xb85c; &#xc815;&#xc758;&#xb458; &#xb2e4; geometry_columns &#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xc815;&#xd655;&#xd558;&#xac8c; &#xb4f1;&#xc9c0; &#xc54a;&#xc744; &#xac81;&#xb2c8;&#xb2e4; &#xb2e4;&#xc74c;&#xc740; typmod&#xb97c; &#xc774;&#xc6a9;&#xd558;&#xb294; &#xc5f4;&#xacfc; &#xc81c;&#xc57d;&#xc870;&#xc774;&#xc6a9;&#xd558;&#xb294; &#xb610;&#xb2e4;&#xb978; &#xc5f4;&#xc744; &#xc815;&#xc758;&#xd558;&#xb294; &#xc608;&#xc2dc;&#xc785;&#xb2c8;&#xb2e4;.
```

```
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)
```

```
&#xb458; &#xb2e4; geometry_columns &#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xc815;&#xd655;&#xd558;&#xac8c; &#xb4f1;&#xc9c0; &#xc54a;&#xc744; &#xac81;&#xb2c8;&#xb2e4; &#xb2e4;&#xc74c;&#xc740; typmod&#xb97c; &#xc774;&#xc6a9;&#xd558;&#xb294; &#xc5f4;&#xacfc; &#xc81c;&#xc57d;&#xc870;&#xc774;&#xc6a9;&#xd558;&#xb294; &#xb610;&#xb2e4;&#xb978; &#xc5f4;&#xc744; &#xc815;&#xc758;&#xd558;&#xb294; &#xc608;&#xc2dc;&#xc785;&#xb2c8;&#xb2e4;.
```

```
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

```
-- --
-- --
```

```
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';
```

```
typmod
-- --
-- --
```

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

```
PostGIS
-- --
-- --
```

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

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

4.7 GIS

```
-- --
-- --
```

SQL 구문을 사용하거나 shapefile 로더/덤&#xd37c
사용하는 두 가지 방법으로 PostGIS/Post-
greSQL 데이터베이스에 데이터를 입&#xb
수 있습니다.

4.7.1 SQL을 이용해 데이터 가져오

사용자 데이터를 텍스트 표현식&#
변환할 수 있다면,. PostGIS에 사용자 데&#
입력하는 가장 쉬운 방법은 형식&#
SQL을 이용하는 것입니다,. Oracle이나 다,
SQL 데이터베이스와 마బ가지로,. SQL
터미널 모니터에 SQL "INSERT" 선언문으&#xb85
가득 బ대용량 텍스트 파일을 송&#
방법으로 데이터를 일괄 로드시&#
수 있습니다.

데이터 업로드 파일(예를 들어 roads .
sql)은 다음ಘ럼 보일 것입니다:

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

"psql" SQL 터미널 모니터를 이용해서 Post-
greSQL로 데이터 파일을 매우 쉽게 송&#xb
수 있습니다.

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

4.7.2 shp2pgsql: ESRI shapefile 로더 이용하기

shp2pgsql 데이터 로더는 ESRI shapefile을,. 도형 형&#
지리형 형식이든,. PostGIS/PostgreSQL 데이터베&
삽입하기에 적합한 SQL로 변환합&#xb2c
이 로더에는 명령행(command line) 플래그&#xb85
구별되는 몇 가지 실행 모드가 존&#
அ령행 로더 외에,. 사용자가 Post-

GIS를 ಘ음 접하는 경우 스크립트를
사용하지 않고 단 한 번 로드하는
데 더 쉽게 사용할 수 있을 뿐만 아&#
명령행 로더가 가진 대부분의 옵&#
가지고 있는 shp2pgsql-gui 그래픽 인터페&#
있습니다,. shp2pgsql-gui를 pgAdmin III의 플러그인&
설정할 수도 있습니다.

- S 다중(multi) 도형 대신 단순 도형을 생이 옵션은 모든 도형이 실제로 단일형(예: 단일 외곽선을 가진 다중 폴리곤 또는 단일 꼭짓점을 가진 다중 포인트)일 경우에만 작동합니다.
- t <dimensionality> 산출 도형이 지정된 차원수󊰀지도록 강제합니다. 차원수맀시하는 데 다음 스트링을 사오 2D, 3DZ, 3DM, 4D
- 입력물이 지정된 차원수보다 낮은 차원일 경우 출력물의 해니 차원은 0으로 채워질 것입니다. 입력물이 지정된 차원수보다 높은 차원일 경우 필요 없는 차제거될 것입니다.
- w WKB 대신 WKT 형식으로 출력합니다. 정확도가 부족하기 때문에 좌퐀 이동될 가능성이 있다는 점을 주의하십시오.
- e 각 선언문을 상호처리를 이용하않고 자체적으로 실행합니다. 오류를 생성하는 몇몇 망가진 도형이 있을 경우 이 옵션을 사오 괜찮은 데이터 대다수를 로드합 수 있습니다. " 덤프" 형식은 항상 상호처리를 이용하기 때문에 -D 플래그와 함૨ 사용할 수 없다점을 주의하십시오.
- W <encoding> 입력 데이터(DBF 파일)의 인코딩 지정합니다. 이 옵션을 사용하모든 속성을 지정된 인코딩의 로 변환합니다. 그 결과로 생성을 SQL 출력물은 SET CLIENT_ENCODING t o UTF8 명령어를 담게 되어 ଱엔드에서 UTF8을 데이내부에서 이용하도록 설정된 어떤 인코딩으로든 재변환할 수 있습니다.
- N <policy> NULL 도형 처리 ଩침 -- insert*(상관없이 삽입), skip(건너뛰기), abort(중단)
- n DBF 파일만 임포트합니다. 사용자 데이터에 대응하는 shapefile이 없다랐동적으로 이 모드로 전환하어 DBF만 로드할 것입니다. 따라서 전을 shapefile집합을 가지고 있지만 도형을 ୼고 속성 데이터만 필요한 경우이 플래그를 설정해야 합니다.
- G (경도/위도가 필요한) 도형 대신 WGS84 경위도(SRID=4326)를 쓰는 지리형을 이용합니다.

-T <tablespace> 새 테 이 블 을 위 한 테 이 블 스 &#xd9c0; 정 합 니 다. -X 파 라 미 터 가 함 께 쓰 인 경 우 가 아 니 라 면 여 전 히 인 덱 스 가 기 본 테 이 블 스 페 이 스 이 용 할 것 입 니 다. PostgreSQL 문 서 는 사 지 정 테 이 블 스 페 이 스 가 필 요 핕 경 우 를 잘 설 명 하 고 있 습 니 다.

-X <tablespace> 새 테 이 블 의 인 덱 스 를 위 한 테 이 블 스 페 이 스 를 지 정 합 니 닩 이 옵 션 은 기 본 키 (primary key) 인 덱 스 에 적 용 되 며 , -I 플 래 그 를 함 께 사 용 용 󊲽 우 GiST 공 간 인 덱 스 에 도 적 용 됩 &

-c When used, this flag will prevent the generation of `ANALYZE` statements. Without the `-Z` flag (default behavior), the `ANALYZE` statements will be generated.

ˤ음은 로더를 이용해서 입력 파생성하고 업로드하는 세션의 예&#x

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

UNIX 파이프(pipe)를 이용하면 모든 변환
및 업로드 작업을 한 번에 끝낼 수
있습니다:

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

4.8 &#xacf5:&#xac04:&#xd14c:&#xc774:&#xbe14:&#xc0dd:&#xc131:

SQL이나 shapefile 로더/덤/퍼를 이용해서
데이터베이스로부터 데이터를
추출할 수 있습니다. SQL 단원에서
공간테이블에 대한 비교 및 쿼리&#xb
할 수 있는 몇몇 연산자에 대해 논&
것입니다.

4.8.1 SQL Injection

```
&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xb85c;&#xbd80;&#xd130; &#xb370;&#xc774;&#xd130;&#xb97c;
&#xcd94;&#xcd9c;&#xd558;&#xb294;&#xac00;&#xc7a5; &#xac04;&#xb2e8;&#xd55c; &#xbc29;&#xbc95;&#xc740; SQL
&#xc120;&#xbcc4;(select) &#xcffc;&#xb9ac;&#xb85c; &#xbc18;&#xd658;&#xb420; &#xb808;&#xcff5;&#xb4dc; &#xbc0f;
&#xc5f4;&#xc758; &#xac1c;&#xc218;&#xb97c; &#xc904;&#xc778; &#xb2e4;&#xc74c; &#xd574;&#xb2f9; &#xacb0;&#xacfc;
&#xc5f4;&#xc744; &#xd30c;&#xc2f1; &#xac00;&#xb2a5;&#xd55c; &#xd14d;&#xc2a4;&#xd2b8; &#xd30c;&#xc77c;&#xb85c;
&#xb364;&#xd504;&#xbcb1b;&#xb294; &#xac83;&#xc785;&#xb2c8;&#xb2e4::
```

```
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)

```

하지만, 반환되는 필드의 개수를
 줄이기 위해 어떤 종류의 제약이
 필요할 때가 있을 ಃ입니다. 속성
 기반 제약의 경우 일반적인 비공&#
 테이블의 경우와 동일한 SQL 문법&#xc74
 쓰면 됩니다. 공간 제약의 경우, 다&#
 유용한 연산자들을 쓸 수 있습니&#xc

ST_Intersects This function tells whether two geometries share any space.

```

= &#xc774; &#xc5f0;&#xc0b0;&#xc790;&#xb294; &#xb450; &#xb3c4;&#xd615;&#xc774; &#xae30;&#xd558;&#xd559;&#xc801;
  &#xb3d9;&#xc77c;&#xd55c;&#xc9c0;&#xb97c; &#xd14c;&#xc2a4;&#xd2b8;&#xd569;&#xb2c8;&#xb2e4;. &#xc608;&#xb97
  &#xb4e4;&#xc5b4;, 'POLYGON((0 0,1 1,1 0,0 0))' &#xacfc; 'POLYGON((0 0,1 1,1 0,0 0))' &#xc774; &#xb3d9;&#xc77c;&#xd5
  &#xb9d0;&#xc785;&#xb2c8;&#xb2e4;(&#xb3d9;&#xc77c;&#xd569;&#xb2c8;&#xb2e4;).

```

다음으로, 이 연산자들을 쿼리에
 쓸 수 있습니다. SQL 명령행에 도형&#xac
 경계 상자를 지정할 때, "ST_GeomFromText()" 함&#xc218
 이용해서 스트링 표현식을 도형&#
 정확하게 변환시జ야 합니다. 해&#
 데이터와 일치하는 가공의 공간
 참조 시스템은 312입니다. 다음은
 그 예시입니다;

```

SELECT road_id, road_name
FROM roads
WHERE ST_OrderingEquals(roads_geom , ST_GeomFromText('LINESTRING(191232 243118,191108 ↵
  243242)',312) ) ;

```

이 쿼리는 해당 값과 동일한 도형&#
 담ૠ 있는 "ROADS_GEOM" 테이블로부터 단&#
 레코드를 반환할 ಃ입니다.

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

```

SELECT road_id, road_name
FROM roads
WHERE roads_geom && ST_GeomFromText('POLYGON(...)',312);

```

가장 흔한 공간 쿼리는 아마도 데&#
 브라우저 또는 웹 매퍼 같은 클라&#
 소프트웨어가 화면 표출을 위해
 "맵 프레임(map frame)" 용량에 해당하는
 데이터를 가져오기 위해 사용하&#
 "프레임 기반(frame-based)" 쿼리일 ಃ입니&#xb2e4
 "&#" 연산자 사용시, 비교 피처로 BOX3D
 또는 도형을 지정할 수 있습니다. &
 하지만 도형을 지정했을 경우 비&#
 작업에 해당 경계 상자가 사용될
 ಃ입니다.

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);

```

화면೐해당하는데이터의투영지정하는데SRID 312를썼다는사실에주의하십시오.

4.8.2 $\mathbb{R}^n$ and $\mathbb{C}^n$

```
pgsql2shp &#xd14c;&#xc774;&#xbe14; &#xb364;&#xd37c;&#xb294; &#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;
&#xc9c1;&#xc811; &#xc5f0;&#xacb0;&#xb418;&#xc5b4; (&#xc544;&#xb9c8;&#xb3c4; &#xcffc;&#xb9ac;&#xac00; &#xc815;&#xc774;
&#xd14c;&#xc774;&#xbe14;&#xc744; shapefile&#xb85c; &#xbcc0;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. &#xae30;&#xbcf8;
&#xbb38;&#xbc95;&#xc740; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc2b5;&#xb2c8;&#xb2e4;:
```

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

```
psql2shp [<options>] <database> <query>
```

다음과 같은 명령행 옵션이 있습&#x

```
-f <filename> &#xd2b9;&#xc815; &#xd30c;&#xc77c;&#xba85;&#xc73c;&#xb85c; &#xcd9c;&#xb825;&#xbb3c;&#xc744; &#xc79
```

```
-h <host> &#xc5f0;&#xacb0;&#xd560; &#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4; &#xd638;&#xc2a4;&#xd2b8;&#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.
```

-p <port> 데이터베이스 호스트 연결시
사용할 포트를 설정합니다.

-P <password> 데이터베이스 연결에 사용합니다.

-u <user> %b370;%c774;%d130;%bca0;%c774;%c2a4; %c5f0;%xabc0;%c5d0; %c0ac;%c6a9;%d560;
 %c0ac;%c6a9;%c790;%xba85;%c744; %c124;%c815;%d569;%b2c8;%b2e4;.

```
-g <geometry column> &#xc9f5;&#xc218;&#xc758;&#xb3c4;&#xd615;&#xc5f4;&#xc744;&#xac00;&#xc9c4;&#xd14c;&#xc774
&#xacbd;&#xc6b0;,&#xc791;&#xc131;&#xc5d0;&#xc774;&#xc6a9;&#xb420;&#xb3c4;&#xd615;&#xc5f4;&#xc7
&#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.
```

-b &#x14; ݴ 너 리 ໤ 서 를 사 용 하 도 록 설 이 옵 션 을 쓰 면 실 행 속 도 가 빨 &#xb7 테 이 블 안 에 있 는 비 (非) 도 형 속 &#x 가 운 데 하 나 라 도 텍 스 트 로 작 &#xc1 캐 스 트 (cast) 가 부 족 할 경 우 실 행 되 않 을 것 입 니 다.

```
-r &#x85c;(raw) &#xbaa8;&#xb4dc;&#xc785;&#xb2c8;&#xb2e4;. gid &#xd544;&#xb4dc;&#xb97c; &#xc0ad;&#xc81c;&#xd558
&#xc5f4; &#xba85;&#xce6d;&#xc744; &#xc81c;&#xc678;&#xd558;&#xc9c0; &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
-m filename &#xc2dd;&#xbcc4;&#xc790;&#xb97c; 10&#xbb38;&#xc790; &#xba85;&#xce6d;&#xc73c;&#xb85c; &#xb2e4;&#xb9e4;&#xd551;(remap)&#xd569;&#xb2c8;&#xb2e4;. &#xd574;&#xb2f9; &#xd30c;&#xc77c;&#xc758; &#xb0b4;&#xc6a9;&#xacf5;&#xbc31;&#xc73c;&#xb85c; &#xad6c;&#xbd84;&#xb41c; &#xb450; &#xc2ec;&#xbcf5;&#xb85c; &#xc774;&#xb8e5;&#xbcf5;&#xc218;&#xc758; &#xd589;&#xc73c;&#xb85c;, &#xd589; &#xb9e8; &#xc55e;&#xacfc; &#xb9e8; &#xb4a4;&#xc55e;&#xacf5;&#xbc31;&#xc774; &#xc5c6;&#xc5b4;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;. VERYLONGSYMBOL SHORT-ONE ANOTHERVERYLONGSYMBOL SHORTER &#xb4f1;&#xacfc; &#xac19;&#xc740; &#xc608;&#xac00; &#xc788;&#xc740;
```

4.9

인덱스 덕분에 공간 데이터베이대용량 데이터셋을 사용할 수 있인덱스 작업을 하지 않으면. 어떤

PostgreSQL and PostGIS support several index methods for spatial data: B-tree, R-tree, GiST, and BRIN. The B-tree index method commonly used for attribute data is not very useful for spatial data, since it only supports storing and querying data in a single dimension. Data such as geometry (which has 2 or more dimensions) requires an index method that supports range query across all the data dimensions. One of the key advantages of PostgreSQL for spatial data handling is that it offers several kinds of index methods which work well for multi-dimensional data: GiST, BRIN and SP-GiST indexes.

The B-tree index method commonly used for attribute data is not very useful for spatial data, since it only supports storing and querying data in a single dimension. Data such as geometry (which has 2 or more dimensions) requires an index method that supports range query across all the data dimensions. One of the key advantages of PostgreSQL for spatial data handling is that it offers several kinds of index methods which work well for multi-dimensional data: GiST, BRIN and SP-GiST indexes.

- **GiST (Generalized Search Tree)** indexes operate by summarizing the spatial extent of ranges of table records. Search is done via a scan of the ranges. BRIN is only appropriate for use for some kinds of data (spatially sorted, with infrequent or no update). But it provides much faster index create time, and much smaller index size.
- **BRIN (Block Range Index)** indexes operate by summarizing the spatial extent of ranges of table records. Search is done via a scan of the ranges. BRIN is only appropriate for use for some kinds of data (spatially sorted, with infrequent or no update). But it provides much faster index create time, and much smaller index size.
- **SP-GiST (Space-Partitioned Generalized Search Tree)** is a generic index method that supports partitioned search trees such as quad-trees, k-d trees, and radix trees (tries).

Spatial indexes store only the bounding box of geometries. Spatial queries use the index as a **primary filter** to quickly determine a set of geometries potentially matching the query condition. Most spatial queries require a **secondary filter** that uses a spatial predicate function to test a more specific spatial condition. For more information on querying with spatial predicates see Section 5.2.

See also the [PostGIS Workshop section on spatial indexes](#), and the [PostgreSQL manual](#).

4.9.1 GiST

GiST indexes operate by summarizing the spatial extent of ranges of table records. Search is done via a scan of the ranges. BRIN is only appropriate for use for some kinds of data (spatially sorted, with infrequent or no update). But it provides much faster index create time, and much smaller index size.

SP-GiST is a generic index method that supports partitioned search trees such as quad-trees, k-d trees, and radix trees (tries).

For more information on querying with spatial predicates see Section 5.2.

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

이 문법은 항상 2D 인덱스를 빌드&#xd560
것입니다. 해당 도형 유형에 PostGIS 2.0
이상 버전이 지원하는 n차원 인덱&
얻으려면. 다음 문법으로 생성할
수 있습니다:

```
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 [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)];
```

4.9.2 GiST 인덱스

BRIN stands for "Block Range Index". It is a general-purpose index method introduced in PostgreSQL 9.5. BRIN is a *lossy* index method, meaning that a secondary check is required to confirm that a record matches a given search condition (which is the case for all provided spatial indexes). It provides much faster index creation and much smaller index size, with reasonable read performance. Its primary purpose is to support indexing very large tables on columns which have a correlation with their physical location within the table. In addition to spatial indexing, BRIN can speed up searches on various kinds of attribute data structures (integer, arrays etc). For more information see the [PostgreSQL manual](#).

GIS 데이터 테이블이 수천 행을 넘&#xac8
되면. 데이터 공간 검색의 속도를
향상시키기 위해 인덱스를 빌드&#
싶게 될 것입니다(사용자의 모든
검색이 속성에 기반하는 경우가
아니라면 말입니다. 그런 경우. 속&#
필드에 대해 일반 인덱스를 빌드&#
됩니다).

A BRIN index stores the bounding box enclosing all the geometries contained in the rows in a contiguous set of table blocks, called a *block range*. When executing a query using the index the block ranges are scanned to find the ones that intersect the query extent. This is efficient only if the data is physically ordered so that the bounding boxes for block ranges have minimal overlap (and ideally are mutually exclusive). The resulting index is very small in size, but is typically less performant for read than a GiST index over the same data.

Building a BRIN index is much less CPU-intensive than building a GiST index. It's common to find that a BRIN index is ten times faster to build than a GiST index over the same data. And because a BRIN index stores only one bounding box for each range of table blocks, it's common to use up to a thousand times less disk space than a GiST index.

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 provide better performance.

For BRIN to be effective, the table data should be stored in a physical order which minimizes the amount of block extent overlap. It may be that the data is already sorted appropriately (for instance, if it is loaded from another dataset that is already sorted in spatial order). Otherwise, this can be accomplished by sorting the data by a one-dimensional spatial key. One way to do this is to create a new table sorted by the geometry values (which in recent PostGIS versions uses an efficient Hilbert curve ordering):

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

Alternatively, data can be sorted in-place by using a GeoHash as a (temporary) index, and clustering on that index:

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

```
"&#xb3c4;&#xd615;" &#xc5f4;&#xc5d0; &#xb300;&#xd574; GiST &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xbe4c;&#xb4dc;&#xc774;&#xb370; &#xd544;&#xc694;&#xd55c; &#xbb38;&#xbc95;&#xc740; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc2b5;&#xb2c8;&#xc774; &#xbb38;&#xbc95;&#xc740; &#xd56d;&#xc0c1; 2D &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xbe4c;&#xb4dc;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xd574;&#xb2f9; &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc5d0; PostGIS 2.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc774; &#xc9c0;&#xc6d0;&#xd558;&#xb294; n&#xcc28;&#xc6d0; &#xc778;&#xb371;&#xc5bb;&#xc73c;&#xb824;&#xba74;. &#xb2e4;&#xc74c; &#xbb38;&#xbc95;&#xc73c;&#xb85c; &#xc0dd;&#xc131;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

```
&#xc774; &#xbb38;&#xbc95;&#xc740; &#xd56d;&#xc0c1; 2D &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xbe4c;&#xb4dc;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xd574;&#xb2f9; &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc5d0; PostGIS 2.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc774; &#xc9c0;&#xc6d0;&#xd558;&#xb294; n&#xcc28;&#xc6d0; &#xc778;&#xb371;&#xc5bb;&#xc73c;&#xb824;&#xba74;. &#xb2e4;&#xc74c; &#xbb38;&#xbc95;&#xc73c;&#xb85c; &#xc0dd;&#xc131;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

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

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

The above commands use the default number of blocks in a range, which is 128. To specify the number of blocks to summarise in a range, use this syntax

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

Keep in mind that a BRIN index only stores one index entry 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 performance penalty by choosing the operator class with the least number of dimensions of the stored geometries

```
"&#xb3c4;&#xd615;" &#xc5f4;&#xc5d0; &#xb300;&#xd574; GiST &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xbe4c;&#xb4dc;&#xc774;&#xb370; &#xd544;&#xc694;&#xd55c; &#xbb38;&#xbc95;&#xc740; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc2b5;&#xb2c8;&#xc774; &#xbb38;&#xbc95;&#xc740; &#xd56d;&#xc0c1; 2D &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xbe4c;&#xb4dc;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xd574;&#xb2f9; &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc5d0; PostGIS 2.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc774; &#xc9c0;&#xc6d0;&#xd558;&#xb294; n&#xcc28;&#xc6d0; &#xc778;&#xb371;&#xc5bb;&#xc73c;&#xb824;&#xba74;. &#xb2e4;&#xc74c; &#xbb38;&#xbc95;&#xc73c;&#xb85c; &#xc0dd;&#xc131;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

```
&#xc774; &#xbb38;&#xbc95;&#xc740; &#xd56d;&#xc0c1; 2D &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xbe4c;&#xb4dc;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xd574;&#xb2f9; &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc5d0; PostGIS 2.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc774; &#xc9c0;&#xc6d0;&#xd558;&#xb294; n&#xcc28;&#xc6d0; &#xc778;&#xb371;&#xc5bb;&#xc73c;&#xb824;&#xba74;. &#xb2e4;&#xc74c; &#xbb38;&#xbc95;&#xc73c;&#xb85c; &#xc0dd;&#xc131;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

An important difference between BRIN and other index types is that the database does not maintain the index dynamically. Changes to spatial data in the table are simply appended to the end of the index. This will cause index search performance to degrade over time. The index can be updated by performing a VACUUM, or by using a special function `brin_summarize_new_values`. For this reason BRIN may be most appropriate for use with data that is read-only, or only rarely changing. For more information refer to the [manual](#).

To summarize using BRIN for spatial data:

- Index build time is very fast, and index size is very small.
- Index query time is slower than GiST, but can still be very acceptable.
- Requires table data to be sorted in a spatial ordering.
- Requires manual index maintenance.
- Most appropriate for very large tables, with low or no overlap (e.g. points), which are static or change infrequently.
- More effective for queries which return relatively large numbers of data records.

4.9.3 GiST

SP-GiST stands for "Space-Partitioned Generalized Search Tree" and is a generic form of indexing for multi-dimensional data types 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 spatial indexing, SP-GiST is used to speed up searches on many kinds of data, such as phone routing, ip routing, substring search, etc. For more information see the [PostgreSQL manual](#).

As it is the case for GiST indexes, SP-GiST indexes are lossy, in the sense that they store the bounding box enclosing spatial objects. SP-GiST indexes can be considered as an alternative to GiST indexes.

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

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

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

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

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 a slower CONCURRENTLY-aware way:

```
CREATE INDEX [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)];
```

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.

4.9.4

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



```
&#xbabb; &#xd558;&#xae30; &#xb54c;&#xbb38;&#xc5d0;, &#xc885;&#xc885; &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xd65c;&#xc6a9;&#xd574;&#xc57c; &#xd560; &#xc80;&#xc0c9;&#xc774; &#xb300;&#xc2e0; &#xae30;&#xbcf8;&#xac12;&#xc804;&#xccb4; &#xb370;&#xc774;&#xd130;&#xc758; &#xc21c;&#xcc28; &#xc2a4;&#xce94;&#xc744; &#xc774;&#xc6a9;&#xb54c;&#xac00; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
&#xc0ac;&#xc6a9;&#xc790;&#xc758; &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xac00; (&#xb610;&#xb294; &#xc0ac;&#xc18d;&#xc131; &#xc778;&#xb371;&#xc2a4;&#xac00;) &#xd65c;&#xc6a9;&#xb418;&#xc9c0; &#xc54a;&#xace0; &#xc788;&#xc0ac;&#xc2e4;&#xc744; &#xc54c;&#xc8c; &#xb418;&#xc5c8;&#xb2e4;&#xba74;, &#xba87; &#xc0ac;&#xc9c0; &#xd574;&#xb294;&#xb2e4;.
```

- Examine the query plan and check your query actually computes the thing you need. An erroneous JOIN, either forgotten or to the wrong table, can unexpectedly retrieve table records multiple times. To get the query plan, execute with `EXPLAIN` in front of the query.
- 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 run `VACUUM` as an off-peak cron job on a regular basis.

- `두 번째, 빈공간 분석으로 해결이 안 될 경우 SET ENABLE_SEQSCAN=OFF 명령어를 통사క제로 쿼리 설계자사 인덱스 정이용하도록 할 수 있습니다. 이 명령어는 공간 인덱스 쿼리일 경한해 드물ಌ 이용해야 합니다. 일 쿼리 설계자는 언제 일렘 B-Tree 인덱사활용해야 하는지 사용자보다 더 잘 알고 있습니다. 사용자 쿼리 실후, 다른 쿼리사 평소처럼 쿼리 설활용하도록 ENABLE_SEQSCAN 을 다시 켤지 고 합니다.`
- `쿼리 설계자사 순차 볰 인덱스 스경중(cost)을 잘못 판단하고 있다면, postgresql.conf 파일의 random_page_cost의 ఒ을 줄여보일"SET random_page_cost=#" 로 써보십시오. 해당 파라기본ఒ은 4이지만, 1또는 2로 설정해ఒ을 ఐ소시킬수록 점점 더 쿼리 설계자사 인덱스 스캔을 활용하될 ಃ입니다.`
- If `SET ENABLE_SEQSCAN TO OFF`; does not help your query, the query may be using a SQL construct that the Postgres planner is not yet able to optimize. It may be possible to rewrite the query in a way that the planner is able to handle. For example, a subquery with an inline SELECT may not produce an efficient plan, but could possibly be rewritten using a LATERAL JOIN.

For more information see the Postgres manual section on [Query Planning](#).

Chapter 5

Spatial Queries

The *raison d'être* of spatial databases is to perform queries inside the database which would ordinarily require desktop GIS functionality. Using PostGIS effectively requires knowing what spatial functions are available, how to use them in queries, and ensuring that appropriate indexes are in place to provide good performance.

5.1 Determining Spatial Relationships

Spatial relationships indicate how two geometries interact with one another. They are a fundamental capability for querying geometry.

5.1.1 Dimensionally Extended 9-Intersection Model

According to the [OpenGIS Simple Features Implementation Specification for SQL](#), "the basic approach to comparing two geometries is to make pair-wise tests of the intersections between the Interiors, Boundaries and Exteriors of the two geometries and to classify the relationship between the two geometries based on the entries in the resulting 'intersection' matrix."

In the theory of point-set topology, the points in a geometry embedded in 2-dimensional space are categorized into three sets:

Boundary

The boundary of a geometry is the set of geometries of the next lower dimension. For POINTs, which have a dimension of 0, the boundary is the empty set. The boundary of a LINESTRING is the two endpoints. For POLYGONS, the boundary is the linework of the exterior and interior rings.

Interior

The interior of a geometry are those points of a geometry that are not in the boundary. For POINTs, the interior is the point itself. The interior of a LINESTRING is the set of points between the endpoints. For POLYGONS, the interior is the areal surface inside the polygon.

Exterior

The exterior of a geometry is the rest of the space in which the geometry is embedded; in other words, all points not in the interior or on the boundary of the geometry. It is a 2-dimensional non-closed surface.

The [Dimensionally Extended 9-Intersection Model](#) (DE-9IM) describes the spatial relationship between two geometries by specifying the dimensions of the 9 intersections between the above sets for each geometry. The intersection dimensions can be formally represented in a 3x3 **intersection matrix**.

For a geometry g the *Interior*, *Boundary*, and *Exterior* are denoted using the notation $I(g)$, $B(g)$, and $E(g)$. Also, $dim(s)$ denotes the dimension of a set s with the domain of $\{0, 1, 2, F\}$:

- 0 => point

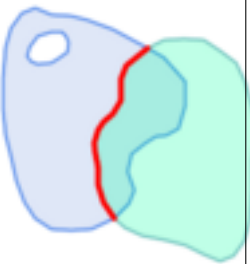
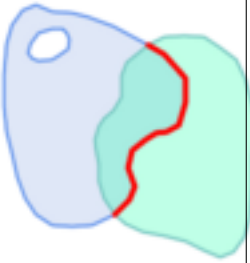
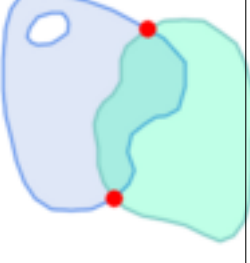
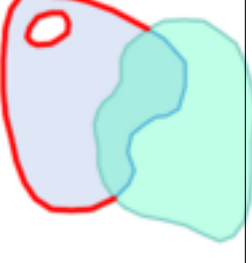
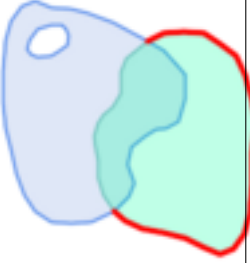
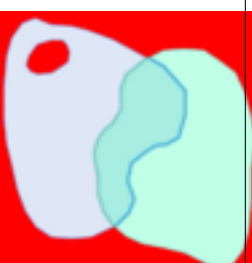
- 1 => line
- 2 => area
- F => empty set

Using this notation, the intersection matrix for two geometries *a* and *b* is:

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

Visually, for two overlapping polygonal geometries, this looks like:



	Interior	Boundary	Exterior
	 $\dim(I(a) \cap I(b)) = 2$	 $\dim(I(a) \cap B(b)) = 1$	 $\dim(I(a) \cap E(b)) = 2$
	 $\dim(B(a) \cap I(b)) = 1$	 $\dim(B(a) \cap B(b)) = 0$	 $\dim(B(a) \cap E(b)) = 1$
	Exterior		
	 $\dim(E(a) \cap I(b)) = 2$	 $\dim(E(a) \cap B(b)) = 1$	 $\dim(E(a) \cap E(b)) = 2$

Reading from left to right and top to bottom, the intersection matrix is represented as the text string '212101212'.

For more information, refer to:

- [OpenGIS Simple Features Implementation Specification for SQL](#) (version 1.1, section 2.1.13.2)
- [Wikipedia: Dimensionally Extended Nine-Intersection Model \(DE-9IM\)](#)
- [GeoTools: Point Set Theory and the DE-9IM Matrix](#)

5.1.2 Named Spatial Relationships

To make it easy to determine common spatial relationships, the OGC SFS defines a set of *named spatial relationship predicates*. PostGIS provides these as the functions `ST_Contains`, `ST_Crosses`, `ST_Disjoint`, `ST_Equals`, `ST_Intersects`, `ST_Overlaps`, `ST_Touches`, `ST_Within`. It also defines the non-standard relationship predicates `ST_Covers`, `ST_CoveredBy`, and `ST_ContainsProperly`.

Spatial predicates are usually used as conditions in SQL `WHERE` or `JOIN` clauses. The named spatial predicates automatically use a spatial index if one is available, so there is no need to use the bounding box operator `&&` as well. For example:

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

For more details and illustrations, see the [PostGIS Workshop](#).

5.1.3 General Spatial Relationships

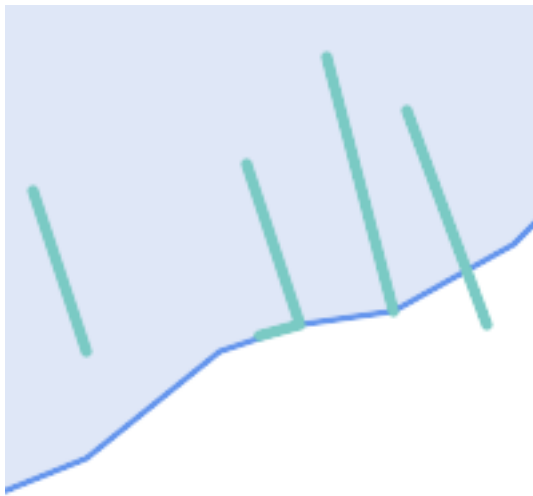
In some cases the named spatial relationships are insufficient to provide a desired spatial filter condition.



For example, consider a linear dataset representing a road network. It may be required to identify all road segments that cross each other, not at a point, but in a line (perhaps to validate some business rule). In this case `ST_Crosses` does not provide the necessary spatial filter, since for linear features it returns `true` only where they cross at a point.

A two-step solution would be to first compute the actual intersection (`ST_Intersection`) of pairs of road lines that spatially intersect (`ST_Intersects`), and then check if the intersection's `ST_GeometryType` is 'LINESTRING' (properly dealing with cases that return `GEOMETRYCOLLECTIONS` of `[MULTI] POINTs`, `[MULTI] LINESTRINGs`, etc.).

Clearly, a simpler and faster solution is desirable.



A second example is locating wharves that intersect a lake's boundary on a line and where one end of the wharf is up on shore. In other words, where a wharf is within but not completely contained by a lake, intersects the boundary of a lake on a line, and where exactly one of the wharf's endpoints is within or on the boundary of the lake. It is possible to use a combination of spatial predicates to find the required features:

- `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`
- ... but needless to say, this is quite complicated.

These requirements can be met by computing the full DE-9IM intersection matrix. PostGIS provides the `ST_Relate` function to do this:

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

To test a particular spatial relationship, an **intersection matrix pattern** is used. This is the matrix representation augmented with the additional symbols {T, * }:

- T => intersection dimension is non-empty; i.e. is in {0, 1, 2}
- * => don't care

Using intersection matrix patterns, specific spatial relationships can be evaluated in a more succinct way. The `ST_Relate` and the `ST_RelateMatch` functions can be used to test intersection matrix patterns. For the first example above, the intersection matrix pattern specifying two lines intersecting in a line is `'1*1***1**'`:

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

For the second example, the intersection matrix pattern specifying a line partly inside and partly outside a polygon is **'102101FF2'**:

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

5.2 Using Spatial Indexes

When constructing queries using spatial conditions, for best performance it is important to ensure that a spatial index is used, if one exists (see Section 4.9). To do this, a spatial operator or index-aware function must be used in a `WHERE` or `ON` clause of the query.

Spatial operators include the bounding box operators (of which the most commonly used is `&&`; see Section 8.10.1 for the full list) and the distance operators used in nearest-neighbor queries (the most common being `<->`; see Section 8.10.2 for the full list.)

Index-aware functions automatically add a bounding box operator to the spatial condition. Index-aware functions include the named spatial relationship predicates `ST_Contains`, `ST_ContainsProperly`, `ST_CoveredBy`, `ST_Covers`, `ST_Crosses`, `ST_Intersects`, `ST_Overlaps`, `ST_Touches`, `ST_Within`, and `ST_3DIntersects`, and the distance predicates `ST_DWithin`, `ST_DFullyWithin`, `ST_3DDFullyWithin`, and `ST_3DDWithin`.)

Functions such as `ST_Distance` do *not* use indexes to optimize their operation. For example, the following query would be quite slow on a large table:

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

This query selects 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 the specified point, ie. one `ST_Distance()` calculation is computed for **every** row in the table.

The number of rows processed can be reduced substantially by using the index-aware function `ST_DWithin`:

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

This query selects the same geometries, but it does it in a more efficient way. This is enabled by `ST_DWithin()` using the `&&` operator internally on an expanded bounding box of the query geometry. If there is a spatial index on `geom`, the query planner will recognize that it can use the index to reduce the number of rows scanned before calculating the distance. The spatial index allows retrieving only records with geometries whose bounding boxes overlap the expanded extent and hence which *might* be within the required distance. The actual distance is then computed to confirm whether to include the record in the result set.

For more information and examples see the [PostGIS Workshop](#).

5.3 Examples of Spatial SQL

The examples in this section make use of a table of linear roads, and a table of polygonal municipality boundaries. The definition of the `bc_roads` table is:

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

The definition of the `bc_municipality` table is:

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

1. *What is the total length of all roads, expressed in kilometers?*

You can answer this question with a very simple piece of SQL:

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

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

2. *How large is the city of Prince George, in hectares?*

This query combines an attribute condition (on the municipality name) with a spatial calculation (of the polygon area):

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

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

3. *What is the largest municipality in the province, by area?*

This query uses a spatial measurement as an ordering value. There are several ways of approaching this problem, but the most efficient is below:

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

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

Note that in order to answer this query we have to calculate the area of every polygon. If we were doing this a lot it would make sense to add an area column to the table that could be indexed for performance. By ordering the results in a descending direction, and then using the PostgreSQL "LIMIT" command we can easily select just the largest value without using an aggregate function like `MAX()`.

4. *What is the length of roads fully contained within each municipality?*

This is an example of a "spatial join", which brings together data from two tables (with a join) using a spatial interaction ("contained") as the join condition (rather than the usual relational approach of joining on a common key):

```
SELECT
  m.name,
  sum(ST_Length(r.geom))/1000 as roads_km
FROM bc_roads AS r
JOIN bc_municipality AS m
```



```

    ON ST_Contains(m.geom, r.geom)
GROUP BY m.name
ORDER BY roads_km;

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

This query takes a while, because every road in the table is summarized into the final result (about 250K roads for the example table). For smaller datasets (several thousand records on several hundred) the response can be very fast.

5. *Create a new table with all the roads within the city of Prince George.*

This is an example of an "overlay", which takes in two tables and outputs a new table that consists of spatially clipped or cut resultants. Unlike the "spatial join" demonstrated above, this query creates new geometries. An overlay is like a turbo-charged spatial join, and is useful for more exact analysis work:

```

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

6. *What is the length in kilometers of "Douglas St" in Victoria?*

```

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

kilometers
-----
4.89151904172838
```

7. *What is the largest municipality polygon that has a hole?*

```

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

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

성능 향상 비–

6.1.1 $\mathbb{R}^n$ and $\mathbb{C}^n$

t테ݴ블ݘ행가운데몇행과만ݼ치하는경계상자를검색하기위한도형연산자를이용하는쿼리를전송해봅시다.이때쿼리최적화기는해당테이블이데이페이지3장과80행만가지고있다고판단합니다.이런저용량테이블인덱스보다순차스캔이훨씬빠이󊱰라고추정할ಃ입니다.따라서인덱스를무시하기로합니다.일반적인경우라면이추정이맞그러나이번예제의경우.연산자경계상자를비교하기위해디스모든도형을불러와야하기때문결국모든TOAST페이지도읽어야합만

사용자시스템에이಄그가있는왕인하려면, "EXPLAIN ANALYZE" PostgreSQL 명령어를실행하십시오. 자세한정쿄ఏ기세부사항을알ૠ싶다면, PostgreSQL 성능메일링리스트에있는다음스레읽어೼수있습니다: <http://archives.postgresql.org/pgsql-performance/2002/msg00030.php>

and newer thread on PostGIS <https://lists.osgeo.org/pipermail/postgis-devel/2017-June/026209.html>

6.1.2 해결방법

PostgreSQL 개발자들은쿼리추정이 TOAST를인식하도록만들어이문제를해하ૠ있습니다. 현시점에서는, 다두가지해결방법이있습니다:

첫번째해결책은쿼리설계자가인덱스를사용하도록강제하는방법입니다. 쿼리를전송하기전서಄로 "SET enable_seqscan TO off;"를전송하십시오. 이렇게하면기적으로쿼리설순차스ಔ을사용할수있는경우사용하지않도록강제합니다. 따평상시처럼 GiST 인덱스를이용할것입니다. 그러나이플래그를매연결할때마다설정해야하ૠ다상황에서쿼리설계자가잘못추하기때&#bb38;에, 쿼리가끝난후에 "SET enable_seqscan TO on;"을전송해야합니다.

두번째해결책은쿼리설계자가추정하는만큼순차스ಔ속도를높이는방법입니다. 경계상자를ಐ시"하는추가열을생성한다음, 이추가열을대상으로일엨여부쿼리하면됩니다. 이번예제의경해당명령어는다음과같습니다:

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

이제 geom_column 대신 bbox 에대해연산자를이용하도록다음과같이사용자쿼리를변경하십시오:

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

물론, mytable의열을변경하౰나추가경우, bbox의 "일관성"을유지해야합라관성을유지시키는가장명료방법은트리౰(trigger)일것입니다. 물사용자응용프로그램을수정해&#xbbox 열의정쿄를항상최신상태로유지하도록하౰나, 모든수정작후매번앞에서나온 UPDATE 쿼리를실수도있습니다.

6.2

PostgreSQL; CLUSTER <table> ON <index>;
 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.

PostgreSQL; PostGIS GiST; GiST; NULL; ALTER TABLE my_table ALTER COLUMN the_geom SET not null;

```
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.
```

HINT: "not null" ALTER TABLE my_table ALTER COLUMN the_geom SET not null;

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

NULL; ALTER TABLE blubb ADD CHECK (geometry is not null);

6.3

3D; 4D; ST_AsText(); ST_AsBinary();

```
ST_Force2D()
&#xd568;&#xc218;&#xb97c; &#xd638;&#xcd9c;&#xd558;&#xb294;&#xb370;, &#xb300;&#xc6a9;&#xb7c9;
&#xb3c4;&#xd615;&#xc758; &#xacbd;&#xc6b0; &#xc774; &#xd568;&#xc218;&#xb294; &#xc2dc;&#xc2a4;&#xd15c;&#xc758;
&#xc790;&#xc6d0;&#xc744; &#xc0c1;&#xb2f9;&#xd788; &#xc7a1;&#xc544;&#xba39;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc774;&#xc790;&#xc6d0; &#xb0ad;&#xbe44;&#xb97c; &#xd53c;&#xd558;&#xb824;&#xba74;, &#xbbf8;&#xb9ac; &#xadf8; &#xcd94;&#xc2dc;&#xc6d0;&#xc744; &#xc644;&#xc804;&#xd788; &#xc0ad;&#xc81c;&#xd558;&#xb294; &#xd3b8;&#xc774; &#xd6a8;&#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

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

```
AddGeometryColumn()
&#xd568;&#xc218;&#xb97c; &#xd1b5;&#xd574; &#xc0ac;&#xc6a9;&#xc790; &#xb3c4;&#xd615;
&#xc5f4;&#xc744; &#xcd94;&#xac00;&#xd588;&#xc744; &#xacbd;&#xc6b0; &#xb3c4;&#xd615; &#xc2dc;&#xc6d0;&#xc5d0;
&#xc81c;&#xc57d;&#xc870;&#xc74; &#xc774; &#xc78;&#xb824; &#xc788;&#xb2e4;&#xb294; &#xc0ac;&#xc2e4;&#xc744;
&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;. &#xc774;&#xb97c; &#xc6b0;&#xd68c;&#xd558;&#xb824;&#xba74;
&#xc81c;&#xc57d;&#xc870;&#xc74;&#xc744; &#xc0ad;&#xc81c;&#xd574;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;. ge-
ometry_columns &#xd14c;&#xc774;&#xbe14; &#xb0b4;&#xbd80; &#xd56d;&#xbaa9;&#xc744; &#xc5c5;&#xb370;&#xc774;&#xd14c;
&#xb2e4;&#xc74c; &#xc81c;&#xc57d;&#xc870;&#xc74;&#xc744; &#xb2e4;&#xc2dc; &#xc0dd;&#xc131;&#xd558;&#xb294;
&#xc78; &#xc78a;&#xc9c0; &#xb9c8;&#xc2ed;&#xc2dc;&#xc624;.
```

```
&#xb300;&#xc6a9;&#xb7c9; &#xd14c;&#xc774;&#xbe14;&#xc758; &#xacbd;&#xc6b0;, &#xc0ac;&#xc6a9;&#xc790;&#xc758;
&#xae30;&#xbcf8; &#xd0a4; &#xb610;&#xb294; &#xb610;&#xb2e4;&#xb978; &#xc0ac;&#xc6a9; &#xac00;&#xb2a5;&#xd55c;
&#xae30;&#xc900;&#xacfc; &#xd568;&#xaed8; WHERE &#xc808;&#xc744; &#xc774;&#xc6a9;&#xd574;&#xc11c; &#xd14c;&#xc77c;
&#xbd80;&#xbd84;&#xb9cc; &#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xd558;&#xb3c4;&#xb85d; &#xc81c;&#xd55c;&#xb2e4;&#xc74c;
&#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xb9c8;&#xb2e4; &#xac04;&#xb2e8;&#xd788; "VACUUM;" &#xc744; &#xc2e4;&#xd589;
&#xd558;&#xb294; &#xbc29;&#xc2dd;&#xc73c;&#xb85c; &#xc774; &#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xc791;&#xc740;
&#xbd80;&#xbd84;&#xc73c;&#xb85c; &#xb098;&#xb204;&#xb294; &#xd3b8;&#xc774; &#xc88b;&#xc744; &#xc218; &#xc788;
&#xc2b5;&#xb2c8;&#xb2e4;. &#xc774;&#xb807;&#xac8c; &#xd558;&#xba74; &#xd544;&#xc694;&#xd55c; &#xc784; &#xc2dc;
&#xb514;&#xc2a4;&#xd06c; &#acf5;&#xac04;&#xc774; &#xae09;&#xaca9;&#xd788; &#xc904;&#xc5b4;&#xb610;&#xd55c;
&#xba87; &#xc885;&#xb958;&#xc758; &#xc2dc;&#xc6d0;&#xc774; &#xc11e;&#xc778; &#xb3c4;&#xd615;&#xac00;
&#xc9c0;&#xace0; &#xc788;&#xc744; &#xacbd;&#xc6b0;, "WHERE dimension(the_geom)>2" &#xb85c; &#xc5c5;&#xb370;
&#xc81c;&#xd55c;&#xd558;&#xba74; &#xc774;&#xbbf8; 2D&#xc778; &#xb3c4;&#xd615;&#xc744; &#xb2e4;&#xc2dc;
&#xc791;&#xc131;&#xd558;&#xb294; &#xc77c;&#xc744; &#xc74;&#xb108;&#xb6f8; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

Chapter 7

PostGIS 도형 활용:
응용 프로그&#xl
빌드

7.1 MapServer 활용

미네소타 맵서버(Minnesota MapServer)는 OpenGIS 웹 매서버 사양서를 준수하는 인터넷 웹 매핑 서버입니다.

- MapServer 홈페이지는 <http://mapserver.org> 입니다.
- <http://www.opengeospatial.org/standards/wms> 에서 OpenGIS 웹 맵 사양서넜 쀾아௏ 수 있습니다.

7.1.1 $\mathbb{R}^n$ and $\mathbb{C}^n$

MapServer와 함께 PostGIS를 사용하려면 MapServer설알아야 하는데 MapServer설정은 이 문서ಔ위를 벗어납니다 이 단원에서특정 PostGIS 문제점 및 설정 상세 정보다룰 것입니다.

MapServer와 함૨ PostGIS를 사용하려면 다음 프로그램이 필요합니다:

- PostGIS 0.6 이상 버전
- MapServer 3.5 이상 버전

MapServer는 다 른 어 떤 PostgreSQL 클 라 이 언 트 &#xc640
동 일 한 방 식 으 로 -- libpq 인 터 페 이 스 &#x
이 용 해 서 PostGIS/PostgreSQL 데 이 터 에 접 근 합 &#x
즉 PostGIS 서 버 에 네 트 워 크 연 결 된 어 &#xl
컴 퓨 터 에 라 도 MapServer를 설 치 할 수 있 &#x
PostGIS를 데 이 터 소 스 로 이 용 할 수 있 &#x
뜻 입 니 다 두 시 스 템 간 의 연 결 이
빠 를 수 록 좋 습 니 다.

1. "--with-postgis" 설정옵션을포함하는,사용원하는옵션으로 MapServer를컴파일하설치하십시오.
2. 사용자 MapServer의맵파일안에 PostGIS 레일&#xd944;쀀하십시오.다음은그예시일자

```
LAYER
  CONNECTIONTYPE postgis
  NAME "widehighways"
  # &#xc6d0;&#xaca9; &#xacf5;&#xac04; ↵
    &#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xc5d0; &#xc5f0;&#xacb0;
  CONNECTION "user=dbuser dbname=gisdatabase host=bigserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
  # 'roads' &#xd14c;&#xc774;&#xbel4;&#xc758; 'geom' &#xc5f4;&#xc5d0;&#xc11c; ↵
    &#xb77c;&#xc778;&#xc744; &#xd68d;&#xb4dd;
  DATA "geom from roads using srid=4326 using unique gid"
  STATUS ON
  TYPE LINE
  # &#xbc94;&#xc704; &#xb0b4;&#xbd80;&#xc758; &#xb77c;&#xc778; ↵
    &#xac00;&#xc6b4;&#xb370; &#xb113;&#xc740; ↵
    &#xace0;&#xc18d;&#xb3c4;&#xb85c;&#xb9cc; &#xb80c;&#xb354;&#xb9c1;
  FILTER "type = 'highway' and numlanes >= 4"
  CLASS
    # &#xcd08;&#xace0;&#xc18d;&#xb3c4;&#xb85c;&#xb97c; 2&#xd53d;&#xc140; ↵
      &#xb108;&#xbe44;&#xc758; &#xbcd1d;&#xc740; &#xc0c9;&#xc73c;&#xb85c;
    EXPRESSION ([numlanes] >= 6)
  STYLE
    COLOR 255 22 22
    WIDTH 2
  END
END
CLASS
  # &#xb2e4;&#xb978; &#xbaa8;&#xb4e0; &#xb3c4;&#xb85c;&#xb97c; 1&#xd53d;&#xc140; ↵
    &#xb108;&#xbe44;&#xc758; &#xc5b4;&#xb450;&#xc6b4; &#xc0c9;&#xc73c;&#xb85c;
  EXPRESSION ([numlanes] < 6)
  STYLE
    COLOR 205 92 82
  END
END
END
```

이예시에서, PostGIS에 특화된 지시&#xc79다음과 같습니다:

CONNECTIONTYPE PostGIS 낀이어의 경우, 언제나
"postgis"입니다.

CONNECTION ˤ음과같은표준키및값&#xc758
집합인'연결스트링(connection string)이데&
연결을관장합니다(기򼿈값은
로감싸여있습니다).

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

```
&#xe44; &#xc5b4; &#xc788; &#xb294; &#xc5f0; &#xacb0; &#xc2a4; &#xd2b8; &#xb9c1; &#xb3c4; &#xc5ec; &#xc804; &#xc720; &#xd6a8; &#xd558; &#xba70; , &#xc5b4; &#xb5a4; &#xd0a4; / &#xac12; &#xc30d; &#xc774; &#xb77c; &#xb3c4; &#xc0dd; &#xb7b5; &#xd560; &#xc218; &#xc788; &#xc2b5; &#xb2c8; &#xb2e4; . &#xcd5c; &#xc18c; &#xd55c; , &#xc5f0; &#xd544; &#xc694; &#xd55c; &#xb370; &#xc774; &#xd130; &#xbca0; &#xc774; &#xc2a4; &#xba85; &#xbc0f; &#xc0ac; &#xbcf4; &#xd1b5; &#xc81c; &#xacf5; &#xd558; &#xae30; &#xbc14; &#xb78d; &#xb2c8; &#xb2e4; .
```

DATA 이 파 라 미 터 의 서 식 은 유 일 한 <primary key> 를 이 용 한 <geocolumn> from <tablename> using srid=<srid> 입 라 때 열 은 맵 에 렌 더 링 될 공 간

```
&#xc5f4;&#xc744; &#xb9d0;&#xd558;&#xace0;, SRID&#xb294; &#xc5f4;&#xc774; &#xc0ac;&#xc6a9;&#xd558;&#xb294;
SRID&#xc774;&#xb7f0; &#xae30;&#xbcf8; &#xd0a4;&#xb294; &#xd14c;&#xc774;&#xbe14; &#xae30;&#xbcf8;
&#xd0a4;(&#xb610;&#xb294; &#xb2e4;&#xb978; &#xc5b4;&#xb5a4; &#xc720;&#xc77c;&#xac12;&#xc744; &#xb2f4;
&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xac00;&#xc9c4; &#xc5f4;)&#xc785;&#xb2c8;&#xb2e4;.
```

```
"using srid" &#xbcf0; "using unique" &#xc808;(clause)&#xc744; &#xc0dd;&#xb7b5;&#xd560; &#xc218; &#xc788;&#xc774;
&#xc774;&#xb7f0; &#xacbd;&#xc6b0; &#xac00;&#xb2a5;&#xd558;&#xb2e4;&#xb74; MapServer&#xac00; &#xc790;
&#xc815;&#xd655;&#xd55c; &#xac12;&#xc744; &#xc54c;&#xc544;&#xb0bc; &#xac83;&#xc785;&#xb2c8;&#xb2e4;
&#xadf8;&#xb7ec;&#xb098; &#xac01; &#xb9f5;&#xc744; &#xadf8;&#xb9ac;&#xae30; &#xc704;&#xd574; &#xc11c;
&#xc0c1;&#xc5d0;&#xc11c; &#xc5b4;&#xb290; &#xc815;&#xb3c4;&#xc758; &#xcd94;&#xac00; &#xcffc;&#xb9ac;
&#xc2e4;&#xd589;&#xd558;&#xb294; &#xc77c;&#xc744; &#xac10;&#xb2f9;&#xd574;&#xc57c; &#xd560; &#xac83;&
```

PROCESSING 기존 연결을 끊지 않고 재사 복수 의 레이 어가 있을 경우 CLOSE_CO
절을 삽입하୴ 속도가 향상됩&
더 자세한 설அ이 필요한 경우
MapServer PostGIS Performance Tips 를 참조하십시오.

FILTER 필터란 일본적으로 SQL 쿼리 안있는 "WHERE" 키워드를 따르는 논리
상응하는 유효한 SQL 스트링이열
합니다. 따라서 예를 들어 6개
이상 차선을 가진 도로만을 렌&
"num_lanes >= 6"라는 필터를 사용하십시오.

3. 사용자 공간 데이 터베이 스 안에
사용자가 그리게 될 모든 레이어
대해 공간(GiST) 인덱스가 ୌ드되어
있는지 확인하십시오.

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

4. MapServer를 이용해서 사용자 레이어¹
쿼리하게 될 경우 사용자의 DATA 구&
"using unique" 절을 삽입해야 할 것입니다.
쿼리 실행시 MapServer는 각 공간 레코´
대해 유일한 식별자를 요구하·
MapServer의 PostGIS 모듈은 이 유일한 식별౹
제공하기 위해 사용자가 설정ൕ
유일값을 이용합니다. 가장 좋Ç
଩க은 테이블 기본 키를 쓰는
것입니다.

7.1.2 FAQ

1. 제 맵 파일 안에 EXPRESSION 을 사용할
때 제 테이블에 해당 값이 분அ
있는 데도 불구하고, 조౴ 문이
항상 거짓을 본환합니다.

```
shapefile&#xacfc;&#xb294; &#xb2ec;&#xb9ac; &#xd45c;&#xd604;&#xc2dd;(expression)&#xc5d0; &#xcc38;&#xc870;&#xb4;  
PostGIS &#xd544;&#xb4dc;&#xb85;&#xc740; &#xc18c;&#xbb38;&#xc790; &#xc5ec;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;
```

```
EXPRESSION ([numlanes] >= 6)
```

2. 제 shapefile에 사용하는 필터가 동일ൕ
데이터를 가진 제 PostGIS 테이블에
대해서는 동작하지 않습니다.


```
FILTER "type = 'highway' and numlanes >= 4"
```

- 일반적으로,주어진맵에더많읍피처를그릴수록shapefile보다PostGIS가느됩니다.상대적으로적은피처(은개)를가진맵의경우,PostGIS가빠른편입니다.고밀도피처(수천개)를가진맵의경우,항상PostGIS가더느른ಃ입니다.그리기작업속도에속상당한문제가있을경우,사용작테이블에공간인덱스를빌드한않았을가능성이있습니다.

쿼리 속도가 빠르려면 사용자
공간 테이블에 유일 키가 있어앿
하며 해당 유일 키에 대해 인덱일
빌드되어 있어야 합니다 사용자
DATA 행에 USING UNIQUE 절을 삽입해서 MapServer가
사용할 유일 키를 설정할 수 있수

```
&#xb124;! MapServer&#xb294; &#xc9c0;&#xb9ac;&#xd615; &#xc5f4;&#xc744; &#xc5b8;&#xc81c;&#xb098; SRID
4326&#xc744; &#xc774;&#xc6a9;&#xd558;&#xb294; &#xb3c4;&#xd615; &#xc5f4;&#xacfc; &#xb3d9;&#xc77c;&#xd558;&#xb294;
&#xc778;&#xc2dd;&#xd569;&#xb2c8;&#xb2e4;. &#xc0ac;&#xc6a9;&#xc790;&#xc758; DATA &#xad6c;&#xbb38;&#xc5d0;
"using srid=4326" &#xc808;&#xc744; &#xc0bd;&#xc785;&#xd588;&#xb294;&#xc9c0;&#xb9cc; &#xd655;&#xc778;&#xd558;
&#xb429;&#xb2c8;&#xb2e4;. &#xb2e4;&#xb978; &#xbaa8;&#xb4e0; &#xc791;&#xc5c5;&#xc740; &#xb3c4;&#xd615;&#xc744;
&#xacbd;&#xc6b0;&#xc640; &#xc815;&#xd655;&#xd788; &#xb3d9;&#xc77c;&#xd558;&#xac8c; &#xc774;&#xb8e8;&#xc5b8;
```

```
MapServer&#x00; &#xc880; &#xb354; &#xbcf5; &#xc7a1; &#xd55c; &#xcffc; &#xb9ac; &#xc758; &#xacb0; &#xacfc; &#xb97c;
&#xc774; &#xd574; &#xd558; &#xb3c4; &#xb85d; &#xb3d5; &#xae30; &#xc704; &#xd55c; &#xc815; &#xbcf4; &#xb97c; &#xcd94; &#
&#xb370; USING &#xd5c8; &#xc704; (pseudo) SQL &#xc808; &#xc744; &#xc774; &#xc6a9; &#xd569; &#xb2c8; &#xb2e4;. &#xb354;
&#xc815; &#xd655; &#xd558; &#xac8c; &#xb294;. &#xbdf0; &#xb610; &#xb294; &#xd558; &#xc704; &#xc9d1; &#xd569; (subselect) &
```

```
(DATA &#xc815;&#xc758;&#xc5d0;&#xc11c; "FROM" &#xc624;&#xb978;&#xcabd;&#xc5d0; &#xc624;&#xb294;) &#xc18c;&#xc2
&#xd14c;&#xc774;&#xbel4;&#xb85c; &#xc0ac;&#xc6a9;&#xd560; &#xb54c;, MapServer&#xac00; &#xac01; &#xd589;&#xc5d0;
&#xb300;&#xd55c; &#xc720;&#xc77c;&#xd55c; &#xc2dd;&#xbcc4;&#xc790;&#xb294; &#xbb3c;&#xb860; &#xd14c;&#xc774;&#
SRID&#xb97c; &#xc790;&#xb3d9;&#xc801;&#xc73c;&#xb85c; &#xd30c;&#xc545;&#xd558;&#xae30; &#xc5b4;&#xb824;&#xc6c
USING &#xc808;&#xc774; MapServer&#xc5d0; &#xc774; &#xb450; &#xac00;&#xc9c0; &#xc815;&#xbcf4;&#xb97c; &#xb2e4;&#
&#xac19;&#xc774; &#xc81c;&#xacf5;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;:
```

```
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> 맵 쿼 리 실 행 시 어 떤 행 인 식 별 하 기 위 해 MapServer 가 각 행 에 대 유 일 ID 를 요 구 합 니 다. 일 반 적 으 시 스 템 테 이 블 에 서 기 본 키 를 인 식 합 니 다. 하 지 만 뷰 및 하 위 알 려 진 유 일 값 열 을 자 동 적 으 &#xb85 갖 추 지 않 습 니 다. MapServer 의 쿼 리 기 &#xb2 활 용 하 고 자 한 다 면. 사 용 자 의 뷰 또 는 하 위 집 합 이 유 일 값 열 &#xc7 포 함 하 는 지 확 인 하 고. USING UNIQUE 절 &#xc74 통 해 그 사 실 을 선 언 하 십 시 오. 예 를 들 어. 이 런 목 적 을 위 해 명 &#xc 테 이 블 의 기 본 키 를 선 택 할 수 있 고. 또 는 결 과 출 력 물 에 서 유 &#xc7 것 이 보 장 되 는 다 른 어 떤 열 이 &#xb7 선 택 할 수 있 습 니 다.

Note



```
"&#xb9f5;  &#xcffc;&#xb9ac;&#xd558;&#xae30;"&#xb780;  &#xd574;&#xb2f9;  &#xc704;&#xce58;&#xc758;
&#xc815;&#xbcf4;&#xb97c;    &#xad6c;&#xd558;&#xae30;    &#xc704;&#xd574;    &#xb9f5;&#xc744;
&#xd074;&#xb9ad;&#xd558;&#xb294;    &#xd589;&#xc704;&#xc785;&#xb2c8;&#xb2e4;.          DATA
&#xc815;&#xc758;&#xc5d0; &#xc788;&#xb294; SQL &#xcffc;&#xb9ac;&#xb97c; &#xd1b5;&#xd55c; "&#xb9f5;
&#xcffc;&#xb9ac;"&#xc640; &#xcc29;&#xac01;&#xd558;&#xc9c0; &#xb9c8;&#xc2ed;&#xc2dc;&#xc624;.
```

```

USING SRID=<sruid> PostGIS&#xb294;, MapServer&#xc5d0; &#xc815;&#xd655;&#xd55c; &#xb370;&#xc774;&#xd130;&#xb97
&#xbc18;&#xd658;&#xd558;&#xae30;&#xc704;&#xd574;, &#xb3c4;&#xd615;&#xc774; &#xc5b4;&#xb5a4; &#xacf5;&#xac
&#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c;&#xc744; &#xc0ac;&#xc6a9;&#xd558;&#xb294;&#xc9c0; &#xc54c;&#xc54
&#xd569;&#xb2c8;&#xb2e4;. &#xc77c;&#xbc18;&#xc801;&#xc73c;&#xb85c; PostGIS &#xb370;&#xc774;&#xd130;&#xbca
&#xc788;&#xb294; "geometry_columns" &#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc11c; &#xc774; &#xc815;&#xbcf4;&#xb
&#xcc3e;&#xc744; &#xc218; &#xc788;&#xc9c0;&#xb9cc;, &#xbdff0; &#xbcf0; &#xd558;&#xc704;&#xc9d1;&#xd569;&#xac
&#xc19;&#xc774; &#xc2e4;&#xc2dc;&#xac04;&#xc73c;&#xb85c;(on the fly) &#xc0dd;&#xc131;&#xb41c; &#xd14c;&#xc7
&#xbd88;&#xac00;&#xb2a5;&#xd569;&#xb2c8;&#xb2e4;. &#xb530;&#xb77c;&#xc11c; USING SRID= &#xc635;&#xc158;
&#xc774;&#xc6a9;&#xd558;&#xba74; DATA &#xc815;&#xc758; &#xc548;&#xc5d0; &#xc815;&#xd655;&#xd55c; SRID&#x
&#xc124;&#xc815;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.

```

7.1.4 **;**

간단한예제로시작해서단계를높여가보도록합시다다음 MapServer 레정의를살펴보십시오::

```
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
```

이 레이어는 도로 테이블의 모든
 도로 도형을 검정색 라인으로 표&#
 것입니다.

이제 최소한 1:100,000 축척으로 확대하&#
 전까지는 고속도로만 보이도록
 하려 한다고 해봅시다,. 다음 두 레&
 이런 효과를 낼 것입니다.

```
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
```

축척이 1:100,000을 초과할 경우 첫 번째
 레이어를 사용해서 "highway" 유형의 도&#

```
&#xac0;&#xc815;&#xc0c9; &#xb77c;&#xc778;&#xc73c;&#xb85c; &#xd45c;&#xcd9c;&#xd569;&#xb2c8;&#xb2e4;. FILTER
&#xc635;&#xc158;&#xc774; "highway" &#xc720;&#xd615;&#xc758; &#xb3c4;&#xb85c;&#xb9cc; &#xd45c;&#xcd9c;&#xb418;&#
&#xd569;&#xb2c8;&#xb2e4;.
```

축척이 1:100,000 미만일 경우 두 번째 레&#
사용해서 고속도로를 두 배 너비&#
빨강색 라인으로, 다른 도로들을
일반적인 검정색 라인으로 표출
할 것입니다.

따라서, MapServer의 기능성만을 이용해재미있는 작업을 두어 개 해보았을 DATA SQL 선언문은 단순한 형식을 유지있습니다, 도로의 명칭이 (어떤 이또다른 테이블에 저장되어 있다가정하면, 도로명을 읽어 와 도로 라벨을 표출하려면 두 테이블을 연결해야 합니다.

```
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
```

이주석(annotation) 레이어는 축척이 1:20,000 이경우 모든 도로에 초록색 라벨을 추가합니다. DATA 정의에 어떻게 SQL 연이용하는지도 보여주고 있습니&#xb

7.2 Java 클라이언트(JDBC)

```
Java &#xd074;&#xb77c;&#xc774;&#xc5b8;&#xd2b8;&#xb294;&#xd14d;&#xc2a4;&#xd2b8;&#xd45c;&#xd604;&#xc2dd;&#xc74d;&#xd1b5;&#xd574;&#xc9c1;&#xc811;&#xb610;&#xb294; PostGIS &#xbc88;&#xb4e4;&#xc5d0;&#xd3ec;&#xd568;&#xb418;&#xc788;&#xb294; JDBC &#xd655;&#xc7a5;&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc744;&#xc774;&#xc6a9;&#xd574; PostgreSQL &#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xc548;&#xc5d0;&#xc788;&#xb294; PostGIS "&#xb3c4;&#xac1d;&#xccb4;&#xc5d0;&#xc811;&#xadfc;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.&#xd655;&#xc7a5;&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xac1d;&#xccb4;&#xb97c;&#xc774;&#xc6a9;&#xd558;&#xb824;&#xba74;.&#xc0ac;&#xc74d; CLASSPATH&#xc5d0; "postgres.jar" JDBC &#xb4dc;&#xb77c;&#xc774;&#xbc84;&#xd328;&#xd0a4;&#xc9c0;&#xc640;&#xd568;&#xaed8; "postgis.jar" &#xd30c;&#xc77c;&#xc774;&#xc874;&#xc7ac;&#xd574;&#xc57c;&#xd569;&#xb2c8;&#xb2e4;.
```

```
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 {
        /*
         * JDBC &#xb4dc;&#xb77c;&#xc774;&#xbc84;&#xb97c; &#xb85c;&#xb4dc;&#xd558;&#xace0; &#xc5f0;&#xacb0;&#xc744; &#xd655;&#xb9bd;&#xd569;&#xb2c8;&#xb2e4;.
         */
        Class.forName("org.postgresql.Driver");
        String url = "jdbc:postgresql://localhost:5432/database";
        conn = DriverManager.getConnection(url, "postgres", "");
        /*
         * &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc744; &#xc5f0;&#xacb0;&#xc5d0; &#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;. addDataType() &#xba54;&#xc18c;&#xb4dc;&#xb97c;
         * &#xd638;&#xcd9c;&#xd558;&#xae30; &#xc804;&#xc5d0; &#xc5f0;&#xacb0;&#xc744; pgSQL &#xc5d0; &#xd2b9;&#xd654;&#xb41c; &#xc5f0;&#xacb0; &#xc2e4;&#xd589;&#xc5d0;
         * &#xc9c0;&#xc815;&#xd574;&#xc57c; &#xd55c;&#xb2e4;&#xb294; &#xc810;&#xc5d0; &#xc8fc;&#xc758;&#xd569;&#xb2c8;&#xb2e4;.
         */
        ((org.postgresql.PGConnection) conn).addDataType("geometry", Class.forName("org.postgis. PGgeometry"));
        ((org.postgresql.PGConnection) conn).addDataType("box3d", Class.forName("org.postgis. PGbox3d"));
        /*
         * &#xc120;&#xc5b8;&#xbb38;&#xc744; &#xc0dd;&#xc131;&#xd574;&#xc11c; &#xc120;&#xbcc4; &#xcffc;&#xb9ac;&#xb97c; &#xc2e4;&#xd589;&#xd569;&#xb2c8;&#xb2e4;.
         */
        Statement s = conn.createStatement();
        ResultSet r = s.executeQuery("select geom,id from geomtable");
        while( r.next() ) {
            /*
             * &#xb3c4;&#xd615;&#xc744; &#xac1d;&#xccb4;&#xb85c; &#xc5bb;&#xc5b4;&#xc628; &#xb2e4;&#xc74c; &#xd574;&#xb2f9; &#xac1d;&#xccb4;&#xb97c; &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc5d0; &#xc9c0;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.
             * &#xb3c4;&#xd615;&#xc744; &#xcd9c;&#xb825;&#xd569;&#xb2c8;&#xb2e4;.
             */
            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();
    }
}

```

"PGgeometry" 객체는 포인트, 라인스트링, 폴리곤, 멀티폴리곤, 멀티라인스링, 멀티폴리곤 등의 유형에 의존하특정 지형 도형 객체(추출 클래스

"도형"의 하위 클래스)를 담고 있는래퍼(wrapper) 객체입니다.

```
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());
        }
    }
}
```

확장 프로그램 객체를 위한 JavaDoc은 기하학적 객체의 다양한 데이터 접근자(accessor) 함수에 대한 참조를 제자

7.3 C 클라이언트(libpq)

...

7.3.1 텍스트 커서

...

7.3.2 바이너리 커서

...

Chapter 8

PostGIS Reference

PostGIS 3.3.9dev

Note

PostGIS 3.3.9dev

Note!

8.1 PostgreSQL PostGIS Geometry/Geography/Box

8.1.1 box2d

box2d — The type representing a 2-dimensional bounding box.

box2d

box3d

The representation contains the values `xmin`, `ymin`, `xmax`, `ymax`. These are the minimum and maximum values of the X and Y extents.

`box2d` objects have a text representation which looks like `BOX (1 2, 5 6)`.

ʤ팅 동작

이 섹션에는 자동 형변환 &#bfd0;&#b9cc; 아이 데이터 유형에 허용되는 명시스팅이 나열됩니다.

변환 대상	동작
<code>box3d</code>	자동적
<code>geometry</code>	자동적

관련 정보

Section [15.7](#)

8.1.2 box3d

`box3d` — The type representing a 3-dimensional bounding box.

설명

`box3d` 지오메트리 또는 지오메트리 셋의 범위를 표현하는데 사용되󊳵간 데이터 타입입니다. `ST_3DExtent`는 `box3d` 객체를 반환합니다.

The representation contains the values `xmin`, `ymin`, `zmin`, `xmax`, `ymax`, `zmax`. These are the minimum and maximum values of the X, Y and Z extents.

`box3d` objects have a text representation which looks like `BOX3D (1 2 3, 5 6 5)`.

스팅 동작

이 섹션에는 자동 형변환 &#bfd0;&#b9cc; 아이 데이터 유형에 허용되는 명시스팅이 나열됩니다.

변환 대상	동작
<code>box</code>	자동적
<code>box2d</code>	자동적
<code>geometry</code>	자동적

관련 정보

Section [15.7](#)

8.1.3 geometry

geometry — geography

geometry

geography

All spatial operations on geometry use the units of the Spatial Reference System the geometry is in.

geometry

geometry

geometry	geometry
box	geometry
box2d	geometry
box3d	geometry
bytea	geometry
geography	geometry
text	geometry

geometry

Section 4.1, Section 4.3

8.1.4 geometry_dump

geometry_dump — A composite type used to describe the parts of complex geometry.

geometry_dump

geometry_dump is a **composite data type** containing the fields:

- geom** - a geometry representing a component of the dumped geometry. The geometry type depends on the originating function.
- path[]** - an integer array that defines the navigation path within the dumped geometry to the **geom** component. The path array is 1-based (i.e. **path[1]** is the first element.)

It is used by the **ST_Dump*** family of functions as an output type to explode a complex geometry into its constituent parts.

geometry_dump

Section 15.6

8.1.5 geography

geography — The type representing spatial features with geodetic (ellipsoidal) coordinate systems.

geometry;

geography; The geography type provides more accurate results by taking the ellipsoidal model into account.

Spatial operations on the geography type provide more accurate results by taking the ellipsoidal model into account.

geometry;

geography; The geography type provides more accurate results by taking the ellipsoidal model into account.

geometry	geometry

geometry;

Section 4.3, Section 4.3

8.2 AddGeometryColumn

8.2.1 AddGeometryColumn

AddGeometryColumn — Adds a new geometry column to an existing table.

Synopsis

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

geometry;

geography; The geography type provides more accurate results by taking the ellipsoidal model into account.

Note

```
&#xbcc0;&#xacbd;&#xc0ac;&#xd56d:: 2.0.0 &#xc774; &#xd568;&#xc218;&#xb294; geome-
try_columns&#xc774; &#xc2dc;&#xc2a4;&#xd15c; &#xce74;&#xd14c;&#xace0;&#xb9ac;&#xc5d0;&#xc11c;
&#xc77d;&#xd788;&#xb294; &#xbdf0;&#xc778; &#xb3d9;&#xc548; &#xb354; &#xc774;&#xc0c1;
geometry_columns&#xc744; &#xc5c5;&#xb370;&#xc774;&#xd2b8; &#xd558;&#xc9c0;
&#xc54a;&#xc2b5;&#xb2c8;&#xb2e4; &#xc774;&#xac83;&#xc740; &#xae30;&#xbcf8;&#xc801;&#xc73c;&#xb85c;
&#xcee8;&#xc2a4;&#xd2b8;&#xb808;&#xc778;&#xd2b8;&#xb97c; &#xb9cc;&#xb4e4;&#xc9c0;
&#xc54a;&#xc9c0;&#xb9cc; &#xb300;&#xc2e0;&#xc5d0; PostgreSQL&#xc758; &#xb0b4;&#xc7a5;&#xb41c;
&#xd0c0;&#xc785; &#xbcc0;&#xacbd; &#xb3d9;&#xc791;&#xc774; &#xb300;&#xc2e0;&#xd569;&#xb2c8;&#xb2e4;
&#xadf8;&#xb798;&#xc11c; &#xc608;&#xb97c; &#xb4e4;&#xc5b4; WGS84 POINT &#xceec;&#xb7fc;&#xc744;
&#xc774; &#xd568;&#xc218;&#xb85c; &#xb9cc;&#xb4e0;&#xb2e4;&#xba74; &#xc774; &#xd568;&#xc218;&#xb294;
&#xb2e4;&#xc74c;&#xacfc; &#xb3d9;&#xb4f1;&#xd569;&#xb2c8;&#xb2e4; ALTER TABLE some_table ADD
COLUMN geom geometry(Point,4326);
&#xbcc0;&#xacbd; &#xc0ac;&#xd56d:: 2.0.0 &#xbc84;&#xc804; &#xc81c;&#xc57d;&#xc870;&#xac74;&#xc758;
&#xad6c;&#xc2dd; &#xc2b5;&#xc131;&#xc774; &#xd544;&#xc694;&#xd560; &#xacbd;&#xc6b0;
&#xae30;&#xbcf8;&#xac12;&#xc778; use_typed &#xb97c; &#xc774;&#xc6a9;&#xd558;&#xb418;
&#xac70;&#xc9d3;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;
```

Note

```
&#xbcc0;&#xacbd; &#xc0ac;&#xd56d;; 2.0.0 &#xbc84;&#xc804;. &#xb354; &#xc774;&#xc0c1; &#xbdf0;&#xb97c;
&#xc9c1;&#xc811; geometry_columns&#xc5d0; &#xb4f1;&#xb85d;&#xc2dc;&#xd0ac; &#xc218;&#xb294;
&#xc5c6;&#xc9c0;&#xb9cc;; &#xb3c4;&#xd615; typmod &#xd14c;&#xc774;&#xbe14; &#xb3c4;&#xd615;&#xc744;
&#xae30;&#xbc18;&#xc73c;&#xb85c; &#xbe4c;&#xb4dc;&#xb41c; , &#xadf8;&#xb9ac;&#xace0;
&#xb798;&#xd37c; &#xd568;&#xc218; &#xc5c6;&#xc774; &#xc0ac;&#xc6a9;&#xb418;&#xb294;
&#xbdf0;&#xb294; &#xc790;&#xc2e0;&#xc758; &#xd398;&#xc5b4;&#xb7f0;&#xd2b8; &#xd14c;&#xc774;&#xbe14;
&#xc5f4;&#xc758; typmod &#xc2b5;&#xc131;&#xc744; &#xc0c1;&#xc18d;&#xd558;&#xae30;
&#xb54c;&#xbb38;&#xc5d0; &#xc2a4;&#xc2a4;&#xb85c;&#xb97c; &#xc815;&#xd655;&#xd788;
&#xb4f1;&#xb85d;&#xc2dc;&#xd0b5;&#xb2c8;&#xb2e4; &#xc774;&#xb7f0; &#xbdf0;
&#xb3c4;&#xd615; &#xc5f4;&#xc744; geometry_columns&#xc5d0; &#xc815;&#xd655;&#xd788;
&#xb4f1;&#xb85d;&#xc2dc;&#xd0a4;&#xb824;&#xba74;, &#xb2e4;&#xb978; &#xb3c4;&#xd615;&#xc744;
&#xcd9c;&#xb825;&#xd558;&#xb294; &#xb3c4;&#xd615; &#xd568;&#xc218;&#xb97c;
&#xc774;&#xc6a9;&#xd558;&#xb294; &#xbdf0;&#xb97c; typmod &#xb3c4;&#xd615;&#xc73c;&#xb85c;
&#xd615;&#xbcc0;&#xd658;&#xc2dc;&#xcf1c;&#xc57c; &#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. Sec-
tion 4.6.3 &#xc744; &#xcc38;&#xc870;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

- ✔ This method implements the **OGC 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

```
&#x0ac1c;&#xc120;&#xc0ac;&#xd56d;. 2.0.0 &#xbc84;&#xc804;. use_typmod &#xc778;&#xc790;&#xac00;&#xcd94;&#xac00;&#  
&#xc81c;&#xc57d;&#xc870;&#xac74;&#xae30;&#xbc18;&#xb3c4;&#xd615;&#xc5f4;&#xb300;&#xc2e0;&#xae30;&#xbcf8;&#  
typmod &#xb3c4;&#xd615;&#xc5f4;&#xc744;&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;
```

예&#xc81c:

```
-- &#xb370;&#xc774;&#xd130;&#xb97c; &#xb2f4;&#xae30; &#xc704;&#xd55c; ↔  
    &#xc2a4;&#xd0a4;&#xb9c8;&#xb97c; &#xc0dd;&#xc131;  
CREATE SCHEMA my_schema;  
-- &#xc0c8;&#xb85c;&#xc6b4; &#xb2e8;&#xc21c; PostgreSQL &#xd14c;&#xc774;&#xbe14; ↔  
    &#xc0dd;&#xc131;  
CREATE TABLE my_schema.my_spatial_table (id serial);
```

```
-- &#xb2e8;&#xc77c; "id"&#xc5f4;&#xc744; &#xac00;&#xc9c4; &#xb2e8;&#xc21c; ↵
&#xd14c;&#xc774;&#xbel4;&#xc744; &#xbcfc4;&#xc5ec;&#xc8fc;&#xb294; ↵
&#xd14c;&#xc774;&#xbel4;&#xc744; &#xc124;&#xba85;&#xd558;&#xae30;
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)

-- &#xd14c;&#xc774;&#xbel4;&#xc5d0; &#xacf5;&#xac04; &#xc5f4;&#xc744; &#xcd94;&#xac00;
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geom',4326,'POINT',2);

-- &#xc608;&#xc804; &#xc81c;&#xc57d;&#xc870;&#xac74; &#xae30;&#xbc18; ↵
&#xc2b5;&#xc131;&#xc744; &#xc774;&#xc6a9;&#xd574;&#xc11c; ↵
&#xd3ec;&#xc778;&#xd2b8;&#xb97c; &#xcd94;&#xac00;
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geom_c',4326,'POINT',2, false);

-- &#xc608;&#xc804; &#xc81c;&#xc57d;&#xc870;&#xac74; &#xc2b5;&#xc131;&#xc744; ↵
&#xc774;&#xc6a9;&#xd574;&#xc11c; &#xb9cc;&#xace1; &#xd3f4;&#xb9ac;&#xace4;&#xc744; ↵
&#xcd94;&#xac00;
SELECT AddGeometryColumn ('my_schema','my_spatial_table','geomcp_c',4326,'CURVEPOLYGON',2, ↵
false);

-- &#xc0c8; &#xb3c4;&#xd615;&#xc744; &#xcd94;&#xac00;&#xd588;&#xc74c;&#xc744; ↵
&#xbcfc4;&#xc5ec;&#xc8fc;&#xb3c4;&#xb85d; &#xd14c;&#xc774;&#xbel4;&#xc744; ↵
&#xb2e4;&#xc2dc; &#xc124;&#xba85;&#xd558;&#xae30;
\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 &#xbdff0;&#xb3c4; &#xc0c8; &#xc5f4;&#xb4e4;&#xc744; ↵
&#xb4f1;&#xb85d;&#xd569;&#xb2c8;&#xb2e4;. --
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
```

관련 정보

DropGeometryColumn, DropGeometryTable, Section 4.6.2, Section 4.6.3

8.2.2 DropGeometryColumn

DropGeometryColumn — 공간 테이블에서 지오메트컬럼을 제거합니다.

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);
```

설명

```
&#xacf5;&#xac04;&#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc11c;&#xb3c4;&#xd615;&#xc5f4;&#xc744;&#xc81c;&#xac70;&#x  
schema_name&#xc774; geometry_columns &#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc788;&#xb294;&#xd14c;&#xc774;&#xbe14;  
&#xd589;&#xc758; f_table_schema &#xd56d;&#xbaa9;&#xacfc;&#xc77c;&#xce58;&#xd574;&#xc57c;&#xd55c;&#xb2e4;&#xb29  
&#xc810;&#xc5d0;&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```



This method implements the **OGC 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

```
&#xbcc0;&#xacbd;      &#xc0ac;&#xd56d;;      2.0.0      &#xbc84;&#xc804;.      &#xd558;&#xc704;
&#xd638;&#xd658;&#xc131;&#xc744;      &#xc704;&#xd574;      &#xc774;      &#xd568;&#xc218;&#xb97c;
&#xc81c;&#xacf5;&#xd569;&#xb2c8;&#xb2e4;.      &#xc774;&#xc81c;&#xb294;      geometry_columns&#xc774;
&#xc2dc;&#xc2a4;&#xd15c;      &#xce74;&#xd0c8;&#xb85c;&#xadf8;&#xb97c;      &#xae30;&#xbc18;&#xc73c;&#xb85c;
&#xd558;&#xb294; &#xbdf0;&#xc774;&#xae30; &#xb54c;&#xbbb38;&#xc5d0;. &#xb2e4;&#xb978; &#xc5b4;&#xb5a4;
&#xd14c;&#xc774;&#xbe14;      &#xc5f4;&#xacfc;&#xb3c4;      &#xb9c8;&#xcc2c;&#xac00;&#xc9c0;&#xb85c;
&#xb3c4;&#xd615;      &#xc5f4;&#xb3c4;      ALTER TABLE      &#xc774;      &#xc774;&#xc6a9;&#xd574;&#xc11c;
&#xc0ad;&#xc81c;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```



예제

```
SELECT DropGeometryColumn ('my_schema','my_spatial_table','geom');
-- &#xacb0;&#xacfc; &#xcd9c;&#xb825;&#xbb3c; --
dropgeometrycolumn
-----
my_schema.my_spatial_table.geom effectively removed.

-- PostGIS 2.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc5d0;&#xc11c; &#xc704; &#xcffc;&#xb9ac;&#xb294; &#xd45c;&#xc900; &#xb300;&#xccb4; &#xd14c;&#xc774;&#xbe14;&#xacfc;&#xb3c4; &#xb3d9;&#xb4f1;&#xd569;&#xb2c8;&#xb2e4; .
-- &#xc591;&#xcabd; &#xbaa8;&#xb450; geometry_columns&#xb85c;&#xbd80;&#xd130; &#xb4f1;&#xb85d; &#xd574;&#xc81c;&#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4; .
ALTER TABLE my_schema.my_spatial_table DROP column geom;
```


Synopsis

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

설명

Returns the integer SRID of the specified geometry column by searching through the GEOMETRY_COLUMNS table. If the geometry column has not been properly added (e.g. with the [AddGeometryColumn](#) function), this function will not work.

예제

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

관련 정보

ST_SRID

8.2.5 Populate_Geometry_Columns

Populate_Geometry_Columns — Ensures geometry columns are defined with type modifiers or have appropriate spatial constraints.

Synopsis

text **Populate_Geometry_Columns**(boolean use_typmod=true);

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

설명

지오메트리 컬럼이 타입 변경자정의되거나 적절한 공간 제약을 가지고 있는지 확인합니다. 이 함공간 관련 테이블들이 geometry_columns 뷰에 올바르게 등록되도록 합니다. 기유형 변경자를 가지지 않는 ન든 지오메트리 컬럼들을 유형 변경지진 지오메트리 컬럼들로 변환엣날식 동작을 원하면 use_typmod=false으로설정

하위 호환성 및 각 차일드 테이블서로 다른 도형 유형을 가질 수도 있는 테이블 상속 같은 공간 필요위해. 구 버전 확인 제약조건 습성󊳄속 지원합니다. 구 버전 습성이 필요하다면. use_typmod=false처럼 새 선택적인자를 거짓으로 패스해야 합니이렇게 하면 유형 변경자는 없지제약조건 3개가 정의된 도형 열을 생성할 것입니다. 다시 말해. ન든 도형 열이 적어도 3개의 제약조건진 테이블에 종속된다는 뜻입&#xc

- `enforce_dims_the_geom` - ensures every geometry has the same dimension (see [ST_NDims](#))
- `enforce_geotype_the_geom` - ensures every geometry is of the same type (see [ST_GeometryType](#))
- `enforce_srid_the_geom` - ensures every geometry is in the same projection (see [ST_SRID](#))

```
oid &#xd14c;&#xc774;&#xbe14;&#xc774;&#xc81c;&#xacf5;&#xb420;&#xacbd;&#xc6b0;,&#xc774;&#xd568;&#xc218;&#xb294;
&#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc788;&#xb294;&#xbaa8;&#xb4e0;&#xb3c4;&#xd615;&#xc5f4;&#xc758;SRID,
&#xcc28;&#xc6d0;,&#xadf8;&#xb9ac;&#xace0;&#xb3c4;&#xd615;&#xc720;&#xd615;&#xc744;&#xacb0;&#xc815;&#xd558;&#xb824;
&#xd558;&#xba70;,&#xd544;&#xc694;&#xd55c;&#xacbd;&#xc6b0;&#xc81c;&#xc57d;&#xc870;&#xac74;&#xc744;&#xcd94;&#xb2e4;
&#xd569;&#xb2c8;&#xb2e4;,&#xd568;&#xc218;&#xac00;&#xc81c;&#xb300;&#xb85c;&#xc791;&#xb3d9;&#xd588;&#xb2e4;&#xc62c;
&#xb14;&#xb978;&#xd589;&#xc774;geometry_columns &#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc0bd;&#xc785;&#xb41c;
&#xc544;&#xb2d0;&#xacbd;&#xc6b0;,&#xc608;&#xc678;&#xac00;&#xc7a1;&#xd600;&#xbb38;&#xc81c;&#xc810;&#xc744;
&#xc124;&#xba85;&#xd558;&#xb294;&#xc624;&#xb958;&#xba54;&#xc2dc;&#xc9c0;&#xac00;&#xb739;&#xb2c8;&#xb2e4;.
```

```
oid &#xd14c;&#xc774;&#xbe14;&#xacfc;&#xd568;&#xaed8;oid &#xbdf0;&#xac00;&#xc81c;&#xacf5;&#xb420;&#xacbd;&#xc6b0;
&#xc774;&#xd568;&#xc218;&#xb294;geometry_columns &#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc62c;&#xb14;&#xb978;
&#xd56d;&#xbaa9;&#xc744;&#xc0bd;&#xc785;&#xd558;&#xba70;&#xbdf0;&#xc5d0;&#xc788;&#xb294;&#xbaa8;&#xb4e0;
&#xb3c4;&#xd615;&#xc758;SRID,&#xcc28;&#xc6d0;,&#xc720;&#xd615;&#xc744;&#xacb0;&#xc815;&#xd558;&#xb824;
&#xd558;&#xc9c0;&#xb9cc;,&#xc81c;&#xc57d;&#xc870;&#xac74;&#xc744;&#xac15;&#xc81c;&#xd558;&#xc9c0;&#xb294;
&#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
&#xc774;&#xd568;&#xc218;&#xc758;&#xd30c;&#xb77c;&#xbbbf8;&#xd130;&#xac00;&#xc5c6;&#xb294;&#xbcc0;&#xc885;&#xc62c;
&#xacf5;&#xac04;&#xc81c;&#xc57d;&#xc870;&#xac74;&#xc744;&#xd14c;&#xc774;&#xbe14;&#xc758;&#xc801;&#xc808;&#xc801;
&#xacf3;&#xc5d0;&#xcd94;&#xac00;&#xd558;&#xba70;&#xba3c;&#xc800;&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xb0b4;
&#xbdf0;&#xc5d0;&#xc788;&#xb294;&#xbaa8;&#xb4e0;&#xacf5;&#xac04;&#xd14c;&#xc774;&#xbe14;&#xbcf0;&#xbdf0;&#xc5d0;
&#xb300;&#xd574;geometry_columns &#xd14c;&#xc774;&#xbe14;&#xc758;&#xc6a9;&#xb7c9;&#xc744;&#xc904;&#xc774;&#xb2e4;
&#xc2dc;&#xcc44;&#xc6b0;&#xb294;&#xd30c;&#xb77c;&#xbbbf8;&#xd130;&#xac00;&#xc788;&#xb294;&#xbcc0;&#xc885;&#xc62c;
&#xc704;&#xd55c;&#xb2e8;&#xc21c;&#xb798;&#xd37c;&#xc785;&#xb2c8;&#xb2e4;,&#xd30c;&#xb77c;&#xbbbf8;&#xd130;&#xc5c6;
&#xb294;&#xbcc0;&#xc885;&#xc740;&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xb0b4;&#xbdf0;&#xc5d0;
&#xac10;&#xc9c0;&#xb41c;&#xb3c4;&#xd615;&#xc5f4;&#xc758;&#xac1c;&#xc218;&#xbcf0;geometry_columns
&#xd14c;&#xc774;&#xbe14;&#xb85c;&#xc0bd;&#xc785;&#xb41c;&#xac1c;&#xc218;&#xb97c;&#xc694;&#xc57d;&#xd574;&#xb18;
&#xd658;&#xd569;&#xb2c8;&#xb2e4;,&#xd30c;&#xb77c;&#xbbbf8;&#xd130;&#xac00;&#xc788;&#xb294;&#xbcf0;&#xc5d0;
&#xb2e8;&#xc21c;&#xd788;geometry_columns &#xd14c;&#xc774;&#xbe14;&#xb85c;&#xc0bd;&#xc785;&#xb41c;&#xd589;
&#xac1c;&#xc218;&#xb97c;&#xb18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

```
&#xc720;&#xd6a8;&#xbcf0;&#xc804;: 1.4.0
```

```
&#xbcc0;&#xacbd;&#xc0ac;&#xd56d;: 2.0.0 &#xbcf0;&#xc804;,&#xc774;&#xc81c;&#xb3c4;&#xd615;&#xc720;&#xd615;&#xc81c;
&#xd55c;&#xd558;&#xae30;&#xc704;&#xd574;&#xd655;&#xc778;&#xc81c;&#xc57d;&#xc870;&#xac74;&#xb300;&#xc720;
&#xd615;&#xbcc0;&#xacbd;&#xc790;&#xb97c;&#xc774;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;,&#xc0c8;use_typecode
&#xb97c;&#xc704;&#xc9d3;&#xc73c;&#xb85c;&#xc124;&#xc815;&#xd558;&#xba74;,&#xd655;&#xc778;&#xc81c;&#xc57d;&#xc2b5;
&#xc131;&#xc744;&#xacc4;&#xc18d;&#xc4f8;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
&#xac1c;&#xc120;&#xc0ac;&#xd56d;: 2.0.0 &#xbcf0;&#xc804;,&#xc5f4;&#xc0dd;&#xc131;&#xc2dc;&#xc720;&#xd615;
&#xbcc0;&#xacbd;&#xc790;&#xc640;&#xd568;&#xaed8;,&#xb610;&#xb294;&#xd655;&#xc778;&#xc81c;&#xc57d;&#xc870;&#xd568;
&#xaed8;&#xc0dd;&#xc131;&#xd560;&#xc9c0;&#xd1b5;&#xc81c;&#xd560;&#xc218;&#xc788;&#xb294;use_typecode
&#xc758;&#xc120;&#xd0dd;&#xc801;&#xc778;&#xc790;&#xac00;&#xcd94;&#xac00;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예제

```
CREATE TABLE public.myspatial_table(gid serial, geom geometry);
INSERT INTO myspatial_table(geom) VALUES(ST_GeomFromText('LINESTRING(1 2, 3 4)',4326) );
-- &#xc774;&#xc81c;&#xc720;&#xd615;&#xc218;&#xc815;&#xc790;(typ modifier)&#xb97c; ←
&#xc774;&#xc6a9;&#xd560;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.&#xc774; ←
&#xcffc;&#xb9ac;&#xac00;&#xc791;&#xb3d9;&#xd558;&#xb824;&#xba74;,&#xc774; ←
&#xb370;&#xc774;&#xd130;&#xac00;&#xc874;&#xc7ac;&#xd574;&#xc57c;&#xb9cc; ←
&#xd569;&#xb2c8;&#xb2e4;.
```

```
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) |

-- &#xb3c4;&#xd615; &#xc5f4;&#xc774; typmod&#xac00; &#xc544;&#xb2c8;&#xac70;&#xb098; <-
&#xc774;&#xbbf8; &#xc81c;&#xc57d;&#xc870;&#xac74;&#xc744; &#xac00;&#xc9c0;&#xace0; <-
&#xc788;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0;,<-
&#xc81c;&#xc57d;&#xc870;&#xac74;&#xc744; &#xc774;&#xc6a9;&#xd558;&#xb3c4;&#xb85d; <-
&#xb3c4;&#xd615; &#xc5f4;&#xc744; &#xbcc0;&#xacbd;&#xd560; <-
&#xac83;&#xc785;&#xb2c8;&#xb2e4; .
-- &#xc774; &#xcffc;&#xb9ac;&#xac00; &#xc791;&#xb3d9;&#xd558;&#xb824;&#xba74;,<-
&#xb370;&#xc774;&#xd130;&#xac00; &#xc874;&#xc7ac;&#xd574;&#xc57c;&#xb9cc; <-
&#xd569;&#xb2c8;&#xb2e4; .
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)

```

8.2.6 UpdateGeometrySRID

UpdateGeometrySRID — Updates the SRID of all features in a geometry column, and the table metadata.

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);
```

설&#xba85:

```
&#xc9c0;&#xc624;&#xba54;&#xd2b8;&#xb9ac; &#xceec;&#xb7fc; geometry_columns &#xba54;&#xd0c0;&#xb370;&#xc774;&#x
&#xbc0f; srid&#xb97c; &#xb300;&#xc0c1;&#xc73c;&#xb85c; &#xbaa8;&#xb4e0; &#xd53c;&#xcc98;&#xb4e4;&#xc758;
```

SRID를 갱신합니다. 주의. 만약 스키&#xb9
지원되지 않으면 schema-aware postgresql installations 설치본&
current_schema() 사용



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

예제

Insert geometries into roads table with a SRID set already using **EWKT** format:

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

ˤ음은 도로 테이블의 이전 SRID와
상관없이 SRID를 4326으로 변경할 것입&#

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

앞의 예시는 다음 DDL 선언문과 동&#xc77

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

로드 작업 중 투영체를 잘못 (또는
'unknown'으로) 적용했는데 모든 데이터&
한벸에 웹 메르카토르로 변환하&#
싶다면, DDL을 통해 가능합니다. 하&#xc9
PostGIS 관리 함수 가운데 한벸에 이런
작업을 할 수 있는 동일한 함수는
없습니다.

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

관련 정보

UpdateRasterSRID, ST_SetSRID, ST_Transform

8.3 <code>8.3

8.3.1 ST_GeomCollFromText

ST_GeomCollFromText — Creates a GeometryCollection or Multi* geometry from a set of geometries.

Synopsis

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

ST_Collect

Collects geometries into a geometry collection. The result is either a Multi* or a GeometryCollection, depending on whether the input geometries have the same or different types (homogeneous or heterogeneous). The input geometries are left unchanged within the collection.

Variant 1: accepts two input geometries

Variant 2: accepts an array of geometries

Variant 3: aggregate function accepting a rowset of geometries.

**Note**

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

**Note**

ST_Collect and **ST_Union** appear similar, but in fact operate quite differently. ST_Collect aggregates geometries into a collection without changing them in any way. ST_Union geometrically merges geometries where they overlap, and splits linestrings at intersections. It may return single geometries when it dissolves boundaries.

1.4.0 ST_Collect (ST_GeomFromText ('POINT(1 2)'),
ST_GeomFromText ('POINT(-2 3)'));

st_astext

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



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

ST_Collect

Collect 2D points.

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

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

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)
```

Collect curves.

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

-----
st_astext
-----
MULTICURVE(CIRCULARSTRING(220268 150415,220227 150505,220227 150406),
  CIRCULARSTRING(220227 150406,220227 150407,220227 150406))
```

Using an array constructor for a subquery.

Using an array constructor for a subquery.

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

Using an array constructor for values.

```
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))
```

Creating multiple collections by grouping geometries in a table.

Creating multiple collections by grouping geometries in a table.

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

ST_Dump, ST_AsBinary

ST_Dump, ST_AsBinary

8.3.2 ST_LineFromMultiPoint

ST_LineFromMultiPoint — Create a line from a multi-point geometry.

Synopsis

geometry **ST_LineFromMultiPoint**(geometry aMultiPoint);

ST_LineFromMultiPoint

Creates a line from a multi-point geometry. The line is created by connecting the points in the order they appear in the multi-point geometry.

Use **ST_MakeLine** to create lines from Point or LineString inputs.



This function supports 3d and will not drop the z-index.

ST_MakeEnvelope

ST_MakeEnvelope(double precision xmin, double precision ymin, double precision xmax, double precision ymax, integer srid=unknown);

```
-- 3D ST_MakeEnvelope(10, 10, 11, 11, 4326);
SELECT ST_AsEWKT(ST_MakeEnvelope(10, 10, 11, 11, 4326));
-- ST_MakeEnvelope(10, 10, 11, 11, 4326);
LINESTRING(10 10, 11 10, 11 11, 10 11, 10 10)
```

ST_MakeEnvelope

ST_MakeEnvelope, **ST_MakeEnvelope**

8.3.3 ST_MakeEnvelope

ST_MakeEnvelope — **ST_MakeEnvelope**(double precision xmin, double precision ymin, double precision xmax, double precision ymax, integer srid=unknown);

Synopsis

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

ST_MakeEnvelope

ST_MakeEnvelope(double precision xmin, double precision ymin, double precision xmax, double precision ymax, integer srid=unknown);

1.5 **ST_MakeEnvelope**(double precision xmin, double precision ymin, double precision xmax, double precision ymax, integer srid=unknown);

ST_MakeEnvelope(double precision xmin, double precision ymin, double precision xmax, double precision ymax, integer srid=unknown);

ST_MakeEnvelope(double precision xmin, double precision ymin, double precision xmax, double precision ymax, integer srid=unknown);

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

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

ST_MakeEnvelope

ST_MakeEnvelope, **ST_MakeEnvelope**, **ST_MakeEnvelope**, **ST_MakeEnvelope**

8.3.4 ST_MakeLine

ST_MakeLine — Create a LineString from a set of geometries. Other geometry types cause an error.

Synopsis

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

Creates a LineString

Creates a LineString containing the points of Point, MultiPoint, or LineString geometries. Other geometry types cause an error.

Variant 1: accepts two input geometries

Variant 2: accepts an array of geometries

Variant 3: aggregate function accepting a rowset of geometries. To ensure the order of the input geometries use `ORDER BY` in the function call, or a subquery with an `ORDER BY` clause.

Repeated nodes at the beginning of input LineStrings are collapsed to a single point. Repeated points in Point and MultiPoint inputs are not collapsed. **ST_RemoveRepeatedPoints** can be used to collapse repeated points from the output LineString.



This function supports 3d and will not drop the z-index.

2.0.0 ಄전부터 라인스트링 구성 요&#xc18입력을 지원하기 시작했습니다.

2.0.0 ಄전부터 라인스트링 구성 요&#xc18입력을 지원하기 시작했습니다.

1.4.0 ಄전부터 사용할 수 있습니다. 이 ಄전부터 ST_MakeLine가 도형 ର열을 입력벱을 수 있습니다. 더 많은 포더 빨리 처리하기 위해 ST_MakeLine 합산 함수를 개선했습니다.

예시: ର열 ಄전 사용하기

Create a line composed of two points.

```
SELECT ST_MakeLine(ARRAY(SELECT ST_Centroid(the_geom) FROM visit_locations ORDER BY ↵
    visit_time));

-- 3D &#xd3ec;&#xc778;&#xd2b8; 3&#xac1c;&#xb85c; 3D &#xb77c;&#xc778; &#xb9cc;&#xb4e4;&#xae30;
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)
```

주어진 3D 포인트 도형 2개로 정의&#xb418 BOX3D를 생성합니다.

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

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

```
&#xd3ec;&#xc778;&#xd2b8;,&#xba40;&#xd2f0;&#xd3ec;&#xc778;&#xd2b8; &#xb610;&#xb294; &#xb77c;&#xc778; &#xb3c4;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc744; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
```

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

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

예시: 배열 버전 사용하기

Create a line from an array formed by a subquery with ordering.

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

Create a 3D line from an array of 3D points

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

-- 3D &#xd3ec;&#xc778;&#xd2b8; 3&#xac1c;&#xb85c; 3D &#xb77c;&#xc778; &#xb9cc;&#xb4e4;&#xae30;
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)
```

예시: 공간 합산 버전

이 예시는 GPS 포인트 배열을 입력밑 도형 항목이 이동 순서대로의 GPS 포인트들로 이루어진 라인스트이 동 하나당 한 개의 레코드를 생성합니다.

Using aggregate ORDER BY provides a correctly-ordered LineString.

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

Prior to PostgreSQL 9, ordering in a subquery can be used. However, sometimes the query plan may not respect the order of the subquery.

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

참ૠ

[ST_RemoveRepeatedPoints](#), [ST_AsText](#), [ST_GeomFromText](#), [ST_MakePoint](#)

8.3.5 ST_MakePoint

ST_MakePoint — Creates a 2D, 3DZ or 4D Point.

Synopsis

geometry **ST_Point**(float x_lon, float y_lat);

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

geometry **ST_MakePoint**(double precision x, double precision y, double precision z, double precision m);

Notes

ST_Point uses the SRID of the geometry. ST_MakePointM uses the SRID of the geometry. ST_MakePoint uses the SRID of the geometry. ST_MakePoint uses the SRID of the geometry.

Use **ST_MakePointM** to make points with XYM coordinates.

While not OGC-compliant, ST_MakePoint is faster and more precise than **ST_GeomFromText** and **ST_PointFromText**. It is also easier to use for numeric coordinate values.



Note

For geodetic coordinates, X is longitude and Y is latitude



This function supports 3d and will not drop the z-index.

Examples

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

-- WGS84
SELECT ST_SetSRID(ST_MakePoint(-71.1043443253471, 42.3150676015829), 4326);

-- 3D
SELECT ST_MakePoint(1, 2, 1.5);

-- Z
SELECT ST_Z(ST_MakePoint(1, 2, 1.5));
result
-----
1.5
```

See Also

ST_GeomFromText, **ST_PointFromText**, **ST_SetSRID**, **ST_MakePointM**

8.3.6 ST_MakePointM

ST_MakePointM — x, y, z, m

Synopsis

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

설명

x, y **좌표 및 단위를 가진 포인트를 생성합니다.**

Use **ST_MakePoint** to make points with XY, XYZ, or XYZM coordinates.



Note

For geodetic coordinates, X is longitude and Y is latitude

예시



Note

ST_AsEWKT is used for text output because **ST_AsText** does not support M values.

Create point with unknown SRID.

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

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

x, y **좌표 및 단위를 가진 포인트를 생성합니다.**

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

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

Get measure of created point.

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

result
-----
10
```

참고

ST_AsEWKT, ST_MakePoint, ST_SetSRID

8.3.7 ST_MakePolygon

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

Synopsis

geometry **ST_MakePolygon**(geometry linestring);

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

설명

주어진 외부 구조(shell)로 형성된 폴&#xb9a
생성합니다. 입력 도형이 닫힌 라&
합니다.

Variant 1: Accepts one shell LineString.

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



Note

이 함수에 멀티라인스트링을
입력할 수는 없습니다.
라인스트링을 생성하려면
ST_LineMerge 또는 **ST_Dump** 를 이용하십시오.



This function supports 3d and will not drop the z-index.

예시; 배열 버전 사용하기

인코딩된 폴리라인 스트링으로&#xb
라인스트링을 생성합니다.

```
SELECT ST_MLineFromText('MULTILINESTRING((1 2, 3 4), (4 5, 6 7))');
```

Create a Polygon from an open LineString, using **ST_StartPoint** and **ST_AddPoint** to close it.

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

인코딩된 폴리라인 스트링으로&#xb
라인스트링을 생성합니다.

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

```
st_asewkt
```

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

Create a Polygon from a LineString with measures

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

```
st_asewkt
```

```
POLYGONM((75.15 29.53 1,77 29 1,77.6 29.5 2,75.15 29.53 2))
```

예시: 내곽 구조를 가진 외곽 구조
개미 구멍을 가진 도넛을 빌드해을

```
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;
```

Create a set of province boundaries with holes representing lakes. The input is a table of province Polygons/MultiPolygons and a table of water linestrings. Lines forming lakes are determined by using `ST_IsClosed`. The province linework is extracted by using `ST_Boundary`. As required by `ST_MakePolygon`, the boundary is forced to be a single LineString by using `ST_LineMerge`. (However, note that if a province has more than one region or has islands this will produce an invalid polygon.) Using a LEFT JOIN ensures all provinces are included even if they have no lakes.

Note



```
NULL &#xc30;&#xc5f4;&#xc744; ST_MakePolygon&#xc5d0; &#xc785;&#xb825;&#xd558;&#xba74; NULL&#xc744;
&#xc18;&#xd658;&#xd558;&#xae30;      &#xb54c;&#xbb38;&#xc5d0;      CASE      &#xad6c;&#xc870;&#xb97c;
&#xd65c;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
```

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

Another technique is to utilize a correlated subquery and the `ARRAY()` constructor that converts a row set to an array.

```

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;

-- &#xc55e;&#xacfc; &#xb3d9;&#xc77c;&#xd55c; &#xc608;&#xc2dc;&#xc774;&#xc9c0;&#xb9cc;; &#xc0c1;&#xad00; &#xd558;&#xc704; &#xcffc;&#xb9ac;&#xc640;
-- &#xd589;&#xc758; &#xc9d1;&#xd569;&#xc744; &#xbc30;&#xc5f4;&#xb85c; &#xbcc0;&#xd658;&#xd558;&#xb294; PostgreSQL &#xb0b4;&#xc7a5; ARRAY() &#xd568;&#xc218;&#xb97c; &#xd65c;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.

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;

```

ST_BuildArea

ST_Polygon

8.3.8 ST_Point

ST_Point — Creates a Point with X, Y and SRID values.

Synopsis

geometry **ST_Point**(float x_lon, float y_lat);

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

ST_MakePoint

Returns a Point with the given X and Y coordinate values. This is the SQL-MM equivalent for **ST_MakePoint** that takes just X and Y.



Note

For geodetic coordinates, X is longitude and Y is latitude

Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with **ST_SetSRID** to mark the srid on the geometry.



This method implements the SQL/MM specification. SQL-MM 3: 6.1.2

ST_Point

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

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

New in 3.2.0: With SRID specified

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

Pre-PostGIS 3.2 syntax

Pre-PostGIS 3.2 syntax

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

3.2 and on you can include the srid

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

PostgreSQL also provides the `::` short-hand for casting

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

If the point coordinates are not in a geodetic coordinate system (such as WGS84), then they must be reprojected before casting to a geography. In this example a point in Pennsylvania State Plane feet (SRID 2273) is projected to WGS84 (SRID 4326).

```
SELECT CAST(ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326) As geography);
```

Section 4.3, [ST_MakePoint](#), [ST_SetSRID](#), [ST_Transform](#), [ST_Point](#), [ST_Point](#), [ST_Point](#)

Section 4.3, [ST_MakePoint](#), [ST_SetSRID](#), [ST_Transform](#), [ST_Point](#), [ST_Point](#), [ST_Point](#)

8.3.9 ST_Point

ST_Point — Creates a Point with X, Y, Z and SRID values.

Synopsis

geometry **ST_MakePoint**(double precision x, double precision y, double precision z, double precision m);

Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with `ST_SetSRID` to mark the srid on the geometry.

Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with `ST_SetSRID` to mark the srid on the geometry.

Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with `ST_SetSRID` to mark the srid on the geometry.

SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

ST_MakePoint, ST_PointFromText, ST_SetSRID, ST_MakePointM

ST_MakePoint, ST_PointFromText, ST_SetSRID, ST_MakePointM

8.3.10 ST_Point

ST_Point — Creates a Point with X, Y, M and SRID values.

Synopsis

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

설명

주어진 좌표값을 가진 ST_Point를 반환좌표값을 가진 ST_PointM를 반환좌표값을 가진 ST_PointM

Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with ST_SetSRID to mark the srid on the geometry.

예시

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

참고

ST_MakePoint, ST_PointFromText, ST_SetSRID, ST_MakePointM

8.3.11 ST_Point

ST_Point — Creates a Point with X, Y, Z, M and SRID values.

Synopsis

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

설명

주어진 좌표값을 가진 ST_Point를 반환좌표값을 가진 ST_PointM를 반환좌표값을 가진 ST_PointM

Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with ST_SetSRID to mark the srid on the geometry.

예시

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

```
SELECT ST_SetSRID(ST_Point(-71.1043443253471, 42.3150676015829), 4326)
```

ST_MakePoint

ST_MakePoint, **ST_Point**, **ST_Point**, **ST_Point**, **ST_SetSRID**

8.3.12 ST_Polygon

ST_Polygon — Creates a Polygon from a LineString with a specified SRID.

Synopsis

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

ST_Polygon

Returns a polygon built from the given LineString and sets the spatial reference system from the `srid`.

ST_Polygon is similar to **ST_MakePolygon** Variant 1 with the addition of setting the SRID.

, **ST_MakePoint**, **ST_SetSRID**



Note

ST_Polygon is similar to **ST_MakePolygon** Variant 1 with the addition of setting the SRID. **ST_LineMerge** and **ST_Dump**



This method implements the **OGC 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.

ST_Polygon

Create a 2D polygon.

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

-- result --
POLYGON((75 29, 77 29, 77 29, 75 29))
```

Create a 3D polygon.

```
SELECT ST_AsEWKT( ST_Polygon( ST_GeomFromEWKT('LINESTRING(75 29 1, 77 29 2, 77 29 3, 75 29 1)'), 4326) );

-- result --
SRID=4326;POLYGON((75 29 1, 77 29 2, 77 29 3, 75 29 1))
```

ST_Polygon

ST_AsEWKT, **ST_AsText**, **ST_GeomFromEWKT**, **ST_GeomFromText**, **ST_LineMerge**, **ST_MakePolygon**

8.3.13 ST_MakeEnvelope

ST_MakeEnvelope — Creates a rectangular Polygon in **Web Mercator** (SRID:3857) using the **XYZ tile system**.

Synopsis

geometry **ST_MakePoint**(double precision x, double precision y, double precision z, double precision m);

ST_MakeEnvelope

Creates a rectangular Polygon giving the extent of a tile in the **XYZ tile system**. The tile is specified by the zoom level Z and the XY index of the tile in the grid at that level. Can be used to define the tile bounds required by **ST_AsMVTGeom** to convert geometry into the MVT tile coordinate space.

By default, the tile envelope is in the **Web Mercator** coordinate system (SRID:3857) using the standard range of the Web Mercator system (-20037508.342789, 20037508.342789). This is the most common coordinate system used for MVT tiles. The optional **bounds** parameter can be used to generate tiles in any coordinate system. It is a geometry that has the SRID and extent of the "Zoom Level zero" square within which the XYZ tile system is inscribed.

The optional **margin** parameter can be used to expand a tile by the given percentage. E.g. **margin=0.125** expands the tile by 12.5%, which is equivalent to **buffer=512** when the tile extent size is 4096, as used in **ST_AsMVTGeom**. This is useful to create a tile buffer to include data lying outside of the tile's visible area, but whose existence affects the tile rendering. For example, a city name (a point) could be near an edge of a tile, so its label should be rendered on two tiles, even though the point is located in the visible area of just one tile. Using expanded tiles in a query will include the city point in both tiles. Use a negative value to shrink the tile instead. Values less than -0.5 are prohibited because that would eliminate the tile completely. Do not specify a margin when using with **ST_AsMVTGeom**. See the example for **ST_AsMVT**.

ST_MakeEnvelope
 2.0.0 **ST_MakeEnvelope**
 SRID **ST_MakeEnvelope**
 2.1.0 **ST_MakeEnvelope**

ST_MakeEnvelope

```
SELECT ST_AsText( ST_TileEnvelope(2, 1, 1) );

st_astext
-----
POLYGON((-10018754.1713945 0,-10018754.1713945 10018754.1713945,0 10018754.1713945,0 ↵
0,-10018754.1713945 0))

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

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

ST_MakeEnvelope

ST_MakeEnvelope

8.3.14 ST_HexagonGrid

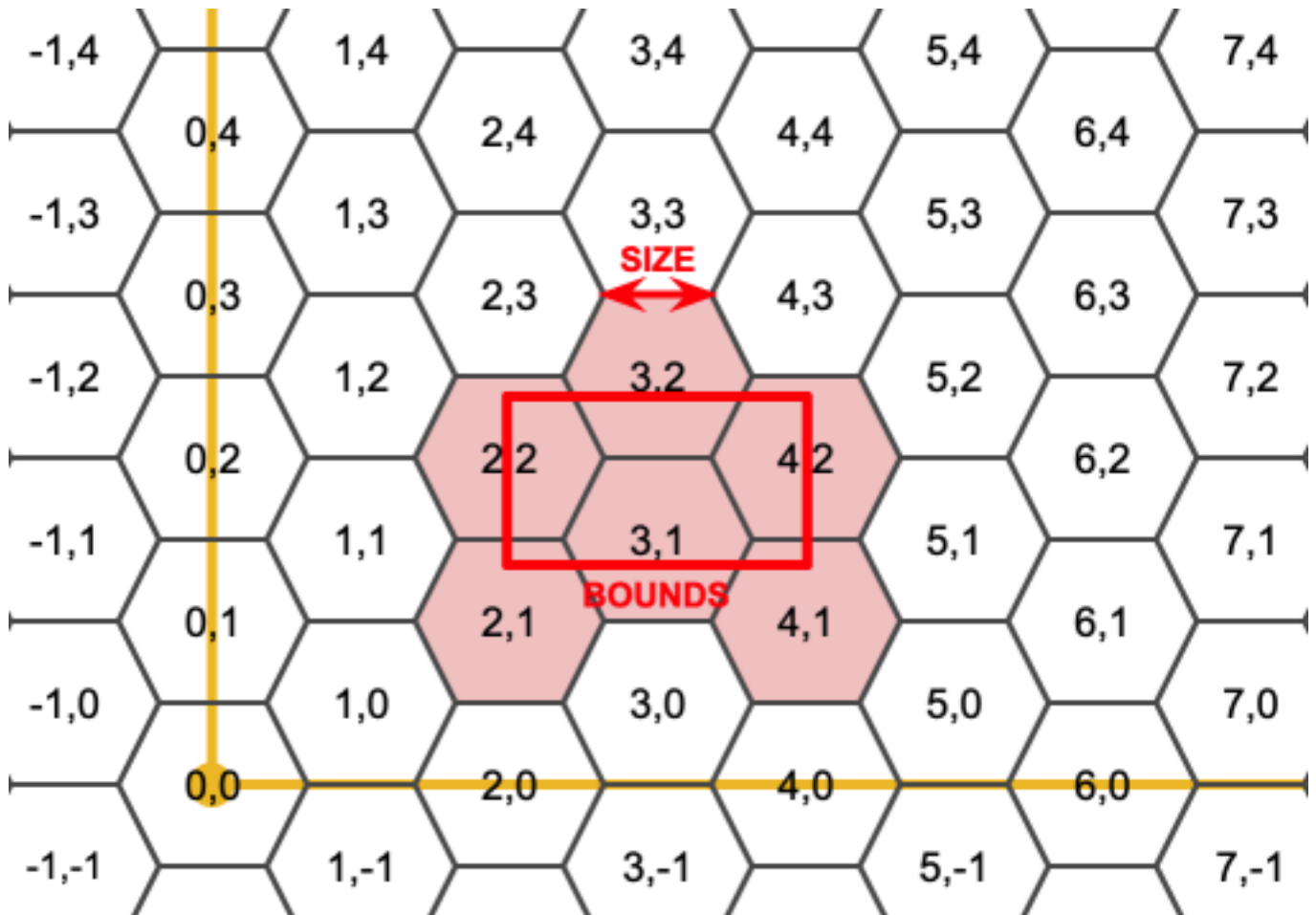
ST_HexagonGrid — Returns a set of hexagons and cell indices that completely cover the bounds of the geometry argument.

Synopsis

geometry **ST_Point**(float x_lon, float y_lat);

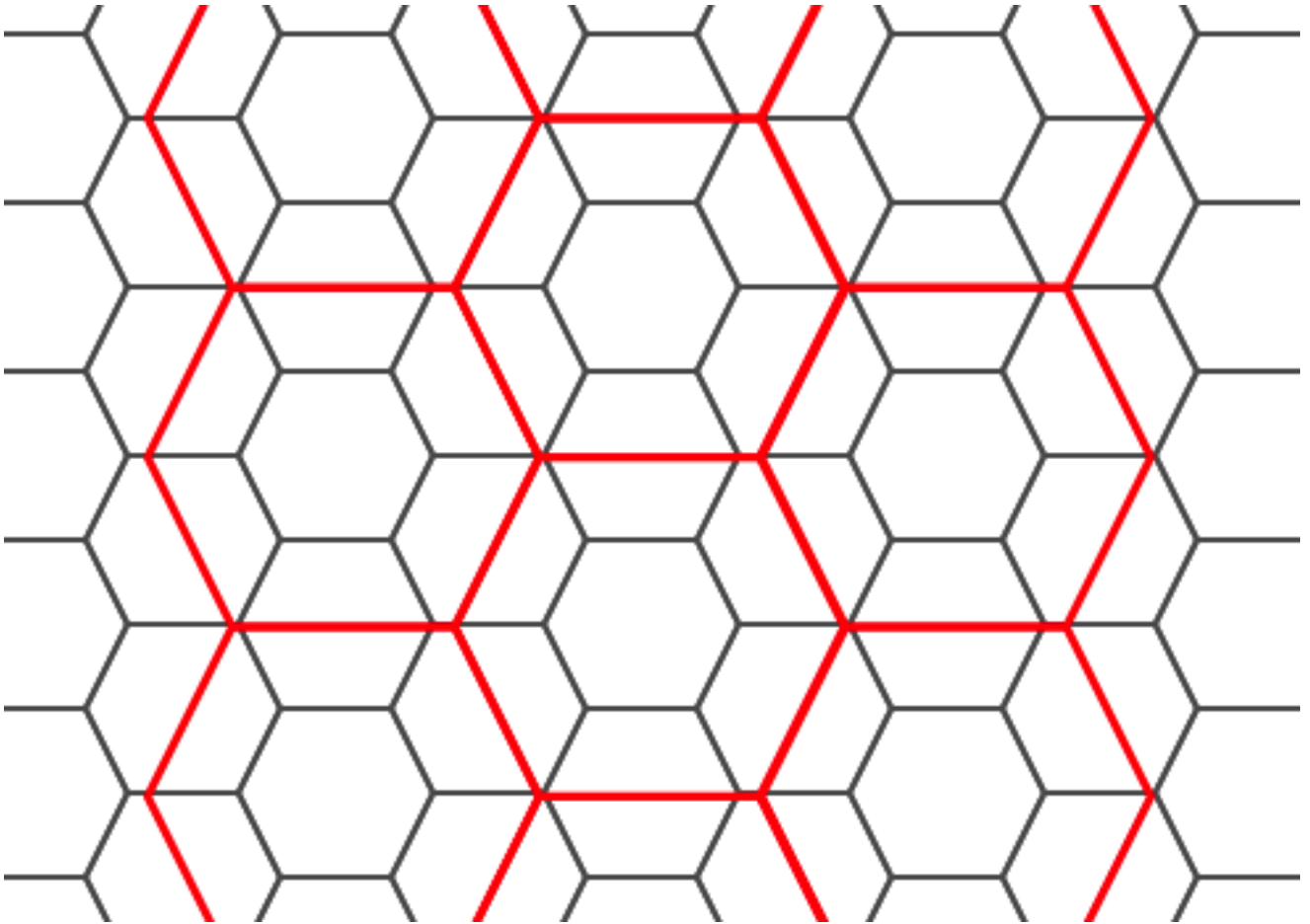
설명

Starts with the concept of a hexagon tiling of the plane. (Not a hexagon tiling of the globe, this is not the **H3** tiling scheme.) For a given planar SRS, and a given edge size, starting at the origin of the SRS, there is one unique hexagonal tiling of the plane, **Tiling**(SRS, Size). This function answers the question: what hexagons in a given **Tiling**(SRS, Size) overlap with a given bounds.



The SRS for the output hexagons is the SRS provided by the bounds geometry.

Doubling or tripling the edge size of the hexagon generates a new parent tiling that fits with the origin tiling. Unfortunately, it is not possible to generate parent hexagon tilings that the child tiles perfectly fit inside.



2.1.0

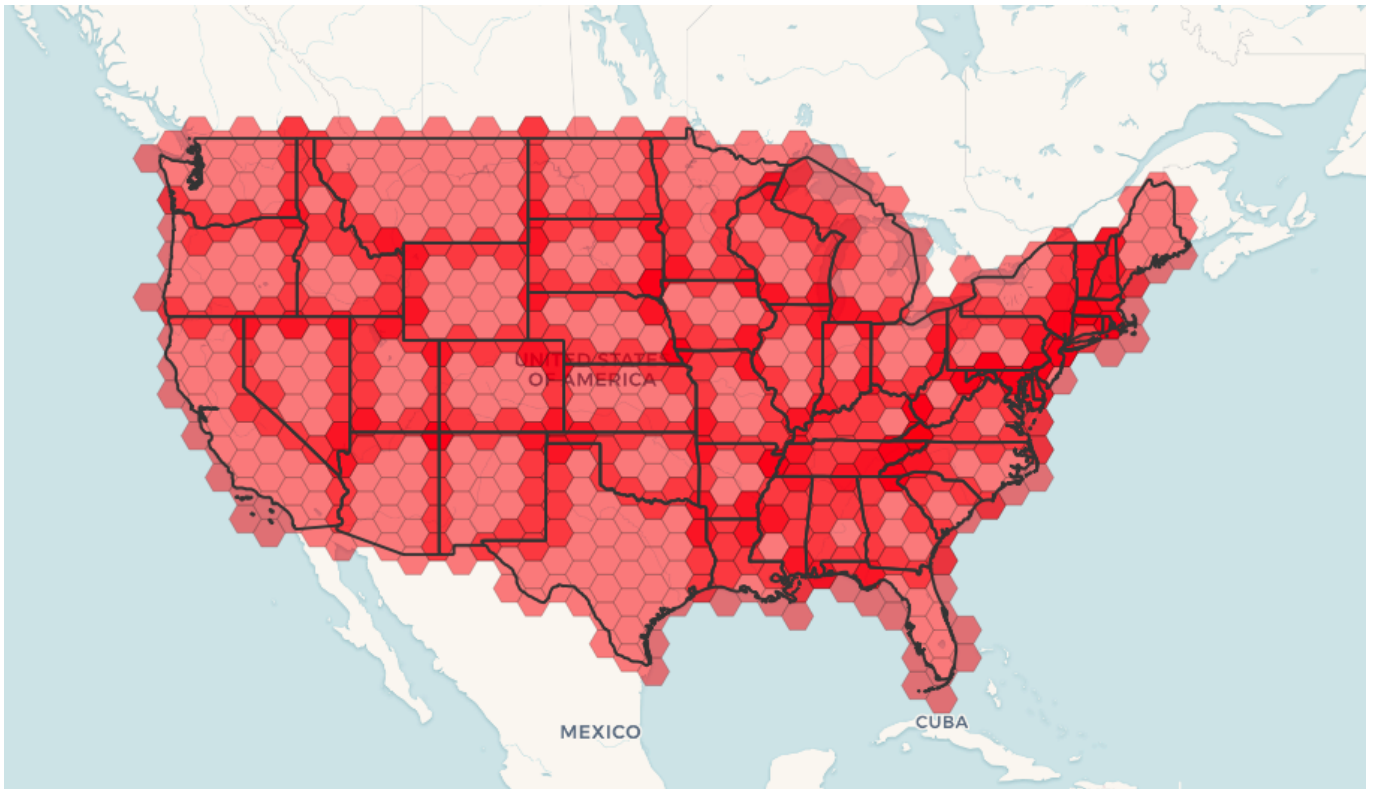
2.1.0

To do a point summary against a hexagonal tiling, generate a hexagon grid using the extent of the points as the bounds, then spatially join to that grid.

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

2.1.0

If we generate a set of hexagons for each polygon boundary and filter out those that do not intersect their hexagons, we end up with a tiling for each polygon.



Tiling states results in a hexagon coverage of each state, and multiple hexagons overlapping at the borders between states.



Note

The LATERAL keyword is implied for set-returning functions when referring to a prior table in the FROM list. So CROSS JOIN LATERAL, CROSS JOIN, or just plain , are equivalent constructs for this example.

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

ST_Hexagon

[ST_EstimatedExtent](#), [ST_MakePoint](#), [ST_Point](#), [ST_SRID](#)

8.3.15 ST_Hexagon

ST_Hexagon — Returns a single hexagon, using the provided edge size and cell coordinate within the hexagon grid space.

Synopsis

geometry **ST_MakePoint**(double precision x, double precision y, double precision z, double precision m);

ST_Hexagon

Uses the same hexagon tiling concept as **ST_HexagonGrid**, but generates just one hexagon at the desired cell coordinate. Optionally, can adjust origin coordinate of the tiling, the default origin is at 0,0.

Hexagons are generated with no SRID set, so use **ST_SetSRID** to set the SRID to the one you expect.

2.1.0 **ST_Hexagon**(float xlon, float ylat, float size, integer srid)

Example: Creating a hexagon at the origin

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

POLYGON((-1 0,-0.5
          -0.866025403784439,0.5
          -0.866025403784439,1
          0,0.5
          0.866025403784439,-0.5
          0.866025403784439,-1 0))
```

ST_MakeEnvelope

ST_MakeEnvelope, **ST_MakePoint**, **ST_SetSRID**

8.3.16 ST_SquareGrid

ST_SquareGrid — Returns a set of grid squares and cell indices that completely cover the bounds of the geometry argument.

Synopsis

geometry **ST_Point**(float xlon, float ylat);

ST_SquareGrid

Starts with the concept of a square tiling of the plane. For a given planar SRS, and a given edge size, starting at the origin of the SRS, there is one unique square tiling of the plane, **Tiling(SRS, Size)**. This function answers the question: what grids in a given **Tiling(SRS, Size)** overlap with a given bounds.

The SRS for the output squares is the SRS provided by the bounds geometry.

Doubling or edge size of the square generates a new parent tiling that perfectly fits with the original tiling. Standard web map tilings in mercator are just powers-of-two square grids in the mercator plane.

2.1.0 **ST_SquareGrid**(float xlon, float ylat, float size, integer srid)

ST_SquareGrid(float xlon, float ylat, float size, integer srid)

The grid will fill the whole bounds of the country, so if you want just squares that touch the country you will have to filter afterwards with **ST_Intersects**.

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

ST_SummaryOverlapsAndTouches

To do a point summary against a square tiling, generate a square grid using the extent of the points as the bounds, then spatially join to that grid. Note the estimated extent might be off from actual extent, so be cautious and at very least make sure you've analyzed your table.

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

ST_SummaryOverlapsAndTouches

This yields the same result as the first example but will be slower for a large number of points

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

ST_Square

[ST_MakeEnvelope](#), [ST_Point](#), [ST_SetSRID](#), [ST_SRID](#)

8.3.17 ST_Square

ST_Square — Returns a single square, using the provided edge size and cell coordinate within the square grid space.

Synopsis

geometry **ST_MakePoint**(double precision x, double precision y, double precision z, double precision m);

ST_Square

Uses the same square tiling concept as [ST_SquareGrid](#), but generates just one square at the desired cell coordinate. Optionally, can adjust origin coordinate of the tiling, the default origin is at 0,0.

Squares are generated with no SRID set, so use [ST_SetSRID](#) to set the SRID to the one you expect.

2.1.0 [ST_SquareGrid](#), [ST_Square](#), [ST_SummaryOverlapsAndTouches](#), [ST_SummaryOverlapsAndTouches](#)

Example: Creating a square at the origin

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

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

ST_MakeEnvelope

ST_MakeLine, **ST_MakePolygon**

8.3.18 ST_Letters

ST_Letters — Returns the input letters rendered as geometry with a default start position at the origin and default text height of 100.

Synopsis

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

ST_Letters

Uses a built-in font to render out a string as a multipolygon geometry. The default text height is 100.0, the distance from the bottom of a descender to the top of a capital. The default start position places the start of the baseline at the origin. Over-riding the font involves passing in a json map, with a character as the key, and base64 encoded TWKB for the font shape, with the fonts having a height of 1000 units from the bottom of the descenders to the tops of the capitals.

The text is generated at the origin by default, so to reposition and resize the text, first apply the **ST_Scale** function and then apply the **ST_Translate** function.

2.1.0 **ST_Letters**(text letters, json font, double height, double width);

ST_Letters(text letters, json font, double height, double width, double dx, double dy);

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



Letters generated by ST_Letters

Example: Scaling and moving words

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

ST_AsTWKB, ST_Scale, ST_Translate

8.4 ST_GeometryAccessor(accessor)**8.4.1 ST_GeometryType(geometry geomA);**

ST_GeometryType(geometry geomA); — ST_GeometryType returns the type of the geometry. The type is one of the following: 'POINT', 'LINESTRING', 'POLYGON', 'MULTIPOINT', 'MULTILINESTRING', 'MULTIPOLYGON', 'GEOMETRYCOLLECTION'. The type is returned as a text string.

Synopsis

text **GeometryType**(geometry geomA);

ST_GeometryType

ST_GeometryType returns the type of the geometry. The type is one of the following: 'POINT', 'LINESTRING', 'POLYGON', 'MULTIPOINT', 'MULTILINESTRING', 'MULTIPOLYGON', 'GEOMETRYCOLLECTION'. The type is returned as a text string.

OGC Simple Features Implementation Specification for SQL 1.1 - 2.1.1.1 - ST_GeometryType(geometry geomA);

Note

ST_GeometryType returns the type of the geometry. The type is one of the following: 'POINT', 'LINESTRING', 'POLYGON', 'MULTIPOINT', 'MULTILINESTRING', 'MULTIPOLYGON', 'GEOMETRYCOLLECTION'. The type is returned as a text string.

2.0.0 - ST_GeometryType(geometry geomA);



This method implements the **OGC 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).

ST_GeometryType

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

ST_GeometryType

ST_GeometryType

8.4.2 ST_Boundary

ST_Boundary — Returns the boundary of a geometry. The boundary of a geometry is the set of points that form the outer edge of the geometry. The boundary of a point is the point itself. The boundary of a line is the set of points that form the line. The boundary of a polygon is the set of points that form the outer edge of the polygon.

Synopsis

geometry **ST_Boundary**(geometry geomA);

ST_Boundary

ST_Boundary — Returns the boundary of a geometry. The boundary of a geometry is the set of points that form the outer edge of the geometry. The boundary of a point is the point itself. The boundary of a line is the set of points that form the line. The boundary of a polygon is the set of points that form the outer edge of the polygon.

boundary) OGC 3.12.3.2 ˨원이 설명하정의됩니다. 이 함수의 결과가 닫때문에, 즉 위상적(位相的)으로 폐때문에, OGC 사양서 3.12.2 단원에서 설명표현적인 도형 원형(primitive)을 이용해결과 ஔ위를 표현할 수 있습니다. GEOS 모듈로 실행

Note

2.0.0 미만 버전에서 이 함수를 GEOMETRYCOLLECTION과 함૨ 사용하면 예외가 발생했습니다. 2.0.0 이후 버전은 대신 (입력을 지원하지 않는다는 의미의) NULL을 반환합니다.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). OGC SPEC s2.1.1.1



This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.17

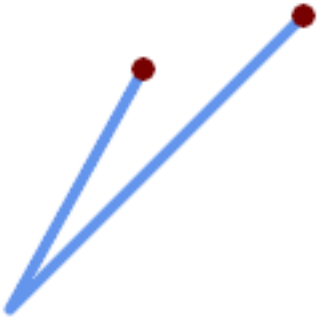
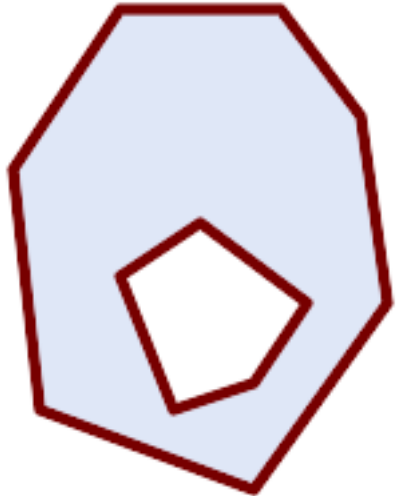


This function supports 3d and will not drop the z-index.

개선 사항 2.1.0 버전부터 삼각형을 지원하기 시작했습니다.

Changed: 3.2.0 support for TIN, does not use geos, does not linearize curves

예시

 <pre> 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)) </pre>	 <pre> 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 \&\#xcd9c;\&\#xb825; 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)) </pre>

```

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))
```

ST_AsText, ST_ExteriorRing, ST_MakePolygon

8.4.3 ST_BoundingDiagonal

ST_BoundingDiagonal — Returns the bounding diagonal of a geometry. The bounding diagonal is the line segment connecting the two vertices of the geometry that are furthest apart. If the geometry is a point, the bounding diagonal is the point itself. If the geometry is a line, the bounding diagonal is the line segment connecting the two endpoints. If the geometry is a polygon, the bounding diagonal is the line segment connecting the two vertices that are furthest apart.

Synopsis

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

ST_BoundingDiagonal

ST_BoundingDiagonal(geom) — Returns the bounding diagonal of a geometry. The bounding diagonal is the line segment connecting the two vertices of the geometry that are furthest apart. If the geometry is a point, the bounding diagonal is the point itself. If the geometry is a line, the bounding diagonal is the line segment connecting the two endpoints. If the geometry is a polygon, the bounding diagonal is the line segment connecting the two vertices that are furthest apart.

fits — If true, the bounding diagonal is returned as a geometry. If false, the bounding diagonal is returned as a text string.

SRID — If not null, the SRID of the returned geometry is set to this value.

Note



ST_BoundingDiagonal(geom) — Returns the bounding diagonal of a geometry. The bounding diagonal is the line segment connecting the two vertices of the geometry that are furthest apart. If the geometry is a point, the bounding diagonal is the point itself. If the geometry is a line, the bounding diagonal is the line segment connecting the two endpoints. If the geometry is a polygon, the bounding diagonal is the line segment connecting the two vertices that are furthest apart.

2.2.0 ST_BoundingDiagonal(geom) — Returns the bounding diagonal of a geometry. The bounding diagonal is the line segment connecting the two vertices of the geometry that are furthest apart. If the geometry is a point, the bounding diagonal is the point itself. If the geometry is a line, the bounding diagonal is the line segment connecting the two endpoints. If the geometry is a polygon, the bounding diagonal is the line segment connecting the two vertices that are furthest apart.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

예시

```
-- &#xd3ec;&#xc778;&#xd2b8;&#xb97c; &#xb458;&#xb7ec;&#xc2fc;  ←
    &#xbc84;&#xd37c;&#xc5d0;&#xc11c; X&#xc758; &#xcd5c;&#xc18c;&#xac12;&#xc744;  ←
    &#xc5bb;&#xc2b5;&#xb2c8;&#xb2e4; .
SELECT ST_X(ST_StartPoint(ST_BoundingDiagonal(
    ST_Buffer(ST_MakePoint(0,0),10)
)));
st_x
-----
-10
```

స고

ST_StartPoint, ST_EndPoint, ST_X, ST_Y, ST_Z, ST_M, &&&

8.4.4 ST_CoordDim

[illegible]

Synopsis

```
integer ST_CoordDim(geometry geomA);
```

설명

ST_Geometry 값의좌표차원을반환합니&#xb2e4
이함수는 MM을준수하는., ST_NDims 와동와함수합니다.

- ✔ This method implements the [OGC 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).

예시

```
SELECT ST_CoordDim('CIRCULARSTRING(1 2 3, 1 3 4, 5 6 7, 8 9 10, 11 12 13)');
--  &#xacb0;&#xacfc; --
3

SELECT ST_CoordDim(ST_Point(1,2));
--  &#xacb0;&#xacfc; --
2
```


다면체 표면, TIN 및 삼각형 예시

```
-- &#xb2e4;&#xba74;&#xccb4; &#xd45c;&#xba74; &#xc608;&#xc2dc;
-- &#xb2e4;&#xba74;&#xccb4; &#xd45c;&#xba74;&#xc744; &#xac01; &#xba74;&#xc73c;&#xb85c;  ←
   &#xbd84;&#xd574;&#xd558;&#xae30;

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 g;
-- &#xacb0;&#xacfc; --
      wkt
-----
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
```

స고

geometry_dump, ST_GeomFromEWKT, ST_Dump, ST_GeometryN, ST_NumGeometries

8.4.7 ST NumPoints

ST_NumPoints — [#b3c4;](#)[#d615;](#)[#c758;](#) [#b0b4;](#)[#c6a9;](#)[#c744;](#) [#c694;](#)[#c57d;](#)[#d55c;](#) [#d14d;](#)[#c2a4;](#)[#x](#)
[#bc18;](#)[#d658;](#)[#d569;](#)[#b2c8;](#)[#b2e4;](#).

Synopsis

```
geometry ST_Points( geometry geom );
```

ST_Dump

A set-returning function (SRF) that extracts the coordinates (vertices) of a geometry. It returns a set of **geometry_dump** rows, each containing a geometry (*geom* field) and an array of integers (*path* field).

- the *geom* field POINTs represent the coordinates of the supplied geometry.
- the *path* field (an integer[]) is an index enumerating the coordinate positions in the elements of the supplied geometry. The indices are 1-based. For example, for a LINESTRING the paths are {i} where i is the nth coordinate in the LINESTRING. For a POLYGON the paths are {i, j} where i is the ring number (1 is outer; inner rings follow) and j is the coordinate position in the ring.

To obtain a single geometry containing the coordinates use **ST_Points**.

Enhanced: 2.1.0 Faster speed. Reimplemented as native-C.

2.0.0 Added support for POLYGONM and POLYGONM with holes. Added support for POLYGONM with holes. Added support for POLYGONM with holes.

1.2.2 Added support for POLYGONM and POLYGONM with holes. Added support for POLYGONM with holes. Added support for POLYGONM with holes.

- ✓ 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.

Classic Explode a Table of LineStrings into nodes

```
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)
```



```
{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)
```

Polyhedral surface cube

```
-- 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)
```

```
-- 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;
-- <table></table> --
      wkt
-----
TRIANGLE((0 0 0,0 1 0,1 1 0,0 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;
-- <table></table> --
      wkt
-----
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))

```

<table></table>

geometry_dump, ST_GeomFromEWKT, ST_Dump, ST_GeometryN, ST_NumGeometries

8.4.8 ST_NumPoints

ST_NumPoints — Returns the number of points in the geometry. If the geometry is a **POINT**, it returns 1. If the geometry is a **LINESTRING**, it returns the number of points in the line. If the geometry is a **POLYGON**, it returns the number of points in the polygon. If the geometry is a **MULTIPOINT**, it returns the number of points in the multi-point. If the geometry is a **MULTILINESTRING**, it returns the number of points in the multi-line. If the geometry is a **MULTIPOLYGON**, it returns the number of points in the multi-polygon. If the geometry is a **GEOMETRYCOLLECTION**, it returns the number of points in the collection.

Synopsis

geometry **ST_NumPoints**(geometry geom);

ST_DumpSegments

A set-returning function (SRF) that extracts the segments of a geometry. It returns a set of **geometry_dump** rows, each containing a geometry (*geom* field) and an array of integers (*path* field).

- **LINESTRING**: the *path* field is an integer array where *i* is the *n*th segment start point in the **LINESTRING**. For a **POLYGON** the paths are *{i, j}* where *i* is the ring number (1 is outer; inner rings follow) and *j* is the segment start point position in the ring.

2.2.0 This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).



This function supports 3d and will not drop the z-index.

ST_AsText

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

path	st_astext
{1,1}	LINESTRING(1 1,3 3)
{1,2}	LINESTRING(3 3,4 4)
{2,1,1}	LINESTRING(5 5,6 6)
{2,1,2}	LINESTRING(6 6,7 7)
{2,1,3}	LINESTRING(7 7,5 5)

(5 rows)

ST_GeomFromEWKT

```
-- 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 g;
-- &#xacc0;&#xaccfc; --
                                wkt
-----
TRIANGLE((0 0 0,0 1 0,1 1 0,0 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 g;
-- &#xacc0;&#xaccfc; --
                                wkt
-----
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))

```

స고

geometry_dump, ST_GeomCollFromText, ST_Dump, ST_NumInteriorRing,

8.4.9 ST_NRings

ST_NRings — Returns a set of `geometry_dump` rows for the exterior and interior rings of a Polygon.

Synopsis

```
geometry ST_ExteriorRing(geometry a_polygon);
```

설명

A set-returning function (SRF) that extracts the rings of a polygon. It returns a set of **geometry_dump** rows, each containing a geometry (*geom* field) and an array of integers (*path* field).

The *geom* field contains each ring as a POLYGON. The *path* field is an integer array of length 1 containing the polygon ring index. The exterior ring (shell) has index 0. The interior rings (holes) have indices of 1 and higher.



Note

```
&#xc774;          &#xd568;&#xc218;&#xb294;          &#xba40;&#xd2f0;&#xd3f4;&#xb9ac;&#xace4;&#xc744;
&#xc785;&#xb825;&#xbc1b;&#xc9c0; &#xbabb; &#xd569;&#xb2c8;&#xb2e4;. &#ba40;&#xd2f0;&#xd3f4;&#xb9ac;&#xace4;
&#xacbd;&#xc6b0;    ST_Dump    &#xd568;&#xc218;&#xc640;    &#xacb0;&#xd569;&#xd574;&#xc11c;
&#xc774;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

Availability: PostGIS 1.1.3. Requires PostgreSQL 7.3 or higher.



This function supports 3d and will not drop the z-index.

General form of query:

General form of query.

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

A polygon with a single hole.

```
SELECT path, ST_AseWKT(geom) As geom
FROM ST_DumpRings (
    ST_GeomFromEWKT('POLYGON((-8149064 5133092 1,-8149064 5132986 1,-8148996 5132839 1,-8148972 5132767 1,-8148958 5132508 1,-8148941 5132466 1,-8148924 5132394 1,-8148903 5132210 1,-8148930 5131967 1,-8148992 5131978 1,-8149237 5132093 1,-8149404 5132211 1,-8149647 5132310 1,-8149757 5132394 1,-8150305 5132788 1,-8149064 5133092 1),
    (-8149362 5132394 1,-8149446 5132501 1,-8149548 5132597 1,-8149695 5132675 1,-8149362 5132394 1))')
    ) as foo;
```

path	geom
{0}	POLYGON((-8149064 5133092 1,-8149064 5132986 1,-8148996 5132839 1,-8148972 5132767 1,-8148958 5132508 1,-8148941 5132466 1,-8148924 5132394 1,-8148903 5132210 1,-8148930 5131967 1,-8148992 5131978 1,-8149237 5132093 1,-8149404 5132211 1,-8149647 5132310 1,-8149757 5132394 1,-8150305 5132788 1,-8149064 5133092 1))
{1}	POLYGON((-8149362 5132394 1,-8149446 5132501 1,-8149548 5132597 1,-8149695 5132675 1,-8149362 5132394 1))

geometry_dump, ST_GeomFromEWKT, ST_Dump, ST_GeometryN, ST_NumGeometries

8.4.10 ST_EndPoint

ST_EndPoint — ST_LineString, ST_CircularString, ST_GeometryN, ST_NumGeometries

Synopsis

geometry **ST_Points**(geometry geom);

LINESTRING, CIRCULARLINESTRING, POINT

LINESTRING, CIRCULARLINESTRING, POINT

```

LINESTRING &#xb610;&#xb294; CIRCULARLINESTRING &#xc774; &#xc544;&#xb2d0; &#xacbd;&#xc6b0; NULL &#xc744;
&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.

```

- ✔ 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

```
&#xbcc0;&#xacbd;      &#xc0ac;&#xd56d;:      2.0  &#xbc84;&#xc804;&#xbd80;&#xd130;  &#xb2e8;&#xc77c;
&#xb3c4;&#xd615;      &#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc744;
&#xc9c0;&#xc6d0;&#xd558;&#xc9c0;      &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;.      PostGIS
&#xc608;&#xc804;      &#xbc84;&#xc804;&#xc774;&#xb77c;&#xba74;      &#xb2e8;&#xc77c;      &#xb77c;&#xc778;
&#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc744;      &#xc785;&#xb825;&#xbc1b;&#xb294;
&#xacbd;&#xc6b0;      &#xc2dc;&#xc791;&#xc810;&#xc744;      &#xbc18;&#xd658;&#xd588;&#xc744;
&#xc81;&#xb2c8;&#xb2e4;.      2.0  &#xbc84;&#xc804;&#xc740;      &#xb2e4;&#xb978;      &#xbaa8;&#xb4e0;
&#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc9c;&#xb7fc;      NULL&#xc744;
&#xbc18;&#xd658;&#xd560;      &#xbfd0;&#xc785;&#xb2c8;&#xb2e4;.      &#xad6c;&#xc2dd;
&#xc2b5;&#xc131;&#xc740;      &#xbb38;&#xc11c;&#xd654;&#xb418;&#xc9c0;      &#xc54a;&#xc740;
&#xae30;&#xb2a5;&#xc774;&#xc9c0;&#xb9cc;, &#xc0ac;&#xc6a9;&#xc790; &#xb370;&#xc774;&#xd130;&#xb97c;
&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc73c;&#xb85c;      &#xc800;&#xc7a5;&#xd588;&#xb2e4;&#xace0;
&#xac00;&#xc815;&#xd55c;      &#xc0ac;&#xc6a9;&#xc790;&#xc758;      &#xacbd;&#xc6b0;      &#xd604;&#xc7ac;
2.0  &#xbc84;&#xc804;&#xc5d0;&#xc11c;      NULL&#xc774;      &#xbc18;&#xd658;&#xb420;      &#xc218;&#xb3c4;
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시

End point of a LineString

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

End point of a non-LineString is NULL

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

End point of a 3D LineString

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

ST_LineString 또는 ST_CircularString 값에 있는 포인트&#xc758
개수를 반환합니다.

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

ST_Envelope

ST_PointN, ST_StartPoint

8.4.11 ST_Envelope

ST_Envelope — Returns the bounding box of the geometry. The bounding box is a rectangle that contains the geometry. The bounding box is defined by the minimum and maximum X and Y coordinates of the geometry. The bounding box is returned as a geometry of the same SRID as the input geometry. The bounding box is returned as a geometry of the same SRID as the input geometry. The bounding box is returned as a geometry of the same SRID as the input geometry.

Synopsis

geometry **ST_Envelope**(geometry g1);

ST_Envelope

ST_Envelope(geometry g1) returns a geometry. The bounding box is a rectangle that contains the geometry. The bounding box is defined by the minimum and maximum X and Y coordinates of the geometry. The bounding box is returned as a geometry of the same SRID as the input geometry. The bounding box is returned as a geometry of the same SRID as the input geometry. The bounding box is returned as a geometry of the same SRID as the input geometry.

ST_Envelope(geometry g1) returns a geometry. The bounding box is a rectangle that contains the geometry. The bounding box is defined by the minimum and maximum X and Y coordinates of the geometry. The bounding box is returned as a geometry of the same SRID as the input geometry. The bounding box is returned as a geometry of the same SRID as the input geometry. The bounding box is returned as a geometry of the same SRID as the input geometry.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.1



This method implements the SQL/MM specification. SQL-MM 3: 5.1.19

ST_Envelope

```
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
      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))
```

Box2D, Box3D, ST_OrientedEnvelope

Box2D, Box3D, ST_OrientedEnvelope

8.4.12 ST_ExteriorRing

ST_ExteriorRing — Returns the exterior ring of a polygon. If the input is a multi-polygon, it returns the exterior ring of the first polygon in the collection.

Synopsis

geometry **ST_ExteriorRing**(geometry a_polygon);

POLYGON

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



Note

This method implements the **OGC 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.

- ✓ This method implements the **OGC 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.

ST_Dump

```
-- ST_Dump
SELECT gid, ST_ExteriorRing(the_geom) AS ering
FROM sometable;

-- ST_Dump
SELECT gid, ST_ExteriorRing(the_geom) AS ering
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)
```

ST_InteriorRingN

ST_Boundary, **ST_NumInteriorRings**

8.4.13 ST_GeometryN

ST_GeometryN — ST_Geometry

Synopsis

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

설명

입력 도형이 도형 집합, (멀티)포인,인 (멀티)라인스트링, 멀티쳤브(multicurve) 또 (멀티)폴리곤이나 다면체 표면일 경우 1-기반 N번째 도형을 반환하&#xba70 그 외의 경우 NULL을 반환합니다.

Note



0.8.0 버전부터 인덱스는 OGC 사양을 따라 1-기반입니다. 예전 버전은 대신 0-기반 인덱스를 시행했습니다.

Note



모든 도형들을 추출하고자 한다면 ST_Dump 함수가 더 효율적이&#xba70 단일 도형에 대해서도 작동할 것입니다.

개선 사항: 2.0.0 버전부터 다면체 표&#xba70 삼각형 및 TIN을 지원하기 시작했습 변경 사항: 2.0.0 미만 버전은 단일 도형 대해 NULL을 반환했을 겁니다. 2.0.0 버전 ST_GeometryN(...,1) 경우에 대한 도형을 반환하변경됐습니다.



This method implements the **OGC 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).

표준 예시

```
--Extracting a subset of points from a 3d multipoint
SELECT n, ST_AsEWKT(ST_GeometryN(geom, n)) As geomewkt
FROM (
VALUES (ST_GeomFromEWKT('MULTIPOINT((1 2 7), (3 4 7), (5 6 7), (8 9 10))') ),
( ST_GeomFromEWKT('MULTICURVE(CIRCULARSTRING(2.5 2.5,4.5 2.5, 3.5 3.5), (10 11, 12 11))') )
)As foo(geom)
CROSS JOIN generate_series(1,100) n
```

```

WHERE n <= ST_NumGeometries(geom);

n |          geomewkt
---+-----
1 | POINT(1 2 7)
2 | POINT(3 4 7)
3 | POINT(5 6 7)
4 | POINT(8 9 10)
1 | CIRCULARSTRING(2.5 2.5,4.5 2.5,3.5 3.5)
2 | LINESTRING(10 11,12 11)

--Extracting all geometries (useful when you want to assign an id)
SELECT gid, n, ST_GeometryN(geom, n)
FROM sometable CROSS JOIN generate_series(1,100) n
WHERE n <= ST_NumGeometries(geom);

```

다면체 표면, TIN 및 삼각형 예시

```

-- &#xb2e4;&#xba74;&#xccb4; &#xd45c;&#xba74; &#xc608;&#xc2dc;
-- &#xb2e4;&#xba74;&#xccb4; &#xd45c;&#xba74;&#xc744; &#xac01; &#xba74;&#xc73c;&#xb85c; ↔
  &#xbd84;&#xd574;&#xd558;&#xae30;
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;
-- &#xacb0;&#xacfc; --
          wkt
-----
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))

```

స고

ST_Dump, ST_NumGeometries

8.4.14 ST_GeometryType

ST_GeometryType — ST_Geometry 값의도형유형을반환&#xd

Synopsis

```
text ST_GeometryType(geometry g1);
```

설명

```
&#xb3c4;&#xd615;&#xc758;&#xc720;&#xd615;&#xc744;'ST_LineString','ST_Polygon','ST_MultiPolygon'&#xb4f1;&#xacfc;
&#xac19;&#xc740;&#xc2a4;&#xd2b8;&#xb9c1;&#xc73c;&#xb85c;&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.&#xc774;
&#xd568;&#xc218;&#xb294;&#xb3c4;&#xd615;&#xc758;&#xb2e8;&#xc704;&#xac00;&#xc124;&#xc815;&#xb3fc;&#xc788;&#
&#xc54c;&#xb824;&#xc8fc;&#xc9c0;&#xc54a;&#xb294;&#xac83;&#xc740;&#xbb3c;&#xb860;.&#xbc18;&#xd658;&#xb41c;
&#xacb0;&#xacfc;&#xbb3c;&#xc774;&#xc2a4;&#xd2b8;&#xb9c1;&#xc774;&#xba70;&#xc811;&#xb450;&#xc0ac;ST&#xac00;
&#xb2ec;&#xb9b0;&#xb2e4;&#xb294;&#xc810;&#xc5d0;&#xc11c;&#xb3c4;GeometryType(geometry)&#xd568;&#xc218;&#xc6
&#xb2e4;&#xb985;&#xb2c8;&#xb2e4;.
```

&#x1c; Ġ
 항: 2.0.0 버 전 부 터 다 면 체 표 &#ba surface) 을 지 원 합 니 다.



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.

예시

```
SELECT ST_GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
-- &#xacb0;&#xacfc;
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))
) )' ));
-- &#xacb0;&#xacfc;
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)
    ) '));
-- &#xacb0;&#xacfc;
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

```

ST_Tin

ST_Tin

8.4.15 ST_HasArc

ST_HasArc — Tests if a geometry contains a circular arc

Synopsis

boolean **ST_IsEmpty**(geometry geomA);

ST_HasArc

ST_HasArc(geometry geomA, text type) → boolean

1.2.2 **ST_HasArc**(geometry geomA, text type, text type2) → boolean



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

ST_HasArc

```

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

```

ST_HasArc

ST_HasArc

8.4.16 ST_InteriorRingN

ST_InteriorRingN — Returns the *n*th interior ring of a polygon. If *n* is 0, the exterior ring is returned. If *n* is 1, the first interior ring is returned, and so on. If *n* is greater than the number of interior rings, an error is raised.

Synopsis

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

<return type>

ST_InteriorRingN returns a geometry of the same type as the input geometry. If the input geometry is a polygon, the result is a polygon. If the input geometry is a multi-polygon, the result is a multi-polygon. If the input geometry is a geometry collection, the result is a geometry collection. If the input geometry is a point, line, or surface, an error is raised.



Note

ST_InteriorRingN returns the *n*th interior ring of a polygon. If *n* is 0, the exterior ring is returned. If *n* is 1, the first interior ring is returned, and so on. If *n* is greater than the number of interior rings, an error is raised.



This method implements the [OGC 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.

<return type>

```
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
```

<return type>

ST_ExteriorRing, ST_M, ST_X, ST_Y, ST_ZMax, ST_ZMin

8.4.17 ST_IsClosed

ST_IsClosed — Returns true if the geometry is a closed line string. If the geometry is a point, line, or surface, an error is raised.

Synopsis

boolean **ST_IsClosed**(geometry g);

LINESTRING

`LINESTRING(0 0, 1 1)::geometry`;
`TRUE`;
`LINESTRING(0 0, 0 1, 1 1, 0 0)::geometry`;
`TRUE`;
`MULTILINESTRING((0 0, 0 1, 1 1, 0 0), (0 0, 1 1))::geometry`;
`TRUE`;
`POINT(0 0)::geometry`;
`TRUE`;
`MULTIPOINT((0 0), (1 1))::geometry`;
`TRUE`;



This method implements the [OGC 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 `ST_IsClosed(NULL)` returns `TRUE`;
`ST_IsClosed('LINESTRING(0 0, 1 1)::geometry')` returns `TRUE`;
`ST_IsClosed('LINESTRING(0 0, 0 1, 1 1, 0 0)::geometry')` returns `TRUE`;
`ST_IsClosed('MULTILINESTRING((0 0, 0 1, 1 1, 0 0), (0 0, 1 1))::geometry')` returns `TRUE`;
`ST_IsClosed('POINT(0 0)::geometry')` returns `TRUE`;
`ST_IsClosed('MULTIPOINT((0 0), (1 1))::geometry')` returns `TRUE`;



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

`ST_IsClosed('LINESTRING(0 0, 1 1)::geometry')` returns `TRUE`;
`ST_IsClosed('LINESTRING(0 0, 0 1, 1 1, 0 0)::geometry')` returns `TRUE`;
`ST_IsClosed('MULTILINESTRING((0 0, 0 1, 1 1, 0 0), (0 0, 1 1))::geometry')` returns `TRUE`;
`ST_IsClosed('POINT(0 0)::geometry')` returns `TRUE`;
`ST_IsClosed('MULTIPOINT((0 0), (1 1))::geometry')` returns `TRUE`;



This function supports Polyhedral surfaces.

ST_IsClosed

```
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)
```


ST_IsClosed

```
-- ST_IsClosed --
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

-- ST_IsClosed --
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
```

ST_IsRing

ST_IsRing

8.4.18 ST_IsCollection

ST_IsCollection — Returns true if the geometry is a collection of points, lines, or surfaces. Returns false if the geometry is a single point, line, or surface.

Synopsis

boolean **ST_IsCollection**(geometry g);

ST_IsCollection

ST_IsCollection returns true if the geometry is a collection of points, lines, or surfaces. Returns false if the geometry is a single point, line, or surface.

- GEOMETRYCOLLECTION
- MULTI{POINT,POLYGON,LINestring,CURVE,SURFACE}
- COMPOUNDCURVE

Note

This function supports 3d and will not drop the z-index.
 This method supports Circular Strings and Curves

ST_IsCollection

```

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)

```

ST_NumGeometries**ST_NumGeometries****8.4.19 ST_IsEmpty**

ST_IsEmpty — Tests if a geometry is empty.

Synopsis

boolean **ST_IsEmpty**(geometry geomA);

ST_IsEmpty

ST_IsEmpty(geometry) returns a boolean value indicating whether the geometry is empty. TRUE if the geometry is empty, FALSE otherwise. The geometry must be a valid geometry. If the geometry is not a valid geometry, an error is raised. The geometry must be a valid geometry. If the geometry is not a valid geometry, an error is raised.

**Note**

SQL-MM ST_IsEmpty(NULL) returns NULL. PostGIS ST_IsEmpty(NULL) returns NULL.



This method implements the [OGC 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**

PostGIS 2.0.0 ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)') returns NULL. PostGIS 2.0.0 ST_GeomFromText('POLYGON EMPTY') returns NULL. PostGIS 2.0.0 ST_GeomFromText('POLYGON((1 2, 3 4, 5 6, 1 2))') returns false.

ST_IsEmpty

```
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)
```

8.4.20 ST_IsPolygonCCW

ST_IsPolygonCCW — Tests if Polygons have exterior rings oriented counter-clockwise and interior rings oriented clockwise.

Synopsis

boolean **ST_IsPolygonCCW** (geometry geom);

설명

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.

2.2.0 버전부터 사용할 수 있습니다.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

참고

ST_ForcePolygonCW , **ST_ForcePolygonCCW** , **ST_IsPolygonCW**

8.4.21 ST_IsPolygonCW

ST_IsPolygonCW — Tests if Polygons have exterior rings oriented clockwise and interior rings oriented counter-clockwise.

Synopsis

boolean **ST_IsPolygonCW** (geometry geom);

ST_IsPolygonCW

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.

2.2.0 This function supports 3d and will not drop the z-index.



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

ST_IsPolygonCCW

`ST_ForcePolygonCW`, `ST_ForcePolygonCCW`, `ST_IsPolygonCW`

8.4.22 ST_IsRing

`ST_IsRing` — Tests if a LineString is closed and simple.

Synopsis

boolean **ST_IsRing**(geometry g);

ST_IsRing

`LINESTRING`; **ST_IsClosed**(`ST_StartPoint`((g)) ~ `ST_Endpoint`((g)))
ST_IsSimple; (`ST_IsClosed`((g)) ~ `ST_IsClosed`((g)))
 TRUE; `ST_IsClosed`((g)) ~ `ST_IsClosed`((g))



This method implements the [OGC 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 `ST_IsRing`(NULL) returns NULL.
 PostGIS `ST_IsRing`(NULL) returns NULL.

ST_IsSimple

```
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)
```

ST_IsSimple

ST_IsClosed, ST_IsSimple, ST_StartPoint, ST_EndPoint

8.4.23 ST_IsSimple

ST_IsSimple — Returns true if the geometry is simple, that is, if it does not contain any self-intersections or any other topological anomalies that would make it invalid. The function returns true if the geometry is simple, false otherwise.

Synopsis

boolean **ST_IsSimple**(geometry geomA);

ST_IsSimple

ST_IsSimple returns true if the geometry is simple, that is, if it does not contain any self-intersections or any other topological anomalies that would make it invalid. The function returns true if the geometry is simple, false otherwise. The function is implemented using the **OpenGIS Simple Features Implementation Specification for SQL 1.1**, s2.1.1.1. The function is implemented using the **SQL/MM specification**, SQL-MM 3: 5.1.8. This function supports 3d and will not drop the z-index.



Note

SQL-MM 3: 5.1.8.1, **ST_IsSimple** (NULL) returns NULL. The function is implemented using the **OpenGIS Simple Features Implementation Specification for SQL 1.1**, s2.1.1.1. The function is implemented using the **SQL/MM specification**, SQL-MM 3: 5.1.8. This function supports 3d and will not drop the z-index.



This method implements the **OGC 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.

ST_MemSize

ST_GeomFromEWKT, ST_X, ST_Y, ST_Z

8.4.25 ST_MemSize

ST_MemSize — ST_Geometry

Synopsis

integer **ST_NRings**(geometry geomA);

ST_MemSize

ST_Geometry

This complements the PostgreSQL built-in [database object functions](#) `pg_column_size`, `pg_size_pretty`, `pg_relation_size`, `pg_total_relation_size`.

Note



`pg_relation_size` which gives the byte size of a table may return byte size lower than `ST_MemSize`. This is because `pg_relation_size` does not add toasted table contribution and large geometries are stored in TOAST tables.

`pg_total_relation_size` - includes, the table, the toasted tables, and the indexes.

`pg_column_size` returns how much space a geometry would take in a column considering compression, so may be lower than `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).

Changed: 2.2.0 name changed to `ST_MemSize` to follow naming convention.

ST_MemSize

```
--Return how much byte space Boston takes up in our Mass data set
SELECT pg_size_pretty(SUM(ST_MemSize(geom))) as totgeomsum,
pg_size_pretty(SUM(CASE WHEN town = 'BOSTON' THEN ST_MemSize(geom) ELSE 0 END)) As bossum,
CAST(SUM(CASE WHEN town = 'BOSTON' THEN ST_MemSize(geom) ELSE 0 END)*1.00 /
      SUM(ST_MemSize(geom))*100 As numeric(10,2)) As perbos
FROM towns;
```

totgeomsum	bossum	perbos
-----	-----	-----
1522 kB	30 kB	1.99

```
SELECT ST_MemSize(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'));
--
```



```
--What percentage of our table is taken up by just the geometry
SELECT pg_total_relation_size('public.neighborhoods') As fulltable_size, sum(ST_MemSize(
    geom)) As geomsizesize,
sum(ST_MemSize(geom)) * 1.00/pg_total_relation_size('public.neighborhoods') * 100 As pergeom
FROM neighborhoods;
fulltable_size geomsizesize pergeom
-----
262144          96238          36.71188354492187500000
```

8.4.26 ST_NDims

ST_NDims — ST_Geometry dimension

Synopsis

integer **ST_NDims**(geometry g1);

Description

ST_NDims returns the dimension of the geometry. The dimension is the number of coordinates in the geometry. For example, a 2D point has a dimension of 2, a 3D point has a dimension of 3, a 2D line has a dimension of 2, a 3D line has a dimension of 3, a 2D polygon has a dimension of 2, a 3D polygon has a dimension of 3, a 2D surface has a dimension of 2, and a 3D surface has a dimension of 3.



This function supports 3d and will not drop the z-index.

Examples

```
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
```

See Also

[ST_CoordDim](#), [ST_Dimension](#), [ST_GeomFromEWKT](#)

8.4.27 ST_NPoints

ST_NPoints — ST_Geometry number of points

Synopsis

integer **ST_NPoints**(geometry g1);

ST_NPoints

ST_NPoints(geometry geomA) — Returns the number of points in the geometry *geomA*.

ST_NPoints(geometry geomA, integer n) — Returns the *n*th point in the geometry *geomA*.

Note

1.3.4 ST_NPoints(geometry geomA, integer n) — Returns the *n*th point in the geometry *geomA*. The function supports 3D geometries and will not drop the z-index. The function also supports Circular Strings and Curves, and Polyhedral Surfaces.

- ✓ This function supports 3d and will not drop the z-index.
- ✓ This method supports Circular Strings and Curves
- ✓ This function supports Polyhedral surfaces.

ST_NPoints

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

```
--      4
```

```
--      3
```

```
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)'));
```

```
--      4
```

ST_NPoints**ST_NPoints****8.4.28 ST_NRings**

ST_NRings(geometry geomA) — Returns the number of rings in the geometry *geomA*.

Synopsis

integer **ST_NRings**(geometry geomA);

ST_Nrings

ST_Nrings(geometry geom) — Returns the number of rings in the input geometry. If the input geometry is a polygon, it returns the number of interior rings. If the input geometry is a multi-polygon, it returns the number of rings in the union of the polygons. If the input geometry is a point, line, or surface, it returns 0.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

ST_NumInteriorRings

```
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 row)	1	0

ST_NumGeometries**ST_NumInteriorRings****8.4.29 ST_NumGeometries**

ST_NumGeometries(geometry geom) — Returns the number of geometries in the input geometry. If the input geometry is a multi-geometry, it returns the number of geometries. If the input geometry is a single geometry, it returns 1. If the input geometry is NULL, it returns 0.

Synopsis

integer **ST_NumGeometries**(geometry geom);

ST_NumGeometries

ST_NumGeometries(geometry geom) — Returns the number of geometries in the input geometry. If the input geometry is a multi-geometry, it returns the number of geometries. If the input geometry is a single geometry, it returns 1. If the input geometry is NULL, it returns 0.

ST_NumGeometries(geometry geom) — Returns the number of geometries in the input geometry. If the input geometry is a multi-geometry, it returns the number of geometries. If the input geometry is a single geometry, it returns 1. If the input geometry is NULL, it returns 0.



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).

ST_NumGeometries

```
--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
```

ST_GeometryN, ST_Multi

ST_GeometryN, ST_Multi

8.4.30 ST_NumInteriorRings

ST_NumInteriorRings — Returns the number of interior rings (holes) in a polygon. The function returns 0 for a simple polygon and NULL for a multi-part geometry.

Synopsis

integer **ST_NumInteriorRings**(geometry a_polygon);

ST_NumInteriorRings

ST_NumInteriorRings returns the number of interior rings (holes) in a polygon. The function returns 0 for a simple polygon and NULL for a multi-part geometry.



This method implements the SQL/MM specification. SQL-MM 3: 8.2.5

ST_NumInteriorRings returns the number of interior rings (holes) in a polygon. The function returns 0 for a simple polygon and NULL for a multi-part geometry.

ST_NumInteriorRings

```
-- Returns the number of interior rings (holes) in a polygon.
SELECT gid, field1, field2, ST_NumInteriorRings(the_geom) AS numholes
FROM sometable;

-- Returns the number of interior rings (holes) in a polygon.
SELECT gid, field1, field2, ST_NumInteriorRings(the_geom) AS numholes
FROM sometable;
```

```
-- <#xadb8;#xb9ac;#xace0; #xba40;#xd2f0;#xd3f4;#xb9ac;#xace4; ←
    <#xb0b4;#xbd80;#xc5d0; #xc788;#xb294; #xbaa8;#xb4e0; #xb0b4;#xacfd; ←
    <#xace0;#xb9ac;#xc758; #xac1c;#xc218;#xb97c; #xc54c;#xace0;#xc790; ←
    <#xd558;#xb294; #xacbd;#xc6b0;
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;
```

<#xcc38;#xace0;

ST_NumInteriorRing, ST_PointN

8.4.31 ST_NumInteriorRing

ST_NumInteriorRing — <#xb3c4;#xd615; #xc548;#xc5d0; #xc788;#xb294; #xd3f4;#xb9ac;#xace4;#xc758; <#xb0b4;#xacfd; #xace0;#xb9ac;#xc758; #xac1c;#xc218;#xb97c; #xbc18;#xd658;#xd569;#xb2c8;#xb2e4;. ST_NumInteriorRings <#xd568;#xc218;#xc640; #xb3d9;#xc77c;#xd569;#xb2c8;#xb2e4;.

Synopsis

integer **ST_NumInteriorRing**(geometry a_polygon);

<#xcc38;#xace0;

ST_NumInteriorRings, ST_PointN

8.4.32 ST_NumPatches

ST_NumPatches — <#xb2e4;#xba74;#xccb4; #xd45c;#xba74; #xc0c1;#xc5d0; #xc788;#xb294; #xba74;#xc758; #xac1c;#xc218;#xb97c; #xbc18;#xd658;#xd569;#xb2c8;#xb2e4;. <#xb2e4;#xba74;#xccb4; #xb3c4;#xd615;#xc544;#xb2cc; #xacbd;#xc6b0; NULL;#xc744; #xbc18;#xd658;#xd560; #xac83;#xc785;#xb2c8;#xb2e4;.

Synopsis

integer **ST_NumPatches**(geometry g1);

<#xc124;#xba85;

<#xb2e4;#xba74;#xccb4; #xd45c;#xba74; #xc0c1;#xc5d0; #xc788;#xb294; #xba74;#xc758; #xac1c;#xc218;#xb97c; #xbc18;#xd658;#xd569;#xb2c8;#xb2e4;. <#xb2e4;#xba74;#xccb4; #xb3c4;#xd615;#xc774; #xc544;#xb2cc; #xacbd;#xc6b0; NULL;#xc744; #xbc18;#xd658;#xd560; #xac83;#xc785;#xb2c8;#xb2e4;. <#xc774; #xd568;#xc744; ST_NumGeometries;#xc640; #xb3d9;#xc77c;#xd558;#xc9c0;#xb9cc; MM <#xba85;#xba85;#xbc95;#xc744; #xc9c0;#xc6d0;#xd569;#xb2c8;#xb2e4;. MM <#xaddc;#xc57d;#xc744; #xc2e0;#xacbd; #xc4f0;#xc9c0; #xc54a;#xb294;#xb2e4;#xba74; ST_NumGeometries;#xc00; #xb354; #xbe60;#xb985;#xb2c8;#xb2e4;.

2.0.0 <#xbc84;#xc804;#xbd80;#xd130; #xc0ac;#xc6a9;#xd560; #xc218; #xc788;#xc2b5;#xb2c8;#xb2e4;.



This function supports 3d and will not drop the z-index.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM ISO/IEC 13249-3: 8.5



This function supports Polyhedral surfaces.

8.4.34 ST_PatchN

ST_PatchN — ST_Geometry 값의 도형 유형을 ଘ환합을

Synopsis

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

설명

도형이 POLYHEDRALSURFACE, POLYHEDRALSURFACEM인 경우 1-기&#xb1c1
Nஈ째 도형 (면)을 ଘ환합니다. 그
외의 경우 NULL을 ଘ환합니다. 이 함다면체표면에 대해 ST_GeometryN과 동일하답을 ଘ환합니다. ST_GeometryN을 이용하답을 더 빠릅니다.



Note

인덱스는 1-기반입니다.



Note

```
&#xbaa8;&#xb4e0;      &#xb3c4;&#xd615;&#xb4e4;&#xc744;      &#xcd94;&#xcd9c;&#xd558;&#xace0;&#xc790;
&#xd55c;&#xb2e4;&#xba74; ST_Dump &#xd568;&#xc218;&#xac00; &#xb354; &#xd6a8;&#xc728;&#xc801;&#xc785;&#xb2c8
```

2.0.0 버전부터 사용할 수 있습니다.



This method implements the SQL/MM specification. SQL-MM ISO/IEC 13249-3: 8.5



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

예시

```
-- &#xb2e4;&#xba74;&#xcdb4; &#xd45c;&#xba74;&#xc758; &#xb450; &#xbc88;&#xc9f8;  ←
    &#xba74;&#xc744; &#xcd94;&#xcd9c;&#xd569;&#xb2c8;&#xb2e4;.
```

```
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))
```

ST_AsEWKT

ST_GeomFromEWKT, **ST_Dump**, **ST_GeometryN**, **ST_NumGeometries**

8.4.35 ST_PointN

ST_PointN — **ST_LineString** **ST_CircularString** **ST_GeometryN** **ST_NumGeometries**

Synopsis

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

ST_PointN

ST_PointN returns the *n*th point of the *linestring* geometry. If *linestring* is a **ST_LineString**, **ST_CircularString**, or **ST_GeometryN**, the result is a **ST_Point**. If *linestring* is a **ST_Geometry**, the result is a **ST_GeometryN**. The *n*th point is the point at the *n*th vertex of the *linestring*. The first vertex is at index 1. The last vertex is at index **ST_NumPoints**(*linestring*). If *n* is less than 1 or greater than **ST_NumPoints**(*linestring*), the result is NULL.

Note



0.8.0 **ST_PointN** returns the *n*th point of the *linestring* geometry. If *linestring* is a **ST_LineString**, **ST_CircularString**, or **ST_GeometryN**, the result is a **ST_Point**. If *linestring* is a **ST_Geometry**, the result is a **ST_GeometryN**. The *n*th point is the point at the *n*th vertex of the *linestring*. The first vertex is at index 1. The last vertex is at index **ST_NumPoints**(*linestring*). If *n* is less than 1 or greater than **ST_NumPoints**(*linestring*), the result is NULL.

Note



ST_PointN returns the *n*th point of the *linestring* geometry. If *linestring* is a **ST_LineString**, **ST_CircularString**, or **ST_GeometryN**, the result is a **ST_Point**. If *linestring* is a **ST_Geometry**, the result is a **ST_GeometryN**. The *n*th point is the point at the *n*th vertex of the *linestring*. The first vertex is at index 1. The last vertex is at index **ST_NumPoints**(*linestring*). If *n* is less than 1 or greater than **ST_NumPoints**(*linestring*), the result is NULL.

- ✓ This method implements the **OGC 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

2.0.0 곈전부터 단일
 도형 멀티라인스트링을
 지원하지 않습니다. PostGIS
 예전 버전이라면 단일 라인
 멀티라인스트링을 입력받는
 경우 시작점을 반환했을
 겁니다. 2.0.0 버전은 다른 ન든
 멀티라인스트링처럼 NULL을
 반환할 &#bfd0;입니다. 음수
 인덱스(-1이 마지막 포인트를
 이용할 수 있습니다.

예시

```
-- &#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc73c;&#xb85c;&#xbd80;&#xd130;  ←
&#xaa8;&#xb4e0; &#xd3ec;&#xc778;&#xd2b8;&#xb97c;  ←
&#xcd94;&#xcd9c;&#xd569;&#xb2c8;&#xb2e4;.
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)

-- &#xc6d0;&#xd615; &#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1; &#xc608;&#xc2dc;
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)"
```

참고**ST_NPoints**

8.4.36 ST_Points

ST_Points — Returns a MultiPoint containing all the coordinates of a geometry. Duplicate points are preserved, including the start and end points of ring geometries. (If desired, duplicate points can be removed by calling **ST_RemoveRepeatedPoints** on the result).

Synopsis

geometry **ST_Points**(geometry geom);

Examples

Returns a MultiPoint containing all the coordinates of a geometry. Duplicate points are preserved, including the start and end points of ring geometries. (If desired, duplicate points can be removed by calling **ST_RemoveRepeatedPoints** on the result).

To obtain information about the position of each coordinate in the parent geometry use **ST_NumPoints**.

M and Z coordinates are preserved if present.



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.

2.3.0

SQL

```
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))
```

See also

ST_RemoveRepeatedPoints, **ST_PointN**

8.4.37 ST_StartPoint

ST_StartPoint — Returns the first point of a LineString.

Synopsis

geometry **ST_StartPoint**(geometry geomA);

Examples

LINESTRING POINT, CIRCULARLINESTRING, LINESTRING, POINT, LINESTRING, CIRCULARLINESTRING, NULL



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

Enhanced: 3.2.0 returns a point for all geometries. Prior behavior returns NULLs if input was not a LineString.



```

&#xbcc0;&#xacbd;    &#xc0ac;&#xd56d;:    2.0.0    &#xbc84;&#xc804;&#xbd80;&#xd130;    &#xb2e8;&#xc77c;
&#xb3c4;&#xd615;                &#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc744;
&#xc9c0;&#xc6d0;&#xd558;&#xc9c0;                &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;.    PostGIS
&#xc608;&#xc804;    &#xbc84;&#xc804;&#xc774;&#xb77c;&#xba74;    &#xb2e8;&#xc77c;    &#xb77c;&#xc778;
&#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc744;    &#xc785;&#xb825;&#xbc1b;&#xb294;
&#xacbd;&#xc6b0;    &#xc2dc;&#xc791;&#xc810;&#xc744;    &#xbc18;&#xd658;&#xd588;&#xc744;
&#xc81;&#xb2c8;&#xb2e4;.    2.0.0    &#xbc84;&#xc804;&#xc740;    &#xb2e4;&#xb978;    &#xaa8;&#xb4e0;
&#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc98;&#xb7fc;    NULL&#xc744;
&#xbc18;&#xd658;&#xd560;    &#xbfd0;&#xc785;&#xb2c8;&#xb2e4;.    &#xad6c;&#xc2dd;
&#xc2b5;&#xc131;&#xc740;    &#xbb38;&#xc11c;&#xd654;&#xb418;&#xc9c0;    &#xc54a;&#xc740;
&#xae30;&#xb2a5;&#xc774;&#xc9c0;&#xb9cc;,,    &#xc0ac;&#xc6a9;&#xc790;    &#xb370;&#xc774;&#xd130;&#xb97c;
&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc73c;&#xb85c;    &#xc800;&#xc7a5;&#xd588;&#xb2e4;&#xace0;
&#xac00;&#xc815;&#xd55c;    &#xc0ac;&#xc6a9;&#xc790;&#xc758;    &#xacbd;&#xc6b0;    &#xd604;&#xc7ac;
2.0    &#xbc84;&#xc804;&#xc5d0;&#xc11c;    NULL&#xc774;    &#xbc18;&#xd658;&#xb420;    &#xc218;&#xb3c4;
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.

```

예시

Start point of a LineString

```

SELECT ST_AsText(ST_StartPoint('LINESTRING(0 1, 0 2)::geometry'));
st_astext
-----
POINT(0 1)

```

Start point of a non-LineString is NULL

```

SELECT ST_StartPoint('POINT(0 1)::geometry') IS NULL AS is_null;
is_null
-----
t

```

Start point of a 3D LineString

```

SELECT ST_AsEWKT(ST_StartPoint('LINESTRING(0 1 1, 0 2 2)::geometry'));
st_asewkt
-----
POINT(0 1 1)

```

ST_LineString 또는 ST_CircularString 값에 있는 포인트&#xc758
개수를 반환합니다.

```

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

```

참고

ST_EndPoint, ST_PointN

8.4.38 ST_Summary

ST_Summary — Returns a text summary of the geometry and geography.

Synopsis

```
text ST_Summary(geometry g);
text ST_Summary(geography g);
```

Parameters

g: A geometry or geography object.

The summary string is composed of several parts separated by semicolons:

- M**: MBR (Minimum Bounding Rectangle) information.
- Z**: Z-value information.
- B**: Boundary information.
- G**: Geometry type information.
- S**: Surface information.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

1.2.2 Returns a text summary of the geometry and geography.

2.0.0 Returns a text summary of the geometry and geography.

2.1.0 Returns a text summary of the geometry and geography.

2.2.0 Returns a text summary of the geometry and geography.

Examples

```
=# 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;
-----+-----
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)

```

ST_X

[PostGIS_DropBBox](#), [PostGIS_AddBBox](#), [ST_Force3DM](#), [ST_Force3DZ](#), [ST_Force2D](#), [geography](#)
[ST_IsValid](#), [ST_IsValidReason](#), [ST_IsValidDetail](#)

8.4.39 ST_X

ST_X — Returns the X coordinate of a Point.

Synopsis

```
float ST_X(geometry a_point);
```

ST_X

ST_X returns the X coordinate of a point. If the geometry is not a point, an error is raised. If the geometry is a point, the X coordinate is returned. If the geometry is a line, the X coordinate of the first point is returned. If the geometry is a polygon, the X coordinate of the first point is returned. If the geometry is a multi-point, the X coordinate of the first point is returned. If the geometry is a multi-line, the X coordinate of the first point is returned. If the geometry is a multi-polygon, the X coordinate of the first point is returned. If the geometry is a geometry collection, the X coordinate of the first point is returned. If the geometry is a null geometry, null is returned.



Note

To get the minimum and maximum X value of geometry coordinates use the functions [ST_XMin](#) and [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.

ST_X

```

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)

```

స고

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

8.4.40 ST_Y

ST_Y — Returns the Y coordinate of a Point.

Synopsis

```
float ST_Y(geometry a_point);
```

설명

포인트의 Y 좌표를 반환합니다. Y
좌표가 없는 경우 NULL을 반환합니&#xb2
포인트만 입력받을 수 있습니다.



Note

To get the minimum and maximum Y value of geometry coordinates use the functions **ST_YMin** and **ST_YMax**.



This method implements the **OGC 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.

예시

```
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)
```

స고

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

8.4.41 ST Z

ST Z — Returns the Z coordinate of a Point.

SELECT ST_Zmflag

```
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
```

ST_CoordDim, ST_NDims

ST_CoordDim, ST_NDims, ST_Dimension

8.5 ST_AddPoint (editor)

8.5.1 ST_AddPoint

ST_AddPoint — Add a point to a LineString before the index *position* (using a 0-based index). If the *position* parameter is omitted or is -1 the point is appended to the end of the LineString.

Synopsis

geometry **ST_AddPoint**(geometry linestring, geometry point);

geometry **ST_AddPoint**(geometry linestring, geometry point, integer position = -1);

ST_AddPoint

Adds a point to a LineString before the index *position* (using a 0-based index). If the *position* parameter is omitted or is -1 the point is appended to the end of the LineString.

1.1.0 Add a point to a LineString before the index *position* (using a 0-based index). If the *position* parameter is omitted or is -1 the point is appended to the end of the LineString.



This function supports 3d and will not drop the z-index.

SELECT ST_Asewkt

Add a point to the end of a 3D line

```
SELECT ST_Asewkt(ST_AddPoint('LINESTRING(0 0 1, 1 1 1)', ST_MakePoint(1, 2, 3)));
st_asewkt
-----
LINESTRING(0 0 1,1 1 1,1 2 3)
```


Guarantee all lines in a table are closed by adding the start point of each line to the end of the line only for those that are not closed.

```
UPDATE sometable
SET geom = ST_AddPoint(geom, ST_StartPoint(geom))
FROM sometable
WHERE ST_IsClosed(geom) = false;
```

ST_RemovePoint, ST_SetPoint

8.5.2 ST_CollectionExtract

ST_CollectionExtract — Given a geometry collection, returns a multi-geometry containing only elements of a specified type.

Synopsis

```
geometry ST_CollectionExtract(geometry collection);
geometry ST_CollectionExtract(geometry collection, integer type);
```

ST_CollectionExtract

Given a geometry collection, returns a homogeneous multi-geometry.

If the *type* is not specified, returns a multi-geometry containing only geometries of the highest dimension. So polygons are preferred over lines, which are preferred over points.

If the *type* is specified, returns a multi-geometry containing only that type. If there are no sub-geometries of the right type, an EMPTY geometry is returned. Only points, lines and polygons are supported. The type numbers are:

- 1 == POINT
- 2 == LINESTRING
- 3 == POLYGON

For atomic geometry inputs, the geometry is returned unchanged if the input type matches the requested type. Otherwise, the result is an EMPTY geometry of the specified type. If required, these can be converted to multi-geometries using **ST_Multi**.



Warning

MultiPolygon results are not checked for validity. If the polygon components are adjacent or overlapping the result will be invalid. (For example, this can occur when applying this function to an **ST_Split** result.) This situation can be checked with **ST_IsValid** and repaired with **ST_MakeValid**.

1.5.0 – 2.0.0



Note

Prior to 1.5.3 this function returned atomic inputs unchanged, no matter type. In 1.5.3 non-matching single geometries returned a NULL result. In 2.0.0 non-matching single geometries return an EMPTY result of the requested type.

ST_AsText

Extract highest-dimension type:

```
SELECT ST_AsText(ST_CollectionExtract(
    'GEOMETRYCOLLECTION( POINT(0 0), LINESTRING(1 1, 2 2) )');
    st_astext
    -----
    MULTILINESTRING((1 1, 2 2))
```

Extract points (type 1 == POINT):

```
SELECT ST_AsText(ST_CollectionExtract(
    'GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(POINT(0 0)))',
    1));
    st_astext
    -----
    MULTIPOINT((0 0))
```

Extract lines (type 2 == LINESTRING):

```
SELECT ST_AsText(ST_CollectionExtract(
    'GEOMETRYCOLLECTION(GEOMETRYCOLLECTION(LINESTRING(0 0, 1 1)),LINESTRING(2 2, 3 3))',
    2));
    st_astext
    -----
    MULTILINESTRING((0 0, 1 1), (2 2, 3 3))
```

ST_CollectionHomogenize

ST_CollectionHomogenize, **ST_Multi**, **ST_IsValid**, **ST_MakeValid**

8.5.3 ST_CollectionHomogenize

ST_CollectionHomogenize — Returns the simplest representation of a geometry collection.

Synopsis

geometry **ST_CollectionHomogenize**(geometry collection);

ST_CollectionHomogenize

Homogeneous (uniform) collections are returned as the appropriate multi-geometry. Heterogeneous (mixed) collections are flattened into a single GeometryCollection. Collections containing a single atomic element are returned as that element. Atomic geometries are returned unchanged. If required, these can be converted to a multi-geometry using **ST_Multi**.

- Homogeneous (uniform) collections are returned as the appropriate multi-geometry.
- Heterogeneous (mixed) collections are flattened into a single GeometryCollection.
- Collections containing a single atomic element are returned as that element.
- Atomic geometries are returned unchanged. If required, these can be converted to a multi-geometry using **ST_Multi**.

**Warning**

This function does not ensure that the result is valid. In particular, a collection containing adjacent or overlapping Polygons will create an invalid MultiPolygon. This situation can be checked with [ST_IsValid](#) and repaired with [ST_MakeValid](#).

2.0.0

Single-element collection converted to an atomic geometry

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0))'));

st_astext
-----
POINT(0 0)
```

Nested single-element collection converted to an atomic geometry:

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(MULTIPOINT((0 0)))'));

st_astext
-----
POINT(0 0)
```

Collection converted to a multi-geometry:

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0),POINT(1 1))'));

st_astext
-----
MULTIPOINT((0 0),(1 1))
```

Nested heterogeneous collection flattened to a GeometryCollection:

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION(POINT(0 0), GEOMETRYCOLLECTION (
  ( LINESTRING(1 1, 2 2)))')));

st_astext
-----
GEOMETRYCOLLECTION(POINT(0 0),LINESTRING(1 1,2 2))
```

Collection of Polygons converted to an (invalid) MultiPolygon:

```
SELECT ST_AsText(ST_CollectionHomogenize('GEOMETRYCOLLECTION (POLYGON ((10 50, 50 50, 50
  10, 10 50)), POLYGON ((90 50, 90 10, 50 10, 50 50, 90 50)))'));

st_astext
-----
MULTIPOLYGON(((10 50,50 50,50 10,10 50)),((90 50,90 10,50 10,50 50,90 50)))
```

[ST_CollectionExtract](#), [ST_Multi](#), [ST_IsValid](#), [ST_MakeValid](#)

8.5.4 ST_CurveToLine

ST_CurveToLine — Converts a geometry containing curves to a linear geometry.

Synopsis

geometry **ST_CurveToLine**(geometry curveGeom, float tolerance, integer tolerance_type, integer flags);

Converts a CIRCULAR STRING to regular LINESTRING or CURVEPOLYGON to POLYGON or MULTISURFACE to MULTIPOLYGON. Useful for outputting to devices that can't support CIRCULARSTRING geometry types

Converts a given geometry to a linear geometry. Each curved geometry or segment is converted into a linear approximation using the given `tolerance` and options (32 segments per quadrant and no options by default).

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.

Availability: 1.3.0

Enhanced: 2.4.0 added support for max-deviation and max-angle tolerance, and for symmetric output.

Enhanced: 3.0.0 implemented a minimum number of segments per linearized arc to prevent topological collapse.



This method implements the [OGC 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

```
SELECT ST_AsText(ST_CurveToLine(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)')));
```

--Result --

```
LINESTRING(220268 150415,220269.95064912 150416.539364228,220271.823415575 150418.17258804,220273.613787707 150419.895736857,220275.317452352 150421.704659462,220276.930305234 150423.594998003,220278.448460847 150425.562198489,220279.868261823 150427.60152176,220281.186287736 150429.708054909,220282.399363347 150431.876723113,
```

220283.50456625 150434.10230186,220284.499233914 150436.379429536,220285.380970099 ↔
150438.702620341,220286.147650624 150441.066277505,
220286.797428488 150443.464706771,220287.328738321 150445.892130112,220287.740300149 ↔
150448.342699654,
220288.031122486 150450.810511759,220288.200504713 150453.289621251,220288.248038775 ↔
150455.77405574,
220288.173610157 150458.257830005,220287.977398166 150460.734960415,220287.659875492 ↔
150463.199479347,
220287.221807076 150465.64544956,220286.664248262 150468.066978495,220285.988542259 ↔
150470.458232479,220285.196316903 150472.81345077,
220284.289480732 150475.126959442,220283.270218395 150477.39318505,220282.140985384 ↔
150479.606668057,
220280.90450212 150481.762075989,220279.5637474 150483.85421628,220278.12195122 ↔
150485.87804878,
220276.582586992 150487.828697901,220274.949363179 150489.701464356,220273.226214362 ↔
150491.491836488,
220271.417291757 150493.195501133,220269.526953216 150494.808354014,220267.559752731 ↔
150496.326509628,
220265.520429459 150497.746310603,220263.41389631 150499.064336517,220261.245228106 ↔
150500.277412127,
220259.019649359 150501.38261503,220256.742521683 150502.377282695,220254.419330878 ↔
150503.259018879,
220252.055673714 150504.025699404,220249.657244448 150504.675477269,220247.229821107 ↔
150505.206787101,
220244.779251566 150505.61834893,220242.311439461 150505.909171266,220239.832329968 ↔
150506.078553494,
220237.347895479 150506.126087555,220234.864121215 150506.051658938,220232.386990804 ↔
150505.855446946,
220229.922471872 150505.537924272,220227.47650166 150505.099855856,220225.054972724 ↔
150504.542297043,
220222.663718741 150503.86659104,220220.308500449 150503.074365683,
220217.994991777 150502.167529512,220215.72876617 150501.148267175,
220213.515283163 150500.019034164,220211.35987523 150498.7825509,
220209.267734939 150497.441796181,220207.243902439 150496,
220205.293253319 150494.460635772,220203.420486864 150492.82741196,220201.630114732 ↔
150491.104263143,
220199.926450087 150489.295340538,220198.313597205 150487.405001997,220196.795441592 ↔
150485.437801511,
220195.375640616 150483.39847824,220194.057614703 150481.291945091,220192.844539092 ↔
150479.123276887,220191.739336189 150476.89769814,
220190.744668525 150474.620570464,220189.86293234 150472.297379659,220189.096251815 ↔
150469.933722495,
220188.446473951 150467.535293229,220187.915164118 150465.107869888,220187.50360229 ↔
150462.657300346,
220187.212779953 150460.189488241,220187.043397726 150457.710378749,220186.995863664 ↔
150455.22594426,
220187.070292282 150452.742169995,220187.266504273 150450.265039585,220187.584026947 ↔
150447.800520653,
220188.022095363 150445.35455044,220188.579654177 150442.933021505,220189.25536018 ↔
150440.541767521,
220190.047585536 150438.18654923,220190.954421707 150435.873040558,220191.973684044 ↔
150433.60681495,
220193.102917055 150431.393331943,220194.339400319 150429.237924011,220195.680155039 ↔
150427.14578372,220197.12195122 150425.12195122,
220198.661315447 150423.171302099,220200.29453926 150421.298535644,220202.017688077 ↔
150419.508163512,220203.826610682 150417.804498867,
220205.716949223 150416.191645986,220207.684149708 150414.673490372,220209.72347298 ↔
150413.253689397,220211.830006129 150411.935663483,
220213.998674333 150410.722587873,220216.22425308 150409.61738497,220218.501380756 ↔
150408.622717305,220220.824571561 150407.740981121,
220223.188228725 150406.974300596,220225.586657991 150406.324522731,220227 150406)

```
--3d example
SELECT ST_AsEWKT(ST_CurveToLine(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)')));
Output
-----
LINESTRING(220268 150415 1,220269.95064912 150416.539364228 1.0181172856673,
220271.823415575 150418.17258804 1.03623457133459,220273.613787707 150419.895736857 1.05435185700189,...AD INFINITUM ...
220225.586657991 150406.324522731 1.32611114201132,220227 150406 3)

--use only 2 segments to approximate quarter circle
SELECT ST_AsText(ST_CurveToLine(ST_GeomFromText('CIRCULARSTRING(220268 150415,220227 150505,220227 150406)'),2));
st_astext
-----
LINESTRING(220268 150415,220287.740300149 150448.342699654,220278.12195122 150485.87804878,
220244.779251566 150505.61834893,220207.243902439 150496,220187.50360229 150462.657300346,
220197.12195122 150425.12195122,220227 150406)

-- Ensure approximated line is no further than 20 units away from
-- original curve, and make the result direction-neutral
SELECT ST_AsText(ST_CurveToLine(
  'CIRCULARSTRING(0 0,100 -100,200 0)::geometry,
    20, -- Tolerance
    1, -- Above is max distance between curve and line
    1 -- Symmetric flag
));
st_astext
-----
LINESTRING(0 0,50 -86.6025403784438,150 -86.6025403784439,200 -1.1331077795296e-13,200 0)
```

ST_LineToCurve

ST_LineToCurve

8.5.5 ST_Scroll

ST_Scroll — Change start point of a closed LineString.

Synopsis

geometry **ST_Scroll**(geometry linestring, geometry point);

ST_Scroll

Changes the start/end point of a closed LineString to the given vertex *point*.

Availability: 3.2.0



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

Make a closed line start at its 3rd vertex

Make a closed line start at its 3rd vertex

```
SELECT ST_AsEWKT(ST_Scroll('SRID=4326;LINESTRING(0 0 1, 10 0 2 0, 5 5 4 2,0 0 0 1)', ' ' ←
    POINT(5 5 4 2)'));

st_asewkt
-----
SRID=4326;LINESTRING(5 5 4 2,0 0 0 1,10 0 2 0,5 5 4 2)
```

ST_Normalize

ST_Normalize

8.5.6 ST_FlipCoordinates

ST_FlipCoordinates — Returns a version of a geometry with X and Y axis flipped.

Synopsis

geometry **ST_FlipCoordinates**(geometry geom);

Returns

Returns a version of the given geometry with X and Y axis flipped. Useful for fixing geometries which contain coordinates expressed as latitude/longitude (Y,X).

2.0.0 „€부터 ¬용할 수 있습니다.



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).

Example

```
SELECT ST_AsEWKT(ST_FlipCoordinates(GeomFromEWKT('POINT(1 2)')));

st_asewkt
-----
POINT(2 1)
```

ST_SwapOrdinates

ST_SwapOrdinates

8.5.7 ST_Force2D

ST_Force2D — Returns a 2D geometry from a 3D geometry.

Synopsis

geometry **ST_Force2D**(geometry geomA);

Description

ST_Force2D returns a 2D geometry from a 3D geometry. The 3D geometry is converted to a 2D geometry by dropping the z-index. The 2D geometry is returned as a geometry of the same type as the input geometry. The 2D geometry is returned as a geometry of the same type as the input geometry.

ST_Force2D returns a 2D geometry from a 3D geometry. The 3D geometry is converted to a 2D geometry by dropping the z-index. The 2D geometry is returned as a geometry of the same type as the input geometry. The 2D geometry is returned as a geometry of the same type as the input geometry.

ST_Force2D returns a 2D geometry from a 3D geometry. The 3D geometry is converted to a 2D geometry by dropping the z-index. The 2D geometry is returned as a geometry of the same type as the input geometry. The 2D geometry is returned as a geometry of the same type as the input geometry.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.

Examples

```
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))
```

ST_Force3D

ST_Force3D

8.5.8 ST_Force3D

ST_Force3D — Returns a 3D geometry from a 2D geometry. The 2D geometry is converted to a 3D geometry by adding a z-index. The 3D geometry is returned as a geometry of the same type as the input geometry. The 3D geometry is returned as a geometry of the same type as the input geometry.

Synopsis

geometry **ST_Force3D**(geometry geomA, float Zvalue = 0.0);

Changes

Forces the geometries into XYZ mode. This is an alias for ST_Force3DZ. If a geometry has no Z component, then a *zvalue* Z coordinate is tacked on.

2.0.0: Added support for Polyhedral surfaces (ST_Force3DZ).

2.1.0: Added support for Circular Strings and Curves (ST_Force3DZ).

Changed: 3.1.0. Added support for supplying a non-zero Z value.



This function supports Polyhedral surfaces.



This method supports Circular Strings and Curves



This function supports 3d and will not drop the z-index.

Usage Examples

```
-- 3D Circular String
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))
```

See Also

[ST_AsEWKT](#), [ST_Force2D](#), [ST_Force3DM](#), [ST_Force3DZ](#)

8.5.9 ST_Force3DZ

ST_Force3DZ — Forces the geometries into XYZ mode.

Synopsis

geometry **ST_Force3DZ**(geometry geomA, float Zvalue = 0.0);

설명

Forces the geometries into XYZ mode. If a geometry has no Z component, then a `zvalue` Z coordinate is tacked on.

&#x1c;Ġ¬항: 2.0.0버전부터다면체표&#xba surface)을지원합니다.

```
&#x0acbd;&#xc0ac;&#xd56d;. 2.1.0&#xbbf8;&#xb9cc;&#xbc84;&#xc804.;&#xc989; 2.0.x&#xbc84;&#xc804;&#xae4c;&  
&#xc774;&#xd568;&#xc218;&#xc758;&#xba85;&#xce6d;&#xc740; ST_Force_3DZ&#xc600;&#xc2b5;&#xb2c8;&#xb2e4;.
```

Changed: 3.1.0. Added support for supplying a non-zero Z value.

- ✔ This function supports Polyhedral surfaces.
- ✔ This function supports 3d and will not drop the z-index.
- ✔ This method supports Circular Strings and Curves

예시

```
-- &#xae30;&#xc874; 3D &#xb3c4;&#xd615;&#xc740; &#xbcc0;&#xacbd;&#xb418;&#xc9c0;  ↵
    &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4; .
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))
```

స&#xace0:

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D

8.5.10 ST Force3DM

ST_Force3DM —> XYM

Synopsis

```
geometry ST_Force3DM(geometry geomA, float Mvalue = 0.0);
```

설명

Forces the geometries into XYM mode. If a geometry has no M component, then a *Mvalue* M coordinate is tacked on. If it has a Z component, then Z is removed

```
&#x00000000; &#xbcc0;&#xacbd;&#xc0ac;&#xd56d;; 2.1.0 &#xbbf8;&#xb9cc;&#xbc84;&#xc804;; &#xc989; 2.0.x &#xbc84;&#xc804;&#xae4c;&  
&#xc774;&#xd568;&#xc218;&#xc758;&#xba85;&#xce6d;&#xc740; ST Force 3DM&#xc774;&#xc5c8;&#xc2b5;&#xb2c8;&#xb2
```

Changed: 3.1.0. Added support for supplying a non-zero M value.

-  This method supports Circular Strings and Curves

예시

```
-- &#xae30;&#xc874; 3D &#xb3c4;&#xd615;&#xc740; &#xbcc0;&#xacbd;&#xb418;&#xc9c0; ↔
    &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4; .
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))
```

స고

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

8.5.11 ST_Force4D

ST Force4D — XYZM

Synopsis

```
geometry ST_Force4D(geometry geomA, float Zvalue = 0.0, float Mvalue = 0.0);
```

설&#xba85:

Forces the geometries into XYZM mode. *zvalue* and *mvalue* is tacked on for missing Z and M dimensions, respectively.

[illegible]

Changed: 3.1.0. Added support for supplying non-zero Z and M values.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

예시

```
-- &#xae30;&#xc874; 3D &#xb3c4;&#xd615;&#xc740; &#xbcc0;&#xacbd;&#xb418;&#xc9c0; ↵
   &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4; .
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))
```

ST_Force4D

ST_AsEWKT, **ST_Force2D**, **ST_Force3DM**, **ST_Force3D**

8.5.12 ST_ForcePolygonCCW

ST_ForcePolygonCCW — Orients all exterior rings counter-clockwise and all interior rings clockwise.

Synopsis

geometry **ST_ForcePolygonCCW** (geometry geom);

ST_ForcePolygonCCW

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.

Availability: 2.4.0



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

ST_ForcePolygonCCW

ST_ForcePolygonCW, **ST_IsPolygonCCW**, **ST_IsPolygonCW**

8.5.13 ST_ForceCollection

ST_ForceCollection — Forces all geometries in a collection to be of the same type.

Synopsis

geometry **ST_ForceCollection**(geometry geomA);

ST_ForceCollection

Forces all geometries in a collection to be of the same type. If the collection contains a mix of geometry types, the result will be a MultiGeometry of the highest dimension. For example, if the collection contains a mix of points, lines, and polygons, the result will be a MultiPolygon. If the collection contains a mix of points, lines, and polygons, and the result will be a MultiPolygon. If the collection contains a mix of points, lines, and polygons, and the result will be a MultiPolygon.

Availability: 2.0.0
 surface) and will not drop the z-index.
 surface) and will not drop the z-index.

1.2.2 ಄전부터 사용할 수 있습니다.
1.3.4 미만 ಄전에서 이 함수에 만곡
도형(curve)을 담고 있는 도형을 입력&#xd
충돌이 일어났습니다. 1.3.4 ಄전부&#xd13
이 ಄그가 해ꯀ됐습니다.

곀경 사항; 2.1.0 미만 버전, 즉 2.0.x 버전까&
이 함수의 명칭은 ST_Force_Collection이었습니&



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

예시

```
SELECT ST_AsEWKT(ST_ForceCollection('POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1))'));
st_asewkt
-----
GEOMETRYCOLLECTION(POLYGON((0 0 1,0 5 1,5 0 1,0 0 1),(1 1 1,3 1 1,1 3 1,1 1 1)))

SELECT ST_AsText(ST_ForceCollection('CIRCULARSTRING(220227 150406,2220227 150407,220227 150406)'));
st_astext
-----
GEOMETRYCOLLECTION(CIRCULARSTRING(220227 150406,2220227 150407,220227 150406))
(1 row)
```

```
-- &#xb2e4;&#xba74;&#xccb4; &#xc608;&#xc2dc; --
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))
)
```

స고

ST_AsEWKT, ST_Force2D, ST_Force3DM, ST_Force3D, ST_GeomFromEWKT

8.5.14 ST_ForcePolygonCW

ST_ForcePolygonCW — Orients all exterior rings clockwise and all interior rings counter-clockwise.

Synopsis

geometry **ST_ForcePolygonCW** (geometry geom);

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.

Availability: 2.4.0



This function supports 3d and will not drop the z-index.



This function supports M coordinates.

ST_ForcePolygonCCW , **ST_IsPolygonCCW** , **ST_IsPolygonCW**

8.5.15 ST_ForceSFS

ST_ForceSFS — SFS 1.1

Synopsis

geometry **ST_ForceSFS**(geometry geomA);
 geometry **ST_ForceSFS**(geometry geomA, text version);



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.

8.5.16 ST_ForceRHR

ST_ForceRHR —

Synopsis

geometry **ST_ForceRHR**(geometry g);

ST_ForcePolygonCW

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**.

ST_ForcePolygonCW (geometry, version) → geometry
 2.0.0
 surface)



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

ST_ForcePolygonCW

```
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)
```

ST_ForcePolygonCCW

ST_ForcePolygonCCW , **ST_ForcePolygonCW** , **ST_IsPolygonCCW** , **ST_IsPolygonCW** , **ST_BuildArea**, **ST_Polygonize**, **ST_Reverse**

8.5.17 ST_ForceCurve

ST_ForceCurve (geometry, version) → geometry
 2.0.0
 (upcast)

Synopsis

geometry **ST_ForceCurve**(geometry g);

ST_ForceCurve

ST_ForceCurve (geometry, version) → geometry
 2.0.0
 (compoundcurve), (multisurface)

ST_MakeValid(geometry geomA) — Converts a geometry to a valid geometry. Note that this function will drop the z-index if the input geometry is 2D.

2.2.0 This function supports 3D and will not drop the z-index.



This function supports 3D and will not drop the z-index.



This method supports Circular Strings and Curves

ST_MakeValid

```
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)
```

ST_LineToCurve

ST_LineToCurve

8.5.18 ST_LineToCurve

ST_LineToCurve — Converts a linear geometry to a curved geometry.

Synopsis

geometry **ST_LineToCurve**(geometry geomA Noncircular);

ST_LineToCurve

Converts plain LINestring/POLYGON to CIRCULAR STRINGs and Curved Polygons. Note much fewer points are needed to describe the curved equivalent.



Note

If the input LINestring/POLYGON is not curved enough to clearly represent a curve, the function will return the same input geometry.

Availability: 1.3.0



This function supports 3D and will not drop the z-index.



This method supports Circular Strings and Curves

8.5.19 ST_Multi

ST_Multi — Returns the geometry as a MULTI* geometry collection. If the geometry is already a collection, it is returned unchanged.

Synopsis

geometry **ST_Multi**(geometry geom);

<return type>

Returns the geometry as a MULTI* geometry collection. If the geometry is already a collection, it is returned unchanged.

<example>

```
SELECT ST_AsText(ST_Multi('POLYGON ((10 30, 30 30, 30 10, 10 10, 10 30))'));
           st_astext
-----
MULTIPOLYGON(((10 30,30 30,30 10,10 10,10 30)))
```

<notes>

ST_AsText

8.5.20 ST_Normalize

ST_Normalize — Returns the geometry as a MULTI* geometry collection. If the geometry is already a collection, it is returned unchanged.

Synopsis

geometry **ST_Normalize**(geometry geom);

<return type>

<example>

```
SELECT ST_AsText(ST_Normalize('POLYGON ((10 30, 30 30, 30 10, 10 10, 10 30))'));
           st_astext
-----
MULTIPOLYGON(((10 30,30 30,30 10,10 10,10 30)))
```

2.3.0

ST_AsText

```
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)
```

ST_Equals

ST_Equals

8.5.21 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);

ST_QuantizeCoordinates

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.

Example

```
SELECT ST_AsText(ST_QuantizeCoordinates('POINT (100.123456 0)::geometry, 4));
st_astext
-----
POINT(100.123455047607 0)

WITH test AS (SELECT 'POINT (123.456789123456 123.456789123456)::geometry AS geom)
SELECT
  digits,
  encode(ST_QuantizeCoordinates(geom, digits), 'hex'),
  ST_AsText(ST_QuantizeCoordinates(geom, digits))
FROM test, generate_series(15, -15, -1) AS digits;
```

digits	encode	st_astext
15	01010000005f9a72083cdd5e405f9a72083cdd5e40	POINT(123.456789123456 123.456789123456) ↩
14	01010000005f9a72083cdd5e405f9a72083cdd5e40	POINT(123.456789123456 123.456789123456) ↩
13	01010000005f9a72083cdd5e405f9a72083cdd5e40	POINT(123.456789123456 123.456789123456) ↩
12	01010000005c9a72083cdd5e405c9a72083cdd5e40	POINT(123.456789123456 123.456789123456) ↩
11	0101000000409a72083cdd5e40409a72083cdd5e40	POINT(123.456789123456 123.456789123456) ↩
10	0101000000009a72083cdd5e40009a72083cdd5e40	POINT(123.456789123455 123.456789123455) ↩
9	0101000000009072083cdd5e40009072083cdd5e40	POINT(123.456789123418 123.456789123418) ↩
8	0101000000008072083cdd5e40008072083cdd5e40	POINT(123.45678912336 123.45678912336) ↩
7	0101000000000070083cdd5e40000070083cdd5e40	POINT(123.456789121032 123.456789121032) ↩
6	0101000000000040083cdd5e40000040083cdd5e40	POINT(123.456789076328 123.456789076328) ↩
5	0101000000000000083cdd5e40000000083cdd5e40	POINT(123.456789016724 123.456789016724) ↩
4	0101000000000000003cdd5e40000000003cdd5e40	POINT(123.456787109375 123.456787109375) ↩
3	01010000000000000003cdd5e40000000003cdd5e40	POINT(123.456787109375 123.456787109375) ↩

```

2 | 010100000000000000000038dd5e400000000038dd5e40 | POINT(123.45654296875 123.45654296875)
1 | 0101000000000000000000dd5e4000000000dd5e40 | POINT(123.453125 123.453125)
0 | 0101000000000000000000dc5e4000000000dc5e40 | POINT(123.4375 123.4375)
-1 | 0101000000000000000000c05e4000000000c05e40 | POINT(123 123)
-2 | 010100000000000000000005e400000000000005e40 | POINT(120 120)
-3 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-4 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-5 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-6 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-7 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-8 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-9 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-10 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-11 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-12 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-13 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-14 | 0101000000000000000000058400000000000005840 | POINT(96 96)
-15 | 0101000000000000000000058400000000000005840 | POINT(96 96)

```

ST_SnapToGrid

ST_SnapToGrid

8.5.22 ST_RemovePoint

ST_RemovePoint — Remove a point from a linestring.

Synopsis

geometry **ST_RemovePoint**(geometry linestring, integer offset);

ST_RemovePoint

Removes a point from a LineString, given its index (0-based). Useful for turning a closed line (ring) into an open linestring.

Enhanced: 3.2.0

1.1.0 **ST_RemovePoint**(geometry linestring, integer offset);



This function supports 3d and will not drop the z-index.

ST_RemovePoint

Guarantees no lines are closed by removing the end point of closed lines (rings). Assumes geom is of type LINESTRING

```

UPDATE sometable
  SET geom = ST_RemovePoint(geom, ST_NPoints(geom) - 1)
FROM sometable
WHERE ST_IsClosed(geom);

```

ST_RemovePoint

ST_AddPoint, ST_NPoints, ST_NumPoints

8.5.23 ST_RemoveRepeatedPoints

ST_RemoveRepeatedPoints — Returns a version of a geometry with duplicate points removed.

Synopsis

geometry **ST_RemoveRepeatedPoints**(geometry geom, float8 tolerance);

Parameters

Returns a version of the given geometry with duplicate consecutive points removed. The function processes only (Multi)LineStrings, (Multi)Polygons and MultiPoints but it can be called with any kind of geometry. Elements of GeometryCollections are processed individually. The endpoints of LineStrings are preserved.

If the *tolerance* parameter is provided, vertices within the tolerance distance of one another are considered to be duplicates.

Enhanced: 3.2.0

2.2.0 [PostGIS 2.2.0](#); [PostGIS 2.2.0](#); [PostGIS 2.2.0](#); [PostGIS 2.2.0](#); [PostGIS 2.2.0](#); [PostGIS 2.2.0](#);



This function supports Polyhedral surfaces.



This function supports 3d and will not drop the z-index.

Examples

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'MULTIPOINT ((1 1), (2 2), (3 3), (2 2))' ));
-----
MULTIPOINT(1 1,2 2,3 3)
```

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'LINESTRING (0 0, 0 0, 1 1, 0 0, 1 1, 2 2)' ));
-----
LINESTRING(0 0,1 1,0 0,1 1,2 2)
```

Example: Collection elements are processed individually.

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'GEOMETRYCOLLECTION (LINESTRING (1 1, 2 2, 2 2, 3 3), POINT (4 4), POINT (4 4), POINT (5 5))' ));
-----
GEOMETRYCOLLECTION(LINESTRING(1 1,2 2,3 3),POINT(4 4),POINT(4 4),POINT(5 5))
```

Example: Repeated point removal with a distance tolerance.

```
SELECT ST_AsText( ST_RemoveRepeatedPoints( 'LINESTRING (0 0, 0 0, 1 1, 5 5, 1 1, 2 2)', 2) ) ←
;
-----
LINESTRING(0 0,5 5,2 2)
```

See Also

[ST_Simplify](#)

8.5.24 ST_Reverse

ST_Reverse — Returns a version of a geometry with the order of points reversed. The function processes only (Multi)LineStrings, (Multi)Polygons and MultiPoints but it can be called with any kind of geometry. Elements of GeometryCollections are processed individually. The endpoints of LineStrings are preserved.

Synopsis

geometry **ST_Reverse**(geometry g1);

Enhanced: 2.4.0

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.

Example

```
SELECT ST_AsText (geom) as line, ST_AsText (ST_Reverse (geom)) As reverseline
FROM
  (SELECT ST_MakeLine (ST_Point (1,2),
                        ST_Point (1,10)) As geom) as foo;
--result
      line      |      reverseline
-----+-----
LINESTRING(1 2,1 10) | LINESTRING(1 10,1 2)
```

8.5.25 ST_Segmentize

ST_Segmentize — Breaks a geometry into segments of a specified maximum length.

Synopsis

geometry **ST_Segmentize**(geometry geom, float max_segment_length);
 geography **ST_Segmentize**(geography geog, float max_segment_length);

Enhanced: 2.4.0

Enhanced: 2.4.0 support for curves was introduced.

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

Enhanced: 2.1.0 Support for 3D geometries was added.

Enhanced: 2.1.0 Support for 3D geometries was added.

`ST_GeomFromText, ST_GeogFromText`
`SELECT ST_Segmentize('LINESTRING(1 2, 3 4)::geometry, 0.5);`
`ST_GeomFromText('MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33), (-45 -33,-46 -32))', 5)`
`ST_GeomFromText('POLYGON((-29 28, -30 40, -29 28))', 10);`

Note

`ST_Segmentize` returns a `MULTILINESTRING` geometry. The first argument is a `LINESTRING` geometry, and the second argument is a float value representing the maximum segment length. The result is a `MULTILINESTRING` geometry where each segment is no longer than the specified length.

ST_Segmentize

```
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)
```

ST_LineSubstring**ST_LineSubstring****8.5.26 ST_SetPoint**

`ST_SetPoint` — Returns a `LINESTRING` geometry with a point inserted at the specified position.

Synopsis

geometry **ST_SetPoint**(geometry linestring, integer zerobasedposition, geometry point);

ST_SetPoint

`ST_SetPoint` returns a `LINESTRING` geometry with a point inserted at the specified position. The first argument is a `LINESTRING` geometry, the second argument is an integer representing the position (starting from 0), and the third argument is a `POINT` geometry.

1.1.0 This function supports 3d and will not drop the z-index.



This function supports 3d and will not drop the z-index.

ST_SetPoint

```
--Change first point in line string from -1 3 to -1 1
SELECT ST_AsText(ST_SetPoint('LINESTRING(-1 2,-1 3)', 0, 'POINT(-1 1)'));
      st_astext
-----
LINESTRING(-1 1,-1 3)

---Change last point in a line string (lets play with 3d linestring this time)
SELECT ST_AsEWKT(ST_SetPoint(foo.geom, ST_NumPoints(foo.geom) - 1, ST_GeomFromEWKT('POINT (-1 1 3)'))
FROM (SELECT ST_GeomFromEWKT('LINESTRING(-1 2 3,-1 3 4, 5 6 7)') As geom) As foo;
      st_asewkt
-----
LINESTRING(-1 2 3,-1 3 4,-1 1 3)

SELECT ST_AsText(ST_SetPoint(g, -3, p))
FROM ST_GeomFromText('LINESTRING(0 0, 1 1, 2 2, 3 3, 4 4)') AS g
      , ST_PointN(g,1) as p;
      st_astext
-----
LINESTRING(0 0,1 1,0 0,3 3,4 4)
```

ST_AddPoint

[ST_AddPoint](#), [ST_NPoints](#), [ST_NumPoints](#), [ST_PointN](#), [ST_RemovePoint](#)

8.5.27 ST_ShiftLongitude

ST_ShiftLongitude — Shifts the longitude coordinates of a geometry between -180..180 and 0..360.

Synopsis

geometry **ST_ShiftLongitude**(geometry geom);

Description

Reads every point/vertex in a geometry, and shifts its longitude coordinate from -180..0 to 180..360 and vice versa if between these ranges. This function is symmetrical so the result is a 0..360 representation of a -180..180 data and a -180..180 representation of a 0..360 data.



Note

This is only useful for data with coordinates in longitude/latitude; e.g. SRID 4326 (WGS 84 geographic)



Warning

1.3.4 미만 버전에서는 버그
때문에멀티포인트를입력받지
못 했습니다. 1.3.4 버전부터
멀티포인트도 입력할 수
있습니다.



This function supports 3d and will not drop the z-index.

జ 선 사 항: 2.0.0 버 전 부 터 다 면 체 표 &#xba surface) 및 TIN 을 지 원 합 니 다.

```
&#xc8fc;&#xc758;; 2.2.0 &#xbbf8;&#xb9cc;&#xbc84;&#xc804;&#xc5d0;&#xc11c;; &#xc774; &#xd568;&#xc218;&#xc758;
&#xba85;&#xce6d;&#xc740; "ST_Shift_Longitude"&#xc600;&#xc2b5;&#xb2c8;&#xb2e4;.
```



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

예시

```
--single point forward transformation
SELECT ST_AsText(ST_ShiftLongitude('SRID=4326;POINT(270 0)::geometry'))

st_astext
-----
POINT(-90 0)


--single point reverse transformation
SELECT ST_AsText(ST_ShiftLongitude('SRID=4326;POINT(-90 0)::geometry'))

st_astext
-----
POINT(270 0)


--for linestrings the functions affects only to the sufficient coordinates
SELECT ST_AsText(ST_ShiftLongitude('SRID=4326;LINESTRING(174 12, 182 13)::geometry'))

st_astext
-----
LINESTRING(174 12,-178 13)
```

స고

ST_WrapX

8.5.28 ST_WrapX

ST WrapX — X값 근처에서 도형을 래핑합&#xb2

Synopsis

geometry **ST_WrapX**(geometry geom, float8 wrap, float8 move);

설명

This function splits the input geometries and then moves every resulting component falling on the right (for negative 'move') or on the left (for positive 'move') of given 'wrap' line in the direction specified by the 'move' parameter, finally re-unioning the pieces together.



Note

이 함수는 경위도 입력물이 한 편에서 다른 편으로 걸쳐 있지 않은 관심 피ಘ를 갖도록 하는 데 유용합니다.

Availability: 2.3.0 requires GEOS



This function supports 3d and will not drop the z-index.

예시

```
-- Move all components of the given geometries whose bounding box
-- falls completely on the left of x=0 to +360
select ST_WrapX(geom, 0, 360);

-- Move all components of the given geometries whose bounding box
-- falls completely on the left of x=-30 to +360
select ST_WrapX(geom, -30, 360);
```

참고

ST_ShiftLongitude

8.5.29 ST_SnapToGrid

ST_SnapToGrid — 입력 도형의 모든 포인트를 정규 그리드로 스냅(snap)시킵니다.

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);


```

-----
LINESTRING(1.112 2.123,4.111 3.237)
--Snap a 4d geometry
SELECT ST_AsEWKT(ST_SnapToGrid(
    ST_GeomFromEWKT('LINESTRING(-1.1115678 2.123 2.3456 1.11111,
        4.111111 3.2374897 3.1234 1.1111, -1.11111112 2.123 2.3456 1.1111112)'),
    ST_GeomFromEWKT('POINT(1.12 2.22 3.2 4.4444)'),
    0.1, 0.1, 0.1, 0.01) );
                                     st_asewkt
-----
LINESTRING(-1.08 2.12 2.3 1.1144,4.12 3.22 3.1 1.1144,-1.08 2.12 2.3 1.1144)

--With a 4d geometry - the ST_SnapToGrid(geom,size) only touches x and y coords but keeps m ←
and z the same
SELECT ST_AsEWKT(ST_SnapToGrid(ST_GeomFromEWKT('LINESTRING(-1.1115678 2.123 3 2.3456,
    4.111111 3.2374897 3.1234 1.1111)'),
    0.01) );
                                     st_asewkt
-----
LINESTRING(-1.11 2.12 3 2.3456,4.11 3.24 3.1234 1.1111)

```

참고

[ST_Snap](#), [ST_AsEWKT](#), [ST_AsText](#), [ST_GeomFromText](#), [ST_GeomFromEWKT](#), [ST_Simplify](#)

8.5.30 ST_Snap

ST_Snap — 입력 도형의 분절 구간과 꼭짓참조 도형의 꼭짓점으로 스냅시&#xc

Synopsis

geometry **ST_Snap**(geometry input, geometry reference, float tolerance);

설명

Snaps the vertices and segments of a geometry to 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.

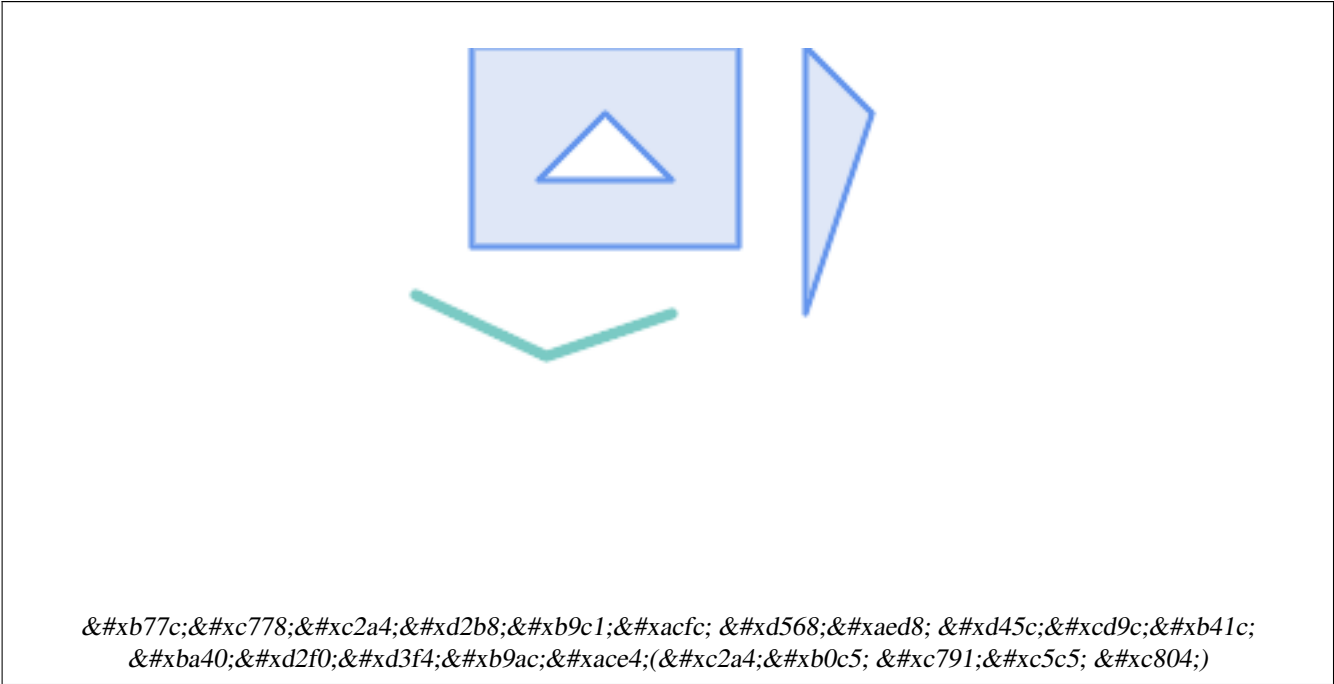
한 도형을 또다른 도형으로 스냅시 노드 및 교차점 계산 과정에서 문일으킬 수 있는) 거의 일치하는 경제거함으로써 오버레이 연산을 강력하게 향상시킬 수 있습니다.

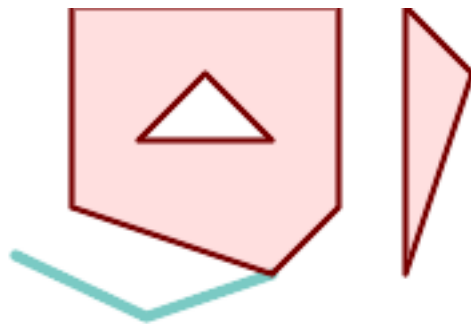
스냅을 너무 많이 하면 유효하지 않은 지형을 생성하는 결과를 가수 있기 때문에., 경험식(heuristics)을 통해 어떤 경우 안전하게 스냅시킬 수 있는지 알아내서 스냅시킬 꼭짓జ수 및 위치를 결정합니다., 하지이런 방식은 몇몇 가능할 수도 있스냅 작업을 제외하게 될 수도 있&#xc



Note
The `ST_IsSimple` function returns `TRUE` if the geometry is simple, meaning it does not contain any self-intersections. The `ST_IsValid` function returns `TRUE` if the geometry is valid, meaning it follows the rules of the geometry type. For example, a line segment is valid but not simple if it intersects itself.

GEOS 2.0.0 and later. The `ST_IsSimple` function is available in the `ST_IsSimple` module. The `ST_IsValid` function is available in the `ST_IsValid` module.



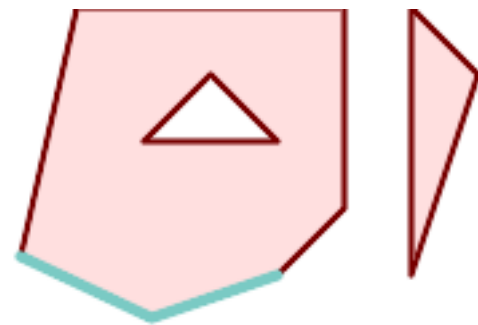


허용 거리 1.01을
 기준으로
 라인스트링에
 스냅된
 멀티폴리곤.
 새로운
 멀티폴리곤은
 참조
 라인스트링과
 함꺍
 표출됩니다.

```
SELECT ST_AsText(ST_Snap(poly,line, ←
      ST_Distance(poly,line)*1.01)) AS polysnap
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)))
```



허용거리 1.25를
기준으로
라인스트링에
스냅된
멀티폴리곤.
새로운
멀티폴리곤은
참조
라인스트링과
함께
표출됩니다.

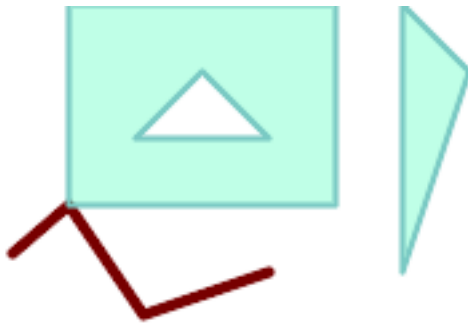
```

SELECT ST_AsText (
    ST_Snap(poly,line, ST_Distance(poly, ↵
ped 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)))
```



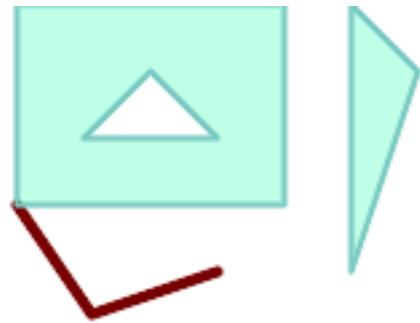
허용 거리 1.01을
 기준으로
 원래
 멀티폴리곤에
 스냅된
 라인스트링.
 새로운
 라인스트링은
 참조
 멀티폴리곤과
 함꺍
 표출됩니다.

```

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)

```



허용 거 리 1.25를
 기준으로
 원래
 멀티폴리곤에
 스냅된
 라인스트링.
 새로운
 라인스트링은
 참조
 멀티폴리곤과
 함께
 표출됩니다.

```
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)
```

స고

ST_SnapToGrid

8.5.31 ST_SwapOrdinates

ST_SwapOrdinates — Swaps the second and third ordinates of a geometry. The second ordinate is the x-axis and the third is the y-axis. The function returns a geometry of the same type as the input geometry.

Synopsis

geometry **ST_SwapOrdinates**(geometry geom, cstring ords);

Parameters

geom: A geometry object.

ords: A string containing the indices of the ordinates to swap. The indices are 1-based and correspond to the x, y, and z axes. For example, '2,3' swaps the y and z axes.

Return Value: A geometry object of the same type as the input geometry, with the ordinates swapped.

- ✓ 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).

Examples

```
-- M&#x2D;2D; 2&#x2D;3D; 3&#x2D;4D; 4&#x2D;5D; 5&#x2D;6D; 6&#x2D;7D; .
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)
```

See Also

[ST_FlipCoordinates](#)

8.6 Geometry Validation

8.6.1 ST_IsValid

ST_IsValid — Tests if a geometry is well-formed in 2D.

Synopsis

boolean **ST_IsValid**(geometry g);
boolean **ST_IsValid**(geometry g, integer flags);

Description

Tests if an ST_Geometry value is well-formed and valid in 2D according to the OGC rules. For geometries with 3 and 4 dimensions, the validity is still only tested in 2 dimensions. For geometries that are invalid, a PostgreSQL NOTICE is emitted providing details of why it is not valid.

For the version with the `flags` parameter, supported values are documented in [ST_IsValidDetail](#) This version does not print a NOTICE explaining invalidity.

For more information on the definition of geometry validity, refer to [Section 4.4](#)



Note
SQL-MM defines the result of ST_IsValid(NULL) to be 0, while PostGIS returns NULL.

Performed by the GEOS module.
The version accepting flags is available starting with 2.0.0.

- ✔ This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#).
- ✔ This method implements the SQL/MM specification. SQL-MM 3: 5.1.9



Note
Neither OGC-SFS nor SQL-MM specifications include a flag argument for ST_IsValid. The flag is a PostGIS extension.

Examples

```
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
```

See Also

[ST_IsSimple](#), [ST_IsValidReason](#), [ST_IsValidDetail](#),

8.6.2 ST_IsValidDetail

ST_IsValidDetail — Returns a `valid_detail` row stating if a geometry is valid or if not a reason and a location.

Synopsis

`valid_detail ST_IsValidDetail(geometry geom, integer flags);`

Description

Returns a `valid_detail` row, containing a boolean (`valid`) stating if a geometry is valid, a varchar (`reason`) stating a reason why it is invalid and a geometry (`location`) pointing out where it is invalid.

Useful to improve on the combination of `ST_IsValid` and `ST_IsValidReason` to generate a detailed report of invalid geometries.

The optional `flags` parameter is a bitfield. It can have the following values:

- 0: Use usual OGC SFS validity semantics.
- 1: Consider certain kinds of self-touching rings (inverted shells and exverted holes) as valid. This is also known as "the ESRI flag", since this is the validity model used by those tools. Note that this is invalid under the OGC model.

Performed by the GEOS module.

Availability: 2.0.0

Examples

```
--First 3 Rejects from a successful quintuplet experiment
SELECT gid, reason(ST_IsValidDetail(geom)), ST_AsText(location(ST_IsValidDetail(geom))) as
    location
FROM
(SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), array_agg(f.line)) As geom, gid
FROM (SELECT ST_Buffer(ST_Point(x1*10,y1), z1) As buff, x1*10 + y1*100 + z1*1000 As gid
      FROM generate_series(-4,6) x1
      CROSS JOIN generate_series(2,5) y1
      CROSS JOIN generate_series(1,8) z1
      WHERE x1 > y1*0.5 AND z1 < x1*y1) As e
      INNER JOIN (SELECT ST_Translate(ST_ExteriorRing(ST_Buffer(ST_Point(x1*10,y1), z1)),
        y1*1, z1*2) As line
      FROM generate_series(-3,6) x1
      CROSS JOIN generate_series(2,5) y1
      CROSS JOIN generate_series(1,10) z1
      WHERE x1 > y1*0.75 AND z1 < x1*y1) As f
ON (ST_Area(e.buff) > 78 AND ST_Contains(e.buff, f.line))
GROUP BY gid, e.buff) As quintuplet_experiment
WHERE ST_IsValid(geom) = false
ORDER BY gid
LIMIT 3;
```

gid	reason	location
5330	Self-intersection	POINT(32 5)
5340	Self-intersection	POINT(42 5)
5350	Self-intersection	POINT(52 5)

```
--simple example
SELECT * FROM ST_IsValidDetail('LINESTRING(220227 150406,220227 150407,22020 150410)');
```

valid	reason	location
t		

See Also

[ST_IsValid](#), [ST_IsValidReason](#)

8.6.3 ST_IsValidReason

ST_IsValidReason — Returns text stating if a geometry is valid, or a reason for invalidity.

Synopsis

```
text ST_IsValidReason(geometry geomA);
text ST_IsValidReason(geometry geomA, integer flags);
```

Description

Returns text stating if a geometry is valid, or if invalid a reason why.

Useful in combination with [ST_IsValid](#) to generate a detailed report of invalid geometries and reasons.

Allowed flags are documented in [ST_IsValidDetail](#).

Performed by the GEOS module.

Availability: 1.4

Availability: 2.0 version taking flags.

Examples

```
-- invalid bow-tie polygon
SELECT ST_IsValidReason(
    'POLYGON ((100 200, 100 100, 200 200,
    200 100, 100 200))'::geometry) as validity_info;
validity_info
-----
Self-intersection[150 150]
```

```
--First 3 Rejects from a successful quintuplet experiment
SELECT gid, ST_IsValidReason(geom) as validity_info
FROM
  (SELECT ST_MakePolygon(ST_ExteriorRing(e.buff), array_agg(f.line)) As geom, gid
  FROM (SELECT ST_Buffer(ST_Point(x1*10,y1), z1) As buff, x1*10 + y1*100 + z1*1000 As gid
        FROM generate_series(-4,6) x1
        CROSS JOIN generate_series(2,5) y1
        CROSS JOIN generate_series(1,8) z1
        WHERE x1 > y1*0.5 AND z1 < x1*y1) As e
        INNER JOIN (SELECT ST_Translate(ST_ExteriorRing(ST_Buffer(ST_Point(x1*10,y1), z1)),
        y1*1, z1*2) As line
        FROM generate_series(-3,6) x1
        CROSS JOIN generate_series(2,5) y1
        CROSS JOIN generate_series(1,10) z1
        WHERE x1 > y1*0.75 AND z1 < x1*y1) As f
  ON (ST_Area(e.buff) > 78 AND ST_Contains(e.buff, f.line))
  GROUP BY gid, e.buff) As quintuplet_experiment
WHERE ST_IsValid(geom) = false
ORDER BY gid
LIMIT 3;

gid |          validity_info
```

```

-----+-----
5330 | Self-intersection [32 5]
5340 | Self-intersection [42 5]
5350 | Self-intersection [52 5]

--simple example
SELECT ST_IsValidReason('LINESTRING(220227 150406,220227 150407,22020 150410)');

st_isvalidreason
-----
Valid Geometry

```

See Also

[ST_IsValid](#), [ST_Summary](#)

8.6.4 ST_MakeValid

ST_MakeValid — Attempts to make an invalid geometry valid without losing vertices.

Synopsis

```

geometry ST_MakeValid(geometry input);
geometry ST_MakeValid(geometry input, text params);

```

Description

The function attempts to create a valid representation of a given invalid geometry without losing any of the input vertices. Valid geometries are returned unchanged.

Supported inputs are: POINTS, MULTIPOINTS, LINESTRINGS, MULTILINESTRINGS, POLYGONS, MULTIPOLYGONS and GEOMETRYCOLLECTIONS containing any mix of them.

In case of full or partial dimensional collapses, the output geometry may be a collection of lower-to-equal dimension geometries, or a geometry of lower dimension.

Single polygons may become multi-geometries in case of self-intersections.

The `params` argument can be used to supply an options string to select the method to use for building valid geometry. The options string is in the format "method=linework|structure keepcollapsed=true|false".

The "method" key has two values.

- "linework" is the original algorithm, and builds valid geometries by first extracting all lines, noding that linework together, then building a value output from the linework.
- "structure" is an algorithm that distinguishes between interior and exterior rings, building new geometry by unioning exterior rings, and then differencing all interior rings.

The "keepcollapsed" key is only valid for the "structure" algorithm, and takes a value of "true" or "false". When set to "false", geometry components that collapse to a lower dimensionality, for example a one-point linestring would be dropped.

Performed by the GEOS module.

Availability: 2.0.0

Enhanced: 2.0.1, speed improvements

Enhanced: 2.1.0, added support for GEOMETRYCOLLECTION and MULTIPOINT.

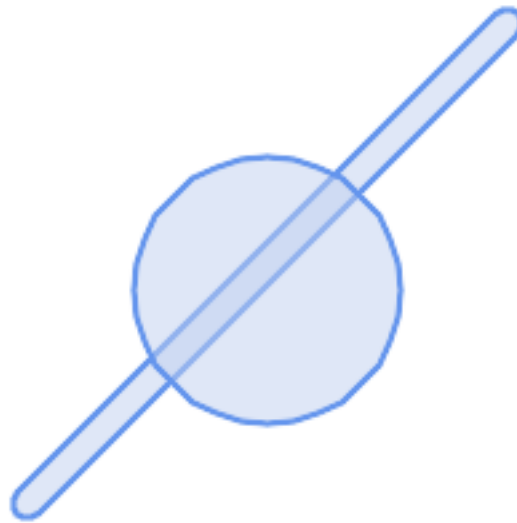
Enhanced: 3.1.0, added removal of Coordinates with NaN values.

Enhanced: 3.2.0, added algorithm options, 'linework' and 'structure' which requires GEOS >= 3.10.0.

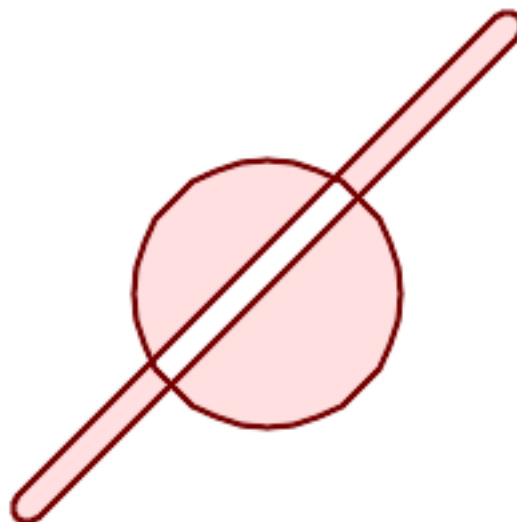


This function supports 3d and will not drop the z-index.

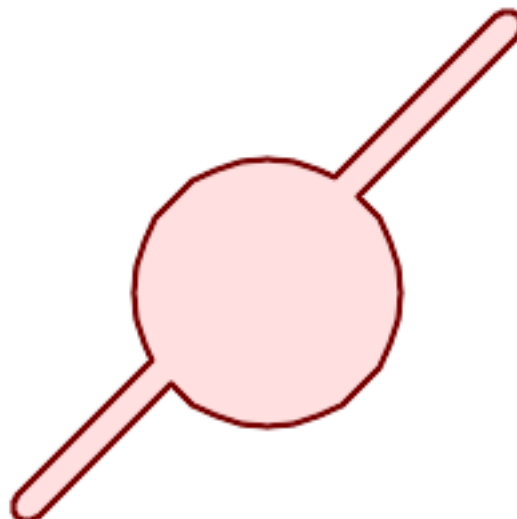
Examples



before_geom: MULTIPOLYGON of 2 overlapping polygons

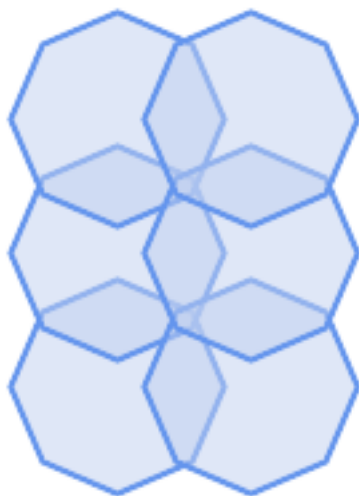


after_geom: MULTIPOLYGON of 4 non-overlapping polygons

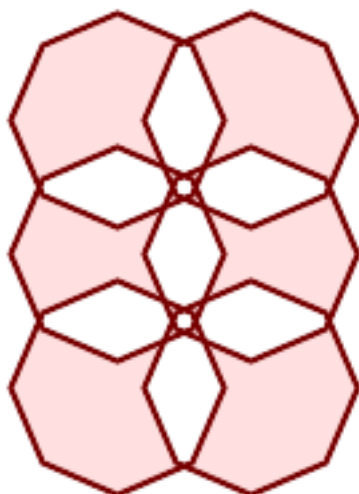


after_geom_structure: MULTIPOLYGON of 1 non-overlapping polygon

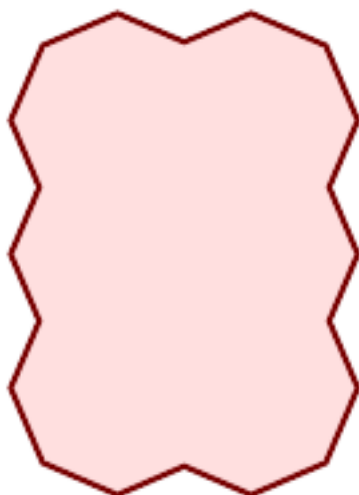
```
SELECT f.geom AS before_geom, ST_MakeValid(f.geom) AS after_geom, ST_MakeValid(f.geom, ←
    'method=structure') AS after_geom_structure
FROM (SELECT 'MULTIPOLYGON(((186 194,187 194,188 195,189 195,190 195,
191 195 192 195 193 194 194 194 194 194 193 195 192 195 191
```

before_geom: MULTIPOLYGON of 6 overlapping polygons



after_geom: MULTIPOLYGON of 14 Non-overlapping polygons



after_geom_structure: MULTIPOLYGON of 1 Non-overlapping polygon

```
SELECT c.geom AS before_geom,
       ST_MakeValid(c.geom) AS after_geom,
       ST_MakeValid(c.geom, 'method=structure') AS after_geom_structure
FROM (SELECT 'MULTIPOLYGON(((91 50,79 22,51 10,23 22,11 50,23 78,51 90,79 78,91 ↵
```

Examples

```
SELECT ST_AsText(ST_MakeValid(
  'LINESTRING(0 0, 0 0)',
  'method=structure keepcollapsed=true'
));

st_astext
-----
POINT(0 0)

SELECT ST_AsText(ST_MakeValid(
  'LINESTRING(0 0, 0 0)',
  'method=structure keepcollapsed=false'
));

st_astext
-----
LINESTRING EMPTY
```

See Also

[ST_IsValid](#), [ST_GeomCollFromText](#), [ST_CollectionExtract](#)

8.7 Spatial Reference System Functions

8.7.1 ST_SetSRID

ST_SetSRID — Set the SRID on a geometry.

Synopsis

geometry **ST_SetSRID**(geometry geom, integer srid);

Description

Sets the SRID on a geometry to a particular integer value. Useful in constructing bounding boxes for queries.



Note

This function does not transform the geometry coordinates in any way - it simply sets the meta data defining the spatial reference system the geometry is assumed to be in. Use [ST_Transform](#) if you want to transform the geometry into a new projection.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#).



This method supports Circular Strings and Curves

Examples

-- Mark a point as WGS 84 long lat --

```
SELECT ST_SetSRID(ST_Point(-123.365556, 48.428611),4326) As wgs84long_lat;
-- the ewkt representation (wrap with ST_AsEWKT) -
SRID=4326;POINT(-123.365556 48.428611)
```

-- Mark a point as WGS 84 long lat and then transform to web mercator (Spherical Mercator) --

```
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)
```

See Also

Section [4.5](#), [ST_SRID](#), [ST_Transform](#), [UpdateGeometrySRID](#)

8.7.2 ST_SRID

ST_SRID — Returns the spatial reference identifier for a geometry.

Synopsis

integer **ST_SRID**(geometry g1);

Description

Returns the spatial reference identifier for the ST_Geometry as defined in spatial_ref_sys table. Section [4.5](#)



Note

spatial_ref_sys table is a table that catalogs all spatial reference systems known to PostGIS and is used for transformations from one spatial reference system to another. So verifying you have the right spatial reference system identifier is important if you plan to ever transform your geometries.



This method implements the [OGC 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

Examples

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)',4326));
--result
4326
```

See Also

Section [4.5](#), [ST_SetSRID](#), [ST_Transform](#), [ST_SRID](#), [ST_SRID](#)

8.7.3 ST_Transform

ST_Transform — Return a new geometry with coordinates transformed to a different spatial reference system.

Synopsis

```
geometry ST_Transform(geometry g1, integer srid);
geometry ST_Transform(geometry geom, text to_proj);
geometry ST_Transform(geometry geom, text from_proj, text to_proj);
geometry ST_Transform(geometry geom, text from_proj, integer to_srid);
```

Description

Returns a new geometry with its coordinates transformed to a different spatial reference system. The destination spatial reference `to_srid` may be identified by a valid SRID integer parameter (i.e. it must exist in the `spatial_ref_sys` table). Alternatively, a spatial reference defined as a PROJ.4 string can be used for `to_proj` and/or `from_proj`, however these methods are not optimized. If the destination spatial reference system is expressed with a PROJ.4 string instead of an SRID, the SRID of the output geometry will be set to zero. With the exception of functions with `from_proj`, input geometries must have a defined SRID.

`ST_Transform` is often confused with `ST_SetSRID`. `ST_Transform` actually changes the coordinates of a geometry from one spatial reference system to another, while `ST_SetSRID()` simply changes the SRID identifier of the geometry.



Note

Requires PostGIS be compiled with PROJ support. Use [PostGIS_Full_Version](#) to confirm you have PROJ support compiled in.



Note

If using more than one transformation, it is useful to have a functional index on the commonly used transformations to take advantage of index usage.



Note

Prior to 1.3.4, this function crashes if used with geometries that contain CURVES. This is fixed in 1.3.4+

Enhanced: 2.0.0 support for Polyhedral surfaces was introduced.

Enhanced: 2.3.0 support for direct PROJ.4 text was introduced.



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.

Examples

Change Massachusetts state plane US feet geometry to WGS 84 long lat

```

SELECT ST_AsText(ST_Transform(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450,
      743265 2967450,743265.625 2967416,743238 2967416))',2249),4326)) As wgs_geom;

wgs_geom
-----
POLYGON((-71.1776848522251 42.3902896512902,-71.1776843766326 42.3903829478009,
-71.1775844305465 42.3903826677917,-71.1775825927231 42.3902893647987,-71.177684
8522251 42.3902896512902));
(1 row)

--3D Circular String example
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromEWKT('SRID=2249;CIRCULARSTRING(743238 2967416 ↵
      1,743238 2967450 2,743265 2967450 3,743265.625 2967416 3,743238 2967416 4)'),4326));

st_asewkt
-----
SRID=4326;CIRCULARSTRING(-71.1776848522251 42.3902896512902 1,-71.1776843766326 ↵
      42.3903829478009 2,
-71.1775844305465 42.3903826677917 3,
-71.1775825927231 42.3902893647987 3,-71.1776848522251 42.3902896512902 4)

```

Example of creating a partial functional index. For tables where you are not sure all the geometries will be filled in, its best to use a partial index that leaves out null geometries which will both conserve space and make your index smaller and more efficient.

```

CREATE INDEX idx_geom_26986_parcel
ON parcels
USING gist
(ST_Transform(geom, 26986))
WHERE geom IS NOT NULL;

```

Examples of using PROJ.4 text to transform with custom spatial references.

```

-- Find intersection of two polygons near the North pole, using a custom Gnomonic 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) ↵
)

```

Configuring transformation behavior

Sometimes coordinate transformation involving a grid-shift can fail, for example if PROJ.4 has not been built with grid-shift files or the coordinate does not lie within the range for which the grid shift is defined. By default, PostGIS will throw an error if a grid shift file is not present, but this behavior can be configured on a per-SRID basis either by testing different `to_proj` values of PROJ.4 text, or altering the `proj4text` value within the `spatial_ref_sys` table.

For example, the `proj4text` parameter `+datum=NAD87` is a shorthand form for the following `+nadgrids` parameter:

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat
```

The @ prefix means no error is reported if the files are not present, but if the end of the list is reached with no file having been appropriate (ie. found and overlapping) then an error is issued.

If, conversely, you wanted to ensure that at least the standard files were present, but that if all files were scanned without a hit a null transformation is applied you could use:

```
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat,null
```

The null grid shift file is a valid grid shift file covering the whole world and applying no shift. So for a complete example, if you wanted to alter PostGIS so that transformations to SRID 4267 that didn't lie within the correct range did not throw an ERROR, you would use the following:

```
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;
```

See Also

Section 4.5, [ST_SetSRID](#), [ST_SRID](#), [UpdateGeometrySRID](#)

8.8 Geometry Input

8.8.1 Well-Known Text (WKT)

8.8.1.1 ST_BdPolyFromText

ST_BdPolyFromText — **Well-Known Text (WKT) to Boundary Polygon**

WKT — Well-Known Text

SRID — SRID

Geometry — Geometry

Synopsis

geometry **ST_BdPolyFromText**(text WKT, integer srid);

WKT — Well-Known Text

SRID — SRID

Geometry — Geometry

Note



WKT is a text representation of a geometry. It is used to store and transport geometry data. The WKT format is defined by the OGC Simple Features Implementation Specification for SQL 1.1. The WKT format is a text representation of a geometry. It is used to store and transport geometry data. The WKT format is defined by the OGC Simple Features Implementation Specification for SQL 1.1. The WKT format is a text representation of a geometry. It is used to store and transport geometry data. The WKT format is defined by the OGC Simple Features Implementation Specification for SQL 1.1.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2

GEOS 3.10.0

1.1.0

관련 정보

ST_BuildArea, ST_BdMPolyFromText

8.8.1.2 ST_BdMPolyFromText

ST_BdMPolyFromText — `멀티라인스트링 WKT 텍스트표현식으로 주어진 임의의 폐쇄라인스트링 집합으로 멀티폴리작성합니다.`

Synopsis

geometry **ST_BdMPolyFromText**(text WKT, integer srid);

설명

`멀티라인스트링 WKT 텍스트 표현식으로 주어진 임의의 폐쇄형 라인스트링 집합으로 멀티폴리곤을 작성합니다.`

Note

WKT `가 멀티라인스트링이 아닐경우 오류가 발생합니다.`
`출력򻬼이 단일 폴리곤으로이루어졌을 경우에도 멀티폴리క제합니다.`
`해당 연산시단일 폴리곤을 출력할ಃ이라고 확신한다면 ST_BdPolyFromText`
`를 사용하౰나,, 또는 PostGIS 특화접근맅을 사용하려 할 경우을 ST_BuildArea() 를 참조하십시오.`



This method implements the **OGC Simple Features Implementation Specification for SQL 1.1**. s3.2.6.2

GEOS `모듈로 실행`

1.1.0 `버전부터 사용할 수 있습니다.`

관련 정보

ST_BuildArea, ST_BdPolyFromText

8.8.1.3 ST_GeogFromText

ST_GeogFromText — `WKT (확장) 표현식으로부터 지지리형 ఒ을 반환합니다.`

Synopsis

geography **ST_GeogFromText**(text EWKT);

설명

WKT 표현식 또는 확장 WKT 표현식으로지리형 객󉲴를 반환합니다. 투영따로 설정하지 않은 경우 SRID 4326으로 가정합니다. 이 함수는 ST_GeographyFromText와 동일합니다. 포인트는 항상 경위형식으로 표현됩니다.

예시

```
-- &#xacbd;&#xc704;&#xb3c4; &#xc88c;&#xd45c;&#xb97c; <←
    &#xc9c0;&#xb9ac;&#xd615;&#xc73c;&#xb85c; &#xbcc0;&#xd658;&#xd558;&#xae30;
ALTER TABLE sometable ADD COLUMN geog geography(POINT,4326);
UPDATE sometable SET geog = ST_GeogFromText('SRID=4326;POINT(' || lon || ' ' || lat || ')') <←
;

--- EPSG:4267, NAD27&#xc744; &#xc774;&#xc6a9;&#xd574;&#xc11c; &#xc9c0;&#xb9ac;&#xd615; <←
    &#xd3ec;&#xc778;&#xd2b8;&#xb97c; &#xc9c0;&#xc815;&#xd558;&#xae30;
SELECT ST_AsEWKT(ST_GeogFromText('SRID=4267;POINT(-77.0092 38.889588)'));
```

관련 정보

ST_AsText, ST_GeographyFromText

8.8.1.4 ST_GeographyFromText

ST_GeographyFromText — WKT (확장) 표현식으로부터지정된지리형 값을 반환합니다.

Synopsis

```
geography ST_GeographyFromText(text EWKT);
```

설명

WKT 표현식으로부터 지리형 객체&#xb97
반환합니다. 투영체를 따로 설정&#xf
않은 경우 SRID 4326으로 가정합니다.

관련 정보

ST_GeogFromText, ST_AsText

8.8.1.5 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

```
geometry ST_GeomCollFromText(text WKT, integer srid);
geometry ST_GeomCollFromText(text WKT);
```


ST_GeomFromText

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 3.2.6.2 - SRID; suite);

WKT null;

Note

WKT; SRID; ST_GeomFromText;



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s3.2.6.2



This method implements the SQL/MM specification.

ST_GeomCollFromText

```
SELECT ST_GeomCollFromText('GEOMETRYCOLLECTION(POINT(1 2),LINESTRING(1 2, 3 4))');
```

ST_GeomFromText, ST_SRID

ST_GeomFromText, ST_SRID

8.8.1.6 ST_GeomFromEWKT

ST_GeomFromEWKT — EWKT(Extended Well-Known Text); ST_GeomFromText;

Synopsis

geometry **ST_GeomFromEWKT**(text EWKT);

ST_GeomFromEWKT

OGC EWKT(Extended Well-Known Text); PostGIS ST_GeomFromEWKT;

**Note**

EWKT OGC SRID; PostGIS;

జ선 사항: 2.0.0 버전부터 다면체 표&#xba surface) 및 TIN을 지원합니다.

- ✔ 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).

예시

```
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,
-71.10414111411543 42.3141545014533,-71.1041287795912 42.3142114839058,
-71.1041188134329 42.3142693656241,-71.1041112482575 42.3143272556118,
-71.1041072845732 42.3143851580048,-71.1041057218871 42.3144430686681,
-71.1041065602059 42.3145009876017,-71.1041097995362 42.3145589148055,
-71.1041166403905 42.3146168544148,-71.1041258822717 42.3146748022936,
-71.1041375307579 42.3147318674446,-71.1041492906949 42.3147711126569,
-71.1041598612795 42.314808571739,-71.1042515013869 42.3151287620809,
-71.1041173835118 42.3150739481917,-71.1040809891419 42.3151344119048,
-71.1040438678912 42.3151191367447,-71.1040194562988 42.3151832057859,
-71.1038734225584 42.3151140942995,-71.1038446938243 42.3151006300338,
-71.1038315271889 42.315094347535,-71.1037393329282 42.315054824985,
-71.1035447555574 42.3152608696313,-71.1033436658644 42.3151648370544,
-71.1032580383161 42.3152269126061,-71.103223066939 42.3152517403219,
-71.1031880899493 42.3152774590236)),
((-71.1043632495873 42.315113108546,-71.1043583974082 42.3151211109857,
-71.1043443253471 42.3150676015829,-71.1043850704575 42.3150793250568,-71.1043632495873 ↵
42.315113108546))))');
```

```
-- 3D &#xc6d0;&#xd615; &#xc2a4;&#xd2b8;&#xb9c1;  
SELECT ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)');
```

```
-- &#xb2e4;&#xba74;&#xc6b4; &#xd45c;&#xba74; &#xc608;&#xc2dc;
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))
)');
```

ST_AsEWKT, ST_GeomFromText, ST_GeomFromEWKT

ST_AsEWKT, ST_GeomFromText, ST_GeomFromEWKT

8.8.1.7 ST_GeomFromMARC21

ST_GeomFromMARC21 — Takes MARC21/XML geographic data as input and returns a PostGIS geometry object.

Synopsis

geometry **ST_GeomFromMARC21** (text marcxml);

ST_GeomFromMARC21

This function creates a PostGIS geometry from a MARC21/XML record, which can contain a **POINT** or a **POLYGON**. In case of multiple geographic data entries in the same MARC21/XML record, a **MULTIPOINT** or **MULTIPOLYGON** will be returned. If the record contains mixed geometry types, a **GEOMETRYCOLLECTION** will be returned. It returns **NULL** if the MARC21/XML record does not contain any geographic data (datafield:034).

LOC MARC21/XML versions supported:

- **MARC21/XML 1.1**

Availability: 3.3.0, requires libxml2 2.6+



Note

The MARC21/XML Coded Cartographic Mathematical Data currently does not provide any means to describe the Spatial Reference System of the encoded coordinates, so this function will always return a geometry with **SRID 0**.



Note

Returned **POLYGON** geometries will always be clockwise oriented.

ST_GeomFromMARC21

Converting MARC21/XML geographic data containing a single **POINT** encoded as **hddd.dddddd**

```
SELECT
    ST_AsText (
        ST_GeomFromMARC21 ( '
            <record xmlns="http://www.loc.gov/MARC21/slim">
              <leader>00000nz a2200000nc 4500</leader>
              <controlfield tag="001">040277569</controlfield>
              <datafield tag="034" ind1=" " ind2=" ">
                <subfield code="d">W004.500000</subfield>
                <subfield code="e">W004.500000</subfield>
                <subfield code="f">N054.250000</subfield>
                <subfield code="g">N054.250000</subfield>
              </datafield>
            </record>
        '
    )
```

```

        </datafield>
    </record>')));

st_astext
-----
POINT(-4.5 54.25)
(1 row)

```

Converting MARC21/XML geographic data containing a single POLYGON encoded as hdddmmss

```

SELECT
    ST_AsText (
        ST_GeomFromMARC21 ('
            <record xmlns="http://www.loc.gov/MARC21/slim">
                <leader>01062cem a2200241 a 4500</leader>
                <controlfield tag="001"> 84696781 </controlfield>
                <datafield tag="034" ind1="1" ind2=" ">
                    <subfield code="a">a</subfield>
                    <subfield code="b">50000</subfield>
                    <subfield code="d">E0130600</subfield>
                    <subfield code="e">E0133100</subfield>
                    <subfield code="f">N0523900</subfield>
                    <subfield code="g">N0522300</subfield>
                </datafield>
            </record>')));

    st_astext
    -----

POLYGON((13.1 52.65,13.516666666666667 52.65,13.516666666666667  ←
        52.383333333333333,13.1 52.383333333333333,13.1 52.65))
(1 row)

```

Converting MARC21/XML geographic data containing a POLYGON and a POINT:

```

SELECT
    ST_AsText (
        ST_GeomFromMARC21 ('
            <record xmlns="http://www.loc.gov/MARC21/slim">
                <datafield tag="034" ind1="1" ind2=" ">
                    <subfield code="a">a</subfield>
                    <subfield code="b">50000</subfield>
                    <subfield code="d">E0130600</subfield>
                    <subfield code="e">E0133100</subfield>
                    <subfield code="f">N0523900</subfield>
                    <subfield code="g">N0522300</subfield>
                </datafield>
                <datafield tag="034" ind1=" " ind2=" ">
                    <subfield code="d">W004.500000</subfield>
                    <subfield code="e">W004.500000</subfield>
                    <subfield code="f">N054.250000</subfield>
                    <subfield code="g">N054.250000</subfield>
                </datafield>
            </record>')));

    st_astext ←
    -----

GEOMETRYCOLLECTION(POLYGON((13.1 52.65,13.516666666666667  ←
        52.65,13.516666666666667 52.383333333333333,13.1 52.383333333333333,13.1  ←
        52.65)),POINT(-4.5 54.25))

```

(1 row)

```
&#xad00;&#xb828; &#xc815;&#xbcf4;
```

```
ST_AsMARC21
```

8.8.1.8 ST_GeometryFromText

ST_GeometryFromText — WKT(Well-Known Text) 로부터 지정된 ST_Geometry 값을 반환합니다. 이 함수는 ST_GeomFromText 함수와 동일합니다.

Synopsis

```
geometry ST_GeometryFromText(text WKT);
geometry ST_GeometryFromText(text WKT, integer srid);
```

```
&#xc124;&#xba85;
```



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#).



This method implements the SQL/MM specification. SQL-MM 3: 5.1.40

```
&#xad00;&#xb828; &#xc815;&#xbcf4;
```

```
ST_GeomFromText
```

8.8.1.9 ST_GeomFromText

ST_GeomFromText — WKT 표현식으로부터 지정된 ST_Geometry 값을 반환합니다.

Synopsis

```
geometry ST_GeomFromText(text WKT);
geometry ST_GeomFromText(text WKT, integer srid);
```

```
&#xc124;&#xba85;
```

OGC WKT(Well-Known Text) 표현식으로부터 PostGIS ST_Geometry 객체를 작성합니다.

Note

ST_GeomFromText 함수의 변종이 2개 있는데; 첫 번째는 SRID를 입력받지 않고 공간 참조 시스템이 정의되지 않은(SRID=0) 도형을 반환합니다. 두 번째는 SRID를 두 번째 인수로 입력받아 해당 SRID를 자체 메타데이터의 일부로 포함하는 도형을 반환합니다.





This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). 3.2.6.2 - SRID; (conformance suite);



This method implements the SQL/MM specification. SQL-MM 3: 5.1.40



This method supports Circular Strings and Curves

**Note**

While not OGC-compliant, **ST_MakePoint** is faster than **ST_GeomFromText** and **ST_PointFromText**. It is also easier to use for numeric coordinate values. **ST_Point** is another option similar in speed to **ST_MakePoint** and is OGC-compliant, but doesn't support anything but 2D points.

**Warning**

PostGIS 2.0.0: **ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)')** is faster than **ST_GeomFromText('POINT(0 0)')**. PostGIS 2.0.0: **SQL/MM** **ST_GeomFromText('MULTILINESTRING((-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.25932))')** is faster than **ST_GeomFromText('POINT(0 0)')**. **ST_GeomFromText('POLYGON((-71.1776585052917 42.3902909739571,-71.1776820268866 42.3903701743239,-71.1776063012595 42.3903825660754,-71.1775826583081 42.3903033653531,-71.1776585052917 42.3902909739571))')** is faster than **ST_GeomFromText('POINT(0 0)')**.

Example

```
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)');
```

관련 정보

ST_GeomFromEWKT, ST_GeomFromWKB, ST_SRID

8.8.1.10 ST_LineFromText

ST_LineFromText — 주어진 SRID와 함께 WKT 표현식으도형을 만듭니다. SRID가 주어지지않은 경우,기본값인 0을 씁니다.

Synopsis

```
geometry ST_LineFromText(text WKT);
geometry ST_LineFromText(text WKT, integer srid);
```

설명

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!

Note

OGC 사양 3.2.6.2 - 선택적인 SRID를
적합성 스위트(conformance suite)로부터
가져오십시오.



Note!

Note

```
&#xc0ac;&#xc6a9;&#xc790; &#xb3c4;&#xd615;&#xc774; &#xbaa8;&#xb450; &#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;
&#xac78; &#xc54c;&#xace0; &#xc788;&#xb2e4;&#xba74;, &#xadf8;&#xb0e5; ST_GeomFromText
&#xd568;&#xc218;&#xb97c; &#xc4f0;&#xb294; &#xd3b8;&#xc774; &#xb354; &#xd6a8;&#xc728;&#xc801;&#xc785;&#xb2c8;
&#xc774; &#xd568;&#xc218;&#xb294; ST_GeomFromText&#xb9cc; &#xd638;&#xcd9c;&#xd558;&#xace0;,
&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc744; &#xbc18;&#xd658;&#xd55c;&#xb2e4;&#xb294;
&#xc720;&#xd6a8;&#xc131; &#xac80;&#xc0ac;&#xb97c; &#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;.
```



This method implements the [OGC 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

8.8.1.12 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);
```

Notes

Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

OGC 3.2.6.2 - SRID; suite);

WKT; null;.

Note



WKT; ST_GeomFromText;

- ✓ This method implements the [OGC 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);
```

Related Functions

[ST_GeomFromText](#)

8.8.1.13 ST_MPolyFromText

ST_MPolyFromText — Makes a MultiPolygon Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

Synopsis

```
geometry ST_MPolyFromText(text WKT, integer srid);
geometry ST_MPolyFromText(text WKT);
```

ST_MultiFromText

Makes a MultiPolygon from WKT with the given SRID. If SRID is not given, it defaults to 0.

OGC 3.2.6.2 - SRID; suite);

WKT; ST_GeomFromText;

Note

WKT; ST_GeomFromText;



This method implements the [OGC 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

ST_MultiFromText

```
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);
```

ST_GeomFromText

ST_GeomFromText, **ST_SRID**

8.8.1.14 ST_PointFromText

ST_PointFromText — SRID; WKT; SRID; ST_GeomFromText;

Synopsis

geometry **ST_PointFromText**(text WKT);
 geometry **ST_PointFromText**(text WKT, integer srid);

설명

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

```
ST_PointFromText (&#xd568;&#xc218;&#xc758; &#xbcc0;&#xc885;&#xc774; 2&#xac1c; &#xc788;&#xb294;&#xb370;,&#xcceb; &#xbcc8;&#xc9f8;&#xb294; SRID=&#xb97c; &#xc785;&#xb825;&#xbcb1;&#xc9c0; &#xc54a;&#xace0;&#xcacf5;&#xcac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c;&#xc774; &#xc815;&#xc758;&#xb418;&#xc9c0;&#xc54a;&#xc740; &#xb3c4;&#xd615;&#xc744; &#xbcb18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. &#xb450;&#xbcc88;&#xc9f8;&#xb294; SRID=&#xb97c; &#xb450; &#xbcc88;&#xc9f8; &#xc778;&#xc218;&#xb85c;&#xc785;&#xb825;&#xbcb1;&#xc544; &#xc790;&#xcceb4; &#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130;&#xc758;&#xc77c;&#xbd80;&#xb85c; &#xd574;&#xb2f9; SRID=&#xb97c; &#xd3ec;&#xd568;&#xd558;&#xb294; ST_Geometry=&#xb97c; &#xbcb18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. spatial_ref_sys &#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xc815;&#xc758;&#xb418;&#xc5b4; &#xc788;&#xb294; SRID=&#xc5ec;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;.
```

Note

```
&#xc0ac;&#xc6a9;&#xc790;      WKT      &#xb3c4;&#xd615;&#xc774;      &#xbaa8;&#xb450;
&#xd3ec;&#xc778;&#xd2b8;&#xb77c;&#xace0;      &#xd655;&#xc2e0;&#xd55c;&#xb2e4;&#xba74;      &#xc774;
&#xd568;&#xc218;&#xb97c;      &#xc0ac;&#xc6a9;&#xd558;&#xc9c0;      &#xb9c8;&#xc2ed;&#xc2dc;&#xc624;
&#xc774;      &#xd568;&#xc218;&#xb294;      &#xcd94;&#xac00;&#xc801;&#xc778;      &#xc720;&#xd6a8;&#xc131;
&#xac80;&#xc0ac;      &#xb2e8;&#xacc4;&#xb97c;      &#xac70;&#xce58;&#xbbc0;&#xb85c;
ST_GeomFromText(&#xbcf4;&#xb2e4; &#xb290;&#xb9bd;&#xb2c8;&#xb2e4;      &#xc0ac;&#xc6a9;&#xc790;&#xac00;
&#xacbd;&#xc704;&#xb3c4; &#xc88c;&#xd45c;&#xb85c;&#xbd80;&#xd130; &#xd3ec;&#xc778;&#xd2b8;&#xb97c;
&#xbe4c;&#xb4dc;&#xd558;&#xace0;      OGC      &#xc900;&#xc218;      &#xc5ec;&#xbd80;&#xbcf4;&#xb2e4;
&#xc131;&#xb2a5; &#xbcf0; &#xc815;&#xd655;&#xb3c4;&#xc5d0; &#xb354; &#xad00;&#xc2ec;&#xc774;
&#xc788;&#xb2e4;&#xba74; &#xc774; &#xb610;&#xb294; OGC&#xb97c; &#xc900;&#xc218;&#xd558;&#xb294;
&#xb3d9;&#xc77c;&#xd55c; &#xc774; &#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;
```

- ✔ This method implements the **OGC Simple Features Implementation Specification for SQL 1.1**. 3.2.6.2 - SRID; (conformance suite);
- ✔ This method implements the SQL/MM specification. SQL-MM 3: 6.1.8

예시

```
SELECT ST_PointFromText('POINT(-71.064544 42.28787)');
SELECT ST_PointFromText('POINT(-71.064544 42.28787)', 4326);
```

&#xad00:&#xb828: &#xc815:&#xbcf4:

ST_GeomFromText, ST_MakePoint, ST_Point, ST_SRID

8.8.1.15 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);
```

설명

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 사양 3.2.6.2 - 선택적인 SRID=를 적합성 스–로부터 가져오십시오.

Note



```
&#xc0ac;&#xc6a9;&#xc790;          WKT          &#xb3c4;&#xd615;&#xc774;          &#xbaa8;&#xb450;
&#xd3f4;&#xb9ac;&#xace4;&#xc774;&#xb77c;&#xace0;  &#xd655;&#xc2e0;&#xd55c;&#xb2e4;&#xba74;,  &#xc774;
&#xd568;&#xc218;&#xb97c; &#xc0ac;&#xc6a9;&#xd558;&#xc9c0; &#xb9c8;&#xc2ed;&#xc2dc;&#xc624;.  &#xc774;
&#xd568;&#xc218;&#xb294; &#xcd94;&#xac00;&#xc801;&#xc778; &#xc720;&#xd6a8;&#xc131; &#xac80;&#xc0ac;
&#xb2e8;&#xacc4;&#xb97c;          &#xac70;&#xce58;&#xbbc0;&#xb85c;          ST_GeomFromText&#xbcf4;&#xb2e4;
&#xb290;&#xb9bd;&#xb2c8;&#xb2e4;.
```

- ✔ This method implements the [OGC 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

예시

```
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
-----
0103000000010000000500000006...

SELECT ST_PolygonFromText('POINT(1 2)') IS NULL as point_is_notpoly;

point_is_not_poly
-----
t
```

관련 정보

ST_GeomFromText

8.8.1.16 ST_WKTTToSQL

ST_WKTToSQL — WKT(Well-Known Text)로부터 지정된 ST_Geometry 값&#x
반환합니다. 이 함수는 ST_GeomFromText 함수&
동일합니다.

Synopsis

```
geometry ST_WKTTToSQL(text WKT);
```

ST_GeomFromText



This method implements the SQL/MM specification. SQL-MM 3: 5.1.34

ST_GeomFromText

ST_GeomFromText

8.8.2 Well-Known Binary (WKB)

8.8.2.1 ST_GeogFromWKB

ST_GeogFromWKB — WKB (Well-Known Binary) or EWKB (Extended Well-Known Binary) representation of a geometry.

Synopsis

geography **ST_GeogFromWKB**(bytea wkb);

ST_GeogFromWKB

ST_GeogFromWKB (WKB or EWKB representation of a geometry). The WKB representation is a binary format for storing and transmitting geometry data. The EWKB representation is an extension of WKB that includes additional information such as SRID and spatial reference system details.

SRID (Spatial Reference ID) is a value that identifies the spatial reference system used for the geometry. It is stored in the WKB header.



This method supports Circular Strings and Curves

ST_GeogFromText

```
-- bytea representation of a geometry (WKB)
SELECT ST_AsText(
  ST_GeogFromWKB(E'\\001\\002\\000\\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)
```

ST_GeogFromText

ST_GeogFromText, **ST_AsBinary**

8.8.2.2 ST_GeomFromEWKB

ST_GeomFromEWKB — EWKB(Extended Well-Known Binary) ST_Geometry

Synopsis

geometry ST_GeomFromEWKB(bytea EWKB);

Note

OGC EWKB(Extended Well-Known Binary) PostGIS ST_Geom



Note

EWKB OGC SRID(PostGIS

surface) TIN



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).

Note

NAD83 LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.25932)



Note

standard_conforming_strings 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\300p ←
\231\323e1')
```

관련 정보

ST_AsBinary, ST_AsEWKB, ST_GeomFromWKB

8.8.2.3 ST_GeomFromWKB

ST_GeomFromWKB — WKB(Well-Known Binary) τ형 표현식과 선택&
SRID로부터 도형 인스턴스를 생성&#xd56

Synopsis

```
geometry ST_GeomFromWKB(bytea geom);
geometry ST_GeomFromWKB(bytea geom, integer srid);
```

설명

```
ST_GeogFromWKB &#xd568;&#xc218;&#xb294;&#xb3c4;&#xd615;&#xc758; WKB &#xd45c;&#xd604;&#xc2dd;&#xacfc;
SRID(&#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c; ID)&#xb97c; &#xbc1b;&#xc544; &#xc801;&#xc808;&#
&#xb3c4;&#xd615; &#xc720;&#xd615;&#xc758; &#xc778;&#xc2a4;&#xd134;&#xc2a4;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#
&#xc774; &#xd568;&#xc218;&#xb294; SQL&#xc5d0;&#xc11c; &#xb3c4;&#xd615; &#xacf5;&#xc7a5;(Geometry Factory)
&#xc5ed;&#xd560;&#xc744; &#xd569;&#xb2c8;&#xb2e4;. &#xc774; &#xd568;&#xc218;&#xb294; ST_WKBToSQL &#xd568;&#x
&#xb300;&#xccb4;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

SRIDॼ 설정하지 않은 경우, 기본값&#xc730(unkown)을 씁니다.



 This method implements the OGC Simple Features Implementation Specification for SQL 1.1. s3.2.7.2 - SRID; (conformance suite);



 This method implements the SQL/MM specification. SQL-MM 3: 5.1.41



 This method supports Circular Strings and Curves

예시

```
-- standard_conforming_strings&#xc774; &#xd65c;&#xc131;&#xd654;&#xb3fc; &#xc788;&#xc9c0; &#xc54a;&#xc740; &#xacbd;&#xc6b0;
-- bytea &#xd45c;&#xd604;&#xc2dd;&#xc774; &#xb2e8;&#xc77c;&#xd55c; &#xc5ed;&#xc2ac;&#xb798;&#xc2dc; (\) &#xb97c; &#xd3ec;&#xd568;&#xd558;&#xae34; &#xd574;&#xb3c4;, &#xd14c;&#xc774;&#xbe14; &#xc0bd;&#xc785;&#xc2dc;&#xc5d0;&#xb294; &#xbe7c;&#xb193;&#xc544;&#xc57c; &#xd569;&#xb2c8;&#xb2e4; .
```

```
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)
```

관련 정보

ST_WKBToSQL, ST_AsBinary, ST_GeomFromEWKB

8.8.2.4 ST_LineFromWKB

```
ST_LineFromWKB — &#xc8fc;&#xc5b4;&#xc9c4; SRID&#xc640; &#xd568;&#xaed8; WKB&#xb85c;&#xbd80;&#xd130;
LINESTRING &#xc744; &#xb9cc;&#xb4ed;&#xb2c8;&#xb2e4;.
```

Synopsis

```
geometry ST_LineFromWKB(bytea WKB);
geometry ST_LineFromWKB(bytea WKB, integer srid);
```

설명

ST_LineFromWKB 함수는도형의 WKB 표현식과
SRID(공간 참조 시스템 ID)를 받아 적절&#
도형 유형의 인스턴스 - 이 경우, LINESTR
도형 - 를 생성합니다. 이 함수는
SQL에서 도형 공장(Geometry Factory) 역할을 합&#xb2c8
SRID를 설정하지 않은 경우, 기본값&#xc77
0을 씁니다. 입력된 bytea가 라인스트&#
아닌 경우, NULL을 반환합니다.



Note

OGC 사양 3.2.6.2 - 선택적인 SRID를
적합성 스위트(conformance suite)로붰터
가져오십시오.

Note



```
&#xc0ac;&#xc6a9;&#xc790; &#xb3c4;&#xd615;&#xc774; &#xbaa8;&#xb450; LINESTRING&#xc774;&#xb780;
&#xac78; &#xc54c;&#xace0; &#xc788;&#xb2e4;&#xba74;, &#xadf8;&#xb0e5; ST_GeomFromWKB
&#xd568;&#xc218;&#xb97c; &#xc4f0;&#xb294; &#xd3b8;&#xc774; &#xb354; &#xd6a8;&#xc728;&#xc801;&#xc785;&#xb2c8;
&#xc774; &#xd568;&#xc218;&#xb294; ST_GeomFromWKB &#xb9cc; &#xd638;&#xcd9c;&#xd558;&#xace0;,
&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc744; &#xbc18;&#xd658;&#xd55c;&#xb2e4;&#xb294;
&#xc720;&#xd6a8;&#xc131; &#xac80;&#xc0ac;&#xb97c; &#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;.
```



This method implements the **OGC 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

예시

```
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
-----
01020000000020000000000000000000F ... | t
```

관련 정보

ST_GeomFromWKB, ST_LinestringFromWKB

8.8.2.5 ST_LinestringFromWKB

```
ST_LineStringFromWKB — &#xc8fc;&#xc5b4;&#xc9c4; SRID&#xc640; &#xd568;&#xae8; WKB&#xb85c;&#xbd80;&#xd130;
&#xb3c4;&#xd615;&#xc744; &#xb9cc;&#xb4ed;&#xb2c8;&#xb2e4;.
```

Synopsis

```
geometry ST_LinestringFromWKB(bytea WKB);
geometry ST_LinestringFromWKB(bytea WKB, integer srid);
```

설&#xba85:

```
ST_LinestringFromWKB &#xd568;&#xc218;&#xb294;&#xb3c4;&#xd615;&#xc758; WKB &#xd45c;&#xd604;&#xc2dd;&#xac
SRID(&#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c; ID)&#xb97c; &#xbc1b;&#xc544; &#xc801;&#xc808;&#
&#xb3c4;&#xd615; &#xc720;&#xd615;&#xc758; &#xc778;&#xc2a4;&#xd134;&#xc2a4; - &#xc774; &#xacbd;&#xc6b0;, LINESTR
&#xb3c4;&#xd615; - &#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;, &#xc774; &#xd568;&#xc218;&#xb294;
SQL&#xc5d0;&#xc11c; &#xb3c4;&#xd615; &#xacf5;&#xc7a5;(Geometry Factory) &#xc5ed;&#xd560;&#xc744; &#xd569;&#xb2c8
SRID&#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc740; &#xacbd;&#xc6b0;, &#xae30;&#xbcf8;&#xac12;&#xc77
0&#xc744; &#xc501;&#xb2c8;&#xb2e4;, &#xc785;&#xb825;&#xb41c; bytea&#xac00; LINESTRING &#xb3c4;&#xd615;&#xc77
&#xc544;&#xb2cc; &#xacbd;&#xc6b0;, NULL&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```



Note

OGC 사양 3.2.6.2 - 선택적인 SRID를
적합성 스위트(conformance suite)로부터
가져오십시오.

Note

`LINESTRING(2 3 4)`;
`ST_GeomFromWKB`
`ST_GeomFromWKB`
`LINESTRING(2 3 4)`;
`ST_GeomFromWKB`
`LINESTRING(2 3 4)`;



This method implements the [OGC 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

Example

```
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;
-----
01020000000200000000000000000000F ... | t
```

Example

`ST_GeomFromWKB`, `ST_LineFromWKB`

8.8.2.6 ST_PointFromWKB

`ST_PointFromWKB` — `SRID`; `WKB`; `SRID`; `WKB`;

Synopsis

geometry `ST_GeomFromWKB`(bytea geom);
 geometry `ST_GeomFromWKB`(bytea geom, integer srid);

Example

`ST_PointFromWKB`; `WKB`; `SRID`; `WKB`; `SRID`; `WKB`;
`SRID`; `WKB`; `SRID`; `WKB`;
`SRID`; `WKB`; `SRID`; `WKB`;



This method implements the [OGC 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

ST_AsText

```
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)
```

ST_GeomFromWKB, ST_LineFromWKB

ST_GeomFromWKB, ST_LineFromWKB

8.8.2.7 ST_WKBToSQL

ST_WKBToSQL — WKB (Well-Known Binary) to SQL/MM geometry. The function takes a WKB geometry and returns a SQL/MM geometry. The function is implemented in the `liblwgeom` library. The function is implemented in the `liblwgeom` library. The function is implemented in the `liblwgeom` library.

Synopsis

geometry **ST_WKBToSQL**(bytea WKB);

ST_WKBToSQL



This method implements the SQL/MM specification. SQL-MM 3: 5.1.36

ST_GeomFromWKB

ST_GeomFromWKB

8.8.3 Other Formats

8.8.3.1 ST_Box2dFromGeoHash

ST_Box2dFromGeoHash — GeoHash BOX2D based on the input GeoHash string.

Synopsis

ST_Box2dFromGeoHash(text geohash, integer precision=full_precision_of_geohash);

ST_Box2dFromGeoHash

GeoHash BOX2D based on the input GeoHash string.

If no precision is specified **ST_Box2dFromGeoHash** returns a BOX2D based on full precision of the input GeoHash string.

ST_Box2dFromGeoHash(text geohash, integer precision=full_precision_of_geohash);

2.1.0 BOX2D based on the input GeoHash string.

ST_Box2dFromGeoHash

```
SELECT ST_Box2dFromGeoHash('9qqj7nmxcggy4d0dbxqz0');

      st_geomfromgeohash
-----
BOX(-115.172816 36.114646,-115.172816 36.114646)

SELECT ST_Box2dFromGeoHash('9qqj7nmxcggy4d0dbxqz0', 0);

      st_box2dfromgeohash
-----
BOX(-180 -90,180 90)

SELECT ST_Box2dFromGeoHash('9qqj7nmxcggy4d0dbxqz0', 10);

      st_box2dfromgeohash
-----
BOX(-115.17282128334 36.1146408319473,-115.172810554504 36.1146461963654)
```

ST_Box2dFromGeoHash

ST_GeoHash, **ST_GeomFromGeoHash**, **ST_PointFromGeoHash**

8.8.3.2 ST_GeomFromGeoHash

ST_GeomFromGeoHash — GeoHash geometry based on the input GeoHash string.

Synopsis

geometry **ST_GeomFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

GeoHash

GeoHash — `ST_GeomFromGeoHash(geo_hash_text)` returns a `geometry` object from a GeoHash string. The GeoHash string is a string of characters from the set `0-9`, `a-v`, excluding `i` and `o`. The string is interpreted as a base-32 number, where `0` is the least significant digit and `v` is the most significant digit. The string is then converted to a decimal number, which is used to calculate the bounding box of the GeoHash. The bounding box is returned as a `geometry` object.

`precision` — `ST_GeomFromGeoHash(geo_hash_text, precision)` returns a `geometry` object from a GeoHash string, with a specified precision. The precision is a positive integer, representing the number of digits in the GeoHash string. The string is interpreted as a base-32 number, where `0` is the least significant digit and `v` is the most significant digit. The string is then converted to a decimal number, which is used to calculate the bounding box of the GeoHash. The bounding box is returned as a `geometry` object.

`precision` — `ST_GeomFromGeoHash(geo_hash_text, precision, srid)` returns a `geometry` object from a GeoHash string, with a specified precision and SRID. The precision is a positive integer, representing the number of digits in the GeoHash string. The string is interpreted as a base-32 number, where `0` is the least significant digit and `v` is the most significant digit. The string is then converted to a decimal number, which is used to calculate the bounding box of the GeoHash. The bounding box is returned as a `geometry` object.

2.1.0 — `ST_GeomFromGeoHash(geo_hash_text, precision, srid)` returns a `geometry` object from a GeoHash string, with a specified precision and SRID. The precision is a positive integer, representing the number of digits in the GeoHash string. The string is interpreted as a base-32 number, where `0` is the least significant digit and `v` is the most significant digit. The string is then converted to a decimal number, which is used to calculate the bounding box of the GeoHash. The bounding box is returned as a `geometry` object.

ST_GeomFromGeoHash

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxnccgyy4d0dbxqz0'));
          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('9qqj7nmxnccgyy4d0dbxqz0', 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('9qqj7nmxnccgyy4d0dbxqz0', 10));
          st_astext
```

```
POLYGON((-115.17282128334 36.1146408319473,-115.17282128334 36.1146461963654,-115.172810554504 36.1146461963654,-115.172810554504 36.1146408319473,-115.17282128334 36.1146408319473))
```

ST_GeomFromGML

`ST_GeoHash`, `ST_Box2dFromGeoHash`, `ST_PointFromGeoHash`

8.8.3.3 ST_GeomFromGML

`ST_GeomFromGML` — `ST_GeomFromGML(gml_text)` returns a `geometry` object from a GML string. The GML string is a string of characters from the set `0-9`, `a-z`, `A-Z`, `+`, `-`, `.`, `;`, `{}[]`, `()<>`, `"/>`. The string is interpreted as a GML string, which is used to calculate the bounding box of the GeoHash. The bounding box is returned as a `geometry` object.

Synopsis

```
geometry ST_GeomFromGML(text geomgml);
geometry ST_GeomFromGML(text geomgml, integer srid);
```

설명

OGC GML 표현식으로부터 PostGIS ST_Geometry 객체를작성합니다.

ST_GeomFromGML('GML (geometry fragment)';(geometry fragment);

지원하는 OGC GML 버전은 다음과 같습습

- GML 3.2.1 네임스페이스
- GML 3.1.1 단순피처프로파일 SF-2 (GML 3.1.0 및 3.0.0 하위호환)
- GML 2.1.2

OGC GML 표준: <http://www.opengeospatial.org/standards/gml>

1.5 ಄전&#bd80;터 사용할 수 있습니다.

LibXML2 1.6 이상 버전이 필요합니다.

&#x1c; Ġ
 항: 2.0.0 버 전 부 터 다 면 체 표 &#ba surface) 및 TIN 을 지 원 합 니 다.

개선 사항: 2.0.0 버전에서 선택적인

기본 SRID 파라미터가 추가됐습니다

- ✔ 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).

GML은 혼 합 차 원 을 (예 를 들 어 동 일 퇒 멀 티 도 형 (MultiGeometry) 안 에 서 2D와 3D를 함 꺈 쓸 수 있 습 니 다 PostGIS 도 형 은 그 렇 지 않 기 때 문 에 사 라 진 Z 차 원 을 찾 은 경 우 ST_GeomFromGML이 전 체 도 형 을 2D로 변 홑

GML은 동일한 멀티 도형 안에서 혼한 를 지원합 니다. PostGIS 도형 은 그렇지 않기 때문에. 이 경우 ST_GeomFromGML이 모든

하위 도형들을 SRS 루트 노드로 재&#xd22

GML ࣨ트 노드로 쓸 수 있는 srsName 속성읷 없을 경우., 오류가 발생합니다.

ST_GeomFromGML'݀ 분 명 한 GML'네 임 스 페 이 스 &#xb9'꼼 꼼 히 따 지 지 않 습 니 다. 일 반 적 &#x'사 용 시 에 는 네 임 스 페 이 스 를 명 &#x'언 급 하 지 않 아 도 됩 니 다. 하 지 만 GML'내 부 에 XLink'피 처 를 사 용 하 고 싶 다 그 래 야 합 니 다.



Note

```
ST_GeomFromGML    &#xd568;&#xc218;&#xb294;    SQL/MM    &#xb9cc;&#xace1;    &#xb3c4;&#xd615;&#xc744;
&#xc9c0;&#xc6d0;&#xd558;&#xc9c0; &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;
```

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>
  >');

```

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>
  >'););

```

SELECT ST_AsEWKT(ST_GeomFromGML('

```

    <gml:PolyhedralSurface>
    <gml:polygonPatches>
      <gml:PolygonPatch>
        <gml:exterior>
          <gml:LinearRing>
            <gml:posList srsDimension="3"
              >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 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:exterior>
        </gml:exterior>
      </gml:exterior>
    </gml:PolyhedralSurface>
  >');

```

```

</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
>'))';

-- &#xadb0;&#xadb1; --
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)))

```

관련정보

Section 2.2.3, [ST_AsGML](#), [ST_GMLToSQL](#)

8.8.3.4 ST_GeomFromGeoJSON

ST_GeomFromGeoJSON — GeoJSON 표현식을 입력򼬛아 PostGIS 도형 객체를 출력합니다.

Synopsis

```

geometry ST_GeomFromGeoJSON(text geomjson);
geometry ST_GeomFromGeoJSON(json geomjson);
geometry ST_GeomFromGeoJSON(jsonb geomjson);

```

설명

GeoJSON 표현식으로부터 PostGIS 도형 객체를 출력합니다.


```
ST_GeomFromGML'JSON';(geometry fragment);&#xb300;&#xd574;&#xc791;&#xb3d9;&#xd569;&#xb2c8;&#xb2e4.;&#xc644;&#xc804;&#xd55c; JSON'&#bb38;&#xc11c;&#xb97c;&#xc0ac;&#xc6f6;&#xd560;&#xacbd;&#xc6b0;&#xc624;&#xb958;&#xac00;&#xbc1c;&#xc0dd;&#xd569;&#xb2c8;&#xb2e4.;
```

Enhanced: 3.0.0 parsed geometry defaults to SRID=4326 if not specified otherwise.

Enhanced: 2.5.0 can now accept json and jsonb as inputs.

2.0.0 버전부터 사용할 수 있습니다.
JSON-C 0.9 이상 버전이 필요합니다.

Note

```
JSON-C      &#xc9c0;&#xc6d0;&#xc744;      &#xd65c;&#xc131;&#xd654;&#xc2dc;&#xd0a4;&#xc9c0;
&#xc54a;&#xc558;&#xb2e4;&#xba74;,    &#xcd9c;&#xb825;&#xbb3c;    &#xb300;&#xc2e0;    &#xc624;&#xb958;
&#xba54;&#xc2dc;&#xc9c0;&#xb97c;    &#xbcf4;&#xac8c;    &#xb420;    &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
JSON-C&#xb97c;    &#xd65c;&#xc131;&#xd654;&#xd558;&#xb824;&#xba74;,    "--with-jsondir=/path/to/json-c"
&#xc778;&#xc790;&#xc640;    &#xd568;&#xaed8;    &#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
&#xc790;&#xc138;&#xd55c;    &#xb0b4;&#xc6a9;&#xc740;    Section    2.2.3    &#xc744;
&#xcc38;&#xc870;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```



This function supports 3d and will not drop the z-index.

예시

```
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)
```

관련 정보

ST_AsText, ST_AsGeoJSON, Section 2.2.3

8.8.3.5 ST_GeomFromKML

ST_GeomFromKML — [#b3c4;#d615;#c758; KML &#d45c;#d604;#xc2dd;#xc744; 입#xb825;#xbc1b;#b3c4;#d615;#xcac1d;#xc9cb4;#xb97c;#xcd9c;#xb825;#xd569;#xb2c8;#xb2e4;.](#)
PostGIS [#b3c4;#d615; 객#xc9cb4;#xb97c;#xcd9c;#xb825;#xd569;#xb2c8;#xb2e4;.](#)

Synopsis

```
geometry ST_GeomFromKML(text geomkml);
```

ST_GeomFromKML

OGC KML <LineString> <coordinates>
<-71.1663, 42.2614
-71.1667, 42.2616</coordinates>
</LineString>
>');
--

ST_GeomFromKML <KML> <Document>
<name>KML fragment</name>
<Style>
<id>1</id>
<name>Red</name>
<color>ff0000</color>
</Style>
<Placemark>
<name>Point</name>
<geometry>
<Point>
<coordinates>
<-71.1663, 42.2614
-71.1667, 42.2616</coordinates>
</Point>
</geometry>
</Placemark>
</Document>
</KML>
</ST_GeomFromKML>

OGC KML <Document> <name>KML fragment</name> <Style> <id>1</id> <name>Red</name> <color>ff0000</color> </Style> <Placemark> <name>Point</name> <geometry> <Point> <coordinates> <-71.1663, 42.2614 -71.1667, 42.2616</coordinates> </Point> </geometry> </Placemark> </Document>

- KML 2.2.0 <Document> <name>KML fragment</name> <Style> <id>1</id> <name>Red</name> <color>ff0000</color> </Style> <Placemark> <name>Point</name> <geometry> <Point> <coordinates> <-71.1663, 42.2614 -71.1667, 42.2616</coordinates> </Point> </geometry> </Placemark> </Document>

OGC KML <Document>: <http://www.opengeospatial.org/standards/kml>

Availability: 1.5, requires libxml2 2.6+



This function supports 3d and will not drop the z-index.

**Note**

ST_GeomFromKML <KML> <Document> <name>KML fragment</name> <Style> <id>1</id> <name>Red</name> <color>ff0000</color> </Style> <Placemark> <name>Point</name> <geometry> <Point> <coordinates> <-71.1663, 42.2614 -71.1667, 42.2616</coordinates> </Point> </geometry> </Placemark> </Document>

ST_GeomFromTWKB

```
SELECT ST_GeomFromTWKB('
  <LineString>
    <coordinates>
      <-71.1663, 42.2614
      -71.1667, 42.2616</coordinates>
    </LineString>
  >');
--
```

ST_GeomFromTWKB

Section 2.2.3, [ST_AsKML](#)

8.8.3.6 ST_GeomFromTWKB

ST_GeomFromTWKB — TWKB("Tiny Well-Known Binary") <KML> <Document> <name>KML fragment</name> <Style> <id>1</id> <name>Red</name> <color>ff0000</color> </Style> <Placemark> <name>Point</name> <geometry> <Point> <coordinates> <-71.1663, 42.2614 -71.1667, 42.2616</coordinates> </Point> </geometry> </Placemark> </Document>

Synopsis

geometry **ST_GeomFromTWKB**(bytea twkb);

ST_GeomFromTWKB

ST_GeomFromTWKB <KML> <Document> <name>KML fragment</name> <Style> <id>1</id> <name>Red</name> <color>ff0000</color> </Style> <Placemark> <name>Point</name> <geometry> <Point> <coordinates> <-71.1663, 42.2614 -71.1667, 42.2616</coordinates> </Point> </geometry> </Placemark> </Document>

예시

```
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)

```

관련 정보

ST_AsTWKB

8.8.3.7 ST_GMLToSQL

ST_GMLToSQL — GML to SQL; ST_Geometry — Geometry; ST_GeomFromGML — GeomFromGML

Synopsis

```
geometry ST_GMLToSQL(text geomgml);
geometry ST_GMLToSQL(text geomgml, integer srid);
```

설명



 This method implements the SQL/MM specification. SQL-MM 3: 5.1.50 (만곡 도형 지&#xc6제외)

1.5 ಄전&#bd80;&#d130; 사용할 수 있습니다,
LibXML2 1.6 이상 ಄전이 필요합니다.

&#x1c; Ġ
 항: 2.0.0 ஄ 전 &#bd80; &#d130; &#b2e4; &#ba74; &#ccb4; &#d45c; &#ba surface) ଏ TIN ݄ 지 원 &#d569; &#b2c8; &#b2e4;.

```
&#x0a1c;&#x120; &#x0a;&#xd56d;: 2.0.0 &#bc84;&#xc804;&#xc5d0;&#xc11c; &#xc120;&#xd0dd;&#xc801;&#xc778;
&#xae30;&#xbcf8; SRID &#xd30c;&#xb77c;&#xbbbf8;&#xd130;&#xac00; &#xcd94;&#xac00;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;
```

관련 정보

Section 2.2.3, ST_GeomFromGML, ST_AsGML

8.8.3.8 ST_LineFromEncodedPolyline

ST_LineFromEncodedPolyline — 인코딩된폴리라인(polyline)으라인스트링을생성합니다

Synopsis

geometry **ST_LineFromEncodedPolyline**(text polyline, integer precision=5);

설명

인코딩된 폴리라인 스트링으로라인스트링을 생성합니다.

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.

참조; <http://developers.google.com/maps/documentation/utilities/polylinealgorithm>

2.2.0 **버전부터 사용할 수 있습니다.**

예시

```
-- 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)
```

관련 정보

ST_AsEncodedPolyline

8.8.3.9 ST_PointFromGeoHash

ST_PointFromGeoHash — GeoHash **스트링으로부터 포인트련 환합니다.**

Synopsis

point **ST_PointFromGeoHash**(text geohash, integer precision=full_precision_of_geohash);

설명

GeoHash 스트링으로부터 포인트를 렘해당 포인트는 GeoHash의 중심점입니다.

precision 을 설정하지 않은 경우. **ST_PointFromGeoHash입력 GeoHash 스트링의 전체 정확도를 기렘으로 포인트를 렘환합니다.**

precision 을 설정한 경우. **ST_PointFromGeoHashఀ GeoHash에나온 그만큼의 문자를 이용해서 포인트를 생성합니다.**

2.1.0 **버전부터 사용할 수 있습니다.**

ST_GeomFromGeoHash

```
SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxnccggy4d0dbxqz0'));
      st_astext
-----
POINT(-115.172816 36.114646)

SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxnccggy4d0dbxqz0', 4));
      st_astext
-----
POINT(-115.13671875 36.123046875)

SELECT ST_AsText(ST_PointFromGeoHash('9qqj7nmxnccggy4d0dbxqz0', 10));
      st_astext
-----
POINT(-115.172815918922 36.1146435141563)
```

ST_Box2dFromGeoHash

ST_GeomFromGeoHash

8.8.3.10 ST_FromFlatGeobufToTable

ST_FromFlatGeobufToTable — Creates a table based on the structure of FlatGeobuf data.

Synopsis

geometry **ST_BdPolyFromText**(text WKT, integer srid);

ST_FromFlatGeobuf

Creates a table based on the structure of FlatGeobuf data. (<http://flatgeobuf.org>).

schema Schema name.

table Table name.

data Input FlatGeobuf data.

Availability: 3.2.0

8.8.3.11 ST_FromFlatGeobuf

ST_FromFlatGeobuf — Reads FlatGeobuf data.

Synopsis

setof anyelement **ST_FromFlatGeobuf**(anyelement Table reference, bytea FlatGeobuf input data);

ST_FromFlatGeobuf

Reads FlatGeobuf data (<http://flatgeobuf.org>). NOTE: PostgreSQL bytea cannot exceed 1GB.

tabletype reference to a table type.

data input FlatGeobuf data.

Availability: 3.2.0

8.9 Geometry Output

8.9.1 Well-Known Text (WKT)

8.9.1.1 ST_AsEWKT

ST_AsEWKT — WKT(Well-Known Text) SRID

Synopsis

```
text ST_AsEWKT(geometry g1);
text ST_AsEWKT(geometry g1, integer maxdecimaldigits=15);
text ST_AsEWKT(geography g1);
text ST_AsEWKT(geography g1, integer maxdecimaldigits=15);
```

Warning

Returns the Well-Known Text representation of the geometry prefixed with the SRID. The optional *maxdecimaldigits* argument may be used to reduce the maximum number of decimal digits after floating point used in output (defaults to 15).

To perform the inverse conversion of EWKT representation to PostGIS geometry use **ST_GeomFromEWKT**.



Warning

Using the *maxdecimaldigits* parameter can cause output geometry to become invalid. To avoid this use **ST_ReducePrecision** with a suitable gridsize first.



Note

The WKT spec does not include the SRID. To get the OGC WKT format use **ST_AsText**.



Warning

WKT format does not maintain precision so to prevent floating truncation, use **ST_AsBinary** or **ST_AsEWKB** format for transport.

Enhanced: 3.1.0 support for optional precision parameter.

2.0.0



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).

예시

```
SELECT ST_AsEWKT('0103000020E61000000100000005000000000000  
0000000000000000000000000000000000000000000000000000000  
F03F000000000000F03F000000000000F03F000000000000F03  
F0000000000000000000000000000000000000000000'::geometry);  
  
----- st_asewkt -----  
SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))  
(1 row)  
  
SELECT ST_AsEWKT('0108000080030000000000000060 ↵  
E30A4100000000785C0241000000000000F03F00000000018  
E20A4100000000485F02410000000000000400000000018  
E20A4100000000305C02410000000000000840')  
  
--st_asewkt---  
CIRCULARSTRING(220268 150415 1,220227 150505 2,220227 150406 3)
```

స고

ST_AsBinary, ST_AsEWKB, ST_AsText, ST_GeomFromEWKT

8.9.1.2 ST_AsText

ST_AsText — `τ형/지리형의 WKT(Well-Known Text) 표현식– SRID 메타데이터 없이 반환합니다.`

Synopsis

```
text ST_AsText(geometry g1);
text ST_AsText(geometry g1, integer maxdecimaldigits = 15);
text ST_AsText(geography g1);
text ST_AsText(geography g1, integer maxdecimaldigits = 15);
```

설명

Returns the OGC **Well-Known Text** (WKT) representation of the geometry/geography. The optional *maxdecimaldigits* argument may be used to limit the number of digits after the decimal point in output ordinates (defaults to 15).

To perform the inverse conversion of WKT representation to PostGIS geometry use `ST_GeomFromText`.



Note

The standard OGC WKT representation does not include the SRID. To include the SRID as part of the output representation, use the non-standard PostGIS function **ST_AsEWKT**



Warning

The textual representation of numbers in WKT may not maintain full floating-point precision. To ensure full accuracy for data storage or transport it is best to use **Well-Known Binary** (WKB) format (see **ST_AsBinary** and *maxdecimaldigits*).

**Warning**

Using the `maxdecimaldigits` parameter can cause output geometry to become invalid. To avoid this use `ST_ReducePrecision` with a suitable gridsize first.

Enhanced: 2.5 - optional parameter precision introduced.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.1



This method implements the SQL/MM specification. SQL-MM 3: 5.1.25



This method supports Circular Strings and Curves

Example:

```
SELECT ST_AsText('010300000001000000050000000000000000
000000000000000000000000000000000000000000000000
F03F000000000000F03F000000000000F03F000000000000F03
F0000000000000000000000000000000000000000000000');
```

```
st_astext
```

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

Full precision output is the default.

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)');
st_astext
-----
POINT(111.1111111 1.1111111)
```

The `maxdecimaldigits` argument can be used to limit output precision.

```
SELECT ST_AsText('POINT(111.1111111 1.1111111)', 2);
st_astext
-----
POINT(111.11 1.11)
```

ST_AsBinary, ST_AsEWKB, ST_AsEWKT, ST_GeomFromText**8.9.2 Well-Known Binary (WKB)****8.9.2.1 ST_AsBinary**

`ST_AsBinary` — Return the OGC/ISO Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.

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);
```


ST_AsBinary

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```

      st_asbinary
-----
\x010300000001000000050000000000000000000000000000000000000000000000000000000000000000
000000f03f000000000000f03f000000000000f03f000000000000f03f000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000

SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326), 'XDR');
      st_asbinary
-----
\x00000000003000000010000000500000000000000000000000000000000000000000000000000000000
00000000003ff00000000000003ff00000000000003ff00000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000

```

ST_AsEWKB

ST_GeomFromWKB, **ST_AsEWKB**, **ST_AsTWKB**, **ST_AsText**,

8.9.2.2 ST_AsEWKB

ST_AsEWKB — Return the Extended Well-Known Binary (EWKB) representation of the geometry with SRID meta data.

Synopsis

```
bytea ST_AsEWKB(geometry g1);
bytea ST_AsEWKB(geometry g1, text NDR_or_XDR);
```

ST_AsEWKB

Returns the **Extended Well-Known Binary** (EWKB) representation of the geometry with SRID metadata. The first function variant defaults to encoding using server machine endian. The second function variant takes a text argument specifying the endian encoding, either little-endian ('NDR') or big-endian ('XDR').

WKB format is useful to read geometry data from the database and maintaining full numeric precision. This avoids the precision rounding that can happen with text formats such as WKT.

To perform the inverse conversion of EWKB to PostGIS geometry use **ST_GeomFromEWKB**.



Note

To get the OGC/ISO WKB format use **ST_AsBinary**. Note that OGC/ISO WKB format does not include the SRID.

PostGIS 3.3.9dev: 2.0.0 „전부터 다면체 표면삼각형 및 TIN을 지원하기 시작했습



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).

사용자가 \$ 변수를 쿼리 결과로 대자ଔ스크립트는 다음과 같이 보다것입니다.

```
<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>
```

స고

ST_LineFromEncodedPolyline, ST_Segmentize

8.9.3.2 ST_AsFlatGeobuf

ST_AsFlatGeobuf — Return a FlatGeobuf representation of a set of rows.

Synopsis

```
bytea ST_AsFlatGeobuf(anyelement set row);
bytea ST_AsFlatGeobuf(anyelement row, bool index);
bytea ST_AsFlatGeobuf(anyelement row, bool index, text geom_name);
```

설명

Return a FlatGeobuf representation (<http://flatgeobuf.org>) of a set of rows corresponding to a FeatureCollection. NOTE: PostgreSQL bytea cannot exceed 1GB.

row row data with at least a geometry column.

`index toggle` spatial index creation. Default is false.

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: 3.2.0

8.9.3.3 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);
```

Warning

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

Example

```
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=
```

8.9.3.4 ST_AsGeoJSON

ST_AsGeoJSON — Return a geometry as a GeoJSON element.

Synopsis

text **ST_AsGeoJSON**(record feature, text geomcolumnname, integer maxdecimaldigits=9, boolean pretty_bool=false);

text **ST_AsGeoJSON**(geometry geom, integer maxdecimaldigits=9, integer options=8);

text **ST_AsGeoJSON**(geography geog, integer maxdecimaldigits=9, integer options=0);

Warning

Returns a geometry as a GeoJSON "geometry", or a row as a GeoJSON "feature". (See the [GeoJSON specifications RFC 7946](#)). 2D and 3D Geometries are both supported. GeoJSON only support SFS 1.1 geometry types (no curve support for example).

The `maxdecimaldigits` argument may be used to reduce the maximum number of decimal places used in output (defaults to 9). If you are using EPSG:4326 and are outputting the geometry only for display, `maxdecimaldigits=6` can be a good choice for many maps.

**Warning**

Using the `maxdecimaldigits` parameter can cause output geometry to become invalid. To avoid this use **ST_ReducePrecision** with a suitable gridsize first.

The `options` argument can be used to add BBOX or CRS in GeoJSON output:

- 0: means no option
- 1: GeoJSON BBOX
- 2: GeoJSON Short CRS (EPSG:4326)
- 4: GeoJSON Long CRS (urn:ogc:def:crs:EPSG::4326)
- 8: GeoJSON Short CRS if not EPSG:4326 (default)

The GeoJSON specification states that polygons are oriented using the Right-Hand Rule, and some clients require this orientation. This can be ensured by using `ST_ForcePolygonCCW`. The specification also requires that geometry be in the WGS84 coordinate system (SRID = 4326). If necessary geometry can be projected into WGS84 using `ST_Transform`: `ST_Transform(geom, 4326)`.

GeoJSON can be tested and viewed online at geojson.io and geojsonlint.com. It is widely supported by web mapping frameworks:

- [OpenLayers GeoJSON Example](#)
- [Leaflet GeoJSON Example](#)
- [Mapbox GL GeoJSON Example](#)

1.3.4

1.5.0
 2.0.0
 (default arg)
 (named arg)

Changed: 3.0.0 support records as input

Changed: 3.0.0 output SRID if not EPSG:4326.



This function supports 3d and will not drop the z-index.

Generate a FeatureCollection:

```
SELECT json_build_object(
  'type', 'FeatureCollection',
  'features', json_agg(ST_AsGeoJSON(t.*)::json)
)
FROM ( VALUES (1, 'one', 'POINT(1 1)::geometry),
              (2, 'two', 'POINT(2 2)'),
              (3, 'three', 'POINT(3 3)')
      ) as t(id, name, geom);
```

```
{"type" : "FeatureCollection", "features" : [{"type": "Feature", "geometry": {"type": "Point", "coordinates": [1,1]}, "properties": {"id": 1, "name": "one"}}, {"type": "Feature", "geometry": {"type": "Point", "coordinates": [2,2]}, "properties": {"id": 2, "name": "two"}}, {"type": "Feature", "geometry": {"type": "Point", "coordinates": [3,3]}, "properties": {"id": 3, "name": "three"}}]}
```

Generate a Feature:

```
SELECT ST_AsGeoJSON(t.*)
FROM (VALUES (1, 'one', 'POINT(1 1)::geometry)) AS t(id, name, geom);
```

st_asgeojson

```
{"type": "Feature", "geometry": {"type": "Point", "coordinates": [1,1]}, "properties": {"id": 1, "name": "one"}}
```

An alternate way to generate Features with an `id` property is to use `JSONB` functions and operators:

```
SELECT jsonb_build_object(
    'type',          'Feature',
    'id',            id,
    'geometry',      ST_AsGeoJSON(geom)::jsonb,
    'properties',    to_jsonb( t.* ) - 'id' - 'geom'
) AS json
FROM (VALUES (1, 'one', 'POINT(1 1)::geometry')) AS t(id, name, geom);
```

json

```

{"id": 1, "type": "Feature", "geometry": {"type": "Point", "coordinates": [1, 1]}, "
  properties": {"name": "one"}}

```

Don't forget to transform your data to WGS84 longitude, latitude to conform with the GeoJSON specification:

```
SELECT ST_AsGeoJSON(ST_Transform(geom,4326)) from fe_edges limit 1;
```

```
st_asgeojson
```

```
{"type": "MultiLineString", "coordinates": [[[[-89.734634999999997, 31.492072000000000], [-89.734955999999997, 31.492237999999997]]]]}
```

3D geometries are supported:

```
SELECT ST_AsGeoJSON('LINESTRING(1 2 3, 4 5 6)');
```

```
{"type": "LineString", "coordinates": [[1, 2, 3], [4, 5, 6]]}
```

స&#xace0:

ST_GeomFromGeoJSON, ST_ForcePolygonCCW, ST_Transform

8.9.3.5 ST AsGML

ST_AsGML — [#b3c4;#d615;#c744](#); GML 2 [#b610;#b294](#); GML 3 [#xbc84;#xc804;#xc694;#xc18c;#xb85c;#xbc18;#d658;#d569;#b2c8;#b2e4](#).

Synopsis

```
text ST_AsGML(geometry geom, integer maxdecimaldigits=15, integer options=0);
text ST_AsGML(geography geog, integer maxdecimaldigits=15, integer options=0, text nprefix=null, text id=null);
text ST_AsGML(integer version, geometry geom, integer maxdecimaldigits=15, integer options=0, text nprefix=null, text id=null);
text ST_AsGML(integer version, geography geog, integer maxdecimaldigits=15, integer options=0, text nprefix=null, text id=null);
```

설명

Return the geometry as a Geography Markup Language (GML) element. The version parameter, if specified, may be either 2 or 3. If no version parameter is specified then the default is assumed to be 2. The `maxdecimaldigits` argument may be used to reduce the maximum number of decimal places used in output (defaults to 15).



Warning

Using the `maxdecimaldigits` parameter can cause output geometry to become invalid. To avoid this use `ST_ReducePrecision` with a suitable gridsize first.

GML 2는. 2.1.2 버전을, GML 3은. 3.1.1 버전을 가리&#xd0'옵션' 인수는 비트필드(bitfield) 구조󌺴&#xCRS 출력 유형을 GML 출력으로 정의하데이터를 위도/경도로 선언하는데 쓸 수 있습니다.

- 0: GML Short CRS (예; EPSG:4326), 기본값
- 1: GML Long CRS (예; urn:ogc:def:crs:EPSG::4326)
- 2: GML 3에 한해, 출력물에서 srsDimension 속성&#xc7제거합니다.
- 4: GML 3에 한해, 라인을 위해 <Curve> 보다 <LineString> 태그를 사용합니다.
- 16: 데이터가 위도/경도(예; srid=4326)라고 선언합니다. 기본적으로는 데이&#평면 좌표라고 가정합니다. 이 옵 축의 순서(axis order)와 관련돼 있어, GML 3.1.1 출력물에 대해서만 쓸모가 있습&#따라서 이 옵션을 설정하면, 데이 경도 위도 대신 위도 경도로 좌표&#순서를 바꿀 것입니다.
- 32: 도형을 둘러싼 상자(envelope)를 출력&#xd5

```
&#xc0ac;&#xc6a9;&#xc790; &#xc9c0;&#xc815; &#xb124;&#xc784;&#xc2a4;&#xd398;&#xc774;&#xc2a4; &#xc811;&#xb450;&#xc124;&#xc815;&#xd558;&#xac70;&#xb098; &#xc811;&#xb450;&#xc0ac;&#xb97c; &#xc0ac;&#xc6a9;&#xd558;&#xc9c0; &#xc54a;&#xb3c4;&#xb85d; &#xc124;&#xc815;(&#xbe44;&#xc5b4; &#xc788;&#xb294; &#xacbd;&#xc6b0;)&#xd558;&#xb294; &#xb370; ' &#xb124;&#xc784;&#xc2a4;&#xd398;&#xc774;&#xc2a4; &#xc811;&#xb450;&#xc0ac;' &#xc778;&#xc218;&#xb97c; &#xc0ac;&#xc6a9;&#xd560; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc774; &#xc778;&#xc218;&#xac00; NULL&#xc774;&#xac70;&#xb098; &#xc0dd;&#xb7b5;&#xb41c; &#xacbd;&#xc6b0; 'gml' &#xc811;&#xb450;&#xc0ac;&#xb97c; &#xc501;&#xb2c8;&#xb2e4;.
```

1.3.2 ಄전부터 사용할 수 있습니다.

1.5.0 버전부터 지리형을 지원합니&#xb2e

[illegible]

arg)를 기본값으로 씁니다.

ਜ Ġ 항.: 2.1.0 ஄ 전 &#bd80; &#d130; GML 3 를 위 해 ID 를 지 원 하 기 시 작 했 습 니 다.

**Note**

ST_AsGML (ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));



This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 17.2



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).

ST_AsGML (ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));

```
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
>
```

ST_AsGML (ST_GeomFromText('POINT(5.234234233242 6.34534534534)',4326), 5, 17);

```
-- ST_AsGML (ST_GeomFromText('POINT(5.234234233242 6.34534534534)',4326), 5, 17);
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
>
```

```
-- ST_AsGML (ST_GeomFromText('LINESTRING(1 2, 3 4, 10 20)',4326), 5, 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
>
```

```
-- &#xc0c1;&#xc790;(envelope)&#xb97c; &#xcd9c;&#xb825;&#xd558;&#xace0; (32), ←  
    &#xc88c;&#xd45c;&#xb97c; &#xb4a4;&#xc9d1;&#xace0; (&xacbd;&#xb3c4; &#xc704;&#xb3c4; ←  
    &#xb300;&#xc2e0; &#xc704;&#xb3c4; &#xacbd;&#xb3c4;) (16), long SRS&#xb97c; ←  
    &#xc0ac;&#xc6a9; (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  
>  
  
-- &#xb2e4;&#xba74;&#xccb4; &#xc608;&#xc2dc; --  
SELECT ST_AsGML(3, ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0) ←  
    ),  
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),  
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),  
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )' ));  
      st_asgml  
      -----  
  
<gml:PolyhedralSurface>  
<gml:polygonPatches>  
  <gml:PolygonPatch>  
    <gml:exterior>  
      <gml:LinearRing>  
        <gml:posList srsDimension="3"  
>0 0 0 0 0 1 0 1 1 0 1 0 0 0 0</gml:posList>  
      </gml:LinearRing>  
    </gml:exterior>  
  </gml:PolygonPatch>  
  <gml:PolygonPatch>  
    <gml:exterior>  
      <gml:LinearRing>  
        <gml:posList srsDimension="3"  
>0 0 0 0 1 0 1 1 0 1 0 0 0 0 0</gml:posList>  
      </gml:LinearRing>  
    </gml:exterior>  
  </gml:PolygonPatch>  
  <gml:PolygonPatch>  
    <gml:exterior>  
      <gml:LinearRing>  
        <gml:posList srsDimension="3"  
>0 0 0 1 0 0 1 0 1 0 0 1 0 0 0</gml:posList>  
      </gml:LinearRing>  
    </gml:exterior>  
  </gml:PolygonPatch>  
  <gml:PolygonPatch>  
    <gml:exterior>  
      <gml:LinearRing>  
        <gml:posList srsDimension="3"  
>1 1 0 1 1 1 1 0 1 1 0 0 1 1 0</gml:posList>  
      </gml:LinearRing>  
    </gml:exterior>  
  </gml:PolygonPatch>  
</gml:PolygonPatch>  
  <gml:exterior>  
    <gml:LinearRing>  
      <gml:posList srsDimension="3"
```

```
>0 1 0 0 1 1 1 1 1 1 1 0</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>
>
```

స고

ST GeomFromGML

8.9.3.6 ST AsKML

ST_AsKML — [#b3c4;#xd615;#xc744](#); GML 2 [#b610;#xb294](#); GML 3 [#xbc84;#xc804](#); [#xc694;#xc18c;#xb85c;#xbc18;#xd658;#xd569;#xb2c8;#xb2e4](#)..

Synopsis

```
text ST_AsKML(geometry geom, integer maxdecimaldigits=15, text nprefix=NULL);
text ST_AsKML(geography geog, integer maxdecimaldigits=15, text nprefix=NULL);
```

설명

도형을 KML(Keyhole Markup Language) 요소로 반환합&#xb2c8
이 함수에는 몇 가지 변종이 있습이
킜력물에 쓰이는 소수점 이하 자이
൜대값(기본값은 15), 버전의 기본값
2이며 기본 네임스페이스는 접두이
쓰지 않습니다.



Warning

Using the `maxdecimaldigits` parameter can cause output geometry to become invalid. To avoid this use `ST_ReducePrecision` with a suitable gridsize first.



Note

PostGIS를 Proj 지원과 함께 컴파일해야
합니다. Proj 지원과 함께
컴파일됐는지 여부를 확인하려&
PostGIS Full Version 을 이용하십시오.

**Note**

1.2.2 ¬용할 수
 ވ습니다. 전 파라미터를
 포함하는 최신 변종은 1.3.2
 전전부터 등장했습니다.

**Note**

개선 사항; 2.0.0 전전부터
 네임스페이스에 접두사를
 추가할 수 있습니다. 기본값은
 접두사를 쓰지 않는 것입니다.

**Note**

Changed: 3.0.0 - Removed the "versioned" variant signature

**Note**

ST_AsKML 출력물은 SRID가 없는
 도형과는 동작하지 않을
 것입니다.



This function supports 3d and will not drop the z-index.

예시

```
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 &#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;
      SELECT ST_AsKML('SRID=4326;LINESTRING(1 2 3, 4 5 6)');
      <LineString
><coordinates
>1,2,3 4,5,6</coordinates
></LineString>
```

참고

ST_AsSVG, ST_AsGML


```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D\textdegree{}M'S.SSS"C'));
      st_aslatlonTEXT
-----
2\textdegree{}19'29.928"S 3\textdegree{}14'3.243"W
```

D, M, S, C ఏ .이 아닌 다른 &#bb38;자들은 그냥 &#bb34;시됩니다.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D degrees, M minutes, S seconds to the C'));
      st_aslatlonTEXT
-----
2 degrees, 19 minutes, 30 seconds to the S 3 degrees, 14 minutes, 3 seconds to the W
```

기본방향 대신 부호가 붙은 도를 사용합니다.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D\textdegree{}M'S.SSS"'));
      st_aslatlonTEXT
-----
-2\textdegree{}19'29.928" -3\textdegree{}14'3.243"
```

소수점이 붙은 도를 사용합니다.

```
SELECT (ST_AsLatLonText('POINT (-3.2342342 -2.32498)', 'D.DDDD degrees C'));
      st_aslatlonTEXT
-----
2.3250 degrees S 3.2342 degrees W
```

지나치게 큰 값은 정궼화됩니다.

```
SELECT (ST_AsLatLonText('POINT (-302.2342342 -792.32498)'));
      st_aslatlonTEXT
-----
72\textdegree{}19'29.928"S 57\textdegree{}45'56.757"E
```

8.9.3.8 ST_AsMARC21

ST_AsMARC21 — Returns geometry as a MARC21/XML record with a geographic datafield (034).

Synopsis

```
text ST_AsMARC21 ( geometry geom , text format='hddmmss' );
```

설명

This function returns a MARC21/XML record with **Coded Cartographic Mathematical Data** representing the bounding box of a given geometry. The `format` parameter allows to encode the coordinates in subfields \$d,\$e,\$f and \$g in all formats supported by the MARC21/XML standard. Valid formats are:

- cardinal direction, degrees, minutes and seconds (default): hddmmss
- decimal degrees with cardinal direction: hddd.dddddd
- decimal degrees without cardinal direction: ddd.dddddd
- decimal minutes with cardinal direction: hddmm.mmmm

- decimal minutes without cardinal direction: dddmm.mmmm
- decimal seconds with cardinal direction: hddmmss.sss

The decimal sign may be also a comma, e.g. hddmm,mmm.

The precision of decimal formats can be limited by the number of characters after the decimal sign, e.g. hddmm.mm for decimal minutes with a precision of two decimals.

This function ignores the Z and M dimensions.

LOC MARC21/XML versions supported:

- **MARC21/XML 1.1**

Availability: 3.3.0



Note

This function does not support non lon/lat geometries, as they are not supported by the MARC21/XML standard (Coded Cartographic Mathematical Data).



Note

The MARC21/XML Standard does not provide any means to annotate the spatial reference system for Coded Cartographic Mathematical Data, which means that this information will be lost after conversion to MARC21/XML.

Example 1: POINT

Converting a POINT to MARC21/XML formatted as hddmmss (default)

```
SELECT ST_AsMARC21('SRID=4326;POINT(-4.504289 54.253312)::geometry');

          st_asmarc21
-----
<record xmlns="http://www.loc.gov/MARC21/slim">
  <datafield tag="034" ind1="1" ind2=" " >
    <subfield code="a">a</subfield>
    <subfield code="d">W0043015</subfield>
    <subfield code="e">W0043015</subfield>
    <subfield code="f">N0541512</subfield>
    <subfield code="g">N0541512</subfield>
  </datafield>
</record>
```

Converting a POLYGON to MARC21/XML formatted in decimal degrees

```
SELECT ST_AsMARC21('SRID=4326;POLYGON((-4.5792388916015625 54.18172660239091,
54.18172660239091,-4.56756591796875 54.196993557130355,-4.546623229980469
54.18313300502024,-4.5792388916015625 54.18172660239091))::geometry, 'hddd.dddd');

          st_asmarc21
-----
<record xmlns="http://www.loc.gov/MARC21/slim">
  <datafield tag="034" ind1="1" ind2=" " >
```



```

        <subfield code="a">a</subfield>
        <subfield code="d">W004.5792</subfield>
        <subfield code="e">W004.5466</subfield>
        <subfield code="f">N054.1970</subfield>
        <subfield code="g">N054.1817</subfield>
    </datafield>
</record>

```

Converting a `GEOMETRYCOLLECTION` to MARC21/XML formatted in decimal minutes. The geometries order in the MARC21/XML output correspond to their order in the collection.

```

SELECT ST_AsMARC21 ('SRID=4326;GEOMETRYCOLLECTION(POLYGON((13.1 52.65,13.516666666666667 52.65,13.516666666666667 52.383333333333333,13.1 52.383333333333333,13.1 52.65)),POINT(-4.5 54.25))':geometry,'hddmm.
mmmm');

          st_asmarc21
-----
<record xmlns="http://www.loc.gov/MARC21/slim">
  <datafield tag="034" ind1="1" ind2=" ">
    <subfield code="a">a</subfield>
    <subfield code="d">E01307.0000</subfield>
    <subfield code="e">E01331.0000</subfield>
    <subfield code="f">N05240.0000</subfield>
    <subfield code="g">N05224.0000</subfield>
  </datafield>
  <datafield tag="034" ind1="1" ind2=" ">
    <subfield code="a">a</subfield>
    <subfield code="d">W00430.0000</subfield>
    <subfield code="e">W00430.0000</subfield>
    <subfield code="f">N05415.0000</subfield>
    <subfield code="g">N05415.0000</subfield>
  </datafield>
</record>

```

ST_GeomFromMARC21

ST_GeomFromMARC21

8.9.3.9 ST_AsMVTGeom

ST_AsMVTGeom — Transforms a geometry into the coordinate space of a MVT tile.

Synopsis

geometry **ST_AsMVTGeom**(geometry geom, box2d bounds, integer extent=4096, integer buffer=256, boolean clip_geom=true);

ST_AsMVTGeom

Transforms a geometry into the coordinate space of a MVT (**Mapbox Vector Tile**) tile, clipping it to the tile bounds if required. The geometry must be in the coordinate system of the target map (using **ST_Transform** if needed). Commonly this is **Web Mercator** (SRID:3857).

The function attempts to preserve geometry validity, and corrects it if needed. This may cause the result geometry to collapse to a lower dimension.

The rectangular bounds of the tile in the target map coordinate space must be provided, so the geometry can be transformed, and clipped if required. The bounds can be generated using [ST_MakeEnvelope](#).

This function is used to convert geometry into the tile coordinate space required by [ST_AsMVT](#).

`geom` is the geometry to transform, in the coordinate system of the target map.

`bounds` is the rectangular bounds of the tile in map coordinate space, with no buffer.

`extent` is the tile extent size in tile coordinate space as defined by the [MVT specification](#). Defaults to 4096.

`buffer` is the buffer size in tile coordinate space for geometry clipping. Defaults to 256.

`clip_geom` is a boolean to control if geometries are clipped or encoded as-is. Defaults to true.

Availability: 2.4.0



Note

From 3.0, Wagyu can be chosen at configure time to clip and validate MVT polygons. This library is faster and produces more correct results than the GEOS default, but it might drop small polygons.

Canonical example

```
SELECT ST_AsText(ST_AsMVTGeom(
    ST_GeomFromText('POLYGON ((0 0, 10 0, 10 5, 0 -5, 0 0))'),
    ST_MakeBox2D(ST_Point(0, 0), ST_Point(4096, 4096)),
    4096, 0, false));
           st_astext
-----
MULTIPOLYGON(((5 4096,10 4091,10 4096,5 4096)),((5 4096,0 4101,0 4096,5 4096)))
```

Canonical example for a Web Mercator tile using a computed tile bounds to query and clip geometry.

```
SELECT ST_AsMVTGeom(
    ST_Transform( geom, 3857 ),
    ST_TileEnvelope(12, 513, 412), extent => 4096, buffer => 64) AS geom
FROM data
WHERE geom && ST_TileEnvelope(12, 513, 412, margin => (64.0 / 4096))
```

ST_AsMVT

[ST_AsMVT](#), [ST_MakeEnvelope](#), [PostGIS_Wagyu_Version](#)

8.9.3.10 ST_AsMVT

ST_AsMVT — Aggregate function returning a MVT representation of a set of rows.

Synopsis

```
bytea ST_AsMVT(anyelement set row);
bytea ST_AsMVT(anyelement row, text name);
bytea ST_AsMVT(anyelement row, text name, integer extent);
bytea ST_AsMVT(anyelement row, text name, integer extent, text geom_name);
bytea ST_AsMVT(anyelement row, text name, integer extent, text geom_name, text feature_id_name);
```

ST_AsMVT

An aggregate function which returns a binary **Mapbox Vector Tile** representation of a set of rows corresponding to a tile layer. The rows must contain a geometry column which will be encoded as a feature geometry. The geometry must be in tile coordinate space and valid as per the **MVT specification**. **ST_AsMVTGeom** can be used to transform geometry into tile coordinate space. Other row columns are encoded as feature attributes.

The **Mapbox Vector Tile** format can store features with varying sets of attributes. To use this capability supply a JSONB column in the row data containing Json objects one level deep. The keys and values in the JSONB values will be encoded as feature attributes.

Tiles with multiple layers can be created by concatenating multiple calls to this function using `||` or `STRING_AGG`.



Important

Do not call with a `GEOMETRYCOLLECTION` as an element in the row. However you can use **ST_AsMVTGeom** to prepare a geometry collection for inclusion.

`row` row data with at least a geometry column.

`name` is the name of the layer. Default is the string "default".

`extent` is the tile extent in screen space as defined by the specification. Default is 4096.

`geom_name` is the name of the geometry column in the row data. Default is the first geometry column. Note that PostgreSQL by default automatically **folds unquoted identifiers to lower case**, which means that unless the geometry column is quoted, e.g. "MyMVTGeom", this parameter must be provided as lowercase.

`feature_id_name` is the name of the Feature ID column in the row data. If NULL or negative the Feature ID is not set. The first column matching name and valid type (smallint, integer, bigint) will be used as Feature ID, and any subsequent column will be added as a property. JSON properties are not supported.

Enhanced: 3.0 - added support for Feature ID.

Enhanced: 2.5.0 - added support parallel query.

Availability: 2.4.0

ST_AsMVTGeom

```
WITH mvtgeom AS
(
  SELECT ST_AsMVTGeom(geom, ST_TileEnvelope(12, 513, 412), extent => 4096, buffer => 64) AS geom,
         name, description
  FROM points_of_interest
  WHERE geom && ST_TileEnvelope(12, 513, 412, margin => (64.0 / 4096))
)
SELECT ST_AsMVT(mvtgeom.*)
FROM mvtgeom;
```

ST_MakeEnvelope

ST_AsMVTGeom, **ST_MakeEnvelope**

8.9.3.11 ST_AsSVG

ST_AsSVG — Returns SVG path data for a geometry.

Synopsis

text **ST_AsSVG**(geometry geom, integer rel=0, integer maxdecimaldigits=15);
 text **ST_AsSVG**(geography geog, integer rel=0, integer maxdecimaldigits=15);

설명

도형을 SVG(Scalar Vector Graphics) 경로 데이터로 반상대 이동(relative move) 관점에서 구현된 경로 데이터를 얻으려면 두 번째 인수로 1을 사용하고, 절대 이동(absolute move) 관점에서 구현된 경로 데이터를 얻으려면 기본값 0을 사용하십시출력물에 쓰인 소수점 이하 자릿최대값(기본값은 15)을 줄이는 데 세번째 인수를 쓸 수도 있습니다.. 'rel' 독립 변수가 0일 때 포인트 도형을 cx/cy로 렌더링하고, 'rel' 독립 변수가 1일 때 x/y로 렌더링합니다.. 멀티포인트 도형들은 쉼표(",")로 구분하고, 도형집합의 도형들은 쌍반점(",")으로 구

Note



1.2.2 버전부터 사용할 수 있습니다.. 1.4.0버전에서 <http://www.w3.org/TR/SVG/paths.html#PathDataBNF> 를 준수하기 위해 절대 경로에 L 명령어를 포함시켰습니다.

변경 사항.. 2.0.0 버전부터 기본 독립 변수(default arg)를 사용하고 명명된 독립 변수(named arg)를 지원합니다..

예시

```
SELECT ST_AsSVG('POLYGON((0 0,0 1,1 1,1 0,0 0))');

      st_assvg
      -
M 0 0 L 0 -1 1 -1 1 0 Z
```

8.9.3.12 ST_AsTWKB

ST_AsTWKB — 도형을 TWKB(Tiny Well-Known Binary)로 반환합니&

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);

설명

도형을 TWKB(Tiny Well-Known Binary) 형식으로 반환합 TWKB는 출력물의 용량을 최소화하는 데 집중한 압축 &#bc14;이너리 형식 입소수점 이하 자릿수 파라미터가 출력물에 어느 정도의 정밀도를 저장할지 결정합니다 기본적으모든 값은 인코딩하기 전에 가장 가까운 단위로 &#bc18;올림됩니다 더 높은 정밀도를 &#bcf5;사하ૠ 싶다면 자릿수를 올리십시오 예를 들어 값이 1이라면 소수점 오른쪽의 &#xc-cab; &#bc88;째 숫자까지 &#bcf4;전될 ಃ입니다 크기 &#bc0f; ઽ 계 상자 파라미터는 객인코딩된 길이 &#bc0f; ઽ 계에 대한 선정보를 출력물에 포앨시킬지 말결정합니다 기본적으로는 포함않습니다 사용자의 클라이언트 소프트웨어가 필요로하지 않는활성화시키지 마십시오 디스크 공간을 소비할 &#bfd0;입니다 디스크 공간을 절약하는 ಃ이 TWKB의 &#baa9;적이 함수의 &#bc30;열 입력 형식은 도형 집합 &#bc0f; 유일 식별자를 식별자를 &#bcf4;전하는 TWKB집합으로 &#bcc0;환하는 데 쓰입니다 집합의 압축을 풀어 그 안에 있는 객체들에 대한 상세 정보에 접근하는 기능을 가진 클유용합니다 **array_agg** 함수를 이용해서 &#bc30;열을 생성할 수 있습니다 다른 파라미터들은 이 함수의 &#b2e8;순 형ઽ 우와 동일하ಌ 실행됩니다.

Note

<https://github.com/TWKB/Specification> 에서 형식 사양서를 찾아௏ 수 있으며 <https://github.com/-TWKB/twkb.js> 에서 자보스크립트 클라이언트를 빌드하기 위한 코드를 찾을 수 있습니다.

Enhanced: 2.4.0 memory and speed improvements.

2.2.0 ஄전부터 사용할 수 있습니다.

예시

```
SELECT ST_AsTWKB('LINESTRING(1 1,5 5)')::geometry);
          st_astwkb
-----
\x02000202020808
```

식별자를 포함하는 종합 TWKB 객쮴&#xb 생성하려면 먼저 "array_agg()"를 통해 원&#xd55

```
SELECT ST_AsTWKB(array_agg(geom), array_agg(gid)) FROM mytable;
```

స고

8.9.3.13 ST AsX3D

Synopsis

설&#xba85:

Note



Note!

'ص션' 인수는 비트필드 구조체입& PostGIS 2.2 이상 버전의 경우,좌표를 X3D 지&#xb 지리공ఄ 노드로 표현할지,그리&#xb 또 y/y 축을 뒤집을지 말지를 표시&#xd558 데 쓰입니&#xb2e4.; 기본적으로 ST_AsX3D 는 데 형식 (long,lat or X,Y)로 출력하지만, X3D의 기본& #위도/경&#xb3c4.; y/x를 선호할 수도 있습&#xb2c

- 0: X/Y(데이터베이스 순서로 정렬된 X/Y(예를 들어 경도/위도 = X,Y가 표준 데이터베이스 순서입니다), 데이ఒ,비(非)공간 좌표(구식 정귝 좌표태그).
- 1: X 및 Y를 뒤집습니다,지리좌표(GeoCoordinate 옵션 스위치와 연결해서 사용할 경우,출력물이 기본값인 "latitude_first"(위도먼저)가 될 것이고 좌표도 마찬가 뒤집힐 것입니다.
- 2: 좌표를 지리공간 지리좌표로 출도형이 WGS84 경위도(SRID 4326)가 아닐 경우 이 옵션은 오류를 &#xbcl;생시킬 것입현재 이 지리좌표 유형만 지원합공간 참조 시스템을 정의하는 X3D 사양 을 참조하십시오,출력물은 기본적으로 GeoCoordinate geoSystem=' "GD" "WE" "longitude_first"' 가 됩니다 X3D 기본값인 GeoCoordinate geoSystem=' "GD" "WE" "latitude_first"' 를 선호한다면,(2 + 1) = 3 을 쓰다.

PostGIS 유형	2D X3D 유형	3D X3D 유형
LINESTRING	아직 구현되지 않았습니다. 구현시 PolyLine2D가 될 것입니다.	다. LineSet
MULTILINESTRING	아직 구현되지 않았습니다. 구현시 PolyLine2D가 될 것입니다.	다. IndexedLineSet
MULTIPOINT	Polypoint2D	PointSet
POINT	공백으로 구분된 좌표를 출력합니다.	공백으로 구분된 좌표를 출력합니다.
(MULTI) POLYGON, POLYHEDRALSURFACE	유효하지 않은 X3D 마크업(markup)입니다.	IndexedFaceSet (현재 내곽 고리를 또다른 입니다. 집합(faceset)으로 출력합니다.
TIN	TriangleSet2D (아직 구현되지 않았습니다.)	IndexedTriangleSet

Note



2న원 도형 지원이 아직
완성되지 않았습니다. 현재
내󊳽 고리를 그저 개별적인
폴리곤으로 그립니다. 이
부분은 현재 작업중입니다.

Lots of advancements happening in 3D space particularly with **X3D Integration with HTML5**

```
&#xb610;&#xb80c;&#xb354;&#xb9c1;&#xb41c;&#xb3c4;&#xd615;&#xc744;&#xbcf4;&#xb294;&#xb370;&#xc774;&#xc6a9;&#xc218;&#xc788;&#xb294;&#xba4b;&#xc9c4;&#xc624;&#xd508;&#xc18c;&#xc2a4; X3D&#xbdf0;&#xc5b4;&#xac00;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xb9e5;. &#xb9ac;&#xb205;&#xc2a4;. &#xc708;&#xb3c4;&#xc6b0;&#xc6a9; Free Wrl&#xbcf4;&#xc774;&#xb108;&#xb9ac;&#xb97c; http://freewrl.sourceforge.net/ &#xc5d0;&#xc11c;&#xb2e4;&#xc6b4;&#xb85c;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xb3c4;&#xd615;&#xc744;&#xbcf4;&#xae30;&#xc704;&#xd574;&#xd328;&#xc774;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

Also check out [PostGIS minimalist X3D viewer](#) that utilizes this function and [x3dDom html/js open source toolkit](#).

2.0.0 ಄전஀터 ISO-IEC-19776-1.2-X3DEncodings-XML을 이용할
수 있습니다.

개선 사항: 2.2.0 버전부터 지리좌표
및축(x/y, 경도/위도) 뒤집기를 지원&#xd56
자세한 내용은 옵션을 살펴보십&#xc

- ✔ 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).

예시: 완전히 기능하는 X3D 문서를
생성합니다. 다음 예시는 FreeWrl 및 다는
X3D 뷰어에서 ௏ 수 있는 정육면체를
생성.할 것입니다.

```
SELECT '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d ↵
-3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
        <Appearance>
          <Material emissiveColor=''0 0 1''/>
        </Appearance>
> ' ||
      ST_AsX3D( ST_GeomFromEWKT('POLYHEDRALSURFACE( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )' ) ) ||
      '</Shape>
    </Transform>
  </Scene>
</X3D>
>' As x3ddoc;
```

x3ddoc


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d
-3.0.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
        <Appearance>
          <Material emissiveColor='0 0 1' />
        </Appearance>
        <IndexedFaceSet coordIndex='0 1 2 3 -1 4 5 6 7 -1 8 9 10 11 -1 12 13 14 15 -1 16 17
18 19 -1 20 21 22 23'>
          <Coordinate point='0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1
1 0 1 1' />
        </IndexedFaceSet>
      </Shape>
    </Transform>
  </Scene>
</X3D>
>
```

PostGIS buildings

Copy and paste the output of this query to [x3d scene viewer](#) and click Show

```
SELECT string_agg('<Shape>' || ST_AsX3D(ST_Extrude(geom, 0,0, i*0.5)) ||
    '<Appearance>'
        '<Material diffuseColor="' || (0.01*i)::text || ' 0.8 0.2" specularColor="' ||
        (0.05*i)::text || ' 0 0.5"/>'
    </Appearance>'
    </Shape>', '')
FROM ST_Subdivide(ST_Letters('PostGIS'),20) WITH ORDINALITY AS f(geom,i);
```



Buildings formed by subdividing PostGIS and extrusion

```
&#xc608;&#xc2dc:: &#xc18c;&#xc218;&#xc810; &#xc815;&#xbc00;&#xb3c4; 6&#xc774;&#xba70; 3&#xb2e8;&#xc704;
&#xc0c1;&#xc2b9;&#xd55c; &#xd314;&#xac01;&#xd615;
```

```
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>
>
```

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 1 0 1 1 0' /></IndexedTriangleSet>
>
```

MultilineString

```
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>
>
```

8.9.3.14 ST_GeoHash

ST_GeoHash — Returns the GeoHash of a geometry.

Synopsis

text **ST_GeoHash**(geometry geom, integer maxchars=full_precision_of_point);

8.10 `&&` (operator)

8.10.1 Bounding Box Operators

8.10.1.1 `&&`

`&&` — A 2D bounding box; B 2D bounding box; A 2D bounding box; B 2D bounding box; TRUE; FALSE;

Synopsis

boolean `&&`(geometry A , geometry B);
boolean `&&`(geography A , geography B);

Notes:

`&&` returns TRUE if the bounding boxes of A and B overlap, FALSE otherwise. The bounding box of a geometry is the minimum bounding rectangle (MBR) that contains the geometry. The bounding box of a geography is the minimum bounding rectangle (MBR) that contains the geography.



Note

`&&` is implemented using the `ST_Contains` function. The `ST_Contains` function returns TRUE if the geometry A contains the geometry B, FALSE otherwise. The `ST_Contains` function is implemented using the `ST_Contains` function.

`&&` is implemented using the `ST_Contains` function. The `ST_Contains` function returns TRUE if the geometry A contains the geometry B, FALSE otherwise. The `ST_Contains` function is implemented using the `ST_Contains` function.

1.5.0 and later: `&&` is implemented using the `ST_Contains` function. The `ST_Contains` function returns TRUE if the geometry A contains the geometry B, FALSE otherwise. The `ST_Contains` function is implemented using the `ST_Contains` function.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

Example:

```
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)
```

Related Functions:

`ST_Intersects`, `ST_Extent`, `|>`, `<`, `<|`, `<`, `~`, `@`

8.10.1.2 `&&(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 );`

`설명`

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.

`예시`

```
SELECT ST_Point(1,1) && ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) AS overlaps;

overlaps
-----
t
(1 row)
```

`참고`

`&&(box2df,geometry)`, `&&(box2df,box2df)`, `~(geometry,box2df)`, `~(box2df,geometry)`, `~(box2df,box2df)`, `@(geometry,box2df)`, `@(box2df,geometry)`, `@(box2df,box2df)`

8.10.1.3 `&&(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 );`

Notes

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.

Example

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_Point(1,1) AS overlaps;

overlaps
-----
t
(1 row)
```

Box2DF

`&&(geometry,box2df)`, `&&(box2df,box2df)`, `~(geometry,box2df)`, `~(box2df,geometry)`, `~(box2df,box2df)`, `@(geometry,box2df)`, `@(box2df,geometry)`, `@(box2df,box2df)`

8.10.1.4 &&(box2df,box2df)

`&&(box2df,box2df)` — Returns `TRUE` if two 2D float precision bounding boxes (`BOX2DF`) intersect each other.

Synopsis

boolean `&&( box2df A , box2df B )`;

Notes

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.

overlaps;

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) && ST_MakeBox2D(ST_Point(1,1), ST_Point(
    (3,3)) AS overlaps;

overlaps
-----
t
(1 row)
```

overlaps_3d;

overlaps_3d(geometry, box2df), overlaps_3d(box2df, geometry), overlaps_3d(geometry, box2df), overlaps_3d(box2df, geometry), overlaps_3d(box2df, box2df), overlaps_3d(geometry, box2df), overlaps_3d(box2df, geometry), overlaps_3d(box2df, box2df)

8.10.1.5 overlaps_3d

overlaps_3d — Returns true if the 3D bounding boxes of the two geometries overlap. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry.

Synopsis

boolean **overlaps_3d**(geometry A , geometry B);

overlaps_3d;

overlaps_3d — Returns true if the 3D bounding boxes of the two geometries overlap. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry.

**Note**

overlaps_3d — Returns true if the 3D bounding boxes of the two geometries overlap. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry.

overlaps_3d — Returns true if the 3D bounding boxes of the two geometries overlap. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry. The bounding box of a geometry is the minimum 3D box that contains the geometry.



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.

overlaps_3d:: 3d overlaps_3d; 3d overlaps_3d; 3d overlaps_3d; 3d overlaps_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

3DM 라인스트링

```
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

참고

&&

8.10.1.6 &&(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);

설명

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.

Example 1:

```
SELECT ST_MakePoint(1,1,1) &&& ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) AS overlaps;

overlaps
-----
t
(1 row)
```

Example 2:

`&&&(gidx,geometry), &&&(gidx,gidx)`

8.10.1.7 `&&&(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 );`

Example 3:

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.

Example 4:

```
SELECT ST_3DMakeBox(ST_MakePoint(0,0,0), ST_MakePoint(2,2,2)) &&& ST_MakePoint(1,1,1) AS overlaps;

overlaps
-----
t
(1 row)
```

ST_3DMakeBox

ST_3DMakeBox(geometry gidx, geometry gidx)

8.10.1.8 ST_3DMakeBox(gidx, gidx)

ST_3DMakeBox(gidx, gidx) — Returns TRUE if two n-D float precision bounding boxes (GIDX) intersect each other.

Synopsis

boolean **ST_3DMakeBox**(gidx A , gidx B);

ST_3DMakeBox

The **ST_3DMakeBox** 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.

ST_3DMakeBox

```
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)
```

ST_3DMakeBox

ST_3DMakeBox(geometry gidx, geometry gidx)

8.10.1.9 ST_3DMakeBox

ST_3DMakeBox — A 3D bounding box is created from the two input geometries. If the two input geometries are 3D, the bounding box is 3D. If the two input geometries are 2D, the bounding box is 2D. If the two input geometries are 1D, the bounding box is 1D. If the two input geometries are 0D, the bounding box is 0D. TRUE if the two input geometries intersect, FALSE otherwise.

Synopsis

boolean **<**(geometry A , geometry B);

Notes

< returns true if the two geometries overlap. The geometries must be of the same dimension. For example, a point cannot overlap a line segment. The function returns false if the two geometries are disjoint, or if one is contained within the other. The function returns true if the two geometries are identical, or if one is a subset of the other. The function returns false if the two geometries are of different dimensions. For example, a point cannot overlap a line segment. The function returns false if the two geometries are disjoint, or if one is contained within the other. The function returns true if the two geometries are identical, or if one is a subset of the other. The function returns false if the two geometries are of different dimensions.



Note

The function **<** is commutative, meaning that **<**(A,B) is equivalent to **<**(B,A). The function **<** is also transitive, meaning that if A overlaps B, and B overlaps C, then A overlaps C. The function **<** is not associative, meaning that **<**(A,(B,C)) is not necessarily equivalent to (**<**(A,B),C). The function **<** is not reflexive, meaning that **<**(A,A) is not necessarily true. The function **<** is not symmetric, meaning that **<**(A,B) is not necessarily equivalent to **<**(B,A).

Examples

```
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)

See Also

[ST_Contains](#), [ST_ContainsProperly](#), [ST_Overlaps](#), [ST_Touches](#)

8.10.1.10 <

< — A overlaps B. The function returns true if the two geometries overlap. The geometries must be of the same dimension. For example, a point cannot overlap a line segment. The function returns false if the two geometries are disjoint, or if one is contained within the other. The function returns true if the two geometries are identical, or if one is a subset of the other. The function returns false if the two geometries are of different dimensions.

Synopsis

boolean **<**(geometry A , geometry B);

ST_Crosses

`<|` `geom1`, `geom2` `boolean` `ST_Crosses` (`geom1`, `geom2`)

ST_Crosses returns true if the two geometries cross, false otherwise. Two geometries cross if they have at least one point in common, but neither is completely contained by the other.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

**Note**

ST_Crosses returns true if the two geometries cross, false otherwise. Two geometries cross if they have at least one point in common, but neither is completely contained by the other.

ST_Overlaps

```
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)

ST_Contains

`&&`, `l`, `>`, `<`

8.10.1.11 ST_Contains

`>` — A `boolean` `ST_Contains` (`geom1`, `geom2`)

ST_Contains returns true if the first geometry contains the second geometry, false otherwise.

Synopsis

boolean `ST_Contains` (`geom1`, `geom2`);

설명

```
&#x5f0;&#xc0b0;&#xc790;&#xb294; &#xb3c4;&#xd615; A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00;
&#xb3c4;&#xd615; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc640; &#xacb9;&#xce58;&#xac70;&#xb098; &#xadf8;
&#xc624;&#xb978;&#xcabd;&#xc5d0; &#xc788;&#xc744; &#xacbd;&#xc6b0;., &#xb610;&#xb294; &#xb354; &#xc815;&#xd655;&
&#xb9d0;&#xd558;&#xc790;&#xba74; &#xb3c4;&#xd615; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc640; &#xacb9;
&#xadf8; &#xc67c;&#xcabd;&#xc5d0; &#xc788;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; TRUE &#xb97c; &#xbc18;&#xd
```



Note

```
&#xc774;      &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294;      &#xb3c4;&#xd615;&#xc5d0;&#xc11c;
&#xc774;&#xc6a9;&#xd560;      &#xc218;&#xb3c4;      &#xc788;&#xb294;      &#xbaa8;&#xb4e0;
&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xd65c;&#xc6a9;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

예시

```
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)

```

సೠ

$\&\&$, $|\&\rangle$, $\&\langle|$, $\&\langle$

8.10.1.12 <<

```
<<— A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00; &#xc624;&#xb85c;&#xc9c0; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790; &#xc67c;&#xcabd;&#xc5d0; &#xc788;&#xc744; &#xacbd;&#xc6b0;&#xc5d0;&#xb9cc; TRUE &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

Synopsis

```
boolean <<( geometry A , geometry B );
```

설명

```
<< &#xc5f0;&#xc0b0;&#xc790;&#xb294; &#xb3c4;&#xd615; A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00;
&#xc624;&#xb85c;&#xc9c0; &#xb3c4;&#xd615; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790; &#xc67c;&#xcabd;&#xc5d0;
&#xc788;&#xc744; &#xacbd;&#xc6b0;&#xc5d0;&#xb9cc; TRUE &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```



Note

```
&#xc774;      &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294;      &#xb3c4;&#xd615;&#xc5d0;&#xc11c;
&#xc774;&#xc6a9;&#xd560;      &#xc218;&#xb3c4;      &#xc788;&#xb294;      &#xbaa8;&#xb4e0;
&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xd65c;&#xc6a9;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

Example 1

```
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)
```

Example 2

>>, >>, <<

8.10.1.13 <<|

<<| — A << B; A is a geometry and B is a geometry. The result is a boolean value. If A is contained within B, the result is true, otherwise false.

Synopsis

boolean <<| (geometry A , geometry B);

Example 1

```
<<| 'POINT (1 2)'::geometry, 'POLYGON ((0 0, 4 3, 6 5, 2 2, 5 6))'::geometry;
      true
<<| 'POINT (1 2)'::geometry, 'POLYGON ((0 0, 4 3, 6 5, 2 2, 5 6))'::geometry;
      true
<<| 'POINT (1 2)'::geometry, 'POLYGON ((0 0, 4 3, 6 5, 2 2, 5 6))'::geometry;
      true
```

**Note**

The <<| operator is a convenience operator for the ST_Contains function. The result is a boolean value. If A is contained within B, the result is true, otherwise false.

Example 2

```
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)

<<, >>, |>>

8.10.1.14 =

= — 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 );
```

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.

```
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

```

ST_Equals

ST_OrderingEquals, **~=**

8.10.1.15 >>

>> — A ݘ 경 계 상 자 가 오 로 지 B ݘ 경 계 상 자 오 른 쪽 에 있 을 경 우 에 만 TRUE 를 반 환 합 니 다.

Synopsis

boolean >>(geometry A , geometry B);

ST_Equals

>> 연 산 자 는 도 형 A ݘ 경 계 상 자 가 오 로 지 도 형 B ݘ 경 계 상 자 오 른 쪽 있 을 경 우 에 만 TRUE 를 반 환 합 니 다.



Note

이 피 연 산 자 (operand) 는 도 형 에 서
 이 용 할 수 도 있 는 모 든
 인 덱 스 를 활 용 할 것 입 니 다.

8.10.1.15

```
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)

8.10.1.16

<<, >>, <<

8.10.1.16 @

@ — B ݘ 경 계 상 자 가 A ݘ 경 계 상 자 를 담 고 있 을 경 우 TRUE 를 반 환 합 니 다.

Synopsis

boolean @(geometry A , geometry B);

8.10.1.17

@ 연 산 자 는 도 형 B ݘ 경 계 상 자 가 도 형 A ݘ 경 계 상 자 를 완 전 히 담 고 있 을 경 우 TRUE 를 반 환 합 니 다.

**Note**

이 피 연 산 자 (operand) 는 도 형 에 서 이 용 할 수 도 있 는 모 든 인 덱 스 를 활 용 할 것 입 니 다.

8.10.1.18

```
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)

```

Notes:

~, &&

8.10.1.17 @ (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);

Notes:

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.

Example:

```

SELECT ST_Buffer(ST_GeomFromText('POINT(2 2)'), 1) @ ST_MakeBox2D(ST_Point(0,0), ST_Point(
  (5,5)) AS is_contained;

 is_contained
-----
t
(1 row)

```

Notes:

&&(geometry, box2df), &&(box2df, geometry), &&(box2df, box2df), ~(geometry, box2df), ~(box2df, geometry), ~(box2df, box2df), @ (box2df, geometry), @ (box2df, box2df)

8.10.1.18 @ (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);

Notes

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.

Examples

```
SELECT ST_MakeBox2D(ST_Point(2,2), ST_Point(3,3)) @ ST_Buffer(ST_GeomFromText('POINT(1 1)'), 10) AS is_contained;
```

is_contained
t

(1 row)

Other Names

`&&(geometry,box2df)`, `&&(box2df,geometry)`, `&&(box2df,box2df)`, `~(geometry,box2df)`, `~(box2df,geometry)`, `~(box2df,box2df)`, `@(geometry,box2df)`, `@(box2df,box2df)`

8.10.1.19 @ (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);

8.10.1.19

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 INdices (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

8.10.1.20

```
SELECT ST_MakeBox2D(ST_Point(2,2), ST_Point(3,3)) @ ST_MakeBox2D(ST_Point(0,0), ST_Point(5,5)) AS is_contained;

is_contained
-----
t
(1 row)
```

8.10.1.21

`&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @&&(geometry,box2df), @&&(box2df,geometry)`

8.10.1.20

`|&>` — A `box2d` or `box3d` is contained within B. Both arguments must be of type `box2d` or `box3d`. If both arguments are of type `box2d`, the result is of type `boolean`. If both arguments are of type `box3d`, the result is of type `boolean`. If one argument is of type `box2d` and the other is of type `box3d`, the result is of type `boolean`.

Synopsis

boolean `|&> ( geometry A , geometry B );`

8.10.1.21

`|&>` — A `box2d` or `box3d` is contained within B. Both arguments must be of type `box2d` or `box3d`. If both arguments are of type `box2d`, the result is of type `boolean`. If both arguments are of type `box3d`, the result is of type `boolean`. If one argument is of type `box2d` and the other is of type `box3d`, the result is of type `boolean`.

**Note**

`&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), ~(box2df,box2df), @&&(geometry,box2df), @&&(box2df,geometry)`

Example 1

```
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)
```

Example 2

Result: **TRUE**, **FALSE**, **FALSE**, **FALSE**

8.10.1.21 |>>

Description: **|>>** — A **boolean** result of the comparison of the bounding box of geometry A and geometry B. The result is **TRUE** if the bounding box of geometry A is strictly above the bounding box of geometry B.

Synopsis

boolean **|>>**(geometry A , geometry B);

Example 1

The **|>>** operator returns **TRUE** if the bounding box of geometry A is strictly above the bounding box of geometry B.

**Note**

The **|>>** operator is only valid for **geometry** and **geometry_z** types. It is not valid for **geometrycollection** or **geometrycollection_z** types. The result is **TRUE** if the bounding box of geometry A is strictly above the bounding box of geometry B.

Example 2

```
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)

```

󌰸고

<<, >>, <<|

8.10.1.22 ~

~ — A의 경계 상자가 B의 경계 상자를 담고 있을 경우 TRUE 를 반환합니다.

Synopsis

boolean ~(geometry A , geometry B);

설명

~ 연산자는 도형 A의 경계 상자가 도형 B의 경계 상자를 완전히 담고 있을 경우 TRUE 를 반환합니다.



Note

이 피연산자(operand)는 도형에서 이용할 수도 있는 모든 인덱스를 활용할 것입니다.

예시

```

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)

```

󌰸고

@, &&

8.10.1.23 ~(geometry,box2df)

~(geometry,box2df) — Returns TRUE if a geometry's 2D bounding box contains a 2D float precision bounding box (GIDX).

Synopsis

boolean `~( geometry A , box2df B );`

Notes

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 INdices (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

Examples

```
SELECT ST_Buffer(ST_GeomFromText('POINT(1 1)'), 10) ~ ST_MakeBox2D(ST_Point(0,0), ST_Point(2,2)) AS contains;

contains
-----
t
(1 row)
```

Operator Syntax

`&&(geometry,box2df), &&(box2df,geometry), &&(box2df,box2df), ~(box2df,geometry), ~(box2df,box2df), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)`

8.10.1.24 ~(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 );`

Notes

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 INdices (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

8.10.1.24

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(5,5)) ~ ST_Buffer(ST_GeomFromText('POINT(2 2)')
, 1) AS contains;
```

```
contains
-----
t
(1 row)
```

8.10.1.25

~(box2df,box2df) — Returns `TRUE` if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).

8.10.1.25

~(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 );`

8.10.1.25

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 INdices (BRIN) was introduced. Requires PostgreSQL 9.5+.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.

8.10.1.25

```
SELECT ST_MakeBox2D(ST_Point(0,0), ST_Point(5,5)) ~ ST_MakeBox2D(ST_Point(2,2), ST_Point
(3,3)) AS contains;
```

```
contains
-----
t
(1 row)
```


Geometry Operators

`&(geometry,box2df), &(box2df,geometry), &(box2df,box2df), ~(geometry,box2df), ~(box2df,geometry), @(geometry,box2df), @(box2df,geometry), @(box2df,box2df)`

8.10.1.26 ~ =

`~ =` — A ݘ 경 계 상 자 와 B ݘ 경 계 상 자 가 동 일 할 경 우 TRUE 를 반 환 합 니 다.

Synopsis

boolean `~=( geometry A , geometry B );`

Notes

`~ =` 연 산 자 는 도 형 / 지 리 형 A ݘ 경 계 상 자 와 도 형 / 지 리 형 B ݘ 경 계 상 자 동 일 할 경 우 TRUE 를 반 환 합 니 다.



Note

이 피 연 산 자 (operand) 는 도 형 에 서 이 용 할 수 도 있 는 모 든 인 덱 스 를 활 용 할 것 입 니 다.

1.5.0 버 전 에 서 습 성 이 변 경 됐 습 니 다



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](#).

Example

```
select 'LINESTRING(0 0, 1 1)::geometry ~ = 'LINESTRING(0 1, 1 0)::geometry as equality;
equality |
-----+
t       |
```

Geometry Operators

[ST_Equals](#), [ST_OrderingEquals](#), =

8.10.2 연 산 자 (operator)

8.10.2.1 <->

`<->` — A 와 B 사 이 의 2 차 원 거 리 를 반 환 합 &


```

--> 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)

PostgreSQL 9.1

```

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)

PostgreSQL 9.5

```

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)

```

--> ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)')::geometry) as d,edabbr, vaabbr
FROM va2005
ORDER BY d limit 10;

```

```

WITH index_query AS (
  SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)')::geometry) as d,edabbr, vaabbr

```

```

WITH index_query AS (
  SELECT ST_Distance(geom, 'SRID=3005;POINT(1011102 450541)')::geometry) as d,edabbr, vaabbr

```


Note



```
&#xb3c4;&#xd615; &#xac00;&#xc6b4;&#xb370; &#xd558;&#xb098;&#xac00;, &#xc608;&#xb97c; &#xb4e4;&#xc5b4;
a.geom&#xc774; &#xc544;&#xb2c8;&#xb77c; 'SRID=3005;LINESTRINGM(0 0 0,0 0 1)':geometry&#xc640;
&#xac19;&#xc774;, (&#xd558;&#xc704; &#xcffc;&#xb9ac;/CTE(common table expression) &#xc548;&#xc5d0;
&#xc788;&#xc9c0; &#xc54a;&#xc740;) &#xc0c1;&#xc218;&#xc77c; &#xacbd;&#xc6b0;&#xc5d0;&#xb9cc;
&#xc778;&#xb371;&#xc2a4;&#xc758; &#xd6a8;&#xacfc;&#xac00; &#xb098;&#xd0c0;&#xb0a9;&#xb2c8;&#xb2e4;.
```

2.2.0; 버 전 부 터 사 용 할 수 있 습 니 다. PostgreSQL 9.5 이 상 버 전 에 서 만 인 덱 스 지 &#xc6d0 supported) 을 사 용 할 수 있 습 니 다.

예시

```
-- &#xae00;&#xc790; &#xadf8;&#xb300;&#xb85c;&#xc758; &#xcffc;&#xb9ac;  ↵
    &#xada4;&#xb3c4;&#xb97c; psql &#xbcc0;&#xc218;&#xb85c;  ↵
    &#xc800;&#xc7a5;&#xd569;&#xb2c8;&#xb2e4;...
\set qt 'ST_AddMeasure(ST_MakeLine(ST_MakePointM(-350,300,0),ST_MakePointM(-410,490,0)) ↵
,10,20)'
-- &#xcffc;&#xb9ac;&#xb97c; &#xc2e4;&#xd589;&#xd569;&#xb2c8;&#xb2e4;!
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)
```

స고

ST_DistanceCPA, ST_ClosestPointOfApproach, ST_IsValidTrajectory

8.10.2.3 <#>

```
<#> — A&#xc640; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790; &#xc0ac;&#xc774;&#xc758; 2&#xcc28;&#xc6d0; &#xac70;&#xb9ac;&#xb97c;&#xbcb18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

Synopsis

```
double precision <#>( geometry A , geometry B );
```

설명

```
<#> &#xc5f0;&#xc0b0;&#xc790;&#xb294; &#xb450; &#xbd80;&#xb3d9;&#xc18c;&#xc218;&#xc810;(floating point) &#xacbd;&#xc0c1;&#xc790; &#xc0ac;&#xc774;&#xc758; &#xac70;&#xb9ac;&#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.&#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;(PostgreSQL 9.1 &#xc774;&#xc0c1; &#xbc84;&#xc804; &#xd544;&#xc694;)&#xb9ac;&#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xb4e4;&#xc744; &#xc77d;&#xc5b4; &#xc62c; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xcdc5c;&#xadfc;&#xc811; &#xadfc;&#xc0ac;&#xce58; &#xac70;&#xb9ac;&#xc758; &#xc21c;&#xc11c;&#xb97c; &#xc815;&#xb370; &#xc720;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
```

Note

```
&#xc774; &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294; &#xb3c4;&#xd615;&#xc5d0; &#xb300;&#xd574;
&#xc774;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc744;&#xc9c0;&#xb3c4; &#xbaa8;&#xb974;&#xb294;
&#xc5b4;&#xb5a4; &#xc778;&#xb371;&#xc2a4;&#xb77c;&#xb3c4; &#xd65c;&#xc6a9;&#xd560;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xc5f0;&#xc0b0;&#xc790;&#xac00; ORDER BY &#xc808;
&#xc548;&#xc5d0; &#xc788;&#xc744; &#xb54c;&#xb9cc; &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xb97c;
&#xc4f4;&#xb2e4;&#xb294; &#xc810;&#xc5d0;&#xc11c; &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xb97c;
&#xc774;&#xc6a9;&#xd558;&#xb294; &#xb2e4;&#xb978; &#xc5f0;&#xc0b0;&#xc790;&#xb4e4;&#xacfc;&#xb294;
&#xb2e4;&#xb985;&#xb2c8;&#xb2e4;.
```

Note

```
&#xb3c4;&#xd615; &#xac00;&#xc6b4;&#xb370; &#xd558;&#xb098;&#xac00;, &#xc608;&#xb97c; &#xb4e4;&#xc5b4;
g1.geom <#>&#xac00; &#xc544;&#xb2c8;&#xb77c; ORDER BY (ST_GeomFromText('POINT(1 2)') <#>
geom)&#xacfc; &#xac19;&#xc774;, &#xc0c1;&#xc218;&#xc77c; &#xacbd;&#xc6b0;&#xc5d0;&#xb9cc;
&#xc778;&#xb371;&#xc2a4;&#xc758; &#xd6a8;&#xacfc;&#xac00; &#xb098;&#xd0c0;&#xb0a9;&#xb2c8;&#xb2e4;.
```

2.0.0 ಄전부터 사용할 수 있습니다.

PostgreSQL 9.1 이상 버전에서만 KNN을 쓸 수 있&

예시

```
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

స고

ST_DWithin, ST_Distance, <->

8.10.2.4 <<->>

```
<<->> — A &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc640; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc758;
&#xc911;&#xc2ec;&#xc810; &#xc0ac;&#xc774;&#xc758; N&#xcce28;&#xc6d0; &#xac70;&#xb9ac;&#xb97c; &#xbc18;&#xd658;&
```

Synopsis

```
double precision <<->>( geometry A , geometry B );
```

설명

<<->연산자는 두 도형의 경계 상자중심점 사이의 N차 (유클리드) 거리반환합니다. 최근접 **근사치** 거리순서를 정렬하는 데 유용합니다.

Note



Note!

```
&#xc774; &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294; &#xb3c4;&#xd615;&#xc5d0; &#xb300;&#xd574;
&#xc774;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc744;&#xc9c0;&#xb3c4; &#xbaa8;&#xb974;&#xb294;
N&#xcc28;&#xc6d0; GiST &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xd65c;&#xc6a9;&#xd560;
&#xac83;&#xc785;&#xb2c8;&#xb2e4; &#xc5f0;&#xc0b0;&#xc790;&#xac00; ORDER BY &#xc808;
&#xc548;&#xc5d0; &#xc788;&#xc744; &#xb54c;&#xb9cc; &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xb97c;
&#xc4f4;&#xb2e4;&#xb294; &#xc810;&#xc5d0;&#xc11c; &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xb97c;
&#xc774;&#xc6a9;&#xd558;&#xb294; &#xb2e4;&#xb978; &#xc5f0;&#xc0b0;&#xc790;&#xb4e4;&#xacfc;&#xb294;
&#xb2e4;&#xb985;&#xb2c8;&#xb2e4;
```

Note



Note!

```

&#xb3c4;&#xd615; &#xac00;&#xc6b4;&#xb370; &#xd558;&#xb098;&#xac00;, &#xc608;&#xb97c; &#xb4e4;&#xc5b4;
a.geom&#xc774; &#xc544;&#xb2c8;&#xb77c; 'SRID=3005;POINT(1011102 450541)':geometry&#xc640;
&#xac19;&#xc774;, (&#xd558;&#xc704; &#xcffc;&#xb9ac;/CTE(common table expression) &#xc548;&#xc5d0;
&#xc788;&#xc9c0; &#xc54a;&#xc740;) &#xc0c1;&#xc218;&#xc77c; &#xacbd;&#xc6b0;&#xc5d0;&#xb9cc;
&#xc778;&#xb371;&#xc2a4;&#xc758; &#xd6a8;&#xacfc;&#xac00; &#xb098;&#xd0c0;&#xb0a9;&#xb2c8;&#xb2e4;

```

2.2.0 ಄전부터 사용할 수 있습니다.
PostgreSQL 9.1 이상 ಄전에서만 KNN을 쓸수 있&

స고

<<#>>, <->

8.10.2.5 <<#>>

<<#>> — A&#xc640; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790; &#xc0ac;&#xc774;&#xc758; N&#xcc28;&#xc6d0; &#xac70;&#xb9ac;&#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.

Synopsis

```
double precision <<#>>( geometry A , geometry B );
```

8.11.1.1 ST_3DIntersects

`<#> 연산자는 두 부동소수점(floating point) 경상자 사이의 거리를 반환합니다.
공간 인덱스(PostgreSQL 9.1 이상 버전 필요)&#xb8
경계 상자들을 읽어 올 수도 있습&#xc
최근접 근사ౘ 거리의 순서를 정&#xc
데 유용합니다.`

Note



`이 피연산자(operand)는 도형에 대해
이용할 수 있을지도 모르는
어떤 인덱스라도 활용할
것입니다. 연산자가 ORDER BY 절
안에 있을 때만 공간 인덱스를
쓴다는 점에서 공간 인덱스를
이용하는 다른 연산자들과는
다릅니다.`

Note



`도형 가운데 하나가, 예를 들어
g1.geom <<#>>가 아니라 ORDER BY (ST_GeomFromText('POINT(1 2)')
<<#>> geom)과 같이, 상수일 경우에만
인덱스의 효과가 나타납니다.`

2.2.0 버전부터 사용할 수 있습니다.

PostgreSQL 9.1 이상 버전에서만 KNN을 쓸 수 있&

8.11.1.1 ST_3DIntersects

`<<#>>, <#>`

8.11 Spatial Relationships

8.11.1 Topological Relationships

8.11.1.1 ST_3DIntersects

ST_3DIntersects — Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area).

Synopsis

boolean **ST_3DIntersects**(geometry geomA , geometry geomB);

Description

Overlaps, Touches, Within all imply spatial intersection. If any of the aforementioned returns true, then the geometries also spatially intersect. Disjoint implies false for spatial intersection.

**Note**

This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries.

Changed: 3.0.0 SFCGAL backend removed, GEOS backend supports TINs.

Availability: 2.0.0



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 implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1

Geometry Examples

```
SELECT ST_3DIntersects(pt, line), ST_Intersects(pt, line)
  FROM (SELECT 'POINT(0 0 2)::geometry As pt, 'LINESTRING (0 0 1, 0 2 3)::geometry As
        line) As foo;
 st_3dintersects | st_intersects
-----+-----
f                | t
(1 row)
```

TIN Examples

```
SELECT ST_3DIntersects('TIN(((0 0 0,1 0 0,0 1 0,0 0 0)))::geometry, 'POINT(.1 .1 0)::
      geometry);
 st_3dintersects
-----
t
```

See Also

[ST_Intersects](#)

8.11.1.2 ST_Contains

ST_Contains — Tests if no points of B lie in the exterior of A, and A and B have at least one interior point in common.

Synopsis

boolean **ST_Contains**(geometry geomA, geometry geomB);

Description

Returns TRUE if geometry B is completely inside geometry A. A contains B if and only if no points of B lie in the exterior of A, and at least one point of the interior of B lies in the interior of A.

A subtlety of the definition is that a geometry does not contain things in its boundary. Thus polygons and lines do *not* contain lines and points lying in their boundary. For further details see [Subtleties of OGC Covers, Contains, Within](#). (The [ST_Covers](#) predicate provides a more inclusive relationship.) However, a geometry does contain itself. (In contrast, in the [ST_ContainsProperly](#) predicate a geometry does *not* properly contain itself.)

ST_Contains is the inverse of [ST_Within](#). So, `ST_Contains(A,B) = ST_Within(B,A)`.



Note

This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries. To avoid index use, use the function `_ST_Contains`.

Performed by the GEOS module

Enhanced: 2.3.0 Enhancement to PIP short-circuit extended to support MultiPoints with few points. Prior versions only supported point in polygon.



Important

Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION



Important

Do not use this function with invalid geometries. You will get unexpected results.

NOTE: this is the "allowable" version that returns a boolean, not an integer.



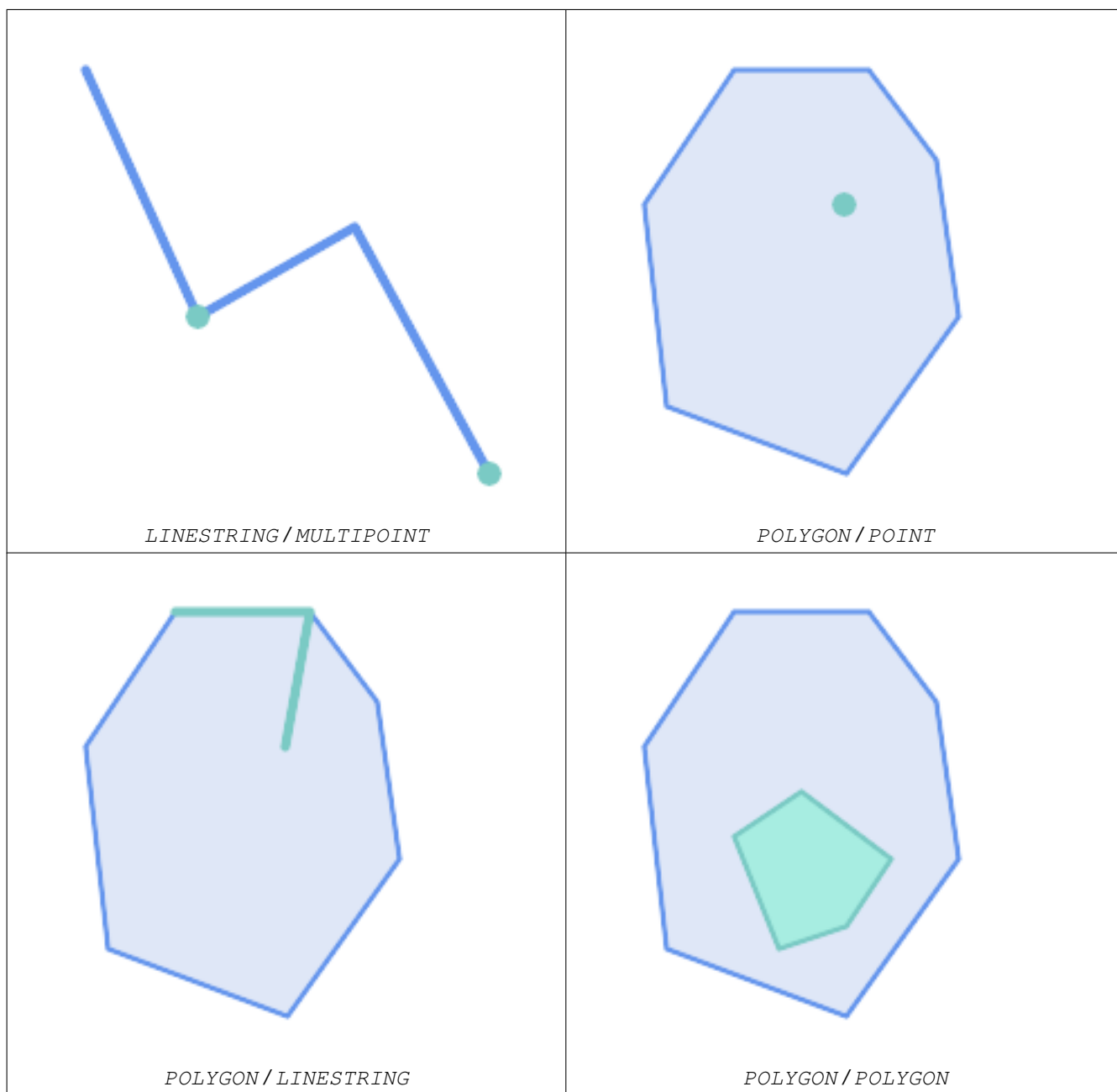
This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 // s2.1.13.3 - same as `within(geometry B, geometry A)`



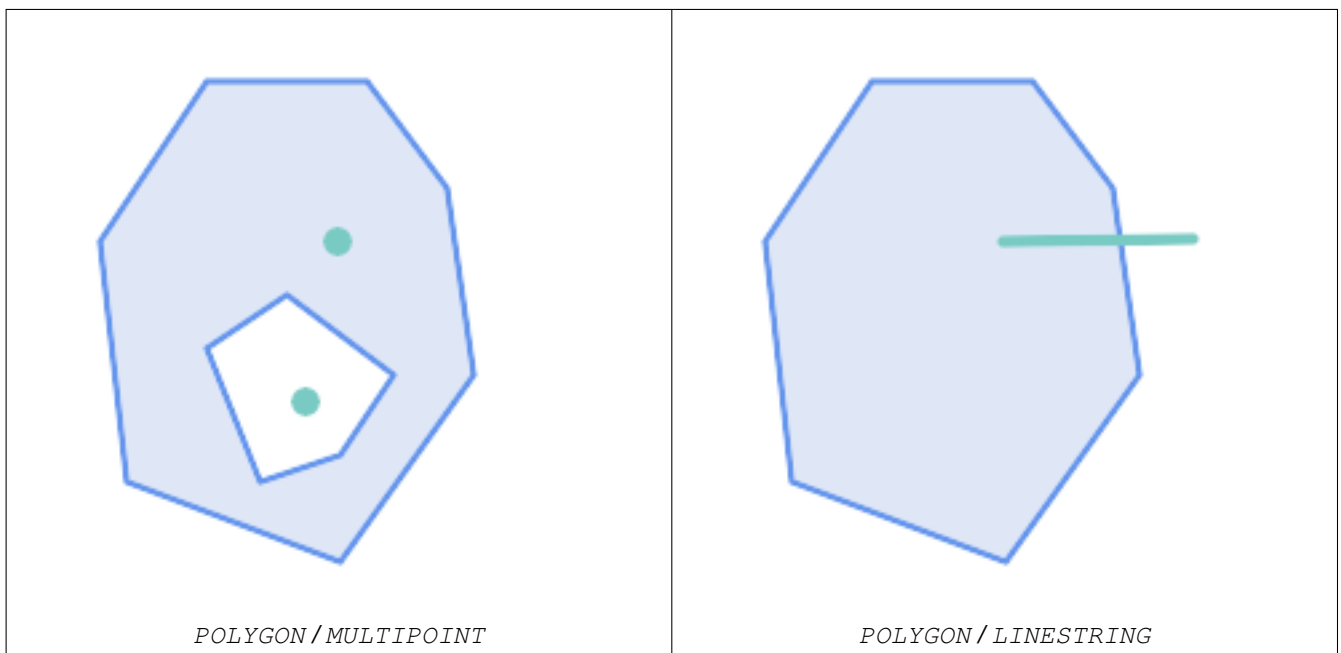
This method implements the SQL/MM specification. SQL-MM 3: 5.1.31

Examples

ST_Contains returns TRUE in the following situations:



The ST_Contains predicate returns FALSE in the following situations:



```
-- A circle within a circle
SELECT ST_Contains(smallc, bigc) As smallcontainsbig,
       ST_Contains(bigc, smallc) As bigcontainssmall,
       ST_Contains(bigc, ST_Union(smallc, bigc)) as bigcontainsunion,
       ST_Equals(bigc, ST_Union(smallc, bigc)) as bigisunion,
       ST_Covers(bigc, ST_ExteriorRing(bigc)) As bigcoversexterior,
       ST_Contains(bigc, ST_ExteriorRing(bigc)) As bigcontainsexterior
FROM (SELECT ST_Buffer(ST_GeomFromText('POINT(1 2)'), 10) As smallc,
       ST_Buffer(ST_GeomFromText('POINT(1 2)'), 20) As bigc) As foo;

-- Result
smallcontainsbig | bigcontainssmall | bigcontainsunion | bigisunion | bigcoversexterior | bigcontainsexterior
-----+-----+-----+-----+-----+-----
f                | t                | t                | t          | t                | f

-- Example demonstrating difference between contains and contains properly
SELECT ST_GeometryType(geomA) As geomtype, ST_Contains(geomA, geomA) AS acontainsa,
       ST_ContainsProperly(geomA, geomA) AS acontainspropa,
       ST_Contains(geomA, ST_Boundary(geomA)) As acontainsba, ST_ContainsProperly(geomA,
       ST_Boundary(geomA)) As acontainspropba
FROM (VALUES ( ST_Buffer(ST_Point(1,1), 5,1) ),
            ( ST_MakeLine(ST_Point(1,1), ST_Point(-1,-1) ) ),
            ( ST_Point(1,1) )
      ) As foo(geomA);

geomtype | acontainsa | acontainspropa | acontainsba | acontainspropba
-----+-----+-----+-----+-----
ST_Polygon | t          | f              | f           | f
ST_LineString | t          | f              | f           | f
ST_Point | t          | t              | f           | f
```

See Also

[ST_Boundary](#), [ST_ContainsProperly](#), [ST_Covers](#), [ST_CoveredBy](#), [ST_Equals](#), [ST_Within](#)

8.11.1.3 ST_ContainsProperly

ST_ContainsProperly — Tests if B intersects the interior of A but not the boundary or exterior.

Synopsis

```
boolean ST_ContainsProperly(geometry geomA, geometry geomB);
```

Description

Returns true if B intersects the interior of A but not the boundary or exterior.

A does not properly contain itself, but does contain itself.

Every point of the other geometry is a point of this geometry's interior. The DE-9IM Intersection Matrix for the two geometries matches `[T**FF*FF*]` used in [ST_Relate](#)

An example use case for this predicate is computing the intersections of a set of geometries with a large polygonal geometry. Since intersection is a fairly slow operation, it can be more efficient to use `containsProperly` to filter out test geometries which lie wholly inside the area. In these cases the intersection is known a priori to be exactly the original test geometry.



Note

This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries. To avoid index use, use the function `_ST_ContainsProperly`.



Note

The advantage of this predicate over [ST_Contains](#) and [ST_Intersects](#) is that it can be computed more efficiently, with no need to compute topology at individual points.

Performed by the GEOS module.

Availability: 1.4.0



Important

Enhanced: 3.0.0 enabled support for `GEOMETRYCOLLECTION`



Important

Do not use this function with invalid geometries. You will get unexpected results.

Examples

```
--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
```



Important
Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION



Important
Do not use this function with invalid geometries. You will get unexpected results.

Performed by the GEOS module
Availability: 1.2.2
NOTE: this is the "allowable" version that returns a boolean, not an integer.
Not an OGC standard, but Oracle has it too.

Examples

```
--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)
```

See Also
[ST_Contains](#), [ST_Covers](#), [ST_ExteriorRing](#), [ST_Within](#)

8.11.1.5 ST_Covers

ST_Covers — Tests if no point in B is outside A

Synopsis
boolean **ST_Covers**(geometry geomA, geometry geomB);
boolean **ST_Covers**(geography geogpolyA, geography geogpointB);

Description
Returns `true` if no point in Geometry/Geography B is outside Geometry/Geography A. Equivalently, tests if every point of geometry B is inside (i.e. intersects the interior or boundary of) geometry A.



Note
This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries. To avoid index use, use the function `_ST_Covers`.



Important
Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION



Important
Do not use this function with invalid geometries. You will get unexpected results.

Performed by the GEOS module

Enhanced: 2.4.0 Support for polygon in polygon and line in polygon added for geography type

Enhanced: 2.3.0 Enhancement to PIP short-circuit for geometry extended to support MultiPoints with few points. Prior versions only supported point in polygon.

Availability: 1.5 - support for geography was introduced.

Availability: 1.2.2

NOTE: this is the "allowable" version that returns a boolean, not an integer.

Not an OGC standard, but Oracle has it too.

Examples

Geometry example

```
--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)
```

Geography Example

```
-- 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
```

See Also

[ST_Contains](#), [ST_CoveredBy](#), [ST_Within](#)

8.11.1.6 ST_Crosses

ST_Crosses — Tests if two geometries have some, but not all, interior points in common.

Synopsis

```
boolean ST_Crosses(geometry g1, geometry g2);
```

Description

Compares two geometry objects and returns `true` if their intersection "spatially cross", that is, the geometries have some, but not all interior points in common. The intersection of the interiors of the geometries must be non-empty and must have dimension less than the maximum dimension of the two input geometries. Additionally, the intersection of the two geometries must not equal either of the source geometries. Otherwise, it returns `false`.

In mathematical terms, this is:

$$a.\text{Crosses}(b) \Leftrightarrow (\dim(I(a) \cap I(b)) < \max(\dim(I(a)), \dim(I(b)))) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

Geometries cross if their DE-9IM Intersection Matrix matches:

- T*T***** for Point/Line, Point/Area, and Line/Area situations
- T*****T** for Line/Point, Area/Point, and Area/Line situations
- 0***** for Line/Line situations

For Point/Point and Area/Area situations this predicate returns `false`.

The OpenGIS Simple Features Specification defines this predicate only for Point/Line, Point/Area, Line/Line, and Line/Area situations. JTS / GEOS extends the definition to apply to Line/Point, Area/Point and Area/Line situations as well. This makes the relation symmetric.



Note

This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries.



Important

Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION



This method implements the [OGC 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

To determine a list of roads that cross a highway, use a query similar to:

```
SELECT roads.id
FROM roads, highways
WHERE ST_Crosses(roads.geom, highways.geom);
```

See Also

[ST_Contains](#), [ST_Overlaps](#)

8.11.1.7 ST_Disjoint

ST_Disjoint — Tests if two geometries are disjoint (they have no point in common).

Synopsis

boolean **ST_Disjoint**(geometry A , geometry B);

Description

Overlaps, Touches, Within all imply geometries are not spatially disjoint. If any of the aforementioned returns true, then the geometries are not spatially disjoint. Disjoint implies false for spatial intersection.



Important

Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION

Performed by the GEOS module



Note

This function call does not use indexes



Note

NOTE: this is the "allowable" version that returns a boolean, not an integer.



This method implements the [OGC 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

Examples

```
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)
```

See Also

[ST_Intersects](#)

8.11.1.8 ST_Equals

ST_Equals — Tests if two geometries include the same set of points.

Synopsis

boolean **ST_Equals**(geometry A, geometry B);

Description

Returns `true` if the given geometries are "spatially equal". Use this for a 'better' answer than `'=`'. Note by spatially equal we mean `ST_Within(A,B) = true` and `ST_Within(B,A) = true` and also mean ordering of points can be different but represent the same geometry structure. To verify the order of points is consistent, use `ST_OrderingEquals` (it must be noted `ST_OrderingEquals` is a little more stringent than simply verifying order of points are the same).



Important

Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION



This method implements the [OGC 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

Examples

```
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)
```

See Also

[ST_IsValid](#), [ST_OrderingEquals](#), [ST_Reverse](#), [ST_Within](#)

8.11.1.9 ST_Intersects

ST_Intersects — Tests if two geometries intersect (they have at least one point in common).

Synopsis

```
boolean ST_Intersects( geometry geomA , geometry geomB );
boolean ST_Intersects( geography geogA , geography geogB );
```

Description

Compares two geometries and returns `true` if they intersect. Geometries intersect if they have any point in common.

For geography, a distance tolerance of 0.00001 meters is used (so points that are very close are considered to intersect).

Geometries intersect if their DE-9IM Intersection Matrix matches one of:

- T*****
- *T*****
- ***T*****
- ****T*****

Spatial intersection is implied by all the other spatial relationship tests, except [ST_Disjoint](#), which tests that geometries do NOT intersect.



Note

This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries.

Changed: 3.0.0 SFCGAL version removed and native support for 2D TINS added.

Enhanced: 2.5.0 Supports GEOMETRYCOLLECTION.

Enhanced: 2.3.0 Enhancement to PIP short-circuit extended to support MultiPoints with few points. Prior versions only supported point in polygon.

Performed by the GEOS module (for geometry), geography is native

Availability: 1.5 support for geography was introduced.



Note

For geography, this function has a distance tolerance of about 0.00001 meters and uses the sphere rather than spheroid calculation.

**Note**

NOTE: this is the "allowable" version that returns a boolean, not an integer.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 //s2.1.13.3 - ST_Intersects(g1, g2) --> Not (ST_Disjoint(g1, g2))



This method implements the SQL/MM specification. SQL-MM 3: 5.1.27



This method supports Circular Strings and Curves



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Geometry Examples

```
SELECT ST_Intersects('POINT(0 0)::geometry, 'LINESTRING ( 2 0, 0 2 ) '::geometry);
st_intersects
-----
f
(1 row)
SELECT ST_Intersects('POINT(0 0)::geometry, 'LINESTRING ( 0 0, 0 2 ) '::geometry);
st_intersects
-----
t
(1 row)

-- Look up in table. Make sure table has a GiST index on geometry column for faster lookup.
SELECT id, name FROM cities WHERE ST_Intersects(geom, 'SRID=4326;POLYGON((28 53,27.707 52.293,27 52,26.293 52.293,26 53,26.293 53.707,27 54,27.707 53.707,28 53))');
id | name
----+-----
 2 | Minsk
(1 row)
```

Geography Examples

```
SELECT ST_Intersects(
  'SRID=4326;LINESTRING(-43.23456 72.4567,-43.23456 72.4568) '::geography,
  'SRID=4326;POINT(-43.23456 72.4567772) '::geography
);

st_intersects
-----
t
```

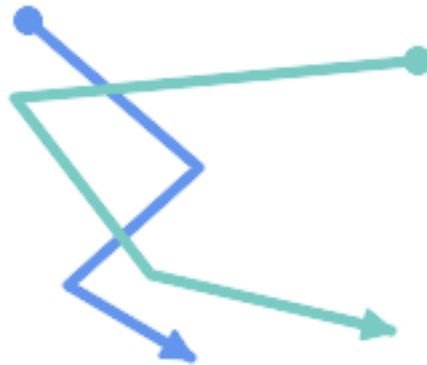
See Also

[ST_3DIntersects](#), [ST_Disjoint](#)

8.11.1.10 ST_LineCrossingDirection

ST_LineCrossingDirection — Returns a number indicating the crossing behavior of two LineStrings.

Example: LINE_MULTICROSS_END_SAME_FIRST_LEFT and LINE_MULTICROSS_END_SAME_FIRST_RIGHT

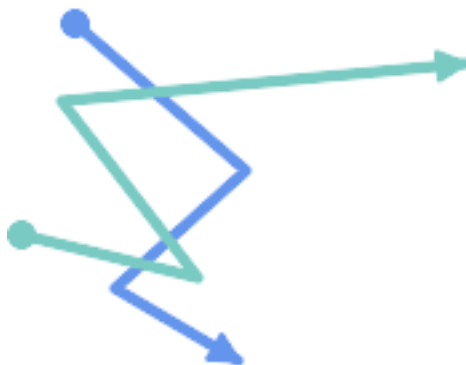


Blue: Line A; Green: Line B

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
      ST_GeomFromText('LINESTRING(25 169,89 114,40 70,86 43)') As lineA,
      ST_GeomFromText('LINESTRING(171 154,20 140,71 74,161 53)') As lineB
    ) As foo;
```

A_cross_B	B_cross_A
3	-3

Example: LINE_MULTICROSS_END_LEFT and LINE_MULTICROSS_END_RIGHT



Blue: Line A; Green: Line B

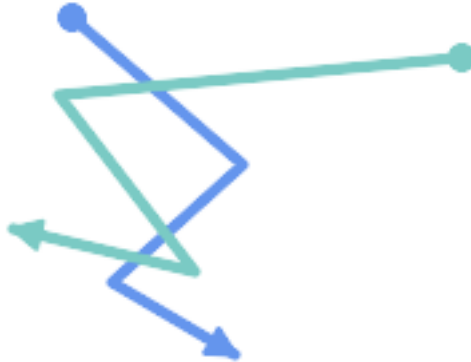
```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
```



```
ST_GeomFromText('LINESTRING(25 169,89 114,40 70,86 43)') As lineA,
ST_GeomFromText('LINESTRING(5 90, 71 74, 20 140, 171 154)') As lineB
) As foo;
```

A_cross_B	B_cross_A
-----+-----	
-2	2

Example: LINE_MULTICROSS_END_LEFT and LINE_MULTICROSS_END_RIGHT



Blue: Line A; Green: Line B

```
SELECT ST_LineCrossingDirection(lineA, lineB) As A_cross_B,
       ST_LineCrossingDirection(lineB, lineA) As B_cross_A
FROM (SELECT
       ST_GeomFromText('LINESTRING(25 169,89 114,40 70,86 43)') As lineA,
       ST_GeomFromText('LINESTRING (171 154, 20 140, 71 74, 2.99 90.16)') As lineB
) As foo;
```

A_cross_B	B_cross_A
-----+-----	
2	-2

```
SELECT s1.gid, s2.gid, ST_LineCrossingDirection(s1.geom, s2.geom)
FROM streets s1 CROSS JOIN streets s2
ON (s1.gid != s2.gid AND s1.geom && s2.geom )
WHERE ST_LineCrossingDirection(s1.geom, s2.geom) > 0;
```

See Also

[ST_Crosses](#)

8.11.1.11 ST_OrderingEquals

ST_OrderingEquals — Tests if two geometries represent the same geometry and have points in the same directional order.

Synopsis

boolean **ST_OrderingEquals**(geometry A, geometry B);

Description

`ST_OrderingEquals` compares two geometries and returns t (TRUE) if the geometries are equal and the coordinates are in the same order; otherwise it returns f (FALSE).



Note

This function is implemented as per the ArcSDE SQL specification rather than SQL-MM. 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

Examples

```
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)
```

See Also

[&&](#), [ST_Equals](#), [ST_Reverse](#)

8.11.1.12 ST_Overlaps

`ST_Overlaps` — Tests if two geometries intersect and have the same dimension, but are not completely contained by each other.

Synopsis

boolean **ST_Overlaps**(geometry A, geometry B);

Description

Returns TRUE if geometry A and B "spatially overlap". Two geometries overlap if they have the same dimension, each has at least one point not shared by the other (or equivalently neither covers the other), and the intersection of their interiors has the same dimension. The overlaps relationship is symmetrical.



Note
This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries. To avoid index use, use the function `_ST_Overlaps`.

Performed by the GEOS module



Important
Enhanced: 3.0.0 enabled support for `GEOMETRYCOLLECTION`

NOTE: this is the "allowable" version that returns a boolean, not an integer.



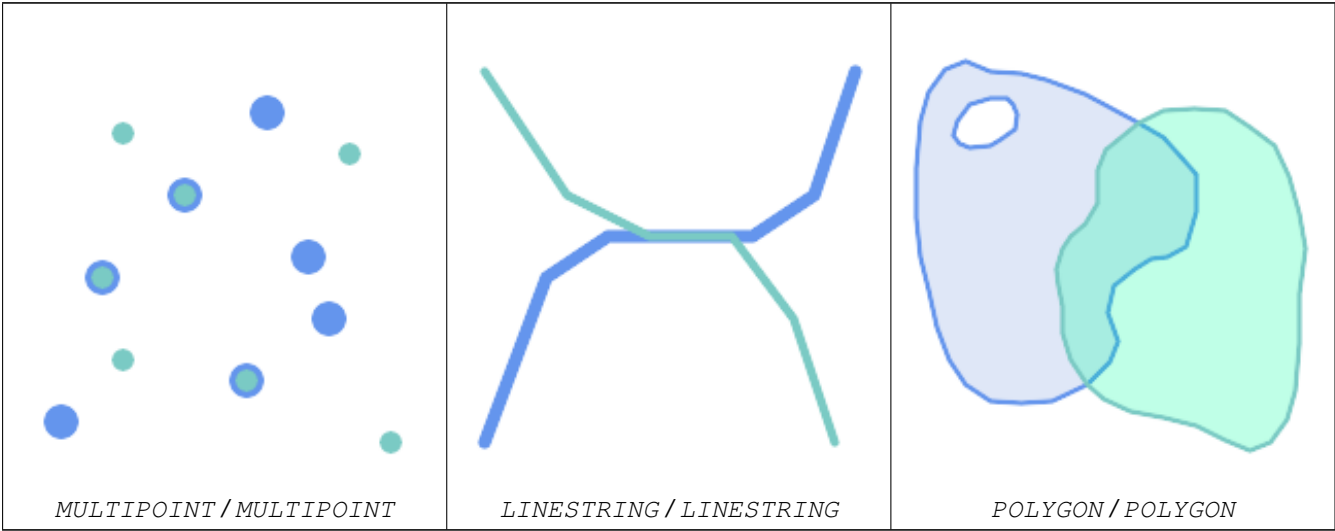
This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 // s2.1.13.3

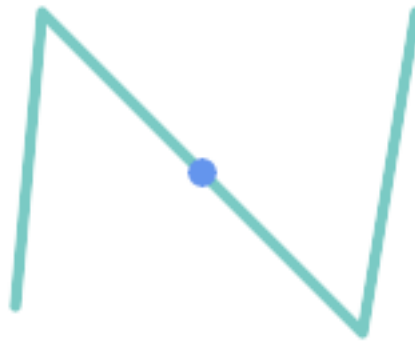


This method implements the SQL/MM specification. SQL-MM 3: 5.1.32

Examples

`ST_Overlaps` returns TRUE in the following situations:





A Point on a LineString is contained, but since it has lower dimension it does not overlap or cross.

```
SELECT ST_Overlaps(a,b) AS overlaps,      ST_Crosses(a,b) AS crosses,
       ST_Intersects(a, b) AS intersects,  ST_Contains(b,a) AS b_contains_a
FROM (SELECT ST_GeomFromText('POINT (100 100)') As a,
       ST_GeomFromText('LINESTRING (30 50, 40 160, 160 40, 180 160)') AS b) AS t
```

overlaps	crosses	intersects	b_contains_a
f	f	t	t



A LineString that partly covers a Polygon intersects and crosses, but does not overlap since it has different dimension.

```
SELECT ST_Overlaps(a,b) AS overlaps,      ST_Crosses(a,b) AS crosses,
       ST_Intersects(a, b) AS intersects,  ST_Contains(a,b) AS contains
FROM (SELECT ST_GeomFromText('POLYGON ((40 170, 90 30, 180 100, 40 170))') AS a,
       ST_GeomFromText('LINESTRING(10 10, 190 190)') AS b) AS t;
```

overlap	crosses	intersects	contains
f	t	t	f



Two Polygons that intersect but with neither contained by the other overlap, but do not cross because their intersection has the same dimension.

```
SELECT ST_Overlaps(a,b) AS overlaps,          ST_Crosses(a,b) AS crosses,
       ST_Intersects(a, b) AS intersects,    ST_Contains(b, a) AS b_contains_a,
       ST_Dimension(a) AS dim_a, ST_Dimension(b) AS dim_b,
       ST_Dimension(ST_Intersection(a,b)) AS dim_int
FROM (SELECT ST_GeomFromText('POLYGON ((40 170, 90 30, 180 100, 40 170))') AS a,
       ST_GeomFromText('POLYGON ((110 180, 20 60, 130 90, 110 180))') AS b) As t;
```

overlaps	crosses	intersects	b_contains_a	dim_a	dim_b	dim_int
t	f	t	f	2	2	2

See Also

ST_Contains, ST_Crosses, ST_Dimension, ST_Intersects

8.11.1.13 ST Relate

ST_Relate — Tests if two geometries have a topological relationship matching an Intersection Matrix pattern, or computes their Intersection Matrix

Synopsis

```
boolean ST_Relate(geometry geomA, geometry geomB, text intersectionMatrixPattern);
text ST_Relate(geometry geomA, geometry geomB);
text ST_Relate(geometry geomA, geometry geomB, integer boundaryNodeRule);
```

Description

These functions allow testing and evaluating the spatial (topological) relationship between two geometries, as defined by the **Dimensionally Extended 9-Intersection Model** (DE-9IM).

The DE-9IM is specified as a 9-element matrix indicating the dimension of the intersections between the Interior, Boundary and Exterior of two geometries. It is represented by a 9-character text string using the symbols 'F', '0', '1', '2' (e.g. 'FF1FF0102').

A specific kind of spatial relationships is evaluated by comparing the intersection matrix to an *intersection matrix pattern*. A pattern can include the additional symbols 'T' and '*'. Common spatial relationships are provided by the named functions [ST_Contains](#), [ST_ContainsProperly](#), [ST_Covers](#), [ST_CoveredBy](#), [ST_Crosses](#), [ST_Disjoint](#), [ST_Equals](#), [ST_Intersects](#), [ST_Overlaps](#), [ST_Touches](#), and [ST_Within](#). Using an explicit pattern allows testing multiple conditions of intersects, crosses, etc in one step. It also allows testing spatial relationships which do not have a named spatial relationship function. For example, the relationship "Interior-Intersects" has the DE-9IM pattern T*****, which is not evaluated by any named predicate.

For more information refer to [Section 5.1](#).

Variant 1: Tests if two geometries are spatially related according to the given `intersectionMatrixPattern`.



Note

Unlike most of the named spatial relationship predicates, this does NOT automatically include an index call. The reason is that some relationships are true for geometries which do NOT intersect (e.g. Disjoint). If you are using a relationship pattern that requires intersection, then include the `&&` index call.



Note

It is better to use a named relationship function if available, since they automatically use a spatial index where one exists. Also, they may implement performance optimizations which are not available with full relate evaluation.

Variant 2: Returns the DE-9IM matrix string for the spatial relationship between the two input geometries. The matrix string can be tested for matching a DE-9IM pattern using [ST_RelateMatch](#).

Variant 3: Like variant 2, but allows specifying a **Boundary Node Rule**. A boundary node rule allows finer control over whether geometry boundary points are considered to lie in the DE-9IM Interior or Boundary. The `boundaryNodeRule` code is: 1: OGC/MOD2, 2: Endpoint, 3: MultivalentEndpoint, 4: MonovalentEndpoint.

This function is not in the OGC spec, but is implied. see s2.1.13.2



This method implements the [OGC 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

Performed by the GEOS module

Enhanced: 2.0.0 - added support for specifying boundary node rule.



Important

Enhanced: 3.0.0 enabled support for `GEOMETRYCOLLECTION`

Examples

Using the boolean-valued function to test spatial relationships.

```
SELECT ST_Relate('POINT(1 2)', ST_Buffer( 'POINT(1 2)', 2), 'OFFFFF212');
st_relate
-----
t

SELECT ST_Relate(POINT(1 2)', ST_Buffer( 'POINT(1 2)', 2), '*FF*FF212');
st_relate
-----
t
```

Testing a custom spatial relationship pattern as a query condition, with `&&` to enable using a spatial index.

```
-- Find compounds that properly intersect (not just touch) a poly (Interior Intersects)

SELECT c.* , p.name As poly_name
  FROM polys AS p
  INNER JOIN compounds As c
        ON c.geom && p.geom
        AND ST_Relate(p.geom, c.geom, 'T*****');
```

Computing the intersection matrix for spatial relationships.

```
SELECT ST_Relate( 'POINT(1 2)',
                  ST_Buffer( 'POINT(1 2)', 2));

st_relate
-----
0FFFFFF212

SELECT ST_Relate( 'LINESTRING(1 2, 3 4)',
                  'LINESTRING(5 6, 7 8)' );

st_relate
-----
FF1FF0102
```

See Also

Section 5.1, [ST_RelateMatch](#), [ST_Contains](#), [ST_ContainsProperly](#), [ST_Covers](#), [ST_CoveredBy](#), [ST_Crosses](#), [ST_Disjoint](#), [ST_Equals](#), [ST_Intersects](#), [ST_Overlaps](#), [ST_Touches](#), [ST_Within](#)

8.11.1.14 ST_RelateMatch

`ST_RelateMatch` — Tests if a DE-9IM Intersection Matrix matches an Intersection Matrix pattern

Synopsis

boolean **ST_RelateMatch**(text intersectionMatrix, text intersectionMatrixPattern);

Description

Tests if a [Dimensionally Extended 9-Intersection Model](#) (DE-9IM) `intersectionMatrix` value satisfies an `intersectionMatrixPattern`. Intersection matrix values can be computed by [ST_Relate](#).

For more information refer to Section 5.1.

Performed by the GEOS module

Availability: 2.0.0

Examples

```
SELECT ST_RelateMatch('101202FFF', 'TTTTTTFFF') ;
-- result --
t
```

Patterns for common spatial relationships matched against intersection matrix values, for a line in various positions relative to a polygon

```

SELECT pat.name AS relationship, pat.val AS pattern,
       mat.name AS position, mat.val AS matrix,
       ST_RelateMatch(mat.val, pat.val) AS match
FROM (VALUES ( 'Equality', 'T1FF1FFF1' ),
            ( 'Overlaps', 'T*T***T**' ),
            ( 'Within', 'T*F**F***' ),
            ( 'Disjoint', 'FF*FF****' )) AS pat(name,val)
CROSS JOIN
  (VALUES ('non-intersecting', 'FF1FF0212'),
         ('overlapping', '1010F0212'),
         ('inside', '1FF0FF212')) AS mat(name,val);

```

relationship	pattern	position	matrix	match
Equality	T1FF1FFF1	non-intersecting	FF1FF0212	f
Equality	T1FF1FFF1	overlapping	1010F0212	f
Equality	T1FF1FFF1	inside	1FF0FF212	f
Overlaps	T*T***T**	non-intersecting	FF1FF0212	f
Overlaps	T*T***T**	overlapping	1010F0212	t
Overlaps	T*T***T**	inside	1FF0FF212	f
Within	T*F**F***	non-intersecting	FF1FF0212	f
Within	T*F**F***	overlapping	1010F0212	f
Within	T*F**F***	inside	1FF0FF212	t
Disjoint	FF*FF****	non-intersecting	FF1FF0212	t
Disjoint	FF*FF****	overlapping	1010F0212	f
Disjoint	FF*FF****	inside	1FF0FF212	f

See Also

Section 5.1, [ST_Relate](#)

8.11.1.15 ST_Touches

ST_Touches — Tests if two geometries have at least one point in common, but their interiors do not intersect.

Synopsis

boolean **ST_Touches**(geometry A, geometry B);

Description

Returns TRUE if A and B intersect, but their interiors do not intersect. Equivalently, A and B have at least one point in common, and the common points lie in at least one boundary. For Point/Point inputs the relationship is always FALSE, since points do not have a boundary.

In mathematical terms, this relationship is:

$$a.Touches(b) \Leftrightarrow (I(a) \cap I(b) = \emptyset) \wedge (a \cap b) \neq \emptyset$$

This relationship holds if the DE-9IM Intersection Matrix for the two geometries matches one of:

- FT*****
- F**T*****
- F***T****



Note
This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries. To avoid using an index, use `_ST_Touches` instead.



Important
Enhanced: 3.0.0 enabled support for `GEOMETRYCOLLECTION`



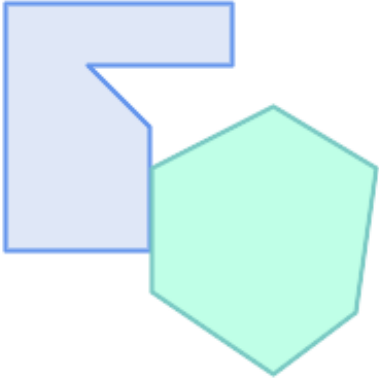
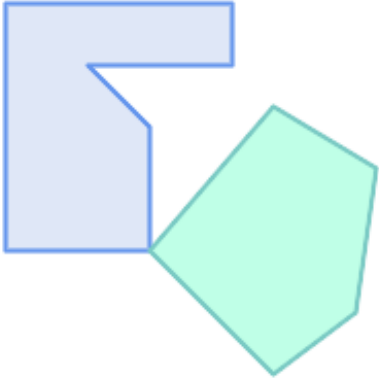
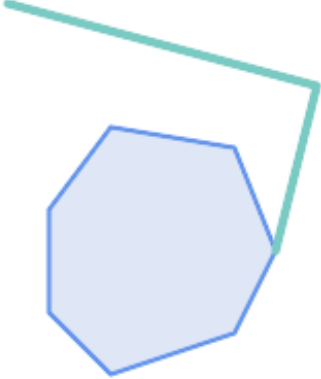
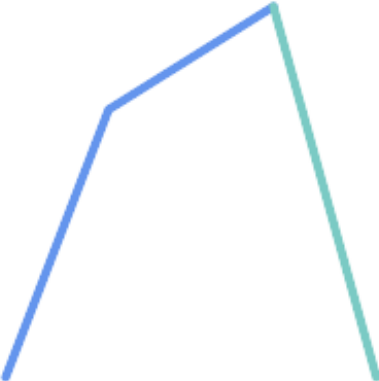
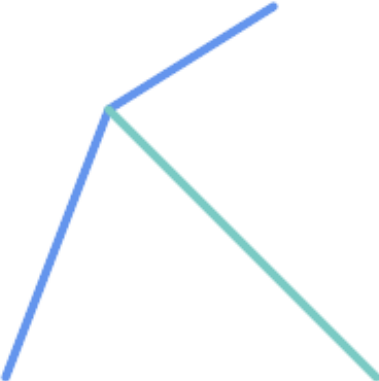
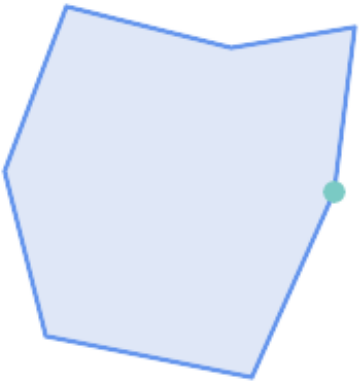
This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.2 // s2.1.13.3



This method implements the SQL/MM specification. SQL-MM 3: 5.1.28

Examples

The `ST_Touches` predicate returns `TRUE` in the following examples.

 <i>POLYGON / POLYGON</i>	 <i>POLYGON / POLYGON</i>	 <i>POLYGON / LINESTRING</i>
 <i>LINESTRING / LINESTRING</i>	 <i>LINESTRING / LINESTRING</i>	 <i>POLYGON / POINT</i>

```
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)

```

8.11.1.16 ST_Within

ST_Within — Tests if no points of A lie in the exterior of B, and A and B have at least one interior point in common.

Synopsis

boolean **ST_Within**(geometry A, geometry B);

Description

Returns TRUE if geometry A is completely inside geometry B. For this function to make sense, the source geometries must both be of the same coordinate projection, having the same SRID. It is a given that if **ST_Within**(A,B) is true and **ST_Within**(B,A) is true, then the two geometries are considered spatially equal.

A subtlety of this definition is that the boundary of a geometry is not within the geometry. This means that lines and points lying in the boundary of a polygon or line are *not* within the geometry. For further details see [Subtleties of OGC Covers, Contains, Within](#). (The **ST_CoveredBy** predicate provides a more inclusive relationship).

ST_Within is the inverse of **ST_Contains**. So, **ST_Within**(A,B) = **ST_Contains**(B,A).



Note

This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries. To avoid index use, use the function **_ST_Within**.

Performed by the GEOS module

Enhanced: 2.3.0 Enhancement to PIP short-circuit for geometry extended to support MultiPoints with few points. Prior versions only supported point in polygon.



Important

Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION



Important

Do not use this function with invalid geometries. You will get unexpected results.

NOTE: this is the "allowable" version that returns a boolean, not an integer.



This method implements the [OGC 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



Note

This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries.



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 ?

Availability: 2.0.0

Examples

```
-- 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
```

See Also

ST_3DDFullyWithin, ST_DWithin, ST_DFullyWithin, ST_3DDistance, ST_Distance, ST_3DMaxDistance, ST_Transform

8.11.2.2 ST_3DDFullyWithin

ST_3DDFullyWithin — Tests if two 3D geometries are entirely within a given 3D distance

Synopsis

```
boolean ST_3DDFullyWithin(geometry g1, geometry g2, double precision distance);
```

Description

Returns true if the 3D geometries are fully within the specified distance of one another. The distance is specified in units defined by the spatial reference system of the geometries. For this function to make sense, the source geometries must both be of the same coordinate projection, having the same SRID.



Note
This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries.

Availability: 2.0.0

- ✔ This function supports 3d and will not drop the z-index.
- ✔ This function supports Polyhedral surfaces.

Examples

```
-- 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
```

See Also

[ST_3DDWithin](#), [ST_DWithin](#), [ST_DFullyWithin](#), [ST_3DMaxDistance](#)

8.11.2.3 ST_DFullyWithin

ST_DFullyWithin — Tests if two geometries are entirely within a given distance

Synopsis

boolean **ST_DFullyWithin**(geometry g1, geometry g2, double precision distance);

Description

Returns true if the geometries are entirely within the specified distance of one another. The distance is specified in units defined by the spatial reference system of the geometries. For this function to make sense, the source geometries must both be of the same coordinate projection, having the same SRID.



Note
This function automatically includes a bounding box comparison that makes use of any spatial indexes that are available on the geometries.

Availability: 1.5.0

Examples

```
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              |
```

See Also

[ST_MaxDistance](#), [ST_DWithin](#), [ST_3DDWithin](#), [ST_3DDFullyWithin](#)

8.11.2.4 ST_DWithin

ST_DWithin — Tests if two geometries are within a given distance

Synopsis

boolean **ST_DWithin**(geometry g1, geometry g2, double precision distance_of_srid);
boolean **ST_DWithin**(geography gg1, geography gg2, double precision distance_meters, boolean use_spheroid = true);

Description

Returns true if the geometries are within a given distance

For geometry: The distance is specified in units defined by the spatial reference system of the geometries. For this function to make sense, the source geometries must be in the same coordinate system (have the same SRID).

For geography: units are in meters and distance measurement defaults to use_spheroid=true. For faster evaluation use use_spheroid=false to measure on the sphere.

**Note**
Use [ST_3DDWithin](#) for 3D geometries.

**Note**
This function call includes a bounding box comparison that makes use of any indexes that are available on the geometries.

 This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#).

- Availability: 1.5.0 support for geography was introduced
- Enhanced: 2.1.0 improved speed for geography. See [Making Geography faster](#) for details.
- Enhanced: 2.1.0 support for curved geometries was introduced.

Prior to 1.3, [ST_Expand](#) was commonly used in conjunction with && and ST_Distance to test for distance, and in pre-1.3.4 this function used that logic. From 1.3.4, ST_DWithin uses a faster short-circuit distance function.

Examples

```
-- Find the nearest hospital to each school
-- that is within 3000 units of the school.
-- We do an ST_DWithin search to utilize indexes to limit our search list
-- that the non-indexable ST_Distance needs to process
-- If the units of the spatial reference is meters then units would be meters
SELECT DISTINCT ON (s.gid) s.gid, s.school_name, s.geom, h.hospital_name
  FROM schools s
    LEFT JOIN hospitals h ON ST_DWithin(s.geom, h.geom, 3000)
 ORDER BY s.gid, ST_Distance(s.geom, h.geom);

-- The schools with no close hospitals
-- Find all schools with no hospital within 3000 units
-- away from the school. Units is in units of spatial ref (e.g. meters, feet, degrees)
SELECT s.gid, s.school_name
  FROM schools s
    LEFT JOIN hospitals h ON ST_DWithin(s.geom, h.geom, 3000)
 WHERE h.gid IS NULL;

-- Find broadcasting towers that receiver with limited range can receive.
-- Data is geometry in Spherical Mercator (SRID=3857), ranges are approximate.

-- Create geometry index that will check proximity limit of user to tower
CREATE INDEX ON broadcasting_towers using gist (geom);

-- Create geometry index that will check proximity limit of tower to user
CREATE INDEX ON broadcasting_towers using gist (ST_Expand(geom, sending_range));

-- Query towers that 4-kilometer receiver in Minsk Hackerspace can get
-- Note: two conditions, because shorter LEAST(b.sending_range, 4000) will not use index.
SELECT b.tower_id, b.geom
  FROM broadcasting_towers b
 WHERE ST_DWithin(b.geom, 'SRID=3857;POINT(3072163.4 7159374.1)', 4000)
    AND ST_DWithin(b.geom, 'SRID=3857;POINT(3072163.4 7159374.1)', b.sending_range);
```

See Also

[ST_Distance](#), [ST_3DDWithin](#)

8.11.2.5 ST_PointInsideCircle

ST_PointInsideCircle — Tests if a point geometry is inside a circle defined by a center and radius.

Synopsis

boolean **ST_PointInsideCircle**(geometry a_point, float center_x, float center_y, float radius);

Description

Returns true if the geometry is a point and is inside the circle with center `center_x`, `center_y` and radius `radius`.



Warning

Does not use spatial indexes. Use [ST_DWithin](#) instead.


```
&#xb2e4;&#xba74;&#xccb4;    &#xd45c;&#xba74;&#xc758;    &#xacbd;&#xc6b0;,    (2.5&#xcc28;&#xc6d0;&#xc774;
&#xc544;&#xb2c8;&#xb77c;)    2&#xcc28;&#xc6d0;    &#xb2e4;&#xba74;&#xccb4;    &#xd45c;&#xba74;&#xb9cc;
&#xc9c0;&#xc6d0;&#xd569;&#xb2c8;&#xb2e4;.    2.5&#xcc28;&#xc6d0;&#xc77c;    &#xacbd;&#xc6b0;
0&#xc774;    &#xc544;&#xb2cc;(non-zero)    &#xb2f5;&#xc744;    &#xb0b4;&#xb193;&#xc744;    &#xc218;&#xb3c4;
&#xc788;&#xc9c0;&#xb9cc;,    XY    &#xd3c9;&#xba74;&#xc0c1;&#xc5d0;    &#xc644;&#xbcbd;&#xd558;&#xac8c;
&#xb193;&#xc774;&#xb294;    &#xba74;&#xb4e4;&#xc758;    &#xba74;&#xc801;&#xb9cc;
&#xbc18;&#xd658;&#xd558;&#xb294; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

지리형 데이터 유형을 이용해서
평방피트 및 평방미터 면적을 반&#
도형을 지리형으로 변환시킨다&#x
점에 주의하십시오(변환하기 전&#x

WGS84 4326 (SRID=2249; POLYGON((743238 2967416, 743238 2967450, 743265 2967450, 743265.625 2967416, 743238 2967416)))::geometry,

) :: geography geog

) as subquery;

sqft_spheroid sqft_sphere sqm_spheroid

928.684405784452 927.049336105925 86.2776044979692

```
select ST_Area(geog) / 0.3048 ^ 2 sqft_spheroid,
       ST_Area(geog, false) / 0.3048 ^ 2 sqft_sphere,
       ST_Area(geog) sqm_spheroid
from (
    select ST_Transform(
        'SRID=2249;POLYGON((743238 2967416,743238 2967450,743265
        2967450,743265.625 2967416,743238 2967416))'::geometry,
        4326
    ) :: geography geog
    ) as subquery;
```

sqft_spheroid sqft_sphere sqm_spheroid

928.684405784452 927.049336105925 86.2776044979692

If your data is in geography already:

```
select ST_Area(geog) / 0.3048 ^ 2 sqft,
       ST_Area(the_geog) sqm
from somegeogtable;
```

ST_3DArea, ST_GeomFromEWKT, ST_LengthSpheroid, ST_Perimeter, ST_Transform

8.12.2 ST_Azimuth

ST_Azimuth — Returns the azimuth in radians of the target point from the origin point, or NULL if the two points are coincident. The azimuth angle is a positive clockwise angle referenced from the positive Y axis (geometry) or the North meridian (geography): North = 0; Northeast = $\pi/4$; East = $\pi/2$; Southeast = $3\pi/4$; South = π ; Southwest = $5\pi/4$; West = $3\pi/2$; Northwest = $7\pi/4$.

Synopsis

```
float ST_Azimuth(geometry origin, geometry target);
float ST_Azimuth(geography origin, geography target);
```

ST_Azimuth

Returns the azimuth in radians of the target point from the origin point, or NULL if the two points are coincident. The azimuth angle is a positive clockwise angle referenced from the positive Y axis (geometry) or the North meridian (geography): North = 0; Northeast = $\pi/4$; East = $\pi/2$; Southeast = $3\pi/4$; South = π ; Southwest = $5\pi/4$; West = $3\pi/2$; Northwest = $7\pi/4$.

For the geography type, the azimuth solution is known as the [inverse geodesic problem](#).

The azimuth is a mathematical concept defined as the angle between a reference vector and a point, with angular units in radians. The result value in radians can be converted to degrees using the PostgreSQL function `degrees()`.

ST_Azimuth

PostGIS wiki section

1.1.0
2.0.0
2.2.0

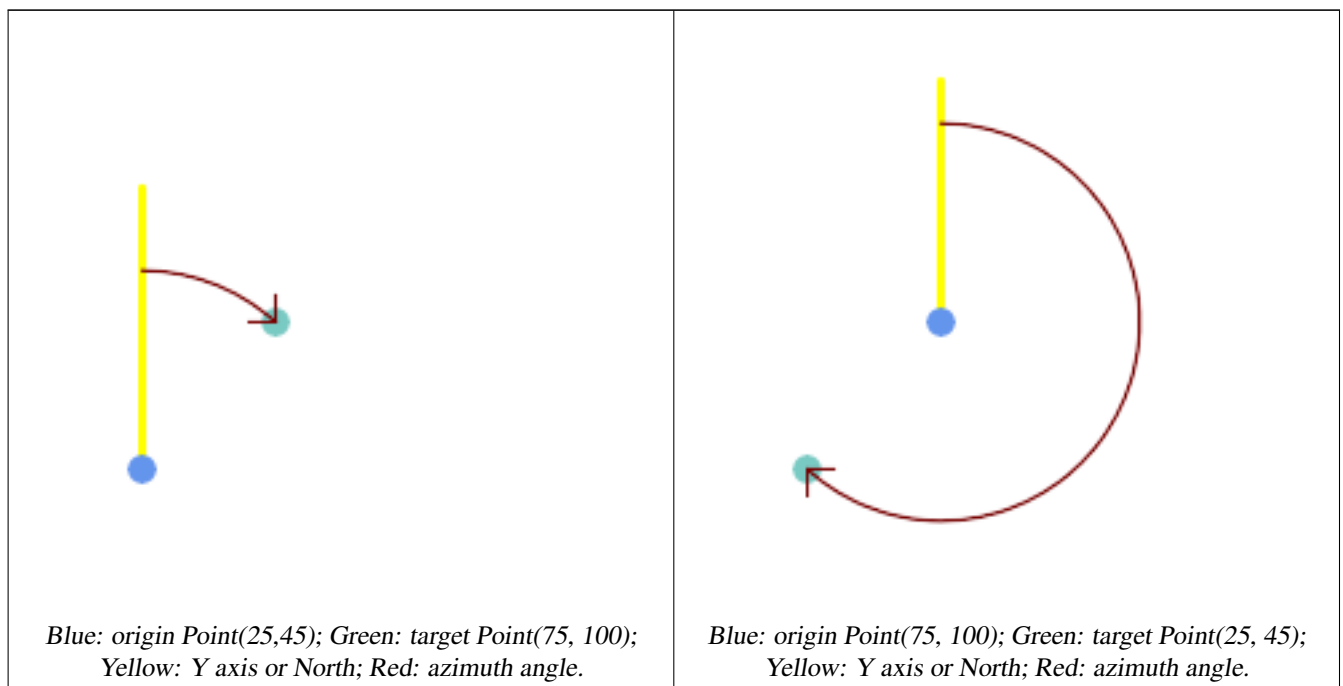
GeographicLib
Proj 4.9.0

degrees

degrees

```
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



ST_Angle

ST_Angle, ST_Translate, ST_Project, PostgreSQL Math Functions

8.12.3 ST_Angle

ST_Angle — longest

Synopsis

```
float ST_Angle(geometry point1, geometry point2, geometry point3, geometry point4);
float ST_Angle(geometry line1, geometry line2);
```

Usage

ST_Angle(geom1, geom2, geom3, geom4) returns the angle in degrees between the line segments defined by the four points. The angle is measured from the line segment defined by the first two points to the line segment defined by the last two points. The angle is always between 0 and 180 degrees.

Variant 1: computes the angle enclosed by the points P1-P2-P3. If a 4th point provided computes the angle points P1-P2 and P3-P4

Variant 2: computes the angle between two vectors S1-E1 and S2-E2, defined by the start and end points of the input lines

ST_Angle(geom1, geom2, geom3, geom4) returns the angle in degrees between the line segments defined by the four points. The angle is measured from the line segment defined by the first two points to the line segment defined by the last two points. The angle is always between 0 and 180 degrees.

Note that $ST\_Angle(P1, P2, P3) = ST\_Angle(P2, P1, P2, P3)$.

Availability: 2.5.0

Examples

ST_Angle(geom1, geom2, geom3, geom4) returns the angle in degrees between the line segments defined by the four points. The angle is measured from the line segment defined by the first two points to the line segment defined by the last two points. The angle is always between 0 and 180 degrees.

```
SELECT degrees( ST_Angle('POINT(0 0)', 'POINT(10 10)', 'POINT(20 0)') );
```

```
degrees
-----
270
```

Angle between vectors defined by four points

```
SELECT degrees( ST_Angle('POINT (10 10)', 'POINT (0 0)', 'POINT(90 90)', 'POINT (100 80)') ) ←
);
```

```
degrees
-----
269.999999999999
```

Angle between vectors defined by the start and end points of lines

```
SELECT degrees( ST_Angle('LINESTRING(0 0, 0.3 0.7, 1 1)', 'LINESTRING(0 0, 0.2 0.5, 1 0)') ) ←
);
```

```
degrees
-----
45
```

See also

[ST_Azimuth](#)

8.12.4 ST_ClosestPoint

ST_ClosestPoint — Returns the 2D point on g1 that is closest to g2. This is the first point of the shortest line from one geometry to the other.

Synopsis

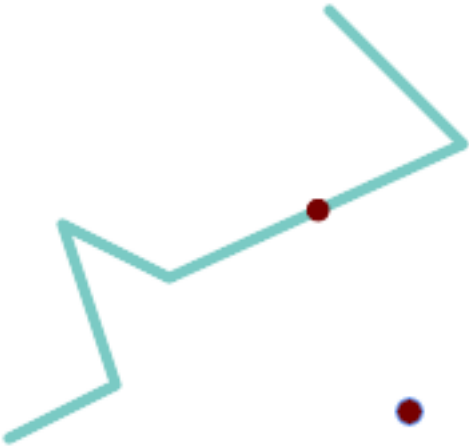
geometry **ST_ClosestPoint**(geometry geom1, geometry geom2);

Returns the 2-dimensional point on geom1 that is closest to geom2. This is the first point of the shortest line between the geometries (as computed by [ST_ShortestLine](#)).



Note
3D version: [ST_3DClosestPoint](#)

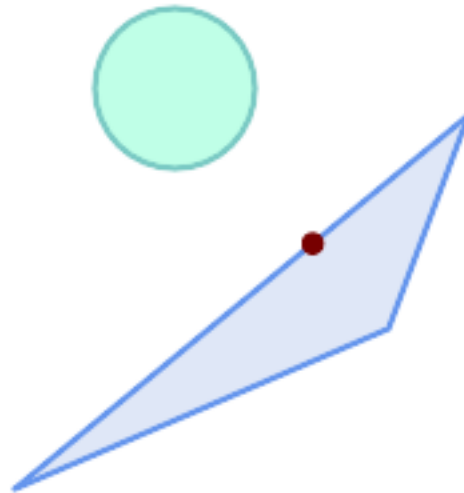
1.5.0 version: [ST_ClosestPoint](#)



The closest point for a Point and a LineString is the point itself. The closest point for a LineString and a Point is a point on the line.

```
SELECT ST_AsText( ST_ClosestPoint(pt,line)) AS cp_pt_line,
       ST_AsText( ST_ClosestPoint(line,pt)) AS cp_line_pt
FROM (SELECT 'POINT (160 40)::geometry AS pt,
            'LINESTRING (10 30, 50 50, 30 110, 70 90, 180 140, 130 190)::geometry AS line ) AS t;

cp_pt_line | cp_line_pt
-----+-----
POINT(160 40) | POINT(125.75342465753425 115.34246575342466)
```



The closest point on polygon A to polygon B

```
SELECT ST_AsText( ST_ClosestPoint(
    'POLYGON ((190 150, 20 10, 160 70, 190 150))',
    ST_Buffer('POINT(80 160)', 30)
)) As ptwkt;
-----
POINT(131.59149149528952 101.89887534906197)
```

స고

ST_3DClosestPoint, ST_Distance, ST_LongestLine, ST_ShortestLine, ST_MaxDistance

8.12.5 ST_3DClosestPoint

```
ST_3DClosestPoint — g2&#xc5d0; &#xac00;&#xc7a5; &#xac00;&#xae4c;&#xc6b4; g1 &#xc0c1;&#xc5d0; &#xc788;&#xb294;
3&#xcc28;&#xc6d0; &#xd3ec;&#xc778;&#xd2b8;&#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. &#xd574;&#xb2f9;
&#xd3ec;&#xc778;&#xd2b8;&#xb294; 3D &#xcd5c;&#xb2e8; &#xb77c;&#xc778;&#xc758; &#xccab; &#xbc88;&#xc9f8;
&#xd3ec;&#xc778;&#xd2b8;&#xc785;&#xb2c8;&#xb2e4;.
```

Synopsis

```
geometry ST_3DClosestPoint(geometry g1, geometry g2);
```

설명

g2에 가장 가까운 g1 상에 있는 3차원
 포인트를 반환합니다. 해당 포인&#x
 3D 최단 라인의 첫 번째 포인트입니
 3D 최단 라인의 3D 길이가 3D 거리입니&#x



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

2.0.0 버전부터 사용할 수 있습니다.

&#x0000; 변 경 사 항 : 2.2.0 버 전 부 터 2D 도 형 두 개 &#입 력 할 경 우. (존 재 하 지 않 는 Z을

0으로가정하는예전습성대신) 2D
포인트를반환합니다. 2D 및 3D의경&#xc
더이상 Z가없을때 Z를 0으로가정&#xd55
않습니다.

예시

ݼ인스트링૏ 포인트 -- 3D, 2D
ન두의 최근접점(closest point)

```
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) ←
```

ݼݸʤ트링과 &#ba40;티포ݸ트 -- 3D, 2D
&#baa8;&#b450;의 최근접점(closest point)

```
SELECT ST_AsEWKT(ST_3DClosestPoint(line,pt)) AS cp3d_line_pt,
       ST_AsEWKT(ST_ClosestPoint(line,pt)) As cp2d_line_pt
FROM (SELECT 'MULTIPOINT(100 100 30, 50 74 1000) '::geometry As pt,
            'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 900) '::
            geometry As line
      ) As foo;
```

cp3d_line_pt	cp2d_line_pt
POINT(54.6993798867619 128.935022917228 11.5475869506606)	POINT(50 75)

멀티라인스트링과 폴리곤 -- 3D, 2D
ન두의 최근접점(closest point)

```
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)
```

స고

ST_AsEWKT, ST_ClosestPoint, ST_3DDistance, ST_3DShortestLine

8.12.6 ST_Distance

ST_Distance — (longest) distance between two geometries in projected units.

Synopsis

```
float ST_HausdorffDistance(geometry g1, geometry g2);
float ST_HausdorffDistance(geometry g1, geometry g2, float densifyFrac);
```

Parameters

g1, g2 — Geometry objects. If **g1** and **g2** are both **POINT** types, the function returns the distance between the two points. If **g1** is a **POINT** and **g2** is a **LINESTRING**, the function returns the distance from the point to the line. If both **g1** and **g2** are **LINESTRING** types, the function returns the distance between the two lines. If **g1** is a **POLYGON** and **g2** is a **POINT**, the function returns the distance from the point to the polygon. If both **g1** and **g2** are **POLYGON** types, the function returns the distance between the two polygons.

For **geography** types defaults to return the minimum geodesic distance between two geographies in meters, compute on the spheroid determined by the SRID. If **use_spheroid** is false, a faster spherical calculation is used.



This method implements the **OGC 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

1.5.0 — **ST_Distance** (geometry g1, geometry g2) returns the distance between g1 and g2 in the units of g1. If g1 and g2 are both **POINT** types, the function returns the distance between the two points. If g1 is a **POINT** and g2 is a **LINESTRING**, the function returns the distance from the point to the line. If both g1 and g2 are **LINESTRING** types, the function returns the distance between the two lines. If g1 is a **POLYGON** and g2 is a **POINT**, the function returns the distance from the point to the polygon. If both g1 and g2 are **POLYGON** types, the function returns the distance between the two polygons.

2.1.0 — **ST_Distance** (geometry g1, geometry g2, float densifyFrac) returns the distance between g1 and g2 in the units of g1. If g1 and g2 are both **POINT** types, the function returns the distance between the two points. If g1 is a **POINT** and g2 is a **LINESTRING**, the function returns the distance from the point to the line. If both g1 and g2 are **LINESTRING** types, the function returns the distance between the two lines. If g1 is a **POLYGON** and g2 is a **POINT**, the function returns the distance from the point to the polygon. If both g1 and g2 are **POLYGON** types, the function returns the distance between the two polygons.

2.1.0 — **ST_Distance** (geography g1, geography g2) returns the distance between g1 and g2 in meters. If g1 and g2 are both **POINT** types, the function returns the distance between the two points. If g1 is a **POINT** and g2 is a **LINESTRING**, the function returns the distance from the point to the line. If both g1 and g2 are **LINESTRING** types, the function returns the distance between the two lines. If g1 is a **POLYGON** and g2 is a **POINT**, the function returns the distance from the point to the polygon. If both g1 and g2 are **POLYGON** types, the function returns the distance between the two polygons.

2.2.0 — **ST_Distance** (geography g1, geography g2, boolean use_spheroid) returns the distance between g1 and g2 in meters. If g1 and g2 are both **POINT** types, the function returns the distance between the two points. If g1 is a **POINT** and g2 is a **LINESTRING**, the function returns the distance from the point to the line. If both g1 and g2 are **LINESTRING** types, the function returns the distance between the two lines. If g1 is a **POLYGON** and g2 is a **POINT**, the function returns the distance from the point to the polygon. If both g1 and g2 are **POLYGON** types, the function returns the distance between the two polygons.

Changed: 3.0.0 - does not depend on SFCGAL anymore.

Geometry example - units in planar degrees 4326 is WGS 84 long lat, units are degrees.

```
SELECT ST_Distance (
  'SRID=4326;POINT(-72.1235 42.3521) '::geometry,
  'SRID=4326;LINESTRING(-72.1260 42.45, -72.123 42.1546) '::geometry );
-----
0.00150567726382282
```

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) ) ←
;
-----
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);
-----
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) ←
);
-----
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) ) ←
;
-----
126.664256056812
```

ST_DistanceSphere

Same as geometry example but note units in meters - use sphere for slightly faster and less accurate computation.

```
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
```

ST_DistanceSphere

ST_3DDistance, **ST_DWithin**, **ST_DistanceSphere**, **ST_DistanceSpheroid**, **ST_MaxDistance**, **ST_HausdorffDistance**, **ST_FrechetDistance**, **ST_Transform**

8.12.7 ST_3DDistance

ST_3DDistance — Compute the 3D distance between two points in a 3D coordinate system. The distance is calculated using the Euclidean distance formula. The function takes two points as input, each defined by its X, Y, and Z coordinates. The distance is returned in the units of the input coordinates.


```

        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

```

󌰸고

ST_Distance, ST_3DClosestPoint, ST_3DDWithin, ST_3DMaxDistance, ST_3DShortestLine, ST_Transform

8.12.8 ST_DistanceSphere

ST_DistanceSphere — 특정 회전타원체가 주어진 두 경위도 도형 사이의 최단 거리&#ଘ환합니다. PostGIS 1.5 미만 버전은 포인&#지원했습니다.

Synopsis

float **ST_DistanceSphere**(geometry geom1lonlatA, geometry geom1lonlatB, float8 radius=6371008);

설명

경위도 포인트 2개 사이의 최단 거&#미터 단위로 ଘ환합니다. SRID가 정&#xc7회전타원체에서 추출한 ଘ경을 가진 지구 구체를 이용합니다. **ST_Distance** 보다는 처리 속도가 &#be60;르지만, 정&#떨어집니다. PostGIS 1.5 미만 버전에서는 포인트에 대해서만 구현돼 있었&#಄전부터 포인트가 아닌 다른 도형 유형을 지원하기 시작했습&#미만 버전에서는 포인트에 대해 구현돼 있었습니다.

변경 사항: 2.2.0 미만 버전에서는 ST_Distance_Sph 명칭이었습니다.

예시

```

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

ST_Distance

ST_DistanceSpheroid

8.12.9 ST_DistanceSpheroid

ST_DistanceSpheroid — Computes the distance between two geometries in spherical coordinates (longitude and latitude) using the WGS84 spheroid. The distance is returned in meters. The function is implemented using the `GeographicLib` library. The function is available in PostGIS 1.5 and later.

Synopsis

`float ST_DistanceSpheroid(geometry geomlonlatA, geometry geomlonlatB, spheroid measurement_spheroid=WGS84);`

ST_DistanceSpheroid

ST_DistanceSpheroid — Computes the distance between two geometries in spherical coordinates (longitude and latitude) using the WGS84 spheroid. The distance is returned in meters. The function is implemented using the `GeographicLib` library. The function is available in PostGIS 1.5 and later.

Note



The `ST_DistanceSpheroid` function is implemented using the `GeographicLib` library. The function is available in PostGIS 1.5 and later. The function is implemented using the `GeographicLib` library. The function is available in PostGIS 1.5 and later.

1.5 `ST_DistanceSpheroid` — Computes the distance between two geometries in spherical coordinates (longitude and latitude) using the WGS84 spheroid. The distance is returned in meters. The function is implemented using the `GeographicLib` library. The function is available in PostGIS 1.5 and later.

2.2.0 `ST_DistanceSpheroid` — Computes the distance between two geometries in spherical coordinates (longitude and latitude) using the WGS84 spheroid. The distance is returned in meters. The function is implemented using the `GeographicLib` library. The function is available in PostGIS 2.2.0 and later.

ST_DistanceSpheroid

```
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

```

ST_Distance

ST_DistanceSphere

8.12.10 ST_FrechetDistance

ST_FrechetDistance — Computes the Fréchet distance between two geometries. 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.

Synopsis

```
float ST_FrechetDistance(geometry g1, geometry g2, float densifyFrac = -1);
```

ST_FrechetDistance

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.

Units are in the units of the spatial reference system of the geometries.

Note



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



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.

GEOS 3.3.0 - requires GEOS >= 3.7.0

Availability: 2.4.0 - requires GEOS >= 3.7.0

ST_FrechetDistance

```
postgres=# SELECT st_frechetdistance('LINESTRING (0 0, 100 0)::geometry, 'LINESTRING (0 0, ↵
          50 50, 100 0)::geometry');
 st_frechetdistance
-----
          70.7106781186548
(1 row)
```

```
SELECT st_frechetdistance('LINESTRING (0 0, 100 0)::geometry, 'LINESTRING (0 0, 50 50, 100 ↵
          0)::geometry, 0.5);
 st_frechetdistance
-----
                    50
(1 row)
```

ST_HausdorffDistance

ST_HausdorffDistance

8.12.11 ST_HausdorffDistance

ST_HausdorffDistance — Returns the Hausdorff distance between two geometries. The Hausdorff distance is a measure of how similar or dissimilar 2 geometries are.

Synopsis

```
float ST_HausdorffDistance(geometry g1, geometry g2);
float ST_HausdorffDistance(geometry g1, geometry g2, float densifyFrac);
```

ST_HausdorffDistance

Returns the **Hausdorff distance** between two geometries. The Hausdorff distance is a measure of how similar or dissimilar 2 geometries are.

The function actually computes the "Discrete Hausdorff Distance". This is the Hausdorff distance computed at discrete points on the geometries. The *densifyFrac* parameter can be specified, to provide a more accurate answer by densifying segments before computing the discrete Hausdorff distance. Each segment is split into a number of equal-length subsegments whose fraction of the segment length is closest to the given fraction.

Units are in the units of the spatial reference system of the geometries.



Note

This algorithm is NOT equivalent to the standard Hausdorff distance. However, it computes an approximation that is correct for a large subset of useful cases. One important case is Linestrings that are roughly parallel to each other, and roughly equal in length. This is a useful metric for line matching.

1.5.0 Returns the Hausdorff distance between two geometries. The Hausdorff distance is a measure of how similar or dissimilar 2 geometries are.



Hausdorff distance (red) and distance (yellow) between two lines

```
SELECT ST_HausdorffDistance(geomA, geomB),
       ST_Distance(geomA, geomB)
FROM (SELECT 'LINESTRING (20 70, 70 60, 110 70, 170 70)::geometry AS geomA,
            'LINESTRING (20 90, 130 90, 60 100, 190 100)::geometry AS geomB) AS t;
st_hausdorffdistance | st_distance
-----+-----
37.26206567625497 |          20
```

Example: Hausdorff distance with densification.

```
SELECT ST_HausdorffDistance(
    'LINESTRING (130 0, 0 0, 0 150)::geometry,
    'LINESTRING (10 10, 10 150, 130 10)::geometry,
    0.5);
-----
70
```

Example: For each building, find the parcel that best represents it. First we require that the parcel intersect with the building geometry. `DISTINCT ON` guarantees we get each building listed only once. `ORDER BY .. ST_HausdorffDistance` selects the parcel that is most similar to the building.

```
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);
```

`ST_FrechetDistance`

8.12.12 ST_Length

`ST_Length` —

ST_Length

ST_Length — Returns the length of a 2D line segment in the units of the SRID. If the SRID is not specified, the length is returned in the units of the geometry's SRID. If the SRID is 0, the length is returned in the units of the geometry's SRID.

```
SELECT ST_Length(ST_GeomFromText('LINESTRING(743238 2967416, 743238 2967450, 743265 2967450, 743265.625 2967416, 743238 2967416)', 2249));
st_length
-----
122.630744000095

-- WGS84
-- ST_Length(ST_GeomFromText('LINESTRING(743238 2967416, 743238 2967450, 743265 2967450, 743265.625 2967416, 743238 2967416)', 2249), 4326);
-- ST_Length(ST_GeomFromText('LINESTRING(743238 2967416, 743238 2967450, 743265 2967450, 743265.625 2967416, 743238 2967416)', 2249), 4326, 4326);
SELECT ST_Length(
    ST_Transform(
        ST_GeomFromEWKT('SRID=4326;LINESTRING(-72.1260 42.45, -72.1240 42.45666, -72.123 42.1546)'),
        4326
    )
);
st_length
-----
34309.4563576191
```

ST_LengthSpheroid

ST_LengthSpheroid — Returns the length of a 2D line segment in the units of the SRID, using a spheroid model for the Earth.

```
-- ST_LengthSpheroid
-- ST_LengthSpheroid(geometry a_2dlinestring, SRID)
-- ST_LengthSpheroid(geometry a_2dlinestring, SRID, SRID)
-- ST_LengthSpheroid(geometry a_2dlinestring, SRID, SRID, SRID)
-- ST_LengthSpheroid(geometry a_2dlinestring, SRID, SRID, SRID, SRID)
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
```

ST_Length2D

ST_Length2D — Returns the length of a 2D line segment in the units of the SRID, using a 2D model for the Earth.

8.12.13 ST_Length2D

ST_Length2D — Returns the length of a 2D line segment in the units of the SRID, using a 2D model for the Earth. The length is returned in the units of the geometry's SRID. If the SRID is 0, the length is returned in the units of the geometry's SRID.

Synopsis

float **ST_Length2D**(geometry a_2dlinestring);

ST_3DLength

ST_3DLength — Returns the 3D length of a geometry in the SRID of the geometry. The SRID is 0 by default. The SRID is 0 by default. The SRID is 0 by default.

ST_3DLength

ST_3DLength, **ST_3DLength**

8.12.14 ST_3DLength

ST_3DLength — Returns the 3D length of a geometry in the SRID of the geometry. The SRID is 0 by default. The SRID is 0 by default. The SRID is 0 by default.

Synopsis

`float ST_3DLength(geometry a_3dlinestring);`

ST_3DLength

ST_3DLength — Returns the 3D length of a geometry in the SRID of the geometry. The SRID is 0 by default. The SRID is 0 by default. The SRID is 0 by default.



This function supports 3d and will not drop the z-index.



This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 7.1, 10.3

ST_3DLength — Returns the 3D length of a geometry in the SRID of the geometry. The SRID is 0 by default. The SRID is 0 by default. The SRID is 0 by default.

ST_3DLength

ST_3DLength — Returns the 3D length of a geometry in the SRID of the geometry. The SRID is 0 by default. The SRID is 0 by default. The SRID is 0 by default.

```
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
```

ST_3DLength

ST_3DLength, **ST_3DLength**

8.12.15 ST_LengthSpheroid

ST_LengthSpheroid — Returns the length of a geometry in the SRID of the geometry. The SRID is 0 by default. The SRID is 0 by default. The SRID is 0 by default.

Synopsis

```
float ST_LengthSpheroid(geometry a_geometry, spheroid a_spheroid);
```

설명

타원체 상에 있는 도형의 길이/둘계산합니다. 이 함수는 도형의 좌경도/위도이며 재투영하지 않고 길이를 얻고 싶을 때 유용합니다. 이 타원체는 개별적인 데이터베유형으로 다음과 같이 작성할 수 있습니다:

```
SPHEROID [<NAME>, <SEMI-MAJOR AXIS>, <INVERSE FLATTENING>]
```

도형 예시

```
SPHEROID ["GRS_1980", 6378137, 298.257222101]
```

1.2.2 버전부터 사용할 수 있습니다.

변경 사항: 2.2.0 미만 버전에서는 ST_Length_Spheroid 명칭이었으며., ST_3DLength_Spheroid라는 동일 할있었습니다.



This function supports 3d and will not drop the z-index.

예시

```
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
```

ST_LongestLine

ST_GeometryN, ST_Length

8.12.16 ST_LongestLine

ST_LongestLine — Returns the 2-dimensional longest line between the points of two geometries. The line returned starts on g1 and ends on g2. The longest line always occurs between two vertices. The function returns the first longest line if more than one is found. The length of the line is equal to the distance returned by **ST_MaxDistance**.

Synopsis

geometry **ST_LongestLine**(geometry g1, geometry g2);

ST_LongestLine

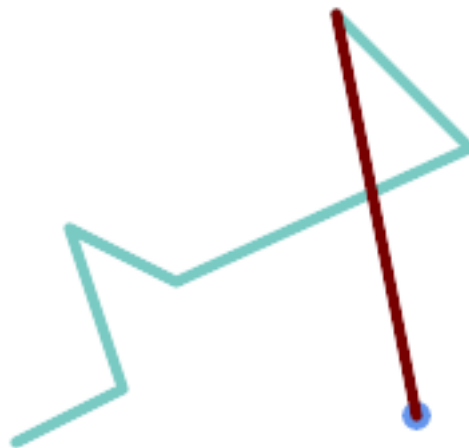
Returns the 2-dimensional longest line between the points of two geometries. The line returned starts on g1 and ends on g2.

The longest line always occurs between two vertices. The function returns the first longest line if more than one is found. The length of the line is equal to the distance returned by **ST_MaxDistance**.

If g1 and g2 are the same geometry, returns the line between the two vertices farthest apart in the geometry. This is a diameter of the circle computed by **ST_MinimumBoundingCircle**

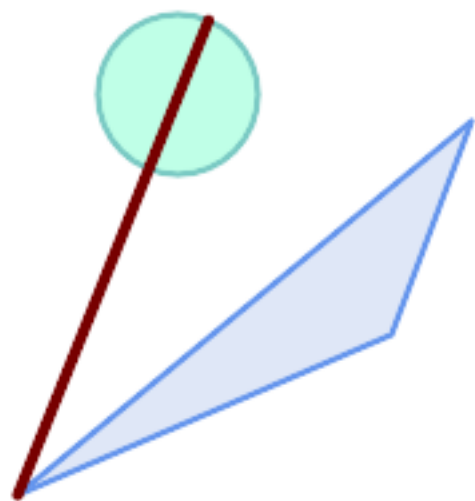
1.5.0 Returns the 2-dimensional longest line between the points of two geometries. The line returned starts on g1 and ends on g2.

ST_LongestLine



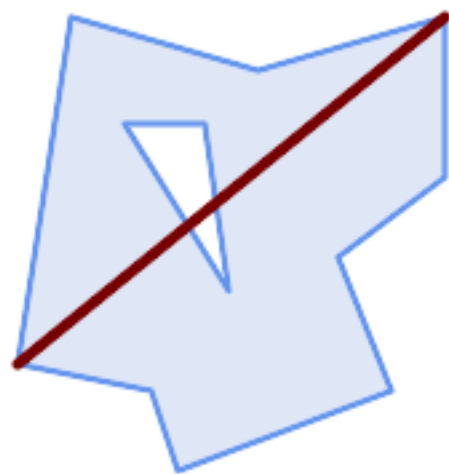
ST_LongestLine

```
SELECT ST_AsText( ST_LongestLine(
    'POINT (160 40)',
    'LINESTRING (10 30, 50 50, 30 110, 70 90, 180 140, 130 190)' )
) AS lline;
-----
LINESTRING(160 40,130 190)
```



Longest line across a single geometry. The length of the line is equal to the Maximum Distance. The line is a diameter of the Minimum Bounding Circle.

```
SELECT ST_AsText( ST_LongestLine(
    'POLYGON ((190 150, 20 10, 160 70, 190 150))',
    ST_Buffer('POINT(80 160)', 30)
) ) AS llinewkt;
-----
LINESTRING(20 10,105.3073372946034 186.95518130045156)
```



Longest line across a single geometry. The length of the line is equal to the Maximum Distance. The line is a diameter of the Minimum Bounding Circle.

```
SELECT ST_AsText( ST_LongestLine( geom, geom)) AS llinewkt,
       ST_MaxDistance( geom, geom) AS max_dist,
       ST_Length( ST_LongestLine(geom, geom)) AS lenll
FROM (SELECT 'POLYGON ((40 180, 110 160, 180 180, 180 120, 140 90, 160 40, 80 10, 70 40, 20 50, 40 180),
       (60 140, 99 77.5, 90 140, 60 140))'::geometry AS geom) AS t;

-----
llinewkt | max_dist | lenll
-----+-----+-----
LINESTRING(20 50,180 180) | 206.15528128088303 | 206.15528128088303
```

స고

ST_MaxDistance, ST_ShortestLine, ST_3DLongestLine, ST_MinimumBoundingCircle

8.12.17 ST_3DLongestLine

ST_3DLongestLine — [#b450](#); [#b3c4](#); [#d615](#); [#xc0ac](#); [#xc774](#); [#xc758](#); [3#xc28](#); [#xc6d0](#); [#xcd5c](#); [#xc7a5](#); (longest) [#b77c](#); [#xc778](#); [#xc744](#); [#bc18](#); [#d658](#); [#d569](#); [#b2c8](#); [#b2e4](#)..

Synopsis

```
geometry ST_3DLongestLine(geometry g1, geometry g2);
```

설명

두 도 형 사 이 의 3차 원 최 장 (longest) 라 인 &#
반 환 합 니 다. 하 나 이 상 의 최 장 라 &
있 을 경 우. 이 함 수 는 첫 번 째 최 장
라 인 만 반 환 할 것 입 니 다. 반 환 되 &#
라 인 은 항 상 g1에 서 시 작 해 서 g2에 서
끝 납 니 다. 이 함 수 가 반 환 하 는 라 &
3차 원 길 이 는 **ST_3DMaxDistance** 함 수 가 g1과 g2에
대 해 반 환 하 는 길 이 와 언 제 나 동 &
2.0.0 버 전 부 터 사 용 할 수 있 습 니 다.

경 사항; 2.2.0 버전부터 2D 도형 두 개&#x
입력할 경우, (존재하지 않는 Z을
0으로 가정하는 예전 습성 대신) 2D
포인트를 반환합니다. 2D 및 3D의 경&#xc
더 이상 Z가 없을 때 Z를 0으로 가정&#xd55
않습니다.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

예시

```
&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xacfc;&#xd3ec;&#xc778;&#xd2b8; -- 3D, 2D  
&#xaa8;&#xb450;&#xc758; &#xcd5c;&#xc7a5; &#xb77c;&#xc778;
```

```
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)

```
&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xacfc; &#xba40;&#xd2f0;&#xd3ec;&#xc778;&#xd2b8; -- 3D, 2D
&#xaa8;&#xb450;&#xc758; &#xcd5c;&#xc7a5; &#xb77c;&#xc778;
```

```
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)
```

```
&#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xacfc; &#xd3f4;&#xb9ac;&#xace4; -- 3D, 2D
&#xaa8;&#xb450;&#xc758; &#xcd5c;&#xc7a5; &#xb77c;&#xc778;
```

```
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)
```

참고

ST_3DClosestPoint, ST_3DDistance, ST_LongestLine, ST_3DShortestLine, ST_3DMaxDistance

8.12.18 ST_MaxDistance

ST_MaxDistance — 두 도형 사이의 2차원 최장 거리를 투영 단위로 반환합니다.

Synopsis

float **ST_MaxDistance**(geometry g1, geometry g2);

설명

두 도형 사이의 2차원 최장 거리를 투영 단위로 반환합니다. g1과 g2가 동일한 도형일 경우 이 함수는 해도형 내에서 서로 가장 멀리 있는 두 꼭짓점 사이의 거리를 반환합니다.

두 도형 사이의 2차원 최장 거리를 투영 단위로 반환합니다. g1과 g2가 동일한 도형일 경우 이 함수는 해도형 내에서 서로 가장 멀리 있는 두 꼭짓점 사이의 거리를 반환합니다.

1.5.0 버전부터 사용할 수 있습니다.

ST_MaxDistance

ST_MaxDistance(geometry g1, geometry g2)

```
SELECT ST_MaxDistance('POINT(0 0)::geometry', 'LINESTRING ( 2 0, 0 2 )::geometry');
-----
2

SELECT ST_MaxDistance('POINT(0 0)::geometry', 'LINESTRING ( 2 2, 2 2 )::geometry');
-----
2.82842712474619
```

Maximum distance between vertices of a single geometry.

```
SELECT ST_MaxDistance('POLYGON ((10 10, 10 0, 0 0, 10 10))::geometry',
                      'POLYGON ((10 10, 10 0, 0 0, 10 10))::geometry');
-----
14.142135623730951
```

ST_Distance

ST_LongestLine, **ST_DFullyWithin**

8.12.19 ST_3DMaxDistance

ST_3DMaxDistance — Returns the maximum 3D distance between any two vertices of the two input geometries. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used.

Synopsis

float **ST_3DMaxDistance**(geometry g1, geometry g2);

ST_3DMaxDistance

ST_3DMaxDistance(geometry g1, geometry g2) — Returns the maximum 3D distance between any two vertices of the two input geometries. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

2.0.0 — Returns the maximum 3D distance between any two vertices of the two input geometries. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used.

2.2.0 — Returns the maximum 3D distance between any two vertices of the two input geometries. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used. The input geometries must be of the same SRID. If the SRID is not specified, the SRID of the first geometry is used.

ST_3DMaxDistance


```
-- &#xb3c4;&#xd615; &#xc608;&#xc2dc; - &#xbbf8;&#xd130; &#xb2e8;&#xc704; (SRID 2163 ↔
&#xbbf8;&#xad6d; &#xb4f1;&#xc801; &#xc804;&#xb3c4;) (2D &#xd3ec;&#xc778;&#xd2b8; ↔
&#xbc0f; &#xb77c;&#xc778;&#xacfc; &#be44;&#xad50;&#xd55c; 3D &#xd3ec;&#xc778;&#xd2b8; ↔
&#xbc0f; &#xb77c;&#xc778;)
-- &#xc8fc;&#xc758;: &#xd604;&#xc7ac; &#xc218;&#xc9c1; &#xb370;&#xc774;&#xd130;&#xb97c; ↔
&#xc9c0;&#xc6d0;&#xd558;&#xc9c0; &#xc54a;&#xc73c;&#xbbc0;&#xb85c; Z&#xb97c; ↔
&#xbcc0;&#xd658;&#xd558;&#xc9c0; &#xc54a;&#xace0; &#xcd5c;&#xc885; ↔
&#xacb0;&#xacfc;&#xbb3c;&#xacfc; &#xb3d9;&#xc77c;&#xd55c; &#xb2e8;&#xc704;&#xb85c; ↔
&#xac00;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.
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
```

참고

[ST_Distance](#), [ST_3DDWithin](#), [ST_3DMaxDistance](#), [ST_Transform](#)

8.12.20 ST_MinimumClearance

ST_MinimumClearance — 도형의 튼튼함(robustness)의 척도도형의 최소 여유(clearance)를 반환합니다

Synopsis

float **ST_MinimumClearance**(geometry g);

설명

(폴리곤일 경우) ST_IsValid 또는 (라인일 경 ST_IsSimple 함수에 따라 유효성에 대한 기준을 만족시키지만, 텍스트 기형식(WKT, KML, GML GeoJSON 등) 또는 이중 정밀도 부동소수점 좌표를 이용하지 않로 이너리 형식(MapInfo TAB 등) 으로 변환하정에서 일어날 수 있는 것처럼, 꼭짓점 가운데 하나가 살짝 이동유효하지 않아지는 도형이 그렇희ૐ한 것은 아닙니다.

The minimum clearance is a quantitative measure of a geometry's robustness to change in coordinate precision. It is the largest distance by which vertices of the geometry can be moved without creating an invalid geometry. Larger values of minimum clearance indicate greater robustness.

ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

- ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.
- ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

To avoid validity issues caused by precision loss, **ST_ReducePrecision** can reduce coordinate precision while ensuring that polygonal geometry remains valid.

2.3.0 ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

ST_MinimumClearanceLine

```
SELECT ST_MinimumClearanceLine('POLYGON ((0 0, 1 0, 1 1, 0.5 3.2e-4, 0 0))');
st_minimumclearance
-----
0.00032
```

ST_MinimumClearanceLine

ST_MinimumClearanceLine, **ST_Crosses**, **ST_Dimension**, **ST_Intersects**

8.12.21 ST_MinimumClearanceLine

ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

Synopsis

Geometry **ST_MinimumClearanceLine**(**geometry** g);

ST_MinimumClearanceLine

Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

GEOS ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

2.3.0 ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

GEOS 3.6.0 ST_MinimumClearanceLine(**geometry** g) — Returns the two-point LineString spanning a geometry's minimum clearance. If the geometry does not have a minimum clearance, **LINESTRING EMPTY** is returned.

예시

```
SELECT ST_AsText(ST_MinimumClearanceLine('POLYGON ((0 0, 1 0, 1 1, 0.5 3.2e-4, 0 0))'));
-----
LINESTRING(0.5 0.00032,0.5 0)
```

󌰸고

ST_MinimumClearance

8.12.22 ST_Perimeter

ST_Perimeter — Returns the length of the boundary of a polygonal geometry or geography.

Synopsis

```
float ST_Perimeter(geometry g1);  
float ST_Perimeter(geography geog, boolean use_spheroid=true);
```

설명

도 형 /지 리 형 이 ST_Surface, ST_MultiSurface(폴 리 곤 , 멀 경 우 도 형 /지 리 형 의 2차 원 둘 레 를 반 환 합 니 다 , 면 이 없 는 도 형 의 경 우 을 반 환 합 니 다 , 선 형 도 형 의 경 우 **ST_Length** 를 이 용 하 십 시 오 , 도 형 유 형 의 경 우 , 도 형 의 공 간 참 조 시 스 템 이 해 당 둘 레 의 측 정 단 위 를 설 정 합 지 리 형 의 경 우 : 측 지 역 문 제 (geodesic inverse problem) 를 이 용 해 서 계 산 을 수 행 합 니 이 때 둘 레 단 위 는 미 터 입 니 다 . Post-GIS 를 PROJ 4.8.0 이 상 버 전 과 함 께 컴 파 일 한 경 우 , SRID가 회 정 타 원 체 를 정 의 합 니 SRID가 없 는 경 우 오 직 WGS84로 설 정 됩 니 use_spheroid=false 일 경 우 , 회 전 타 원 체 대 신 구 체 상 에 서 근 사 치 를 계 산 할 것 이 함 수 는 현 재 ST_Perimeter2D와 동 일 하 지 동 향 후 더 높 은 차 원 을 지 원 하 기 위 동 변 경 될 수 도 있 습 니 다 .



This method implements the **OGC 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

```
&#xacc; &#x120; &#x0ac; &#xd56d;: 2.0.0 &#xbc84; &#xc804; &#xbd80; &#xd130; &#xc9c0; &#xb9ac; &#xd615; &#xc744; &#xc9c0; &#xc6d0; &#xd569; &#xb2c8; &#xb2e4;.
```

예&#xc2dc:: 도&#xd615:

폴리곤및멀티폴리곤의둘레를
피트단위로반환합니다. 투영체
EPSG:2249가매사추세츠주피트단위평&
때문에피트단위라는사실을주&#

```
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)
```

WGS84 to UTM conversion

WGS84 to UTM conversion. The UTM zone is 18Q, the UTM easting is 37.3790462565251, the UTM northing is 122.634666195949.

```
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

-- WGS84 to UTM conversion --

```
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

ST_GeogFromText, ST_GeomFromText, ST_Length

ST_GeogFromText, ST_GeomFromText, ST_Length

8.12.23 ST_Perimeter2D

ST_Perimeter2D — Returns the 2D perimeter of a polygonal geometry. Alias for **ST_Perimeter**.

Synopsis

```
float ST_Perimeter2D(geometry geomA);
```

ST_Perimeter2D

ST_Perimeter2D (geometry geomA) returns the 2D perimeter of the polygonal geometry geomA. The result is of type float. The function is commutative. The function is not null-safe. If geomA is null, the function returns null.

Note



ST_Perimeter2D (geometry geomA) returns the 2D perimeter of the polygonal geometry geomA. The result is of type float. The function is commutative. The function is not null-safe. If geomA is null, the function returns null.

ST_Perimeter

ST_Perimeter

8.12.24 ST_3DPerimeter

ST_3DPerimeter — Returns the 3D perimeter of a polygonal geometry. Alias for **ST_3DPerimeter**.

Synopsis

```
float ST_3DPerimeter(geometry geomA);
```

ST_3DPerimeter

ST_3DPerimeter (geometry geomA) returns the 3D perimeter of the polygonal geometry geomA. The result is of type float. The function is commutative. The function is not null-safe. If geomA is null, the function returns null.



This function supports 3d and will not drop the z-index.



This method implements the SQL/MM specification. SQL-MM ISO/IEC 13249-3: 8.1, 10.5

ST_3DPerimeter (geometry geomA) returns the 3D perimeter of the polygonal geometry geomA. The result is of type float. The function is commutative. The function is not null-safe. If geomA is null, the function returns null.

ST_3DPerimeter

ST_3DPerimeter (the_geom) RETURNS NUMERIC, the 3D perimeter of the geometry in the SRID of the geometry. The geometry must be a closed surface.

```
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

ST_GeomFromEWKT

ST_GeomFromEWKT, **ST_Perimeter**, **ST_Perimeter2D**

8.12.25 ST_Project

ST_Project (geometry, float distance, float azimuth) RETURNS GEOMETRY, the point on the line segment defined by the geometry and the distance and azimuth. The geometry must be a line segment.

Synopsis

geography **ST_Project**(geography g1, float distance, float azimuth);

ST_Project

ST_Project (geometry, float distance, float azimuth) RETURNS GEOMETRY, the point on the line segment defined by the geometry and the distance and azimuth. The geometry must be a line segment.

ST_Project (geography g1, float distance, float azimuth) RETURNS GEOGRAPHY, the point on the line segment defined by the geography and the distance and azimuth. The geography must be a line segment.

ST_Project (geometry, float distance, float azimuth) RETURNS GEOMETRY, the point on the line segment defined by the geometry and the distance and azimuth. The geometry must be a line segment.

2.0.0 **ST_Project** (geometry, float distance, float azimuth) RETURNS GEOMETRY, the point on the line segment defined by the geometry and the distance and azimuth. The geometry must be a line segment.

Enhanced: 2.4.0 Allow negative distance and non-normalized azimuth.

ST_Project (geometry, float distance, float azimuth) RETURNS GEOMETRY, the point on the line segment defined by the geometry and the distance and azimuth. The geometry must be a line segment.

```
SELECT ST_AsText(ST_Project('POINT(0 0)::geography, 100000, radians(45.0)));
POINT(0.635231029125537 0.639472334729198)
```

󌰸고

[ST_Azimuth](#), [ST_Distance](#), [PostgreSQL Math Functions](#)

8.12.26 ST_ShortestLine

ST_ShortestLine — Returns the 2-dimensional shortest line between two geometries. The line returned starts in `geom1` and ends in `geom2`. If `geom1` and `geom2` intersect the result is a line with start and end at an intersection point. The length of the line is the same as [ST_Distance](#) returns for `g1` and `g2`.

Synopsis

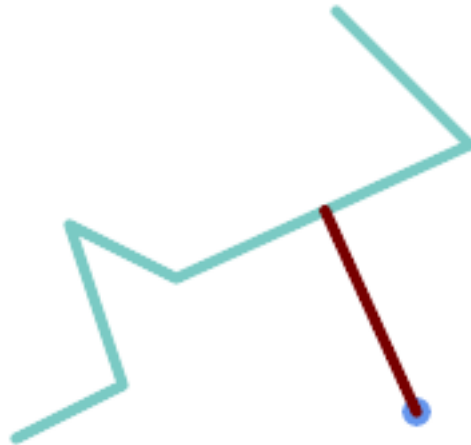
geometry **ST_ShortestLine**(geometry `geom1`, geometry `geom2`);

설명

Returns the 2-dimensional shortest line between two geometries. The line returned starts in `geom1` and ends in `geom2`. If `geom1` and `geom2` intersect the result is a line with start and end at an intersection point. The length of the line is the same as [ST_Distance](#) returns for `g1` and `g2`.

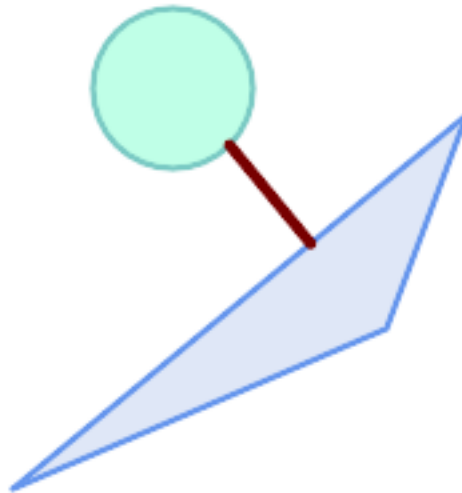
1.5.0 버전부터 사용할 수 있습니다.

예시



Shortest line between Point and LineString

```
SELECT ST_AsText( ST_ShortestLine(
    'POINT (160 40)',
    'LINESTRING (10 30, 50 50, 30 110, 70 90, 180 140, 130 190)' )
) As sline;
-----
LINESTRING(160 40,125.75342465753425 115.34246575342466)
```



Shortest line between Polygons

```
SELECT ST_AsText( ST_ShortestLine(
    'POLYGON ((190 150, 20 10, 160 70, 190 150))',
    ST_Buffer('POINT(80 160)', 30)
    ) ) AS llinewkt;
-----
LINESTRING(131.59149149528952 101.89887534906197,101.21320343559644 138.78679656440357)
```

స고

ST_ClosestPoint, ST_Distance, ST_LongestLine, ST_MaxDistance

8.12.27 ST_3DShortestLine

ST_3DShortestLine — [#b450](#); [#b3c4](#); [#d615](#); [#c0ac](#); [#c774](#); [#c758](#); [3#xc28](#); [#xc6d0](#); [#xcd5c](#); [#b2e8](#); (shortest [#b77c](#); [#c778](#); [#c744](#); [#bc18](#); [#d658](#); [#d569](#); [#b2c8](#); [#b2e4](#)).

Synopsis

```
geometry ST_3DShortestLine(geometry g1, geometry g2);
```

설명

두 도 형 사 이 의 3 차 원 최 단 (shortest) 라 인 򼬘 환 합 니 다. 하 나 이 상 의 최 단 라 있 을 경 우, 이 함 수 는 첫 번 째 최 단 라 인 만 򼬘 환 할 것 입 니 다. g1 과 g2 가 단 한 개 의 포 인 트 에 서 만 교 차 할 경 우, 이 함 수 는 교 차 점 에 서 시 작 끝 나 는 라 인 을 򼬘 환 할 것 입 니 다. g1 과 g2 가 한 개 이 상 의 포 인 트 에 서 교 차 할 경 우, 이 함 수 는 동 일 한 포 시 작 하 고 끝 나 는 라 인 을 򼬘 환 하 해 당 포 인 트 는 교 차 하 는 포 인 트 가 운 데 어 떤 포 인 트 라 도 될 수 있 라

&#x18; ٘ И ʔ ݼ 인 은 항 상 g1 에 서 시 작 g2 에 서 끝 납 니 다. 이 함 수 가 &#x18; 환 &#xd558 ݼ 인 의 3 న 원 길 이 는 **ST_3DDistance** 함 수 가 g1 과 g2 에 대 해 &#x18; 환 하 는 길 이 와 언 &#xc81 동 일 합 니 다.

2.0.0 ಄전부터 사용할 수 있습니다.

변경사항; 2.2.0 버전부터 2D 도형 두 개&#x
입력할 경우, (존재하지 않는 Z을
0으로 가정하는 예전 습성 대신) 2D
포인트를 반환합니다. 2D 및 3D의 경&#xc
더 이상 Z가 없을 때 Z를 0으로 가정&#xd55
않습니다.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.

예시

라인스트링ૼ 포인트 -- 3D, 2D
ન두의 최단 라인

```
SELECT ST_AsEWKT(ST_3DShortestLine(line,pt)) AS shl3d_line_pt,
        ST_AsEWKT(ST_ShortestLine(line,pt)) As shl2d_line_pt
FROM (SELECT 'POINT(100 100 30) '::geometry As pt,
        'LINESTRING (20 80 20, 98 190 1, 110 180 3, 50 75 1000) ' :: ↵
        geometry As line
        ) As foo;
```

shl3d_line_pt shl2d_line_pt

```

LINESTRING(54.6993798867619 128.935022917228 11.5475869506606,100 100 30) | ←
    LINESTRING(73.0769230769231 115.384615384615,100 100)

```

라인스트링ૼ &#ba40;티포인트 -- 3D, 2D
&#baa8;두의 최단 라인

```
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(54.6993798867619 128.935022917228 11.5475869506606,100 100 30) | LINESTRING ←  
(50 75.50 74)
```

```

SELECT ST_AsEWKT(ST_3DShortestLine(poly, mline)) As shl3d,
       ST_AsEWKT(ST_ShortestLine(poly, mline)) As shl2d
FROM (SELECT ST_GeomFromEWKT('POLYGON((175 150 5, 20 40 5, 35 45 5, 50 60 5,
100 100 5, 175 150 5))') As poly,
       ST_GeomFromEWKT('MULTILINESTRING((175 155 2, 20 40 20, 50 60 -2, 125
100 1, 175 155 1),
       (1 10 2, 5 20 1))') As mline ) As foo;
       shl3d | shl2d
-----|-----
LINESTRING(39.993580415989 54.1889925532825 5,40.4078575708294 53.6052383805529
5.03423778139177) | LINESTRING(20 40,20 40)

```

ST_3DShortestLine

[ST_3DClosestPoint](#), [ST_3DDistance](#), [ST_LongestLine](#), [ST_ShortestLine](#), [ST_3DMaxDistance](#)

8.13 Overlay Functions

8.13.1 ST_ClipByBox2D

ST_ClipByBox2D — Computes the portion of a geometry falling within a rectangle.

Synopsis

geometry **ST_ClipByBox2D**(geometry geom, box2d box);

Description

Clips a geometry by a 2D box in a fast and tolerant but possibly invalid way. Topologically invalid input geometries do not result in exceptions being thrown. The output geometry is not guaranteed to be valid (in particular, self-intersections for a polygon may be introduced).

Performed by the GEOS module.

Availability: 2.2.0

Examples

```

-- Rely on implicit cast from geometry to box2d for the second parameter
SELECT ST_ClipByBox2D(geom, ST_MakeEnvelope(0,0,10,10)) FROM mytab;

```

See Also

[ST_Intersection](#), [ST_MakeBox2D](#), [ST_MakeEnvelope](#)

8.13.2 ST_Difference

ST_Difference — Computes a geometry representing the part of geometry A that does not intersect geometry B.


```

    ST_Difference(
        'LINESTRING(50 100, 50 200)::geometry',
        'LINESTRING(50 50, 50 150)::geometry'
    )
);

st_astext
-----
LINESTRING(50 150,50 200)

```

The difference of 3D points.

```

SELECT ST_AsEWKT( ST_Difference(
    'MULTIPOINT(-118.58 38.38 5,-118.60 38.329 6,-118.614 38.281 7)' :: geometry,
    'POINT(-118.614 38.281 5)' :: geometry
) );

st_asewkt
-----
MULTIPOINT(-118.6 38.329 6,-118.58 38.38 5)

```

See Also

[ST_SymDifference](#), [ST_Intersection](#), [ST_Union](#)

8.13.3 ST_Intersection

ST_Intersection — Computes a geometry representing the shared portion of geometries A and B.

Synopsis

```

geometry ST_Intersection( geometry geomA , geometry geomB , float8 gridSize = -1 );
geography ST_Intersection( geography geogA , geography geogB );

```

Description

Returns a geometry representing the point-set intersection of two geometries. In other words, that portion of geometry A and geometry B that is shared between the two geometries.

If the geometries have no points in common (i.e. are disjoint) then an empty atomic geometry of appropriate type is returned.

If the optional `gridSize` argument is provided, the inputs are snapped to a grid of the given size, and the result vertices are computed on that same grid. (Requires GEOS-3.9.0 or higher)

ST_Intersection in conjunction with [ST_Intersects](#) is useful for clipping geometries such as in bounding box, buffer, or region queries where you only require the portion of a geometry that is inside a country or region of interest.

Note



Geography: For geography this is really a thin wrapper around the geometry implementation. It first determines the best SRID that fits the bounding box of the 2 geography objects (if geography objects are within one half zone UTM but not same UTM will pick one of those) (favoring UTM or Lambert Azimuthal Equal Area (LAEA) north/south pole, and falling back on mercator in worst case scenario) and then intersection in that best fit planar spatial ref and retransforms back to WGS84 geography.

**Warning**

This function will drop the M coordinate values if present.

**Warning**

If working with 3D geometries, you may want to use SFGAL based [ST_3DIntersection](#) which does a proper 3D intersection for 3D geometries. Although this function works with Z-coordinate, it does an averaging of Z-Coordinate.

Performed by the GEOS module

Enhanced: 3.1.0 accept a gridSize parameter - requires GEOS >= 3.9.0

Changed: 3.0.0 does not depend on SFCGAL.

Availability: 1.5 support for geography data type was introduced.



This method implements the [OGC 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



This function supports 3d and will not drop the z-index. However, the result is computed using XY only. The result Z values are copied, averaged or interpolated.

Examples

```
SELECT ST_AsText(ST_Intersection('POINT(0 0)::geometry', 'LINESTRING ( 2 0, 0 2 ) '::
  geometry));
  st_astext
-----
GEOMETRYCOLLECTION EMPTY

SELECT ST_AsText(ST_Intersection('POINT(0 0)::geometry', 'LINESTRING ( 0 0, 0 2 ) '::
  geometry));
  st_astext
-----
POINT(0 0)
```

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.geom, trails.geom))).geom clipped_geom
    from country
    inner join trails on ST_Intersects(country.geom, trails.geom)
) as clipped
where ST_Dimension(clipped.clipped_geom) = 1;
```

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.geom, poly.geom),
           0.0
         )
       ) clipped_geom
from country
       inner join poly on ST_Intersects(country.geom, poly.geom)
where not ST_IsEmpty(ST_Buffer(ST_Intersection(country.geom, poly.geom), 0.0));
```

Examples: 2.5Dish

Note this is not a true intersection, compare to the same example using [ST_3DIntersection](#).

```
select ST_AsText(ST_Intersection(linestring, polygon)) As wkt
from   ST_GeomFromText('LINESTRING Z (2 2 6,1.5 1.5 7,1 1 8,0.5 0.5 8,0 0 10)') AS ↵
       linestring
CROSS JOIN ST_GeomFromText('POLYGON((0 0 8, 0 1 8, 1 1 8, 1 0 8, 0 0 8))') AS polygon;

           st_astext
-----
LINESTRING Z (1 1 8,0.5 0.5 8,0 0 10)
```

See Also

[ST_3DIntersection](#), [ST_Difference](#), [ST_Union](#), [ST_Dimension](#), [ST_Dump](#), [ST_Force2D](#), [ST_SymDifference](#), [ST_Intersects](#), [ST_Multi](#)

8.13.4 ST_MemUnion

ST_MemUnion — Aggregate function which unions geometries in a memory-efficient but slower way

Synopsis

geometry **ST_MemUnion**(geometry set geomfield);

Description

An aggregate function that unions the input geometries, merging them to produce a result geometry with no overlaps. The output may be a single geometry, a MultiGeometry, or a Geometry Collection.



Note

Produces the same result as [ST_Union](#), but uses less memory and more processor time. This aggregate function works by unioning the geometries incrementally, as opposed to the [ST_Union](#) aggregate which first accumulates an array and then unions the contents using a fast algorithm.



This function supports 3d and will not drop the z-index. However, the result is computed using XY only. The result Z values are copied, averaged or interpolated.

Examples

```
SELECT id,
       ST_MemUnion(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

See Also

[ST_Union](#)

8.13.5 ST_Node

ST_Node — Nodes a collection of lines.

Synopsis

geometry **ST_Node**(geometry geom);

Description

Returns a (Multi)LineString representing the fully noded version of a collection of linestrings. The noding preserves all of the input nodes, and introduces the least possible number of new nodes. The resulting linework is dissolved (duplicate lines are removed).

This is a good way to create fully-noded linework suitable for use as input to [ST_Polygonize](#).



This function supports 3d and will not drop the z-index.

Performed by the GEOS module.

Availability: 2.0.0

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.

Examples

Noding a 3D LineString which self-intersects

```
SELECT ST_AsText (
    ST_Node('LINESTRINGZ(0 0 0, 10 10 10, 0 10 5, 10 0 3)::geometry')
) As output;
output
-----
MULTILINESTRING Z ((0 0 0,5 5 4.5),(5 5 4.5,10 10 10,0 10 5,5 5 4.5),(5 5 4.5,10 0 3))
```

Noding two LineStrings which share common linework. Note that the result linework is dissolved.

```
SELECT ST_AsText (
    ST_Node('MULTILINESTRING ((2 5, 2 1, 7 1), (6 1, 4 1, 2 3, 2 5))::geometry')
) As output;
output
-----
MULTILINESTRING((2 5,2 3),(2 3,2 1,4 1),(4 1,2 3),(4 1,6 1),(6 1,7 1))
```

See Also

[ST_UnaryUnion](#)

8.13.6 ST_Split

ST_Split — Returns a collection of geometries created by splitting a geometry by another geometry.

Synopsis

```
geometry ST_Split(geometry input, geometry blade);
```

Description

The function supports splitting a `LineString` by a `(Multi)Point`, `(Multi)LineString` or `(Multi)Polygon` boundary, or a `(Multi)Polygon` by a `LineString`. When a `(Multi)Polygon` is used as the blade, its linear components (the boundary) are used for splitting the input. The result geometry is always a collection.

This function is in a sense the opposite of [ST_Union](#). Applying `ST_Union` to the returned collection should theoretically yield the original geometry (although due to numerical rounding this may not be exactly the case).



Note

If the the input and blade do not intersect due to numerical precision issues, the input may not be split as expected. To avoid this situation it may be necessary to snap the input to the blade first, using [ST_Snap](#) with a small tolerance.

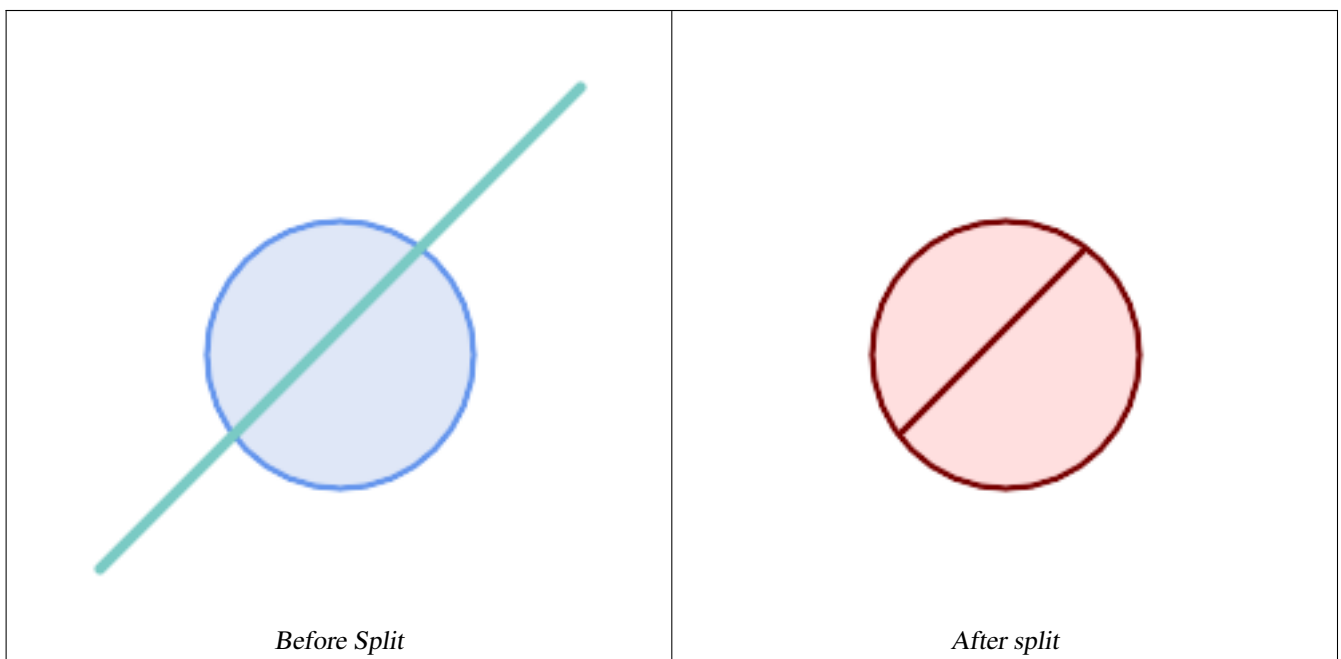
Availability: 2.0.0 requires GEOS

Enhanced: 2.2.0 support for splitting a line by a multiline, a multipoint or (multi)polygon boundary was introduced.

Enhanced: 2.5.0 support for splitting a polygon by a multiline was introduced.

Examples

Polygon split by a Line



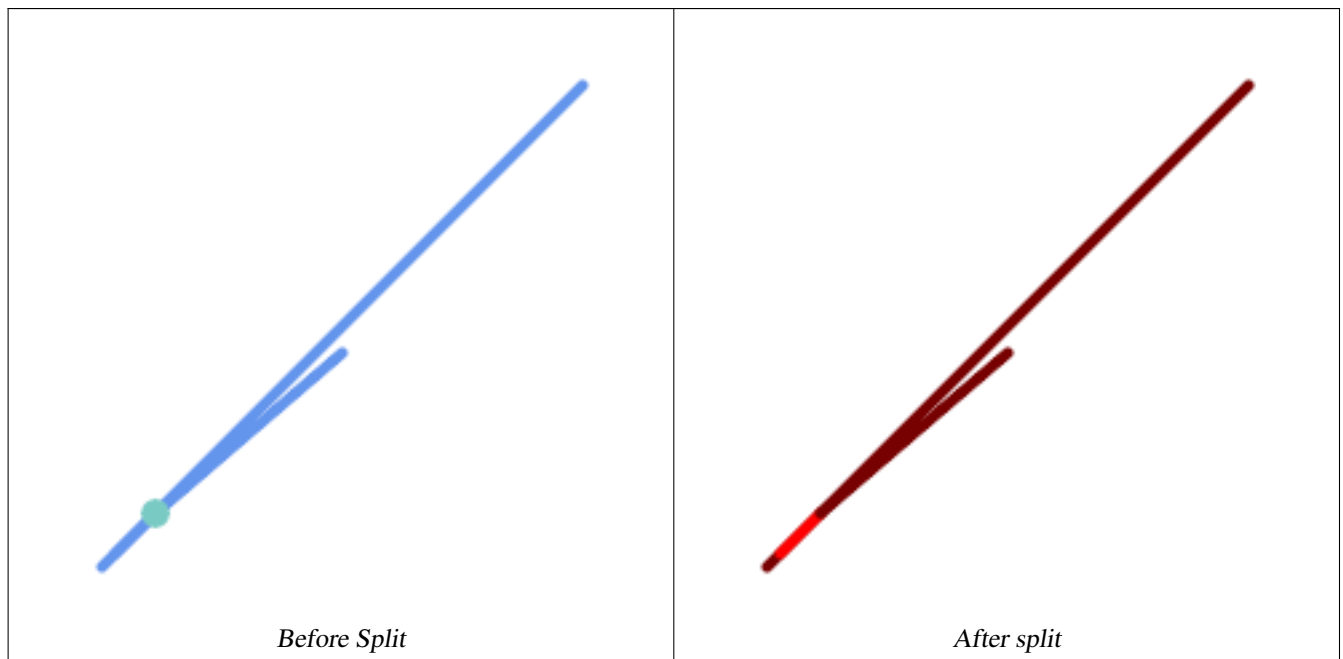

```

SELECT ST_AsText( ST_Split(
    ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50), -- circle
    ST_MakeLine(ST_Point(10, 10),ST_Point(190, 190)) -- line
));

-- result --
GEOMETRYCOLLECTION(
  POLYGON((150 90,149.039264020162 80.2454838991936,146.193976625564 70.8658283817455,..),
    POLYGON(..))
)

```

MultiLineString split by a Point, where the point lies exactly on both LineStrings.



```

SELECT ST_AsText(ST_Split(
    'MULTILINESTRING((10 10, 190 191), (15 15, 30 30, 100 90))',
    ST_Point(30,30))) As split;

split
-----
GEOMETRYCOLLECTION(
  LINESTRING(10 10,30 30),
  LINESTRING(30 30,190 190),
  LINESTRING(15 15,30 30),
  LINESTRING(30 30,100 90)
)

```

LineString split by a Point, where the point does not lie exactly on the line. Shows using **ST_Snap** to snap the line to the point to allow it to be split.

```

WITH data AS (SELECT
  'LINESTRING(0 0, 100 100)::geometry AS line,
  'POINT(51 50):: geometry AS point
)
SELECT ST_AsText( ST_Split(line, point)) AS no_split,
       ST_AsText( ST_Split( ST_Snap(line, point, 1), point)) AS split

```

```
FROM data;

no_split | split
-----+-----
GEOMETRYCOLLECTION(LINESTRING(0 0,100 100)) | GEOMETRYCOLLECTION(LINESTRING(0 0,51 50), ↵
LINESTRING(51 50,100 100))
```

See Also

[ST_Snap](#), [ST_Union](#)

8.13.7 ST_Subdivide

ST_Subdivide — Computes a rectilinear subdivision of a geometry.

Synopsis

setof geometry **ST_Subdivide**(geometry geom, integer max_vertices=256, float8 gridSize = -1);

Description

Returns a set of geometries that are the result of dividing `geom` into parts using rectilinear lines, with each part containing no more than `max_vertices`.

`max_vertices` must be 5 or more, as 5 points are needed to represent a closed box. `gridSize` can be specified to have clipping work in fixed-precision space (requires GEOS-3.9.0+).

Point-in-polygon and other spatial operations are normally faster for indexed subdivided datasets. Since the bounding boxes for the parts usually cover a smaller area than the original geometry bbox, index queries produce fewer "hit" cases. The "hit" cases are faster because the spatial operations executed by the index recheck process fewer points.

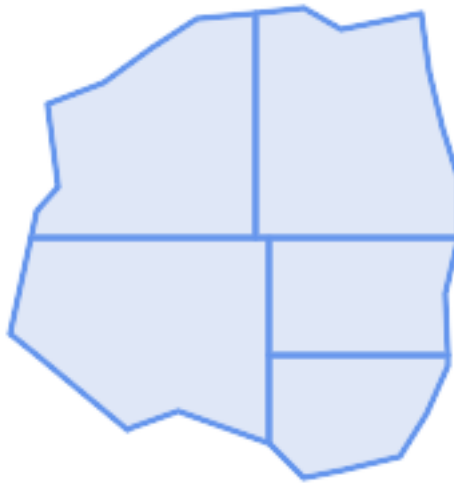


Note
This is a **set-returning function** (SRF) that return a set of rows containing single geometry values. It can be used in a SELECT list or a FROM clause to produce a result set with one record for each result geometry.

- Performed by the GEOS module.
- Availability: 2.2.0
- Enhanced: 2.5.0 reuses existing points on polygon split, vertex count is lowered from 8 to 5.
- Enhanced: 3.1.0 accept a gridSize parameter, requires GEOS >= 3.9.0 to use this new feature.

Examples

Example: Subdivide a polygon into parts with no more than 10 vertices, and assign each part a unique id.



Subdivided to maximum 10 vertices

```
SELECT row_number() OVER() As rn, ST_AsText(geom) As wkt
FROM (SELECT ST_SubDivide(
  'POLYGON((132 10,119 23,85 35,68 29,66 28,49 42,32 56,22 64,32 110,40 119,36 150,
  57 158,75 171,92 182,114 184,132 186,146 178,176 184,179 162,184 141,190 122,
  190 100,185 79,186 56,186 52,178 34,168 18,147 13,132 10))'::geometry,10)) AS f( ↵
  geom);
```

rn	wkt
1	POLYGON((119 23,85 35,68 29,66 28,32 56,22 64,29.8260869565217 100,119 100,119 ↵ 23))
2	POLYGON((132 10,119 23,119 56,186 56,186 52,178 34,168 18,147 13,132 10))
3	POLYGON((119 56,119 100,190 100,185 79,186 56,119 56))
4	POLYGON((29.8260869565217 100,32 110,40 119,36 150,57 158,75 171,92 182,114 ↵ 184,114 100,29.8260869565217 100))
5	POLYGON((114 184,132 186,146 178,176 184,179 162,184 141,190 122,190 100,114 ↵ 100,114 184))

Example: Densify a long geography line using `ST_Segmentize(geography, distance)`, and use `ST_Subdivide` to split the resulting line into sublines of 8 vertices.



The densified and split lines.

```
SELECT ST_AsText( ST_Subdivide(
    ST_Segmentize('LINESTRING(0 0, 85 85)::geography,
    1200000)::geometry, 8));
```

```
LINESTRING(0 0,0.487578359029357 5.57659056746196,0.984542144675897 ↵
    11.1527721155093,1.50101059639722 16.7281035483571,1.94532113630331 21.25)
LINESTRING(1.94532113630331 21.25,2.04869538062779 22.3020741387339,2.64204641967673 ↵
    27.8740533545155,3.29994062412787 33.443216802941,4.04836719489742 ↵
    39.0084282520239,4.59890468420694 42.5)
LINESTRING(4.59890468420694 42.5,4.92498503922732 44.5680389206321,5.98737409390639 ↵
    50.1195229244701,7.3290919767674 55.6587646879025,8.79638749938413 60.1969505994924)
LINESTRING(8.79638749938413 60.1969505994924,9.11375579533779 ↵
    61.1785363177625,11.6558166691368 66.6648504160202,15.642041247655 ↵
    72.0867690601745,22.8716627200212 77.3609628116894,24.6991785131552 77.8939011989848)
LINESTRING(24.6991785131552 77.8939011989848,39.4046096622744 ↵
    82.1822848017636,44.7994523421035 82.5156766227011)
LINESTRING(44.7994523421035 82.5156766227011,85 85)
```

Example: Subdivide the complex geometries of a table in-place. The original geometry records are deleted from the source table, and new records for each subdivided result geometry are inserted.

```
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;
```

Example: Create a new table containing subdivided geometries, retaining the key of the original geometry so that the new table can be joined to the source table. Since ST_Subdivide is a set-returning (table) function that returns a set of single-value rows, this syntax automatically produces a table with one row for each result part.

```
CREATE TABLE subdivided_geoms AS
SELECT pkey, ST_Subdivide(geom) AS geom
FROM original_geoms;
```

See Also

[ST_ClipByBox2D](#), [ST_Segmentize](#), [ST_Split](#), [ST_NPoints](#)

8.13.8 ST_SymDifference

ST_SymDifference — Computes a geometry representing the portions of geometries A and B that do not intersect.

Synopsis

geometry **ST_SymDifference**(geometry geomA, geometry geomB, float8 gridSize = -1);

Description

Returns a geometry representing the portions of geometries A and B that do not intersect. This is equivalent to ST_Union(A, B) - ST_Intersection(A, B). It is called a symmetric difference because ST_SymDifference(A, B) = ST_SymDifference(B, A).

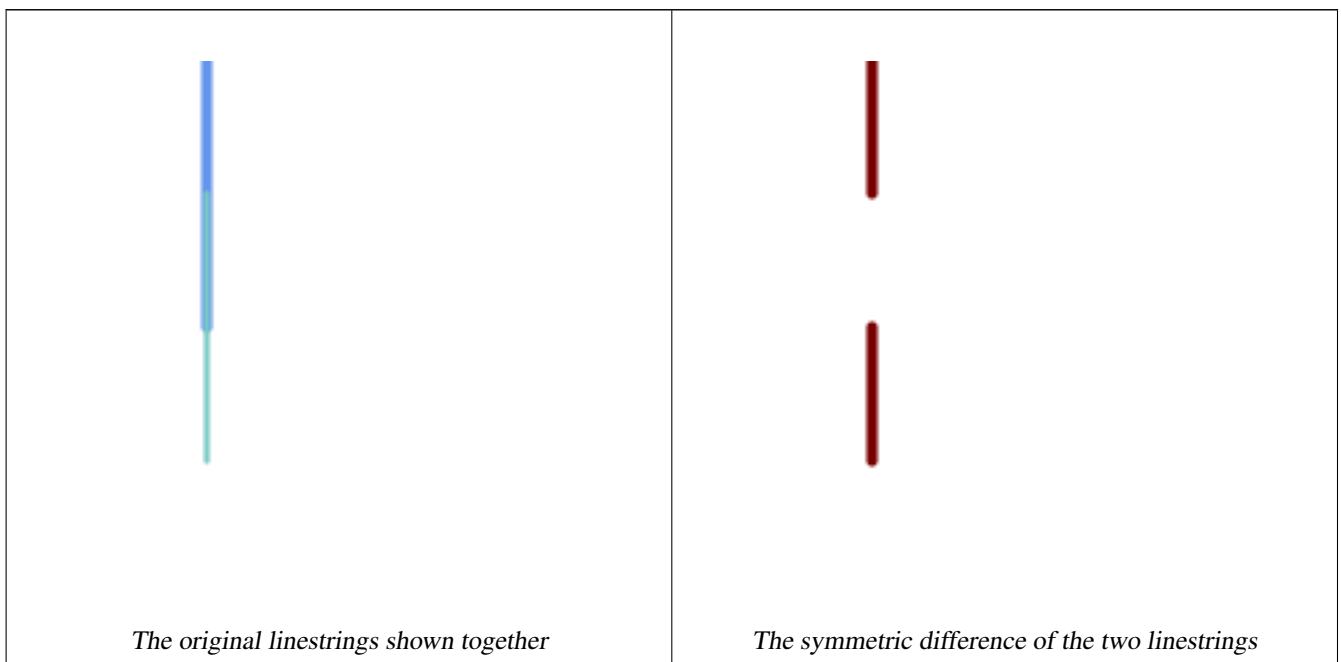
If the optional `gridSize` argument is provided, the inputs are snapped to a grid of the given size, and the result vertices are computed on that same grid. (Requires GEOS-3.9.0 or higher)

Performed by the GEOS module

Enhanced: 3.1.0 accept a `gridSize` parameter - requires GEOS $\geq$ 3.9.0

- ✔ This method implements the [OGC 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. However, the result is computed using XY only. The result Z values are copied, averaged or interpolated.

Examples



```
--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))
```

See Also

[ST_Difference](#), [ST_Intersection](#), [ST_Union](#)

8.13.9 ST_UnaryUnion

ST_UnaryUnion — Computes the union of the components of a single geometry.

Synopsis

```
geometry ST_UnaryUnion(geometry geom, float8 gridSize = -1);
```

Description

A single-input variant of [ST_Union](#). The input may be a single geometry, a MultiGeometry, or a GeometryCollection. The union is applied to the individual elements of the input.

This function can be used to fix MultiPolygons which are invalid due to overlapping components. However, the input components must each be valid. An invalid input component such as a bow-tie polygon may cause an error. For this reason it may be better to use [ST_MakeValid](#).

Another use of this function is to node and dissolve a collection of linestrings which cross or overlap to make them [simple](#). (To add nodes but not dissolve duplicate linework use [ST_Node](#).)

It is possible to combine **ST_UnaryUnion** with [ST_GeomCollFromText](#) to fine-tune how many geometries are be unioned at once. This allows trading off between memory usage and compute time, striking a balance between **ST_Union** and [ST_MemUnion](#).

If the optional `gridSize` argument is provided, the inputs are snapped to a grid of the given size, and the result vertices are computed on that same grid. (Requires GEOS-3.9.0 or higher)



This function supports 3d and will not drop the z-index. However, the result is computed using XY only. The result Z values are copied, averaged or interpolated.

Enhanced: 3.1.0 accept a `gridSize` parameter - requires GEOS >= 3.9.0

Availability: 2.0.0

See Also

[ST_Union](#), [ST_MemUnion](#), [ST_MakeValid](#), [ST_GeomCollFromText](#), [ST_Node](#)

8.13.10 ST_Union

ST_Union — Computes a geometry representing the point-set union of the input geometries.

Synopsis

```
geometry ST_Union(geometry g1, geometry g2);
geometry ST_Union(geometry g1, geometry g2, float8 gridSize);
geometry ST_Union(geometry[] g1_array);
geometry ST_Union(geometry set g1field);
geometry ST_Union(geometry set g1field, float8 gridSize);
```

Description

Unions the input geometries, merging geometry to produce a result geometry with no overlaps. The output may be an atomic geometry, a MultiGeometry, or a Geometry Collection. Comes in several variants:

Two-input variant: returns a geometry that is the union of two input geometries. If either input is NULL, then NULL is returned.

Array variant: returns a geometry that is the union of an array of geometries.

Aggregate variant: returns a geometry that is the union of a rowset of geometries. The ST_Union() function is an "aggregate" function in the terminology of PostgreSQL. That means that it operates on rows of data, in the same way the SUM() and AVG() functions do and like most aggregates, it also ignores NULL geometries.

See [ST_UnaryUnion](#) for a non-aggregate, single-input variant.

The ST_Union array and set variants use the fast Cascaded Union algorithm described in <http://blog.cleverelephant.ca/2009/01/-must-faster-unions-in-postgis-14.html>

A gridSize can be specified to work in fixed-precision space. The inputs are snapped to a grid of the given size, and the result vertices are computed on that same grid. (Requires GEOS-3.9.0 or higher)



Note

[ST_GeomCollFromText](#) may sometimes be used in place of ST_Union, if the result is not required to be non-overlapping. ST_Collect is usually faster than ST_Union because it performs no processing on the collected geometries.

Performed by the GEOS module.

ST_Union creates MultiLineString and does not sew LineStrings into a single LineString. Use [ST_LineMerge](#) to sew LineStrings.

NOTE: this function was formerly called GeomUnion(), which was renamed from "Union" because UNION is an SQL reserved word.

Enhanced: 3.1.0 accept a gridSize parameter - requires GEOS >= 3.9.0

Changed: 3.0.0 does not depend on SFCGAL.

Availability: 1.4.0 - ST_Union was enhanced. ST_Union(geomarray) was introduced and also faster aggregate collection in PostgreSQL.



This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3



Note

Aggregate version is not explicitly defined in OGC SPEC.



This method implements the SQL/MM specification. SQL-MM 3: 5.1.19 the z-index (elevation) when polygons are involved.



This function supports 3d and will not drop the z-index. However, the result is computed using XY only. The result Z values are copied, averaged or interpolated.

Examples

Aggregate example

```
SELECT id,
       ST_Union(geom) as singlegeom
FROM sometable f
GROUP BY id;
```

Non-Aggregate example

```
select ST_AsText(ST_Union('POINT(1 2)' :: geometry, 'POINT(-2 3)' :: geometry))

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

select ST_AsText(ST_Union('POINT(1 2)' :: geometry, 'POINT(1 2)' :: geometry))

st_astext
-----
POINT(1 2)
```

3D example - sort of supports 3D (and with mixed dimensions!)

```
select ST_AsEWKT(ST_Union(geom))
from (
    select 'POLYGON((-7 4.2,-7.1 4.2,-7.1 4.3, -7 4.2))'::geometry geom
    union all
    select 'POINT(5 5 5)'::geometry geom
    union all
    select 'POINT(-2 3 1)'::geometry geom
    union all
    select 'LINESTRING(5 5 5, 10 10 10)'::geometry geom
) as foo;

st_asewkt
-----
GEOMETRYCOLLECTION(POINT(-2 3 1),LINESTRING(5 5 5,10 10 10),POLYGON((-7 4.2 5,-7.1 4.2 5,-7.1 4.3 5,-7 4.2 5)));
```

3d example not mixing dimensions

```
select ST_AsEWKT(ST_Union(geom))
from (
    select 'POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2, -7 4.2 2))'::geometry geom
    union all
    select 'POINT(5 5 5)'::geometry geom
    union all
    select 'POINT(-2 3 1)'::geometry geom
    union all
    select 'LINESTRING(5 5 5, 10 10 10)'::geometry geom
) as foo;

st_asewkt
-----
GEOMETRYCOLLECTION(POINT(-2 3 1),LINESTRING(5 5 5,10 10 10),POLYGON((-7 4.2 2,-7.1 4.2 3,-7.1 4.3 2,-7 4.2 2)))

--Examples using new Array construct
SELECT ST_Union(ARRAY(SELECT geom FROM sometable));

SELECT ST_AsText(ST_Union(ARRAY[ST_GeomFromText('LINESTRING(1 2, 3 4)'),
    ST_GeomFromText('LINESTRING(3 4, 4 5)')])) As wktunion;

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


See Also

[ST_GeomCollFromText](#), [ST_UnaryUnion](#), [ST_MemUnion](#), [ST_Intersection](#), [ST_Difference](#), [ST_SymDifference](#)

8.14 **ST_Buffer**

8.14.1 **ST_Buffer**

ST_Buffer — Computes a geometry covering all points within a given distance from a geometry.

Synopsis

```
geometry ST_Buffer(geometry g1, float radius_of_buffer, text buffer_style_parameters = '');
geometry ST_Buffer(geometry g1, float radius_of_buffer, integer num_seg_quarter_circle);
geography ST_Buffer(geography g1, float radius_of_buffer, text buffer_style_parameters);
geography ST_Buffer(geography g1, float radius_of_buffer, integer num_seg_quarter_circle);
```

Description

Computes a POLYGON or MULTIPOLYGON that represents all points whose distance from a geometry/geography is less than or equal to a given distance. A negative distance shrinks the geometry rather than expanding it. A negative distance may shrink a polygon completely, in which case POLYGON EMPTY is returned. For points and lines negative distances always return empty results.

For geometry, the distance is specified in the units of the Spatial Reference System of the geometry. For geography, the distance is specified in meters.

The optional third parameter controls the buffer accuracy and style. The accuracy of circular arcs in the buffer is specified as the number of line segments used to approximate a quarter circle (default is 8). The buffer style can be specified by providing a list of blank-separated key=value pairs as follows:

- `'quad_segs=#'` : number of line segments used to approximate a quarter circle (default is 8).
- `'endcap=round|flat|square'` : endcap style (defaults to "round"). 'butt' is accepted as a synonym for 'flat'.
- `'join=round|mitre|bevel'` : join style (defaults to "round"). 'miter' is accepted as a synonym for 'mitre'.
- `'mitre_limit=#.#'` : mitre ratio limit (only affects mitered join style). 'miter_limit' is accepted as a synonym for '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 applicable to LINESTRING geometry and does not affect POINT or POLYGON geometries. By default end caps are square.

Note

For geography, this is a wrapper around the geometry implementation. It determines a planar spatial reference system that best fits the bounding box of the geography object (trying UTM, Lambert Azimuthal Equal Area (LAEA) North/South pole, and finally Mercator). The buffer is computed in the planar space, and then transformed back to WGS84. This may not produce the desired behavior if the input object is much larger than a UTM zone or crosses the dateline

Note

Buffer output is always a valid polygonal geometry. Buffer can handle invalid inputs, so buffering by distance 0 is sometimes used as a way of repairing invalid polygons. [ST_MakeValid](#) can also be used for this purpose.

**Note**

Buffering is sometimes used to perform a within-distance search. For this use case it is more efficient to use `ST_DWithin`.

**Note**

This function ignores the Z dimension. It always gives a 2D result even when used on a 3D geometry.

Enhanced: 2.5.0 - `ST_Buffer` geometry support was enhanced to allow for side buffering specification `side=both|left|right`.

Availability: 1.5 - `ST_Buffer` was enhanced to support different endcaps and join types. These are useful for example to convert road linestrings into polygon roads with flat or square edges instead of rounded edges. Thin wrapper for geography was added.

GEOS [#baa8;#b4c8;#b85c; #c2e4;#d589](#);

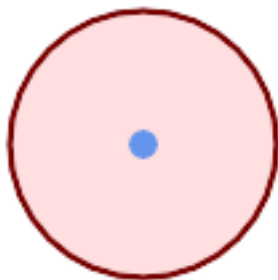


This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3



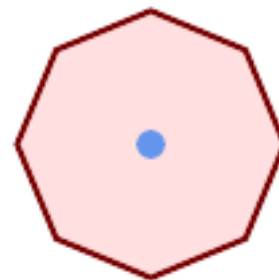
This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.30

Examples



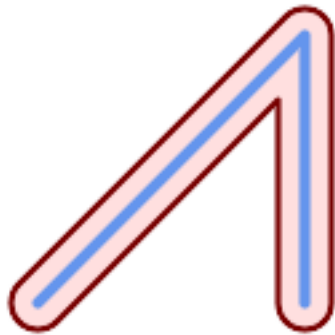
`quad_segs=8` ([#ae30;#bcf8;#ac12](#);))

```
SELECT ST_Buffer(
  ST_GeomFromText('POINT(100 90)'),
  50, 'quad_segs=8');
```



`quad_segs=2` ([#800;#c9c8](#);))

```
SELECT ST_Buffer(
  ST_GeomFromText('POINT(100 90)'),
  50, 'quad_segs=2');
```



endcap=round join=round (*기 ை 값*)

```
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');
```



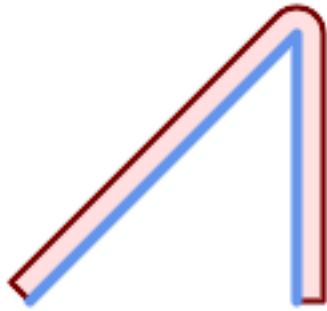
join=bevel

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=bevel');
```



join=mitre mitre_limit=5.0 (*마 이 터 제 한 기 ை 값*)

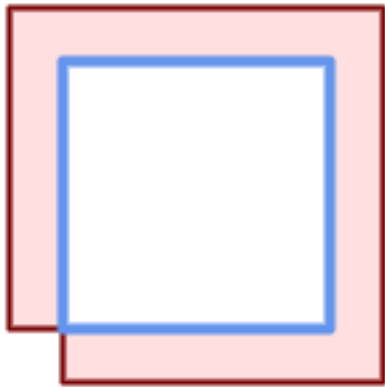
```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'join=mitre mitre_limit=5.0');
```

*side=left*

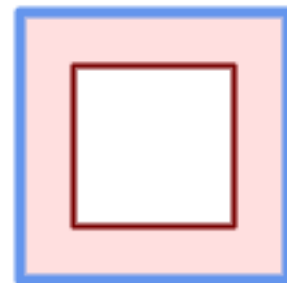
```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'side=left');
```

*side=right*

```
SELECT ST_Buffer(
  ST_GeomFromText(
    'LINESTRING(50 50,150 150,150 50)'
  ), 10, 'side=right');
```

*right-hand-winding, polygon boundary side=left*

```
SELECT ST_Buffer(
  ST_ForceRHR(
    ST_Boundary(
      ST_GeomFromText(
        'POLYGON ((50 50, 50 150, 150 150, 150 50, 50 50))'
      ), 20, 'side=left');
```

*right-hand-winding, polygon boundary side=right*

```
SELECT ST_Buffer(
  ST_ForceRHR(
    ST_Boundary(
      ST_GeomFromText(
        'POLYGON ((50 50, 50 150, 150 150, 150 50, 50 50))'
      ), 20, 'side=right');
```

```
--A buffered point approximates a circle
-- A buffered point forcing approximation of (see diagram)
```

```
-- 2 points per quarter circle is poly with 8 sides (see diagram)
SELECT ST_NPoints(ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50)) As ←
    promisingcircle_pcount,
ST_NPoints(ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50, 2)) As lamecircle_pcount;

promisingcircle_pcount | lamecircle_pcount
-----+-----
33 | 9

--A lighter but lamer circle
-- only 2 points per quarter circle is an octagon
--Below is a 100 meter octagon
-- Note coordinates are in NAD 83 long lat which we transform
to Mass state plane meter and then buffer to get measurements in meters;
SELECT ST_AsText(ST_Buffer(
ST_Transform(
ST_SetSRID(ST_Point(-71.063526, 42.35785), 4269), 26986)
,100,2)) As octagon;
-----
POLYGON((236057.59057465 900908.759918696,236028.301252769 900838.049240578,235
957.59057465 900808.759918696,235886.879896532 900838.049240578,235857.59057465
900908.759918696,235886.879896532 900979.470596815,235957.59057465 901008.759918
696,236028.301252769 900979.470596815,236057.59057465 900908.759918696))
```

ST_GeomCollFromText

ST_DWithin, **ST_SetSRID**, **ST_Transform**, **ST_Union**, **ST_MakeValid**

8.14.2 ST_BuildArea

ST_BuildArea — Creates a polygonal geometry formed by the linework of a geometry.

Synopsis

geometry **ST_BuildArea**(geometry geom);

ST_BuildArea

Creates an areal geometry formed by the constituent linework of the input geometry. The input can be LINESTRINGS, MULTILINESTRINGS, POLYGONS, MULTIPOLYGONS, and GeometryCollections. The result is a Polygon or MultiPolygon, depending on input. If the input linework does not form polygons, NULL is returned.

This function assumes all inner geometries represent holes



Note

Creates an areal geometry formed by the constituent linework of the input geometry. The input can be LINESTRINGS, MULTILINESTRINGS, POLYGONS, MULTIPOLYGONS, and GeometryCollections. The result is a Polygon or MultiPolygon, depending on input. If the input linework does not form polygons, NULL is returned.

1.1.0

ST_BuildArea



These will create a donut

```
--using polygons
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;

--using linestrings
SELECT ST_BuildArea(ST_Collect(smallc,bigc))
FROM (SELECT
  ST_ExteriorRing(ST_Buffer(
    ST_GeomFromText('POINT(100 90)'), 25)) As smallc,
  ST_ExteriorRing(ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50)) As bigc) As foo;
```

ST_Union

ST_Node, **ST_MakePolygon**, **ST_MakeValid**, **ST_BdPolyFromText**, **ST_BdMPolyFromText** (wrappers to this function with standard OGC interface)

8.14.3 ST_Centroid

ST_Centroid — Returns the centroid of a geometry.

Synopsis

```
geometry ST_Centroid(geometry g1);
geography ST_Centroid(geography g1, boolean use_spheroid=true);
```

ST_Centroid

Computes a point which is the geometric center of mass of a geometry. For [MULTI]POINTS, the centroid is the arithmetic mean of the input coordinates. For [MULTI]LINESTRINGS, the centroid is computed using the weighted length of each line segment. For [MULTI]POLYGONS, the centroid is computed in terms of area. If an empty geometry is supplied, an empty

GEOMETRYCOLLECTION is returned. If NULL is supplied, NULL is returned. If CIRCULARSTRING or COMPOUNDCURVE are supplied, they are converted to linestring with CurveToLine first, then same than for LINESTRING

For mixed-dimension input, the result is equal to the centroid of the component Geometries of highest dimension (since the lower-dimension geometries contribute zero "weight" to the centroid).

Note that for polygonal geometries the centroid does not necessarily lie in the interior of the polygon. For example, see the diagram below of the centroid of a C-shaped polygon. To construct a point guaranteed to lie in the interior of a polygon use [ST_PointOnSurface](#).

New in 2.3.0 : supports CIRCULARSTRING and COMPOUNDCURVE (using CurveToLine)

Availability: 2.4.0 support for geography was introduced.



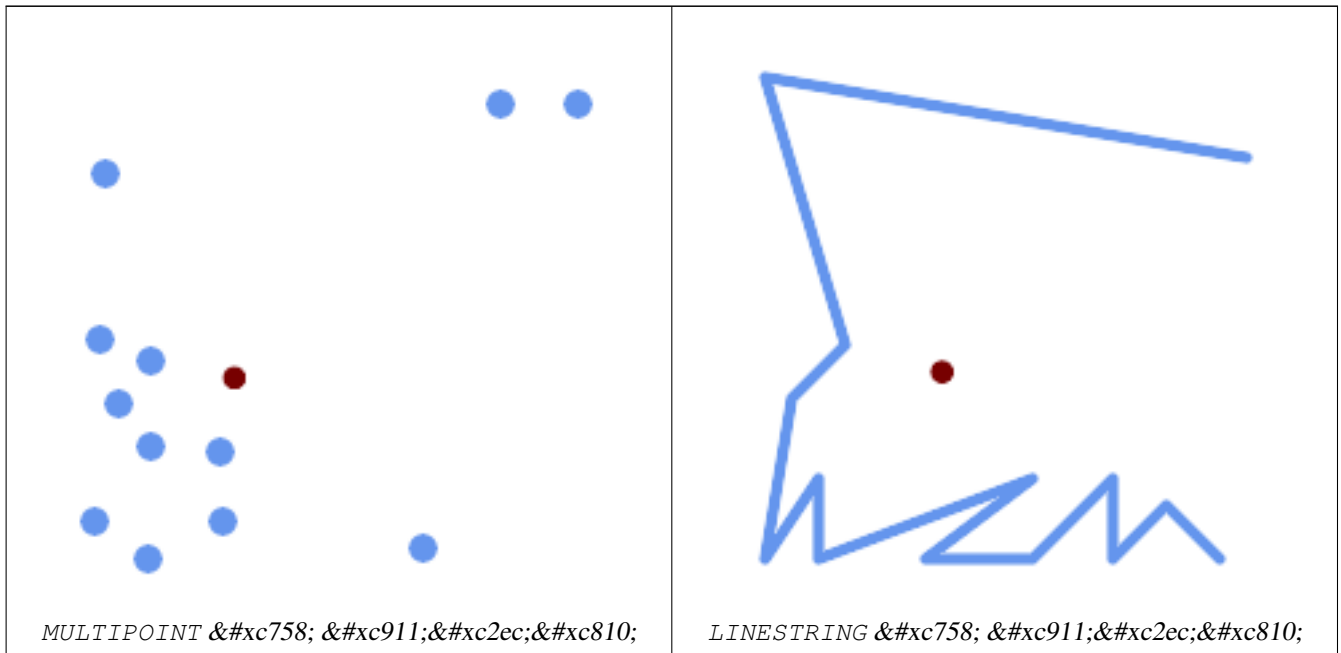
This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#).

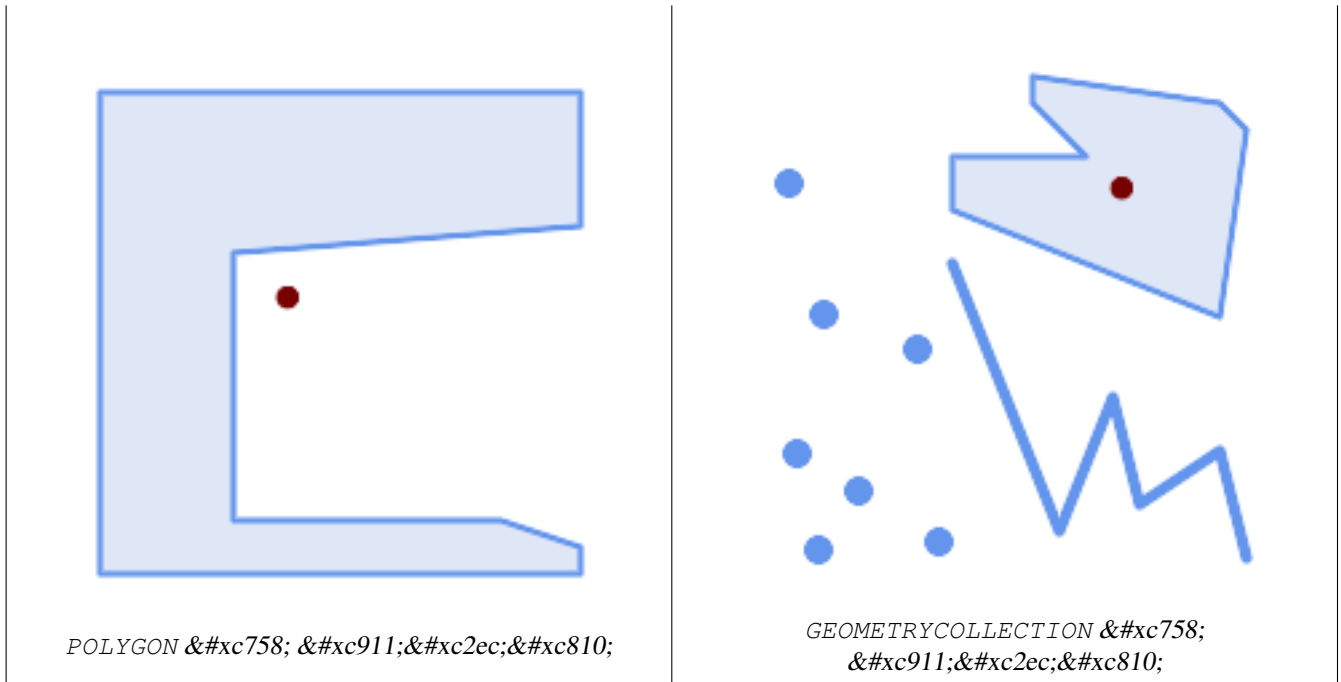


This method implements the SQL/MM specification. SQL-MM 3: 8.1.4, 9.5.5

Centroid

In the following illustrations the red dot is the centroid of the source geometry.





```
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 )'));
```

```
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)
```

ST_Centroid

ST_PointOnSurface, **ST_GeometricMedian**

8.14.4 ST_ChaikinSmoothing

ST_ChaikinSmoothing — Returns a smoothed version of a geometry, using the Chaikin algorithm

Synopsis

geometry **ST_ChaikinSmoothing**(geometry geom, integer nIterations = 1, boolean preserveEndpoints = false);

ST_Smooth

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

`ST_Smooth(geometry geom, integer iterations, integer simplify_tolerance)`



Note

Note that returned geometry will get more points than the original. To reduce the number of points again use one of the simplification functions on the result. (see [ST_Simplify](#) and [ST_SimplifyVW](#))

Availability: 2.5.0

ST_SmoothPolygon

A triangle is smoothed

```
select ST_AsText(ST_ChaikinSmoothing(geom)) smoothed
FROM (SELECT 'POLYGON((0 0, 8 8, 0 16, 0 0))'::geometry geom) As foo;
-- smoothed
-- POLYGON((0 0, 2 2, 6 6, 10 10, 14 14, 16 16, 14 14, 10 10, 6 6, 2 2, 0 0))
```

ST_Simplify

[ST_Simplify](#), [ST_SimplifyVW](#)

8.14.5 ST_ConcaveHull

`ST_ConcaveHull` — Computes a possibly concave geometry that encloses all input geometry vertices

Synopsis

geometry `ST_ConcaveHull`(geometry param_geom, float param_pctconvex, boolean param_allow_holes = false);

ST_ConcaveHull

A concave hull of a geometry is a possibly concave geometry that encloses the vertices of the input geometry. In the general case the concave hull is a Polygon. The polygon will not contain holes unless the optional `param_allow_holes` argument is specified as true. The concave hull of two or more collinear points is a two-point LineString. The concave hull of one or more identical points is a Point.

One can think of a concave hull as "shrink-wrapping" a set of points. This is different to the [convex hull](#), which is more like wrapping a rubber band around the points. The concave hull generally has a smaller area and represents a more natural boundary

for the input points. Like the convex hull, the vertices of a concave hull are a subset of the input points, and all other input points are contained within it.

The `param_pctconvex` controls the concaveness of the computed hull. A value of 1 produces the convex hull. A value of 0 produces a hull of maximum concaveness (but still a single polygon). Values between 1 and 0 produce hulls of increasing concaveness. Choosing a suitable value depends on the nature of the input data, but often values between 0.3 and 0.1 produce reasonable results.

Technically, the `param_pctconvex` determines a length as a fraction of the difference between the longest and shortest edges in the Delaunay Triangulation of the input points. Edges longer than this length are "eroded" from the triangulation. The triangles remaining form the concave hull.

For point and linear inputs, the hull will enclose all the points of the inputs. For polygonal inputs, the hull will enclose all the points of the input *and also* all the areas covered by the input. If you want a point-wise hull of a polygonal input, convert it to points first, using [ST_Points](#).

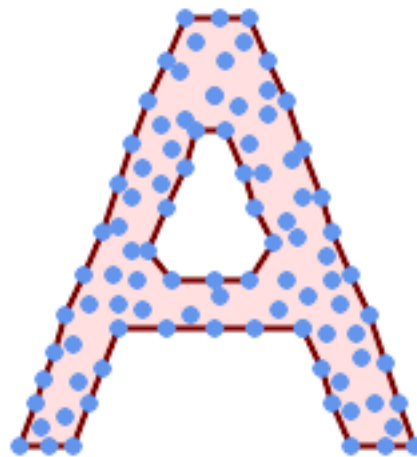
This is not an aggregate function. To compute the concave hull of a set of geometries use [ST_GeomCollFromText](#) (e.g. `ST_ConcaveHull( ST_Collect( geom ), 0.80)`).

2.0.0 – 2.0.10, 2.0.11, 2.0.12, 2.0.13, 2.0.14, 2.0.15, 2.0.16, 2.0.17, 2.0.18, 2.0.19, 2.0.20, 2.0.21, 2.0.22, 2.0.23, 2.0.24, 2.0.25, 2.0.26, 2.0.27, 2.0.28, 2.0.29, 2.0.30, 2.0.31, 2.0.32, 2.0.33, 2.0.34, 2.0.35, 2.0.36, 2.0.37, 2.0.38, 2.0.39, 2.0.40, 2.0.41, 2.0.42, 2.0.43, 2.0.44, 2.0.45, 2.0.46, 2.0.47, 2.0.48, 2.0.49, 2.0.50, 2.0.51, 2.0.52, 2.0.53, 2.0.54, 2.0.55, 2.0.56, 2.0.57, 2.0.58, 2.0.59, 2.0.60, 2.0.61, 2.0.62, 2.0.63, 2.0.64, 2.0.65, 2.0.66, 2.0.67, 2.0.68, 2.0.69, 2.0.70, 2.0.71, 2.0.72, 2.0.73, 2.0.74, 2.0.75, 2.0.76, 2.0.77, 2.0.78, 2.0.79, 2.0.80, 2.0.81, 2.0.82, 2.0.83, 2.0.84, 2.0.85, 2.0.86, 2.0.87, 2.0.88, 2.0.89, 2.0.90, 2.0.91, 2.0.92, 2.0.93, 2.0.94, 2.0.95, 2.0.96, 2.0.97, 2.0.98, 2.0.99, 2.0.100, 2.0.101, 2.0.102, 2.0.103, 2.0.104, 2.0.105, 2.0.106, 2.0.107, 2.0.108, 2.0.109, 2.0.110, 2.0.111, 2.0.112, 2.0.113, 2.0.114, 2.0.115, 2.0.116, 2.0.117, 2.0.118, 2.0.119, 2.0.120, 2.0.121, 2.0.122, 2.0.123, 2.0.124, 2.0.125, 2.0.126, 2.0.127, 2.0.128, 2.0.129, 2.0.130, 2.0.131, 2.0.132, 2.0.133, 2.0.134, 2.0.135, 2.0.136, 2.0.137, 2.0.138, 2.0.139, 2.0.140, 2.0.141, 2.0.142, 2.0.143, 2.0.144, 2.0.145, 2.0.146, 2.0.147, 2.0.148, 2.0.149, 2.0.150, 2.0.151, 2.0.152, 2.0.153, 2.0.154, 2.0.155, 2.0.156, 2.0.157, 2.0.158, 2.0.159, 2.0.160, 2.0.161, 2.0.162, 2.0.163, 2.0.164, 2.0.165, 2.0.166, 2.0.167, 2.0.168, 2.0.169, 2.0.170, 2.0.171, 2.0.172, 2.0.173, 2.0.174, 2.0.175, 2.0.176, 2.0.177, 2.0.178, 2.0.179, 2.0.180, 2.0.181, 2.0.182, 2.0.183, 2.0.184, 2.0.185, 2.0.186, 2.0.187, 2.0.188, 2.0.189, 2.0.190, 2.0.191, 2.0.192, 2.0.193, 2.0.194, 2.0.195, 2.0.196, 2.0.197, 2.0.198, 2.0.199, 2.0.200, 2.0.201, 2.0.202, 2.0.203, 2.0.204, 2.0.205, 2.0.206, 2.0.207, 2.0.208, 2.0.209, 2.0.210, 2.0.211, 2.0.212, 2.0.213, 2.0.214, 2.0.215, 2.0.216, 2.0.217, 2.0.218, 2.0.219, 2.0.220, 2.0.221, 2.0.222, 2.0.223, 2.0.224, 2.0.225, 2.0.226, 2.0.227, 2.0.228, 2.0.229, 2.0.230, 2.0.231, 2.0.232, 2.0.233, 2.0.234, 2.0.235, 2.0.236, 2.0.237, 2.0.238, 2.0.239, 2.0.240, 2.0.241, 2.0.242, 2.0.243, 2.0.244, 2.0.245, 2.0.246, 2.0.247, 2.0.248, 2.0.249, 2.0.250, 2.0.251, 2.0.252, 2.0.253, 2.0.254, 2.0.255, 2.0.256, 2.0.257, 2.0.258, 2.0.259, 2.0.260, 2.0.261, 2.0.262, 2.0.263, 2.0.264, 2.0.265, 2.0.266, 2.0.267, 2.0.268, 2.0.269, 2.0.270, 2.0.271, 2.0.272, 2.0.273, 2.0.274, 2.0.275, 2.0.276, 2.0.277, 2.0.278, 2.0.279, 2.0.280, 2.0.281, 2.0.282, 2.0.283, 2.0.284, 2.0.285, 2.0.286, 2.0.287, 2.0.288, 2.0.289, 2.0.290, 2.0.291, 2.0.292, 2.0.293, 2.0.294, 2.0.295, 2.0.296, 2.0.297, 2.0.298, 2.0.299, 2.0.300, 2.0.301, 2.0.302, 2.0.303, 2.0.304, 2.0.305, 2.0.306, 2.0.307, 2.0.308, 2.0.309, 2.0.310, 2.0.311, 2.0.312, 2.0.313, 2.0.314, 2.0.315, 2.0.316, 2.0.317, 2.0.318, 2.0.319, 2.0.320, 2.0.321, 2.0.322, 2.0.323, 2.0.324, 2.0.325, 2.0.326, 2.0.327, 2.0.328, 2.0.329, 2.0.330, 2.0.331, 2.0.332, 2.0.333, 2.0.334, 2.0.335, 2.0.336, 2.0.337, 2.0.338, 2.0.339, 2.0.340, 2.0.341, 2.0.342, 2.0.343, 2.0.344, 2.0.345, 2.0.346, 2.0.347, 2.0.348, 2.0.349, 2.0.350, 2.0.351, 2.0.352, 2.0.353, 2.0.354, 2.0.355, 2.0.356, 2.0.357, 2.0.358, 2.0.359, 2.0.360, 2.0.361, 2.0.362, 2.0.363, 2.0.364, 2.0.365, 2.0.366, 2.0.367, 2.0.368, 2.0.369, 2.0.370, 2.0.371, 2.0.372, 2.0.373, 2.0.374, 2.0.375, 2.0.376, 2.0.377, 2.0.378, 2.0.379, 2.0.380, 2.0.381, 2.0.382, 2.0.383, 2.0.384, 2.0.385, 2.0.386, 2.0.387, 2.0.388, 2.0.389, 2.0.390, 2.0.391, 2.0.392, 2.0.393, 2.0.394, 2.0.395, 2.0.396, 2.0.397, 2.0.398, 2.0.399, 2.0.400, 2.0.401, 2.0.402, 2.0.403, 2.0.404, 2.0.405, 2.0.406, 2.0.407, 2.0.408, 2.0.409, 2.0.410, 2.0.411, 2.0.412, 2.0.413, 2.0.414, 2.0.415, 2.0.416, 2.0.417, 2.0.418, 2.0.419, 2.0.420, 2.0.421, 2.0.422, 2.0.423, 2.0.424, 2.0.425, 2.0.426, 2.0.427, 2.0.428, 2.0.429, 2.0.430, 2.0.431, 2.0.432, 2.0.433, 2.0.434, 2.0.435, 2.0.436, 2.0.437, 2.0.438, 2.0.439, 2.0.440, 2.0.441, 2.0.442, 2.0.443, 2.0.444, 2.0.445, 2.0.446, 2.0.447, 2.0.448, 2.0.449, 2.0.450, 2.0.451, 2.0.452, 2.0.453, 2.0.454, 2.0.455, 2.0.456, 2.0.457, 2.0.458, 2.0.459, 2.0.460, 2.0.461, 2.0.462, 2.0.463, 2.0.464, 2.0.465, 2.0.466, 2.0.467, 2.0.468, 2.0.469, 2.0.470, 2.0.471, 2.0.472, 2.0.473, 2.0.474, 2.0.475, 2.0.476, 2.0.477, 2.0.478, 2.0.479, 2.0.480, 2.0.481, 2.0.482, 2.0.483, 2.0.484, 2.0.485, 2.0.486, 2.0.487, 2.0.488, 2.0.489, 2.0.490, 2.0.491, 2.0.492, 2.0.493, 2.0.494, 2.0.495, 2.0.496, 2.0.497, 2.0.498, 2.0.499, 2.0.500, 2.0.501, 2.0.502, 2.0.503, 2.0.504, 2.0.505, 2.0.506, 2.0.507, 2.0.508, 2.0.509, 2.0.510, 2.0.511, 2.0.512, 2.0.513, 2.0.514, 2.0.515, 2.0.516, 2.0.517, 2.0.518, 2.0.519, 2.0.520, 2.0.521, 2.0.522, 2.0.523, 2.0.524, 2.0.525, 2.0.526, 2.0.527, 2.0.528, 2.0.529, 2.0.530, 2.0.531, 2.0.532, 2.0.533, 2.0.534, 2.0.535, 2.0.536, 2.0.537, 2.0.538, 2.0.539, 2.0.540, 2.0.541, 2.0.542, 2.0.543, 2.0.544, 2.0.545, 2.0.546, 2.0.547, 2.0.548, 2.0.549, 2.0.550, 2.0.551, 2.0.552, 2.0.553, 2.0.554, 2.0.555, 2.0.556, 2.0.557, 2.0.558, 2.0.559, 2.0.560, 2.0.561, 2.0.562, 2.0.563, 2.0.564, 2.0.565, 2.0.566, 2.0.567, 2.0.568, 2.0.569, 2.0.570, 2.0.571, 2.0.572, 2.0.573, 2.0.574, 2.0.575, 2.0.576, 2.0.577, 2.0.578, 2.0.579, 2.0.580, 2.0.581, 2.0.582, 2.0.583, 2.0.584, 2.0.585, 2.0.586, 2.0.587, 2.0.588, 2.0.589, 2.0.590, 2.0.591, 2.0.592, 2.0.593, 2.0.594, 2.0.595, 2.0.596, 2.0.597, 2.0.598, 2.0.599, 2.0.600, 2.0.601, 2.0.602, 2.0.603, 2.0.604, 2.0.605, 2.0.606, 2.0.607, 2.0.608, 2.0.609, 2.0.610, 2.0.611, 2.0.612, 2.0.613, 2.0.614, 2.0.615, 2.0.616, 2.0.617, 2.0.618, 2.0.619, 2.0.620, 2.0.621, 2.0.622, 2.0.623, 2.0.624, 2.0.625, 2.0.626, 2.0.627, 2.0.628, 2.0.629, 2.0.630, 2.0.631, 2.0.632, 2.0.633, 2.0.634, 2.0.635, 2.0.636, 2.0.637, 2.0.638, 2.0.639, 2.0.640, 2.0.641, 2.0.642, 2.0.643, 2.0.644, 2.0.645, 2.0.646, 2.0.647, 2.0.648, 2.0.649, 2.0.650, 2.0.651, 2.0.652, 2.0.653, 2.0.654, 2.0.655, 2.0.656, 2.0.657, 2.0.658, 2.0.659, 2.0.660, 2.0.661, 2.0.662, 2.0.663, 2.0.664, 2.0.665, 2.0.666, 2.0.667, 2.0.668, 2.0.669, 2.0.670, 2.0.671, 2.0.672, 2.0.673, 2.0.674, 2.0.675, 2.0.676, 2.0.677, 2.0.678, 2.0.679, 2.0.680, 2.0.681, 2.0.682, 2.0.683, 2.0.684, 2.0.685, 2.0.686, 2.0.687, 2.0.688, 2.0.689, 2.0.690, 2.0.691, 2.0.692, 2.0.693, 2.0.694, 2.0.695, 2.0.696, 2.0.697, 2.0.698, 2.0.699, 2.0.700, 2.0.701, 2.0.702, 2.0.703, 2.0.704, 2.0.705, 2.0.706, 2.0.707, 2.0.708, 2.0.709, 2.0.710, 2.0.711, 2.0.712, 2.0.713, 2.0.714, 2.0.715, 2.0.716, 2.0.717, 2.0.718, 2.0.719, 2.0.720, 2.0.721, 2.0.722, 2.0.723, 2.0.724, 2.0.725, 2.0.726, 2.0.727, 2.0.728, 2.0.729, 2.0.730, 2.0.731, 2.0.732, 2.0.733, 2.0.734, 2.0.735, 2.0.736, 2.0.737, 2.0.738, 2.0.739, 2.0.740, 2.0.741, 2.0.742, 2.0.743, 2.0.744, 2.0.745, 2.0.746, 2.0.747, 2.0.748, 2.0.749, 2.0.750, 2.0.751, 2.0.752, 2.0.753, 2.0.754, 2.0.755, 2.0.756, 2.0.757, 2.0.758, 2.0.759, 2.0.760, 2.0.761, 2.0.762, 2.0.763, 2.0.764, 2.0.765, 2.0.766, 2.0.767, 2.0.768, 2.0.769, 2.0.770, 2.0.771, 2.0.772, 2.0.773, 2.0.774, 2.0.775, 2.0.776, 2.0.777, 2.0.778, 2.0.779, 2.0.780, 2.0.781, 2.0.782, 2.0.783, 2.0.784, 2.0.785, 2.0.786, 2.0.787, 2.0.788, 2.0.789, 2.0.790, 2.0.791, 2.0.792, 2.0.793, 2.0.794, 2.0.795, 2.0.796, 2.0.797, 2.0.798, 2.0.799, 2.0.800, 2.0.801, 2.0.802, 2.0.803, 2.0.804, 2.0.805, 2.0.806, 2.0.807, 2.0.808, 2.0.809, 2.0.810, 2.0.811, 2.0.812, 2.0.813, 2.0.814, 2.0.815, 2.0.816, 2.0.817, 2.0.818, 2.0.819, 2.0.820, 2.0.821, 2.0.822, 2.0.823, 2.0.824, 2.0.825, 2.0.826, 2.0.827, 2.0.828, 2.0.829, 2.0.830, 2.0.831, 2.0.832, 2.0.833, 2.0.834, 2.0.835, 2.0.836, 2.0.837, 2.0.838, 2.0.839, 2.0.840, 2.0.841, 2.0.842, 2.0.843, 2.0.844, 2.0.845, 2.0.846, 2.0.847, 2.0.848, 2.0.849, 2.0.850, 2.0.851, 2.0.852, 2.0.853, 2.0.854, 2.0.855, 2.0.856, 2.0.857, 2.0.858, 2.0.859, 2.0.860, 2.0.861, 2.0.862, 2.0.863, 2.0.864, 2.0.865, 2.0.866, 2.0.867, 2.0.868, 2.0.869, 2.0.870, 2.0.871, 2.0.872, 2.0.873, 2.0.874, 2.0.875, 2.0.876, 2.0.877, 2.0.878, 2.0.879, 2.0.880, 2.0.881, 2.0.882, 2.0.883, 2.0.884, 2.0.885, 2.0.886, 2.0.887, 2.0.888, 2.0.889, 2.0.890, 2.0.891, 2.0.892, 2.0.893, 2.0.894, 2.0.895, 2.0.896, 2.0.897, 2.0.898, 2.0.899, 2.0.900, 2.0.901, 2.0.902, 2.0.903, 2.0.904, 2.0.905, 2.0.906, 2.0.907, 2.0.908, 2.0.909, 2.0.910, 2.0.911, 2.0.912, 2.0.913, 2.0.914, 2.0.915, 2.0.916, 2.0.917, 2.0.918, 2.0.919, 2.0.920, 2.0.921, 2.0.922, 2.0.923, 2.0.924, 2.0.925, 2.0.926, 2.0.927, 2.0.928, 2.0.929, 2.0.930, 2.0.931, 2.0.932, 2.0.933, 2.0.934, 2.0.935, 2.0.936, 2.0.937, 2.0.938, 2.0.939, 2.0.940, 2.0.941, 2.0.942, 2.0.943, 2.0.944, 2.0.945, 2.0.946, 2.0.947, 2.0.948, 2.0.949, 2.0.950, 2.0.951, 2.0.952, 2.0.953, 2.0.954, 2.0.955, 2.0.956, 2.0.957, 2.0.958, 2.0.959, 2.0.960, 2.0.961, 2.0.962, 2.0.963, 2.0.964, 2.0.965, 2.0.966, 2.0.967, 2.0.968, 2.0.969, 2.0.970, 2.0.971, 2.0.972, 2.0.973, 2.0.974, 2.0.975, 2.0.976, 2.0.977, 2.0.978, 2.0.979, 2.0.980, 2.0.981, 2.0.982, 2.0.983, 2.0.984, 2.0.985, 2.0.986, 2.0.987, 2.0.988, 2.0.989, 2.0.990, 2.0.991, 2.0.992, 2.0.993, 2.0.994, 2.0.995, 2.0.996, 2.0.997, 2.0.998, 2.0.999, 2.0.1000, 2.0.1001, 2.0.1002, 2.0.1003, 2.0.1004, 2.0.1005, 2.0.1006, 2.0.1007, 2.0.1008, 2.0.1009, 2.0.1010, 2.0.1011, 2.0.1012, 2.0.1013, 2.0.1014, 2.0.1015, 2.0.1016, 2.0.1017, 2.0.1018, 2.0.1019, 2.0.1020, 2.0.1021, 2.0.1022, 2.0.1023, 2.0.1024, 2.0.1025, 2.0.1026, 2.0.1027, 2.0.1028, 2.0.1029, 2.0.1030, 2.0.1031, 2.0.1032, 2.0.1033, 2.0.1034, 2.0.1035, 2.0.1036, 2.0.1037, 2.0.1038, 2.0.1039, 2.0.1040, 2.0.1041, 2.0.1042, 2.0.1043, 2.0.1044, 2.0.1045, 2.0.1046, 2.0.1047, 2.0.1048, 2.0.1049, 2.0.1050, 2.0.1051, 2.0.1052, 2.0.1053, 2.0.1054, 2.0.1055, 2.0.1056, 2.0.1057, 2.0.1058, 2.0.1059, 2.0.1060, 2.0.1061, 2.0.1062, 2.0.1063, 2.0.1064, 2.0.1065, 2.0.1066, 2.0.1067, 2.0.1068, 2.0.1069, 2.0.1070, 2.0.1071, 2.0.1072, 2.0.1073, 2.0.1074, 2.0.1075, 2.0.1076, 2.0.1077, 2.0.1078, 2.0.1079, 2.0.1080, 2.0.1081, 2.0.1082, 2.0.1083, 2.0.1084, 2.0.1085, 2.0.1086, 2.0.1087, 2.0.1088, 2.0.1089, 2.0.1090, 2.0.1091, 2.0.1092, 2.0.1093, 2.0.1094, 2.0.1095, 2.0.1096, 2.0.1097, 2.0.1098, 2.0.1099, 2.0.1100, 2.0.1101, 2.0.1102, 2.0.1103, 2.0.1104, 2.0.1105, 2.0.1106, 2.0.1107, 2.0.1108, 2.0.1109, 2.0.1110, 2.0.1111, 2.0.1112, 2.0.1113, 2.0.1114, 2.0.1115, 2.0.1116, 2.0.1117, 2.0.1118, 2.0.1119, 2.0.1120, 2.0.1121, 2.0.1122, 2.0.1123, 2.0.1124, 2.0.1125, 2.0.1126, 2.0.1127, 2.0.1128, 2.0.1129, 2.0.1130, 2.0.1131, 2.0.1132, 2.0.1133, 2.0.1134, 2.0.1135, 2.0.1136, 2.0.1137, 2.0.1138, 2.0.1139, 2.0.1140, 2.0.1141, 2.0.1142, 2.0.1143, 2.0.1144, 2.0.1145, 2.0.1146, 2.0.1147, 2.0.1148, 2.0.1149, 2.0.1150, 2.0.1151, 2.0.1152, 2.0.1153, 2.0.1154, 2.0.1155, 2.0.1156, 2.0.1157, 2.0.1158, 2.0.1159, 2.0.1160, 2.0.1161, 2.0.1162, 2.0.1163, 2.0.1164, 2.0.1165, 2.0.1166, 2.0.1167, 2.0.1168, 2.0.1169, 2.0.1170, 2.0.1171, 2.0.1172, 2.0.1173, 2.0.1174, 2.0.1175, 2.0.1176, 2.0.1177, 2.0.1178, 2.0.1179, 2.0.1180, 2.0.1181, 2.0.1182, 2.0.1183, 2.0.1184, 2.0.1185, 2.0.1186, 2.0.1187, 2.0.1188, 2.0.1189, 2.0.1190, 2.0.1191, 2.0.1192, 2.0.1193, 2.0.1194, 2.0.1195, 2.0.1196, 2.0.1197, 2.0.1198, 2.0.1199, 2.0.1200, 2.0.1201, 2.0.1202, 2.0.1203, 2.0.1204, 2.0.1205, 2.0.1206, 2.0.1207, 2.0.1208, 2.0.1209, 2.0.1210, 2.0.1211, 2.0.1212, 2.0.1213, 2.0.1214, 2.0.1215, 2.0.1216, 2.0.1217, 2.0.1218, 2.0.1219, 2.0.1220, 2.0.1221, 2.0.1222, 2.0.1223, 2.0.1224, 2.0.1225, 2.0.1226, 2.0.1227, 2.0.1228, 2.0.1229, 2.0.1230, 2.0.1231, 2.0.1232, 2.0.1233, 2.0.1234, 2.0.1235, 2.0.1236, 2.0.1237, 2.0.1238, 2.0.1239, 2.0.1240, 2.0.1241, 2.0.1242, 2.0.1243, 2.0.1244, 2.0.1245, 2.0.1246, 2.0.1247, 2.0.1248, 2.0.1249, 2.0.1250, 2.0.1251, 2.0.1252, 2.0.1253, 2.0.1254, 2.0.1255, 2.0.1256, 2.0.1257, 2.0.1258, 2.0.1259, 2.0.1260, 2.0.1261, 2.0.1262, 2.0.1263, 2.0.1264, 2.0.1265, 2.0.1266, 2.0.1267, 2.0.1268, 2.0.1269, 2.0.1270, 2.0.1271, 2.0.1272, 2.0.1273, 2.0.1274, 2.0.1275, 2.0.1276, 2.0.1277, 2.0.1278, 2.0.1279, 2.0.1280, 2.0.1281, 2.0.1282, 2.0.1283, 2.0.1284, 2.0.1285, 2.0.1286, 2.0.1287, 2.0.1288, 2.0.1289, 2.0.1290, 2.0.1291, 2.0.1292

```

    , (150 146), (38 136), (111 170), (124 94), (26 59), (60 41), (71 162), (41 64), ↵
    (88 110), (122 34), (151 97), (157 56), (39 146), (88 33), (159 45), (47 56), ↵
    (138 40), (129 165), (33 48), (106 31), (169 147), (37 122), (71 109), (163 89), ↵
    (37 156), (82 170), (180 72), (29 142), (46 41), (59 155), (124 106), (157 80), ↵
    (175 82), (56 50), (62 116), (113 95), (144 167))',
    0.1 ) );
---st_astext--
POLYGON ((18 142, 21 151, 27 160, 35 167, 44 172, 54 175, 64 178, 74 180, 84 181, 94 181, ↵
104 181, 114 181, 124 181, 134 179, 144 177, 153 173, 162 168, 171 162, 177 154, 182 ↵
145, 184 135, 173 134, 161 134, 150 133, 139 132, 136 142, 128 149, 119 153, 109 155, 99 ↵
155, 89 155, 79 153, 69 150, 61 144, 63 134, 72 128, 82 125, 92 123, 102 121, 112 119, ↵
122 118, 132 116, 142 113, 151 110, 161 106, 170 102, 178 96, 185 88, 189 78, 190 68, ↵
189 58, 185 49, 179 41, 171 34, 162 29, 153 25, 143 23, 133 21, 123 19, 113 19, 102 19, ↵
92 19, 82 19, 72 21, 62 22, 52 25, 43 29, 33 34, 25 41, 19 49, 14 58, 10 72, 21 73, 31 ↵
74, 42 74, 53 76, 56 66, 63 58, 71 51, 81 48, 91 46, 101 45, 111 46, 121 47, 131 50, 140 ↵
55, 145 64, 144 74, 135 80, 125 83, 115 85, 105 87, 95 89, 85 91, 75 93, 65 95, 55 98, ↵
45 102, 37 107, 29 114, 22 122, 19 132, 18 142))

```



Concave Hull of a MultiPoint, allowing holes

```

SELECT ST_AsText( ST_ConcaveHull(
    'MULTIPOINT ((132 64), (114 64), (99 64), (81 64), (63 64), (57 49), (52 36), (46 ↵
    20), (37 20), (26 20), (32 36), (39 55), (43 69), (50 84), (57 100), (63 118), ↵
    (68 133), (74 149), (81 164), (88 180), (101 180), (112 180), (119 164), (126 ↵
    149), (132 131), (139 113), (143 100), (150 84), (157 69), (163 51), (168 36), ↵
    (174 20), (163 20), (150 20), (143 36), (139 49), (132 64), (99 151), (92 138), ↵
    (88 124), (81 109), (74 93), (70 82), (83 82), (99 82), (112 82), (126 82), (121 ↵
    96), (114 109), (110 122), (103 138), (99 151), (34 27), (43 31), (48 44), (46 ↵
    58), (52 73), (63 73), (61 84), (72 71), (90 69), (101 76), (123 71), (141 62), ↵
    (166 27), (150 33), (159 36), (146 44), (154 53), (152 62), (146 73), (134 76), ↵
    (143 82), (141 91), (130 98), (126 104), (132 113), (128 127), (117 122), (112 ↵
    133), (119 144), (108 147), (119 153), (110 171), (103 164), (92 171), (86 160), ↵
    (88 142), (79 140), (72 124), (83 131), (79 118), (68 113), (63 102), (68 93), ↵
    (35 45))',
    0.15, true ) );
---st_astext--
POLYGON ((43 69, 50 84, 57 100, 63 118, 68 133, 74 149, 81 164, 88 180, 101 180, 112 180, ↵
119 164, 126 149, 132 131, 139 113, 143 100, 150 84, 157 69, 163 51, 168 36, 174 20, 163 ↵
20, 150 20, 143 36, 139 49, 132 64, 114 64, 99 64, 81 64, 63 64, 57 49, 52 36, 46 20, ↵
37 20, 26 20, 32 36, 35 45, 39 55, 43 69), (88 124, 81 109, 74 93, 83 82, 99 82, 112 82, ↵
121 96, 114 109, 110 122, 103 138, 92 138, 88 124))

```

Using with `ST_Collect` to compute the concave hull of a geometry set.

```
-- Compute estimate of infected area based on point observations
SELECT disease_type,
       ST_ConcaveHull( ST_Collect(obs_pnt), 0.3 ) AS geom
FROM disease_obs
GROUP BY disease_type;
```

[ST_ConvexHull](#), [ST_GeomCollFromText](#), [ST_AlphaShape](#), [ST_OptimalAlphaShape](#)

8.14.6 ST_ConvexHull

`ST_ConvexHull` — Computes the convex hull of a geometry.

Synopsis

geometry **`ST_ConvexHull`**(geometry geomA);

Computes the convex hull of a geometry. The convex hull is the smallest convex geometry that encloses all geometries in the input.

One can think of the convex hull as the geometry obtained by wrapping an rubber band around a set of geometries. This is different from a **concave hull** which is analogous to "shrink-wrapping" the geometries. A convex hull is often used to determine an affected area based on a set of point observations.

In the general case the convex hull is a Polygon. The convex hull of two or more collinear points is a two-point LineString. The convex hull of one or more identical points is a Point.

This is not an aggregate function. To compute the convex hull of a set of geometries, use [ST_GeomCollFromText](#) to aggregate them into a geometry collection (e.g. `ST_ConvexHull(ST_Collect(geom))`).

GEOS



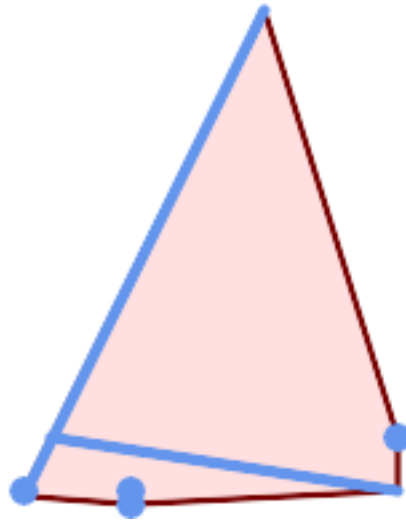
This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#). s2.1.1.3



This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.16



This function supports 3d and will not drop the z-index.



Convex Hull of a MultiLinestring and a MultiPoint

```
SELECT ST_AsText(ST_ConvexHull(
  ST_Collect(
    ST_GeomFromText('MULTILINESTRING((100 190,10 8),(150 10, 20 30))'),
    ST_GeomFromText('MULTIPOINT(50 5, 150 30, 50 10, 10 10)')
  )
));
---st_astext---
POLYGON((50 5,10 8,10 10,100 190,150 30,150 10,50 5))
```

Using with ST_Collect to compute the convex hulls of geometry sets.

```
--Get estimate of infected area based on point observations
SELECT d.disease_type,
  ST_ConvexHull(ST_Collect(d.geom)) As geom
FROM disease_obs As d
GROUP BY d.disease_type;
```

[ST_GeomCollFromText](#), [ST_ConcaveHull](#), [ST_MinimumBoundingCircle](#)

8.14.7 ST_DelaunayTriangles

ST_DelaunayTriangles — Returns the Delaunay triangulation of the vertices of a geometry.

Synopsis

geometry **ST_DelaunayTriangles**(geometry g1, float tolerance, int4 flags);

Return the **Delaunay triangulation** of the vertices of the input geometry. Output is a COLLECTION of polygons (for flags=0) or a MULTILINESTRING (for flags=1) or TIN (for flags=2). The tolerance, if any, is used to snap input vertices together.

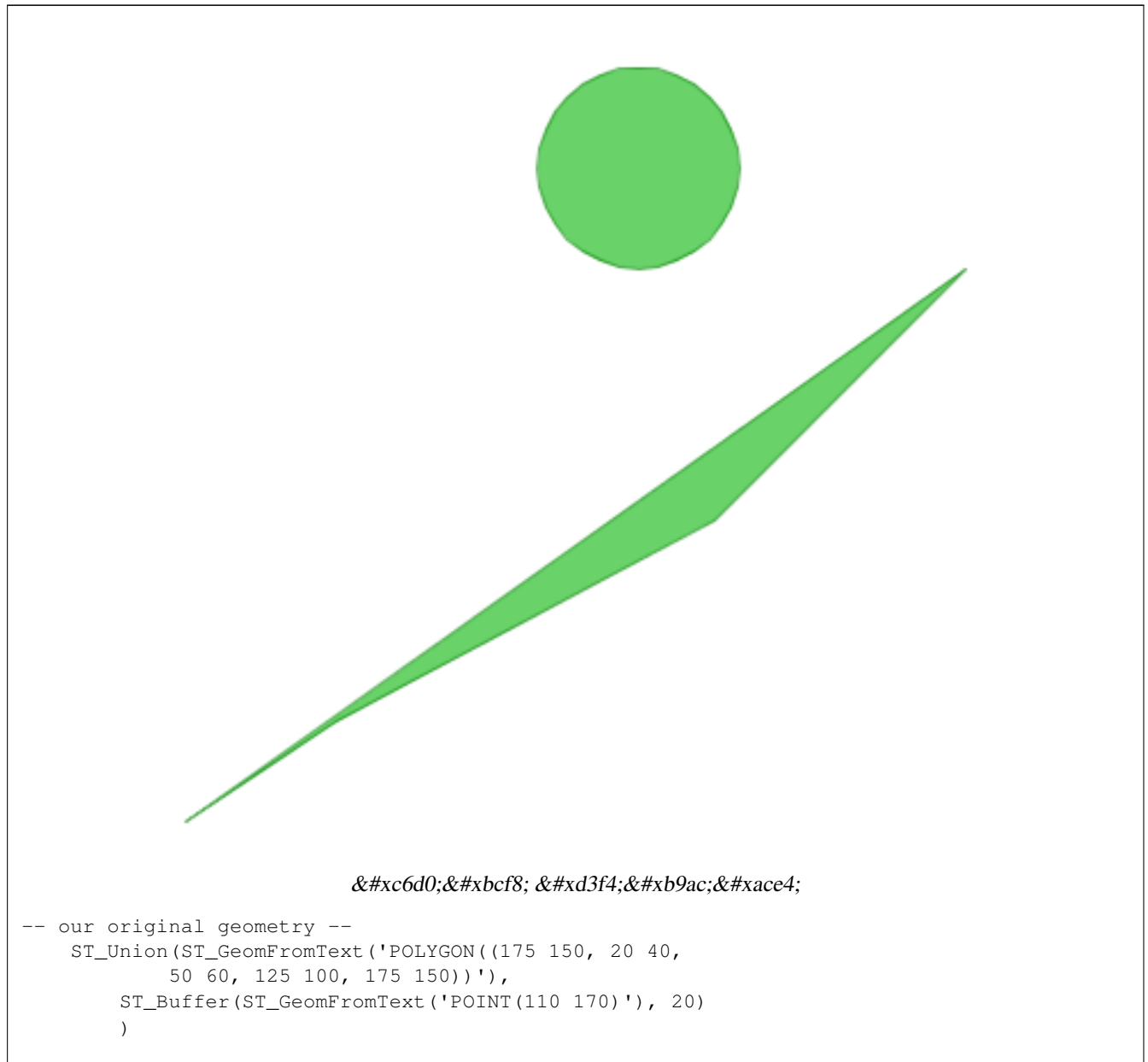
GEOS

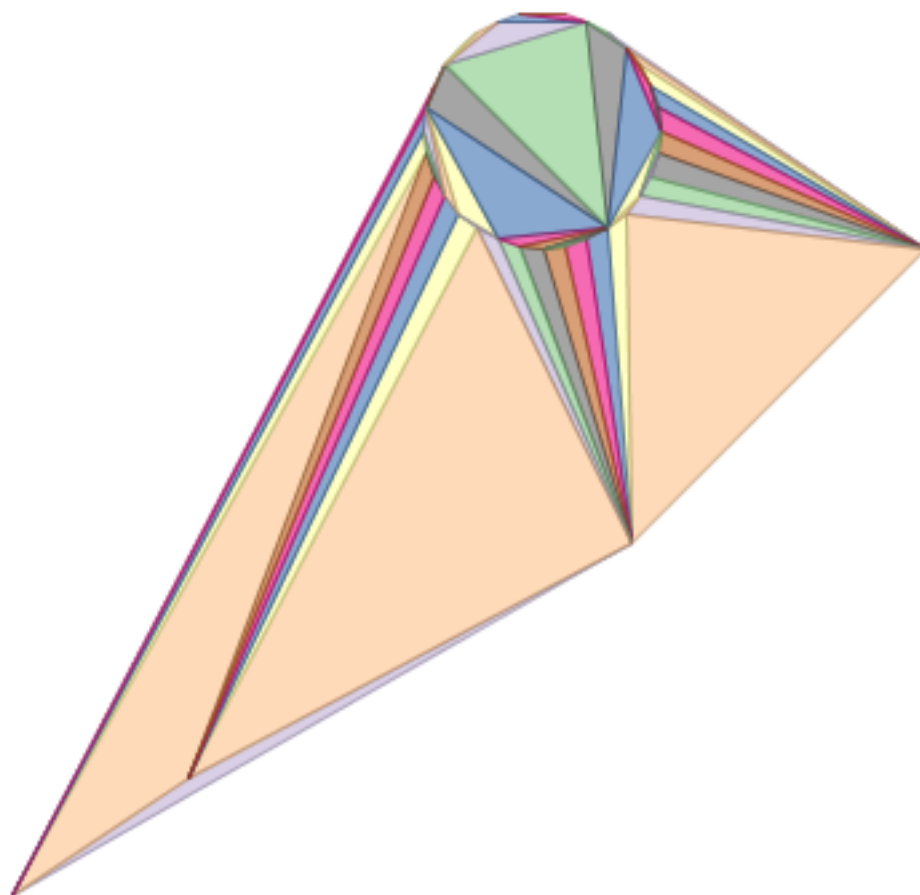
2.1.0

✓ This function supports 3d and will not drop the z-index.

✓ This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

2

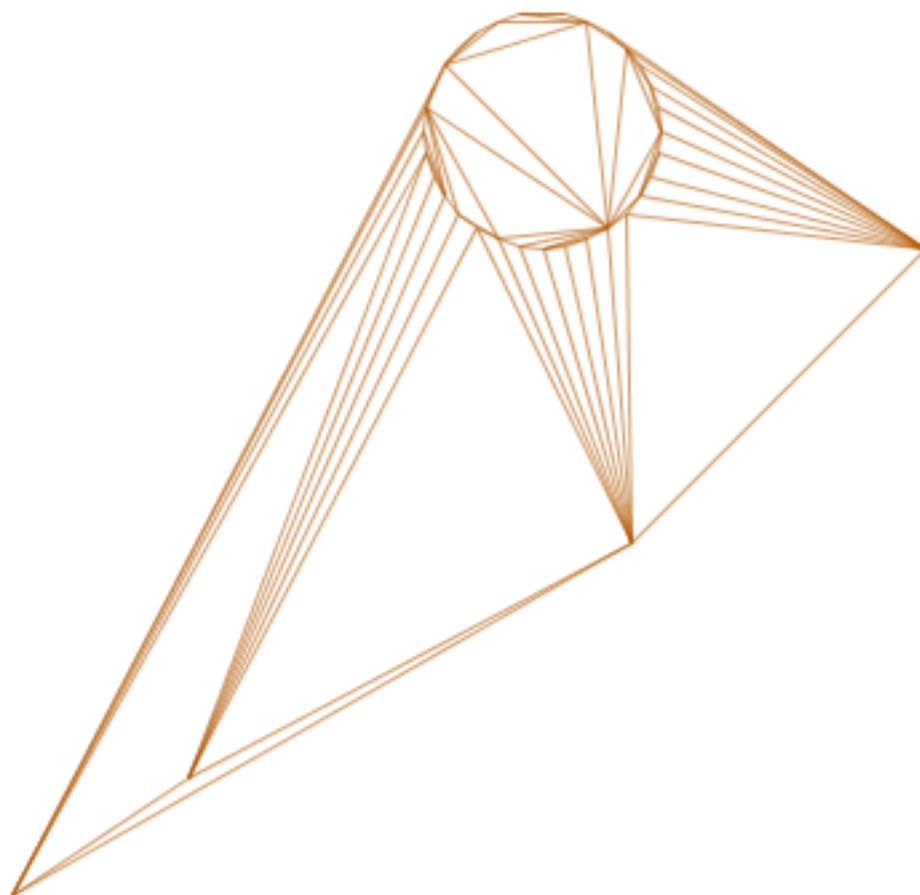




```

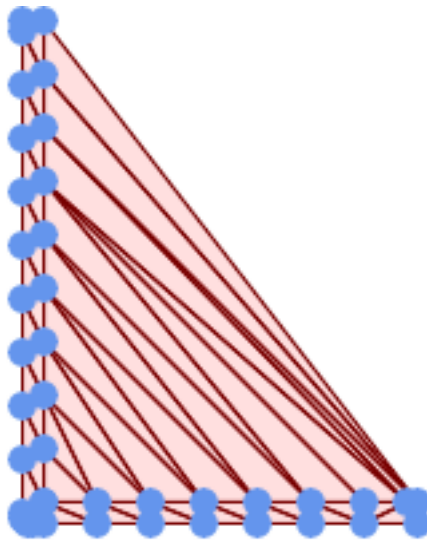
-- geometries overlaid multilinestring triangles
SELECT
  ST_DelaunayTriangles(
    ST_Union(ST_GeomFromText('POLYGON((175 150, 20 40,
      50 60, 125 100, 175 150))'),
    ST_Buffer(ST_GeomFromText('POINT(110 170)'), 20)
  )
  As dtriag;

```



```
-- &#xba40;&#xd2f0;&#xb77c;&#xc778;&#xc2a4;&#xd2b8;&#xb9c1;&#xc778; &#xb4e4;&#xb85c;&#xb124;  
    &#xc0bc;&#xac01;&#xd615;
```

```
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;
```

```
-- &#xc0bc;&#xac01;&#xd615; &#xd3f4;&#xb9ac;&#xace4; 55&#xac1c;&#xc778; &#xd3ec;&#xc778;&#xd2b8;
45&#xac1c;&#xc758; &#xb4e4;&#xb85c;&#xb124; &#xc0bc;&#xac01;&#xd615;
```

```
-- 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))
:
:
```

Example 1

```
-- 3D &#xba40;&#xd2f0;&#xd3ec;&#xc778;&#xd2b8; --
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)))
```

8.14.8 ST_FilterByM

ST_FilterByM — Removes vertices based on their M value

Synopsis

geometry **ST_FilterByM**(geometry geom, double precision min, double precision max = null, boolean returnM = false);

Ĥ명

Filters out vertex points based on their M-value. Returns a geometry with only vertex points that have a M-value larger or equal to the min value and smaller or equal to the max value. If max-value argument is left out only min value is considered. If fourth argument is left out the m-value will not be in the resulting geometry. If resulting geometry have too few vertex points left for its geometry type an empty geometry will be returned. In a geometry collection geometries without enough points will just be left out silently.

This function is mainly intended to be used in conjunction with ST_SetEffectiveArea. ST_EffectiveArea sets the effective area of a vertex in its m-value. With ST_FilterByM it then is possible to get a simplified version of the geometry without any calculations, just by filtering



Note

There is a difference in what ST_SimplifyVW returns when not enough points meet the criteria compared to ST_FilterByM. ST_SimplifyVW returns the geometry with enough points while ST_FilterByM returns an empty geometry



Note

Note that the returned geometry might be invalid



Note

This function returns all dimensions, including the Z and M values

Availability: 2.5.0

예시

A linestring is filtered

```
SELECT ST_AsText(ST_FilterByM(geom,30)) simplified
FROM (SELECT ST_SetEffectiveArea('LINESTRING(5 2, 3 8, 6 20, 7 25, 10 10)::geometry) geom ↵
) As foo;
-result
      simplified
-----
LINESTRING(5 2,7 25,10 10)
```

참고

[ST_SetEffectiveArea](#), [ST_SimplifyVW](#)

8.14.9 ST_GeneratePoints

ST_GeneratePoints — Generates random points contained in a Polygon or MultiPolygon.

Synopsis

```
geometry ST_GeneratePoints( g geometry , npoints integer );
geometry ST_GeneratePoints( geometry g , integer npoints , integer seed );
```

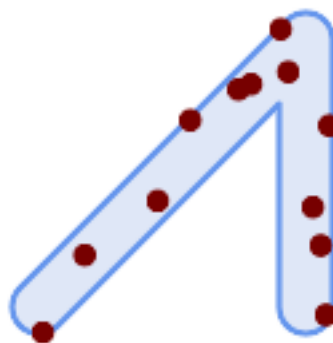
Notes:

ST_GeneratePoints generates a given number of pseudo-random points which lie within the input area. The optional `seed` is used to regenerate a deterministic sequence of points, and must be greater than zero.

2.3.0 Added `seed` parameter.

Enhanced: 3.0.0, added `seed` parameter

Example:



Generated 12 Points overlaid on top of original polygon using a random seed value 1996

```
SELECT ST_GeneratePoints(geom, 12, 1996)
FROM (
  SELECT ST_Buffer(
    ST_GeomFromText(
      'LINESTRING(50 50,150 150,150 50)'),
    10, 'endcap=round join=round') AS geom
  ) AS s;
```

8.14.10 ST_GeometricMedian

ST_GeometricMedian — Returns the geometric median of a set of points. The geometric median is the point that minimizes the sum of the distances to all the points in the set.

Synopsis

geometry **ST_GeometricMedian** (geometry geom, float8 tolerance = NULL, int max_iter = 10000, boolean fail_if_not_converged = false);

Computes the approximate geometric median of a MultiPoint geometry using the Weiszfeld algorithm. The geometric median is the point minimizing the sum of distances to the input points. It provides a centrality measure that is less sensitive to outlier points than the centroid (center of mass).

The algorithm iterates 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 produces an error and exits, unless fail_if_not_converged is set to false (the default).

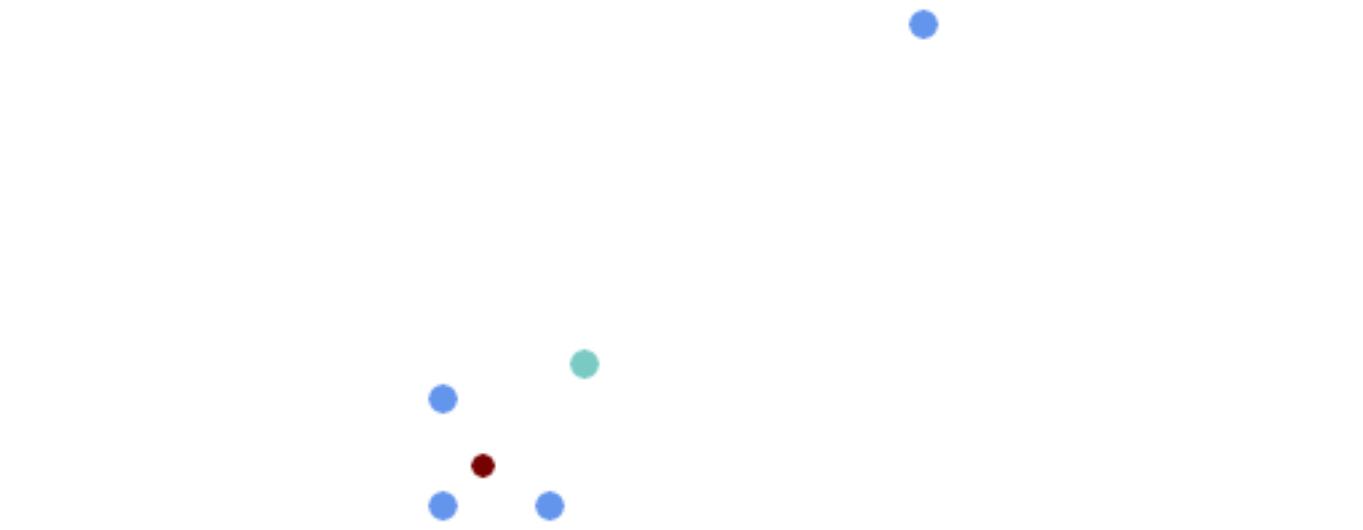
If a tolerance argument is not provided, the tolerance value is calculated based on the extent of the input geometry.

If present, the input point M values are interpreted as their relative weights.

2.3.0 Added support for M as weight of points.

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.



Comparison of the geometric median (red) and centroid (turquoise) of a MultiPoint.

```
WITH test AS (  
SELECT 'MULTIPOINT((10 10), (10 40), (40 10), (190 190))'::geometry geom  
SELECT  
  ST_AsText(ST_Centroid(geom)) centroid,  
  ST_AsText(ST_GeometricMedian(geom)) median  
FROM test;  
  
centroid | median  
-----+-----  
POINT(62.5 62.5) | POINT(25.01778421249728 25.01778421249728)  
(1 row)
```

ST_Centroid

ST_Centroid

8.14.11 ST_LineMerge

ST_LineMerge — Return the lines formed by sewing together a MultiLineString.

Synopsis

```
geometry ST_LineMerge(geometry amultilinestring);
geometry ST_LineMerge(geometry amultilinestring, boolean directed);
```

ST_LineMerge

Returns a LineString or MultiLineString formed by joining together the line elements of a MultiLineString. Lines are joined at their endpoints at 2-way intersections. Lines are not joined across intersections of 3-way or greater degree.

If **directed** is TRUE, then ST_LineMerge will not change point order within LineStrings, so lines with opposite directions will not be merged



Note

Only use with MultiLineString/LineStrings. Other geometry types return an empty GeometryCollection

GEOS 3.10.0-rc1

Enhanced: 3.3.0 accept a directed parameter - requires GEOS >= 3.11.0

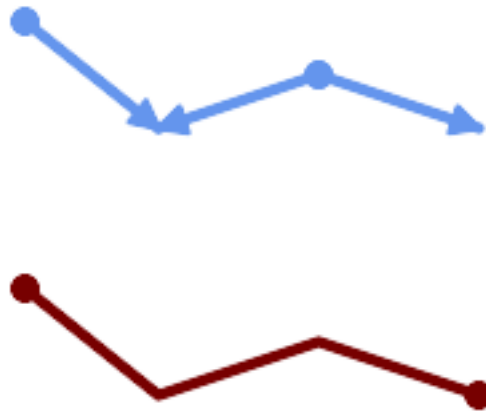
1.1.0 GEOS 3.10.0-rc1



Warning

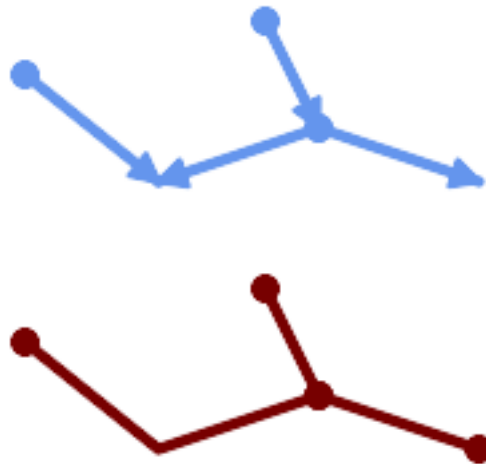
This function strips the M dimension.

예시



Merging lines with different orientation.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((10 160, 60 120), (120 140, 60 120), (120 140, 180 120))'
));
-----
LINESTRING(10 160,60 120,120 140,180 120)
```

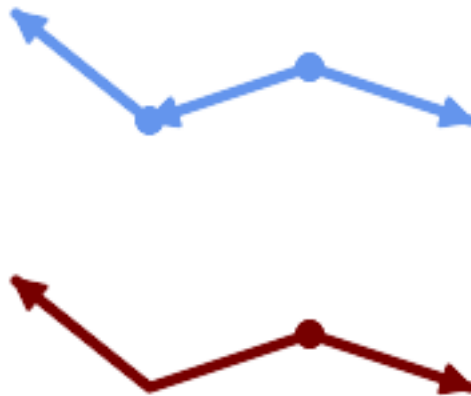


Lines are not merged across intersections with degree > 2.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((10 160, 60 120), (120 140, 60 120), (120 140, 180 120), (100 180, 120 140))'
));
-----
MULTILINESTRING((10 160,60 120,120 140),(100 180,120 140),(120 140,180 120))
```

If merging is not possible due to non-touching lines, the original MultiLineString is returned.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33), (-45.2 -33.2,-46 -32))'
));
-----
MULTILINESTRING((-45.2 -33.2,-46 -32), (-29 -27,-30 -29.7,-36 -31,-45 -33))
```



Lines with opposite directions are not merged if directed = TRUE.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((60 30, 10 70), (120 50, 60 30), (120 50, 180 30))',
TRUE));
-----
MULTILINESTRING((120 50,60 30,10 70), (120 50,180 30))
```

Example showing Z-dimension handling.

```
SELECT ST_AsText(ST_LineMerge(
'MULTILINESTRING((-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 6), (-29 -27 12,-30 -29.7 5), (-45 -33 1,-46 -32 11))'
));
-----
LINESTRING Z (-30 -29.7 5,-29 -27 11,-30 -29.7 10,-36 -31 5,-45 -33 1,-46 -32 11)
```

ST_Segmentize, ST_LineSubstring

ST_Segmentize, ST_LineSubstring

8.14.12 ST_MaximumInscribedCircle

ST_MaximumInscribedCircle — Returns the maximum inscribed circle of a geometry. The circle is defined by its center point and its radius. The circle is the largest circle that can be inscribed within the geometry.

Synopsis

(geometry, geometry, double precision) **ST_MaximumInscribedCircle**(geometry geom);

ST_MaximumInscribedCircle

Finds the largest circle that is contained within a (multi)polygon, or which does not overlap any lines and points. Returns a record with fields:

- `center` - center point of the circle
- `nearest` - a point on the geometry nearest to the center
- `radius` - radius of the circle

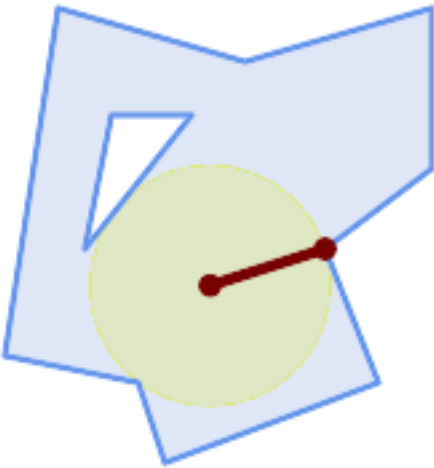
For polygonal inputs, the circle is inscribed within the boundary rings, using the internal rings as boundaries. For linear and point inputs, the circle is inscribed within the convex hull of the input, using the input lines and points as further boundaries.

Availability: 3.1.0 - requires GEOS >= 3.9.0.

ST_MinimumBoundingCircle

ST_MinimumBoundingCircle

ST_MinimumBoundingCircle

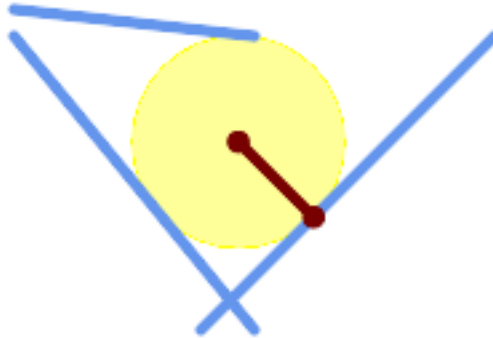


Maximum inscribed circle of a polygon. Center, nearest point, and radius are returned.

```
SELECT radius, ST_AsText(center) AS center, ST_AsText(nearest) AS nearest
FROM ST_MaximumInscribedCircle(
  'POLYGON ((40 180, 110 160, 180 180, 180 120, 140 90, 160 40, 80 10, 70 40, 20 50, 40 180),
    (60 140, 50 90, 90 140, 60 140))');

```

radius	center	nearest
45.165845650018	POINT(96.953125 76.328125)	POINT(140 90)



Maximum inscribed circle of a multi-linestring. Center, nearest point, and radius are returned.

ST_MinimumBoundingRadius

ST_MinimumBoundingRadius

8.14.13 ST_MinimumBoundingCircle

ST_MinimumBoundingCircle — Returns the smallest circle polygon that contains a geometry.

Synopsis

geometry **ST_MinimumBoundingCircle**(geometry geomA, integer num_segs_per_qt_circ=48);

ST_MinimumBoundingCircle

Returns the smallest circle polygon that contains a geometry.

Note

The **ST_MinimumBoundingCircle** function returns the smallest circle polygon that contains a geometry. The circle is defined by its center point and its radius. The center point is the point on the geometry that is closest to the center of the circle. The radius is the distance from the center point to the farthest point on the geometry. The function returns the circle as a polygon with 48 segments per quadrant.


```

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))

```

ST_MinimumBoundingRadius

ST_GeomCollFromText, ST_MinimumBoundingRadius

8.14.14 ST_MinimumBoundingRadius

ST_MinimumBoundingRadius — Returns the center point and radius of the smallest circle that contains a geometry.

Synopsis

(geometry, double precision) **ST_MinimumBoundingRadius**(geometry geom);

ST_MinimumBoundingRadius

Computes the center point and radius of the smallest circle that contains a geometry. Returns a record with fields:

- **center** - center point of the circle
- **radius** - radius of the circle

Use in conjunction with **ST_GeomCollFromText** to get the minimum bounding circle of a set of geometries.

2.3.0 **ST_MinimumBoundingRadius**(geometry geom);

ST_MinimumBoundingRadius

ST_GeomCollFromText, ST_MinimumBoundingCircle

ST_MinimumBoundingRadius

```

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

8.14.15 ST_OrientedEnvelope

ST_OrientedEnvelope — Returns a minimum-area rectangle containing a geometry.

Synopsis

geometry **ST_OrientedEnvelope**(geometry geom);

Parameters

Returns the minimum-area rotated rectangle enclosing a geometry. Note that more than one such rectangle may exist. May return a Point or LineString in the case of degenerate inputs.

Availability: 2.5.0

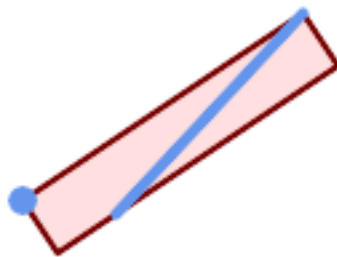
See also

[ST_Envelope](#) [ST_MinimumBoundingCircle](#)

Examples

```
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))
```

8.14.16 ST_OffsetCurve

ST_OffsetCurve — Returns an offset line at a given distance and side from an input line.

Synopsis

geometry **ST_OffsetCurve**(geometry line, float signed_distance, text style_parameters=');

설명

Return an offset line at a given distance and side from an input line. All points of the returned geometries are not further than the given distance from the input geometry. Useful for computing parallel lines about a center line.

For positive distance the offset is on the left side of the input line and retains the same direction. For a negative distance it is on the right side and in the opposite direction.

거리의단위는공간참조시스템단위로측정됩니다.

Note that output may be a MULTILINESTRING or EMPTY for some jigsaw-shaped input geometries.

다음과같이연산을조정하기위선택적인세번째파라미터로공구분된키=값쌍의목록을설정할수있습니다:

- 'quad_segs=#' : 사분원(quarter circle)의근사치를계산데쓰이는선분의개수(기본값은원마이터(mitre)'의동의어로'마이터(miter)'쓸수있습니다.
- 'join=round|mitre|bevel' : 접합스타일(기본값은'원마이터(mitre)'의동의어로'마이터(miter)'쓸수있습니다.
- 'mitre_limit=#.#' : 마이터비율제한(마이터접스타일만영향을받습니다). 'mitre_limit'의동의어로'miter_limit'도쓸수있습니다.

GEOS 모듈로실행

2.0 버전부터사용할수있습니다.

Enhanced: 2.5 - added support for GEOMETRYCOLLECTION and MULTILINESTRING



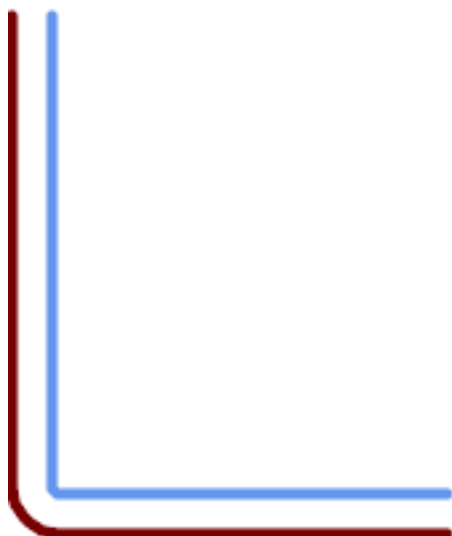
Note

This function ignores the Z dimension. It always gives a 2D result even when used on a 3D geometry.

예시

도로주위로열린버퍼를계산합의동의어로'miter_limit'도쓸수있습니다.

```
SELECT ST_Union(
  ST_OffsetCurve(f.geom, f.width/2, 'quad_segs=4 join=round'),
  ST_OffsetCurve(f.geom, -f.width/2, 'quad_segs=4 join=round')
) as track
FROM someroadstable;
```



```

    15, 'quad_segs=4
    join=round' &#x0f;
    15&#xb2e8;&#xc704;

```

```

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)

```



```

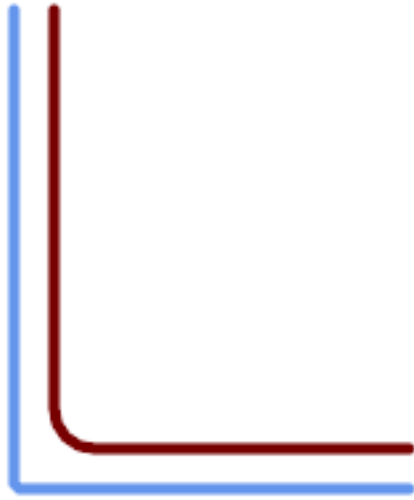
    -15, 'quad_segs=4
    join=round' &#x0f;
    -15&#xb2e8;&#xc704;

```

```

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)

```



```

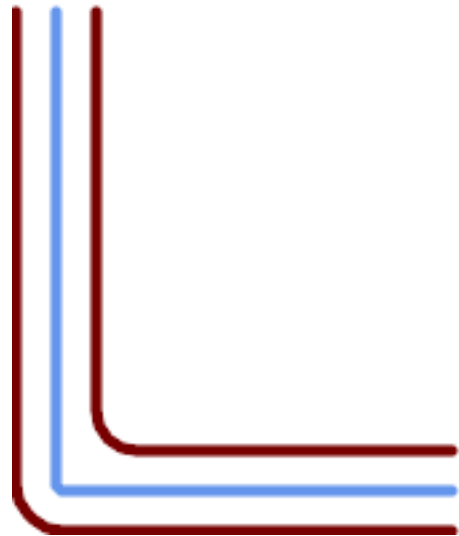
    &#xb354; &#xb465; &#xae00; &#xac8c;
    &#xb9cc; &#xb4e4; &#xae30; &#xc704; &#xd55c;
    &#xc774; &#xc911; &#xc624; &#xd504; &#xc14b;
    &#xccab; &#xbc88; &#xc9f8; &#xcffc; &#xb9ac; &#xac00;
    &#xbc29; &#xd5a5; &#xc744;
    &#xc5ed; &#xc804; &#xc2dc; &#xd0a8; &#xb2e4; &#xb294;
    &#xc810; &#xc5d0;
    &#xc8fc; &#xc758; &#xd558; &#xc2ed; &#xc2dc; &#xc624;
    &#xc989; -30 + 15 = -15 &#xc785; &#xb2c8; &#xb2e4;.

```

```

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)

```



```

    &#xd3c9; &#xd589; &#xd55c;
    &#xb77c; &#xc778; &#xc744; &#xc5bb; &#xae30;
    &#xc704; &#xd55c; &#xc815; &#xadde;
    &#xc624; &#xd504; &#xc14b; 15 &#xc640;
    &#xacb0; &#xd569; &#xb41c; &#xb354;
    &#xb465; &#xae00; &#xac8c;
    &#xb9cc; &#xb4e4; &#xae30; &#xc704; &#xd55c;
    &#xc774; &#xc911; &#xc624; &#xd504; &#xc14b;
    &#xc6d0; &#xbcf8; &#xacfc; &#xc911; &#xcca9;.

```

```

SELECT ST_AsText(ST_Collect(
  ST_OffsetCurve(geom, 15, 'quad_segs=4
    join=round'),
  ST_OffsetCurve(ST_OffsetCurve(geom,
    -30, 'quad_segs=4 join=round'), -15,
    'quad_segs=4 join=round')
)
) As parallel_curves
FROM ST_GeomFromText(
'LINESTRING(164 16,144 16,124 16,104
  16,84 16,64 16,
  44 16,24 16,20 16,18 16,17 17,
  16 18,16 20,16 40,16 60,16 80,16 100,
  16 120,16 140,16 160,16 180,16 195)')
  As geom;
-- parallel curves --
MULTILINESTRING((164 1,18
  1,12.2597485145237 2.1418070123307,
  7.39339828220179
  5.39339828220179,5.39339828220179 7.393398282201
  2.14180701233067 12.2597485145237,1 18,1
  195),
  (164 31,46 31,40.2597485145236
  32.1418070123307,35.3933982822018 35.39339828220
  32.1418070123307 40.2597485145237,31
  46,31 195))

```



```

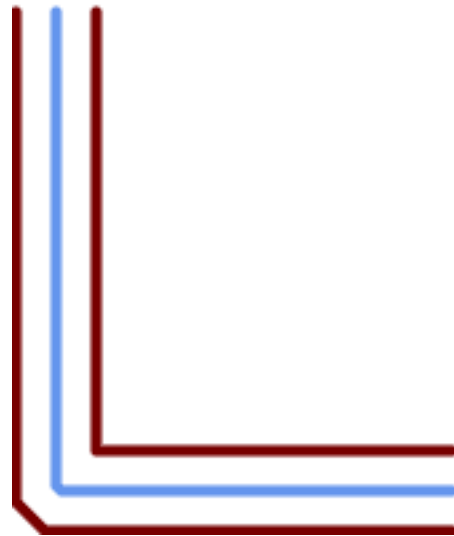
-- output --
LINESTRING(164 1,18 1,7.39339828220179 5.39339828220179,
5.39339828220179 7.39339828220179,1 18,1 195)

```

```

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)

```



```

-- output --
MULTILINESTRING((164 1,11.7867965644036 1,1 11.7867965644036,
(31 195,31 31,164 31))

```

```

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,
(31 195,31 31,164 31))

```

ST_Buffer

ST_Buffer

8.14.17 ST_PointOnSurface

ST_PointOnSurface — Computes a point guaranteed to lie in a polygon, or on a geometry.

Synopsis

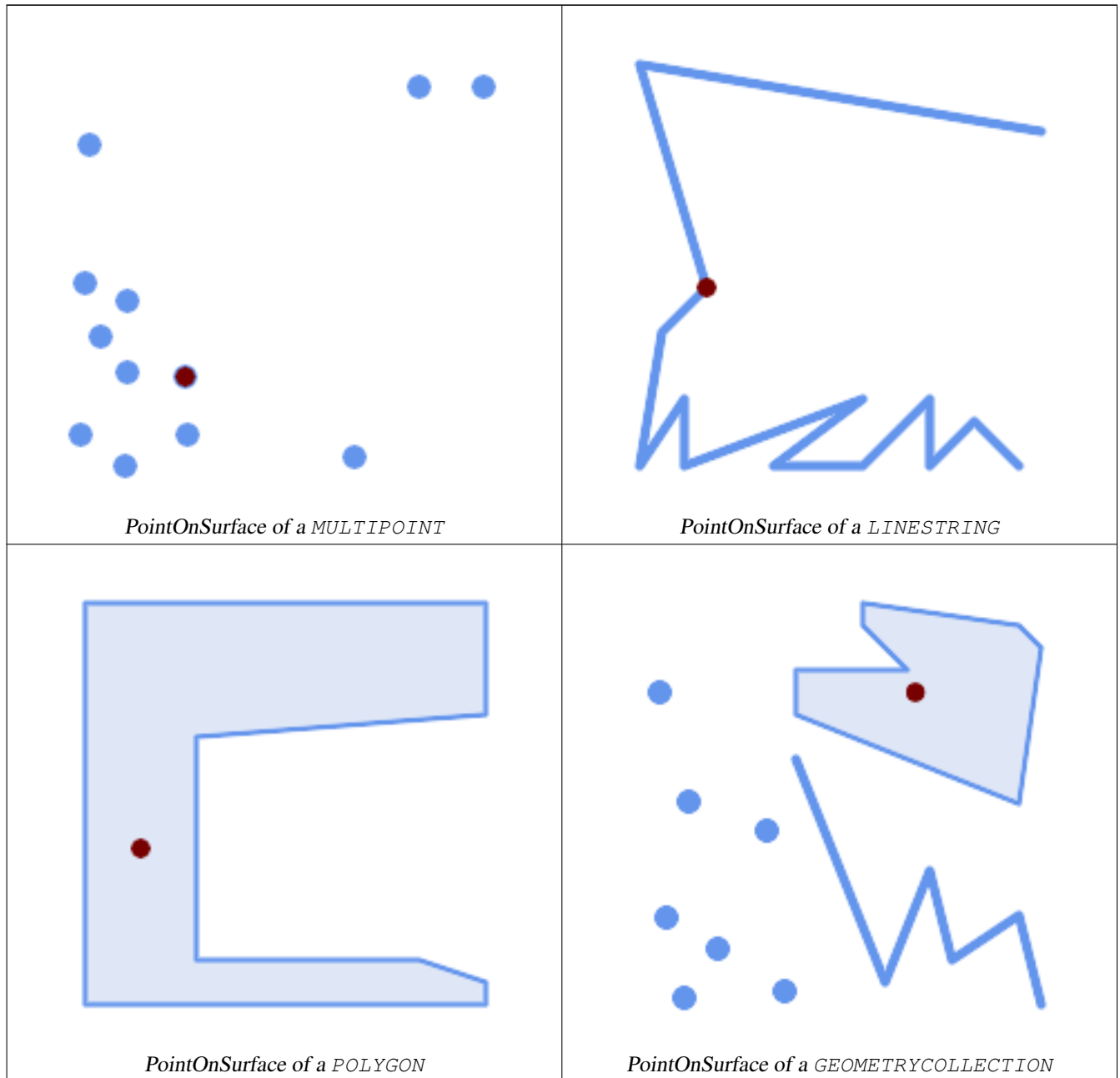
geometry **ST_PointOnSurface**(geometry g1);

ST_PointOnSurface

Returns a **POINT** which is guaranteed to lie in the interior of a surface (**POLYGON**, **MULTIPOLYGON**, and **CURVED POLYGON**). In PostGIS this function also works on line and point geometries.

- ✔ This method implements the [OGC 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. The specifications define ST_PointOnSurface for surface geometries only. PostGIS extends the function to support all common geometry types. Other databases (Oracle, DB2, ArcSDE) seem to support this function only for surfaces. SQL Server 2008 supports all common geometry types.
- ✔ This function supports 3d and will not drop the z-index.

예시



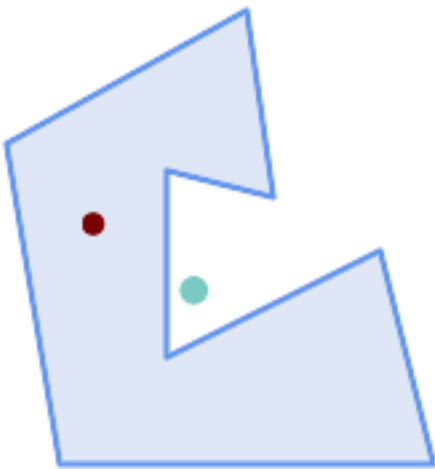
```
SELECT ST_AsText(ST_PointOnSurface('POINT(0 5)::geometry'));
-----
POINT(0 5)
```

```
SELECT ST_AsText(ST_PointOnSurface('LINESTRING(0 5, 0 10)::geometry));
-----
POINT(0 5)

SELECT ST_AsText(ST_PointOnSurface('POLYGON((0 0, 0 5, 5 5, 5 0, 0 0))::geometry));
-----
POINT(2.5 2.5)

SELECT ST_AsEWKT(ST_PointOnSurface(ST_GeomFromEWKT('LINESTRING(0 5 1, 0 0 1, 0 10 2)')));
-----
POINT(0 0 1)
```

Example: The result of ST_PointOnSurface is guaranteed to lie within polygons, whereas the point computed by **ST_Centroid** may be outside.



Red: point on surface; Green: centroid

```
SELECT ST_AsText(ST_PointOnSurface(geom)) AS pt_on_surf,
       ST_AsText(ST_Centroid(geom)) AS centroid
FROM (SELECT 'POLYGON ((130 120, 120 190, 30 140, 50 20, 190 20,
                        170 100, 90 60, 90 130, 130 120))::geometry AS geom) AS t;

pt_on_surf | centroid
-----+-----
POINT(62.5 110) | POINT(100.18264840182648 85.11415525114155)
```

Warning:

ST_Centroid, **ST_MaximumInscribedCircle**

8.14.18 ST_Polygonize

ST_Polygonize — Computes a collection of polygons formed from the linework of a set of geometries.

Synopsis

```
geometry ST_Polygonize(geometry set geomfield);
geometry ST_Polygonize(geometry[] geom_array);
```

ST_Polygonize

Creates a GeometryCollection containing the polygons formed by the constituent linework of a set of geometries. Input linework must be correctly noded for this function to work properly.

**Note**

To ensure input is fully noded use [ST_Node](#) on the input geometry before polygonizing.

**Note**

GeometryCollections are often difficult to deal with with third party tools. Use [ST_Dump](#) to convert the polygonize result into separate polygons.

GEOS 3.10.0-rc1

1.0.0RC1

Example 1: Polygonizing a set of edges

```
SELECT ST_AsEWKT(ST_Polygonize(geom_4269)) As geomtextrep
FROM (SELECT 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(geom_4269) As polycoll
      FROM (SELECT 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)
```

ST_Polygonize

[ST_Node](#), [ST_Dump](#)

8.14.19 ST_ReducePrecision

ST_ReducePrecision — Returns a valid geometry with points rounded to a grid tolerance.

Synopsis

geometry **ST_ReducePrecision**(geometry g, float8 gridsz);

Returns

Returns a valid geometry with all points rounded to the provided grid tolerance, and features below the tolerance removed.

Unlike **ST_SnapToGrid** the returned geometry will be valid, with no ring self-intersections or collapsed components.

Precision reduction can be used to:

- match coordinate precision to the data accuracy
- reduce the number of coordinates needed to represent a geometry
- ensure valid geometry output to formats which use lower precision (e.g. text formats such as WKT, GeoJSON or KML when the number of output decimal places is limited).
- export valid geometry to systems which use lower or limited precision (e.g. SDE, Oracle tolerance value)

Availability: 3.1.0 - requires GEOS >= 3.9.0.

Examples

```
SELECT ST_AsText(ST_ReducePrecision('POINT(1.412 19.323)', 0.1));
      st_astext
-----
POINT(1.4 19.3)

SELECT ST_AsText(ST_ReducePrecision('POINT(1.412 19.323)', 1.0));
      st_astext
-----
POINT(1 19)

SELECT ST_AsText(ST_ReducePrecision('POINT(1.412 19.323)', 10));
      st_astext
-----
POINT(0 20)
```

Precision reduction can reduce number of vertices

```
SELECT ST_AsText(ST_ReducePrecision('LINESTRING (10 10, 19.6 30.1, 20 30, 20.3 30, 40 40)', 1));
      st_astext
-----
LINESTRING (10 10, 20 30, 40 40)
```

Precision reduction splits polygons if needed to ensure validity

```
SELECT ST_AsText(ST_ReducePrecision('POLYGON ((10 10, 60 60.1, 70 30, 40 40, 50 10, 10 10))', 10));
      st_astext
-----
MULTIPOLYGON (((60 60, 70 30, 40 40, 60 60)), ((40 40, 50 10, 10 10, 40 40)))
```

See also

ST_SnapToGrid, **ST_Simplify**, **ST_SimplifyVW**

8.14.20 ST_SharedPaths

ST_SharedPaths — Returns the shared paths between two geometries. The result is a geometry of type `LINESTRING` containing the shared paths between the two input geometries. If the two input geometries do not share any paths, the result is an empty geometry.

Synopsis

geometry ST_SharedPaths(geometry lineal1, geometry lineal2);

Parameters

lineal1, lineal2: Geometry objects of type `LINESTRING` or `MULTILINESTRING`. The result is a geometry of type `LINESTRING` containing the shared paths between the two input geometries. If the two input geometries do not share any paths, the result is an empty geometry.

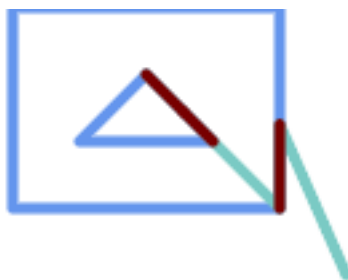
GEOS 3.10.0

2.0.0

See Also



ST_SharedPaths(geometry lineal1, geometry lineal2);



```

-- same example but linestring orientation flipped
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)))

```

ST_Dump, ST_GeometryN, ST_NumGeometries

ST_Dump, ST_GeometryN, ST_NumGeometries

8.14.21 ST_Simplify

ST_Simplify — Returns a simplified version of a geometry, using the Douglas-Peucker algorithm.

Synopsis

geometry **ST_Simplify**(geometry geomA, float tolerance);
 geometry **ST_Simplify**(geometry geomA, float tolerance, boolean preserveCollapsed);

설명

더글러스-패커(Douglas-Peucker) 알고리즘을
 이용해서 입력 도형의 "단순화"된
 버전을 반환합니다. 실제로는 [&#ba40;&
 [&#ba40;티]폴리곤과만 작동하지만, 어&#
 종류의 도형도 입력할 수 있다고
 해도 과언은 아닙니다. 객체별 기&#
 단순화 작업을 하기 때문에 이 함&#
 도형집합도 입력할 수 있습니다.

The "preserve collapsed" flag will retain objects that would otherwise be too small given the tolerance. For example, a 1m long line simplified with a 10m tolerance. If `preserveCollapsed` argument is specified as true, the line will not disappear. This flag is useful for rendering engines, to avoid having large numbers of very small objects disappear from a map leaving surprising gaps.



Note

반환되는 도형이 단순성을
 잃을 수도 있다는 점에
 주의하십시오(**ST_IsSimple** 참조).



Note

위상(topology)이 보전되지 않아
 유효하지 않은 도형이 반환될
 수도 있습니다. 위상을
 유지하려면 **ST_SimplifyPreserveTopology** 함수를
 이용하십시오.

1.2.2 버전부터 사용할 수 있습니다.

예시

원을 너무 단순화시키면 팔각형&#
 거췐 삼각형이 되고 맙니다.

```
SELECT ST_Npoints(geom) AS np_before,
       ST_NPoints(ST_Simplify(geom,0.1)) AS np01_notbadcircle,
       ST_NPoints(ST_Simplify(geom,0.5)) AS np05_notquitecircle,
       ST_NPoints(ST_Simplify(geom,1)) AS np1_octagon,
       ST_NPoints(ST_Simplify(geom,10)) AS np10_triangle,
       (ST_Simplify(geom,100) is null) AS np100_geometrygoesaway
FROM
  (SELECT ST_Buffer('POINT(1 3)', 10,12) As geom) AS foo;
```


ST_Simplify

ST_Simplify

8.14.23 ST_SimplifyPolygonHull

ST_SimplifyPolygonHull — Computes a simplified topology-preserving outer or inner hull of a polygonal geometry.

Synopsis

geometry **ST_SimplifyPolygonHull**(geometry param_geom, float vertex_fraction, boolean is_outer = true);

ST_SimplifyPolygonHull

Computes a simplified topology-preserving outer or inner hull of a polygonal geometry. An outer hull completely covers the input geometry. An inner hull is completely covered by the input geometry. The result is a polygonal geometry formed by a subset of the input vertices. MultiPolygons and holes are handled and produce a result with the same structure as the input.

The reduction in vertex count is controlled by the `vertex_fraction` parameter, which is a number in the range 0 to 1. Lower values produce simpler results, with smaller vertex count and less concaveness. For both outer and inner hulls a vertex fraction of 1.0 produces the original geometry. For outer hulls a value of 0.0 produces the convex hull (for a single polygon); for inner hulls it produces a triangle.

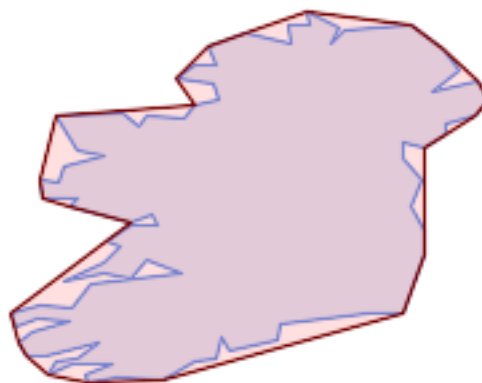
The simplification process operates by progressively removing concave corners that contain the least amount of area, until the vertex count target is reached. It prevents edges from crossing, so the result is always a valid polygonal geometry.

To get better results with geometries that contain relatively long line segments, it might be necessary to "segmentize" the input, as shown below.

GEOS 3.10.0 - requires GEOS >= 3.11.0

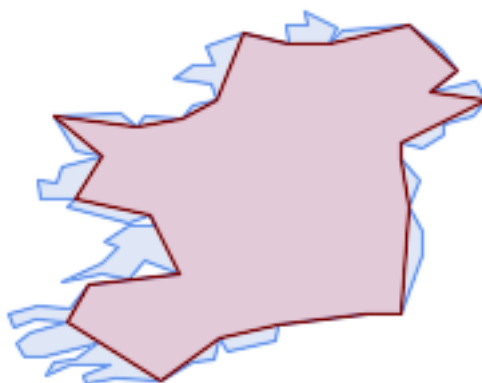
Availability: 3.3.0 - requires GEOS >= 3.11.0

ST_SimplifyPolygonHull



Outer hull of a Polygon

```
SELECT ST_SimplifyPolygonHull(
  'POLYGON ((131 158, 136 163, 161 165, 173 156, 179 148, 169 140, 186 144, 190 137, 185 ↵
    131, 174 128, 174 124, 166 119, 158 121, 158 115, 165 107, 161 97, 166 88, 166 79, 158 ↵
    57, 145 57, 112 53, 111 47, 93 43, 90 48, 88 40, 80 39, 68 32, 51 33, 40 31, 39 34, ↵
    49 38, 34 38, 25 34, 28 39, 36 40, 44 46, 24 41, 17 41, 14 46, 19 50, 33 54, 21 55, 13 ↵
    52, 11 57, 22 60, 34 59, 41 68, 75 72, 62 77, 56 70, 46 72, 31 69, 46 76, 52 82, 47 ↵
    84, 56 90, 66 90, 64 94, 56 91, 33 97, 36 100, 23 100, 22 107, 29 106, 31 112, 46 116, ↵
    36 118, 28 131, 53 132, 59 127, 62 131, 76 130, 80 135, 89 137, 87 143, 73 145, 80 ↵
    150, 88 150, 85 157, 99 162, 116 158, 115 165, 123 165, 122 170, 134 164, 131 158))',
  0.3);
```



Inner hull of a Polygon

```
SELECT ST_SimplifyPolygonHull(
  'POLYGON ((131 158, 136 163, 161 165, 173 156, 179 148, 169 140, 186 144, 190 137, 185 ↵
    131, 174 128, 174 124, 166 119, 158 121, 158 115, 165 107, 161 97, 166 88, 166 79, 158 ↵
    57, 145 57, 112 53, 111 47, 93 43, 90 48, 88 40, 80 39, 68 32, 51 33, 40 31, 39 34, ↵
    49 38, 34 38, 25 34, 28 39, 36 40, 44 46, 24 41, 17 41, 14 46, 19 50, 33 54, 21 55, 13 ↵
    52, 11 57, 22 60, 34 59, 41 68, 75 72, 62 77, 56 70, 46 72, 31 69, 46 76, 52 82, 47 ↵
    84, 56 90, 66 90, 64 94, 56 91, 33 97, 36 100, 23 100, 22 107, 29 106, 31 112, 46 116, ↵
    36 118, 28 131, 53 132, 59 127, 62 131, 76 130, 80 135, 89 137, 87 143, 73 145, 80 ↵
    150, 88 150, 85 157, 99 162, 116 158, 115 165, 123 165, 122 170, 134 164, 131 158))',
  0.3, false);
```



Outer hull simplification of a MultiPolygon, with segmentization

```
SELECT ST_SimplifyPolygonHull(
    ST_Segmentize(ST_Letters('xt'), 2.0),
    0.1);
```

స고

ST_ConvexHull, ST_SimplifyVW, ST_ConcaveHull, ST_Segmentize

8.14.24 ST_SimplifyVW

ST_SimplifyVW — Returns a simplified version of a geometry, using the Visvalingam-Whyatt algorithm

Synopsis

```
geometry ST_SimplifyVW(geometry geomA, float tolerance);
```

설&#xba85:

乄ʤ베일링검-와이어트(Visvalingam-Whyatt)알이용해서입력도형의"단순화"된버전을반환합니다. 실제로는 [멀멀티]폴리곤과만작동하지만,어종류의도형도입력할수있다고해도과언은아닙니다. 객체별기단순화작업을하기때문에이함도형집합도입력할수있습니다.



Note

```
&#xb18;&#xd658;&#xb418;&#xb294;      &#xb3c4;&#xd615;&#xc774;      &#xb2e8;&#xc21c;&#xc131;&#xc744;
&#xc783;&#xc744;      &#xc218;&#xb3c4;      &#xc788;&#xb2e4;&#xb294;      &#xc810;&#xc5d0;
&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;(ST IsSimple &#xcc38;&#xc870;).
```

Note

`ST_SimplifyPreserveTopology` simplifies a geometry by removing vertices that are not on the boundary of the minimum bounding box of the geometry. The function takes a geometry and a tolerance value as input. The tolerance value is the maximum distance between a vertex and the boundary of the minimum bounding box. Vertices that are within this distance are removed. The function returns a simplified geometry.

Note

`ST_Simplify` simplifies a geometry by removing vertices that are not on the boundary of the minimum bounding box of the geometry. The function takes a geometry and a tolerance value as input. The tolerance value is the maximum distance between a vertex and the boundary of the minimum bounding box. Vertices that are within this distance are removed. The function returns a simplified geometry.

2.2.0 `ST_Simplify` and `ST_SimplifyPreserveTopology` functions.

`ST_Simplify`

`ST_Simplify` simplifies a geometry by removing vertices that are not on the boundary of the minimum bounding box of the geometry. The function takes a geometry and a tolerance value as input. The tolerance value is the maximum distance between a vertex and the boundary of the minimum bounding box. Vertices that are within this distance are removed. The function returns a simplified geometry.

```
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)
```

`ST_SetEffectiveArea`

`ST_SetEffectiveArea`, `ST_Simplify`, `ST_SimplifyPreserveTopology`, `ST_Simplify`

8.14.25 ST_SetEffectiveArea

`ST_SetEffectiveArea` — Sets the effective area for each vertex, using the Visvalingam-Whyatt algorithm.

Synopsis

geometry `ST_SetEffectiveArea`(geometry geomA, float threshold = 0, integer set_area = 1);

`ST_SetEffectiveArea`

`ST_SetEffectiveArea` sets the effective area for each vertex, using the Visvalingam-Whyatt algorithm. The function takes a geometry and a tolerance value as input. The tolerance value is the maximum distance between a vertex and the boundary of the minimum bounding box. Vertices that are within this distance are removed. The function returns a simplified geometry.

`ST_SetEffectiveArea` simplifies a geometry by removing vertices that are not on the boundary of the minimum bounding box of the geometry. The function takes a geometry and a tolerance value as input. The tolerance value is the maximum distance between a vertex and the boundary of the minimum bounding box. Vertices that are within this distance are removed. The function returns a simplified geometry.

있습니다. 또다른 옵션은 임계치산으로 설정하는 것입니다. 이럴 경유효 범위를 M값으로 가진 전체 도반환하는데. 이 도형을 클라이언매우 빨리 단순화하는 데 쓸 수 있실제로는 [멀티]라인. [멀티]폴리곤작동하지만. 어떤 종류의 도형도 입력할 수 있다고 해도 과언은 아객체&#bcc4; 기반으로 단순화 작업을 하기 때문에 이 함수에 도형 집합입력할 수 있습니다.

**Note**

도형이 단순성을 잃을 수도 있다는 점에 주의하십시오(**ST_IsSimple** 참조).

**Note**

위상(topology)이 보전되지 않아 유효하지 않은 도형이 도형이 도형이 수도 있습니다. 위상을 유지하려면 **ST_SimplifyPreserveTopology** 함수를 이용하십시오.

**Note**

출력 도형은 M값으로 가지고 있던 정보를 모두 잃게 될 것입니다.

**Note**

이 함수는 3차원을 처리하며 세 뎈째 차원이 유효 범위에 영향을 미칠 것입니다.

2.2.0 도전부터 사용할 수 있습니다.

예시

라인스트링의 유효 범위를 계산임계치를 0으로 설정하기 때문에 입력 도형 안에 있는 모든 꼭짓점도환합니다.

```
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;
-- &#xacb0;&#xacfc; --
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)
```

ST_SimplifyVW

ST_SimplifyVW

8.14.26 ST_TriangulatePolygon

ST_TriangulatePolygon — Computes the constrained Delaunay triangulation of polygons

Synopsis

geometry **ST_TriangulatePolygon**(geometry geom);

ST_TriangulatePolygon

Computes the constrained Delaunay triangulation of polygons. Holes and Multipolygons are supported.

The "constrained Delaunay triangulation" of a polygon is a set of triangles formed from the vertices of the polygon, and covering it exactly, with the maximum total interior angle over all possible triangulations. It provides the "best quality" triangulation of the polygon.

Availability: 3.3.0 - requires GEOS >= 3.11.0

Example

Triangulation of a square.

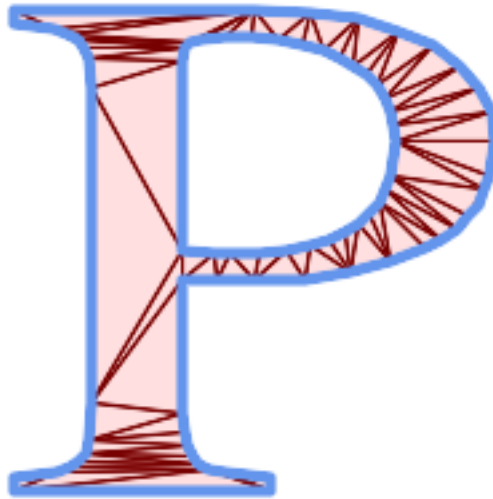
```
SELECT ST_AsText(
  ST_TriangulatePolygon('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))'));

          st_astext
-----
GEOMETRYCOLLECTION(POLYGON((0 0,0 1,1 1,0 0)),POLYGON((1 1,1 0,0 0,1 1)))
```

Example

Triangulation of the letter P.

```
SELECT ST_AsText(ST_TriangulatePolygon(
  'POLYGON ((26 17, 31 19, 34 21, 37 24, 38 29, 39 43, 39 161, 38 172, 36 176, 34 179, 30 ↵
    181, 25 183, 10 185, 10 190, 100 190, 121 189, 139 187, 154 182, 167 177, 177 169, ↵
    184 161, 189 152, 190 141, 188 128, 186 123, 184 117, 180 113, 176 108, 170 104, 164 ↵
    101, 151 96, 136 92, 119 89, 100 89, 86 89, 73 89, 73 39, 74 32, 75 27, 77 23, 79 ↵
    20, 83 18, 89 17, 106 15, 106 10, 10 10, 10 15, 26 17), (152 147, 151 152, 149 157, ↵
    146 162, 142 166, 137 169, 132 172, 126 175, 118 177, 109 179, 99 180, 89 180, 80 ↵
    179, 76 178, 74 176, 73 171, 73 100, 85 99, 91 99, 102 99, 112 100, 121 102, 128 ↵
    104, 134 107, 139 110, 143 114, 147 118, 149 123, 151 128, 153 141, 152 147)))'
));
```



Polygon Triangulation

ST_DelaunayTriangles

ST_ConstrainedDelaunayTriangles, **ST_Tessellate**

8.14.27 ST_VoronoiLines

ST_VoronoiLines — Returns the boundaries of the Voronoi diagram of the vertices of a geometry.

Synopsis

geometry **ST_VoronoiLines**(g1 geometry , tolerance float8 , extend_to geometry);

ST_VoronoiLines

ST_VoronoiLines computes a two-dimensional **Voronoi diagram** from the vertices of the supplied geometry and returns the boundaries between cells in that diagram as a MultiLineString. Returns null if input geometry is null. Returns an empty geometry collection if the input geometry contains only one vertex. Returns an empty geometry collection if the extend_to envelope has zero area.

ST_VoronoiLines(geometry, tolerance, extend_to)

- 'tolerance' : The tolerance value is used to snap vertices that are close to each other. The default value is 0.0. The tolerance value is used to snap vertices that are close to each other. The default value is 0.0. The tolerance value is used to snap vertices that are close to each other. The default value is 0.0.
- 'extend_to' : If a geometry is supplied as the "extend_to" parameter, the diagram will be extended to cover the envelope of the "extend_to" geometry, unless that envelope is smaller than the default envelope (default = NULL, default envelope is boundingbox of input geometry extended by about 50% in each direction).

ST_VoronoiLines(geometry, tolerance, extend_to)

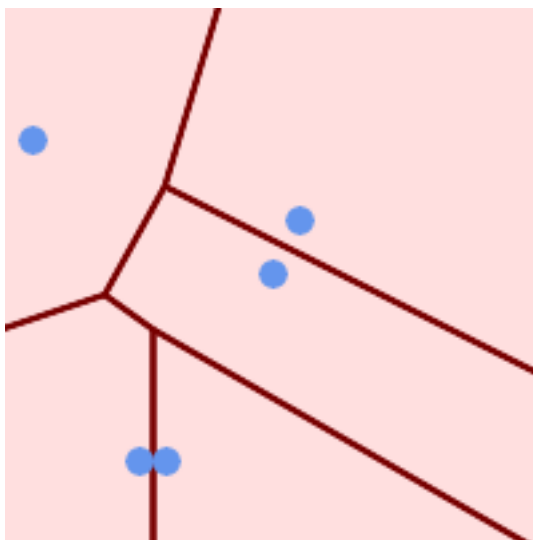
2.3.0 **ST_VoronoiLines**(geometry, tolerance, extend_to)

ST_VoronoiLines

- 'extend_to' : If a geometry is supplied as the "extend_to" parameter, the diagram will be extended to cover the envelope of the "extend_to" geometry, unless that envelope is smaller than the default envelope (default = NULL, default envelope is boundingbox of input geometry extended by about 50% in each direction).

GEOS

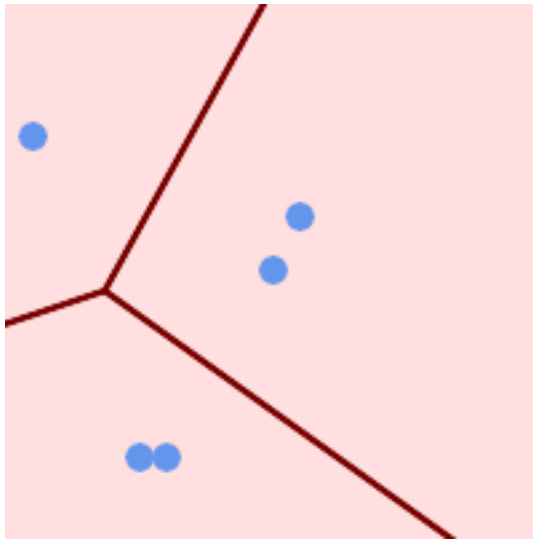
2.3.0



Points overlaid on top of Voronoi diagram

```
SELECT
  ST_VoronoiPolygons(geom) As geom
FROM (SELECT 'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120) '::geometry As geom )
      As g;

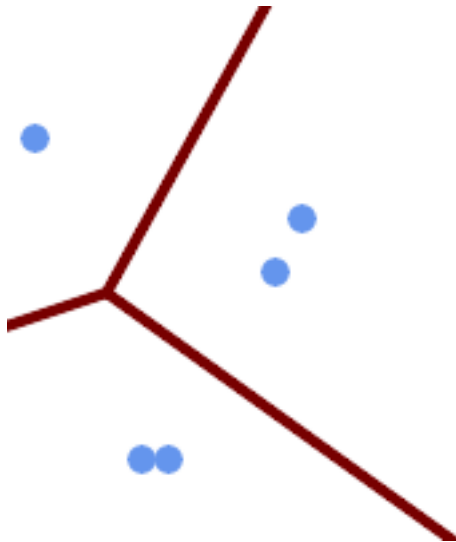
-- ST_AsText
GEOMETRYCOLLECTION(POLYGON((-110 43.3333333333333,-110 270,100.5 270,59.3478260869565
132.826086956522,36.8181818181818 92.2727272727273,-110 43.3333333333333)),
POLYGON((55 -90,-110 -90,-110 43.3333333333333,36.8181818181818 92.2727272727273,55
79.2857142857143,55 -90)),
POLYGON((230 47.5,230 -20.7142857142857,55 79.2857142857143,36.8181818181818
92.2727272727273,59.3478260869565 132.826086956522,230 47.5)),POLYGON((230
-20.7142857142857,230 -90,55 -90,55 79.2857142857143,230 -20.7142857142857)),
POLYGON((100.5 270,230 270,230 47.5,59.3478260869565 132.826086956522,100.5 270)))
```



허용 오차 거리가 30 단위인
보로노이 다이어그램

```
SELECT ST_VoronoiPolygons(geom, 30) As geom
FROM (SELECT 'MULTIPOINT (50 30, 60 30, 100 100,10 150, 110 120) '::geometry As geom ) ↵
      As g;

-- ST_AsText \&\#xcd9c;\&\#xb825;\&\#xbb3c;
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 with tolerance of 30 units as MultiLineString

```
SELECT ST_VoronoiLines(geom, 30) As geom
FROM (SELECT 'MULTIPOINT (50 30, 60 30, 100 100, 10 150, 110 120)'::geometry As geom ) ↔
      As g

-- ST_AsText '\\#xcd9c;\\#xb825;\\#xbb3c;
MULTILINESTRING((135.555555555556 270,36.8181818181818 92.2727272727273) ↔
, (36.8181818181818 92.2727272727273,-110 43.3333333333333), (230 ↔
-45.7142857142858,36.8181818181818 92.2727272727273))
```

󌰸고

[ST_DelaunayTriangles](#), [ST_VoronoiLines](#), [ST_GeomCollFromText](#)

8.15 Affine Transformations

8.15.1 ST_Affine

ST_Affine — Apply a 3D affine transformation to a geometry.

Synopsis

geometry **ST_Affine**(geometry geomA, float a, float b, float c, float d, float e, float f, float g, float h, float i, float xoff, float yoff, float zoff);
 geometry **ST_Affine**(geometry geomA, float a, float b, float d, float e, float xoff, float yoff);

Description

Applies a 3D affine transformation to the geometry to do things like translate, rotate, scale in one step.

Version 1: The call

```
ST_Affine(geom, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff)
```

represents the transformation matrix

```

/ a  b  c  xoff \
| d  e  f  yoff |
| g  h  i  zoff |
\ 0  0  0    1 /

```

and the vertices are transformed as follows:

```

x' = a*x + b*y + c*z + xoff
y' = d*x + e*y + f*z + yoff
z' = g*x + h*y + i*z + zoff

```

All of the translate / scale functions below are expressed via such an affine transformation.

Version 2: Applies a 2d affine transformation to the geometry. The call

```
ST_Affine(geom, a, b, d, e, xoff, yoff)
```

represents the transformation matrix

```

/ a  b  0  xoff \      / a  b  xoff \
| d  e  0  yoff |  rsp. | d  e  yoff |
| 0  0  1    0 |      \ 0  0    1  /
\ 0  0  0    1 /

```

and the vertices are transformed as follows:

```

x' = a*x + b*y + xoff
y' = d*x + e*y + yoff
z' = z

```

This method is a subcase of the 3D method above.

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.

Availability: 1.1.2. Name changed from Affine to ST_Affine in 1.2.2



Note

Prior to 1.3.4, this function crashes if used with geometries that contain CURVES. This is fixed in 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

Examples

```

--Rotate a 3d line 180 degrees about the z axis. Note this is long-hand for doing ↔
ST_Rotate();
SELECT ST_AseWKT(ST_Affine(geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), 0, 0, ↔
0, 1, 0, 0, 0)) As using_affine,
       ST_AseWKT(ST_Rotate(geom, pi())) As using_rotate
FROM (SELECT ST_GeomFromEWKT('LINESTRING(1 2 3, 1 4 3)') As geom) As foo;
using_affine      |      using_rotate

```

```

-----+-----
LINESTRING(-1 -2 3,-1 -4 3) | LINESTRING(-1 -2 3,-1 -4 3)
(1 row)

--Rotate a 3d line 180 degrees in both the x and z axis
SELECT ST_AsEWKT(ST_Affine(geom, cos(pi()), -sin(pi()), 0, sin(pi()), cos(pi()), -sin(pi()) ←
, 0, sin(pi()), cos(pi()), 0, 0, 0))
      FROM (SELECT ST_GeomFromEWKT('LINESTRING(1 2 3, 1 4 3)') As geom) As foo;
      st_asewkt
-----
LINESTRING(-1 -2 -3,-1 -4 -3)
(1 row)

```

See Also

[ST_Rotate](#), [ST_Scale](#), [ST_Translate](#), [ST_TransScale](#)

8.15.2 ST_Rotate

ST_Rotate — Rotates a geometry about an origin point.

Synopsis

```

geometry ST_Rotate(geometry geomA, float rotRadians);
geometry ST_Rotate(geometry geomA, float rotRadians, float x0, float y0);
geometry ST_Rotate(geometry geomA, float rotRadians, geometry pointOrigin);

```

Description

Rotates geometry rotRadians counter-clockwise about the origin point. The rotation origin can be specified either as a POINT geometry, or as x and y coordinates. If the origin is not specified, the geometry is rotated about POINT(0 0).

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.

Enhanced: 2.0.0 additional parameters for specifying the origin of rotation were added.

Availability: 1.1.2. Name changed from Rotate to ST_Rotate in 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).

Examples

```

--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)
```

See Also

[ST_Affine](#), [ST_RotateX](#), [ST_RotateY](#), [ST_RotateZ](#)

8.15.3 ST_RotateX

ST_RotateX — Rotates a geometry about the X axis.

Synopsis

geometry **ST_RotateX**(geometry geomA, float rotRadians);

Description

Rotates a geometry geomA - rotRadians about the X axis.

**Note**

ST_RotateX(geomA, rotRadians) is short-hand for ST_Affine(geomA, 1, 0, 0, 0, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0).

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.

Availability: 1.1.2. Name changed from RotateX to ST_RotateX in 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).

Examples

```
--Rotate a line 90 degrees along x-axis
SELECT ST_AsEWKT(ST_RotateX(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
           st_asewkt
-----
LINESTRING(1 -3 2,1 -1 1)
```

See Also[ST_Affine](#), [ST_RotateY](#), [ST_RotateZ](#)**8.15.4 ST_RotateY**

ST_RotateY — Rotates a geometry about the Y axis.

Synopsis

geometry **ST_RotateY**(geometry geomA, float rotRadians);

Description

Rotates a geometry geomA - rotRadians about the y axis.

**Note**

ST_RotateY(geomA, rotRadians) is short-hand for ST_Affine(geomA, cos(rotRadians), 0, sin(rotRadians), 0, 1, 0, -sin(rotRadians), 0, cos(rotRadians), 0, 0, 0).

Availability: 1.1.2. Name changed from RotateY to ST_RotateY in 1.2.2

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.



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).

Examples

```
--Rotate a line 90 degrees along y-axis
SELECT ST_AsEWKT(ST_RotateY(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
           st_asewkt
-----
LINESTRING(3 2 -1,1 1 -1)
```

See Also[ST_Affine](#), [ST_RotateX](#), [ST_RotateZ](#)**8.15.5 ST_RotateZ**

ST_RotateZ — Rotates a geometry about the Z axis.

Synopsis

geometry **ST_RotateZ**(geometry geomA, float rotRadians);

Description

Rotates a geometry geomA - rotRadians about the Z axis.



Note

This is a synonym for ST_Rotate



Note

ST_RotateZ(geomA, rotRadians) is short-hand for SELECT ST_Affine(geomA, cos(rotRadians), -sin(rotRadians), 0, sin(rotRadians), cos(rotRadians), 0, 0, 0, 1, 0, 0, 0).

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.

Availability: 1.1.2. Name changed from RotateZ to ST_RotateZ in 1.2.2



Note

Prior to 1.3.4, this function crashes if used with geometries that contain CURVES. This is fixed in 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).

Examples

```
--Rotate a line 90 degrees along z-axis
SELECT ST_AsEWKT(ST_RotateZ(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), pi()/2));
           st_asewkt
-----
LINESTRING(-2 1 3,-1 1 1)

--Rotate a curved circle around z-axis
SELECT ST_AsEWKT(ST_RotateZ(geom, pi()/2))
FROM (SELECT ST_LineToCurve(ST_Buffer(ST_GeomFromText('POINT(234 567)'), 3)) As geom) As
      foo;

CURVEPOLYGON(CIRCULARSTRING(-567 237,-564.87867965644 236.12132034356,-564
234,-569.12132034356 231.87867965644,-567 237))
```


See Also

[ST_Affine](#), [ST_RotateX](#), [ST_RotateY](#)

8.15.6 ST_Scale

ST_Scale — Scales a geometry by given factors.

Synopsis

```
geometry ST_Scale(geometry geomA, float XFactor, float YFactor, float ZFactor);
geometry ST_Scale(geometry geomA, float XFactor, float YFactor);
geometry ST_Scale(geometry geom, geometry factor);
geometry ST_Scale(geometry geom, geometry factor, geometry origin);
```

Description

Scales the geometry to a new size by multiplying the ordinates with the corresponding factor parameters.

The version taking a geometry as the `factor` parameter allows passing a 2d, 3dm, 3dz or 4d point to set scaling factor for all supported dimensions. Missing dimensions in the `factor` point are equivalent to no scaling the corresponding dimension.

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**

Prior to 1.3.4, this function crashes if used with geometries that contain CURVES. This is fixed in 1.3.4+

Availability: 1.1.0.

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.

Enhanced: 2.2.0 support for scaling all dimension (`factor` parameter) was introduced.

Enhanced: 2.5.0 support for scaling relative to a local origin (`origin` parameter) was introduced.



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.

Examples

```

--Version 1: scale X, Y, Z
SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 0.75, 0.8));
          st_asewkt
-----
LINESTRING(0.5 1.5 2.4,0.5 0.75 0.8)

--Version 2: Scale X Y
SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 0.75));
          st_asewkt
-----
LINESTRING(0.5 1.5 3,0.5 0.75 1)

--Version 3: Scale X Y Z M
SELECT ST_AsEWKT(ST_Scale(ST_GeomFromEWKT('LINESTRING(1 2 3 4, 1 1 1 1)'),
          ST_MakePoint(0.5, 0.75, 2, -1)));
          st_asewkt
-----
LINESTRING(0.5 1.5 6 -4,0.5 0.75 2 -1)

--Version 4: Scale X Y using false origin
SELECT ST_AsText(ST_Scale('LINESTRING(1 1, 2 2)', 'POINT(2 2)', 'POINT(1 1)::geometry'));
          st_astext
-----
LINESTRING(1 1,3 3)

```

See Also[ST_Affine](#), [ST_TransScale](#)**8.15.7 ST_Translate****ST_Translate** — Translates a geometry by given offsets.**Synopsis**

```

geometry ST_Translate(geometry g1, float deltax, float deltax);
geometry ST_Translate(geometry g1, float deltax, float deltax, float deltaz);

```

Description

Returns a new geometry whose coordinates are translated delta x,delta y,delta z units. Units are based on the units defined in spatial reference (SRID) for this geometry.

**Note**

Prior to 1.3.4, this function crashes if used with geometries that contain CURVES. This is fixed in 1.3.4+

Availability: 1.2.2



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

Move a point 1 degree longitude

```
SELECT ST_AsText(ST_Translate(ST_GeomFromText('POINT(-71.01 42.37)',4326),1,0)) As ↵
    wgs_transgeomtxt;

    wgs_transgeomtxt
    -----
    POINT(-70.01 42.37)
```

Move a linestring 1 degree longitude and 1/2 degree latitude

```
SELECT ST_AsText(ST_Translate(ST_GeomFromText('LINESTRING(-71.01 42.37,-71.11 42.38)',4326) ↵
    ,1,0.5)) As wgs_transgeomtxt;
           wgs_transgeomtxt
           -----
    LINESTRING(-70.01 42.87,-70.11 42.88)
```

Move a 3d point

```
SELECT ST_AsEWKT(ST_Translate(CAST('POINT(0 0 0)' As geometry), 5, 12,3));
    st_asewkt
    -----
    POINT(5 12 3)
```

Move a curve and a point

```
SELECT ST_AsText(ST_Translate(ST_Collect('CURVEPOLYGON(CIRCULARSTRING(4 3,3.12 0.878,1 ↵
    0,-1.121 5.1213,6 7, 8 9,4 3))','POINT(1 3)'),1,2));

-----

GEOMETRYCOLLECTION(CURVEPOLYGON(CIRCULARSTRING(5 5,4.12 2.878,2 2,-0.121 7.1213,7 9,9 11,5 ↵
    5)),POINT(2 5))
```

See Also

[ST_Affine](#), [ST_AsText](#), [ST_GeomFromText](#)

8.15.8 ST_TransScale

ST_TransScale — Translates and scales a geometry by given offsets and factors.

Synopsis

geometry **ST_TransScale**(geometry geomA, float deltaX, float deltaY, float XFactor, float YFactor);

Description

Translates the geometry using the deltaX and deltaY args, then scales it using the XFactor, YFactor args, working in 2D only.



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**

Prior to 1.3.4, this function crashes if used with geometries that contain CURVES. This is fixed in 1.3.4+

Availability: 1.1.0.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_AsEWKT(ST_TransScale(ST_GeomFromEWKT('LINESTRING(1 2 3, 1 1 1)'), 0.5, 1, 1, 2));
           st_asewkt
-----
LINESTRING(1.5 6 3,1.5 4 1)

--Buffer a point to get an approximation of a circle, convert to curve and then translate ↔
  1,2 and scale it 3,4
SELECT ST_AsText(ST_Transscale(ST_LineToCurve(ST_Buffer('POINT(234 567)', 3)),1,2,3,4));

-----

CURVEPOLYGON(CIRCULARSTRING(714 2276,711.363961030679 2267.51471862576,705 ↔
  2264,698.636038969321 2284.48528137424,714 2276))
```

See Also

[ST_Affine](#), [ST_Translate](#)

8.16 Clustering Functions

8.16.1 ST_ClusterDBSCAN

ST_ClusterDBSCAN — Window function that returns a cluster id for each input geometry using the DBSCAN algorithm.

Synopsis

integer **ST_ClusterDBSCAN**(geometry winset geom, float8 eps, integer minpoints);

Description

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.

An input geometry will be added to a cluster if it is either:

- 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 that border geometries may be within `eps` distance of core geometries in more than one cluster; in this case, either assignment would be correct, and the border geometry will be arbitrarily assigned to one of the available clusters. In these cases, it is possible for a correct cluster to be generated with fewer than `minpoints` geometries. When assignment of a border geometry is ambiguous, repeated calls to `ST_ClusterDBSCAN` will produce identical results if an `ORDER BY` clause is included in the window definition, but cluster assignments may differ from other implementations of the same algorithm.

**Note**

Input geometries that do not meet the criteria to join any other cluster will be assigned a cluster number of `NULL`.

Availability: 2.3.0

Examples

Assigning a cluster number to each polygon within 50 meters of each other. Require at least 2 polygons per cluster

8.16.2 ST_ClusterIntersecting

ST_ClusterIntersecting — Aggregate function that clusters the input geometries into connected sets.

Synopsis

```
geometry[] ST_ClusterIntersecting(geometry set g);
```

Description

ST_ClusterIntersecting is an aggregate function that returns an array of GeometryCollections, where each GeometryCollection represents an interconnected set of geometries.

Availability: 2.2.0

Examples

```
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))
```

See Also

[ST_ClusterDBSCAN](#), [ST_ClusterKMeans](#), [ST_ClusterWithin](#)

8.16.3 ST_ClusterKMeans

ST_ClusterKMeans — Window function that returns a cluster id for each input geometry using the K-means algorithm.

Synopsis

```
integer ST_ClusterKMeans(geometry winset geom, integer number_of_clusters, float max_radius);
```

Description

Returns **K-means** cluster number for each input geometry. The distance used for clustering is the distance between the centroids for 2D geometries, and distance between bounding box centers for 3D geometries. For POINT inputs, M coordinate will be treated as weight of input and has to be larger than 0.

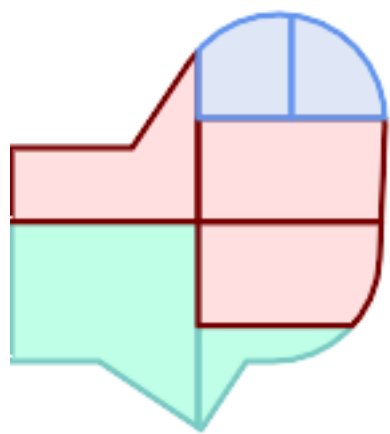
`max_radius`, if set, will cause **ST_ClusterKMeans** to generate more clusters than `k` ensuring that no cluster in output has radius larger than `max_radius`. This is useful in reachability analysis.

Enhanced: 3.2.0 Support for `max_radius`
Enhanced: 3.1.0 Support for 3D geometries and weights
Availability: 2.3.0

Examples

Generate dummy set of parcels for examples:

```
CREATE TABLE parcels AS
SELECT lpad((row_number() over())::text,3,'0') As parcel_id, geom,
('{residential, commercial}'::text[])[1 + mod(row_number()OVER(),2)] As type
FROM
  ST_Subdivide(ST_Buffer('SRID=3857;LINESTRING(40 100, 98 100, 100 150, 60 90)'::geometry ←
    40, 'endcap=square'),12) As geom;
```



Parcels color-coded by cluster number (cid)

```
SELECT ST_ClusterKMeans(geom, 3) OVER() AS cid, parcel_id, geom
FROM parcels;
```

cid	parcel_id	geom
0	001	0103000000...
0	002	0103000000...
1	003	0103000000...
0	004	0103000000...
1	005	0103000000...
2	006	0103000000...
2	007	0103000000...

Partitioning parcel clusters by type:

```
SELECT ST_ClusterKMeans(geom, 3) over (PARTITION BY type) AS cid, parcel_id, type
FROM parcels;
```


cid	parcel_id	type
1	005	commercial
1	003	commercial
2	007	commercial
0	001	commercial
1	004	residential
0	002	residential
2	006	residential

Example: Clustering a preaggregated planetary-scale data population dataset using 3D clustering and weighting. Identify at least 20 regions based on **Kontur Population Data** that do not span more than 3000 km from their center:

```
create table kontur_population_3000km_clusters as
select
  geom,
  ST_ClusterKMeans(
    ST_Force4D(
      ST_Transform(ST_Force3D(geom), 4978), -- cluster in 3D XYZ CRS
      mvalue := population -- set clustering to be weighed by population
    ),
    20, -- aim to generate at least 20 clusters
    max_radius := 3000000 -- but generate more to make each under 3000 km radius
  ) over () as cid
from
  kontur_population;
```



World population clustered to above specs produces 46 clusters. Clusters are centered at well-populated regions (New York, Moscow). Greenland is one cluster. There are island clusters that span across the antimeridian. Cluster edges follow Earth's curvature.

See Also

[ST_ClusterDBSCAN](#), [ST_ClusterIntersecting](#), [ST_ClusterWithin](#), [ST_Subdivide](#), [ST_Force3D](#), [ST_Force4D](#),

8.16.4 ST_ClusterWithin

ST_ClusterWithin — Aggregate function that clusters the input geometries by separation distance.

Synopsis

```
geometry[] ST_ClusterWithin(geometry set g, float8 distance);
```

Description

`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.)

Availability: 2.2.0

Examples

```
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))
```

See Also

[ST_ClusterDBSCAN](#), [ST_ClusterKMeans](#), [ST_ClusterIntersecting](#)

8.17 Bounding Box Functions

8.17.1 Box2D

`Box2D` — Returns a `BOX2D` representing the 2D extent of a geometry.

Synopsis

`box2d` **Box2D**(geometry geom);

Description

Returns a **box2d** representing the 2D extent of the geometry.

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.



This method supports Circular Strings and Curves



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Examples

```
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)
```

See Also

[Box3D](#), [ST_GeomFromText](#)

8.17.2 Box3D

Box3D — Returns a BOX3D representing the 3D extent of a geometry.

Synopsis

```
box3d Box3D(geometry geom);
```

Description

Returns a [box3d](#) representing the 3D extent of the geometry.

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.



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.

Examples

```
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)
```

See Also[Box2D](#), [ST_GeomFromEWKT](#)**8.17.3 ST_EstimatedExtent**

ST_EstimatedExtent — Returns the estimated extent of a spatial table.

Synopsis

```
box2d ST_EstimatedExtent(text schema_name, text table_name, text geocolumn_name, boolean parent_only);
box2d ST_EstimatedExtent(text schema_name, text table_name, text geocolumn_name);
box2d ST_EstimatedExtent(text table_name, text geocolumn_name);
```

Description

Returns the estimated extent of a spatial table as a [box2d](#). The current schema is used if not specified. The estimated extent is taken from the geometry column's statistics. This is usually much faster than computing the exact extent of the table using [ST_Extent](#) or [ST_3DExtent](#).

The default behavior is to also use statistics collected from child tables (tables with INHERITS) if available. If `parent_only` is set to TRUE, only statistics for the given table are used and child tables are ignored.

For PostgreSQL >= 8.0.0 statistics are gathered by VACUUM ANALYZE and the result extent will be about 95% of the actual one. For PostgreSQL < 8.0.0 statistics are gathered by running `update_geometry_stats()` and the result extent is exact.

**Note**

In the absence of statistics (empty table or no ANALYZE called) this function returns NULL. Prior to version 1.5.4 an exception was thrown instead.

Availability: 1.0.0

Changed: 2.1.0. Up to 2.0.x this was called `ST_Estimated_Extent`.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_EstimatedExtent('ny', 'edges', 'geom');
--result--
BOX(-8877653 4912316,-8010225.5 5589284)

SELECT ST_EstimatedExtent('feature_poly', 'geom');
--result--
BOX(-124.659652709961 24.6830825805664,-67.7798080444336 49.0012092590332)
```

See Also[ST_Extent](#), [ST_3DExtent](#)**8.17.4 ST_Expand**

ST_Expand — Returns a bounding box expanded from another bounding box or a geometry.

Synopsis

```
geometry ST_Expand(geometry geom, float units_to_expand);
geometry ST_Expand(geometry geom, float dx, float dy, float dz=0, float dm=0);
box2d ST_Expand(box2d box, float units_to_expand);
box2d ST_Expand(box2d box, float dx, float dy);
box3d ST_Expand(box3d box, float units_to_expand);
box3d ST_Expand(box3d box, float dx, float dy, float dz=0);
```

Description

Returns a bounding box expanded from the bounding box of the input, either by specifying a single distance with which the box should be expanded on both axes, or by specifying an expansion distance for each axis. Uses double-precision. Can be used for distance queries, or to add a bounding box filter to a query to take advantage of a spatial index.

In addition to the version of `ST_Expand` accepting and returning a geometry, variants are provided that accept and return **box2d** and **box3d** data types.

Distances are in the units of the spatial reference system of the input.

`ST_Expand` is similar to **ST_Buffer**, except while buffering expands a geometry in all directions, `ST_Expand` expands the bounding box along each axis.



Note

Pre version 1.3, `ST_Expand` was used in conjunction with **ST_Distance** to do indexable distance queries. For example, `geom && ST_Expand('POINT(10 20)', 10) AND ST_Distance(geom, 'POINT(10 20)') < 10`. This has been replaced by the simpler and more efficient **ST_DWithin** function.

Availability: 1.5.0 behavior changed to output double precision instead of float4 coordinates.

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.

Enhanced: 2.3.0 support was added to expand a box by different amounts in different dimensions.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Examples



Note

Examples below use US National Atlas Equal Area (SRID=2163) which is a meter projection

```
--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));
--
SRID=2163;POLYGON((2312970 110666,2312970 110686,2312990 110686,2312990 110666,2312970 110666))

```

See Also

[ST_Buffer](#), [ST_DWithin](#), [ST_SRID](#)

8.17.5 ST_Extent

ST_Extent — Aggregate function that returns the bounding box of geometries.

Synopsis

`box2d ST_Extent(geometry set geomfield);`

Description

An aggregate function that returns a **box2d** bounding box that bounds a set of geometries.

The bounding box coordinates are in the spatial reference system of the input geometries.

ST_Extent is similar in concept to Oracle Spatial/Locator's `SDO_AGGR_MBR`.



Note

ST_Extent returns boxes with only X and Y ordinates even with 3D geometries. To return XYZ ordinates use **ST_3DExtent**.



Note

The returned `box3d` value does not include a SRID. Use **ST_SetSRID** to convert it into a geometry with SRID meta-data. The SRID is the same as the input geometries.

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.



This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Examples



Note
Examples below use Massachusetts State Plane ft (SRID=2249)

```
SELECT ST_Extent(geom) as bextent FROM sometable;
                                st_bextent
-----
BOX(739651.875 2908247.25,794875.8125 2970042.75)

--Return extent of each category of geometries
SELECT ST_Extent(geom) as bextent
FROM sometable
GROUP BY category ORDER BY category;
```

bextent	name
BOX(778783.5625 2951741.25,794875.8125 2970042.75)	A
BOX(751315.8125 2919164.75,765202.6875 2935417.25)	B
BOX(739651.875 2917394.75,756688.375 2935866)	C

```
--Force back into a geometry
-- and render the extended text representation of that geometry
SELECT ST_SetSRID(ST_Extent(geom),2249) as bextent FROM sometable;
```

bextent
SRID=2249;POLYGON((739651.875 2908247.25,739651.875 2970042.75,794875.8125 2970042.75,794875.8125 2908247.25,739651.875 2908247.25))

See Also

[ST_EstimatedExtent](#), [ST_3DExtent](#), [ST_SetSRID](#)

8.17.6 ST_3DExtent

ST_3DExtent — Aggregate function that returns the 3D bounding box of geometries.

Synopsis

box3d **ST_3DExtent**(geometry set geomfield);

Description

An aggregate function that returns a **box3d** (includes Z ordinate) bounding box that bounds a set of geometries. The bounding box coordinates are in the spatial reference system of the input geometries.



Note
The returned **box3d** value does not include a SRID. Use **ST_SetSRID** to convert it into a geometry with SRID meta-data. The SRID is the same as the input geometries.

Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.

Changed: 2.0.0 In prior versions this used to be called 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).

Examples

```
SELECT ST_3DExtent(foo.geom) As b3extent
FROM (SELECT ST_MakePoint(x,y,z) As geom
      FROM generate_series(1,3) As x
           CROSS JOIN generate_series(1,2) As y
           CROSS JOIN generate_series(0,2) As Z) As foo;

-----
BOX3D(1 1 0,3 2 2)

--Get the extent of various elevated circular strings
SELECT ST_3DExtent(foo.geom) As b3extent
FROM (SELECT ST_Translate(ST_Force_3DZ(ST_LineToCurve(ST_Buffer(ST_Point(x,y),1))),0,0,z)  ←
      As geom
      FROM generate_series(1,3) As x
           CROSS JOIN generate_series(1,2) As y
           CROSS JOIN generate_series(0,2) As Z) As foo;

-----
BOX3D(1 0 0,4 2 2)
```

See Also

[ST_Extent](#), [ST_Force3DZ](#), [ST_SetSRID](#)

8.17.7 ST_MakeBox2D

ST_MakeBox2D — Creates a BOX2D defined by two 2D point geometries.

Synopsis

box2d **ST_MakeBox2D**(geometry pointLowLeft, geometry pointUpRight);

Description

Creates a **box2d** defined by two Point geometries. This is useful for doing range queries.

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, geom
FROM features
WHERE geom && ST_SetSRID(ST_MakeBox2D(ST_Point(-989502.1875, 528439.5625),
      ST_Point(-987121.375 ,529933.1875)),2163)
```

See Also

[ST_Point](#), [ST_SetSRID](#), [ST_SRID](#)

8.17.8 ST_3DMakeBox

ST_3DMakeBox — Creates a BOX3D defined by two 3D point geometries.

Synopsis

box3d **ST_3DMakeBox**(geometry point3DLowLeftBottom, geometry point3DUpRightTop);

Description

Creates a **box3d** defined by two 3D Point geometries.



This function supports 3D and will not drop the z-index.

Changed: 2.0.0 In prior versions this used to be called 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)
```

See Also

[ST_MakePoint](#), [ST_SetSRID](#), [ST_SRID](#)

8.17.9 ST_XMax

ST_XMax — Returns the X maxima of a 2D or 3D bounding box or a geometry.

Synopsis

float **ST_XMax**(box3d aGeomorBox2DorBox3D);

Description

Returns the X maxima of a 2D or 3D bounding box or a geometry.



Note

Although this function is only defined for box3d, it also works for box2d and geometry values due to automatic casting. However, it will not accept a geometry or box2d text representation, since those do not auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_XMax('BOX3D(1 2 3, 4 5 6)');
st_xmax
-----
4

SELECT ST_XMax(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_xmax
-----
5

SELECT ST_XMax(CAST('BOX(-3 2, 3 4)' As box2d));
st_xmax
-----
3
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to
a BOX3D
SELECT ST_XMax('LINESTRING(1 3, 5 6)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_XMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227
150406 3)'));
st_xmax
-----
220288.248780547
```

See Also

[ST_XMin](#), [ST_YMax](#), [ST_YMin](#), [ST_ZMax](#), [ST_ZMin](#)

8.17.10 ST_XMin

ST_XMin — Returns the X minima of a 2D or 3D bounding box or a geometry.

Synopsis

```
float ST_XMin(box3d aGeomorBox2DorBox3D);
```

Description

Returns the X minima of a 2D or 3D bounding box or a geometry.



Note

Although this function is only defined for box3d, it also works for box2d and geometry values due to automatic casting. However it will not accept a geometry or box2d text representation, since those do not auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_XMin('BOX3D(1 2 3, 4 5 6)');
st_xmin
-----
1

SELECT ST_XMin(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_xmin
-----
1

SELECT ST_XMin(CAST('BOX(-3 2, 3 4)' As box2d));
st_xmin
-----
-3
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to
a BOX3D
SELECT ST_XMin('LINESTRING(1 3, 5 6)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_XMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227
150406 3)'));
st_xmin
-----
220186.995121892
```

See Also

[ST_XMax](#), [ST_YMax](#), [ST_YMin](#), [ST_ZMax](#), [ST_ZMin](#)

8.17.11 ST_YMax

ST_YMax — Returns the Y maxima of a 2D or 3D bounding box or a geometry.

Synopsis

```
float ST_YMax(box3d aGeomorBox2DorBox3D);
```

Description

Returns the Y maxima of a 2D or 3D bounding box or a geometry.



Note

Although this function is only defined for box3d, it also works for box2d and geometry values due to automatic casting. However it will not accept a geometry or box2d text representation, since those do not auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_YMax('BOX3D(1 2 3, 4 5 6)');
st_ymax
-----
5

SELECT ST_YMax(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_ymax
-----
6

SELECT ST_YMax(CAST('BOX(-3 2, 3 4)' As box2d));
st_ymax
-----
4
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to
a BOX3D
SELECT ST_YMax('LINESTRING(1 3, 5 6)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_YMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227
150406 3)'));
st_ymax
-----
150506.126829327
```

See Also

[ST_XMin](#), [ST_XMax](#), [ST_YMin](#), [ST_ZMax](#), [ST_ZMin](#)

8.17.12 ST_YMin

ST_YMin — Returns the Y minima of a 2D or 3D bounding box or a geometry.

Synopsis

```
float ST_YMin(box3d aGeomorBox2DorBox3D);
```

Description

Returns the Y minima of a 2D or 3D bounding box or a geometry.



Note

Although this function is only defined for box3d, it also works for box2d and geometry values due to automatic casting. However it will not accept a geometry or box2d text representation, since those do not auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_YMin('BOX3D(1 2 3, 4 5 6)');
st_ymin
-----
2

SELECT ST_YMin(ST_GeomFromText('LINESTRING(1 3 4, 5 6 7)'));
st_ymin
-----
3

SELECT ST_YMin(CAST('BOX(-3 2, 3 4)' As box2d));
st_ymin
-----
2
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to
a BOX3D
SELECT ST_YMin('LINESTRING(1 3, 5 6)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_YMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227
150406 3)'));
st_ymin
-----
150406
```

See Also

[ST_GeomFromEWKT](#), [ST_XMin](#), [ST_XMax](#), [ST_YMax](#), [ST_ZMax](#), [ST_ZMin](#)

8.17.13 ST_ZMax

ST_ZMax — Returns the Z maxima of a 2D or 3D bounding box or a geometry.

Synopsis

```
float ST_ZMax(box3d aGeomorBox2DorBox3D);
```

Description

Returns the Z maxima of a 2D or 3D bounding box or a geometry.



Note

Although this function is only defined for box3d, it also works for box2d and geometry values due to automatic casting. However it will not accept a geometry or box2d text representation, since those do not auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_ZMax('BOX3D(1 2 3, 4 5 6)');
st_zmax
-----
6

SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING(1 3 4, 5 6 7)'));
st_zmax
-----
7

SELECT ST_ZMax('BOX3D(-3 2 1, 3 4 1)');
st_zmax
-----
1
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to
a BOX3D
SELECT ST_ZMax('LINESTRING(1 3 4, 5 6 7)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_ZMax(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227
150406 3)'));
st_zmax
-----
3
```

See Also

[ST_GeomFromEWKT](#), [ST_XMin](#), [ST_XMax](#), [ST_YMax](#), [ST_YMin](#), [ST_ZMax](#)

8.17.14 ST_ZMin

ST_ZMin — Returns the Z minima of a 2D or 3D bounding box or a geometry.

Synopsis

```
float ST_ZMin(box3d aGeomorBox2DorBox3D);
```

Description

Returns the Z minima of a 2D or 3D bounding box or a geometry.



Note

Although this function is only defined for box3d, it also works for box2d and geometry values due to automatic casting. However it will not accept a geometry or box2d text representation, since those do not auto-cast.



This function supports 3d and will not drop the z-index.



This method supports Circular Strings and Curves

Examples

```
SELECT ST_ZMin('BOX3D(1 2 3, 4 5 6)');
st_zmin
-----
3

SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING(1 3 4, 5 6 7)'));
st_zmin
-----
4

SELECT ST_ZMin('BOX3D(-3 2 1, 3 4 1)');
st_zmin
-----
1
--Observe THIS DOES NOT WORK because it will try to auto-cast the string representation to
a BOX3D
SELECT ST_ZMin('LINESTRING(1 3 4, 5 6 7)');

--ERROR: BOX3D parser - doesn't start with BOX3D(

SELECT ST_ZMin(ST_GeomFromEWKT('CIRCULARSTRING(220268 150415 1,220227 150505 2,220227
150406 3)'));
st_zmin
-----
1
```

See Also

[ST_GeomFromEWKT](#), [ST_GeomFromText](#), [ST_XMin](#), [ST_XMax](#), [ST_YMax](#), [ST_YMin](#), [ST_ZMax](#)

8.18 Linear Referencing

8.18.1 ST_LineInterpolatePoint

ST_LineInterpolatePoint — Returns a point interpolated along a line at a fractional location.

Synopsis

geometry **ST_LineInterpolatePoint**(geometry a_linestring, float8 a_fraction);

ST_LineLocatePoint

ST_LineLocatePoint(geometry, point) float8
 Returns the location of point along the line geometry, as a fraction of the total line length. The location is returned as a float8 value from 0.0 to 1.0, where 0.0 represents the start of the line and 1.0 represents the end. If the point is not on the line, the function returns the location of the point on the line segment that is closest to the point. The function is implemented using the Douglas-Peucker algorithm.

**Note**

This function computes points in 2D and then interpolates values for Z and M, while **ST_LineInterpolatePoint** computes points in 3D and only interpolates the M value.

**Note**

1.1.1 ST_LineLocatePoint(geometry, point) float8
 Returns the location of point along the line geometry, as a fraction of the total line length. The location is returned as a float8 value from 0.0 to 1.0, where 0.0 represents the start of the line and 1.0 represents the end. If the point is not on the line, the function returns the location of the point on the line segment that is closest to the point. The function is implemented using the Douglas-Peucker algorithm.

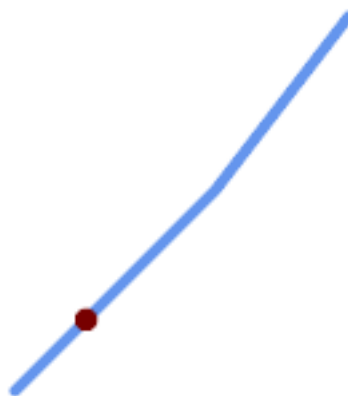
0.8.2 ST_LineLocatePoint(geometry, point) float8
 Returns the location of point along the line geometry, as a fraction of the total line length. The location is returned as a float8 value from 0.0 to 1.0, where 0.0 represents the start of the line and 1.0 represents the end. If the point is not on the line, the function returns the location of the point on the line segment that is closest to the point. The function is implemented using the Douglas-Peucker algorithm.

1.1.1 ST_LineLocatePoint(geometry, point) float8
 Returns the location of point along the line geometry, as a fraction of the total line length. The location is returned as a float8 value from 0.0 to 1.0, where 0.0 represents the start of the line and 1.0 represents the end. If the point is not on the line, the function returns the location of the point on the line segment that is closest to the point. The function is implemented using the Douglas-Peucker algorithm.

ST_LineLocatePoint(geometry, point) float8
 Returns the location of point along the line geometry, as a fraction of the total line length. The location is returned as a float8 value from 0.0 to 1.0, where 0.0 represents the start of the line and 1.0 represents the end. If the point is not on the line, the function returns the location of the point on the line segment that is closest to the point. The function is implemented using the Douglas-Peucker algorithm.



This function supports 3d and will not drop the z-index.

ST_LineInterpolatePoint

ST_LineInterpolatePoint(geometry, fraction) geometry
 Returns a point on the line geometry at the specified fraction of the total line length. The fraction is a float8 value from 0.0 to 1.0, where 0.0 represents the start of the line and 1.0 represents the end. If the fraction is outside the range [0, 1], the function returns the point at the start or end of the line, whichever is closer.


```
-- The point 20% along a line

SELECT ST_AsEWKT( ST_LineInterpolatePoint(
    'LINESTRING(25 50, 100 125, 150 190)',
    0.2 ));
-----
POINT(51.5974135047432 76.5974135047432)
```

The mid-point of a 3D line:

```
SELECT ST_AsEWKT( ST_LineInterpolatePoint('
    LINESTRING(1 2 3, 4 5 6, 6 7 8)',
    0.5 ));
-----
POINT(3.5 4.5 5.5)
```

The closest point on a line to a point:

```
SELECT ST_AsText( ST_LineInterpolatePoint( line.geom,
    ST_LineLocatePoint( line.geom, 'POINT(4 3)'))
FROM (SELECT ST_GeomFromText('LINESTRING(1 2, 4 5, 6 7)') As geom) AS line;
-----
POINT(3 4)
```

ST_LineInterpolatePoint

ST_LineInterpolatePoints, ST_LineInterpolatePoint, ST_LineMerge

8.18.2 ST_LineInterpolatePoint

ST_LineInterpolatePoint — Returns a point interpolated along a 3D line at a fractional location.

Synopsis

geometry **ST_LineInterpolatePoint**(geometry a_linestring, float8 a_fraction);

ST_LineInterpolatePoint

ST_LineInterpolatePoint computes points in 2D and then interpolates the values for Z and M, while this function computes points in 3D and only interpolates the M value.



Note

ST_LineInterpolatePoint computes points in 2D and then interpolates the values for Z and M, while this function computes points in 3D and only interpolates the M value.

2.0.0 **ST_LineInterpolatePoint** computes points in 2D and then interpolates the values for Z and M, while this function computes points in 3D and only interpolates the M value.



This function supports 3d and will not drop the z-index.

Return point 20% along 3D line

Return point 20% along 3D line

```
-- 2D LINESTRING(25 50, 100 125, 150 190) as the_line) As
    st_asewkt
FROM (SELECT ST_GeomFromEWKT('LINESTRING(25 50, 100 125, 150 190)') as the_line) As
    foo;
POINT(51.5974135047432 76.5974135047432)
```

ST_LineInterpolatePoint

ST_LineInterpolatePoint, **ST_LineMerge**

8.18.3 ST_LineInterpolatePoints

ST_LineInterpolatePoints — Returns points interpolated along a line at a fractional interval.

Synopsis

geometry **ST_LineInterpolatePoints**(geometry a_linestring, float8 a_fraction, boolean repeat);

ST_LineInterpolatePoints

Returns one or more points interpolated along a line at a fractional interval. The first argument must be a **LINESTRING**. The second argument is a float8 between 0 and 1 representing the spacing between the points as a fraction of line length. If the third argument is false, at most one point will be constructed (which is equivalent to **ST_LineInterpolatePoint**.)

If the result has zero or one points, it is returned as a **POINT**. If it has two or more points, it is returned as a **MULTIPOINT**.

Availability: 2.5.0



This function supports 3d and will not drop the z-index.



This function supports M coordinates.



```
--Return points each 20% along a 2D line
SELECT ST_AsText(ST_LineInterpolatePoints('LINESTRING(25 50, 100 125, 150 190)', 0.20))
-----
MULTIPOINT((51.5974135047432 76.5974135047432),(78.1948270094864 103.194827009486) ←
, (104.132163186446 130.37181214238),(127.066081593223 160.18590607119),(150 190))
```

ST_LineInterpolatePoint, ST_LineLocatePoint

ST_LineLocatePoint — Returns the fractional location of the closest point on a line to a point.

```
float8 ST_LineLocatePoint(geometry a_linestring, geometry a_point);
```

입력 포인트에 가장 가까운 위치있는 라인스트링 상의 포인트를 나타내는 **2차원 라인** 전운 길이의 비율을 0에서 1 사이의 부동소수점 데이터형(float)으로 반환합니다.

반환된 위치를 이용해서 포인트(ST_LineSubstring)을 추출할 수 있습니다.

이 함수는 주소의 개수의 근사치구하는 데 유용합니다.

1.1.0 버전부터 사용할 수 있습니다.

변경 사항: 2.1.0 미만 버전 즉 2.0.x 버전까이 함수의 명칭은 ST Line Locate Point였습니다

예시

```
-- &#xc70;&#xb9ac;&#xb97c; &#xb530;&#xb77c; &#xb788;&#xb294; <
&#xd3ec;&#xc778;&#xd2b8;&#xc758; &#xbc88;&#xc9c0;&#xc218;&#xb97c; &#xb300;&#xcda9; <
&#xc5b4;&#xb9bc;&#xc9d0;&#xc791;&#xd558;&#xae30;

-- foo &#xc804;&#xccb4;&#xac00; &#xc9d1; &#xc911;&#xc2ec;&#xc810; &#xbc0f; <
&#xac70;&#xb9ac;&#xcc98;&#xb7fc; &#xbcfc4;&#xc774;&#xb294; &#xac00;&#xc9dc; <
&#xb370;&#xc774;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xae30; &#xc704;&#xd55c; <
&#xac83;&#xc774;&#xb77c;&#xb294; &#xc810;&#xc5d0; &#xc8fc;&#xc758;

-- ST_DWithin(&#xc744; &#xc774;&#xc6a9;&#xd574;&#xc11c; &#xac70;&#xb9ac;&#xc5d0;&#xc11c; <
&#xb108;&#xbbb34; &#xba40;&#xb9ac; &#xb5a8;&#xc5b4;&#xc9c4; &#xc9d1;&#xb4e4;&#xc744; <
&#xc81c;&#xc678;

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

-- &#xd3ec;&#xc778;&#xd2b8; &#xb610;&#xb294; &#xb2e4;&#xb978; &#xb3c4;&#xd615;&#xc5d0; <
&#xac00;&#xc7a5; &#xac00;&#xae4c;&#xc6b4; &#xb77c;&#xc778; &#xc0c1;&#xc758; <
&#xd3ec;&#xc778;&#xd2b8;&#xb97c; &#xcc3e;&#xae30;

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)
```

స고

ST_DWithin, ST_Length2D, ST_LineInterpolatePoint, ST_LineSubstring

8.18.5 ST_LineSubstring

ST LineSubstring — Returns the part of a line between two fractional locations.

Synopsis

```
geometry ST_LineSubstring(geometry a linestring, float8 startfraction, float8 endfraction);
```

설명

Computes the line which is the section of the input line starting and ending at the given fractional locations. The first argument must be a LINESTRING. The second and third arguments are values in the range [0, 1] representing the start and end locations as fractions of line length. The Z and M values are interpolated for added endpoints if present.

```
'&#x2dc;*&#x791;','&#xacfc;','&#xb05d;','&#xc774;*&#xb3d9;*&#xc77c;*&#xd55c;*&#xac12;*&#xc77c;*&#xacbd;*&#xc6b0;*&#xc774;*&#xd568;*&#xc218;*&#xb294; ST_LineInterpolatePoint &#xd568;*&#xc218;*&#xc640;*&#xac19;*&#xc544;*&#xc9d1;*&#xb2c8;*&#xb2e
```



Note

This only works with LINESTRINGs. To use on contiguous MULTILINESTRINGs first join them with [ST_LineMerge](#).



Note

```
1.1.1   &#x30;&#xd3;&#xd1;&#xbd8;&#xd13;    &#xc774;    &#xd568;&#xc218;&#xb294;    M   &#xbc0f;  
Z&#xac12;(&#xc774; &#xc788;&#xc744; &#xabd;&#xc6b0;)&#xb3c4; &#xbcf4;&#xac04;&#xd569;&#xb2c8;&#xb2e4;.  
&#xc774;&#xc804;    &#xb30;&#xd3;&#xd1;&#xcd0;&#xc11c;&#xb294;    &#xb450;    &#xac12;&#xc744;  
&#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc558;&#xc2b5;&#xb2c8;&#xb2e4;.
```

1.1.0 버전부터 사용할 수 있습니다.

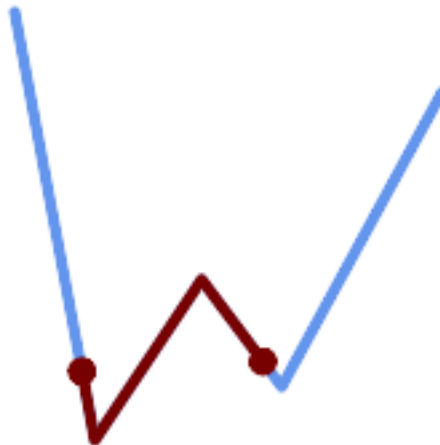
1.1.1 ಄전부터 Z ఏ M을 지원합니다.

¬항; 2.1.0 미만버전, 즉 2.0.x &#bc84;전까&
이함수의 명칭은 ST Line Substring이었습니&#xa



This function supports 3d and will not drop the z-index.

예시



중ఄ 1/3 범위(0.333, 0.666)를 중&#xcxa9;해서
출력한 라인스트&#xb9c1:

```
SELECT ST_AsText(ST_LineSubstring( 'LINESTRING (20 180, 50 20, 90 80, 120 40, 180 150)', ←
    0.333, 0.666));
```

```
LINESTRING (45.17311810399485 45.74337011202746, 50 20, 90 80, 112.97593050157862 49.36542599789519) ↵
```

If start and end locations are the same, the result is a POINT.

```
SELECT ST_AsText(ST_LineSubstring( 'LINESTRING(25 50, 100 125, 150 190)', 0.333, 0.333));
-----
POINT(69.2846934853974 94.2846934853974)
```

A query to cut a LineString into sections of length 100 or shorter. It uses `generate_series()` with a `CROSS JOIN LATERAL` to produce the equivalent of a FOR loop.

```
WITH data(id, geom) AS (VALUES
    ( 'A', 'LINESTRING( 0 0, 200 0)::geometry ),
    ( 'B', 'LINESTRING( 0 100, 350 100)::geometry ),
    ( 'C', 'LINESTRING( 0 200, 50 200)::geometry )
)
SELECT id, i,
       ST_AsText( ST_LineSubstring( geom, startfrac, LEAST( endfrac, 1 ) ) ) AS geom
FROM (
    SELECT id, geom, ST_Length(geom) len, 100 sublen FROM data
    ) AS d
CROSS JOIN LATERAL (
    SELECT i, (sublen * i) / len AS startfrac,
           (sublen * (i+1)) / len AS endfrac
    FROM generate_series(0, floor( len / sublen )::integer ) AS t(i)
    -- skip last i if line length is exact multiple of sublen
    WHERE (sublen * i) / len <> 1.0
    ) AS d2;
```

id	i	geom
A	0	LINESTRING(0 0,100 0)
A	1	LINESTRING(100 0,200 0)
B	0	LINESTRING(0 100,100 100)
B	1	LINESTRING(100 100,200 100)
B	2	LINESTRING(200 100,300 100)
B	3	LINESTRING(300 100,350 100)
C	0	LINESTRING(0 200,50 200)

ST_LineSubstring

[ST_Length](#), [ST_LineInterpolatePoint](#), [ST_LineMerge](#)

8.18.6 ST_LocateAlong

ST_LocateAlong — Returns the point(s) on a geometry that match a measure value.

Synopsis

geometry **ST_LocateAlong**(geometry ageom_with_measure, float8 a_measure, float8 offset);

ST_LocateAlong

Returns the location(s) along a measured geometry that have the given measure values. The result is a Point or MultiPoint. Polygonal inputs are not supported.

If `offset` is provided, the result is offset to the left or right of the input line by the specified distance. A positive offset will be to the left, and a negative one to the right.

**Note**

Use this function only for linear geometries with an M component

The semantic is specified by the *ISO/IEC 13249-3 SQL/MM Spatial* standard.

1.1.0 `ST_LocateAlong(geometry geomA, float8 measure_start, float8 measure_end, float8 offset)`
 Returns the portion of a geometry that match a measure range.

`ST_LocateAlong(geometry geomA, float8 measure_start, float8 measure_end, float8 offset)`
 Returns the portion of a geometry that match a measure range.



This function supports M coordinates.



This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.13

Example

```
SELECT ST_AsText (
  ST_LocateAlong (
    'MULTILINESTRINGM((1 2 3, 3 4 2, 9 4 3),(1 2 3, 5 4 5))'::geometry,
    3 ));
```

```
-----
MULTIPOINT M ((1 2 3), (9 4 3), (1 2 3))
```

See Also

[ST_LocateBetween](#), [ST_LocateBetweenElevations](#), [ST_InterpolatePoint](#)

8.18.7 ST_LocateBetween

`ST_LocateBetween` — Returns the portions of a geometry that match a measure range.

Synopsis

geometry **ST_LocateBetween**(geometry geomA, float8 measure_start, float8 measure_end, float8 offset);

Example

```
SELECT ST_AsText (
  ST_LocateBetween (
    'MULTILINESTRINGM((1 2 3, 3 4 2, 9 4 3),(1 2 3, 5 4 5))'::geometry,
    3, 4, 0));
```

```
-----
MULTIPOINT M ((1 2 3), (3 4 2), (9 4 3))
```

Clipping a non-convex POLYGON may produce invalid geometry.

The semantic is specified by the *ISO/IEC 13249-3 SQL/MM Spatial* standard.

1.1.0 ST_Locate_Between_Measures(geometry, measure1, measure2);

2.0.0 ST_Locate_Along_Measure(geometry, measure, distance);

Enhanced: 3.0.0 - added support for POLYGON, TIN, TRIANGLE.



This function supports M coordinates.

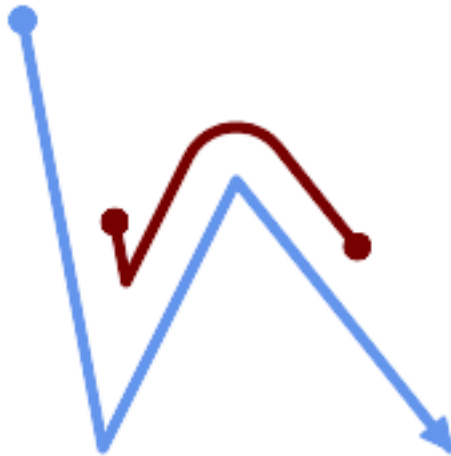


This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1

SQL/MM IEC 13249-3: 5.1

```
SELECT ST_AsText (
  ST_LocateBetween (
    'MULTILINESTRING M ((1 2 3, 3 4 2, 9 4 3), (1 2 3, 5 4 5))':: geometry,
    1.5, 3 ));
```

```
GEOMETRYCOLLECTION M (LINESTRING M (1 2 3, 3 4 2, 9 4 3), POINT M (1 2 3))
```



A LineString with the section between measures 2 and 8, offset to the left

```
SELECT ST_AsText ( ST_LocateBetween (
  ST_AddMeasure('LINESTRING (20 180, 50 20, 100 120, 180 20)', 0, 10),
  2, 8,
  20
));
```

```
MULTILINESTRING ((54.49835019899045 104.53426957938231, 58.70056060327303 ↵
  82.12248075654186, 69.16695286779743 103.05526528559065, 82.11145618000168 ↵
  128.94427190999915, 84.24893681714357 132.32493442618113, 87.01636951231555 ↵
  135.21267035596549, 90.30307285299679 137.49198684843182, 93.97759758337769 ↵
  139.07172433557758, 97.89298381958797 139.8887023914453, 101.89263860095893 ↵
  139.9102465862721, 105.81659870902816 139.13549527600819, 109.50792827749828 ↵
  137.5954340631298, 112.81899532549731 135.351656550512, 115.6173761888606 ↵
  132.49390095108848, 145.31017306064817 95.37790486135405))
```


ST_LocateBetweenElevations

[ST_LocateAlong](#), [ST_LocateAlong](#), [ST_LocateBetween](#)

8.18.8 ST_LocateBetweenElevations

ST_LocateBetweenElevations — Returns the portions of a geometry that lie in an elevation (Z) range.

Synopsis

geometry **ST_LocateBetweenElevations**(geometry geom_mline, float8 elevation_start, float8 elevation_end);

ST_LocateBetweenElevations

Returns a geometry (collection) with the portions of a geometry that lie in an elevation (Z) range.

Clipping a non-convex POLYGON may produce invalid geometry.

1.4.0 [ST_LocateBetweenElevations](#) (geometry geom_mline, float8 elevation_start, float8 elevation_end);

Enhanced: 3.0.0 - added support for POLYGON, TIN, TRIANGLE.



This function supports 3d and will not drop the z-index.

ST_LocateBetweenElevations

```
SELECT ST_AsText (
  ST_LocateBetweenElevations (
    'LINESTRING(1 2 3, 4 5 6)::geometry',
    2, 4 ));

          st_astext
-----
MULTILINESTRING Z ((1 2 3,2 3 4))

SELECT ST_AsText (
  ST_LocateBetweenElevations (
    'LINESTRING(1 2 6, 4 5 -1, 7 8 9)',
    6, 9)) As ewelev;

          ewelev
-----
GEOMETRYCOLLECTION Z (POINT Z (1 2 6),LINESTRING Z (6.1 7.1 6,7 8 9))
```

ST_LocateBetweenElevations

[ST_Dump](#), [ST_LocateAlong](#), [ST_LocateBetween](#)

8.18.9 ST_InterpolatePoint

ST_InterpolatePoint — Returns the point on a geometry that lies at a given fraction of the total length of the geometry. The fraction must be between 0 and 1. The geometry must be a LINESTRING, POLYGON, or TIN. The function will return a POINT geometry. The function will return a POINT geometry. The function will return a POINT geometry.

Synopsis

float8 **ST_InterpolatePoint**(geometry linear_geom_with_measure, geometry point);

Description

Returns an interpolated measure value of a linear measured geometry at the location closest to the given point.



Note

Use this function only for linear geometries with an M component

2.0.0 Changes



This function supports 3d and will not drop the z-index.

Example

```
SELECT ST_InterpolatePoint('LINESTRING M (0 0 0, 10 0 20)', 'POINT(5 5)');
-----
10
```

See also

[ST_AddMeasure](#), [ST_LocateAlong](#), [ST_LocateBetween](#)

8.18.10 ST_AddMeasure

ST_AddMeasure — Interpolates measures along a linear geometry.

Synopsis

geometry **ST_AddMeasure**(geometry geom_mline, float8 measure_start, float8 measure_end);

Description

Changes

1.5.0 Changes



This function supports 3d and will not drop the z-index.

예시

```
SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRING(1 0, 2 0, 4 0)'),1,4)) As ewelev;
           ewelev
-----
LINESTRINGM(1 0 1,2 0 2,4 0 4)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRING(1 0 4, 2 0 4, 4 0 4)'),10,40)) As ewelev;
           ewelev
-----
LINESTRING(1 0 4 10,2 0 4 20,4 0 4 40)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('LINESTRINGM(1 0 4, 2 0 4, 4 0 4)'),10,40)) As ewelev;
           ewelev
-----
LINESTRINGM(1 0 10,2 0 20,4 0 40)

SELECT ST_AsText(ST_AddMeasure(
ST_GeomFromEWKT('MULTILINESTRINGM((1 0 4, 2 0 4, 4 0 4),(1 0 4, 2 0 4, 4 0 4)'),10,70)) As ←
           ewelev;
                                     ewelev
-----
MULTILINESTRINGM((1 0 10,2 0 20,4 0 40),(1 0 40,2 0 50,4 0 70))
```

8.19 Trajectory Functions

8.19.1 ST_IsValidTrajectory

ST_IsValidTrajectory — Tests if the geometry is a valid trajectory.

Synopsis

boolean **ST_IsValidTrajectory**(geometry line);

Description

Tests if a geometry encodes a valid trajectory. A valid trajectory is represented as a **LINESTRING** with measures (M values). The measure values must increase from each vertex to the next.

Valid trajectories are expected as input to spatio-temporal functions like **ST_ClosestPointOfApproach**

Availability: 2.2.0



This function supports 3d and will not drop the z-index.

Examples

```
-- 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

```

See Also

[ST_ClosestPointOfApproach](#)

8.19.2 ST_ClosestPointOfApproach

ST_ClosestPointOfApproach — Returns a measure at the closest point of approach of two trajectories.

Synopsis

```
float8 ST_ClosestPointOfApproach(geometry track1, geometry track2);
```

Description

Returns the smallest measure at which points interpolated along the given trajectories are at the smallest distance.

Inputs must be valid trajectories as checked by [ST_IsValidTrajectory](#). Null is returned if the trajectories do not overlap in their M ranges.

See [ST_LocateAlong](#) for getting the actual points at the given measure.

Availability: 2.2.0



This function supports 3d and will not drop the z-index.

Examples

```

-- 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
```

See Also

[ST_IsValidTrajectory](#), [ST_DistanceCPA](#), [ST_LocateAlong](#), [ST_AddMeasure](#)

8.19.3 ST_DistanceCPA

ST_DistanceCPA — Returns the distance between the closest point of approach of two trajectories.

Synopsis

```
float8 ST_DistanceCPA(geometry track1, geometry track2);
```

Description

Returns the minimum distance two moving objects have ever been each other.

Inputs must be valid trajectories as checked by [ST_IsValidTrajectory](#). Null is returned if the trajectories do not overlap in their M ranges.

Availability: 2.2.0



This function supports 3d and will not drop the z-index.

Examples

```
-- 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
```

See Also

[ST_IsValidTrajectory](#), [ST_ClosestPointOfApproach](#), [ST_AddMeasure](#), [ST_Distance](#)

8.19.4 ST_CPAWithin

ST_CPAWithin — Tests if the closest point of approach of two trajectories is within the specified distance.

Synopsis

boolean **ST_CPAWithin**(geometry track1, geometry track2, float8 dist);

Description

Tests whether two moving objects have ever been closer than the specified distance.

Inputs must be valid trajectories as checked by **ST_IsValidTrajectory**. False is returned if the trajectories do not overlap in their M ranges.

Availability: 2.2.0



This function supports 3d and will not drop the z-index.

Examples

```
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_CPAWithin(a,b,2), ST_DistanceCPA(a,b) distance FROM inp;
```

st_cpawithin	distance
t	1.96521473776207

See Also

ST_IsValidTrajectory, **ST_ClosestPointOfApproach**, **ST_DistanceCPA**, **ST_Distance**

8.20 SFCGAL

8.20.1 postgis_sfcgal_version

postgis_sfcgal_version — SFCGAL

Synopsis

text **postgis_sfcgal_version**(void);

설명

2.1.0 ಄전부터 사용할 수 있습니다.

- ✔ 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).

స고

```
postgis_sfcgal_version
```

8.20.2 postgis_sfcgal_version

`postgis_sfcgal_version` — Returns the full version of SFCGAL in use including CGAL and Boost versions

Synopsis

```
text postgis_sfcgal_version(void);
```

설명

Returns the full version of SFCGAL in use including CGAL and Boost versions

2.1.0 버전부터 사용할 수 있습니다.

- ✔ 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).

స고

```
postgis_sfcgal_version
```

8.20.3 ST_3DArea

ST_3DArea — 3న원 표면 도형의 면적을 계&#xc0입체일 &#xachd;우 0을 반환할 것입니다.

Synopsis

```
floatST_3DArea(geometry geom1);
```

ST_3DConvexHull

2.1.0 This method needs SFCGAL backend.



This method needs SFCGAL backend.



This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 8.1, 10.5



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).

ST_3DConvexHull

ST_3DConvexHull(geom) As cube_surface_area,
ST_3DConvexHull(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);

```
SELECT ST_3DConvexHull(geom) As cube_surface_area,
       ST_3DConvexHull(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

ST_3DConvexHull

ST_Area, ST_MakeSolid, ST_IsSolid, ST_Area

8.20.4 ST_3DConvexHull

ST_3DConvexHull — This method needs SFCGAL backend.

Synopsis

geometry **ST_3DConvexHull**(geometry geom1);

ST_3DConvexHull

2.1.0 This method needs SFCGAL backend.



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



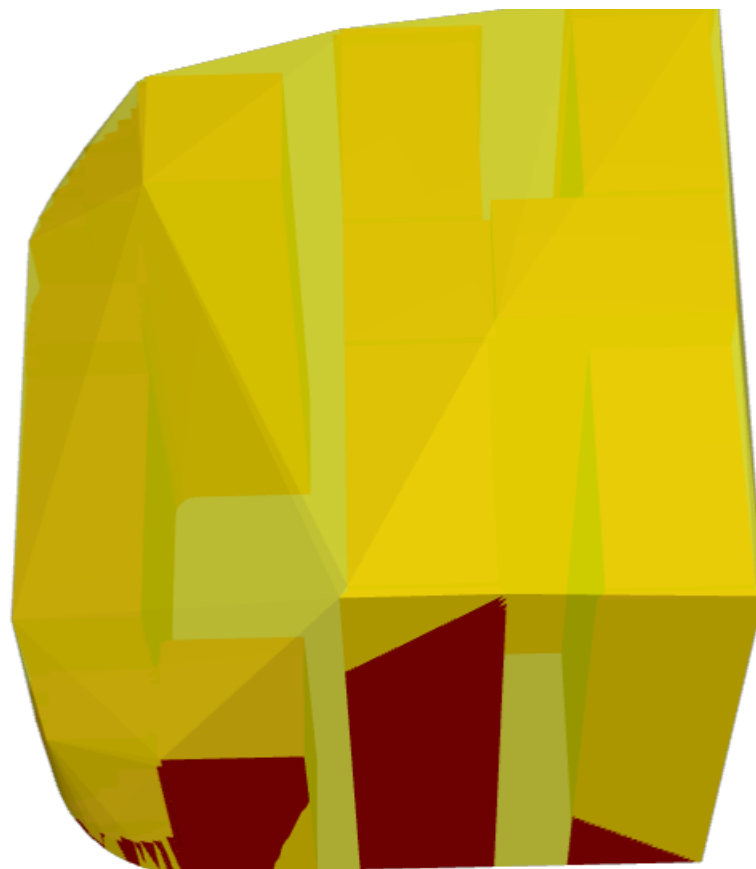
This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Example

```
SELECT ST_AsText(ST_3DConvexHull('LINESTRING Z(0 0 5, 1 5 3, 5 7 6, 9 5 3, 5 7 5, 6 3 5) ↵
 '::geometry));
```

```
POLYHEDRALSURFACE Z (((1 5 3,9 5 3,0 0 5,1 5 3)),((1 5 3,0 0 5,5 7 6,1 5 3)),((5 7 6,5 7 ↵
 5,1 5 3,5 7 6)),((0 0 5,6 3 5,5 7 6,0 0 5)),((6 3 5,9 5 3,5 7 6,6 3 5)),((0 0 5,9 5 3,6 ↵
 3 5,0 0 5)),((9 5 3,5 7 5,5 7 6,9 5 3)),((1 5 3,5 7 5,9 5 3,1 5 3)))
```

```
WITH f AS (SELECT i, ST_Extrude(geom, 0,0, i ) AS geom
FROM ST_Subdivide(ST_Letters('CH'),5) WITH ORDINALITY AS sd(geom,i)
)
SELECT ST_3DConvexHull(ST_Collect(f.geom) )
FROM f;
```



Original geometry overlaid with 3D convex hull

Related Functions

[ST_Letters](#), [ST_AsX3D](#)

8.20.5 ST_3DIntersection

ST_3DIntersection — 3󌰨원 교차를 수행합니다.

Synopsis

```
geometry ST_3DIntersection(geometry geom1, geometry geom2);
```

설명

```
geom1&#xacfc; geom2&#xac00; &#xacf5;&#xc720;&#xd558;&#xb294; &#xbd80;&#xbd84;&#xc744; &#xb3c4;&#xd615;&#xc73c;
&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

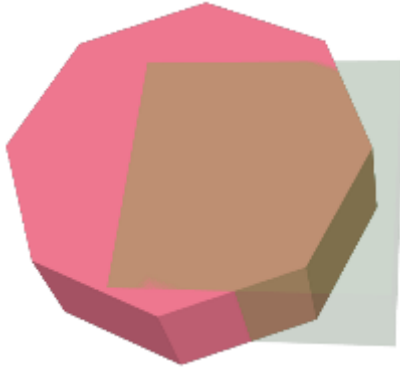
2.1.0 버전부터 사용할 수 있습니다.

- ✔ This method needs SFCGAL backend.
- ✔ This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1
- ✔ 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).

예시

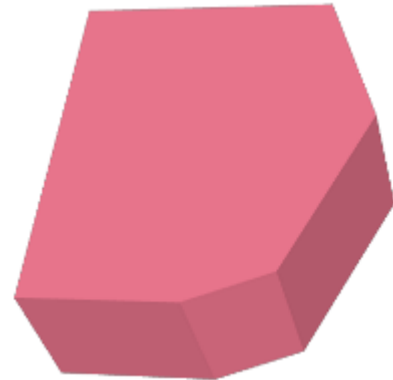
PostGIS [ST_AsX3D](#) ॼ 통해 3차원 영상을 생성한 다음 [X3Dom HTML](#) 자바스크립트 렌더링 라이브러리 를 이용해서 [HTML](#)로 렌&#xb3

```
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;
```



원본 3차원
 도형들이
 중೉합니다.
 geom2를
 반투명하게
 표현했습니다.

```
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;
```



```
geom1&#xacfc; geom2&#xac00;
&#xad50;&#xcc28;&#xd55c; &#xbd80;&#xbd84;
```

3న원 라인스트링과 폴리곤

```
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)
```

7

```
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)))
```

두물체의입체교차또한물체인
두물체가교차하는부분(ST_Dimension 함&#xc2
3을반환)

```
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 ((13.333333333333 13.333333333333 10,20 20 0,20 20 ←
10,13.333333333333 13.333333333333 10)),
(20 20 10,16.666666666667 23.333333333333 10,13.333333333333 13.333333333333 ←
10,20 20 10)),
(20 20 0,16.666666666667 23.333333333333 10,20 20 10,20 20 0)),
(13.333333333333 13.333333333333 10,10 10 0,20 20 0,13.333333333333 ←
13.333333333333 10)),
(16.666666666667 23.333333333333 10,12 28 10,13.333333333333 13.333333333333 ←
10,16.666666666667 23.333333333333 10)),
(20 20 0,9.999999999999 30 0,16.666666666667 23.333333333333 10,20 20 0)),
(10 10 0,9.999999999999 30 0,20 20 0,10 10 0)),((13.333333333333 ←
13.333333333333 10,12 12 10,10 10 0,13.333333333333 13.333333333333 10)),
(12 28 10,12 12 10,13.333333333333 13.333333333333 10,12 28 10)),
(16.666666666667 23.333333333333 10,9.999999999999 30 0,12 28 ←
10,16.666666666667 23.333333333333 10)),
(10 10 0,0 20 0,9.999999999999 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)),
(9.999999999999 30 0,11 29 10,12 28 10,9.999999999999 30 0)),((0 20 0,2 20 ←
10,9.999999999999 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)),
(9.999999999999 30 0,2 20 10,11 29 10,9.999999999999 30 0)),((11 11 10,11 29 ←
10,2 20 10,11 11 10)))

8.20.6 ST_3DDifference

ST_3DDifference — 3D difference of two surfaces.

Synopsis

```
geometry ST_3DDifference(geometry geom1, geometry geom2);
```

설명

geom2의 일부분이 아닌 geom1의 부분을
반환합니다.

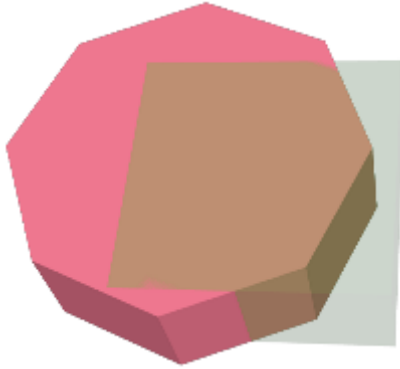
2.2.0 버전부터 사용할 수 있습니다.

- ✔ This method needs SFCGAL backend.
- ✔ This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1
- ✔ 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).

예시

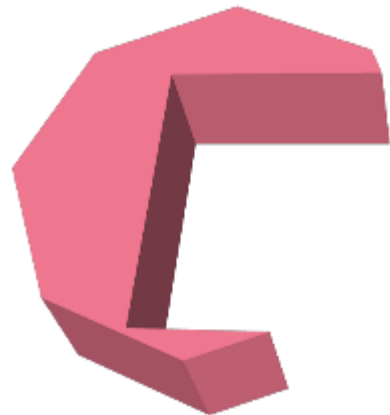
PostGIS **ST_AsX3D** 를 통해 3차원 영상을 생성한 다음 **X3Dom HTML** 자바스크립트 렌더링 라이브러리 를 이용해서 **HTML**로 렌&#xb3

```
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;
```



geom2

```
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;
```



geom2

ST_3DUnion

ST_Extrude, ST_AsX3D, ST_3DIntersection ST_3DUnion

8.20.7 ST_3DUnion

ST_3DUnion — Perform 3D union.

Synopsis

geometry **ST_3DUnion**(geometry geom1, geometry geom2);
 geometry **ST_3DUnion**(geometry set g1field);

2.2.0

Availability: 3.3.0 aggregate variant was added



This method needs SFCGAL backend.



This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1



This function supports 3d and will not drop the z-index.



This function supports Polyhedral surfaces.



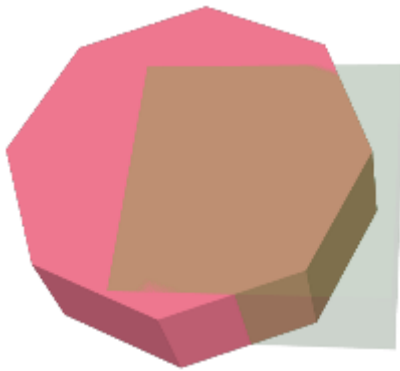
This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

Aggregate variant: returns a geometry that is the 3D union of a rowset of geometries. The `ST_3DUnion()` function is an "aggregate" function in the terminology of PostgreSQL. That means that it operates on rows of data, in the same way the `SUM()` and `AVG()` functions do and like most aggregates, it also ignores NULL geometries.

SQL Example:

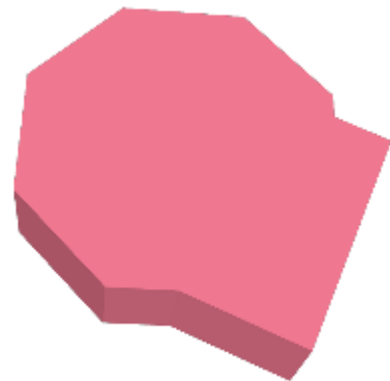
PostGIS **ST_AsX3D** returns a 3D geometry from a 2D geometry. The `ST_3DUnion()` function is an "aggregate" function in the terminology of PostgreSQL. That means that it operates on rows of data, in the same way the `SUM()` and `AVG()` functions do and like most aggregates, it also ignores NULL geometries.

```
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;
```



geom1 and geom2 are 3D geometries. The result of the 3D union operation is a single 3D geometry representing the combined volume of both extruded polygons.

```
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;
```



geom1 and geom2 are 3D geometries. The result of the 3D union operation is a single 3D geometry representing the combined volume of both extruded polygons.

SQL Example:

`ST_Extrude`, `ST_AsX3D`, `ST_3DIntersection` `ST_3DDifference`

8.20.8 ST_AlphaShape

`ST_AlphaShape` — Computes a possible concave geometry using the CGAL Alpha Shapes algorithm.

Synopsis

geometry `ST_AlphaShape`(geometry geom, float alpha, boolean allow_holes = false);

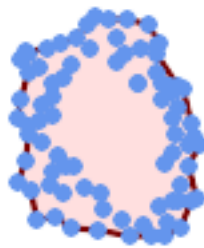
ST_ConcaveHull

Assume we are given a set S of points in 2D [...] and we would like to have something like "the shape formed by these points". This is quite a vague notion and there are probably many possible interpretations, the α -shape being one of them. Alpha shapes can be used for shape reconstruction from a dense unorganized set of data points. Indeed, an α -shape is demarcated by a frontier, which is a linear approximation of the original shape [1]. [1] F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. Technical Report CSD-TR-97-013, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, 1997. Source: [CGAL ALpha Shapes](#) This function compute the concave hull of a set of geometry, but using CGAL and a different algorithm than ST_ConcaveHull performed by the GEOS module. See : [Concave Hulls in JTS](#)

Availability: 3.3.0 - requires SFCGAL >= 1.4.1.



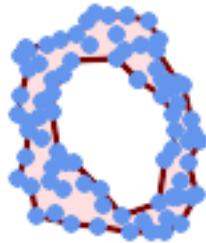
This method needs SFCGAL backend.

ST_ConcaveHull

Concave Hull of a MultiPoint (same example As ST_OptimalAlphaShape)

```
SELECT ST_AsText(ST_AlphaShape('MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70),
(88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),
(36 61),(32 65),
(81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 29),
(27 84),(52 98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 97),
(27 77),(39 88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 64),
(69 86),(60 90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 16),
(38 46),(31 59),(34 86),(45 90),(64 97))'::geometry,80.2));
```

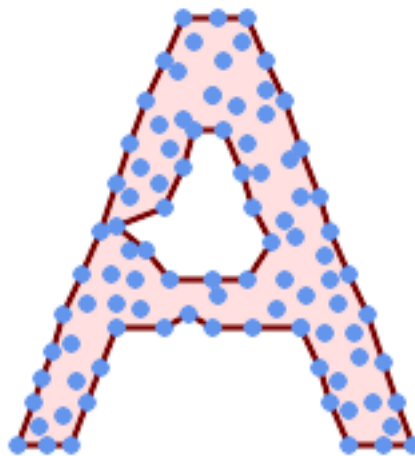
```
POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,37 23,30 22,28 33,
23 36,26 44,27 54,23 60,24 67,
27 77,24 82,26 85,34 86,39 88,45 90,49 95,52 98,57 97,64 97,72 95,76 88,75 84,83 72,
85 71,88 58,89 53));
```



Concave Hull of a MultiPoint, allowing holes (same example as ST_OptimalAlphaShape)

```
SELECT ST_AsText(ST_AlphaShape('MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50),(81 70) ←
, (88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30),(36 61) ←
, (32 65),(81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
    (78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 ←
    29),(27 84),(52 98),(72 95),(85 71),
    (75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 ←
    97),(27 77),(39 88),(60 81),
    (80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 ←
    64),(69 86),(60 90),(50 86),(43 80),(36 73),
    (36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 ←
    16),(38 46),(31 59),(34 86),(45 90),(64 97))'::geometry, 100.1,true))

POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,37 23,30 22,28 ←
33,23 36,26 44,27 54,23 60,24 67,27 77,24 82,26 85,34 86,39 88,45 90,49 95,52 98,57 ←
97,64 97,72 95,76 88,75 84,83 72,85 71,88 58,89 53),
    (36 61,36 68,40 75,43 80,50 86,60 81,68 73,77 67,81 60,82 54,81 47,78 43,76 ←
    27,62 22,54 32,48 34,44 42,38 46,36 61))
```



Concave Hull of a MultiPoint, allowing holes (same example as ST_ConcaveHull)


```
SELECT ST_AlphaShape(
    'MULTIPOINT ((132 64), (114 64), (99 64), (81 64), (63 64), (57 49), (52 36), (46 20), (37 20), (26 20), (32 36), (39 55), (43 69), (50 84), (57 100), (63 118), (68 133), (74 149), (81 164), (88 180), (101 180), (112 180), (119 164), (126 149), (132 131), (139 113), (143 100), (150 84), (157 69), (163 51), (168 36), (174 20), (163 20), (150 20), (143 36), (139 49), (132 64), (99 151), (92 138), (88 124), (81 109), (74 93), (70 82), (83 82), (99 82), (112 82), (126 82), (121 96), (114 109), (110 122), (103 138), (99 151), (34 27), (43 31), (48 44), (46 58), (52 73), (63 73), (61 84), (72 71), (90 69), (101 76), (123 71), (141 62), (166 27), (150 33), (159 36), (146 44), (154 53), (152 62), (146 73), (134 76), (143 82), (141 91), (130 98), (126 104), (132 113), (128 127), (117 122), (112 133), (119 144), (108 147), (119 153), (110 171), (103 164), (92 171), (86 160), (88 142), (79 140), (72 124), (83 131), (79 118), (68 113), (63 102), (68 93), (35 45))'::geometry,102.2,
    true);

POLYGON((134 80,136 75,130 63,135 45,132 44,126 28,117 24,110 24,98 24,80 27,82 39,72 51,60 48,56 34,52 52,42 50,
    34 54,39 66,40 81,34 90,36 100,40 116,36 123,39 128,51 129,58 132,68 135,74 142,78 147,86 146,96 146,
    108 142,114 132,112 126,112 116,116 110,120 108,125 108,128 106,125 96,132 87,134 80))
```

స고

ST_ConcaveHull, ST_OptimalAlphaShape

8.20.9 ST ApproximateMedialAxis

ST_ApproximateMedialAxis — &#xba74; &#xb3c4;&#xd615;&#xc758; &#xadfc;&#xc0ac; &#xc911;&#xc2ec;&#xd95;&#xc744; &#xacc4;&#xc0b0;&#xd569;&#xb2c8;&#xb2e4;.

Synopsis

```
geometry ST_ApproximateMedialAxis(geometry geom);
```

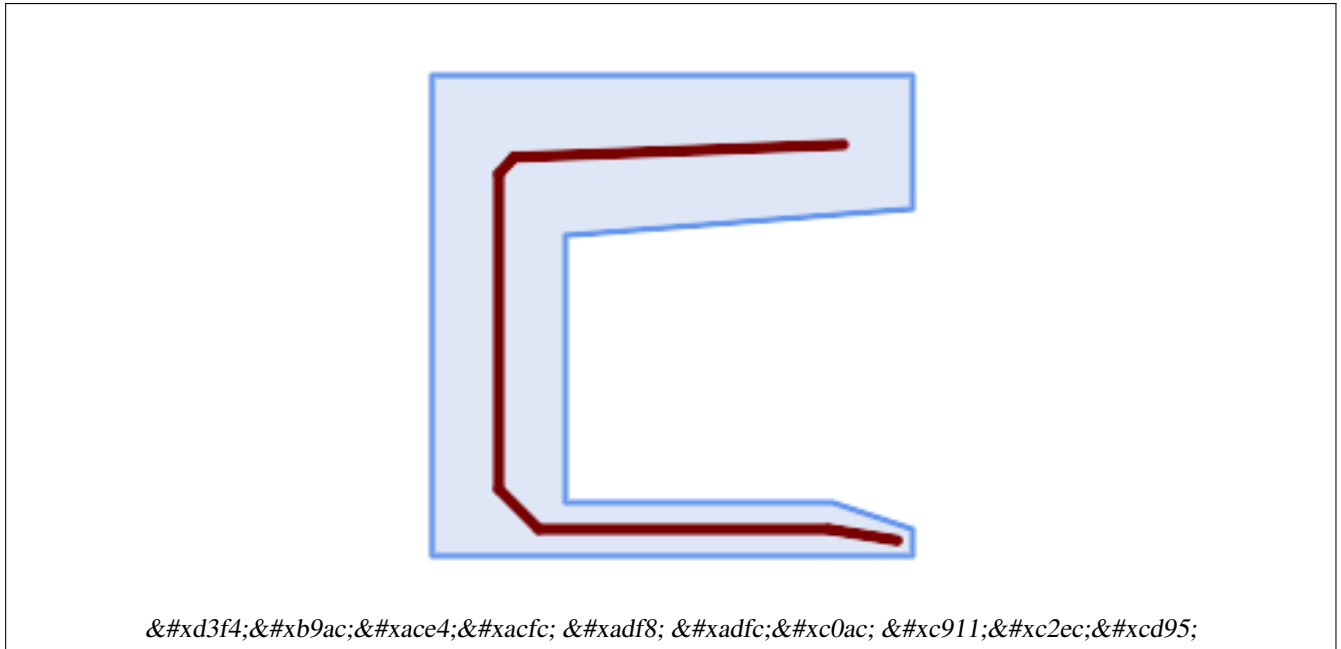
설&#xba85:

```
&#xc785;&#xb825; &#ba74; &#b3c4;&#xd615;&#xc758; &#xc9c1;&#xc120; &#xace8;&#xaca9;&#xc744; &#xbc14;&#xd0d5;&#x
&#xadfc;&#xc0ac; &#xc911;&#xc2ec;&#xcd95;(approximate medial axis)&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;
&#xc0ac;&#xc6a9; &#xac00;&#xb2a5;&#xd55c; &#xbc84;&#xc804;(1.2.0 &#xc774;&#xc0c1;)&#xc5d0; &#xb300;&#xd574;
&#xbe4c;&#xb4dc;&#xd588;&#xc744; &#xb54c; SFCGAL&#xc5d0; &#xd2b9;&#xd654;&#xb41c; API&#xb97c; &#xc774;&#xc6a9;
&#xadf8;&#xb807;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; &#xc774; &#xd568;&#xc218;&#xb294; (&#xb354; &#xb290;&#x
ST_StraightSkeleton &#xd568;&#xc218;&#xb97c; &#xb458;&#xb7ec;&#xc2fc; &#xb798;&#xd37c;(wrapper)&#xc5d0; &#xbd88;&#x
2.2.0 &#xbc84;&#xc804;&#xbd80;&#xd130; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;
```

- ✔ 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).

예시

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



సೠ

8.20.10 ST_ConstrainedDelaunayTriangles

ST_ConstrainedDelaunayTriangles — Return a constrained Delaunay triangulation around the given input geometry.

Synopsis

```
geometry ST_Tessellate(geometry geom);
```

설명

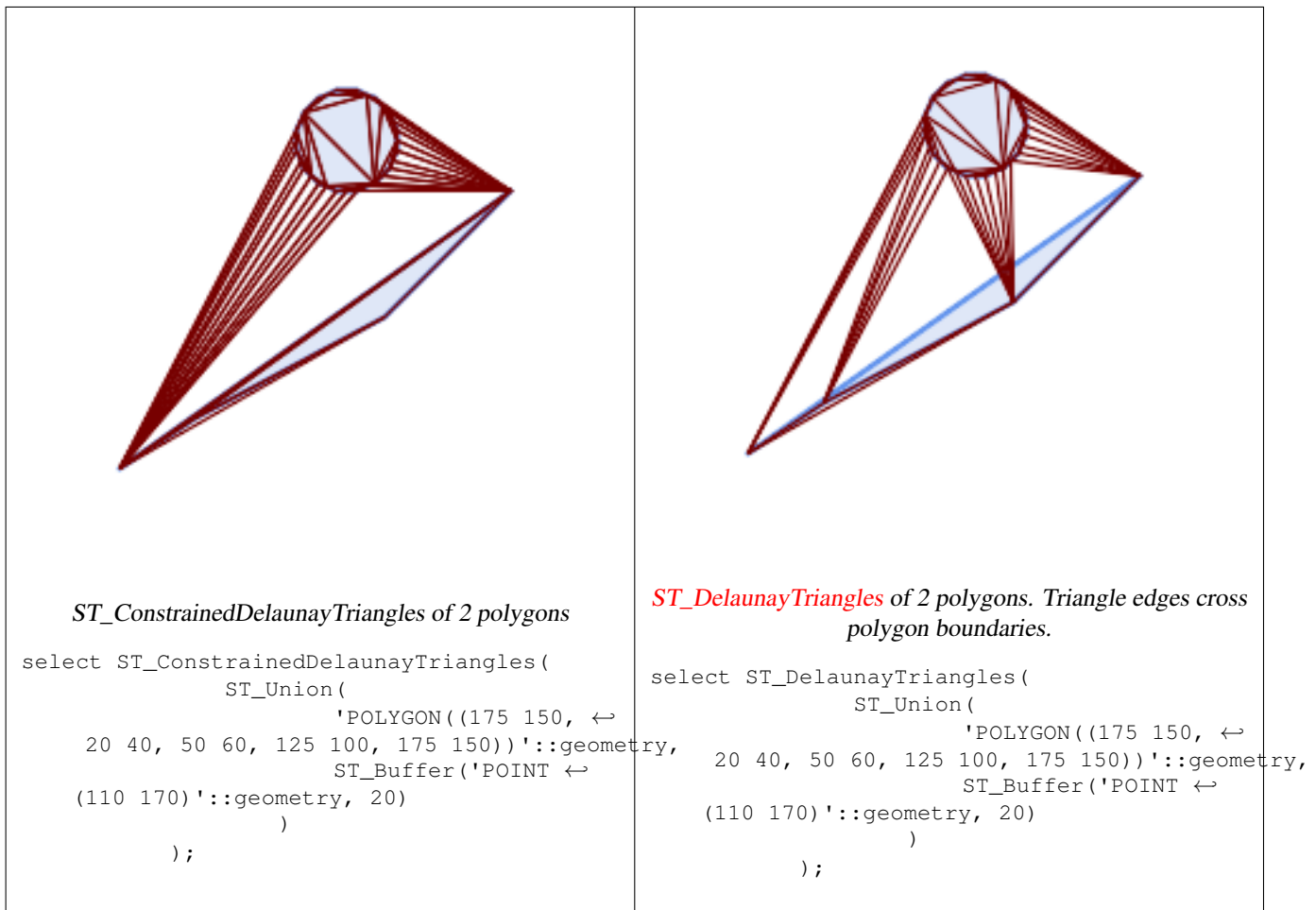
Return a **Constrained Delaunay triangulation** around the vertices of the input geometry. Output is a TIN.



2.1.0 버전부터 사용할 수 있습니다.



예시



ST_DelaunayTriangles

ST_DelaunayTriangles, ST_MakeSolid, ST_IsSolid, ST_Area

8.20.11 ST_Extrude

ST_Extrude — Extrude a 2D geometry into 3D space.

Synopsis

ST_Extrude(geometry geom, float x, float y, float z);

ST_Extrude

2.1.0 – This function supports Polyhedral surfaces.



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).

Buffer and Extrude

PostGIS **ST_AsX3D** returns a 3D geometry from a 2D geometry. The **ST_AsX3D** function takes a 2D geometry and returns a 3D geometry. The **ST_AsX3D** function is used to create a 3D geometry from a 2D geometry. The **ST_AsX3D** function is used to create a 3D geometry from a 2D geometry. The **ST_AsX3D** function is used to create a 3D geometry from a 2D geometry.

```
SELECT ST_Buffer(ST_GeomFromText('POINT (100 90)'),
50, 'quad_segs=2', 0, 0, 30);
```



The 2D buffer of a point is a light blue octagon with a blue outline. The buffer is created using the ST_Buffer function with a radius of 50 and 2 quadrants per segment.

```
ST_Extrude(ST_Buffer(ST_GeomFromText('POINT (100 90)'),
50, 'quad_segs=2'), 0, 0, 30);
```



The 3D extruded buffer of a point is a green 3D octagonal prism. The prism is created using the ST_Extrude function with a height of 30. The 3D geometry is returned as a PolyhedralSurfaceZ.

Synopsis

boolean **ST_IsPlanar**(geometry geom);

2.2.0

2.2.0: This function checks if the geometry is planar. It returns true if the geometry is planar and false otherwise. It is a wrapper around the SFCGAL function `ST_IsPlanar`. It is available since version 2.2.0. It is a wrapper around the SFCGAL function `ST_IsPlanar`. It is available since version 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).

8.20.14 ST_IsSolid

ST_IsSolid — Checks if the geometry is a solid. It returns true if the geometry is a solid and false otherwise. It is a wrapper around the SFCGAL function `ST_IsSolid`. It is available since version 2.2.0.

Synopsis

boolean **ST_IsSolid**(geometry geom1);

2.2.0

2.2.0: This function checks if the geometry is a solid. It returns true if the geometry is a solid and false otherwise. It is a wrapper around the SFCGAL function `ST_IsSolid`. It is available since version 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).

8.20.15 ST_MakeSolid

ST_MakeSolid — Creates a solid from a geometry. It returns a solid geometry. It is a wrapper around the SFCGAL function `ST_MakeSolid`. It is available since version 2.2.0.

Synopsis

geometry **ST_MakeSolid**(geometry geom1);

ST_MinkowskiSum

2.2.0 This method needs SFCGAL backend.



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).

8.20.16 ST_MinkowskiSum

ST_MinkowskiSum — Returns the Minkowski sum of two geometries.

Synopsis

geometry **ST_MinkowskiSum**(geometry geom1, geometry geom2);

ST_MinkowskiSum

ST_MinkowskiSum(geom1, geom2) returns the Minkowski sum of the two input geometries. The Minkowski sum of two geometries is the set of all points that can be formed by adding the coordinates of the two geometries. For example, the Minkowski sum of a point and a line segment is a line segment of the same length and orientation, but translated to the position of the point.

The Minkowski sum of two polygons is a new polygon that is the union of all the points that can be formed by adding the coordinates of the two polygons. This is useful for many applications, such as collision detection and path planning. The Minkowski sum of two polygons can be computed using the SFCGAL backend, which is available in PostGIS 3.3.0 and later.

The Minkowski sum of two polygons can be computed using the SFCGAL backend, which is available in PostGIS 3.3.0 and later. The Minkowski sum of two polygons is a new polygon that is the union of all the points that can be formed by adding the coordinates of the two polygons. This is useful for many applications, such as collision detection and path planning. The Minkowski sum of two polygons can be computed using the SFCGAL backend, which is available in PostGIS 3.3.0 and later.

The Minkowski sum of two polygons can be computed using the SFCGAL backend, which is available in PostGIS 3.3.0 and later. The Minkowski sum of two polygons is a new polygon that is the union of all the points that can be formed by adding the coordinates of the two polygons. This is useful for many applications, such as collision detection and path planning. The Minkowski sum of two polygons can be computed using the SFCGAL backend, which is available in PostGIS 3.3.0 and later.

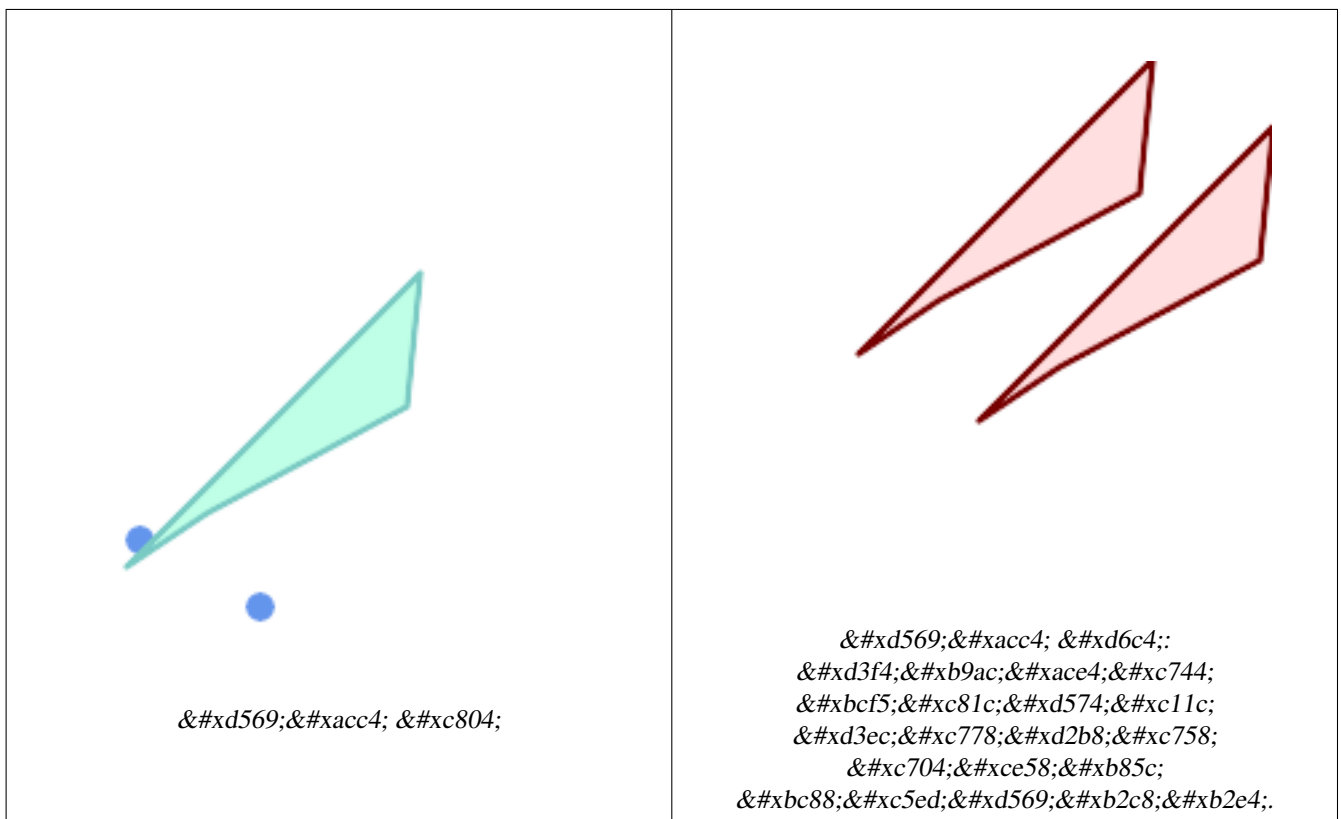
2.1.0 This method needs SFCGAL backend.



This method needs SFCGAL backend.

ST_MinkowskiSum

ST_MinkowskiSum(geom1, geom2) returns the Minkowski sum of the two input geometries. The Minkowski sum of two geometries is the set of all points that can be formed by adding the coordinates of the two geometries. For example, the Minkowski sum of a point and a line segment is a line segment of the same length and orientation, but translated to the position of the point.



```
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))
)
```

8.20.17 ST_OptimalAlphaShape

ST_OptimalAlphaShape — Computes a possible concave geometry using the CGAL Alpha Shapes algorithm after have computed the "optimal" alpha value.

Synopsis

geometry**ST_OptimalAlphaShape**(**geometry** param_geom, boolean allow_holes = false, integer nb_components);

Parameters

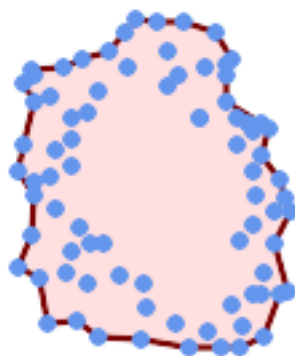
Computes the "optimal" alpha-shapes of the set of geometry. CGAL can automatically find the optimal value of alpha. This version uses it to find an "optimal" alpha-shape. The result is a single polygon. It will not contain holes unless the optional `param_allow_holes` argument is specified as true. The result will be generated such that the number of solid component of the alpha shape is equal to or smaller than `param_nb_components`.

Availability: 3.3.0 - requires SFCGAL >= 1.4.1.



This method needs SFCGAL backend.

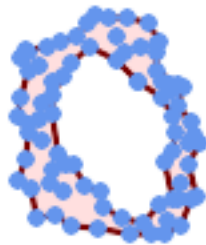
Concave Hull of a MultiPoint



Concave Hull of a MultiPoint (same example as ST_AlphaShape)

```
SELECT ST_AsText(ST_OptimalAlphaShape('MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50) ←
    ,(81 70),
    (88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 ←
    30),(36 61),(32 65),
    (81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
    (78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 ←
    29),(27 84),(52 98),(72 95),(85 71),
    (75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 ←
    97),(27 77),(39 88),(60 81),
    (80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 ←
    64),(69 86),(60 90),(50 86),(43 80),(36 73),
    (36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 ←
    16),(38 46),(31 59),(34 86),(45 90),(64 97))'::geometry));

POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,37 23,30 22,28 ←
    33,23 36,
    26 44,27 54,23 60,24 67,27 77,24 82,26 85,34 86,39 88,45 90,49 95,52 98,57 ←
    97,64 97,72 95,76 88,75 84,75 77,83 72,85 71,83 64,88 58,89 53))
```



Concave Hull of a MultiPoint, allowing holes (same example as ST_AlphaShape)

```
SELECT ST_AsText(ST_OptimalAlphaShape('MULTIPOINT((63 84),(76 88),(68 73),(53 18),(91 50) ←
, (81 70),(88 29),(24 82),(32 51),(37 23),(27 54),(84 19),(75 87),(44 42),(77 67),(90 30) ←
, (36 61),(32 65),(81 47),(88 58),(68 73),(49 95),(81 60),(87 50),
(78 16),(79 21),(30 22),(78 43),(26 85),(48 34),(35 35),(36 40),(31 79),(83 ←
29),(27 84),(52 98),(72 95),(85 71),
(75 84),(75 77),(81 29),(77 73),(41 42),(83 72),(23 36),(89 53),(27 57),(57 ←
97),(27 77),(39 88),(60 81),
(80 72),(54 32),(55 26),(62 22),(70 20),(76 27),(84 35),(87 42),(82 54),(83 ←
64),(69 86),(60 90),(50 86),(43 80),(36 73),
(36 68),(40 75),(24 67),(23 60),(26 44),(28 33),(40 32),(43 19),(65 16),(73 ←
16),(38 46),(31 59),(34 86),(45 90),(64 97))'::geometry, allow_holes => ←
true));
```

```
POLYGON((89 53,91 50,87 42,90 30,88 29,84 19,78 16,73 16,65 16,53 18,43 19,37 23,30 22,28 ←
33,23 36,26 44,27 54,23 60,24 67,27 77,24 82,26 85,34 86,39 88,45 90,49 95,52 98,57 ←
97,64 97,72 95,76 88,75 84,75 77,83 72,85 71,83 64,88 58,89 53),(36 61,36 68,40 75,43 ←
80,50 86,60 81,68 73,77 67,81 60,82 54,81 47,78 43,81 29,76 27,70 20,62 22,55 26,54 ←
32,48 34,44 42,38 46,36 61))
```

స고

ST_ConcaveHull, ST_AlphaShape

8.20.18 ST_Orientation

ST_Orientation — 표면의 방향(orientation)을 결정합니&#

Synopsis

```
integer ST_Orientation(geometry geom);
```

설명

이 함수는 폴리곤에만 적용됩니&#x
폴리곤이 반시계 방향이면 -1을, 시&
방향이면 1을 반환합니다.

2.1.0 This method needs SFCGAL backend.



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.

8.20.19 ST_StraightSkeleton

ST_StraightSkeleton — (straight skeleton)

Synopsis

geometry ST_StraightSkeleton(geometry geom);

;

2.1.0 This method needs SFCGAL backend.



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



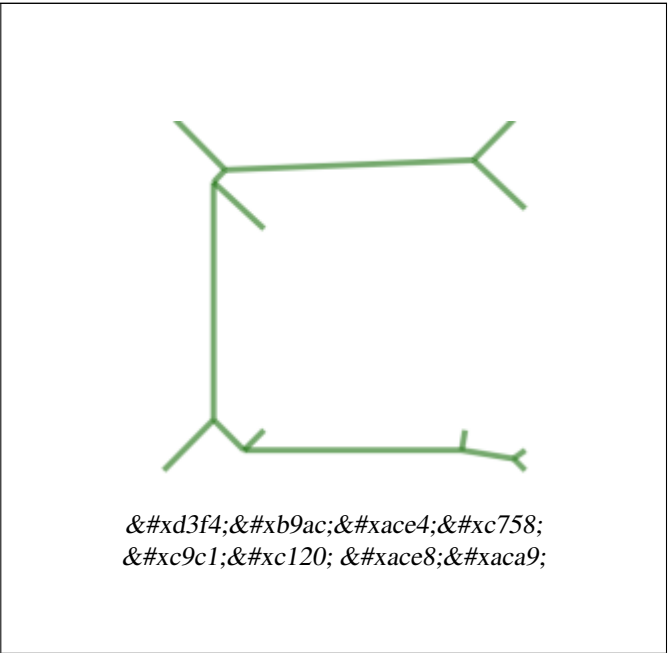
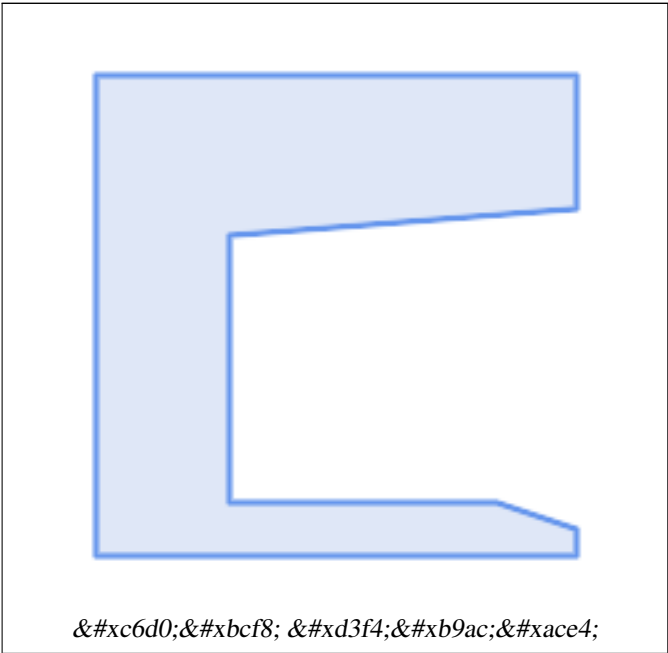
This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

;

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



;

8.20.20 ST_Tessellate

ST_Tessellate — Takes a geometry and tessellates it into a set of triangles. The tessellation is done using a Delaunay triangulation algorithm. The function returns a geometry of type TIN (Triangulated Irregular Network). The function is implemented using the SFCGAL backend.

Synopsis

geometry **ST_Tessellate**(geometry geom);

Parameters

geom: The geometry to be tessellated. It can be a point, line, polygon, or a collection of these. The function will tessellate the geometry into a set of triangles. The function is implemented using the SFCGAL backend.

2.1.0: The function was added in version 2.1.0 of the SFCGAL backend.



This method needs SFCGAL backend.



This function supports 3d and will not drop the z-index.



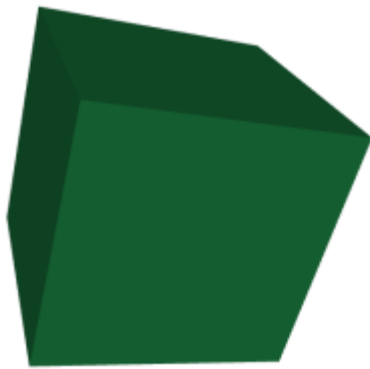
This function supports Polyhedral surfaces.



This function supports Triangles and Triangulated Irregular Network Surfaces (TIN).

See Also

```
SELECT ST_GeomFromText('POLYHEDRALSURFACE
  Z( ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
      ((0 0 0, 0 1 0, 1 1 0, 1
0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0
0 0, 1 1 0)), ((1 1 0, 1 1 1, 1 0 1, 1
0 0, 0 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
```



PostGIS 3.3.9dev

```
SELECT ST_Tessellate(ST_GeomFromText('
POLYHEDRALSURFACE Z( ((0 0 0, 0 0 1, 0 1 1, 0 1
((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0
0)), ((0 0 0, 1 0 0, 1 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_AsText

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



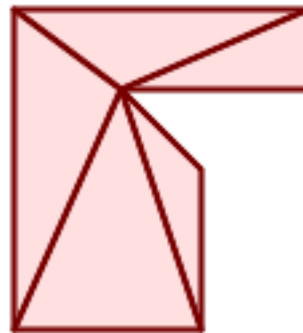
PostGIS 3.3.9dev

```
SELECT 'POLYGON (( 10 190, 10 70, 80 70, ↵
      80 130, 50 160, 120 160, 120 190, 10 190 ))'
```



원본 폴리곤

```
SELECT
  ST_Tessellate('POLYGON (( 10 190, ↵
    10 70, 80 70, 80 130, 50 160, 120 160, 120 190,
    ST_AsText &#xcd9c;&#xb825;&#xbb3c;:
    TINN6680130, 50 160, 80 70, 80 130)), ((50 ↵
    160, 10 190, 10 70, 50 160)),
    ((80 70, 50 160, 10 70, 80 70)) ↵
    , ((120 160, 120 190, 50 160, 120 160)),
    ((120 190, 10 190, 50 160, 120 190)))
```



다듬어진
폴리곤

참고

[ST_ConstrainedDelaunayTriangles](#), [ST_DelaunayTriangles](#)

8.20.21 ST_Volume

ST_Volume — 3차원 입체의 부피를 계산합&#xb2
표면 도형을 입력하면 (닫힌 도형&
0을 반환할 것입니다.

Synopsis

```
float ST_Volume(geometry geom1);
```

설명

2.2.0 버전부터 사용할 수 있습니다.



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).
- ✔ This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 9.1 (same as ST_3DVolume)

예시

WKT로닫힌표면을생성했을때입&#xc6a
 면으로󍳨급됩니다. 이닫힌표면입
 입체로만들려면, ST_MakeSolid 함수를이&#xc6a
 합니다. 면도형은부피&xac00;없습니다
 음은그사실을보여주는예시&#xc

```

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

```

స고

ST_3DArea, ST_MakeSolid, ST_IsSolid

8.21 장기 실행 트랜잭션(Long Transaction) 지원

Note



```
&#xc0ac;&#xc6a9;&#xc790;&#xb294;  &#xc9c1;&#xb82c;  &#xac00;&#xb2a5;  &#xd2b8;&#xb79c;&#xc7ad;&#xc158;
&#xb808;&#xbca8;(serializable      transaction      level)      &#xc744;      &#xc774;&#xc6a9;&#xd574;&#xc57c;
&#xd569;&#xb2c8;&#xb2e4;.      &#xadf8;&#xb807;&#xc9c0;      &#xc54a;&#xc73c;&#xba74;
&#xc7a0;&#xae08;      &#xae30;&#xc81c;(locking      mechanism)&#xac00;      &#xc911;&#xb2e8;&#xb420;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

8.21.1 AddAuth

AddAuth — %현재 %트랜잭션에 %이용될 %승인 %토큰을 %추가합니다.

Synopsis

```
boolean AddAuth(text auth_token);
```


설명

현재트랜잭션에이용될승인토&#
추가합니다.

Adds the current transaction identifier and authorization token to a temporary table called `temp_lock_have_table`.

1.1.3 ಄전부터 사용할 수 있습니다.

예시

```
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;

-- &#xc624;&#xb958; --
ERROR:  UPDATE where "gid" = '353' requires authorization 'priscilla'
```

స고

LockRow

8.21.2 CheckAuth

CheckAuth — 승인 토큰을 바탕으로, 테이&#xbe1
대해 행들의 업데이트 및 삭제를
금지/허락하는 트리거를 생성합&#

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);

설명

```
&#xc2b9;&#xc778; &#xd1a0;&#xd070;&#xc744; &#xbc14;&#xd0d5;&#xc73c;&#xb85c;, &#xd14c;&#xc774;&#xbe14;&#xc5d0;
&#xb300;&#xd574; &#xd589;&#xb4e4;&#xc758; &#xc5c5;&#xb370;&#xc774;&#xd2b8; &#xbc0f; &#xc0ad;&#xc81c;&#xb97c;
&#xae08;&#xc9c0;/&#xd5c8;&#xb77d;&#xd558;&#xb294; &#xd2b8;&#xb9ac;&#xac70;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#
<rowid_col> &#xc5f4;&#xc744; &#xc774;&#xc6a9;&#xd574;&#xc11c; &#xd589;&#xc744; &#xc2dd;&#xbcc4;&#xd569;&#xb2c8;d
a_schema_name&#xc744; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc73c;&#xba74;, &#xd604;&#xc7ac; &#xc2a4;&#xd0
&#xd14c;&#xc774;&#xbe14;&#xc744; &#xac80;&#xc0c9;&#xd569;&#xb2c8;&#xb2e4;.
```

Note



해당 테이블에 이미 승인
트리거가 존재할 경우, 함수가
오류를 일으킵니다.
트랜잭션 지원이 활성화돼
있지 않을 경우, 함수가 예외를
발생시킵니다.

1.1.3 버전부터 사용할 수 있습니다.

124;85;

```
&#xc7a5;&#xae30; &#xc2e4;&#xd589; &#xd2b8;&#xb79c;&#xc7ad;&#xc158; &#xc9c0;&#xc6d0;&#xc744; &#xd65c;&#xc131;&#xc774; &#xd568;&#xc218;&#xb294; &#xd544;&#xc218;&#xc801;&#xc778; &#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130; &#xd14c;&#xc774;&#xbe14;&#xb4e4;&#xc744; &#xc0dd;&#xc131;&#xd558;&#xb294;&#xb370;, &#xc774; &#xb2e8;&#xc6d0;&#xc788;&#xb294; &#xb2e4;&#xb978; &#xd568;&#xc218;&#xb4e4;&#xc744; &#xc774;&#xc6a9;&#xd558;&#xae30; &#xc804;&#xd55c;&#xbc88; &#xd638;&#xcd9c;&#xd574;&#xc918;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;. &#xb450; &#xbc88; &#xd638;&#xc0c1;&#xad00;&#xc5c6;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
authorization_table &#xc774;&#xb77c;&#xb294; &#xba54;&#xd0c0; &#xd14c;&#xc774;&#xbe14; &#xbc0f; authorized &#xb77c;&#xb294; &#xbdf0;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
```

1.1.3 버전부터 사용할 수 있습니다.

608;2dc;

```
SELECT EnableLongTransactions();
-- &#xacb0;&#xacfc; --
Long transactions support enabled
```

cc38;ace0;**DisableLongTransactions****8.21.5 LockRow**

LockRow — 테이블 내부의 특정 행에 대한잠금/승인을 설정합니다.

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);
```

124;85;

```
&#xd14c;&#xc774;&#xbe14; &#xb0b4;&#xbd80;&#xc758; &#xd2b9;&#xc815; &#xd589;&#xc5d0; &#xb300;&#xd55c; &#xc7a0;&#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;. <authid>&#xb294; &#xd14d;&#xc2a4;&#xd2b8; &#xac12;, <expires>&#xb294; &#xae30;&#xbcf8;&#xac12;&#xc774; now()+1&#xc2dc;&#xac04;&#xc778; &#xd0c0;&#xc784;&#xc2a4;&#xd0ec;&#xd504;&#xc7 &#xc7a0;&#xae08;&#xc774; &#xd560;&#xb2f9;&#xb41c; &#xacbd;&#xc6b0; 1&#xc744; &#xbc18;&#xd658;&#xd558;&#xae0;, &#xadf8;&#xb807;&#xc9c0; &#xc54a;&#xc744; (&#xc774;&#xbbf8; &#xb2e4;&#xb978; &#xc2b9;&#xc778;&#xc744; &#xbc1b;&#xc7a0;&#xaca8; &#xc788;&#xb294;) &#xacbd;&#xc6b0; 0&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

1.1.3 버전부터 사용할 수 있습니다.

608;2dc;

```
SELECT LockRow('public', 'towns', '2', 'joey');
LockRow
-----
1
```

```
--
--      &#xc870;&#xc774;&#xac00; &#xc774;&#xbbf8; &#xb808;&#xcf54;&#xb4dc;&#xb97c;  ↵
--      &#xc7a0;&#xad9c;&#xae30; &#xb54c;&#xbb38;&#xc5d0;  ↵
--      &#xd50c;&#xb9ac;&#xc2e4;&#xb77c;&#xac00; &#xc7a0;&#xae00; &#xc218;  ↵
--      &#xc5c6;&#xc2b5;&#xb2c8;&#xb2e4;.
SELECT LockRow('public', 'towns', '2', 'priscilla');
LockRow
-----
0
```

참고

UnlockRows

8.21.6 UnlockRows

UnlockRows — Removes all locks held by an authorization token.

Synopsis

integer **UnlockRows**(text auth_token);

설명

```
&#xd2b9;&#xc815; &#xc2b9;&#xc778; ID&#xac00; &#xc2e4;&#xd589;&#xd55c; &#xaa8;&#xb4e0; &#xc7a0;&#xae08;&#xc744;
&#xc81c;&#xac70;&#xd569;&#xb2c8;&#xb2e4;. &#xd574;&#xc81c;&#xb41c; &#xc7a0;&#xae08; &#xac1c;&#xc218;&#xb97c;
&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

1.1.3 버전부터 사용할 수 있습니다.

예시

```
SELECT LockRow('towns', '353', 'priscilla');
      SELECT LockRow('towns', '2', 'priscilla');
      SELECT UnLockRows('priscilla');
      UnLockRows
      -----
      2
```

참고

LockRow

8.22 Version Functions

8.22.1 PostGIS_Extensions_Upgrade

PostGIS_Extensions_Upgrade — Packages and upgrades PostGIS extensions (e.g. postgis_raster, postgis_topology, postgis_sfcgal) to latest available version.

Synopsis

text **PostGIS_Extensions_Upgrade()**;

Description

Packages and upgrades PostGIS extensions to latest version. Only extensions you have installed in the database will be packaged and upgraded if needed. Reports full PostGIS version and build configuration infos after. This is short-hand for doing multiple CREATE EXTENSION .. FROM unpackaged and ALTER EXTENSION .. UPDATE for each PostGIS extension. Currently only tries to upgrade extensions postgis, postgis_raster, postgis_sfcgal, postgis_topology, and postgis_tiger_geocoder.

Availability: 2.5.0



Note

Changed: 3.3.0 support for upgrades from any PostGIS version. Does not work on all systems.
Changed: 3.0.0 to repack loose extensions and support postgis_raster.

Examples

```
SELECT PostGIS_Extensions_Upgrade();
```

```
NOTICE: Packaging extension postgis
NOTICE: Packaging extension postgis_raster
NOTICE: Packaging extension postgis_sfcgal
NOTICE: Extension postgis_topology is not available or not packagable for some reason
NOTICE: Extension postgis_tiger_geocoder is not available or not packagable for some reason
      reason
      postgis_extensions_upgrade
-----
Upgrade completed, run SELECT postgis_full_version(); for details
(1 row)
```

See Also

Section 3.4, [PostGIS_GEOS_Version](#), [PostGIS_Lib_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_PROJ_Version](#), [PostGIS_Version](#)

8.22.2 PostGIS_Full_Version

PostGIS_Full_Version — Reports full PostGIS version and build configuration infos.

Synopsis

text **PostGIS_Full_Version()**;

Description

Reports full PostGIS version and build configuration infos. Also informs about synchronization between libraries and scripts suggesting upgrades as needed.

Examples

```
SELECT PostGIS_Full_Version();
                                     postgis_full_version
-----
PostGIS="3.0.0dev r17211" [EXTENSION] PGSQL="110" GEOS="3.8.0dev-CAPI-1.11.0 df24b6bb" ↔
SFCGAL="1.3.6" PROJ="Rel. 5.2.0, September 15th, 2018"
GDAL="GDAL 2.3.2, released 2018/09/21" LIBXML="2.9.9" LIBJSON="0.13.1" LIBPROTOBUF="1.3.1" ↔
WAGYU="0.4.3 (Internal)" TOPOLOGY RASTER
(1 row)
```

See Also

Section 3.4, [PostGIS_GEOS_Version](#), [PostGIS_Lib_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_PROJ_Version](#), [PostGIS_Wagyu_V](#), [PostGIS_Version](#)

8.22.3 PostGIS_GEOS_Version

`PostGIS_GEOS_Version` — Returns the version number of the GEOS library.

Synopsis

```
text PostGIS_GEOS_Version();
```

Description

Returns the version number of the GEOS library, or NULL if GEOS support is not enabled.

Examples

```
SELECT PostGIS_GEOS_Version();
       postgis_geos_version
-----
3.11.0-CAPI-1.5.0
(1 row)
```

See Also

[PostGIS_Full_Version](#), [PostGIS_Lib_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_PROJ_Version](#), [PostGIS_Version](#)

8.22.4 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();
```

Description

Returns the version number of the liblwgeom library/

Examples

```
SELECT PostGIS_Liblwgeom_Version();
postgis_liblwgeom_version
-----
2.3.3 r15473
(1 row)
```

See Also

[PostGIS_Full_Version](#), [PostGIS_Lib_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_PROJ_Version](#), [PostGIS_Version](#)

8.22.5 PostGIS_LibXML_Version

`PostGIS_LibXML_Version` — Returns the version number of the libxml2 library.

Synopsis

text `PostGIS_LibXML_Version()`;

Description

Returns the version number of the LibXML2 library.

Availability: 1.5

Examples

```
SELECT PostGIS_LibXML_Version();
postgis_libxml_version
-----
2.7.6
(1 row)
```

See Also

[PostGIS_Full_Version](#), [PostGIS_Lib_Version](#), [PostGIS_PROJ_Version](#), [PostGIS_GEOS_Version](#), [PostGIS_Version](#)

8.22.6 PostGIS_Lib_Build_Date

`PostGIS_Lib_Build_Date` — Returns build date of the PostGIS library.

Synopsis

text `PostGIS_Lib_Build_Date()`;

Description

Returns build date of the PostGIS library.

Examples

```
SELECT PostGIS_Lib_Build_Date();
 postgis_lib_build_date
-----
2008-06-21 17:53:21
(1 row)
```

8.22.7 PostGIS_Lib_Version

`PostGIS_Lib_Version` — Returns the version number of the PostGIS library.

Synopsis

text `PostGIS_Lib_Version()`;

Description

Returns the version number of the PostGIS library.

Examples

```
SELECT PostGIS_Lib_Version();
 postgis_lib_version
-----
1.3.3
(1 row)
```

See Also

[PostGIS_Full_Version](#), [PostGIS_GEOS_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_PROJ_Version](#), [PostGIS_Version](#)

8.22.8 PostGIS_PROJ_Version

`PostGIS_PROJ_Version` — Returns the version number of the PROJ4 library.

Synopsis

text `PostGIS_PROJ_Version()`;

Description

Returns the version number of the PROJ4 library, or NULL if PROJ4 support is not enabled.

Examples

```
SELECT PostGIS_PROJ_Version();
 postgis_proj_version
-----
Rel. 4.4.9, 29 Oct 2004
(1 row)
```


See Also

[PostGIS_Full_Version](#), [PostGIS_GEOS_Version](#), [PostGIS_Lib_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_Version](#)

8.22.9 PostGIS_Wagyu_Version

`PostGIS_Wagyu_Version` — Returns the version number of the internal Wagyu library.

Synopsis

text `PostGIS_Wagyu_Version()`;

Description

Returns the version number of the internal Wagyu library, or `NULL` if Wagyu support is not enabled.

Examples

```
SELECT PostGIS_Wagyu_Version();
 postgis_wagyu_version
-----
 0.4.3 (Internal)
(1 row)
```

See Also

[PostGIS_Full_Version](#), [PostGIS_GEOS_Version](#), [PostGIS_PROJ_Version](#), [PostGIS_Lib_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_Version](#)

8.22.10 PostGIS_Scripts_Build_Date

`PostGIS_Scripts_Build_Date` — Returns build date of the PostGIS scripts.

Synopsis

text `PostGIS_Scripts_Build_Date()`;

Description

Returns build date of the PostGIS scripts.

Availability: 1.0.0RC1

Examples

```
SELECT PostGIS_Scripts_Build_Date();
 postgis_scripts_build_date
-----
 2007-08-18 09:09:26
(1 row)
```

See Also

[PostGIS_Full_Version](#), [PostGIS_GEOS_Version](#), [PostGIS_Lib_Version](#), [PostGIS_LibXML_Version](#), [PostGIS_Version](#)

8.22.11 PostGIS_Scripts_Installed

PostGIS_Scripts_Installed — Returns version of the PostGIS scripts installed in this database.

Synopsis

text **PostGIS_Scripts_Installed**();

Description

Returns version of the PostGIS scripts installed in this database.

**Note**

If the output of this function doesn't match the output of [PostGIS_Scripts_Released](#) you probably missed to properly upgrade an existing database. See the [Upgrading](#) section for more info.

Availability: 0.9.0

Examples

```
SELECT PostGIS_Scripts_Installed();
 postgis_scripts_installed
-----
1.5.0SVN
(1 row)
```

See Also

[PostGIS_Full_Version](#), [PostGIS_Scripts_Released](#), [PostGIS_Version](#)

8.22.12 PostGIS_Scripts_Released

PostGIS_Scripts_Released — Returns the version number of the postgis.sql script released with the installed PostGIS lib.

Synopsis

text **PostGIS_Scripts_Released**();

Description

Returns the version number of the postgis.sql script released with the installed PostGIS lib.

**Note**

Starting with version 1.1.0 this function returns the same value of [PostGIS_Lib_Version](#). Kept for backward compatibility.

Availability: 0.9.0

2.2.0



Note

GDAL
GDAL_DATA
수 있 습 니 다.

예 시

postgis.gdal_datapath 를 설 정 한 다 음 재 설 정 해 &

```
SET postgis.gdal_datapath TO '/usr/local/share/gdal.hidden';
SET postgis.gdal_datapath TO default;
```

특 정 데 이 터 베 이 스 를 대 상 으 로
윈 도 우 상 에 서 설 정 해 보 십 시 오.

```
ALTER DATABASE gisdb
SET postgis.gdal_datapath = 'C:/Program Files/PostgreSQL/9.3/gdal-data';
```

참 고

PostGIS_GDAL_Version, ST_Transform

8.23.3 postgis.gdal_enabled_drivers

postgis.gdal_enabled_drivers — PostGIS 환 경 에 서 사 용 할 수 있 &#
GDAL 드 라 이 버 를 설 정 하 는 설 정 옵 &#xc
GDAL 설 정 변 수 GDAL_SKIP 에 영 향 을 미 칩 니 &#

설 명

PostGIS 환 경 에 서 사 용 할 수 있 는 GDAL 드 &#xb7
설 정 하 는 설 정 옵 션 입 니 다. GDAL 설 &#xc
변 수 GDAL_SKIP 에 영 향 을 미 칩 니 다. PostgreSQL &#xc7
설 정 파 일 postgresql.conf 안 에 서 이 옵 션 을
설 정 할 수 있 습 니 다. 또 연 결 이 나
상 호 처 리 단 계 에 서 도 설 정 할 수
있 습 니 다.

PostgreSQL 을 구 동 시 키 는 과 정 에 서 사 용 &#
수 있 는 드 라 이 버 목 록 과 함 께 환 &#
변 수 POSTGIS_GDAL_ENABLED_DRIVERS 를 패 스 (pass) 시 켜 postgis.gdal
의 초 기 값 을 설 정 할 수 도 있 습 니 &#
드 라 이 버 의 축 약 명 또 는 코 드 를
통 해 활 성 화 된 GDAL 특 화 드 라 이 버 &#xb
지 정 할 수 있 습 니 다. 드 라 이 버 의
축 약 명 또 는 코 드 는 GDAL 래 스 터 형 &#xc
에 서 찾 을 수 있 습 니 다. 각 드 라 이 &
사 이 에 공 백 을 삽 입 하 면 복 수 의
드 라 이 버 를 지 정 할 수 있 습 니 다.

Note

```
postgis.gdal_enabled_drivers &#xb97c; &#xc704;&#xd574; &#xc0ac;&#xc6a9;&#xd560;
&#xc218; &#xc788;&#xb294; &#xd2b9;&#xbcc4; &#xc54;&#xb4dc;&#xac00; &#xc138;
&#xac1c; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc774; &#xc54;&#xb4dc;&#xb4e4;&#xc740;
&#xb300;&#xc18c;&#xbb38;&#xc790;&#xb97c; &#xad6c;&#xbd84;&#xd569;&#xb2c8;&#xb2e4;.
```

- `DISABLE_ALL` 은 모든 `GDAL` 드라이஄를 비활성화시킵니다. `DISABLE_ALL` 이 있을 경우, `postgis.gdal_enabled_drivers` 안에 있는 다른 모든 값을 무시합니다.

- `ENABLE_ALL` 은 모든 `GDAL` 드라이஄를 활성화시킵니다.

- `VSICURL` 은 `GDAL`의 `/vsicurl/` 가상 파일 시스템을 활성화시킵니다.

```
postgis.gdal_enabled_drivers &#xac00; DISABLE_ALL&#xb85c; &#xc131;&#xc815;&#xb410;&#xc744;
&#xacbd;&#xc6b0;, DB &#xc678;&#xbd80; &#xb798;&#xc2a4;&#xd130;, ST_FromGDALRaster(),
ST_AsGDALRaster(), ST_AsTIFF(), ST_AsJPEG() &#xadf8;&#xb9ac;&#xace0; ST_AsPNG()&#xb97c;
&#xc0ac;&#xc6a9;&#xd558;&#xb824; &#xd55c;&#xb2e4;&#xba74; &#xbaa8;&#xb450; &#xc624;&#xb958;
&#xba54;&#xc2dc;&#xc9c0;&#xb97c; &#xbcf4;&#xac8c; &#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

**Note**

표준 PostGIS 설ౘ시, `postgis.gdal_enabled_drivers` 는 `DISABLE_ALL`로 설정됩니다.

**Note**

`GDAL_SKIP`에 대한 추가 정보는 `GDAL`의 **Configuration Options** 에서 찾아보 수 있습니다.

2.2.0 ஄전부터 사용할 수 있습니다.

예시

```
postgis.gdal_enabled_drivers &#xb97c; &#xc124;&#xc815;&#xd55c; &#xb2e4;&#xc74c; &#xc7ac;&#xc124;&#xc815;&#xb370;
&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xc640;&#xc758; &#xbaa8;&#xb4e0; &#xc0c8; &#xc5f0;&#xacb0;&#xb31;
&#xc5d4;&#xb4dc;&#xb97c; &#xc124;&#xc815;&#xd574;&#xbd05;&#xc2dc;&#xb2e4;.
```

```
ALTER DATABASE mygisdb SET postgis.gdal_enabled_drivers TO 'GTiff PNG JPEG';
```

```
&#xc11c;&#xb84;&#xc640;&#xc758; &#xbaa8;&#xb4e0; &#xc0c8; &#xc5f0;&#xacb0;&#xc5d0; &#xae30;&#xbcf8; &#xd65c;&#xb4dc;
&#xb77c;&#xc774;&#xb84;&#xb97c; &#xc124;&#xc815;&#xd574;&#xbd05;&#xc2dc;&#xb2e4;. &#xc288;&#xd37c;&#xad8c;&#xd55c;
&#xbcf0; PostgreSQL 9.4 &#xc774;&#xc0c1; &#xb84;&#xc804;&#xc774; &#xd544;&#xc694;&#xd569;&#xb2 &#xb610;
&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;, &#xc138;&#xc158;, &#xc0ac;&#xc6a9;&#xc790; &#xc124;&#xb2e8;
&#xacc4;&#xc5d0;&#xc11c; &#xc774; &#xc124;&#xc815;&#xc744; &#xbb34;&#xc2dc;&#xd560; &#xc218; &#xc5c6;&#xc744;
```

```
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;
```

```
GDAL_ENABLED_DRIVERS = 'ENABLE_ALL';
```

```
GDAL_ENABLED_DRIVERS = 'DISABLE_ALL';
```

ST_FromGDALRaster, ST_AsGDALRaster, ST_AsTIFF, ST_AsPNG, ST_AsJPEG, postgis.enable_outdb_rasters

8.23.4 postgis.enable_outdb_rasters

postgis.enable_outdb_rasters — DB **ENABLE_OUTDB_RASTERS** (PostgreSQL 12.0 and later) or **GDAL_ENABLED_DRIVERS** (PostgreSQL 11.0 and later) configuration parameter.

ENABLE_OUTDB_RASTERS

DB **ENABLE_OUTDB_RASTERS** (PostgreSQL 12.0 and later) or **GDAL_ENABLED_DRIVERS** (PostgreSQL 11.0 and later) configuration parameter. The parameter controls whether the GDAL drivers are enabled for out-of-database raster operations. The default value is **ENABLE_ALL**. The parameter can be set to **ENABLE_ALL**, **DISABLE_ALL**, or a comma-separated list of driver names. The parameter is only used for out-of-database raster operations. The parameter is not used for in-database raster operations.

PostgreSQL **GDAL_ENABLED_DRIVERS** (PostgreSQL 11.0 and later) configuration parameter. The parameter controls whether the GDAL drivers are enabled for out-of-database raster operations. The default value is **ENABLE_ALL**. The parameter can be set to **ENABLE_ALL**, **DISABLE_ALL**, or a comma-separated list of driver names. The parameter is only used for out-of-database raster operations. The parameter is not used for in-database raster operations.



Note

postgis.enable_outdb_rasters (PostgreSQL 12.0 and later) or **GDAL_ENABLED_DRIVERS** (PostgreSQL 11.0 and later) configuration parameter. The parameter controls whether the GDAL drivers are enabled for out-of-database raster operations. The default value is **ENABLE_ALL**. The parameter can be set to **ENABLE_ALL**, **DISABLE_ALL**, or a comma-separated list of driver names. The parameter is only used for out-of-database raster operations. The parameter is not used for in-database raster operations.



Note

GDAL_ENABLED_DRIVERS (PostgreSQL 11.0 and later) configuration parameter. The parameter controls whether the GDAL drivers are enabled for out-of-database raster operations. The default value is **ENABLE_ALL**. The parameter can be set to **ENABLE_ALL**, **DISABLE_ALL**, or a comma-separated list of driver names. The parameter is only used for out-of-database raster operations. The parameter is not used for in-database raster operations.

2.2.0 **GDAL_ENABLED_DRIVERS** (PostgreSQL 11.0 and later) configuration parameter. The parameter controls whether the GDAL drivers are enabled for out-of-database raster operations. The default value is **ENABLE_ALL**. The parameter can be set to **ENABLE_ALL**, **DISABLE_ALL**, or a comma-separated list of driver names. The parameter is only used for out-of-database raster operations. The parameter is not used for in-database raster operations.

ENABLE_OUTDB_RASTERS

```
SET postgis.enable_outdb_rasters TO True;
SET postgis.enable_outdb_rasters = default;
SET postgis.enable_outdb_rasters = True;
SET postgis.enable_outdb_rasters = False;
```

Set for specific database

```
ALTER DATABASE mygisdb SET postgis.backend = sfcgal;
```

Setting for whole database cluster. You need to reconnect to the database for changes to take effect.

```
--writes to postgres.auto.conf
ALTER SYSTEM postgis.enable_outdb_rasters = true;
--Reloads postgres conf
SELECT pg_reload_conf();
```

postgis.gdal_enabled_drivers

postgis.gdal_vsi_options

8.23.5 postgis.gdal_vsi_options

postgis.gdal_vsi_options — DB configuration parameter for GDAL VSI options.

postgis.gdal_vsi_options

A string configuration to set options used when working with an out-db raster. **Configuration options** control things like how much space GDAL allocates to local data cache, whether to read overviews, and what access keys to use for remote out-db data sources.

2.2.0 **postgis.gdal_vsi_options**

postgis.gdal_vsi_options

```
postgis.enable_outdb_rasters &#x2013; postgis.gdal_vsi_options
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxx AWS_SECRET_ACCESS_KEY=
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY';
```

Set **postgis.gdal_vsi_options** just for the *current transaction* using the **LOCAL** keyword:

```
SET LOCAL postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY';
```

postgis.enable_outdb_rasters

postgis.gdal_enabled_drivers

8.24 Troubleshooting Functions

8.24.1 PostGIS_AddBBox

PostGIS_AddBBox — Add bounding box to the geometry.

Synopsis

PostGIS_AddBBox(geometry geomA);

Description

Add bounding box to the geometry. This would make bounding box based queries faster, but will increase the size of the geometry.



Note

Bounding boxes are automatically added to geometries so in general this is not needed unless the generated bounding box somehow becomes corrupted or you have an old install that is lacking bounding boxes. Then you need to drop the old and readd.



This method supports Circular Strings and Curves

Examples

```
UPDATE sometable
SET geom = PostGIS_AddBBox(geom)
WHERE PostGIS_HasBBox(geom) = false;
```

See Also

[PostGIS_DropBBox](#), [PostGIS_HasBBox](#)

8.24.2 PostGIS_DropBBox

PostGIS_DropBBox — Drop the bounding box cache from the geometry.

Synopsis

geometry **PostGIS_DropBBox**(geometry geomA);

Description

Drop the bounding box cache from the geometry. This reduces geometry size, but makes bounding-box based queries slower. It is also used to drop a corrupt bounding box. A tale-tell sign of a corrupt cached bounding box is when your ST_Intersects and other relation queries leave out geometries that rightfully should return true.



Note

Bounding boxes are automatically added to geometries and improve speed of queries so in general this is not needed unless the generated bounding box somehow becomes corrupted or you have an old install that is lacking bounding boxes. Then you need to drop the old and readd. This kind of corruption has been observed in 8.3-8.3.6 series whereby cached bboxes were not always recalculated when a geometry changed and upgrading to a newer version without a dump reload will not correct already corrupted boxes. So one can manually correct using below and readd the bbox or do a dump reload.



This method supports Circular Strings and Curves

Examples

```
--This example drops bounding boxes where the cached box is not correct
--The force to ST_AsBinary before applying Box2D forces a ↔
  recalculation of the box, and Box2D applied to the table ↔
  geometry always
  -- returns the cached bounding box.
  UPDATE sometable
SET geom = PostGIS_DropBBox(geom)
WHERE Not (Box2D(ST_AsBinary(geom)) = Box2D(geom));

  UPDATE sometable
SET geom = PostGIS_AddBBox(geom)
WHERE Not PostGIS_HasBBOX(geom);
```

See Also

[PostGIS_AddBBox](#), [PostGIS_HasBBox](#), [Box2D](#)

8.24.3 PostGIS_HasBBox

PostGIS_HasBBox — Returns TRUE if the bbox of this geometry is cached, FALSE otherwise.

Synopsis

boolean **PostGIS_HasBBox**(geometry geomA);

Description

Returns TRUE if the bbox of this geometry is cached, FALSE otherwise. Use [PostGIS_AddBBox](#) and [PostGIS_DropBBox](#) to control caching.



This method supports Circular Strings and Curves

Examples

```
SELECT geom
FROM sometable WHERE PostGIS_HasBBox(geom) = false;
```

See Also

[PostGIS_AddBBox](#), [PostGIS_DropBBox](#)

Chapter 9

PostGIS 자주 묻는 &#xc

1. *PostGIS 활용에 대한 예제,지침서,워크등을 어디에서 찾아௏ 수 있을까*
 A step by step tutorial guide workshop [Introduction to PostGIS](#). It includes packaged data as well as intro to working with OpenGeo Suite. It is probably the best tutorial on PostGIS.[BostonGIS에서도 PostGIS:멍청이도 할 수 있는 지침서 를 제공합니다,이 지침서는 윈도우사용자를 더 중시하고 있습니다](#)
2. *PostGIS 1.5에서 돌아가던 응용 프로그데스크톱 툴들이 PostGIS 2.0에서 동작ꫀ않습니다,어떻게 고칠 수 있을까 더 이상 지원되지 않는 상당수응함수들이 PostGIS 2.0의 코드 기ଘ에서 제거되었습니다,이 것은 GeoServer나 MapServer, QuantumGIS, 그리고 OpenJump와 같은 몇몇 third-party 툴들 뿐만아니라,많은 응용 프영향을 주고 있습니다,이 것을 해결하는 두가지 방법이 있습니 thrid-party 응용들의 경우,당신은 이 이짎은 부분들이 수정된 최신버전업그레이드 하는 방법을 시도함수 있습니다,당신이 직접 개발함코드인 경우,당신의 코드가 더이 제거된 함수들을 사용하지 않도 수정할 수 있습니다,이 함수들응대부분은 ST_Union, ST_Length의 이전 버전들응 ST_를 사용하지 않는 것들일 것입마지막 수단으로,을 설치하당신이 필요한 만큼 legacy . sql의 일부수정하십시오,을 legacy . sql파일은 postgis.sql. 과 같은 폴더에 위치하고 있습니 제거되었던 총 200개의 조금은 이구버전 함수를 얻기 위해 postgis.sql그를 spatial_ref_sys.sql설치하신 뒤 해당 파일을 설 됩니다.*
3. *osm2pgsql을 통해 OpenStreetMap 데이 터를 로드할 때 다음 과 같은 오 류가 발 생합니 ERROR: operator class "gist_geometry_ops" does not exist for access method "gist". PostGIS 1.5에서 는 잘 됐는데 말이 죠.*

```

PostGIS 2 &#x00ac;&#xc6a9;&#xc790; &#xc9c0;&#xce68;&#xc11c;
&#xd074;&#xb798;&#xc2a4;&#xac00; gist_geometry_ops&#xc5d0;&#xc11c; gist_geometry_ops_2d&#xb85c;&#xbcc0;&#xa
gist_geometry_ops&#xb294;&#xc644;&#xc804;&#xd788; &#xc0ad;&#xc81c;&#xb418;&#xc5c8;&#xc2b5;&#xb2c8;&#xb2e4;
PostGIS 2 &#x00ac;&#xc804;&#xbd80;&#xd130; 3D&#xb97c; &#xc9c0;&#xc6d0;&#xd558;&#xae30; &#xc704;&#xd574;
N-D &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xb3c4;&#xc785;&#xd588;&#xb294;&#xb370; &#xad6c;
&#xba85;&#xce6d; gist_geometry_ops&#xac00; &#xbd80;&#xc815;&#xd655;&#xd558;&#xae0; &#xd63c;&#xb3d9;&#xb4
&#xc5ec;&#xaca8;&#xc84c;&#xae30; &#xb54c;&#xbbb38;&#xc785;&#xb2c8;&#xb2e4; &#xd14c;&#xc774;&#xbe14; &#xbco
&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xb294; &#xacfc;&#xc815;&#xc758; &#xc77c;&#xbd8
&#xba87;&#xba87; &#xad6c; &#xbc84;&#xc804; &#xc751;&#xc6a9; &#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc740;
&#xc5f0;&#xc0b0;&#xc790; &#xd074;&#xb798;&#xc2a4; &#xba85;&#xc744; &#xc815;&#xd655;&#xd788; &#xcc38;&#xc8
&#xae30;&#xbcf8; 2D &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc0ac;&#xc6a9;&#xd558;&#xae0;&#xc790; &#xd558;&#xc
&#xacbd;&#xc6b0; &#xc774;&#xb807;&#xac8c; &#xc815;&#xd655;&#xd788; &#xcc38;&#xc870;&#xd560; &#xd544;&#xc6
&#xc5c6;&#xc2b5;&#xb2c8;&#xb2e4; &#xb530;&#xb77c;&#xc11c; &#xc774;&#xb7f0; &#xacbd;&#xc6b0;&#xb77c;&#xba
&#xc778;&#xb371;&#xc2a4; &#xc0dd;&#xc131; &#xba85;&#xb839;&#xc5b4;&#xb97c; &#xb2e4;&#xc74c;&#xacfc;
&#xac19;&#xc774; &#xbcc0;&#xacbd;&#xd558;&#xc2ed;&#xc2dc;&#xc624; &#xb098;&#xc05c; &#xc608;&#xc5d0;&#xc11

```

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

```
&#xc88b;&#xc740; &#xc608;&#xb85c;
```

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

```

&#xc0ac;&#xc6a9;&#xc790;&#xac00; &#xc5f0;&#xc0b0;&#xc790; &#xd074;&#xb798;&#xc2a4;&#xb97c; &#xc9c0;&#xc81
&#xd558;&#xb294; &#xc720;&#xc77c;&#xd55c; &#xacbd;&#xc6b0;&#xb294; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc77
3D &#xacf5;&#xac04; &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xae0;&#xc790; &#xd560;
&#xb54c;&#xbfd0;&#xc785;&#xb2c8;&#xb2e4;

```

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

```

&#xc548;&#xd0c0;&#xae5d;&#xac8c;&#xb3c4; &#xc0ac;&#xc6a9;&#xc790;&#xac00; &#xbcc0;&#xacbd;&#xd560; &#xc21
&#xc5c6;&#xb294; &#xad6c; &#xbc84;&#xc804; gist_geometry_ops&#xac00; &#xd558;&#xb4dc; &#xcf54;&#xb529;&#xb4
&#xc788;&#xb294; &#xcef4;&#xd30c;&#xc77c;&#xb41c; &#xcf54;&#xb4dc;&#xb97c; &#xc368;&#xc57c;&#xb9cc;
&#xd560; &#xacbd;&#xc6b0; PostGIS 2.0.2 &#xc774;&#xc0c1; &#xb84;&#xc804;&#xc5d0; &#xd328;&#xd0a4;&#xc9d5;&
legacy_gist.sql &#xc744; &#xc774;&#xc6a9;&#xd574;&#xc11c; &#xad6c; &#xbc84;&#xc804; &#xd074;&#xb798;&#
&#xc0dd;&#xc131;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4; &#xd558;&#xc9c0;&#xb9cc; &#xc774;
&#xd574;&#xacb0; &#bc29;&#bc95;&#xc744; &#xc4f8; &#xacbd;&#xc6b0; &#xc774;&#xd6c4; &#xc5b4;&#xb5a4;
&#xc2dc;&#xc810;&#xc5d0;&#xc11c; &#xd574;&#xb2f9; &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc0ad;&#xc81c;&#xd55
&#xc5f0;&#xc0b0;&#xc790; &#xd074;&#xb798;&#xc2a4; &#xc5c6;&#xc774; &#xc7ac;&#xc0dd;&#xc131;&#xd558;&#xb29
&#xac83;&#xc774; &#xc88b;&#xc2b5;&#xb2c8;&#xb2e4; &#xc774;&#xb807;&#xac8c; &#xd558;&#xba74; &#xd5a5;&#xd
&#xb2e4;&#xc2dc; &#xc5c5;&#xadf8;&#xb808;&#xc774;&#xb4dc;&#xb97c; &#xd574;&#xc57c; &#xd560; &#xb54c;
&#xc218;&#xae0;&#xb97c; &#xb35c;&#xac8c; &#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;

```

4. *PostgreSQL 9.0* 을 운용 중 이 며 *OpenJump*, *SafeFME* 와 그 리 몇 몇 다 른 툴 들 에 서 지 오 메 트 &#xb9 더 는 읽 ౰ 나 본 수 없 습 니 다.

```

PostgreSQL 9.0+&#xc5d0;&#xc11c; bytea&#xb370;&#xc774;&#xd130;&#xb97c; &#xc704;&#xd55c; &#xb514;&#xd3f4;&#
&#xc778;&#xcf54;&#xb529;&#xc740; hex&#xb85c; &#xbc14;&#xb00c;&#xc5c8;&#xae0; &#xc608;&#xc804; JDBC
&#xb4dc;&#xb77c;&#xc774;&#xbe0c;&#xb294; &#xc5ec;&#xc804;&#xd788; escape format&#xc744; &#xcde8;&#xd569;&#
&#xc774;&#xac83;&#xc740; &#xc608;&#xc804; JDBC &#xb4dc;&#xb77c;&#xc774;&#xb84;&#xb97c; &#xc0ac;&#xc6a9;
Java &#xc5b4;&#xd50c;&#xb9ac;&#xcf00;&#xc774;&#xc158;&#xc774;&#xb098; &#xc624;&#xb798;&#xb41c; ST_AsBinary
&#xc791;&#xb3d9;&#xc744; &#xc694;&#xd558;&#xb294; &#xc608;&#xc804; npgsql&#xb4dc;&#xb77c;&#xc774;&#xb84;
&#xc0ac;&#xc6a9;&#xd558;&#xb294; .NET &#xc5b4;&#xd50c;&#xb9ac;&#xcf00;&#xc774;&#xc158; &#xac19;&#xc740;
&#xba87;&#xba87; &#xc5b4;&#xd50c;&#xb9ac;&#xcf00;&#xc774;&#xc158;&#xc5d0; &#xc601;&#xd5a5;&#xc744;
&#xb07c;&#xce69;&#xb2c8;&#xb2e4; &#xc774;&#xac83;&#xc744; &#xb2e4;&#xc2dc; &#xc791;&#xb3d9;&#xc2dc;&#xd0
&#xc704;&#xd55c; &#xb450; &#xac00;&#xc9c0; &#xc811;&#xadfc; &#bc29;&#bc95;&#xc774; &#xc788;&#xc2b5;&#xb2
driver&#xb97c; &#xcd5c;&#xc2e0; PostgreSQL 9.0&#xb84;&#xc804;&#xc73c;&#xb85c; &#xc5c5;&#xadf8;&#xb808;&#xc7
&#xc2dc;&#xd0ac; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4; &#xcd5c;&#xc2e0; PostgreSQL&#xb84;&#xc804;&#xc
http://jdbc.postgresql.org/download.html&#xc5d0;&#xc11c; &#xb2e4;&#xc6b4;&#xbcb1b;&#xc73c;&#xc2e4; &#xc218;

```

있습니다만약 .NET app을 실행중이락 Npgsql 2.0.11또는 그 이상의 벴전을 사용자 수 있습니다. 이는 [Francisco Figueiredo's Npgsql 2.0.11 released blog entry](#)에 설명되어 있는 것과 같이 http://pgfoundry.org/frs/?group_id=1000140에 서 다운򼬛으실 수 있습니다. 만약 PostgreSQL driver를 업그 레이 하는 것이 옵션사항이 아니라명 아래를 입력함으로써 이전 򼬩수 디폴트 설정을 할 수 있습니다:

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

5. 지오메트리 󌻬럼을 보기 위해 PgAdmin을 사용하려고 했으나 비어있 왜 그 런지요?

PgAdmin은 큰 지오메트리에 대해 아뮠 보여주지 않습니다. 지오메트리 󌻬럼에 있는 데이 터를 검증하는 가장 좋은 򼬩򼮕은?

```
-- &#xc774; &#xc2a4;&#xd06c;&#xb9bd;&#xd2b8;&#xb294; &#xc0ac;&#xc6a9;&#xc790;&#xc758; &#xbaa8;&#xb4e0; geom &#xd56d;&#xbaa9;&#xc774; &#xcc28; &#xc788;&#xb2e4;&#xba74; &#xc5b4;&#xb5a4; &#xb808;&#xcf54;&#xb4dc;&#xb3c4; &#xbcb18;&#xd658;&#xd558;&#xc9c0; &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;.
SELECT somefield FROM mytable WHERE geom IS NULL;
```

```
-- &#xc5bc;&#xb9c8;&#xb098; &#xd070; &#xc9c0;&#xc624;&#xba54;&#xd2b8;&#xb9ac;&#xac00; &#xcffc;&#xb9ac;&#xb418;&#xb294;&#xc9c0; &#xc54c;&#xace0; &#xc2f6;&#xb2e4;&#xba74;
-- &#xc9c0;&#xc624;&#xba54;&#xd2b8;&#xb9ac; &#xcceec;&#xb7fc;&#xc5d0; &#xb4e4;&#xc5b4;&#xc788;&#xb294; &#xc5b4;&#xb5a4; &#xb3c4;&#xd615;&#xc758; &#xc810; &#xc218;&#xc911; &#xac00;&#xc7a5; &#xd070; &#xac83;&#xc744; &#xb9d0;&#xd574; &#xc90c;
SELECT MAX(ST_NPoints(geom)) FROM sometable;
```

6. 어떠한 종류의 지오메트리 오뻀 저장할 수 있습니까?

포인트 라인스트링 폴리곤 멀트 ୀ티라인스트링 멀티폴리곤 그 지오메트리 󌻬렉션(GeometryCollection) 도형이 저장할 수 있습니다. PostGIS 2.0 이상 򼮄이 򼬏 다면체 표면(Polyhedral Surface)도 기본 도유형으로 저장할 수 있습니다. Z, M, ZM 확장자를 가지는 오픈GIS WKT 형수 GIS Well Known Text Format)이 이런 도형들을 지정하 있습니다. 현재 다음 세 가지 데이 형을 지원합니다. 그 세 가지 유형은 측정시 평면좌표계를 이용하는 표준 OGC 도형 데이터 유 구체 또는 편구체 상에서 계산하 측지좌표계를 이용하는 지리 데이터 유형 그리고 PostGIS 공간 유형 계보에 새롭게 추가된 래스터 데이터 분석 򼬏 저장을 위한 래수 래스터 전용 FAQ도 있습니다. 더 자 설명은 Chapter 13 과 Chapter 12 를 참조하십시이

7. 혼동되네요. 제가 지오메트리- geometry- 또는 지형-geography- 중 어떤 데이터사용하여 저장해야 합니까?

짧은 답변; 지리형(geography)은 장거리(long range distance) 맄위 측정을 지원하는 새로데이터 유형이지만; 이 유형을 대상으로 하는 계산 대부분은 도형의 경우보다 느립니다. 지리이용할 경우; 평면좌표계를 자󁕌 필요는 없습니다. 사용자가 전세계에 걸친 데이터를 가지고 있고 거리 򼰏 길이를 측정하는 데에만 관심이 있을 경우 일ଘ전지리형이 최선입니다. 도형 데이유형은 훨씬 많은 함수가 지원하제3자 도구의 광맄위한 지원을 ଛ으며. 도형을 대상으로 하는 연산이 더 빠릅니다 -- 대용량 도사우 때로는 10ର &#be60;르기도 합니사용자가 &#acf5;간 참조 시스템(Spatial Reference System)에 꽤 익숙하거나. 사용자 데전부가 단일 &#acf5;간 참조 시스템(SRID) 의 적용을 ଛ는 국지적인 데이터 처리하는 경우. 또는 상당한 양이 &#acf5;간 처리 작업을 해야할 경우 도형이 최선입니다. 주의; 각 유형 장점을 취하기 위해 1단계 작업(one-off)만으로 꽤 쉽게 두 유형을 변환수 있습니다. 현재 어떤 지원을 ଛ는지 그리고 ଛ지 못 하는지 알고 싶다면 Section 15.11 를 참조하십시긴 답변; 보다 더 긴 답변을 원하십시오. Section 4.3.3 and function type matrix를 참조하십시오.

8. *geography*에 관한 *geographic region*이 얼마나 큰지와 같이 더 복잡하고 심오한 질문이 있습니다. *geography* 󌻬럼을 이용하여 타당한 답들을 얻을 수 있나요? 예를 들어 극지닉 같은 제한사할 있나요? 필드안의 모든 것은 ଘ전 Server 2008가 가지고 있는 것처럼), 스피등에 맞아 떨어져야 하나요? 이 섹션에서 답변하기에는 질문 너무 깊고 복잡합니다. Section 4.3.4을 참오.
9. 어떻게 *GIS* 객체 를 데이 터베이 스삽입할 수 있나요?

First, you need to create a table with a column of type "geometry" or "geography" to hold your GIS data. Storing geography type data is a little different than storing geometry. Refer to Section 4.3 for details on storing geography. Geometry사용을 위해; `psql` 로 데이터베이스접속하시고 다음 `SQL`을 실행해 보오.

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

도형 열 정의가 실패하는 경우, 아마도 PostGIS 함수와 객체를 해당 데이터베이스에 로드하지 않압 PostGIS 2.0 미만 ஄전을 사용하고 있을 겁니다. Section 2.2 를 참조하십시오. 그리

SQL insert geometry into gtest (ID, NAME, GEOM) VALUES (1, 'First Geometry', ST_GeomFromText('LINESTRING(2 3,4 5,6 5,7 8)'));

GIS object reference

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

GIS object reference

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

GIS object reference

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

- SQL insert geometry into gtest (ID, NAME, GEOM) VALUES (1, 'First Geometry', ST_GeomFromText('LINESTRING(2 3,4 5,6 5,7 8)'));
- GIS object reference

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

11. `어떻게큰테이블에서공간쿼리속도를높일수있나요?`

```
&#xd070;&#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xc11c;&#xc758;&#xbe60;&#xb978;&#xcffc;&#xb9ac;&#xb294;
&#xacf5;&#xac04;&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xc758;&#xc874;&#xc7ac;&#xc774;&#xc720;
&#xc9c0;&#xc6d0;&#xc640;&#xd568;&#xae8;).&#xadf8;&#xb7ec;&#xbbc0;&#xb85c;&#xc88b;&#xc740;&#xc778;&#xb;
&#xac00;&#xc9c0;&#xb294;&#xac83;&#xc774;&#xc911;&#xc694;&#xd569;&#xb2c8;&#xb2e4;.geometry&#xccec;&#xb;
&#xd568;&#xae8;&#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xacf5;&#xac04;&#xc778;&#xb371;&#xc2a4;&#xb97c;
&#xad6c;&#xcd95;&#xd558;&#xae30;&#xc704;&#xd574;&#xb2e4;&#xc74c;&#xacfc;&#xac19;&#xc740;"CREATE
INDEX"&#xd568;&#xc218;&#xb97c;&#xc0ac;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;;
```

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

"USING GIST"옵션은 GiST(Generalized Search Tree)인덱스를사서벴를알려줍니다.

Note



GiST 인덱스들은손실이있다󊳠가정합니다. 손실인덱스들은인덱스구축을위해 proxy 오뻀젝트(공간의경우벴운딩벵스)를사용합니다.

사용자는 PostgreSQL 쿼리플래너가언È이것을사용할지에합당한ીಁ할수있도록충분한정보를갖¬보장해야합니다. 이렇게하기위해서는 geometry테이블에 "통계수É벸드시실행해야합니다 PostgreSQL 8.0.x 그리󊳠그이상벴전의경우 VACUUM ANALYZE명령어을실행하십시오 PostgreSQL 7.4.x 그이하의벴전의경우 SELECT UPDATE_GEOME명령어를실행하십시오.

12. `왜 PostgreSQL R-Tree indexes를지원하지않나요?`

PostGIS의초기벴전들은 PostgreSQL R-Tree인덱스사용했습니다. 그러나 PostgreSQL R-Tree는벴전0.6이후완전히폐기되었󊳠.공간인덱싱은 R-Tree-over-GiST scheme를이용해제공됩니다. 우리테스터들이 R-Tree와 GiST의검색속도가유사하다²것을보여주었습니다. 원래의 PostgreSQL R-Tree는 GIS피ಘ들에적합하지않&#두가지한계점이있었습니다(Ç한계점은일벸적인 R-Tree의쳬셉때아닌 PostgreSQL native R-Tree구현상의򻬸제임에주의하십시오);

- PostgreSQL의 R-Tree인덱스는 8K이상의크기&#가지는피ಘ를다루지못했습&#xb; GiST인덱스는피ಘ자신대신에벴운딩벵스를쓰는"손실"트릭&#이용해할수있습니다.
- PostgreSQL의 R-Tree인덱스는"null safe"하지않습&#x그래서널지오메트리가포함&#xb지오메트리쳬럼에대한인덱&#xc생성이실패할수있습니다.

13. `왜 AddGeometryColumn ( ) 함수를 사용해야 하난그리고 다른 모든 OpenGIS 들도 사용해하나요 ?`

```
&#xb9cc;&#xc57d; OpenGIS &#xc9c0;&#xc6d0; &#xae30;&#xb2a5;&#xb4e4;&#xc744; &#xc0ac;&#xc6a9;&#xd558;&#xae30;&#xc6d0;&#xd558;&#xc9c0; &#xc54a;&#xc73c;&#xc2e0;&#xb2e4;&#xba74; &#xc0ac;&#xc6a9;&#xd560; &#xd544;&#xc690;&#xc5c6;&#xc2b5;&#xb2c8;&#xb2e4;. &#xb2e8;&#xc21c;&#xd788; &#xad6c; &#xbc84;&#xc804;&#xc758; &#xd14c;&#xc744; &#xc0dd;&#xc131;&#xd558;&#xace0; CREATE &#xbb38;&#xc5d0;&#xc11c; geometry &#xd589;&#xb4e4;&#xc744; &#xc815;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;. &#xbaa8;&#xb4e0; &#xc9c0;&#xc624;&#xba54;&#xd2b8;&#xb900; &#xc758; SRID&#xb97c; &#xac00;&#xc9c8; &#xac83;&#xc774;&#xba70;, OpenGIS meta-data &#xd14c;&#xc774;&#xbe14;&#xc801;&#xc808;&#xd558;&#xac8c; &#xcc44;&#xc6cc;&#xc9c0;&#xc9c0; &#xc54a;&#xc744; &#xac83;&#xc785;&#xb2c8;&#xadf8;&#xb7ec;&#xb098;. &#xc774;&#xac83;&#xc740; PostGIS &#xae30;&#xbc18;&#xc758; &#xb300;&#xbd80;&#xbd80;&#xc560;&#xd50c;&#xb9ac;&#xcf00;&#xc774;&#xc158;&#xb4e4;&#xc774; &#xc2e4;&#xd328;&#xd558;&#xb3c4;&#xb85c;&#xd560; &#xac83; &#xc785;&#xb2c8;&#xb2e4;. &#xadf8;&#xb807;&#xae30;&#xc5d0; geometry &#xd14c;&#xc774;&#xbe14;&#xc0dd;&#xc131;&#xd558;&#xae30; &#xc704;&#xd574;&#xc11c; &#xc77c;&#xbc18;&#xc801;&#xc73c;&#xb85c; AddGeometryColumn ( ) &#xc774;&#xc0ac;&#xc6a9;&#xd1a0;&#xb85d; &#xad8c;&#xc7a5;&#xd569;&#xb2c8;&#xb2e4; geometry_columns meta-data&#xb97c; &#xd65c;&#xc6a9;&#xd558;&#xb294; &#xc560;&#xd50c;&#xb9ac;&#xcf00;&#xc774;&#xc911; &#xd558;&#xb098;&#xc785;&#xb2c8;&#xb2e4;. &#xad6c;&#xccb4;&#xc801;&#xc73c;&#xb85c; &#xb9d0;&#xd558; MapServer&#xb294; &#xd53c;&#xcc98;&#xc758; on-the-fly reprojection &#xc2dc; &#xd604;&#xc7ac; map projection&#xc5d0; Geometrv &#xd589;&#xc758; SRID&#xb97c; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;
```

14. 다른오브젝트의반경이내모&#xb4
오브젝트들을찾을수있는가&#xc7
좋은방법은무엇인가요?

```
&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xb97c;&#xac00;&#xc7a5;&#xd6a8;&#xc728;&#xc801;&#xc73a;&#xc0ac;&#xc6a9;&#xd558;&#xae30;&#xc704;&#xd574;&#xc11c;&#xb294;&#xbcb4;&#xb529;&#xbcb5;&#xc2a4;&#xd14c;&#xc2a4;&#xd2b8;&#xc640;&#xbcb8;&#xacbd;&#xd14c;&#xc2a4;&#xd2b8;&#xb97c;&#xacb0;&#xd569;&#xd55c;&#xbcb8;&#xacbd;&#xcffc;&#xb9ac;&#xb97c;&#xd558;&#xb294;&#xac83;&#xc774;&#xac00;&#xc7a5;&#xc88b;&#xc2b5;&#xbcb4;&#xc6b4;&#xb529;&#xbcb5;&#xc2a4;&#xd14c;&#xc2a4;&#xd2b8;&#xb294;&#xacf5;&#xac04;&#xc778;&#xb370;&#xc0ac;&#xc6a9;&#xd558;&#xba70;&#xd6c4;&#xc5d0;&#xbcb8;&#xacbd;&#xd14c;&#xc2a4;&#xd2b8;&#xb97c;&#xc801;&#xc6a9;&#xd560;&#xb370;&#xc774;&#xd130;&#xc758;&#xbd80;&#xbd84;&#xc9d1;&#xd569;&#xc5d0;&#xb300;&#xd55c;&#xbe60;&#xb978;&#xc5d1;&#xc138;&#xc2a4;&#xb97c;&#xc81c;&#xacf5;&#xd569;&#xb2c8;&#xb2e4;ST_DWithin(geometry, geometry, distance) &#xd568;&#xc218;&#xb294;&#xc778;&#xb371;&#xc2a4;&#xb97c;&#xc774;&#xc6a9;&#xd55c;&#xac70;&#xb9ac;&#xac80;&#xc0c9;&#xc744;&#xc218;&#xd589;&#xd558;&#xb294;&#xc720;&#xc6a9;&#xd55c;&#xbcb29;&#xbcb95;&#xc785;&#xb2c8;&#xb2e4;.&#xc774;&#xd568;&#xc218;&#xb294;&#xbcb8;&#xacbd;&#xc744;&#xb458;&#xb7ec;&#xc2f8;&#xae30;&#xc5d0;&#xcda9;&#xbd84;&#xd55c;&#xd06c;&#xae30;&#xac80;&#xc0c9;&#xc9c1;&#xc0ac;&#xac01;&#xd615;&#xc744;&#xc0dd;&#xc131;&#xd558;&#xba70;&#xc791;&#xb3d0;&#xadf8;&#xb7f0;&#xb4a4;&#xc778;&#xb371;&#xc2a4;&#xb85c;&#xac80;&#xc0c9;&#xb41c;&#xbd80;&#xbd84;&#xc9c1;&#xc815;&#xd655;&#xd55c;&#xac70;&#xb9ac;&#xac80;&#xc0c9;&#xc744;&#xc218;&#xd589;&#xd569;&#xb2c8;&#xb2e4;&#xc608;&#xb97c;&#xb4e4;&#xc5b4;POINT(1000 1000)&#xc758; 100 &#bbf8;&#xd130;&#xc774;&#xb0b4;&#xc758;&#xbaa8;&#xb4e0;&#xc624;&#xbe0c;&#xc81d;&#xd2b8;&#xb4e4;&#xc744;&#xcc3e;&#xae30;&#xc704;&#xd574;&#xc11c;&#xb2e4;&#xc74c;&#xc758;&#xcffc;&#xb9ac;&#xb97c;&#xd1b5;&#xd574;&#xc798;&#xc218;&#xd589;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;:
```

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

15. 어떻게 쿼리의 일부로 좌표계
변환을 수행할 수 있습 니까?

To perform a reprojection, both the source and destination coordinate systems must be defined in the SPATIAL_REF_SYS table, and the geometries being reprojected must already have an SRID set on them. Once that is done, a reprojection is as simple as referring to the desired destination SRID. The below projects a geometry to NAD 83 long lat. The below will only work if the srid of geom is not -1 (not undefined spatial ref)

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

16. 제법 큰 지오메트리에 ST_AsEWKT와 ST_AsText&#
하였습니다. 그랬더니 이것이

```
&#xe48; &#xd544; &#xb4dc; &#xb97c; &#xbc18; &#xd658; &#xd588; &#xc2b5; &#xb2c8; &#xb2e4;. &#xc65c; &#xc774; &#xb
&#xac83; &#xc778; &#xac00; &#xc694; ?
```

```
&#xc544; &#xb9c8; &#xb3c4; &#xd070; &#xd14d; &#xc2a4; &#xd2b8; &#xb97c; &#xbcf4; &#xc5ec; &#xc8fc; &#xc9c0;
&#xbabb; &#xd558; &#xb294; PgAdmin &#xc774; &#xb098; &#xb2e4; &#xb978; &#xd234; &#xc744; &#xc4f0; &#xace0;
&#xc788; &#xae30; &#xb54c; &#xbb38; &#xc77c; &#xac83; &#xc785; &#xb2c8; &#xb2e4;. &#xb9cc; &#xc57d; &#xc0ac; &#xc6
&#xc9c0; &#xc624; &#xba54; &#xd2b8; &#xb9ac; &#xac00; &#xcda9; &#bd84; &#xd788; &#xd06c; &#xb2e4; &#xba74;.
&#xc774; &#xb7f0; &#xd234; &#xc5d0; &#xc11c; &#xb294; &#xacf5; &#bc31; &#xc73c; &#xb85c; &#xb098; &#xd0c0; &#xb0a
&#xac83; &#xc785; &#xb2c8; &#xb2e4;. PSQL &#xc744; &#xc0ac; &#xc6a9; &#xd558; &#xc2dc; &#ba74; WKT &#xb85c;
&#xbcf4; &#xac70; &#xb098; &#xcd9c; &#xb825; &#xd558; &#xc2e4; &#xc218; &#xc788; &#xc2b5; &#xb2c8; &#xb2e4;.
```

```
--To check number of geometries are really blank
SELECT count(gid) FROM geotable WHERE geom IS NULL;
```

17. *ST_Intersects* 를 했 는 데 두 지 오 메 트 리 가 교 차 하 지 않 는 다 는 메 시 지 를 받 았 습 니 다. 교 차 하 고 있 음 을 내 가 아 는 데 말 이 죠 !!! 왜 이 럴 까 일 반 적 으 로 두 가 지 경 우 에 발 사 용 자 의 지 오 메 트 리 가 유 효 하 않 거 나 - **ST_IsValid** 확 인. 사 용 자 가 ST_AsText 가 잘 라 낸 유 효 숫 자 를 가 지 고 교 차 판 단 하 는 경 우 입 니 다. 소 숫 점 이 하 의 많 은 숫 자 들 이 사 용 자 에 보 이 지 않 습 니 다.
18. 저 는 PostGIS 를 사 용 하 여 소 프 트 웨 어 개 발 해 출 시 하 였 습 니 다. 제 소 프 PostGIS 처 럼 GPL 을 라 이 센 스 로 가 져 야 하 나 요 ? 만 약 PostGIS 를 사 용 할 경 우 제 코 드 를 공 개 해 야 만 하 나 요 ?

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.

19. *Why are the results of overlay operations and spatial predicates sometimes inconsistent?*

This is usually presented as a specific case, such as

- Why is `ST_Contains( A, ST_Intersection( A, B ) )` false ?
- Why is `ST_Contains( ST_Union( A, B ), A )` false ?
- Why is `ST_Union( A, ST_Difference( A, B ) )` not equal to A ?
- Why does `ST_Difference( A, B )` intersect the interior of B ?

The reason is that PostGIS represents geometry and performs operations using finite-precision floating-point numbers. This provides the illusion of computing using real numbers - but it's only an illusion. Inevitably, small inaccuracies occur, which cause results of different operations to be slightly inconsistent. Furthermore, PostGIS operations contain error-prevention code which may perturb input geometries by tiny amounts in order to prevent robustness errors from occurring. These minor alterations also may produce computed results which are not fully consistent. The discrepancy between results should always be very small. But queries should not rely on exact consistency when comparing overlay results. Instead, consider using an area or distance-based tolerance in geometric comparisons.

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](#)

10.1 10.1 10.1

10.1.1 getfaceedges_returntype

`getfaceedges_returntype` — A composite type that consists of a sequence number and an edge number.

설명

A composite type that consists of a sequence number and an edge number. This is the return type for `ST_GetFaceEdges` and `GetNodeEdges` functions.

1. sequence 는 정 수 형 입 니 다.: 지 형 스 키 &#x
및 SRID 를 정 의 하 는 topology.topology 테 이 블 에
정 의 돼 있 는 지 형 을 참 조 하 십 &#xc2d
2. edge 는 정 수 형 입 니 다.: 경 계 선 의 식

10.1.2 TopoGeometry

TopoGeometry — A composite type representing a topologically defined geometry.

설명

ʹ정지형레이어에있는,특정유ଏ특정ID를가진지형도형을참조합성유형입니다. TopoGeometry를구성하다요소는 topology_id, layer_id, id, type 정수형속성입니다

1. topology_id 는 정 수 형 입 니 다: 지 형 스 키 및 SRID 를 정 의 하 는 topology.topology 테 이 블 에 정 의 돼 있 는 지 형 을 참 조 하 십 싐
2. layer_id 는 정 수 형 입 니 다: TopoGeometry 가 속 해 있 는 레 이 어 테 이 블 의 layer_id 입 니 다 topology_id 와 layer_id 를 조 합 하 면 topology.layers 테 이 블 유 일 한 참 조 (unique reference) 를 제 공 합 니 다.
3. id 는 정 수 형 입 니 다: 해 당 지 형 레 지 형 도 형 을 유 일 하 게 정 의 하 는 자 동 생 성 된 순 차 번 호 입 니 다.
4. 1 에 서 4 까 지 의 type 정 수 가 도 형 유 형 정 의 합 니 다. 1: [멀 티] 포 인 트. 2: [멀 티] 포 인 트. 3: [멀 티] 폴 리 까. 4: 도 형 집 합.

형변환 습성

이 단 원 에 서 이 런 데 이 터 유 형 에 사 용 할 수 있 는 자 동 형 변 환 은 물 롕 시 적 인 형 변 환 도 소 개 합 니 다.

형변환결과	습성
도형	자동적

ValidateTopology

CreateTopoGeom

10.1.3 validate_topology_return_type

`validate_topology_return_type` — A composite type that consists of an error message and `id1` and `id2` to denote location of error. This is the return type for `ValidateTopology`.

validate_topology_return_type

`error`: (varchar) Message describing the topology error. `id1`: (integer) ID of the geometry or face that is the source of the error. `id2`: (integer) ID of the geometry or face that is the target of the error. `id1` and `id2` are both 0 if the error is not location-specific.

- `error`: Message describing the topology error. `id1`: ID of the geometry or face that is the source of the error. `id2`: ID of the geometry or face that is the target of the error. `id1` and `id2` are both 0 if the error is not location-specific.
- `id1`: ID of the geometry or face that is the source of the error. `id2`: ID of the geometry or face that is the target of the error. `id1` and `id2` are both 0 if the error is not location-specific.
- `id2`: ID of the geometry or face that is the target of the error. `id1`: ID of the geometry or face that is the source of the error. `id1` and `id2` are both 0 if the error is not location-specific.

ValidateTopology

ValidateTopology

10.2 TopoElement

10.2.1 TopoElement

`TopoElement` — A composite type that consists of a `geom_type` (geometry type), a `geom_id` (integer), a `face_id` (integer), a `parent_id` (integer), a `child_id` (integer), a `geom_data` (bytea), and a `face_data` (bytea).

설명

단순 또는 계층 **TopoGeometry** 의 구성 요소
1개를 표현하는 데 쓰이는 정수형
2개의 배열입니다.

˨순 TopoGeometry의 경우, 배열의 첫 번째
요소가 지형적 원시형(topological primitive)의 식&
나타내고 두 번째 요소가 그 유형(1:
node, 2: edge, 3: face)을 나타냅니다, 계층 TopoGeometry의
경우 배열의 첫 번째 요소가 자식
TopoGeometry의 식별자를 나타내고 두 번째
요소가 해당 레이어 식별자를 나&#

Note



```
&#xc5b4;&#xb5a4;&#xacca4;&#xce35; TopoGeometry&#xac00; &#xc8fc;&#xc5b4;&#xc9c0;&#xb354;&#xb77c;&#xb3c4;,&#xbaa8;&#xb4e0; &#xc790;&#xc2dd; TopoGeometry &#xc694;&#xc18c;&#xb4e4;&#xc740;,&#xd574;&#xb2f9; TopoGeometry&#xc758; &#xb808;&#xc774;&#xc5b4;&#xc5d0; &#xb300;&#xd574; topology.layer &#xb808;&#xcf54;&#xb4dc;&#xc5d0; &#xc9c0;&#xc815;&#xb41c; &#xb3d9;&#xc77c;&#xd55c; &#xc790;&#xc2dd; &#xb808;&#xc774;&#xc5b4;&#xc5d0;&#xc11c; &#xb098;&#xc635;&#xb2c8;&#xb2e4;.
```

예시

```
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}
```

```
-- topoelement&#xc5d0; &#xc694;&#xc18c; 3&#xac1c;&#xb97c; &#xac00;&#xc9c4; <=
    &#xbc30;&#xc5f4;&#xc744; &#xb123;&#xc73c;&#xba74; &#xc5b4;&#xb5bb;&#xac8c; <=
    &#xb418;&#xb294;&#xc9c0; &#xbcf4;&#xc5ec;&#xc8fc;&#xb294; &#xc608;&#xc2dc;
-- &#xc8fc;&#xc758;: topoelement&#xb294; &#xc694;&#xc18c; 2&#xac1c;&#xb97c; <=
    &#xac00;&#xc9c4; &#xbc30;&#xc5f4;&#xc774;&#xc5b4;&#xc57c; &#xd558;&#xbbc0;&#xb85c; <=
    &#xcc28;&#xc6d0; &#xd655;&#xc778;&#xc5d0; &#xc2e4;&#xd328;&#xd569;&#xb2c8;&#xb2e4;.
SELECT ARRAY[1,2,3]::topology.topoelement;
ERROR:  value for domain topology.topoelement violates check constraint "dimensions"
```

స고

GetTopoGeomElements, TopoElementArray, TopoGeometry, TopoGeom_addElement, TopoGeom_remElement

10.2.2 TopoElementArray

TopoElementArray — An array of TopoElement objects.

설명

1개 이상의 TopoElement 객체들의 배열로, 일반적으로 TopoElement 객체의 구성 요소 넘󊲨주는 데 쓰입니다.

예시

```
SELECT '{{1,2},{4,3}}'::topology.topoelementarray As tea;
      tea
-----
{{1,2},{4,3}}

-- &#xb354; &#xc124;&#xba85;&#xc801;&#xc778; &#xb3d9;&#xc77c;&#xd55c; &#xcffc;&#xb9ac; --
SELECT ARRAY[ARRAY[1,2], ARRAY[4,3]]::topology.topoelementarray As tea;
      tea
-----
{{1,2},{4,3}}

-- &#xc9c0;&#xd615;&#xacfc; &#xd568;&#xaed8; &#xd328;&#xd0a4;&#xc9d5;&#xb41c; &#xb30; &#xc5f4; &#xc885;&#xd569; &#xd568;&#xc218;&#xb97c; &#xc774;&#xc6a9; --
SELECT topology.TopoElementArray_Agg(ARRAY[e,t]) As tea
      FROM generate_series(1,4) As e CROSS JOIN generate_series(1,3) As t;
      tea
-----
{{1,1},{1,2},{1,3},{2,1},{2,2},{2,3},{3,1},{3,2},{3,3},{4,1},{4,2},{4,3}}
```

```
SELECT '{{1,2,4},{3,4,5}}'::topology.topoelementarray As tea;
ERROR:  value for domain topology.topoelementarray violates check constraint "dimensions"
```

స&#xace0:

TopoElement, GetTopoGeomElementArray, TopoElementArray Agg

10.3 TopoGeometry

10.3.1 AddTopoGeometryColumn

AddTopoGeometryColumn — 기존테이블에 TopoGeometry 열을추가하고, topology.layer에 이새열을레이얶추가하며. 새 layer id를 반환합니다.

Synopsis

```
integer AddTopoGeometryColumn(varchar topology_name, varchar schema_name, varchar table_name, varchar column_name,
varchar feature_type);
```

```
integer AddTopologyGeometryColumn(varchar topology_name, varchar schema_name, varchar table_name, varchar column_name,
varchar feature_type, integer child_layer);
```

설명

```
&#xac01; TopoGeometry &#xac1d; &#xccb4; &#xb294; &#xd2b9; &#xc815; &#xc9c0; &#xd615; &#xc758; &#xd2b9; &#xc815;
&#xb808; &#xc774; &#xc5b4; &#xc5d0; &#xc18d; &#xd569; &#xb2c8; &#xb2e4;. TopoGeometry &#xac1d; &#xccb4; &#xb97c;
&#xc0dd; &#xc131; &#xd558; &#xae30; &#xc804; &#xc5d0; &#xd574; &#xb2f9; &#xc9c0; &#xd615; &#xb808; &#xc774; &#xc5b4; &#
&#xc0dd; &#xc131; &#xd574; &#xc57c; &#xd569; &#xb2c8; &#xb2e4;. &#xc9c0; &#xd615; &#xb808; &#xc774; &#xc5b4; &#xb780;
&#xc9c0; &#xd615; &#xacfc; &#xd53c; &#xcc98; &#xd14c; &#xc774; &#xbe14; &#xc758; &#xc870; &#xd569; &#xc785; &#xb2c8; &#x
&#xc720; &#xd615; &#xbc0f; &#xacc4; &#xce35; &#xc815; &#xbcf4; &#xb3c4; &#xb2f4; &#xace0; &#xc788; &#xc2b5; &#xb2c8; &#x
AddTopoGeometryColumn() &#xd568; &#xc218; &#xb97c; &#xc774; &#xc6a9; &#xd574; &#xc11c; &#xb808; &#xc774; &#xc5b4; &#x
&#xc0dd; &#xc131; &#xd574; &#xbcf4; &#xaca0; &#xc2b5; &#xb2c8; &#xb2e4.:
```



```
&#xc774; &#xd568;&#xc218;&#xb294; &#xc694;&#xccad;&#xd55c; &#xc5f4;&#xc744; &#xd14c;&#xc774;&#xbe14;&#xc5d0;
&#xcd94;&#xac00;&#xd558;&#xace0; topology.layer &#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xb808;&#xcf54;&#xb4dc;&#xb97c;
&#xc8fc;&#xc5b4;&#xc9c4; &#xbaa8;&#xb4e0; &#xc815;&#xbcf4;&#xc640; &#xd568;&#xae8; &#xcd94;&#xac00;&#xd560;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
[child_layer]&#xb97c; &#xb530;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; (&#xb610;&#xb294;
NULL&#xb85c; &#xc124;&#xc815;&#xd560;) &#xacbd;&#xc6b0;, &#xd574;&#xb2f9; &#xb808;&#xc774;&#xc5b4;&#xac00;
(&#xc6d0;&#xc2dc;&#xd615; &#xc9c0;&#xd615; &#xc694;&#xc18c;&#xb4e4;&#xb85c; &#xc774;&#xb8e8;&#xc5b4;&#xc9c4;)
&#xae30;&#xbcf8; TopoGeometry&#xb97c; &#xb2f4;&#xc744; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xadf8;&#xb807;&#xc9c0;
&#xc54a;&#xc744; &#xacbd;&#xc6b0;, &#xc774; &#xb808;&#xc774;&#xc5b4;&#xb294; (child_layer&#xc5d0;&#xc11c; &#xb098;
TopoGeometry&#xb4e4;&#xb85c; &#xc774;&#xb8e8;&#xc5b4;&#xc9c4;) &#xacc4;&#xce35; TopoGeometry&#xb97c; &#xb2f4;&#xb808;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
&#xb808;&#xc774;&#xc5b4;&#xb97c; &#xc0dd;&#xc131;&#xd588;&#xb2e4;&#xba74; (AddTopoGeometryColumn &#xd568;&#xc774;
&#xb808;&#xc774;&#xc5b4;&#xc758; ID&#xb97c; &#xb818;&#xd658;&#xd569;&#xb2c8;&#xb2e4;) &#xd574;&#xb2f9; &#xb808;
TopoGeometry&#xac1d;&#xccb4;&#xb97c; &#xc791;&#xc131;&#xd560; &#xc900;&#xbe44;&#xac00; &#xb41c; &#xac83;&#xc785;
```

```
&#xc720;&#xd6a8;&#xd55c; feature_type &#xc740; POINT, LINE, POLYGON, COLLECTION&#xc785;&#xb2c8;&#xb2e4;.
```

Availability: 1.1

예시

```
-- &#xc774; &#xc608;&#xc2dc;&#xb97c; &#xc704;&#xd574; ma_topo ↔
&#xc2a4;&#xd0a4;&#xb9c8;&#xc5d0; &#xc0c8; &#xd14c;&#xc774;&#xbe14;&#xc744; ↔
&#xc0dd;&#xc131;&#xd588;&#xb2e4;&#xb294; &#xc810;&#xc5d0; ↔
&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
-- &#xb2e4;&#xb978; &#xc2a4;&#xd0a4;&#xb9c8;&#xc5d0; &#xc0dd;&#xc131;&#xd560; &#xc218; ↔
&#xc788;&#xc5c8;&#xb294;&#xb370;&#xb3c4; &#xb9d0;&#xc785;&#xb2c8;&#xb2e4;. ↔
&#xadf8;&#xb7ac;&#xb354;&#xb77c;&#xba74; topology_name&#xacfc; schema_name&#xc774; ↔
&#xb2ec;&#xb77c;&#xc84c;&#xc744; &#xac81;&#xb2c8;&#xb2e4;.
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');
```

참고

[DropTopoGeometryColumn](#), [toTopoGeom](#), [CreateTopology](#), [CreateTopoGeom](#)

10.3.2 DropTopology

DropTopology — 이용에 주의하십시오: 지형
스키마를 삭제하고 topology.topology 테이블에
해당 참조를, 그리고 geometry_columns 테이블에
해당 스키마에 있는 테이블의 참삭
제합니다.

Synopsis

integer **DropTopology**(varchar topology_schema_name);

DropTopoGeometryColumn

`DropTopoGeometryColumn(schema_name text, table_name text, column_name text)`

Removes the topology geometry column from the specified table. The column must be of type `geometry` or `geometry_z`. The topology must be of type `topology`. The column must be the primary key of the table.

Availability: 1.1

DropTopoGeometryColumn

`DropTopoGeometryColumn(schema_name text, table_name text, column_name text)`

Removes the topology geometry column from the specified table. The column must be of type `geometry` or `geometry_z`. The topology must be of type `topology`. The column must be the primary key of the table.

```
SELECT topology.DropTopoGeometryColumn('ma_topo');
```

DropTopoGeometryColumn**DropTopoGeometryColumn****DropTopoGeometryColumn**

`DropTopoGeometryColumn(schema_name text, table_name text, column_name text)`

Removes the topology geometry column from the specified table. The column must be of type `geometry` or `geometry_z`. The topology must be of type `topology`. The column must be the primary key of the table.

Synopsis

`DropTopoGeometryColumn(schema_name text, table_name text, column_name text)`

DropTopoGeometryColumn

`DropTopoGeometryColumn(schema_name text, table_name text, column_name text)`

Removes the topology geometry column from the specified table. The column must be of type `geometry` or `geometry_z`. The topology must be of type `topology`. The column must be the primary key of the table.

Availability: 1.1

DropTopoGeometryColumn

```
SELECT topology.DropTopoGeometryColumn('ma_topo', 'parcel_topo', 'topo');
```

;

AddTopoGeometryColumn

10.3.4 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**();

;

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`.

2.3.0 „€ௐ터 ¬ک할 수 있습니다.

;

```
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)
```

;

AddTopoGeometryColumn

10.3.5 TopologySummary

TopologySummary — Takes a topology name and provides summary totals of types of objects in topology.

Synopsis

text **TopologySummary**(varchar topology_schema_name);

<div>

Takes a topology name and provides summary totals of types of objects in topology.

2.0.0

<div>

```
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
```

<div>

[Topology_Load_Tiger](#)

10.3.6 ValidateTopology

ValidateTopology — Returns a set of validate_topology_returntype objects detailing issues with topology.

Synopsis

setof validate_topology_returntype **ValidateTopology**(varchar toponame, geometry bbox);

<div>

Returns a set of [validate_topology_returntype](#) objects detailing issues with topology, optionally limiting the check to the area specified by the `bbox` parameter.

List of possible errors, what they mean and what the returned ids represent are displayed below:

Message	id1	id2	Meaning
coincident nodes	Identifier of first node.	Identifier of second node.	Two nodes have the same geometry.
edge crosses node(Ÿ드를 가로지르는 경계선)	Identifier of the edge.	Identifier of the node.	An edge has a node in its interior. See ST_Relate .
invalid edge(유효하지 않은 경계선)	Identifier of the edge.		An edge geometry is invalid. See ST_IsValid .
edge not simple(단순하지 않은 경계선)	Identifier of the edge.		An edge geometry has self-intersections. See ST_IsSimple .
edge crosses edge(경계선을 가로지르는 경계선)	Identifier of first edge.	Identifier of second edge.	Two edges have an interior intersection. See ST_Relate .
edge start node geometry mismatch(경계선의 시작 노드 도형 불일치)	Identifier of the edge.	Identifier of the indicated start node.	The geometry of the node indicated as the starting node for an edge does not match the first point of the edge geometry. See ST_StartPoint .
edge end node geometry mismatch(경계선의 종단 노드 도형 불일치)	Identifier of the edge.	Identifier of the indicated end node.	The geometry of the node indicated as the ending node for an edge does not match the last point of the edge geometry. See ST_EndPoint .
face without edges(경계선이 없는 표면)	Identifier of the orphaned face.		No edge reports an existing face on either of its sides (left_face, right_face).
face has no rings(À리표는 표면)	Identifier of the partially-defined face.		Edges reporting a face on their sides do not form a ring.
face has wrong mbr	Identifier of the face with wrong mbr cache.		Minimum bounding rectangle of a face does not match minimum bounding box of the collection of edges reporting the face on their sides.
hole not in advertised face	Signed identifier of an edge, identifying the ring. See GetRingEdges .		A ring of edges reporting a face on its exterior is contained in different face.
not-isolated node has not-containing_face	Identifier of the ill-defined node.		A node which is reported as being on the boundary of one or more edges is indicating a containing face.

&#xc624;&#xb958;	id1	id2	Meaning
isolated node has containing_face	Identifier of the ill-defined node.		A node which is not reported as being on the boundary of any edges is lacking the indication of a containing face.
isolated node has wrong containing_face	Identifier of the misrepresented node.		A node which is not reported as being on the boundary of any edges indicates a containing face which is not the actual face containing it. See GetFaceContainingPoint .
invalid next_right_edge	Identifier of the misrepresented edge.	Signed id of the edge which should be indicated as the next right edge.	The edge indicated as the next edge encountered walking on the right side of an edge is wrong.
invalid next_left_edge	Identifier of the misrepresented edge.	Signed id of the edge which should be indicated as the next left edge.	The edge indicated as the next edge encountered walking on the left side of an edge is wrong.
mixed face labeling in ring	Signed identifier of an edge, identifying the ring. See GetRingEdges .		Edges in a ring indicate conflicting faces on the walking side. This is also known as a "Side Location Conflict".
non-closed ring	Signed identifier of an edge, identifying the ring. See GetRingEdges .		A ring of edges formed by following next_left_edge/next_right_edge attributes starts and ends on different nodes.
face has multiple shells	Identifier of the contended face.	Signed identifier of an edge, identifying the ring. See GetRingEdges .	More than a one ring of edges indicate the same face on its interior.

1.0.0 ಄전부터 사용할 수 있습니다.
개선 사항: 2.0.0 ಄전에서 경계선 가&#xb85
탐지를 더 효율적으로 개선하ૠ,
이전 ಄전에 존재했던 긍정 오류(fa
positive)를 해결했습니다.
변경 사항: 2.2.0 ಄전에서 'edge crosses node'를 오&#xb95
설명문과 일치시키기 위해 id1과 id2&#xc75
값을 서 로 ఔ꿈습니다.

Changed: 3.2.0 added optional bbox parameter, perform face labeling and edge linking checks.

예&#xc2dc:

```
SELECT * FROM topology.ValidateTopology('ma_topo');
      error      | id1 | id2
-----+-----+-----
face without edges |    1 |
```

`validateTopology_returntype, Topology_Load_Tiger`

10.3.7 ValidateTopologyRelation

ValidateTopologyRelation — Returns info about invalid topology relation records

Synopsis

setof record **ValidateTopologyRelation**(varchar toponame);

Returns a set records giving information about invalidities in the relation table of the topology.

Availability: 3.2.0

ValidateTopology

10.3.8 FindTopology

FindTopology — Returns a topology record by different means.

Synopsis

topology **FindTopology**(TopoGeometry topogeom);
 topology **FindTopology**(regclass layerTable, name layerColumn);
 topology **FindTopology**(name layerSchema, name layerTable, name layerColumn);
 topology **FindTopology**(text topoName);
 topology **FindTopology**(int id);

Takes a topology identifier or the identifier of a topology-related object and returns a topology.topology record.

Availability: 3.2.0

```
SELECT name(findTopology('features.land_parcel', 'feature'));
       name
-----
city_data
(1 row)
```

FindLayer

10.3.9 FindLayer

FindLayer — Returns a topology.layer record by different means.

Synopsis

```
topology.layer FindLayer(TopoGeometry tg);
topology.layer FindLayer(regclass layer_table, name feature_column);
topology.layer FindLayer(name schema_name, name table_name, name feature_column);
topology.layer FindLayer(integer topology_id, integer layer_id);
```

설명

Takes a layer identifier or the identifier of a topology-related object and returns a topology.layer record.

Availability: 3.2.0

예시

```
SELECT layer_id(findLayer('features.land_parcels', 'feature'));
 layer_id
-----
        1
(1 row)
```

참고

FindTopology

10.4 Topology Statistics Management

Adding elements to a topology triggers many database queries for finding existing edges that will be split, adding nodes and updating edges that will node with the new linework. For this reason it is useful that statistics about the data in the topology tables are up-to-date.

PostGIS Topology population and editing functions do not automatically update the statistics because a updating stats after each and every change in a topology would be overkill, so it is the caller's duty to take care of that.



Note

That the statistics updated by autovacuum will NOT be visible to transactions which started before autovacuum process completed, so long-running transactions will need to run ANALYZE themselves, to use updated statistics.

10.5 지형 작성자

10.5.1 CreateTopology

CreateTopology — 새 지형 스키마를 생성하고 topology.topology 테이블에 이 새 스키마를 등자

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);
```

Creates a new topology

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.

SQL/MM **ST_InitTopoGeo** 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.

Availability: 1.1

Enhanced: 2.0 added the signature accepting `hasZ`

Creates a new topology

Creates a new topology 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.

```
SELECT topology.CreateTopology('ma_topo', 26986, 0.5);
```

Creates a new topology 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.

```
SELECT topology.CreateTopology('ri_topo', 3438) As topoid;
topoid
-----
2
```

Creates a new topology

Section [4.5, ST_InitTopoGeo, Topology_Load_Tiger](#)

10.5.2 CopyTopology

CopyTopology — Copies a topology from one schema to another. The `existing_topology_name` is the name of the topology to be copied. The `new_name` is the name of the new topology. 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.

Synopsis

```
integer CopyTopology(varchar existing_topology_name, varchar new_name);
```


Topology

existing_topology_name, SRID, new_topology_name, existing_topology_name, Topo-Geometry

Note



topology.layer, schema_name, table_name, feature_column, TopoGeometry, SRID, ID, existing_topology_name, Topo-Geometry

2.0.0, ma_topo, ma_topo_bakup

SQL

```
SELECT topology.CopyTopology('ma_topo', 'ma_topo_bakup');
```

ST_InitTopoGeo

Section 4.5, [CreateTopology](#)

10.5.3 ST_InitTopoGeo

ST_InitTopoGeo — topology.topology, topology.topology, topology.topology, topology.topology

Synopsis

text **ST_InitTopoGeo**(varchar topology_schema_name);

Topology

CreateTopology, CreateTopology, CreateTopology, CreateTopology, CreateTopology, CreateTopology, CreateTopology, CreateTopology

Availability: 1.1



This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.17


```
-- <#xd14c;#xc774;#xbe14; #<#xbc0f; #<#xc9c0;#<#xd615; #<#xb3c4;#<#xd615;#<#xc744;  <#xc0dd;#<#xc131;#<#xd569;#<#xb2c8;#<#xb2e4;. --
CREATE TABLE ri.roads(gid serial PRIMARY KEY, road_name text);

SELECT topology.AddTopoGeometryColumn('ri_topo', 'ri', 'roads', 'topo', 'LINE');
```

<#<#xc38;#<#xace0;

[AddTopoGeometryColumn](#), [CreateTopology](#), [DropTopology](#)

10.5.5 TopoGeo_AddPoint

TopoGeo_AddPoint — Adds a point to an existing topology and returns its identifier. The given point will snap to existing nodes or edges within given tolerance. An existing edge may be split by the snapped point.

Synopsis

integer **TopoGeo_AddPoint**(varchar atopology, geometry apoint, float8 tolerance);

<#<#xc124;#<#xba85;

Adds a point to an existing topology and returns its identifier. The given point will snap to existing nodes or edges within given tolerance. An existing edge may be split by the snapped point.

2.0.0 <#<#xbc84;#<#xc804;#<#xbd80;#<#xd130; #<#xc0ac;#<#xc6a9;#<#xd560; #<#xc218; #<#xc788;#<#xc2b5;#<#xb2c8;#<#xb2e4;.

<#<#xc38;#<#xace0;

[TopoGeo_AddLineString](#), [TopoGeo_AddPolygon](#), [AddNode](#), [CreateTopology](#)

10.5.6 TopoGeo_AddLineString

TopoGeo_AddLineString — Adds a linestring to an existing topology using a tolerance and possibly splitting existing edges/faces. Returns edge identifiers.

Synopsis

SETOF integer **TopoGeo_AddLineString**(varchar atopology, geometry aline, float8 tolerance);

<#<#xc124;#<#xba85;

Adds a linestring to an existing topology and returns a set of edge identifiers forming it up. The given line will snap to existing nodes or edges within given tolerance. Existing edges and faces may be split by the line.



Note

Updating statistics about topologies being loaded via this function is up to caller, see [maintaining statistics during topology editing and population](#).

2.0.0 <#<#xbc84;#<#xc804;#<#xbd80;#<#xd130; #<#xc0ac;#<#xc6a9;#<#xd560; #<#xc218; #<#xc788;#<#xc2b5;#<#xb2c8;#<#xb2e4;.

TopoGeo_AddPolygon

[TopoGeo_AddPoint](#), [TopoGeo_AddPolygon](#), [AddEdge](#), [CreateTopology](#)

10.5.7 TopoGeo_AddPolygon

TopoGeo_AddPolygon — Adds a polygon to an existing topology using a tolerance and possibly splitting existing edges/faces. Returns face identifiers.

Synopsis

SETOF integer **TopoGeo_AddPolygon**(varchar atopology, geometry apoly, float8 tolerance);

TopoGeo_AddPolygon

Adds a polygon to an existing topology and returns a set of face identifiers forming it up. The boundary of the given polygon will snap to existing nodes or edges within given tolerance. Existing edges and faces may be split by the boundary of the new polygon.



Note

Updating statistics about topologies being loaded via this function is up to caller, see [maintaining statistics during topology editing and population](#).

2.0.0

TopoGeo_AddPolygon

[TopoGeo_AddPoint](#), [TopoGeo_AddLineString](#), [AddFace](#), [CreateTopology](#)

10.6 ST_AddIsoNode

10.6.1 ST_AddIsoNode

ST_AddIsoNode — Adds an isolated node to a topology. The node is added to the topology and its face is set to NULL. The node is added to the topology and its face is set to NULL. The node is added to the topology and its face is set to NULL.

Synopsis

integer **ST_AddIsoNode**(varchar atopology, integer aface, geometry apoint);

설명

atopology 지형의 aface ID(faceid)를 가진 기존 표면포인트 위치가 apoint 인 ૠ립된 노드인ඔ가하ૠ 새 노드의 ID(nodeid)를 반환합포인트 도형과 지형의 공간 참조 시스템(SRID)이 다를 경우, apoint 가 포인트도형이 아닐 경우 포인트가 NULL일 경우 또는 포인트가 기존 경걄선또는 외곽선일지라도) 교차하는 경우 예외가 발생합니다 해당 포위치에 이미 노드가 존재할 경우 예외가 발생합니다.

```
aface &#xac00; NULL&#xc774; &#xc544;&#xb2cc;&#xb370; apoint &#xac00; &#xd45c;&#xba74; &#xb0b4;&#xbd80;&#xc5d0;
&#xc788;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; &#xc608;&#xc678;&#xac00; &#xbc1c;&#xc0dd;&#xd569;&#xb2c8;&#
```

Availability: 1.1



This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X+1.3.1

예시

స고

AddNode, CreateTopology, DropTopology, ST_Intersects

10.6.2 ST AddIsoEdge

```
ST_AddIsoEdge — &#xc9c0;&#xd615;&#xc5d0; &#xb450; &#xac1c;&#xc758; &#xace0;&#xb9bd;&#xb41c; &#xae30;&#xc874;
&#xb178;&#xb4dc; anode&#xc640; anothernode &#xb97c; &#xc5f0;&#xacb0;&#xd558;&#xb294; alinestring &#xb3c4;&
&#xc815;&#xc758;&#xd558;&#xb294; &#xace0;&#xb9bd;&#xb41c; &#xacbd;&#xacc4;&#xc120;&#xc744; &#xcd94;&#xac00;&#x
&#xc0c8; &#xacbd;&#xacc4;&#xc120;&#xc758; ID&#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

Synopsis

```
integer ST_AddIsoEdge(varchar atopology, integer anode, integer anothernode, geometry alinestring);
```

설명

```
&#xc9c0;&#xd615;&#xc5d0;&#xb450;&#xac1c;&#xc758;&#xae0;&#xb9bd;&#xb41c;&#xae30;&#xc874;&#xb178;&#xb4dc;
anode&#xc640;anothernode&#xb97c;&#xc5f0;&#xacb0;&#xd558;&#xb294;alinearstring&#xb3c4;&#xd615;&#xc774;
&#xc815;&#xc758;&#xd558;&#xb294;&#xae0;&#xb9bd;&#xb41c;&#xacbd;&#xacc4;&#xc120;&#xc744;&#xcd94;&#xac00;&#x
&#xc0c8;&#xacbd;&#xacc4;&#xc120;&#xc758;ID(edgeid)&#xb97c;&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

```
alinestring &#xb3c4;&#xd615;&#xacfc; &#xc9c0;&#xd615;&#xc758; &#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a
&#xb2e4;&#xb97c; &#xacbd;&#xc6b0;, &#xc785;&#xb825; &#xc778;&#xc218; &#xac00;&#xc6b4;&#xb370; &#xd558;&#xb098;&
NULL&#xc77c; &#xacbd;&#xc6b0;, &#xd558;&#xb098; &#xc774;&#xc0c1;&#xc758; &#xd45c;&#xba74;&#xc774; &#xb178;&#xb
&#xb2f4;&#xace0; &#xc788;&#xc744; &#xacbd;&#xc6b0;, &#xb610;&#xb294; &#xb178;&#xb4dc;&#xac00; &#xae30;&#xc874;
&#xacbd;&#xacc4;&#xc120;&#xc758; &#xc2dc;&#xc791; &#xb610;&#xb294; &#xc885;&#xb2e8; &#xb178;&#xb4dc;&#xc77c;
&#xacbd;&#xc6b0;, &#xc608;&#xc678;&#xac00; &#xbclc;&#xc0dd;&#xd569;&#xb2c8;&#xb2e4;.
```

```
alinestring &#xc774; anode &#xc640; anothernode &#xac00; &#xc18d; &#xd574; &#xc788; &#xb294; &#xd45c; &#xba74;
&#xb0b4; &#xbd80; &#xc5d0; &#xc5c6; &#xc744; &#xacbd; &#xc6b0.; &#xc608; &#xc678; &#xac00; &#xbc1c; &#xc0dd; &#xd569; &#
```

anode, anothernode, alinestring, ˜, ޑ, ࠐ, 및, ࢅ, 단, 아, 닐, 경, 우, 예, 외, ఀ, 발, 생, 합, 니, 다.

Availability: 1.1



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.4

예, 시

참, 고

ST_AddIsoNode, ST_IsSimple, ST_Within

10.6.3 ST_AddEdgeNewFaces

ST_AddEdgeNewFaces — 새, 경, 계, 선, 을, 추, ఀ, 하, 고, 그, 과, 정, 에, 서, 경, 계, 선, 이, 표, 면, 을, 분, 할, 򬲽, 우, 원, 본, 표, 면, 을, 삭, 제, 하, 고, 새, 표, ,, 개, 로, 교, &#ccb4;, 합, 니, 다.

Synopsis

integer ST_AddEdgeNewFaces(varchar atopolology, integer anode, integer anothernode, geometry acurve);

설, 명

새, 경, 계, 선, 을, 추, ఀ, 하, 고, 그, 과, 정, 에, 򬲽, 계, 선, 이, 표, 면, 을, 분, 할, 할, 경, 우, 원, 표, 면, 을, 삭, 제, 하, 고, 새, 표, 면, 2개, 로, 교, 새, 로, 추, ఀ, 된, 경, 계, 선, 의, ID를, 반, 환, 합, 모, 든, 기, 존, 의, 결, 합, 된, 경, 계, 선, 및, 관, 򬲰, 과, 에, 맞, 춰, 업, 데, 이, 트, 합, 니, 다.

어, 떤, 인, 수, ఀ, NULL일, 경, 우, 주, 어, 진, 노, 때, 려, 지, 지, 않, 은, 경, 우, (노, 드, ఀ, 이, 미, 지, 형, 스, 키, 마, 의, node 테, 이, 블, 에, 존, 재, 합, 니, 다), acurve ఀ, LINESTRING 이, 아, 닐, 경, 우, anode 와, anothernode ఀ, acurve 의, 시, 작, 점, 및, 종, 단, 점, 읔, 닐, 경, 우, 오, 류, ఀ, 발, 생, 합, 니, 다.

acurve 도, 형, 과, 지, 형, 의, 공, ఄ, 참, 조, 시, 스, 다, 를, 경, 우, 예, 외, ఀ, 발, 생, 합, 니, 다.

2.0 ஄, 전, 부, 터, 사, 용, 할, 수, 있, 습, 니, 다.



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.12

예, 시

참, 고

ST_RemEdgeNewFace

ST_AddEdgeModFace

10.6.4 ST_AddEdgeModFace

ST_AddEdgeModFace — Add a new edge to a face, modifying the face's geometry. The edge is added to the face's geometry, and the face's geometry is updated. The edge is added to the face's geometry, and the face's geometry is updated. The edge is added to the face's geometry, and the face's geometry is updated.

Synopsis

integer **ST_AddEdgeModFace**(varchar topology, integer anode, integer anothernode, geometry acurve);

설명

integer **ST_AddEdgeModFace**(varchar topology, integer anode, integer anothernode, geometry acurve);

Note



integer **ST_AddEdgeModFace**(varchar topology, integer anode, integer anothernode, geometry acurve);

integer **ST_AddEdgeModFace**(varchar topology, integer anode, integer anothernode, geometry acurve);

integer **ST_AddEdgeModFace**(varchar topology, integer anode, integer anothernode, geometry acurve);

integer **ST_AddEdgeModFace**(varchar topology, integer anode, integer anothernode, geometry acurve);

integer **ST_AddEdgeModFace**(varchar topology, integer anode, integer anothernode, geometry acurve);



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.13

예시

참고

ST_RemEdgeModFace

ST_AddEdgeNewFaces

10.6.5 ST_RemEdgeNewFace

ST_RemEdgeNewFace — Remove a new edge from a face, creating a new face. The edge is removed from the face's geometry, and a new face is created. The edge is removed from the face's geometry, and a new face is created. The edge is removed from the face's geometry, and a new face is created.

Synopsis

integer **ST_RemEdgeNewFace**(varchar atopology, integer anedge);

설명

경계선을 제거하고, 제거한 경계선을 두 표면을 분할하고 있을 경우, 원표면들을 삭제하고 새 표면 하나대체합니다.

새로 생성된 표면의 ID를 반환하거 또는 새 표면이 생성되지 않은 경우 NULL을 반환합니다. 제거한 경계선을 세계 표면에 매달려 있거나, 고립있거나, 또는 (세계 표면이 다른 쪽표면을 침범하게 할 수 있게) 갇혀 있을 경우 새 표면을 생성할 수 없로 모든 기존의 결합된 경계선 및 관결 과에 맞춰 업데이트합니다.

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).

어떤 인수가 NULL일 경우, 주어진 경알려지지 않은 경우(경계선이 이지형 스키마의 edge 테이블에 존재핷 합니다), 지형 명칭이 유효하지 않경우 오류가 발생합니다.

2.0 버전부터 사용할 수 있습니다.



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.14

예시

참고

ST_RemEdgeModFace

ST_AddEdgeNewFaces

10.6.6 ST_RemEdgeModFace

ST_RemEdgeModFace — 경계선을 제거하고, 제거한 경계선이 두 표면을 분할하고 있을 경우, 표면 하나를 삭제하고 다른 표면을 두 표면의 &#acf5;간을 차지하수정합니다.

Synopsis

integer **ST_RemEdgeModFace**(varchar atopology, integer anedge);

설명

경 계 선 을 제 거 하 고, 제 거 한 경 계 두 표 면 을 분 할 하 고 있 을 경 우, 표 하 나 를 삭 제 하 고 다 른 표 &#ba74; 을 두 표 &#ba74; 의 공 간 을 차 지 하 도 록 수 정 우 선 적 으 로 오 른 쪽 에 있 는 표 &#ba74; 유 지 합 니 다. 역 시 오 른 쪽 의 표 &#ba74; 유 지 하 는 ST_AddEdgeModFace 함 수 와 대 칭 을 이 위 해 서 입 니 다. 제 거 한 경 계 선 의 위 치 에 남 아 있 는 표 &#ba74; 의 ID 를 반 환 모 든 기 존 의 결 합 된 경 계 선 및 관 결 과 에 맞 춰 업 데 이 트 합 니 다.

기존 TopoGeometry를 정의하는 데 쓰이는
경계선을 제거하지 않습니다. 두
표면 가운데 하나만 (다른 하나가
정의하는 TopoGeometry는 없으며) 어떤 TopoGe-
ometry를 정의하는 경우 두 표면을 합&#xc
않습니다.

```
&#xc5b4;&#xb5a4; &#xc778;&#xc218;&#xac00; NULL&#xc77c; &#xacbd;&#xc6b0;, &#xc8fc;&#xc5b4;&#xc9c4; &#xacbd;&#xac
&#xc54c;&#xb824;&#xc9c0;&#xc9c0; &#xc54a;&#xc740; &#xacbd;&#xc6b0;(&#xacbd;&#xacc4;&#xc120;&#xc774; &#xc774;&#xc
&#xc9c0;&#xd615; &#xc2a4;&#xd0a4;&#xb9c8;&#xc758; edge &#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xc874;&#xc7ac;&#xd5
&#xd569;&#xb2c8;&#xb2e4;), &#xc9c0;&#xd615; &#xba85;&#xce6d;&#xc774; &#xc720;&#xd6a8;&#xd558;&#xc9c0; &#xc54a;&
&#xacbd;&#xc6b0; &#xc624;&#xb958;&#xac00; &#xbc1c;&#xc0dd;&#xd569;&#xb2c8;&#xb2e4;.
```

2.0 ಄전&#bd80;터 사용할 수 있습니다.



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.15

예시

స고

ST_AddEdgeModFace

ST_RemEdgeNewFace

10.6.7 ST_ChangeEdgeGeom

[illegible]

Synopsis

text **ST** **ChangeEdgeGeom**(varchar atopology, integer anedge, geometry acurve);

설명

지형 구조를 건드리는 일 없이 경&#
형상을 변경합니다.

If any arguments are null, the given edge does not exist in the edge table of the topology schema, the `acurve` is not a `LINestring`, or the modification would change the underlying topology then an error is thrown.

acurve 도형과 지형의 공간 참조 시스실를 경우 예외가 발생합니다.

새 acurve 가 단순 도형이 아닐 경우,오다 발생합니다.

예전 위치에서 새 위치로 경계선이동시킬 때 장애물에 부딪히는 경우 오류가 발생합니다.

1.1.0 버전부터 사용할 수 있습니다.

개선 사항: 2.0.0버전부터 지형의 일괌강제합니다.



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details X.3.6

예시

```
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
```

స고

ST_AddEdgeModFace

ST_RemEdgeModFace

ST_ModEdgeSplit

10.6.8 ST_ModEdgeSplit

ST_ModEdgeSplit — `기존경계선을따라새노&b4dc;추가한다음원본경계선을수정새경계선을추가해서경계선을분할합니다`.

Synopsis

```
integer ST_ModEdgeSplit(varchar atopology, integer anedge, geometry apoint);
```

설명

기존경계선을따라새노드를추다음,원본경계선을수정하ૠ새경계선을추가해서경계선을분기존의모든결합된경계선들및관계성을새로이분할된경계선맞춰업데이트합니다.새로추가노드의식별자를반환합니다.

Availability: 1.1

ST ModEdgesSplit



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

예시

```
-- &#xacbd;&#xacc4;&#xc120; &#xcd94;&#xac00; --
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227592 893910, 227600 893910)', 26986) ) As edgeid;

-- edgeid --
3

-- &#xacbd;&#xacc4;&#xc120; &#xbd84;&#xd560; --
SELECT topology.ST_ModEdgeSplit('ma_topo', 3, ST_SetSRID(ST_Point(227594,893910),26986) ) As node_id;
node_id
-----
7
```

స고

ST_NewEdgesSplit, ST_ModEdgeHeal, ST_NewEdgeHeal, AddEdge

10.6.9 ST_ModEdgeHeal

ST_ModEdgeHeal — Heals two edges by deleting the node connecting them, modifying the first edge and deleting the second edge. Returns the id of the deleted node.

Synopsis

```
int ST_ModEdgeHeal(varchar atopology, integer anedge, integer anotheredge);
```

설명

Heals two edges by deleting the node connecting them, modifying the first edge and deleting the second edge. Returns the id of the deleted node. Updates all existing joined edges and relationships accordingly.

2.0 ಄전&#bd80;터 사용할 수 있습니다.



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

స고

ST_ModEdgeSplit ST_NewEdgesSplit

10.6.10 ST_NewEdgeHeal

ST_NewEdgeHeal — Heals two edges by deleting the node connecting them, deleting both edges, and replacing them with an edge whose direction is the same as the first edge provided.

Synopsis

```
int ST_NewEdgeHeal(vchar atopology, integer anedge, integer anotheredge);
```

설명

Heals two edges by deleting the node connecting them, deleting both edges, and replacing them with an edge whose direction is the same as the first edge provided. Returns the id of the new edge replacing the healed ones. Updates all existing joined edges and relationships accordingly.

2.0 ಄전부터 사용할 수 있습니다.



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

స고

ST_ModEdgeHeal ST_ModEdgeSplit ST_NewEdgesSplit

10.6.11 ST MovelsoNode

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 anode, geometry apoint);
```

설명

```
&#xc9c0;&#xd615;&#xc548;&#xc5d0;&#xc788;&#xb294;&#xace0;&#xb9bd;&#xb41c;&#xb178;&#xb4dc;&#xb97c;&#xc5b4;&#xc704;&#xce58;&#xc5d0;&#xc11c;&#xb2e4;&#xb978;&#xc704;&#xce58;&#xb85c;&#xc774;&#xb3d9;&#xd569;&#xb2c8;&#xc0c8;apoint&#xb3c4;&#xd615;&#xc758;&#xc704;&#xce58;&#xc5d0;&#xb178;&#xb4dc;&#xac00;&#xc874;&#xc7ac;&#xacbd;&#xc6b0;&#xc624;&#xb958;&#xac00;&#xbc1c;&#xc0dd;&#xd569;&#xb2c8;&#xb2e4;.
```

If any arguments are null, the `apoint` is not a point, the existing node is not isolated (is a start or end point of an existing edge), new node location intersects an existing edge (even at the end points) or the new location is in a different face (since 3.2.0) then an exception is thrown.

포인트 도형과 지형의 공간 참조
시스템(SRID)이 다를 경우 예외가 발&#xc0

2.0.0 버전부터 사용할 수 있습니다.

Enhanced: 3.2.0 ensures the nod cannot be moved in a different face



This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X.3.2

예시

```
-- &#xd45c;&#xba74;&#xc774; &#xc5c6;&#xb294; &#xace0;&#xb9bd; &#xb178;&#xb4dc;&#xb97c;  ←
    &#xcd94;&#xac00; --
SELECT topology.ST_AddIsoNode('ma_topo',  NULL, ST_GeomFromText('POINT(227579 893916)',  ←
    26986) ) As nodeid;
    nodeid
-----
    7
-- &#xc0c8; &#xb178;&#xb4dc;&#xb97c; &#xc774;&#xb3d9; --
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
```

󌰸고

ST_AddIsoNode

10.6.12 ST_NewEdgesSplit

ST_NewEdgesSplit — 기존 경계선을 따라 새 노드 추가한 다음 원본 경계선을 삭제 새 경계선 2개 로 교체해서 경계선 분할합니 다 새 경계선들을 &#acb0;합새 로 생성된 노드의 ID를 반환합니

Synopsis

integer **ST_NewEdgesSplit**(varchar atopology, integer anedge, geometry apoint);

설명

기존 경계선을 따라 apoint 포인트 위새 노드를 생성한 다음 원본 경계선을 삭제 하고 새 경계선 2개 교체해서 경계선 ID anedge 인 경계선을 분할합니 다 새 경계선들을 &#acb0;합새 로 생성된 노드의 ID를 반환합니 기존 의 모든 &#acb0;합된 경계선들 &#bc0f; 관계성을 &#bd84;할된 경계선에 맞춰 업데이트합니다

포인트 도형과 지형의 공간 &#ccc38;조 시스템(SRID)이 다를 경우 apoint 가 포인트 도형이 아닐 경우 포인트의 위치에 이미 노드가 존재할 경우 경계선이 기경계선과 일치하지 않을 경우 또포인트가 경계선 안에 있지 않을 경우 예외가 &#bc1c;생합니다

Availability: 1.1



This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X.3.8

예시

```
-- &#xacbd;&#xacc4;&#xc120; &#xcd94;&#xac00; --
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575 893917,227592 893900) ←
', 26986) ) As edgeid;
-- &#acb0;&#xacfc; --
edgeid
-----
2
-- &#xc0c8; &#xacbd;&#xacc4;&#xc120;&#xc744; &#xbd84;&#xd560; --
SELECT topology.ST_NewEdgesSplit('ma_topo', 2, ST_GeomFromText('POINT(227578.5 893913.5)', ←
26986) ) As newnodeid;
newnodeid
-----
6
```

ST_RemoveIsoNode

ST_ModEdgeSplit ST_ModEdgeHeal ST_NewEdgeHeal AddEdge

10.6.13 ST_RemoveIsoNode

ST_RemoveIsoNode — Removes an isolated node and returns description of action. If the node is not isolated, then an exception is thrown.

Synopsis

text **ST_RemoveIsoNode**(varchar atopology, integer anode);

ST_RemoveIsoNode

Removes an isolated node and returns description of action. If the node is not isolated, then an exception is thrown.

Availability: 1.1



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3

ST_RemoveIsoNode

```
-- Removes an isolated node and returns description of action. If the node is not isolated, then an exception is thrown.
SELECT topology.ST_RemoveIsoNode('ma_topo', 7) As result;
-----
Isolated node 7 removed
```

ST_RemoveIsoNode

ST_AddIsoNode

10.6.14 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);

ST_RemoveIsolatedEdge

Removes an isolated edge and returns description of action. If the edge is not isolated, then an exception is thrown.

Availability: 1.1



This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3

ST_RemoveIsolatedEdge

```
-- <math>\text{ST\_RemoveIsolatedEdge}(\text{geometry} \text{ geometry}, \text{integer} \text{ edge\_id}) \text{ returns } \text{text}</math>
--
SELECT topology.ST_RemoveIsolatedEdge('ma_topo', 7) As result;
-----
Isolated node 7 removed
```

ST_AddIsoNode**ST_AddIsoNode****10.7 ST_RemoveIsolatedEdge****10.7.1 GetEdgeByPoint**

GetEdgeByPoint — Finds the edge-id of an edge that intersects a given point.

Synopsis

integer **GetEdgeByPoint**(varchar atopology, geometry apoint, float8 toll);

ST_GetEdgeById

Retrieves the id of an edge that intersects a Point.

ST_GetEdgeById(varchar atopology, geometry apoint, float8 tolerance) returns integer

tolerance = 0

If apoint doesn't intersect an edge, returns 0 (zero).

ST_GetEdgeById(varchar atopology, geometry apoint, float8 tolerance) returns integer

**Note**

ST_GetEdgeById(varchar atopology, geometry apoint, float8 tolerance) returns integer

GEOS

2.0.0

GetEdgeByPoint

AddEdge — Adds a new edge to a topology. The edge is defined by two points (start and end) and a tolerance. The edge is added to the topology and the tolerance is set to the tolerance of the topology.

```
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;

-- ERROR: Two or more edges found
```

GetFaceByPoint

AddEdge, **GetNodeByPoint**, **GetFaceByPoint**

10.7.2 GetFaceByPoint

GetFaceByPoint — Finds face intersecting a given point.

Synopsis

integer **GetFaceByPoint**(varchar atopoology, geometry apoint, float8 toll);

FindFaceByPoint

Finds a face referenced by a Point, with given tolerance.

The function will effectively look for a face intersecting a circle having the point as center and the tolerance as radius.

If no face intersects the given query location, 0 is returned (universal face).

If more than one face intersect the query location an exception is thrown.

2.0.0 — Enhanced: 3.2.0 more efficient implementation and clearer contract, stops working with invalid topologies.

Enhanced: 3.2.0 more efficient implementation and clearer contract, stops working with invalid topologies.

FindFaceByPoint

```
SELECT topology.GetFaceByPoint('ma_topo',geom, 10) As withlmtol, topology.GetFaceByPoint('ma_topo',geom,0) As withnotol
FROM ST_GeomFromEWKT('POINT(234604.6 899382.0)') As geom;
withlmtol | withnotol
-----+-----
1 | 0
```

```
SELECT topology.GetFaceByPoint('ma_topo',geom, 1) As nearnode
FROM ST_GeomFromEWKT('POINT(227591.9 893900.4)') As geom;

-- ERROR: Two or more faces found
```


GetFaceContainingPoint

GetFaceContainingPoint, **AddFace**, **GetNodeByPoint**, **GetEdgeByPoint**

10.7.3 GetFaceContainingPoint

GetFaceContainingPoint — Finds the face containing a point.

Synopsis

integer **GetFaceContainingPoint**(text atopology, geometry apoint);

Return Value

Returns the id of the face containing a point.

An exception is thrown if the point falls on a face boundary.



Note

The function relies on a valid topology, using edge linking and face labeling.

Availability: 3.2.0

GetFaceGeometry

ST_GetFaceGeometry

10.7.4 GetNodeByPoint

GetNodeByPoint — Finds the node-id of a node at a point location.

Synopsis

integer **GetNodeByPoint**(varchar atopology, geometry apoint, float8 tol1);

Return Value

Retrieves the id of a node at a point location.

The function returns an integer (id-node) given a topology, a POINT and a tolerance. If tolerance = 0 means exact intersection, otherwise retrieves the node from an interval.

If apoint doesn't intersect a node, returns 0 (zero).

If use tolerance > 0 and there is more than one node near the point then an exception is thrown.



Note

GetNodeByPoint(topology, point, tolerance) = ST_Intersects(topology, point, tolerance) ? ST_DWithin(topology, point, tolerance) : 0

GEOS 3.10.0

2.0.0

GetNodeByPoint

GetNodeByPoint — Returns the ID of the node closest to the point. The ID is the same as the one returned by **GetFaceByPoint**.

```
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;

-- ERROR: Two or more nodes found
```

GetEdgeByPoint

GetEdgeByPoint, **GetFaceByPoint**

10.7.5 GetTopologyID

GetTopologyID — Returns the ID of the topology. The ID is the same as the one returned by **GetFaceByPoint**.

Synopsis

integer **GetTopologyID**(varchar toponame);

GetTopologyID

GetTopologyID — Returns the ID of the topology. The ID is the same as the one returned by **GetFaceByPoint**.

Availability: 1.1

GetTopologyID

```
SELECT topology.GetTopologyID('ma_topo') As topo_id;
topo_id
-----
      1
```

CreateTopology

CreateTopology, **DropTopology**, **GetTopologyName**, **GetTopologySRID**

10.7.6 GetTopologySRID

GetTopologySRID — Returns the SRID of the topology. The SRID is the same as the one returned by **GetFaceByPoint**.

Synopsis

integer **GetTopologyID**(varchar toponame);

설명

지형의 명칭을 입력받아 topology.topology 테있는 지형의 공간 참조 식별자를 반환합니다.

2.0.0 버전부터 사용할 수 있습니다.

예시

```
SELECT topology.GetTopologySRID('ma_topo') As SRID;
SRID
-----
4326
```

참고

CreateTopology, DropTopology, GetTopologyName, GetTopologyID

10.7.7 GetTopologyName

GetTopologyName — 지형의 ID를 입력받아 지형(스명칭을 반환합니다.

Synopsis

varchar **GetTopologyName**(integer topology_id);

설명

지형의 지형 ID를 입력받아 topology.topology 테지형의 지형명(스키마)을 반환합니다.

Availability: 1.1

예시

```
SELECT topology.GetTopologyName(1) As topo_name;
topo_name
-----
ma_topo
```

참고

CreateTopology, DropTopology, GetTopologyID, GetTopologySRID

10.7.8 ST_GetFaceEdges

ST_GetFaceEdges — aface

Synopsis

getfaceedges_returntype **ST_GetFaceEdges**(varchar atopology, integer aface);

설명

aface

2.0

2.0



This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.5

예시

```
-- &#xd45c;&#xba74; 1&#xc758; &#xacbd;&#xacc4;&#xb97c; &#xc774;&#xb8e8;&#xb294;  ←
    &#xacbd;&#xacc4;&#xc120;&#xb4e4;&#xc744; &#xbc18;&#xd658;
SELECT (topology.ST_GetFaceEdges('tt', 1)).*;
-- &#xacb0;&#xacfc; --
sequence | edge
-----+-----
1 | -4
2 | 5
3 | 7
4 | -6
5 | 1
6 | 2
7 | 3
(7 rows)
```

```
-- &#xd45c;&#xba74; 1&#xc758; &#xacbd;&#xacc4;&#xb97c; &#xc774;&#xb8e8;&#xc740;  ←
    &#xacbd;&#xacc4;&#xc120;&#xb4e4;&#xc758; &#xb3c4;&#xd615;&#xacfc;
-- &#xc21c;&#xcc28; &#xbc0f; &#xacbd;&#xacc4;&#xc120; ID&#xb97c; &#xbc18;&#xd658;
-- &#xb3c4;&#xd615;&#xacfc; &#xc21c;&#xcc28;&#xb9cc; &#xd544;&#xc694;&#xd560;  ←
    &#xacbd;&#xc6b0;, ST_GetFaceGeometry &#xd568;&#xc218;&#xb97c; &#xc774;&#xc6a9;&#xd560;  ←
    &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
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;
```

참고

GetRingEdges, AddFace, ST_GetFaceGeometry

10.7.9 ST_GetFaceGeometry

ST_GetFaceGeometry — Returns the geometry of a face in a topology. The face is identified by its ID. The topology is identified by its name.

Synopsis

geometry ST_GetFaceGeometry(varchar atopology, integer aface);

Availability: 1.1

This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.16

Availability: 1.1



This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.16

Example:

```
-- AddFace topology.ST_GetFaceGeometry('ma_topo', 1) As facegeomwkt;
-- WKT topology.ST_GetFaceGeometry('ma_topo', 1) As facegeomwkt;
SELECT ST_AsText(topology.ST_GetFaceGeometry('ma_topo', 1)) As facegeomwkt;
-- WKT topology.ST_GetFaceGeometry('ma_topo', 1) As facegeomwkt;
-- WKT topology.ST_GetFaceGeometry('ma_topo', 1) As 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))
```

Example:

AddFace

10.7.10 GetRingEdges

GetRingEdges — Returns the edges of a ring in a topology. The ring is identified by its ID. The topology is identified by its name.

Synopsis

getfaceedges_returntype GetRingEdges(varchar atopology, integer aring, integer max_edges=null);

설명

입력 경계선 쪽을 따라 탐색한 부있는 경계선 식별자 들의 정렬된 집합을 반환합니다. 각 출력물은 순차(sequence) 및 부호가 있는 경계선 ID(edgeid)이루어집니다. 순차 번호는 1부터 시작합니다.

양의 부호를 가진 경계선 ID를 넘겨 경우 해당 경계선의 왼쪽에서 탐시작해서 경계선의 방향으로 진음의 부호를 가진 경계선 ID를 넘겨 경우 해당 경계선의 오른쪽에서 탐색을 시작해서 경계선 방향의 반대로 진행합니다.

max_edges 가 NULL이 아닐 경우 함수가 해당 개수 이상의 레코드를 반환하지 않습니다. 유효하지 않을 수도 있지형을 처리할 때 이 파라미터를 안전 장치로 쓸 수 있다는 의미입이

**Note**

이 함수는 메타데이터를 링크하는 경계선 고리를 이용합니다.

2.0.0 버전부터 사용할 수 있습니다.

참고

ST_GetFaceEdges, GetNodeEdges

10.7.11 GetNodeEdges

GetNodeEdges — 입력 노드를 따르는 경계선정렬된 집합을 반환합니다.

Synopsis

getfaceedges_returntype **GetNodeEdges**(varchar atopology, integer anode);

설명

입력 노드를 따르는 경계선들의 정렬된 집합을 반환합니다. 각 출력물은 순차 및 부호를 가진 경계선 ID로 구성됩니다. 순차 번호는 1부터 시작합니다. 음의 부호를 가진 경계선은 입력 노드로 부터 시작합니다. 음의 부호를 가진 경계선은 입력 노드로 끝납페쇄형 경계선은 두 번 (두 부호를 가지고) 표출될 ಃ입니다. 북쪽부시작하는 시계 방향으로 정렬됩이

**Note**

The `ST_Polygonize` function takes a `geometry` argument and returns a `geometry`. It is used to create a polygon from a set of edges. The function is defined in the `PostGIS` library.

2.0 The `ST_Polygonize` function is defined in the `PostGIS` library.

ST_Polygonize

`getfaceedges_returntype`, `GetRingEdges`, `ST_Azimuth`

10.8 ST_Polygonize

10.8.1 Polygonize

Polygonize — Finds and registers all faces defined by topology edges.

Synopsis

text **Polygonize**(varchar toponame);

ST_Polygonize

Registers all faces that can be built out a topology edge primitives.

The `ST_Polygonize` function is defined in the `PostGIS` library. It is used to create a polygon from a set of edges. The function is defined in the `PostGIS` library.

**Note**

The `ST_Polygonize` function is defined in the `PostGIS` library. It is used to create a polygon from a set of edges. The function is defined in the `PostGIS` library.

**Note**

The `ST_Polygonize` function is defined in the `PostGIS` library. It is used to create a polygon from a set of edges. The function is defined in the `PostGIS` library.

2.0.0 The `ST_Polygonize` function is defined in the `PostGIS` library.

ST_Polygonize

`AddFace`, `ST_Polygonize`

10.8.2 AddNode

```
AddNode — &#xc124;&#xc815;&#xd55c; &#xc9c0;&#xd615; &#xc2a4;&#xd0a4;&#xb9c8;&#xc758; &#xb178;&#xb4dc;
&#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xd3ec;&#xc778;&#xd2b8; &#xb178;&#xb4dc;&#xb97c; &#xcd94;&#xac00;&#xd558;&#
&#xc0c8; &#xb178;&#xb4dc;&#xc758; &#xb178;&#xb4dc; ID(nodeid)&#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
&#xc0c8; &#xd3ec;&#xc778;&#xd2b8; &#xc704;&#xce58;&#xc5d0; &#xc774;&#xbbf8; &#xb178;&#xb4dc;&#xac00; &#xc874;&#
&#xacbd;&#xc6b0;,&#xae30;&#xc874; &#xb178;&#xb4dc; ID&#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

Synopsis

```
integer AddNode(varchar toponame, geometry apoint, boolean allowEdgeSplitting=false, boolean computeContainingFace=false);
```

설명

```
&#xc124;&#xc815;&#xd55c;&#xc9c0;&#xd615;&#xc2a4;&#xd0a4;&#xb9c8;&#xc758;&#xb178;&#xb4dc;&#xd14c;&#xc774;&#xd3ec;&#xc778;&#xd2b8;&#xb178;&#xb4dc;&#xb97c;&#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;. AddEdge &#xd568;&#xd638;&#xcd9c;&#xd560;&#xb54c;&#xacbd;&#xacc4;&#xc120;&#xc758;&#xc2dc;&#xc791;&#xc810;&#xbc0f;&#xc885;&#xc790;&#xb3d9;&#xc801;&#xc73c;&#xb85c;&#xcd94;&#xac00;&#xd558;&#xbbc0;&#xb85c;,&#xacbd;&#xacc4;&#xc120;&#xb178;&#xb4dc;&#xb97c;&#xb530;&#xb85c;&#xcd94;&#xac00;&#xd560;&#xd544;&#xc694;&#xb294;&#xc5c6;&#xc2b5;&#xc0c8;&#xb178;&#xb4dc;&#xb97c;&#xac00;&#xb85c;&#xc9c0;&#xb974;&#xb294;&#xacbd;&#xacc4;&#xc120;&#xc774;&#xc788;&#xc744;&#xacbd;&#xc6b0;,allowEdgeSplitting &#xd30c;&#xb77c;&#xbbbf8;&#xd130;&#xac12;&#xc5d0;&#xb530;&#xb77c;&#xc608;&#xc678;&#xac00;&#xbc1c;&#xc0dd;&#xd558;&#xac70;&#xb098;&#xb610;&#xb294;&#xd574;&#xacbd;&#xacc4;&#xc120;&#xc744;&#xbd84;&#xd560;&#xd569;&#xb2c8;&#xb2e4;.
```

```
computeContainingFace &#xac00; &#xc38;&#xc77c; &#xacbd;&#xc6b0; &#xc0c8;&#xb85c; &#xcd94;&#xac00;&#xb41c;
&#xb178;&#xb4dc;&#xb97c; &#xc815;&#xd655;&#xd788; &#xb2f4;&#xace0; &#xc788;&#xb294; &#xd45c;&#xba74;&#xc744;
&#xacc4;&#xc0b0;&#xd569;&#xb2c8;&#xb2e4;.
```

Note



```
apoint  &#xb3c4;&#xd615;&#xc758; &#xc704;&#xce58;&#xc5d0; &#xc774;&#xbbf8; &#xb178;&#xb4dc;&#xac00;
&#xc874;&#xc7ac;&#xd560;          &#xacbd;&#xc6b0;,          &#xc0c8;          &#xb178;&#xb4dc;&#xb97c;
&#xcd94;&#xac00;&#xd558;&#xc9c0; &#xc54a;&#xace0; &#xae30;&#xc874; &#xb178;&#xb4dc; ID(nodeid)&#xb97c;
&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

2.0.0 버전부터 사용할 수 있습니다.

예시

```
SELECT topology.AddNode('ma_topo', ST_GeomFromText('POINT(227641.6 893816.5)', 26986) ) As ↵
    nodeid;
-- &#xacb0;&#xacfc; --
nodeid
-----
4
```

స고

AddEdge, CreateTopology

10.8.3 AddEdge

AddEdge — Adds a new edge to the topology. The edge is defined by the `toponame` and `aline` parameters. The `edgeid` parameter is optional and can be used to specify the ID of the edge to be added. The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`. The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`.

Synopsis

integer **AddEdge**(varchar toponame, geometry aline);

<code>AddEdge</code>

<code>AddEdge</code> Adds a new edge to the topology. The edge is defined by the `toponame` and `aline` parameters. The `edgeid` parameter is optional and can be used to specify the ID of the edge to be added. The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`. The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`.

Note



`aline` must be a line segment. The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`. The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`.

Note



The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`. The `edgeid` parameter is only valid if the `toponame` parameter is set to `'ma_topo'`.

GEOS `GEOSGeomFromText`

2.0.0 `GEOSGeomFromText`

<code>AddEdge</code>

```
SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227575.8 893917.2,227591.9 893900.4)', 26986) ) As edgeid;
-- <code>AddEdge</code>
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;
-- &#xacb0;&#xacf; --
edgeid
-----
2

SELECT topology.AddEdge('ma_topo', ST_GeomFromText('LINESTRING(227591.2 893900, 227591.9
893900.4,
227704.5 893778.5)', 26986) ) As edgeid;
-- &#xc624;&#xb958; &#xbcl;&#xc0dd; --
ERROR:  Edge intersects (not on endpoints) with existing edge 1

```

참고

[TopoGeo_AddLineString, CreateTopology, Section 4.5](#)

10.8.4 AddFace

AddFace — 지형에 표면 원시형(face primitive)을 등록한그 식별자를 반환합니다.

Synopsis

integer **AddFace**(varchar toponame, geometry apolygon, boolean force_new=false);

설명

지형에 표면 원시형(face primitive)을 등록한그 식별자를 반환합니다.

새로 추가된 표면의 left_face 및 right_face 항목정환한 값을 가지도록 그 외곽선형성하는 경계선들 및 표면이 담있는 경계선들을 업데이트합니표면이 담고 있는 고립된 노드들이 containing_face 항목이 정환한 값을 가지도록 업데이트할 것입니다.



Note

이 함수는 edge 테이블의 next_left_edge 및 next_right_edge 항목을 이용하지도 설정하지도 않습니다.

대상 지형이 유효하다고 (자체 교경계선을 담고 있지 않다고) 가정기존 경계선들이 폴리곤 외곽선완전하게 정의하고 있지 않거나 또는 폴리곤이 기존 표면과 중즩경우 예외가 발생합니다.

apolygon 도형과 동일한 표면이 이򻯸 존경우 force_new 가 (기본값인) 거짓이면

```
&#xae30;&#xc874; &#xd45c;&#xba74;&#xc758; &#xd45c;&#xba74; ID&#xb97c; &#xbc18;&#xd658;&#xd558;&#xae0;,
force_new&#xac00; &#xcc38;&#xc774;&#xba74; &#xc0c8;&#xb85c; &#xb4f1;&#xb85d;&#xb41c; &#xd45c;&#xba74;&#xc5d0;
&#xc0c8; ID&#xb97c; &#xd560;&#xb2f9;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

Note



Note!

```
&#xae30;&#xc874;      &#xd45c;&#xba74;&#xc744;      &#xc0c8;&#xb85c;      &#xb4f1;&#xb85d;&#xd560;
&#xacbd;&#xc6b0;      (force_new      =      true)      &#xacbd;&#xacc4;&#xc120;;      &#xb178;&#xb4dc;;
&#xad00;&#xacc4;&#xc131;      &#xd14c;&#xc774;&#xbe14;&#xc5d0;      &#xc788;&#xb294;      &#xae30;&#xc874;
&#xd45c;&#xba74;&#xc5d0;      &#xb300;&#xd55c;      &#xc608;&#xc804;      &#xcc38;&#xc870;&#xb4e4;&#xc744;
&#xd574;&#xacb0;&#xd558;&#xae30;      &#xc704;&#xd55c;      &#xc5b4;&#xb5a4;      &#xc791;&#xb3d9;&#xb3c4;
&#xd558;&#xc9c0;      &#xc54a;&#xc744;      &#xac83;&#xc785;&#xb2c8;&#xb2e4;.      &#xb610;      &#xae30;&#xc874;
&#xd45c;&#xba74;      &#xb808;&#xcf54;&#xb4dc;&#xc758;      MBR      &#xd56d;&#xbaa9;&#xb3c4;
&#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xd558;&#xc9c0;      &#xc54a;&#xc744;      &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
&#xd638;&#xcd9c;&#xc790;&#xac00;      &#xd574;&#xacb0;&#xd574;&#xc57c;      &#xd560;
&#xbb38;&#xc81c;&#xc785;&#xb2c8;&#xb2e4;.
```

Note



Note!

```
apolygon  &#xb3c4;&#xd615;&#xc758;    srid  &#xc640;    &#xc9c0;&#xd615;&#xc5d0;    &#xb300;&#xd574;
&#xc815;&#xc758;&#xb41c;    srid    &#xac00;    &#xb3d9;&#xc77c;&#xd574;&#xc57c;&#xb9cc;
&#xd569;&#xb2c8;&#xb2e4;.    &#xadf8;&#xb807;&#xc9c0;    &#xc54a;&#xc744;    &#xacbd;&#xc6b0;
&#xc720;&#xd6a8;&#xd558;&#xc9c0;    &#xc54a;&#xc740;    &#xacf5;&#xac04;    &#xcc38;&#xc870;
&#xc2dc;&#xc2a4;&#xd15c;    &#xc624;&#xb958;&#xac00;    &#xbc1c;&#xc0dd;&#xd560;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

2.0.0 버전부터 사용할 수 있습니다.

예시

```
-- &#xba3c;&#xc800; generate_series&#xb97c; &#xbc18;&#xbcf5;&#xc790;(iterator)&#xb85c; <-
&#xc774;&#xc6a9;&#xd574;&#xc11c; &#xacbd;&#xacc4;&#xc120;&#xc744; <-
&#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;.
-- (&#xb2e4;&#xc74c; &#xcffc;&#xb9ac;&#xb294; generate_series&#xc758; <-
&#xcd5c;&#xb300;&#xac12; &#xc124;&#xc815; &#xb54c;&#xbb38;&#xc5d0; <-
&#xd3ec;&#xc778;&#xd2b8; 10,000&#xac1c; &#xbbf8;&#xb9cc;&#xc778; <-
&#xd3f4;&#xb9ac;&#xace4;&#xc5d0; &#xb300;&#xd574;&#xc11c;&#xb9cc; <-
&#xb3d9;&#xc791;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.)
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;
-- &#xacb0;&#xacfc; --
    edgeid
-----
    3
    4
    5
    6
    7
    8
```

```

    9
    10
    11
    12
(10 rows)
--  &#xadb8; &#xb2e4;&#xc74c; &#xd45c;&#xba74;&#xc744;  ←
    &#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;. --

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;
-- &#xacb0;&#xacfc; --
faceid
-----
1

```

참고

AddEdge, CreateTopology, Section 4.5

10.8.5 ST_Simplify

ST_Simplify — 더글러스-패커(Douglas-Peucker) 알고리즘통해 입력 TopoGeometry 의 "단순화된" 도형 버전을 반환합니다.

Synopsis

geometry **ST_Simplify**(TopoGeometry tg, float8 tolerance);

설명

각 구성 요소의 경계선에 더글러패커(Douglas-Peucker) 알고리즘을 작용해서 입력 TopoGeometry 의 "단순화된" 도형 버전을 반환합니다.

Note



반환된 도형이 단순하지
 않거나, 유효하지 않을 수도
 있습니다.
 단순성/유효성을 유지하는
 데 구성 요소의 경계선을
 분할하는 것이 도움이 될 수도
 있습니다.

GEOS 모듈로 실행

2.1.0 버전부터 사용할 수 있습니다.

ST_Simplify

ST_IsSimple, **ST_IsValid**, **ST_ModEdgeSplit**

10.8.6 RemoveUnusedPrimitives

RemoveUnusedPrimitives — Removes topology primitives which not needed to define existing TopoGeometry objects.

Synopsis

```
int RemoveUnusedPrimitives(text topology_name, geometry bbox);
```

ST_ModEdgeHeal, **ST_RemEdgeModFace**

Finds all primitives (nodes, edges, faces) that are not strictly needed to represent existing TopoGeometry objects and removes them, maintaining topology validity (edge linking, face labeling) and TopoGeometry space occupation.

No new primitive identifiers are created, but rather existing primitives are expanded to include merged faces (upon removing edges) or healed edges (upon removing nodes).

Availability: 3.3.0

ST_ModEdgeHeal, **ST_RemEdgeModFace**

10.9 TopoGeometry

10.9.1 CreateTopoGeom

CreateTopoGeom — Creates a topology geometry object for layer denoted by `layer_id` and registers it in the relations table in the `toponame` schema. `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.

Synopsis

```
topogeometry CreateTopoGeom(varchar toponame, integer tg_type, integer layer_id, topoelementarray tg_objs);
topogeometry CreateTopoGeom(varchar toponame, integer tg_type, integer layer_id);
```

ST_ModEdgeHeal, **ST_RemEdgeModFace**

Creates a topogeometry object for layer denoted by `layer_id` and registers it in the relations table in the `toponame` schema. `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.

ST_ModEdgeHeal, **ST_RemEdgeModFace**

ST_ModEdgeHeal, **ST_RemEdgeModFace**

Availability: 1.1

예시: 기존 경계선으로부터 형성

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);
```

예시: 면형 도형을 가장 잘 추측한 TopoGeometry로 변환

표면들의 집합으로 형성돼야 하도형을 가지고 있다고 해봅시다. 예를 들어 blockgroups 테이블이 있는데 각 블록 그룹의 TopoGeometry를 알고 싶습니다 데이터가 완벽하게 정렬돼 있다다음과 같이 할 수 있습니다:

```
-- TopoGeometry &#xc5f4;&#xc744; &#xc0dd;&#xc131; --
SELECT topology.AddTopoGeometryColumn(
    'topo_boston',
    'boston', 'blockgroups', 'topo', 'POLYGON');

-- addtopogeometrycolumn --
1

-- &#xbaa8;&#xb4e0; &#xac83;&#xc774; &#xacbd;&#xacc4;&#xc120;&#xacfc; ←
&#xc644;&#xbcbd;&#xd558;&#xac8c; &#xc815;&#xb82c;&#xb41c;&#xb2e4;&#xace0; ←
&#xac00;&#xc815;&#xd558;&#xace0;
-- &#xc5f4;&#xc744; &#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xd569;&#xb2c8;&#xb2e4; .
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 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;
```

TopoGeometry

AddTopoGeometryColumn, toTopoGeom ST_CreateTopoGeo, ST_GetFaceGeometry, TopoElementArray, TopoElementArray_Agg

10.9.2 toTopoGeom

toTopoGeom — Converts a simple Geometry into a topo geometry.

Synopsis

topogeometry **toTopoGeom**(geometry geom, varchar toponame, integer layer_id, float8 tolerance);

topogeometry **toTopoGeom**(geometry geom, topogeometry topogeom, float8 tolerance);

TopoGeometry

TopoGeometry (geometry geom, varchar toponame, integer layer_id, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

TopoGeometry (topogeometry topogeom, float8 tolerance)

예시

다음은 완전히 자급자족하는 독작업 흐름(workflow)입니다.

```
-- &#xc544;&#xc9c1; &#xc9c0;&#xd615;&#xc744; &#xc124;&#xc815;&#xd558;&#xc9c0;  ←
&#xc54a;&#xc558;&#xb2e4;&#xba74; &#xb2e4;&#xc74c; &#xcffc;&#xb9ac;&#xb97c;  ←
&#xc2e4;&#xd589;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
-- &#xc5b4;&#xb5a4; &#xd5c8;&#xc6a9; &#xc624;&#xcc28;&#xb3c4;  ←
&#xd5c8;&#xc6a9;&#xd558;&#xc9c0; &#xc54a;&#xb294; &#xc9c0;&#xd615;&#xc744;  ←
&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
SELECT topology.CreateTopology('topo_boston_test', 2249);
-- &#xc0c8; &#xd14c;&#xc774;&#xbel4; &#xc0dd;&#xc131;
CREATE TABLE nei_topo(gid serial primary key, nei varchar(30));
-- &#xc0c8; &#xd14c;&#xc774;&#xbel4;&#xc5d0; TopoGeometry &#xc5f4;&#xc744; &#xcd94;&#xac00;
SELECT topology.AddTopoGeometryColumn('topo_boston_test', 'public', 'nei_topo', 'topo', ' ←
MULTIPOLYGON') As new_layer_id;
new_layer_id
-----
1

-- &#xc0c8; TopoGeometry &#xc5f4;&#xc744; &#xcc44;&#xc6b0;&#xb294; &#xb370; &#xc0c8;  ←
&#xb808;&#xc774;&#xc5b4; ID&#xb97c; &#xc774;&#xc6a9;
-- &#xc0c8; &#xb808;&#xc774;&#xc5b4;&#xc5d0; &#xd5c8;&#xc6a9; &#xc624;&#xcc28;&#xac00; 0  ←
&#xc778; TopoGeometry&#xb97c; &#xcd94;&#xac00;
INSERT INTO nei_topo(nei, topo)
SELECT nei, topology.toTopoGeom(geom, 'topo_boston_test', 1)
FROM neighborhoods
WHERE gid BETWEEN 1 and 15;

-- &#xc791;&#xc5c5; &#xb0b4;&#xc6a9;&#xc744; &#xd655;&#xc778;&#xd558;&#xb824;&#xba74;  ←
&#xb2e4;&#xc74c; &#xcffc;&#xb9ac;&#xb97c; &#xc774;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;. --
SELECT * FROM
    topology.TopologySummary('topo_boston_test');

-- &#xc694;&#xc57d;&#xbb38; --
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

-- &#xbaa8;&#xb4e0; TopoGeometry &#xd3f4;&#xb9ac;&#xace4;&#xc744; 10  ←
&#xbbf8;&#xd130;&#xc529; &#xc904;&#xc785;&#xb2c8;&#xb2e4;.
UPDATE nei_topo SET topo = ST_Buffer(clearTopoGeom(topo), -10);

-- &#xc704;&#xc758; &#xc5f0;&#xc0b0;&#xc73c;&#xb85c; &#xb0a8;&#xc740;  ←
&#xc544;&#xbb34;&#xac83;&#xb3c4; &#xc5c6;&#xb294; &#xd1a0;&#xc9c0;(no-one-lands)  ←
&#xb97c; &#xc9d1;&#xacc4;&#xd569;&#xb2c8;&#xb2e4;.
-- GRASS&#xc5d0;&#xc11c;&#xb294; &#xc774;&#xac78; "polygon0 layer"&#xb77c;&#xace0;  ←
&#xbd80;&#xb97c; &#xac81;&#xb2c8;&#xb2e4;.
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
);
```


స고

CreateTopology, AddTopoGeometryColumn, CreateTopoGeom, TopologySummary, clearTopoGeom

10.9.3 TopoElementArray_Agg

TopoElementArray_Agg — Returns a `topoelementarray` for a set of `element_id`, type arrays (`topoelements`).

Synopsis

```

topoelementarray TopoElementArray_Agg(topoelement set tefield);

```

설명

TopoElement 집합으로부터 **TopoElementArray** 를 생성하데 쓰입니다.

2.0.0 버전부터 사용할 수 있습니다.

예시

```
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}}
```

స고

TopoElement, TopoElementArray

10.10 TopoGeometry

10.10.1 clearTopoGeom

clearTopoGeom — Clears the content of a topo geometry.

Synopsis

```
topogeometry clearTopoGeom(topogeometry topogeom);
```

설명

TopoGeometry 의내용을지워서텅빈 **TopoGeometry** 로바꿉니다. **toTopoGeom** 함수와결합해서&#xae30객체볰더높은계층레벨೐있는모든의존적인객체의형상을교&#데가장유용합니다.

2.1 ಄전부터 사용할 수 있습니다..


```
-- TopoGeometry tg;
--      &#xc81c;&#xac70;&#xd569;&#xb2c8;&#xb2e4;.
UPDATE mylayer SET tg = TopoGeom_remElement(tg, '{43,3}');
```

[TopoGeom_addElement](#), [CreateTopoGeom](#)

10.10.4 TopoGeom_addTopoGeom

`TopoGeom_addTopoGeom` — Adds element of a TopoGeometry to the definition of another TopoGeometry.

Synopsis

topogeometry **TopoGeom_addTopoGeom**(topogeometry tgt, topogeometry src);

Adds the elements of a [TopoGeometry](#) to the definition of another TopoGeometry, possibly changing its cached type (type attribute) to a collection, if needed to hold all elements in the source object.

The two TopoGeometry objects need be defined against the *same* topology and, if hierarchically defined, need be composed by elements of the same child layer.

Availability: 3.2

```
-- Set an "overall" TopoGeometry value to be composed by all
-- elements of specific TopoGeometry values
UPDATE mylayer SET tg_overall = TopoGeom_addTopogeom(
    TopoGeom_addTopoGeom(
        clearTopoGeom(tg_overall),
        tg_specific1
    ),
    tg_specific2
);
```

[TopoGeom_addElement](#), [clearTopoGeom](#), [CreateTopoGeom](#)

10.10.5 toTopoGeom

`toTopoGeom` — Adds a geometry shape to an existing topo geometry.

Refer to [toTopoGeom](#).

10.11 TopoGeometry

10.11.1 GetTopoGeomElementArray

`GetTopoGeomElementArray` — Returns a `topoelementarray` (an array of `topoelements`) containing the topological elements and type of the given `TopoGeometry` (primitive elements).

Synopsis

`topoelementarray GetTopoGeomElementArray`(varchar toponame, integer layer_id, integer tg_id);

`topoelementarray topoelement GetTopoGeomElementArray`(topogeometry tg);

설명

지형의 요소 및 입력 TopoGeometry 의 유형(요소)를 담고 있는 **TopoElementArray** 를 반환합이 함수는 요소들을 데이터셋이 아니라 배열로 반환한다는 점만 제외하면 GetTopoGeomElements 함수와 비슷합니 tg_id는 topology.layer 테이블에 있는 layer_id 가 지정레이어 안의 지형이 가지고 있는 TopoGeometry 객체의 TopoGeometry ID입니다.

Availability: 1.1

예시

참고

GetTopoGeomElements, TopoElementArray

10.11.2 GetTopoGeomElements

`GetTopoGeomElements` — Returns a set of `topoelement` objects containing the topological `element_id`,`element_type` of the given `TopoGeometry` (primitive elements).

Synopsis

`setof topoelement GetTopoGeomElements`(varchar toponame, integer layer_id, integer tg_id);

`setof topoelement GetTopoGeomElements`(topogeometry tg);

설명

Returns a set of `element_id`,`element_type` (topoelements) corresponding to primitive topology elements **TopoElement** (1: nodes, 2: edges, 3: faces) that a given topogeometry object in `toponame` schema is composed of.

tg_id는 topology.layer 테이블에 있는 layer_id 가 지정레이어 안의 지형이 가지고 있는 TopoGeometry 객체의 TopoGeometry ID입니다.

2.0.0 버전부터 사용할 수 있습니다.

ST_SRID

ST_SRID

[GetTopoGeomElementArray](#), [TopoElement](#), [TopoGeom_addElement](#), [TopoGeom_remElement](#)

10.11.3 ST_SRID

ST_SRID — Returns the spatial reference identifier for a topogeometry.

Synopsis

integer **ST_SRID**(topogeometry tg);

ST_SRID

Returns the spatial reference identifier for the ST_Geometry as defined in spatial_ref_sys table. [Section 4.5](#)



Note

spatial_ref_sys table is a table that catalogs all spatial reference systems known to PostGIS and is used for transformations from one spatial reference system to another. So verifying you have the right spatial reference system identifier is important if you plan to ever transform your geometries.

Availability: 3.2.0



This method implements the SQL/MM specification. SQL-MM 3: 14.1.5

ST_SRID

```
SELECT ST_SRID(ST_GeomFromText('POINT(-71.1043 42.315)', 4326));
--result
4326
```

ST_SRID

[Section 4.5](#), [ST_SetSRID](#), [ST_Transform](#), [ST_SRID](#)

10.12 TopoGeometry

10.12.1 AsGML

AsGML — TopoGeometry to GML


```

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>
>

```

이전 예시와 동일하지만 네임스쓰지 않습니다.

```

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>
>

```

󌰸고

CreateTopoGeom, ST_CreateTopoGeo

10.12.2 AsTopoJSON

AsTopoJSON — TopoGeometry

Synopsis

text AsTopoJSON(topogeometry tg, regclass edgeMapTable);

설명

TopoGeometry; TopoJSON 표현식을 반환합니다. edgeMapTable 이 NULL이 아닐 경우 경계선 식볐호(arc) 인덱스에 매핑하는 탐색/&#xc80쓰일 것입니다. 최종 문서에 조밀&#원호" 배열을 사용할 수 있게 하기 위해서입니다. 테이블을 설정할 경우 테이블이 "순차(serial)" 유형 "arc_id" 항목과 정수형 "edge_id" 항목을 가지고 있어야 합니다. 코&#해당 항목에 인덱스를 추가하는 편이 좋습니다.

**Note**

TopoJSON `완전한 TopoJSON ସ서는, 이 함수가 반환토막 정보(snippet)에 더해, 일부 헤더와 함께 실제 원호를 담고 있어야 할 것입니다. TopoJSON 사양서 를 참조하십원호 "edgeMapTable" 테이블에서는 1-기반입니다.`

`완전한 TopoJSON ସ서는, 이 함수가 반환토막 정보(snippet)에 더해, 일부 헤더와 함께 실제 원호를 담고 있어야 할 것입니다. TopoJSON 사양서 를 참조하십원호 "edgeMapTable" 테이블에서는 1-기반입니다.`

`2.1.0 버전부터 사용할 수 있습니다.`

`개선 사항 2.2.1 버전부터 점형(puntal) 입렬지원하기 시작했습니다.`

참고

ST_AsGeoJSON

예시

```
CREATE TEMP TABLE edgemap(arc_id serial, edge_id int unique);

-- &#xd5e4;&#xb354;
SELECT '{ "type": "Topology", "transform": { "scale": [1,1], "translate": [0,0] }, "objects ←
": { '

-- &#xac1d;&#xccb4;
UNION ALL SELECT '' || feature_name || '": ' || AsTopoJSON(feature, 'edgemap')
FROM features.big_parcel_s WHERE feature_name = 'P3P4';

-- &#xc6d0;&#xd638;
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
), arcdump 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 arcdump
  GROUP BY arc_id
  ORDER BY arc_id
)
SELECT '}', "arcs": [' UNION ALL
SELECT array_to_string(array_agg(a), E',\n') from arcs

-- &#xaf2c;&#xb9ac;&#xb9d0;
UNION ALL SELECT ']}'::text as t;
```

```
--
{ "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]]
]]
}
```

10.13 ST_Equals

10.13.1 Equals

Equals — Returns true if two TopoGeometry objects are equal. The objects must be of the same type (Point, LineString, Polygon, etc.) and have the same geometry.

Synopsis

boolean **Equals**(topogeometry tg1, topogeometry tg2);

Examples

Example 1: Returns true if two TopoGeometry objects are equal. The objects must be of the same type (Point, LineString, Polygon, etc.) and have the same geometry.



Note

The ST_Equals function is only available in PostGIS 3.3.0 and later. It is not available in earlier versions of PostGIS. The function is not available in the PostGIS 2.x series.

1.1.0 Returns true if two TopoGeometry objects are equal. The objects must be of the same type (Point, LineString, Polygon, etc.) and have the same geometry.



This function supports 3d and will not drop the z-index.

Examples

Example 1: Returns true if two TopoGeometry objects are equal. The objects must be of the same type (Point, LineString, Polygon, etc.) and have the same geometry.

Example 2: Returns true if two TopoGeometry objects are equal. The objects must be of the same type (Point, LineString, Polygon, etc.) and have the same geometry.

10.13.2 $\frac{1}{2} \pi$

교차 — 두 TopoGeometry에서 나온 원시형의
쌍 가운데 하나라도 교차할 경우
참을 반환합니다.

Synopsis

```
boolean Intersects(topogeometry tg1, topogeometry tg2);
```

설명

ѐ TopoGeometry에서 나온 원시형의 쌍 가욶하나라도 교차할 경우 참을 반환을

Note



이 함수는 도형 집합인 TopoGeometry를
지원하지 않습니다. 또 서로
다른 지형으로 이루어진 두
TopoGeometry를 비교하지 못 합니다. 또한 현재 계층 TopoGeometry(다른 TopoGeome-
try들로 구성된 TopoGeometry)도 지원하지
않습니다.

1.1.0 ಄전부터 사용할 수 있습니다.



This function supports 3d and will not drop the z-index.

예시

స고

ST Intersects

10.14 Importing and exporting Topologies

Once you have created topologies, and maybe associated topological layers, you might want to export them into a file-based format for backup or transfer into another database.

Using the standard dump/restore tools of PostgreSQL is problematic because topologies are composed by a set of tables (4 for primitives, an arbitrary number for layers) and records in metadata tables (topology.topology and topology.layer). Additionally, topology identifiers are not univoque across databases so that parameter of your topology will need to be changes upon restoring it.

In order to simplify export/restore of topologies a pair of executables are provided: `pgtopo_export` and `pgtopo_import`. Example usage:

```
pgtopo_export dev_db topol | pgtopo_import topol | psql staging_db
```

10.14.1 Using the Topology exporter

The `pgtopo_export` script takes the name of a database and a topology and outputs a dump file which can be used to import the topology (and associated layers) into a new database.

By default `pgtopo_export` writes the dump file to the standard output so that it can be piped to `pgtopo_import` or redirected to a file (refusing to write to terminal). You can optionally specify an output filename with the `-f` cmdline switch.

By default `pgtopo_export` includes a dump of all layers defined against the given topology. This may be more data than you need, or may be non-working (in case your layer tables have complex dependencies) in which case you can request skipping the layers with the `--skip-layers` switch and deal with those separately.

Invoking `pgtopo_export` with the `--help` (or `-h` for short) switch will always print short usage string.

The dump file format is a compressed tar archive of a `pgtopo_export` directory containing at least a `pgtopo_dump_version` file with format version info. As of version 1 the directory contains tab-delimited CSV files with data of the topology primitive tables (node, edge_data, face, relation), the topology and layer records associated with it and (unless `--skip-layers` is given) a custom-format PostgreSQL dump of tables reported as being layers of the given topology.

10.14.2 Using the Topology importer

The `pgtopo_import` script takes a `pgtopo_export` format topology dump and a name to give to the topology to be created and outputs an SQL script reconstructing the topology and associated layers.

The generated SQL file will contain statements that create a topology with the given name, load primitive data in it, restores and registers all topology layers by properly linking all TopoGeometry values to their correct topology.

By default `pgtopo_import` reads the dump from the standard input so that it can be used in conjunction with `pgtopo_export` in a pipeline. You can optionally specify an input filename with the `-f` cmdline switch.

By default `pgtopo_import` includes in the output SQL file the code to restore all layers found in the dump.

This may be unwanted or non-working in case your target database already have tables with the same name as the ones in the dump. In that case you can request skipping the layers with the `--skip-layers` switch and deal with those separately (or later).

SQL to only load and link layers to a named topology can be generated using the `--only-layers` switch. This can be useful to load layers AFTER resolving the naming conflicts or to link layers to a different topology (say a spatially-simplified version of the starting topology).

Chapter 11

11. Raster Data Types

11.1 Raster Data Types

The raster data types in PostGIS are defined in the `raster2pgsql` module. The raster data types are defined in the `raster2pgsql` module. The raster data types are defined in the `raster2pgsql` module.

11.1.1 raster2pgsql

The `raster2pgsql` module is used to create raster data in PostGIS. It is a command-line utility that takes a raster file as input and creates a raster table in a PostGIS database. The raster data types are defined in the `raster2pgsql` module.

Since the `raster2pgsql` is compiled as part of PostGIS most often (unless you compile your own GDAL library), the raster types supported by the executable will be the same as those compiled in the GDAL dependency library. To get a list of raster types your particular `raster2pgsql` supports use the `-G` switch. These should be the same as those provided by your PostGIS install documented here [ST_GDALDrivers](#) if you are using the same GDAL library for both.

Note

이 도구의 구 버전은 파이썬
 스크립트였습니다. 현재는
 실행파일이 파이썬 스크립트를
 대체했습니다. 사용자가 해당
 파이썬 스크립트를 필요로
 할 경우, GDAL PostGIS 래스터 드라이룄
 활용 에서 파이썬 스크립트의
 예들을 찾을 수 있습니다. raster2pgsql
 파이썬 스크립트가 PostGIS 향후
 룄전과 호환되지 않을 수
 있으며, 더 이상 지원되지도
 않는다는 점을 주의해주시기
 룄랍니다.

Note

정렬된 래스터 집합에서 특정
 인자(factor)의 오룄뷰를 성생하는
 경우, 오룄뷰들이 정렬되지
 않을 수도 있습니다. <http://trac.osgeo.org/postgis/ticket/1764>
 페이지에서 오룄뷰가
 정렬되지 않는 예시를 찾아본
 수 있습니다.

활용예;

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

-? 도움말 화면을 표출합니다. 어&#xb5a4
 인수도 쓰지 않을 경우에도 도움
 표출될 것입니다.

-G 지원하는 래스터 형식을 나열합
 claldlp -- 이들은 상호룄타적인 옵션룄

-c 새 테이&#xbel4;을 생성한 다음 래스해
 해당 테이&#xbel4;을 채웁니다. 이것기
 본 모드 입니다.

-a 기존 테이&#xbel4;에 래스터(들)을 추룄

-d 기존 테이&#xbel4;을 삭제하고, 새 테생
 성한 다음 래스터(들)로 해당
 테이&#xbel4;을 채웁니다.

-p 준들 모드로, 테이&#xbel4;만 생성합래

래스터 공간 처리: 래스터 을탈룄

-C raster_columns 뷰에 래스터를 제대로 등위
 한 SRID, 픽셀 크기 등의 래스터
 제약조건을 적용합니다.

-x 최대 룄위(extent) 제약조건을 해제플
 -C 플래그와 함께 쓰일 경우에만
 적용됩니다.


```
-n COLUMN &#xd30a5;&#xc77c;&#xc35f4;&#xc758; &#xba85;&#xc6d;&#xc744; &#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;,-F &#xd50c;&#xb798;&#xadf8;&#xc640; &#xd568;&#xaed8; &#xc368;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;,-q PostgreSQL &#xc2dd;&#xbcc4;&#xc790;&#xb97c; &#xb530;&#xc634;&#xd45c;&#xb85c; &#xac10;&#xc309;&#xb2c8;&#xb2e4;,-I &#xb798;&#xc2a4;&#xd130; &#xc5f4;&#xc5d0; GiST &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;,-M &#xb798;&#xc2a4;&#xd130; &#xd14c;&#xc774;&#xbe14;&#xc744; &#xbe48;&#xacf5;&#xac04; &#xbd84;&#xc11d;,(value analyze)&#xd569;&#xb2c8;&#xb2e4;,-k Keeps empty tiles and skips NODATA value checks for each raster band. Note you save time in checking, but could end up with far more junk rows in your database and those junk rows are not marked as empty tiles.-T tablespace &#xc0c8; &#xd14c;&#xc774;&#xbe14; &#xc6a9; &#xd14c;&#xc774;&#xbe14;&#xc2a4;&#xd398;&#xc778;&#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;,-X &#xd50c;&#xb798;&#xadf8;&#xb97c; &#xd568;&#xaed8; &#xc0ac;&#xc6a9;&#xd558;&#xc9c0; &#xc54a;&#xc73c;&#xba74; (&#xae30;&#xbcf8; &#xd0a4;&#xb97c; &#xd3ec;&#xc778;&#xb371;&#xc2a4;&#xb4e4;&#xc774; &#xc5ec;&#xc804;&#xd788; &#xae30;&#xbcf8; &#xd14c;&#xc774;&#xc0ac;&#xc6a9;&#xd560; &#xac83;&#xc774;&#xb77c;&#xb294; &#xc810;&#xc5d0; &#xc8fc;&#xc758;&#xd558;&#xb2e4;,-X tablespace &#xd14c;&#xc774;&#xbe14;&#xc758; &#xc0c8; &#xc778;&#xb371;&#xc2a4; &#xc6a9; &#xd14c;&#xc778;&#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;,-I &#xd50c;&#xb798;&#xadf8;&#xb97c; &#xd568;&#xaed8; &#xc0ac;&#xc6a9;&#xd558;&#xba74; &#xae30;&#xbcf8; &#xd0a4;&#xc640; &#xacf5;&#xac04; &#xc778;&#xb371;&#xd568;&#xaed8; &#xc801;&#xc6a9;&#xb429;&#xb2c8;&#xb2e4;,-Y max_rows_per_copy=50 Use copy statements instead of insert statements. Optionally specify max_rows_per_copy; default 50 when not specified.-e &#xac01; &#xc120;&#xc5b8;&#xbb38;&#xc744; &#xac1c;&#xbcc4;&#xc801;&#xc73c;&#xb85c; &#xc2e4;&#xd589;&#xd558;&#xc0c1;&#xd638;&#xcc98;&#xb9ac;,(transaction)&#xb97c; &#xc774;&#xc6a9;&#xd558;&#xc9c0; &#xc54a;&#xc2b5;&#xb-E ENDIAN &#xc0dd;&#xc131;&#xb41c; &#xb798;&#xc2a4;&#xd130; &#xbc14;&#xc774;&#xb108;&#xb9ac; &#xc0b0;&#xcd9c;&#xba54;&#xbaa8;&#xb9ac; &#xb0b4;&#xbd80; &#xb370;&#xc774;&#xd130; &#xc21c;&#xc11c;,(endianness)&#xb97c; &#xc870;&#xc815;&#xd569;&#xb2c8;&#xb2e4;,-XDR&#xc740; 0, &#xae30;&#xbcf8;&#xac12;&#xc778; NDR&#xc740; 1&#xb85c; &#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;,. &#xd604;&#xc7ac;,, NDR &#xc0b0;&#xcd9c;&#xbb3c;&#xc9c0;&#xc6d0;&#xd569;&#xb2c8;&#xb2e4;,-V version &#xc0b0;&#xcd9c;&#xbb3c; &#xd615;&#xc2dd;&#xc758; &#xbc84;&#xc804;&#xc744; &#xc124;&#xc815;&#xd569;&#xae30;&#xbcf8;&#xac12;&#xc740; 0&#xc785;&#xb2c8;&#xb2e4;,. &#xd604;&#xc7ac;,, 0&#xb9cc; &#xc9c0;&#xc6d0;&#xb85c;&#xb354;&#xb97c; &#xd1b5;&#xd574; &#xc785;&#xb825; &#xd30c;&#xc77c;&#xc744; &#xc0dd;&#xc131;&#xd558;&#xd569; 100x100&#xd0c0;&#xc77c; &#xbb49;&#xce58;&#xb85c; &#xb098;&#xb204;&#xc5b4; &#xc5c5;&#xb85c; &#xb4dc;&#xd558;&#xc138;&#xc158;&#xc758; &#xc608;&#xc2dc;&#xb294; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc2b5;&#xb2c8;&#xb2e4;:
```

Note

```
&#xc608;&#xb97c;      &#xb4e4;&#xc5b4;      public.demelevation      &#xb300;&#xc2e0;
demelevation      &#xc744;      &#xc4f0;&#xb294;      &#xac83;&#xcc98;&#xb7fc;      &#xc2a4;&#xd0a4;&#xb9c8;
&#xba85;&#xc744;      &#xc0dd;&#xb7b5;&#xd560;      &#xc218;      &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;
&#xadf8;&#xb798;&#xb3c4;      &#xb798;&#xc2a4;&#xd130;      &#xd14c;&#xc774;&#xbe14;&#xc740;
&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;      &#xb610;&#xb294;      &#xc0ac;&#xc6a9;&#xc790;&#xc758;
&#xae30;&#xbcf8;      &#xc2a4;&#xd0a4;&#xb9c8;&#xb97c;      &#xae30;&#xbcb8;&#xc73c;&#xb85c;
&#xc0dd;&#xc131;&#xb420;      &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

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

UNIX 파이프를 이용하면 변환 및 업&#xb8
작업을 한 번에 끝낼 수 있습니다.

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



```

&#xb9e4;&#xc0ac;&#xcd94;&#xc138;&#xce20;&#xc8fc;&#xc758;&#xbbf8;&#xd130;&#xb2e8;&#xc704;&#xd56d;&#xacf5;&#x
&#xd0c0;&#xc77c;&#xb798;&#xc2a4;&#xd130;&#xb4e4;&#xc744;aerial&#xc774;&#xb77c;&#xb294;&#xc2a4;&#xd0a4;&#x
&#xb85c;&#xb4dc;&#xd558;&#xace0;&#xc804;&#xccb4;&#xbdf0;&#xc640;2&#xb808;&#xbca8;&#xbc0f;4&#xb808;&#xbca8;
&#xc624;&#xbc84;&#xbdf0;&#xd14c;&#xc774;&#xbe14;&#xc744;&#xc0dd;&#xc131;&#xd55c;&#xb2e4;&#xc74c;&#xbcf5;&#x
&#xbaa8;&#xb4dc;&#xb97c;&#xd1b5;&#xd574;(&#xc911;&#xac04;&#xb2e8;&#xacc4;&#xd30c;&#xc77c;&#xc5c6;&#xc774;
DB&#xb85c;&#xc9c1;&#xc811;)&#xc0bd;&#xc785;&#xd558;&#xba70;&#xac15;&#xc81c;&#xb85c;&#xbaa8;&#xb4e0;
&#xc791;&#xc5c5;&#xc744;&#xc0c1;&#xd638;&#xcc98;&#xb9ac;&#xd558;&#xc9c0;&#xc54a;&#xb3c4;&#xb85d;-e&#xd50c;&#x
&#xc0ac;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;(&#xc791;&#xc5c5;&#xc774;&#xc644;&#xb8cc;&#xb418;&#xae38;
&#xae30;&#xb2e4;&#xb9ac;&#xc9c0;&#xc54a;&#xace0;&#xd14c;&#xc774;&#xbe14;&#xc5d0;&#xb4e4;&#xc5b4;&#xc624;&#x
&#xb370;&#xc774;&#xd130;&#xb97c;&#xbc14;&#xb85c;&#xc0b4;&#xd3b4;&#xbcf4;&#xace0;&#xc790;&#xd560;&#xb54c;
&#xc720;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;).&#xb798;&#xc2a4;&#xd130;&#xb97c;128x128&#xd53d;&#xc140;&#xd0c0;&#x
&#xbd84;&#xc808;&#xd55c;&#xb2e4;&#xc74c;&#xb798;&#xc2a4;&#xd130;&#xc81c;&#xc57d;&#xc870;&#xac74;&#xc744;
&#xc801;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.&#xd14c;&#xc774;&#xbe14;&#xc0bd;&#xc785;&#xb300;&#xc2e0;
&#xbcf5;&#xc0ac;&#xbaa8;&#xb4dc;&#xb97c;&#xc774;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.&#xd0c0;&#xc77c;&#x
&#xc798;&#xb77c;&#xc838;&#xb098;&#xc628;&#xd0c0;&#xc77c;&#xd30c;&#xc77c;&#xba85;&#xc744;&#xb2f4;&#xc744;
&#xc218;&#xc788;&#xb3c4;&#xb85d;-F&#xd50c;&#xb798;&#xadf8;&#xb85c;"filename"&#xc774;&#xb77c;&#xb294;&#xd544;
&#xd3ec;&#xd568;&#xc2dc;&#xd0a4;&#xc2ed;&#xc2dc;&#xc624;.

```

```

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

```

```

-- &#xc9c0;&#xc6d0;&#xb418;&#xb294;&#xb798;&#xc2a4;&#xd130;&#xc720;&#xd615;&#xc758; ↵
    &#xbaa9;&#xb85d;&#xc744;&#xc5bb;&#xc73c;&#xb824;&#xba74;:
raster2pgsql -G

```

```

-G&#xc635;&#xc158;&#xc774;&#xb2e4;&#xc74c;&#xacfc;&#xac19;&#xc740;&#xbaa9;&#xb85d;&#xc744;&#xcd9c;&#xb825;&#x
&#xac83;&#xc785;&#xb2c8;&#xb2e4;;

```

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 (.sdat)
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

11.1.2 PostGIS

USGS LULC Composite Theme Grid
 Arc/Info Export E00 GRID
 ZMap Plus Grid
 NOAA NGS Geoid Height Grids

1. Create a table named `myrast` with a primary key `rid` and a column `rast` of type `raster`.

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

2. Create a table named `myrast` with a primary key `rid` and a column `rast` of type `raster`. Then, create a table named `myrast_st` with a primary key `rid` and a column `rast_st` of type `raster`. Finally, create a table named `myrast_convexhull` with a primary key `rid` and a column `rast_convexhull` of type `raster`.
3. Create a table named `myrast` with a primary key `rid` and a column `rast` of type `raster`. Then, create a table named `myrast_st` with a primary key `rid` and a column `rast_st` of type `raster`. Finally, create a table named `myrast_convexhull` with a primary key `rid` and a column `rast_convexhull` of type `raster`.

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

USGS LULC Composite Theme Grid
 Arc/Info Export E00 GRID
 ZMap Plus Grid
 NOAA NGS Geoid Height Grids

Note

PostGIS 2.0 raster2pgsql uses GDAL to access raster data, and can take advantage of a key GDAL feature: the ability to read from rasters that are **stored remotely** in cloud "object stores" (e.g. AWS S3, Google Cloud Storage).

Efficient use of cloud stored rasters requires the use of a "cloud optimized" format. The most well-known and widely used is the **"cloud optimized GeoTIFF"** format. Using a non-cloud format, like a JPEG, or an un-tiled TIFF will result in very poor performance, as the system will have to download the entire raster each time it needs to access a subset.

First, load your raster into the cloud storage of your choice. Once it is loaded, you will have a URI to access it with, either an "http" URI, or sometimes a URI specific to the service. (e.g., "s3://bucket/object"). To access non-public buckets, you will need to supply GDAL config options to authenticate your connection. Note that this command is *reading* from the cloud raster and *writing* to the database.

4. **AddRasterConstraints** raster2pgsql -s 990000 -t 256x256 -I -R /vsis3/your.bucket.com/your_file.tif your_table | psql your_db

11.1.3 Using "out db" cloud rasters

The raster2pgsql tool uses GDAL to access raster data, and can take advantage of a key GDAL feature: the ability to read from rasters that are **stored remotely** in cloud "object stores" (e.g. AWS S3, Google Cloud Storage).

Efficient use of cloud stored rasters requires the use of a "cloud optimized" format. The most well-known and widely used is the **"cloud optimized GeoTIFF"** format. Using a non-cloud format, like a JPEG, or an un-tiled TIFF will result in very poor performance, as the system will have to download the entire raster each time it needs to access a subset.

First, load your raster into the cloud storage of your choice. Once it is loaded, you will have a URI to access it with, either an "http" URI, or sometimes a URI specific to the service. (e.g., "s3://bucket/object"). To access non-public buckets, you will need to supply GDAL config options to authenticate your connection. Note that this command is *reading* from the cloud raster and *writing* to the database.

```
AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxxxxxxx \
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx \
raster2pgsql \
-s 990000 \
-t 256x256 \
-I \
-R \
/vsis3/your.bucket.com/your_file.tif \
your_table \
| psql your_db
```

Once the table is loaded, you need to give the database permission to read from remote rasters, by setting two permissions, **postgis.enable_outdb_rasters** and **postgis.gdal_enabled_drivers**.

```
SET postgis.enable_outdb_rasters = true;
SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

To make the changes sticky, set them directly on your database. You will need to re-connect to experience the new settings.

```
ALTER DATABASE your_db SET postgis.enable_outdb_rasters = true;
ALTER DATABASE your_db SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

For non-public rasters, you may have to provide access keys to read from the cloud rasters. The same keys you used to write the raster2pgsql call can be set for use inside the database, with the **postgis.gdal_vsi_options** configuration. Note that multiple options can be set by space-separating the key=value pairs.

```
SET postgis.gdal_vsi_options = 'AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
```

Once you have the data loaded and permissions set you can interact with the raster table like any other raster table, using the same functions. The database will handle all the mechanics of connecting to the cloud data when it needs to read pixel data.

- `srid` 래스터의 공간 참조 식별자입래 Section 4.5 목록에 있는 항목이어야 합니
- `scale_x` 기하하적 공간 좌표와 픽셀의 비례(축척)입니다. 래스터 열에 있 모든 타일이 동일한 `scale_y` 를 가지며 `scale_y` 제약조이 적용될 경우에만 사용할 수 있습니다. 자세한 내용 **ST_ScaleX** 를 참조하십시오.
- `scale_y` 기하하적 공간 좌표와 픽셀의 비례(축척)입니다. 래스터 열에 있 모든 타일이 동일한 `scale_y` 를 가지며 `scale_y` 제약조이 적용될 경우에만 사용할 수 있습니다. 자세한 내용 **ST_ScaleY** 를 참조하십시오.
- `blocksize_x` 각 래스터 타일의 너비(가로 방향의 픽셀 개수)입니다. 자세한 사항은 **ST_Width** 를 참조하십시오.
- `blocksize_y` 각 래스터 타일의 높이(세로 방향의 픽셀 개수)입니다. 자세한 사항은 **ST_Height** 를 참조하십시오.
- `same_alignment` 모든 래스터 타일이 동일한 방향으로 정렬된 경우 참인 불 값 자세한 내용은 **ST_SameAlignment** 를 참조하십시오.
- `regular_blocking` 래스터 열이 공간적 유일세 제약조이 및 커버리지 타일 제약이 가질 경우 해당 값은 참입니다. 그 외의 경우는 거짓이 됩니다.
- `num_bands` 사용자 래스터 집합의 각 타이 내장한 밴드의 개수입니다. 다음 열과 동일한 정보를 제공합니다 **ST_NumBands**
- `pixel_types` 각 밴드에 대한 픽셀 유형을 정의하는 배열입니다. 이 배열에 사용자 밴드 개수와 동일한 개수 요소들이 들어󊰈 󊲃입니다. `pixel_types` 는 다음 **ST_BandPixelType** 에서 정의된 픽셀 유형 󊰀운데 하나입니다.
- `nodata_values` 각 밴드에 대한 `nodata_value` 를 의미대정밀도(double precision) 수(數)입니다. 이 배사용자 밴드 개수와 동일한 개수 요소들이 들어󊰈 󊲃입니다. 이 숫 대부분의 연산에서 &#bbb34;시되어야 할 각 밴드에 대한 픽셀 값을 정의 이 정보는 **ST_BandNoDataValue** 가 제공하는 정보 유사합니다.
- `out_db` 데이터베이스 외부에서 래스데이터를 유지하는지 여부 보여주는 불 값 플래그의 배열입 이 배열에는 사용자 밴드 개수와 동일한 개수의 요소들이 들어󊰈 󊲃입니다.

- `extent` 사용자 래스터 집합이 포함하모든 래스터 열의 범위(extent)입니다.해당 집합의 범위를 변경시킬 추데이터를 로드할 계획이라면,로작업 전에 **DropRasterConstraints** 함수를 실행하곮작업 후에 **AddRasterConstraints** 함수로 제약조건다시 적용시키는 편이 좋습니다.
- `spatial_index` 래스터 열이 공간 인덱스를 가진 경우 참인 불 값입니다.

11.2.2 $\mathbb{R}^n$ and $\mathbb{C}^n$

raster_overviews 는 오 버 봏 에 이 용 되 는 래 &#xc2a4
테 이 블 열 에 관 한 정 보 볰 오 버 봏
활 용 시 알 면 유 용 한 추 가 정 보 를
카 탈 로 그 화 합 니 다. 오 버 봏 테 이 &#x
raster_columns 볰 raster_overviews 양 쪽 에 모 두 카 탈 &#xb85
오 버 봏 가 나 름 대 로 래 스 터 이 긴
하 지 만 동 시 에 고 해 상 도 테 이 블 &#x
저 해 상 도 썸 네 일 이 라 는 추 가 적 &#x
특 별 목 적 을 담 당 하 고 있 기 때 문 &#x
래 스 터 로 드 작 업 중 -1 스 위 치 를 사
주 래 스 터 테 이 블 과 함 ૨ 오 버 봏
테 이 블 을 생 성 합 니 다. 또 는 **AddOverview-**
Constraints 함 수 를 통 해 직 접 생 성 할 수 도
있 습 니 다.

오버򽿰테이블은 다른 래스터 테&#
동일한 제약조건은 물론 오직 오&#
특화된 제약조건에 관한 추가 정&#
포함합니다.

Note



Note!

```
raster_overviews &#xc5d0; &#xb2f4;&#xae34; &#xc815;&#xbcf4;&#xb294; raster_columns &#xc758;
&#xc815;&#xbcf4;&#xb97c; &#xbcf5;&#xc81c;&#xd55c; &#xac83;&#xc774; &#xc544;&#xb2d9;&#xb2c8;&#xb2e4;.
raster_columns &#xc5d0;&#xc11c; &#xd604;&#xc7ac; &#xc624;&#xbcb4;&#xbdf0;
&#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xad00;&#xd55c; &#xc815;&#xbcf4;&#xb97c; &#xc5bb;&#xc5b4;&#xc57c;
&#xd560; &#xacbd;&#xc6b0; raster_overviews &#xc640; raster_columns &#xc744; &#xd568;&#xaed8;
&#xc5f0;&#xacb0;(join)&#xc2dc;&#xcf1c; &#xd544;&#xc694;&#xd55c; &#xc644;&#xc804;&#xd55c;
&#xc815;&#xbcf4;&#xb97c; &#xc5bb;&#xc73c;&#xc2ed;&#xc2dc;&#xc624;.
```

오버򽿰의 주된 존재 이유는 다음 두 가지입니다:

1. ୠ른 매핑 확대를 위해 흔히 핵욠
테이블의 저해상도 모사를 이욠
2. 고해상도 페어런트보다 오버򽿰
기반으로 하는 계산이 일반적으
더 빠릅니다. 레코드 개수도 더
적고 각 픽셀이 커벸하는 면적도
더 크기 때문입니다. 오버򽿰가
지원하는 고해상도 테이블 대산
계산보다야 정확하지 않지만, 어림잡이(rule-of-thumb) 계산의 경우 오버충분할 수 있습니다.

raster_overviews
사용자 지침서

- o_table_catalog 오버뷰 테이블을 담고 있는데이터베이스입니다. 이 열은 언현재 데이터베이스를 읽어들일 것입니다.
- o_table_schema 오버뷰 테이블이 속해 있는데이터베이스 스키마입니다.
- o_table_name 오버뷰 테이블 명입니다.
- o_raster_column 오버뷰 테이블이 담고 있는래스터 열입니다.
- r_table_catalog 해당 오버뷰 서비스를 담고 있는 래스터 테이블 카탈로그입이 열은 언제나 현재 데이터베이읽어들일 것입니다.
- r_table_schema 해당 오버뷰 서비스가 속해 있는 래스터 테이블 스키마입니다.
- r_table_name 해당 오버뷰가 서비스하는 래스터 테이블 명입니다.
- r_raster_column 해당 오버뷰가 서비스하는 래스터 열입니다.
- overview_factor 오버뷰 테이블의 피라미드레벨입니다. 숫자가 커질수록 테해상도는 낮아집니다. raster2pgsql은 이미폴더를 򼬛아 각 이미지 파일의 오ૄ산하고 개별적으로 로드합니항상 레벨 1을 원 파일로 가정합니레벨 2는 각 타일이 원래 이미지 4장을 모사하게 됩니다. 따라서 예를 들어 사용자가 125x125로 뭉치려&#x5000x5000픽셀 이미지 파일들이 있는 폴 경우., 각 이미지 파일에 대해 사용기저 테이블이 (5000*5000)/(125*125) = 1600 레코드를 담게 되고., 사용자의 o_2 (l=2) 테이블이 최대 (1600/2^2) = 400행., o_3 (l=3) 테이블은 최대 (1600/2^3) = 200행의 레코드를 담게 될 것입니나눌 수 있는 숫자가 아닐 경우., 몇몇 파편(scrap) 타일(완전히 채워지않은 타일)을 얻게 될 것입니다. raster2pgsql이 생성한 각 오버뷰 타일의 픽셀 개수가 해당 페어런트와 동개수이지만., 각 픽셀이 표현하는 영역이 더 넓은 (원 이미지의 2^overview_factor 픽셀에 해당하는) 저해상도라는 점에 주의하십시오.

11.3 PostgreSQL ʤ터를 이용하&#xb29 사용자 지정 응용 프&#xb8 빌드하기

The fact that PostGIS raster provides you with SQL functions to render rasters in known image formats gives you a lot of options for rendering them. For example you can use OpenOffice / LibreOffice for rendering as demonstrated in [Rendering PostGIS Raster graphics with LibreOffice Base Reports](#). In addition you can use a wide variety of languages as demonstrated in this section.

11.3.1 다른래스터함수와함께
ST_AsPNG를이용해서 PHP 예제를
출력하기

이단원에서, PHP PostgreSQL 드라이버와 **ST_AsGDAL**
계열함수를사용해서 래스터의
1, 2, 3 밴드를 PHP 요청 스트림(request stream)으로
출력하는 방법을 설명할 것입니&#
이후 PHP 요청 스트림을 "img src" HTML 태그로
임베딩할 수 있습니다.

예시 쿼리는 여러 래스터 함수를
함께 결합(combine)해서 특정 WGS84 경계 상&#xc
교차하는 모든 타일을 얻은 다음,
ST_Union 함수를 통해 교차하는 타일들&#xc
반환되는 모든 밴드를 통합(union)해&#xc
ST_Transform 함수를 이용해 사용자가 지&#xc815
투영으로 변환한 후, **ST_AsPNG** 함수로 결
PNG 파일로 출력하는 방법을 보여줌

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

를 통 해 메 사 추 세 츠 주 의 피 트 단 &#
항 공 사 진 의 래 스 터 이 미 지 를 얻 &#x
후 . 다 음 쿼 리 를 호 출 하 십 시 오 .

```
<?php
/** test_raster.php&#xc758; &#xb0b4;&#xc6a9; */
$conn_str = 'dbname=mydb host=localhost port=5432 user=myuser password=mypwd';
$dbconn = pg_connect($conn_str);
header('Content-Type: image/png');
/** &#xd2b9;&#xc815; &#xd22c;&#xc601;&#xccb4;&#xb97c; &#xc694;&#xad6c;&#xd560;  ←
    &#xacbd;&#xc6b0; &#xc774;&#xac78; &#xc0ac;&#xc6a9;&#xd558;&#xace0;,  ←
    &#xadf8;&#xb807;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0;  ←
    &#xb9e4;&#xc0ac;&#xcd94; &#xc138;&#xce20; &#xc8fc; &#xbbf8;&#xd130; &#xb2e8;&#xc704;  ←
    &#xd3c9;&#xba74;&#xc744; &#xc0ac;&#xc6a9;&#xd560; &#xac83; */
if (!empty( $_REQUEST['srid'] ) && is_numeric( $_REQUEST['srid'] ) ){
    $input_srid = intval($_REQUEST['srid']);
}
else { $input_srid = 26986; }
/** PostgreSQL 9.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xc758; &#xacbd;&#xc6b0; bytea_output  ←
    &#xc9d1;&#xd569;&#xc774; &#xd544;&#xc694;&#xd560; &#xc218;&#xb3c4;  ←
    &#xc788;&#xc9c0;&#xb9cc;, 8.4 &#xbc84;&#xc804;&#xc740; &#xd544;&#xc694;&#xc5c6;&#xc74c;  ←
    */
$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]);
?>

```

11.3.2 ST_AsPNG, ST_AsGDALRaster, ST_Union, ST_Transform, ASP.NET C#

ST_AsGDALRaster, ST_Union, ST_Transform, ASP.NET C#

ASP.NET bin

ST_Union, ST_Transform, ASP.NET C#

Section 11.3.1

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

ASP.NET C#

```

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

```

```

// 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;
    }
}

```

11.3.3 래스터 쿼리를 이미지 파용지정된 파일로 출력하는 쿼리를 이용하단순한 Java 콘솔 응용 프로그램입니이미지 한 장을 반환해서 지정된 파일로 출력하는 쿼리를 이용하단순한 Java 콘솔 응용 프로그램입니다운로드할 수 있습니다.

이미지 한 장을 반환해서 지정된 파일로 출력하는 쿼리를 이용하단순한 Java 콘솔 응용 프로그램입니다운로드할 수 있습니다.

다음과 같은 명령어를 통해 예시 코드를 컴파일할 수 있습니다.

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

다음과 같은 명령행을 이용해서 컴파일한 코드를 호출하십시오.

```
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
```

```
// SaveQueryImage.java &#xc6a9; &#xcf54;&#xb4dc;
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();
        /** &#xc0ac;&#xc6a9;&#xc790;&#xac00; &#xc694;&#xccad;&#xd55c; ←
            &#xd30c;&#xc77c;&#xba85;&#xc73c;&#xb85c; &#xcd9c;&#xb825; **/
        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);
}
}
}

```

11.3.4 PLPython을 이용해서 SQL을 통해이미지를 덤프하기

```
&#xc11c;&#xbc84;&#xb514;&#xb809;&#xd130;&#xb9ac;&#xc5d0;&#xac01;&#xb808;&#xcf54;&#xb4dc;&#xb97c;&#xd30c;&#xc0dd;&#xc131;&#xd558;&#xb294;PLPython&#xc800;&#xc7a5;&#xd568;&#xc218;&#xc785;&#xb2c8;&#xb2e4;.PLPython&#xc124;&#xce58;&#xd574;&#xc57c;&#xd569;&#xb2c8;&#xb2e4;.PLPythonu&#xc640;PLPythonu3u&#xbaa8;&#xb450;&#xc798;&#xc791;&#xb3d9;&#xd560;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
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;
```

```
-- PostgreSQL &#xc11c;&#xabc84;&#xc5d0; &#xc5ec;&#xb7ec; &#xd06c;&#xae30;&#xb85c;  ←
&#xc774;&#xbbbf8;&#xc9c0; 5&#xc7a5;&#xc744; &#xc791;&#xc131;&#xd569;&#xb2c8;&#xb2e4; .
-- PostgreSQL &#xb370;&#xbaac;(daemon) &xacc4;&#xc815;&#xc774; &#xd3f4;&#xb354;&#xc758;  ←
&#xc4f0;&#xae30; &#xad8c;&#xd55c;&#xc744; &#xac00;&#xc838;&#xc57c;  ←
&#xd55c;&#xb2e4;&#xb294; &#xc810;&#xc5d0;  ←
&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624; .
-- &#xc774; &#xcffc;&#xb9ac;&#xb294; &#xc0dd;&#xc131;&#xb41c;  ←
&#xd30c;&#xc77c;&#xba85;&#xc744; &#xd654;&#xba74;&#xc5d0;  ←
&#xcd9c;&#xb825;&#xd569;&#xb2c8;&#xb2e4;;

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
```



```
-- &#xb798;&#xc2a4;&#xd130;; 5 x 5 &#xd53d;&#xc140;; ,&#xbc34;&#xb4dc; 3&#xac1c;, PT_8BUI ←  
    &#xd53d;&#xc140; &#xc720;&#xd615;;, NODATA = 0  
(2, ('01000003009A9999999999A93F9A9999999999A9BF000000E02B274A' ||  
'4100000000771956410000000000000000000000000000000000000000000000 ←  
      FFFFFFFF050005000400FDFFDFEFDFEFDFEFDFEFDF9FAFEF' ||  
' ←  
      EFCF9FBFDFEFDFCFAFEFEFE04004E627AADD16076B4F9FE6370A9F5FE59637AB0E54F58617087040046566487A1506  
      ')::raster);
```

12.1

12.1.1 geomval

geomval — (τ 형 객 쒴 를 담 고 있 는) geom 과 (래 &#τ 드 의 이 중 정 ଀ 도 픽 셀 값 을 담 &#있 는) val, 두 개 의 필 드 를 가 진 공 간 데 이 터 형 입 니 다.

설명

```
geomval&#xc740; &#xbcf5;&#xd569; &#xb370;&#xc774;&#xd130; &#xc720;&#xd615;&#xc73c;&#xb85c;, geom &#xd544;&#xb4
&#xccc38;&#xc870;&#xd558;&#xb294; &#xb3c4;&#xd615; &#xac1d;&#xccb4;&#xc640; &#xb798;&#xc2a4;&#xd130; &#xbc34;&#
&#xb0b4;&#xbd80;&#xc758; &#xd2b9;&#xc815; &#xae30;&#xd558; &#xc704;&#xce58;&#xc5d0; &#xc788;&#xb294; &#xd53d;&
&#xb098;&#xd0c0;&#xb0b4;&#xb294; &#xc774;&#xc911; &#xc815;&#xbc00;&#xb3c4; &#xac12;&#xc778; val&#xb85c;
&#xad6c;&#xc131;&#xb429;&#xb2c8;&#xb2e4;. ST_DumpAsPolygon&#xbcf0; &#xb798;&#xc2a4;&#xd130; &#xad50;&#xcc28;
&#xacc4;&#xc5f4; &#xd568;&#xc218;&#xb4e4;&#xc774; &#xb798;&#xc2a4;&#xd130; &#xbc34;&#xb4dc;&#xb97c; &#xb3c4;&#
&#xd3f4;&#xb9ac;&#xace4;&#xb4e4;&#xb85c; &#xbd84;&#xd574;&#xd558;&#xae30; &#xc704;&#xd55c; &#xcd9c;&#xb825;&#x
&#xc720;&#xd615;&#xc73c;&#xb85c; &#xc774; &#xb370;&#xc774;&#xd130;&#xd615;&#xc744; &#xc774;&#xc6a9;&#xd569;&#
```

స고

Section 15.6

12.1.2 addbandarg

addbandarg — 새로운밴드의속성및초기&#xac12
정의하는 ST_AddBand 함수의입력물로
이용되는복합데이터형입니다

설명

새로운밴드의속성및초기값을정의하는 ST_AddBand 함수의입력물로이용되는복합데이터형입니다

```
index integer &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbc34;&#xb4dc;&#xb4e4; &#xc0ac;&#xc774; &#xc5b4;&#xb5a4;
&#xc704;&#xce58;&#xc5d0; &#xc0c8; &#xbc34;&#xb4dc;&#xb97c; &#xcd94;&#xac00;&#xd560;&#xc9c0; &#xc9c0;&#xc2
1-&#xae30;&#xbc18; &#xac12;&#xc785;&#xb2c8;&#xb2e4;. NULL&#xc77c; &#xacbd;&#xc6b0;. &#xb798;&#xc2a4;&#xd1
&#xbc34;&#xb4dc;&#xc758; &#xb9c8;&#xc9c0;&#xb9c9;&#xc5d0; &#xc0c8; &#xbc34;&#xb4dc;&#xb97c; &#xcd94;&#xac
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
pixeltype text ST_BandPixelType &#xc5d0;&#xc11c;&#xc124;&#xba85;&#xd558;&#xace0;&#xc788;&#xb294;&#xd53d;&#x
&#xc720;&#xd615;&#xb4e4;&#xac00;&#xc6b4;&#xb370;&#xd558;&#xb098;&#xc785;&#xb2c8;&#xb2e4;.
```



```
initialvalue double precision &#xc0c8;&#xbc34;&#xb4dc;&#xc758;&#xbaa8;&#xb4e0;&#xd53d;&#xc140;&#xb4e4;&#xc7
&#xc124;&#xc815;&#xd558;&#xac8c;&#xb420;&#xcd08;&#xae30;&#xac12;&#xc785;&#xb2c8;&#xb2e4;.
```

```
nodataval double precision &#xc0c8;&#xbc34;&#xb4dc;&#xc758; NODATA &#xac12;&#xc785;&#xb2c8;&#xb2e4;. NULL&#xacbd;&#xc6b0;,&#xc0c8;&#xbc34;&#xb4dc;&#xc5d0; NODATA &#xac12;&#xc744;&#xd560;&#xb2f9;&#xd558;&#xc9c5;&#xc54a;&#xc744; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

󌰸고

ST AddBand

12.1.3 rastbandarg

rastbandarg — 래스터 및 해당 래스터의 밴&#xb4d
인덱스를 표현해야 할 경우 쓰이&#
복합 데이터형입니다.

설명

래스터 및 해당 래스터의 밴드 인표현해야 할 경우 쓰이는 복합 데일

```
rast raster &#xd574;&#xb2f9; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc9c0;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.
```

nband integer ޘ스터의 밴드를 지시하는
1-기반 값입니다

స고

&#xb0b4:&#xc7a5:&#xb9f5:&#xb300:&#xc218:&#xcf5c:&#xbc31:&#xd568:&#xc218:

12.1.4 raster

raster — 래스&#xd130: 공&#xac04: 데이터형입니&#xb2e4:.

&#xc124:&#xba85:

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

```
GDAL &#xc9c0;&#xc6d0;&#xacfc; &#xd568;&#xaed8; &#xcef4;&#xd30c;&#xc77c;&#xd55c; PostGIS&#xac00;
&#xd544;&#xc694;&#xd569;&#xb2c8;&#xb2e4;. &#xd604;&#xc7ac; &#xb798;&#xc2a4;&#xd130;&#xb97c;
&#xc81c;&#xc57d; &#xc5c6;&#xc774; &#xb3c4;&#xd615; &#xc720;&#xd615;&#xc73c;&#xb85c;
&#xbcc0;&#xd658;&#xd560; &#xc218; &#xc788;&#xc9c0;&#xb9cc; &#xc774; &#xbcc0;&#xd658;
&#xc791;&#xc5c5;&#xc740; &#xb798;&#xc2a4;&#xd130;&#xc758; ST_ConvexHull &#xc744;
&#xbcb8;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. &#xc774; &#xc790;&#xb3d9; &#xd615;&#xbcc0;&#xd658;
&#xae30;&#xb2a5;&#xc740; &#xac00;&#xae4c;&#xc6b4; &#xc2dc;&#xc77c; &#xb0b4;&#xc5d0;
&#xc81c;&#xcac70;&#xb420; &#xc218; &#xc788;&#xc73c;&#xbbc0;&#xb85c; &#xc758;&#xc874;&#xd558;&#xc9c0;
&#xc54a;&#xb294; &#xd3b8;&#xc774; &#xc88b;&#xc2b5;&#xb2c8;&#xb2e4;.
```

ST_Reclass (Casting Behavior)

ST_Reclass — Reclassify a raster by using a mapping table. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values.

ST_Reclass	ST_Reclass
ST_Reclass	ST_Reclass

ST_Reclass

Chapter 12

12.1.5 reclassarg

reclassarg — Reclassify a raster by using a mapping table. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values.

ST_Reclass

ST_Reclass — Reclassify a raster by using a mapping table. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values.

nband integer — Reclassify a raster by using a mapping table. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values.

reclassexpr text — Reclassify a raster by using a mapping table. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values.

1. [a-b] = a <= x <= b
2. (a-b) = a < x <= b
3. [a-b) = a <= x < b
4. (a-b) = a < x < b

'(' 기 호 는 선 택 적 이 기 때 ସ 에 (a-b) 와 a-b 의 의 미 는 동 일 합 니 다.

pixeltype text ST_BandPixelType — Reclassify a raster by using a mapping table. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values.

nodataval double precision — Reclassify a raster by using a mapping table. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values. The mapping table is a table with two columns: the first column is the original value, and the second column is the new value. The mapping table is used to reclassify the raster values.

예 ଴ 드 2 를 255 ఀ NODATA ఒ 인 8BUI 로 재 분 류 &

SELECT ROW(2, '0-100:1-10, 101-500:11-150, 501 - 10000: 151-254', '8BUI', 255)::reclassarg;

```
1BB'::reclassarg;
1BB'::reclassarg;
```

```
SELECT ROW(1, '0-100]:0, (100-255:1', '1BB', NULL)::reclassarg;
```

```
ST_Reclass
```

12.1.6 summarystats

summarystats — **ST_SummaryStats**, **ST_SummaryStatsAgg**

```
ST_SummaryStats, ST_SummaryStatsAgg
```

ST_SummaryStats, **ST_SummaryStatsAgg**

count integer

sum double precision

mean double precision

stdev double precision

min double precision

max double precision

```
ST_SummaryStats, ST_SummaryStatsAgg
```

12.1.7 unionarg

unionarg — **ST_Union**

```
ST_Union
```

ST_Union

nband integer

uniontype text

```
&#xccc38;&#xace0;
```

```
ST_Union
```

12.2 래스터 관리

12.2.1 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 nodata_values=true, boolean out_db=true, boolean extent=true);
boolean AddRasterConstraints(name rasttable, name rastcolumn, text[] VARIADIC constraints);
boolean AddRasterConstraints(name rastschema, name rasttable, name rastcolumn, text[] VARIADIC constraints);
boolean AddRasterConstraints(name rastschema, name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=true,
boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true, boolean same_alignment=true, boolean regular_blocking=true,
boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true, boolean out_db=true, boolean extent=true);
```

```
&#xc124;&#xba85;
```

```
raster_columns &#xb798;&#xc2a4;&#xd130; &#xce74;&#xd0c8;&#xb85c;&#xadf8;&#xc5d0; &#xc815;&#xbcf4;&#xb97c;
&#xd45c;&#xcd9c;&#xd558;&#xb294; &#xb370; &#xc4f0;&#xc774;&#xb294;, &#xb798;&#xc2a4;&#xd130; &#xc5f4;&#xc5d0;
&#xb300;&#xd55c; &#xc81c;&#xc57d;&#xc870;&#xac74;&#xb4e4;&#xc744; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
rastschema &#xb294; &#xd574;&#xb2f9; &#xd14c;&#xc774;&#xbe14;&#xc774; &#xc18d;&#xd574; &#xc788;&#xb294;
&#xd14c;&#xc774;&#xbe14; &#xc2a4;&#xd0a4;&#xb9c8;&#xc758; &#xba85;&#xce6d;&#xc785;&#xb2c8;&#xb2e4;. srid
&#xb294; SPATIAL_REF_SYS &#xd14c;&#xc774;&#xbe14;&#xc5d0; &#xc788;&#xb294; &#xd56d;&#xbaa9;&#xc744; &#ccc38;&#xc870;&#xd558;
&#xc815;&#xc218;&#xd615; &#xac12;&#xc774;&#xc5b4;&#xc57c;&#xb9cc; &#xd569;&#xb2c8;&#xb2e4;.
```

```
raster2pgsql &#xb85c;&#xb354;&#xb294; &#xb798;&#xc2a4;&#xd130; &#xd14c;&#xc774;&#xbe14;&#xc744; &#xb4f1;&#xb798;
&#xb370; &#xc774; &#xd568;&#xc218;&#xb97c; &#xc774;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
```

```
&#xc785;&#xb825;&#xd560; &#xc720;&#xd6a8;&#xd55c; &#xc81c;&#xc57d;&#xc870;&#xac74; &#xba85;&#xce6d;&#xc5d0;
&#xb300;&#xd55c; &#xc790;&#xc138;&#xd55c; &#xb0b4;&#xc6a9;&#xc740; Section 11.2.1 &#xb97c; &#ccc38;&#xc870;&#xd558;
```

- `blocksize` 는 X 및 Y 블록 크기를 둘 다 설정을
- `blocksize_x` 는 X 타일(각 타일의 너비 픽셀 개수)을 설정합니다.
- `blocksize_y` 는 Y 타일(각 타일의 높이 픽셀 개수)을 설정합니다.
- `extent` 는 전체 테이블의 범위를 계산제약조건을 적용합니다. 모든 래해당 &#bc94;위 내에 들어와야만 합니다.
- `num_bands` 는 밴드의 개수입니다.
- `pixel_types` 각 밴드에 대해 픽셀 유형의 배열을 읽어 들입니다. 모든 밴드 N이 동일한 픽셀 유형을 가지도록 보장합니다.

- `regular_blocking` ensures they all have same alignment meaning any two tiles you compare will return true for. Refer to [ST_SameAlignment](#).
- `srid` SRID
- `--`

Note

`이 함수는 테이블에 이미`
`존재하는 데이터에 제약조을을`
`적용합니다 따라서 이 함수ఀ`
`작동하려면 먼저 래스터`
`열을 생성한 다음 데이터를`
`로드해야만 합니다`

Note

`사용자ఀ 제약조을을 적용한`
`다음 추ఀ 데이터를 로드해야`
`할 쪽우 사용자 데이터의`
`맄위ఀ 변쪽된다면 DropRasterCon-`
`straints 함수를 실행하는 편이`
`좋습니다`

2.0.0 ஄전부터 사용할 수 있습니다

예시 데이터에 기ଘ한 열에 ఀ능을
 한 모든 제약조을이 적용하기

```

CREATE TABLE myrasters(rid SERIAL primary key, rast raster);
INSERT INTO myrasters(rast)
SELECT ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0.3, -0.3, 2, 2, 0, 0,4326), 1, '8BSI'::
  text, -129, NULL);

SELECT AddRasterConstraints('myrasters'::name, 'rast'::name);

-- raster_columns &#xbdf0;&#xc5d0; &#xc815;&#xd655;&#xd788; &#xb4f1;&#xb85d;&#xb410;&#xb294;&#xc9c0; &#xd655;&#xc778; --
SELECT srid, scale_x, scale_y, blocksize_x, blocksize_y, num_bands, pixel_types,
  nodata_values
  FROM raster_columns
  WHERE r_table_name = 'myrasters';

srid | scale_x | scale_y | blocksize_x | blocksize_y | num_bands | pixel_types |
-----+-----+-----+-----+-----+-----+-----+
4326 |      2 |      2 |      1000 |      1000 |          1 | {8BSI}      | {0}
  
```

```
CREATE TABLE public.myrasters2(rid SERIAL primary key, rast raster);
INSERT INTO myrasters2(rast)
SELECT ST_AddBand(ST_MakeEmptyRaster(1000, 1000, 0.3, -0.3, 2, 2, 0, 0,4326), 1, '8BSI':: ↵
    text, -129, NULL);

SELECT AddRasterConstraints('public'::name, 'myrasters2'::name, 'rast'::name, ' ↵
    regular_blocking', 'blocksize');
-- Adding regular blocking constraint
NOTICE: Adding regular blocking constraint
NOTICE: Adding blocksize-X constraint
NOTICE: Adding blocksize-Y constraint
```

```
Section 11.2.1, ST_AddBand, ST_MakeEmptyRaster, DropRasterConstraints, ST_BandPixelType, ST_SRID
```

12.2.2 DropRasterConstraints

DropRasterConstraints — Removes raster constraints. The constraints to be removed are specified by the `constraints` argument. The constraints to be removed are specified by the `constraints` argument. The constraints to be removed are specified by the `constraints` argument.

Synopsis

DropRasterConstraints(name rasttable, name rastcolumn, boolean srid, boolean scale_x, boolean scale_y, boolean blocksize_x, boolean blocksize_y, boolean same_alignment, boolean regular_blocking, boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true, boolean out_db=true, boolean extent=true);

DropRasterConstraints(name rastschema, name rasttable, name rastcolumn, boolean srid=true, boolean scale_x=true, boolean scale_y=true, boolean blocksize_x=true, boolean blocksize_y=true, boolean same_alignment=true, boolean regular_blocking=false, boolean num_bands=true, boolean pixel_types=true, boolean nodata_values=true, boolean out_db=true, boolean extent=true);

DropRasterConstraints(name rastschema, name rasttable, name rastcolumn, text[] constraints);

```
AddRasterConstraints('public'::name, 'myrasters2'::name, 'rast'::name, ' ↵
    regular_blocking', 'blocksize');
```

AddRasterConstraints — Adds raster constraints. The constraints to be added are specified by the `constraints` argument. The constraints to be added are specified by the `constraints` argument. The constraints to be added are specified by the `constraints` argument.

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 nodata_values=true, boolean out_db=true, boolean extent=true);

```
DROP TABLE mytable
```

DropRasterConstraints — Removes raster constraints. The constraints to be removed are specified by the `constraints` argument. The constraints to be removed are specified by the `constraints` argument. The constraints to be removed are specified by the `constraints` argument.


```
ovschema; refschema;
search_path;
&#xd14c;&#xc774;&#xbe14;&#xc744; &#xc774;&#xc6a9;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
2.0.0 &#xbc84;&#xc804;&#xbd80;&#xd130; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시

```
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);

-- raster_overviews &#xbd0;&#xc5d0; &#xc815;&#xd655;&#xd788; ↔
&#xb4f1;&#xb85d;&#xb410;&#xb294;&#xc9c0; &#xd655;&#xc778; --
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)
```

참고

Section 11.2.2, [DropOverviewConstraints](#), [ST_CreateOverview](#), [AddRasterConstraints](#)

12.2.4 DropOverviewConstraints

DropOverviewConstraints — 또 다른 래스터 열의 미리래스터 열을 태그 해제합니다.

Synopsis

```
boolean DropOverviewConstraints(name ovschema, name ovtable, name ovcolumn);
boolean DropOverviewConstraints(name ovtable, name ovcolumn);
```

설명

```
raster_overviews &#xb798;&#xc2a4;&#xd130; &#xc74;&#xd0c8;&#xb85c;&#xadf8;&#xc5d0; &#xb610; &#xb2e4;&#xb978;
&#xb798;&#xc2a4;&#xd130; &#xc5f4;&#xc758; &#xbbf8;&#xb9ac;&#xbcf4;&#xae30;&#xb97c; &#xd45c;&#xcd9c;&#xd558;&#xb370; &#xc4f0;&#xc774;&#xb294; &#xb798;&#xc2a4;&#xd130; &#xc5f4;&#xc758; &#xc81c;&#xc57d;&#xc870;&#xac74;&#xc81c;&#xac70;&#xd569;&#xb2c8;&#xb2e4;.
```



```
ovschema &#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb97c;&#xc0dd;&#xb7b5;&#xd560;&#xacbd;&#xc6b0;, search_path
&#xb97c;&#xc2a4;&#xce94;&#xd558;&#xba74;&#xc11c;&#xcc98;&#xc74c;&#xbc1c;&#xacac;&#xd55c;&#xd14c;&#xc774;&#x
&#xc774;&#xc6a9;&#xd560;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

2.0.0 ಄전부터 사용할 수 있습니다.

󌰸고

Section 11.2.2, AddOverviewConstraints, DropRasterConstraints

12.2.5 PostGIS_GDAL_Version

PostGIS_GDAL_Version — PostGIS; GDAL

Synopsis

```
text PostGIS_GDAL_Version();
```

설명

PostGIS ఀ 이 용 하 À 있 는 GDAL 라 이 브 러 리 버 전 을 반 환 합 니 다. 또 GDAL 이 자 살 데 이 터 파 일 을 찾 을 수 있 는 지 도 확 인 해 서 보 À 할 ಃ 입 니 다.

예시

```
SELECT PostGIS_GDAL_Version();
       postgis_gdal_version
-----
GDAL 1.11dev, released 2013/04/13
```

స고

```
postgis.gdal_datapath
```

12.2.6 PostGIS Raster Lib Build Date

PostGIS_Raster_Lib_Build_Date — 전체 래스터 라이ฌ러¹ ์드 날짜를 반환합니다.

Synopsis

```
text PostGIS_Raster_Lib_Build_Date();
```

설명

래스터의 빌드 날짜를 보고합니&#x

PostGIS_Raster_Lib_Build_Date();

```
SELECT PostGIS_Raster_Lib_Build_Date();
postgis_raster_lib_build_date
-----
2010-04-28 21:15:10
```

PostGIS_Raster_Lib_Version

PostGIS_Raster_Lib_Version

12.2.7 PostGIS_Raster_Lib_Version

PostGIS_Raster_Lib_Version — Returns the version of the PostGIS raster library. Only those formats with `can_write=True` can be used by `ST_AsGDALRaster`.

Synopsis

text **PostGIS_Raster_Lib_Version()**;

PostGIS_Raster_Lib_Version

PostGIS_Raster_Lib_Version — Returns the version of the PostGIS raster library. Only those formats with `can_write=True` can be used by `ST_AsGDALRaster`.

PostGIS_Raster_Lib_Version

```
SELECT PostGIS_Raster_Lib_Version();
postgis_raster_lib_version
-----
2.0.0
```

PostGIS_Lib_Version

PostGIS_Lib_Version

12.2.8 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);

short_name	long_name	can_write
AAIGrid	Arc/Info ASCII Grid	t
ACE2	ACE2	f
ADRG	ARC Digitized Raster Graphics	f
AIG	Arc/Info Binary Grid	f
AirSAR	AirSAR Polarimetric Image	f
ARG	Azavea Raster Grid format	t
BAG	Bathymetry Attributed Grid	f
BIGGIF	Graphics Interchange Format (.gif)	f
BLX	Magellan topo (.blx)	t
BMP	MS Windows Device Independent Bitmap	f
BSB	Maptech BSB Nautical Charts	f
PAux	PCI .aux Labelled	f
PCIDSK	PCIDSK Database File	f
PCRaster	PCRaster Raster File	f
PDF	Geospatial PDF	f
PDS	NASA Planetary Data System	f
PDS4	NASA Planetary Data System 4	t
PLMOSAIC	Planet Labs Mosaics API	f
PLSCENES	Planet Labs Scenes API	f
PNG	Portable Network Graphics	t
PNM	Portable Pixmap Format (netpbm)	f
PRF	Racurs PHOTOMOD PRF	f
R	R Object Data Store	t
Rasterlite	Rasterlite	t
RDA	DigitalGlobe Raster Data Access driver	f
RIK	Swedish Grid RIK (.rik)	f
RMF	Raster Matrix Format	f
ROI_PAC	ROI_PAC raster	f
RPFTOC	Raster Product Format TOC format	f
RRASTER	R Raster	f
RS2	RadarSat 2 XML Product	f
RST	Idrisi Raster A.1	t
SAFE	Sentinel-1 SAR SAFE Product	f
SAGA	SAGA GIS Binary Grid (.sdat, .sg-grd-z)	t
SAR_CEOS	CEOS SAR Image	f

SDTS	SDTS Raster	f
SENTINEL2	Sentinel 2	f
SGI	SGI Image File Format 1.0	f
SNODAS	Snow Data Assimilation System	f
SRP	Standard Raster Product (ASRP/USRP)	f
SRTMHGT	SRTMHGT File Format	t
Terragen	Terragen heightfield	f
TIL	EarthWatch .TIL	f
TSX	TerraSAR-X Product	f
USGSDEM	USGS Optional ASCII DEM (and CDED)	t
VICAR	MIPL VICAR file	f
VRT	Virtual Raster	t
WCS	OGC Web Coverage Service	f
WMS	OGC Web Map Service	t
WMTS	OGC Web Map Tile Service	t
XPM	X11 Pixmap Format	t
XYZ	ASCII Gridded XYZ	t
ZMap	ZMap Plus Grid	t

예시: 각 드라이버에 대한 옵션 목&

```
-- JPEG&#xc758; &#xc0dd;&#xc131; &#xc635;&#xc158; XML &#xc5f4;&#xc744; &#xd45c;&#xb85c; ←
&#xcd9c;&#xb825; --
-- &#xc774; &#xc0dd;&#xc131; &#xc635;&#xc158;&#xb4e4;&#xc744; ST_AsGDALRaster&#xc758; ←
&#xc778;&#xc218;&#xb85c; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xb2e4;&#xb294; ←
&#xc810;&#xc744; &#xae30;&#xc5b5;&#xd558;&#xc2ed;&#xc2dc;&#xc624; .
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 geneate 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)

```
-- GeoFiff&#xc5d0; &#xb300;&#xd55c; &#xc0dd;&#xc131; &#xc635;&#xc158;&#xc758; ←
&#xac00;&#xacf5;&#xb418;&#xc9c0; &#xc54a;&#xc740; XML &#xcd9c;&#xb825;&#xb3c; --
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  
>  
  
-- GeoFiff&#xc5d0; &#xb300;&#xd55c; &#xc0dd;&#xc131; &#xc635;&#xc158; XML &#xc5f4;&#xc744; ↵  
    &#xd45c;&#xb85c; &#xcd9c;&#xb825; --  
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, ↵
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	↩
		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)		

స고

ST_AsGDALRaster, **ST_SRID**, **postgis.gdal_enabled_drivers**

12.2.9 ST_Count

ST_Count — Generates a set of vector contours from the provided raster band, using the **GDAL contouring algorithm**.

Synopsis

raster **ST_TPI**(raster rast, integer nband, raster customextent, text pixeltype="32BF" , boolean interpolate_nodata=FALSE);

설명

Generates a set of vector contours from the provided raster band, using the **GDAL contouring algorithm**.

When the `fixed_levels` parameter is a non-empty array, the `level_interval` and `level_base` parameters are ignored.

The `polygonize` parameter currently has no effect. Use the **ST_Polygonize** function to convert contours into polygons.

Return values are a set of records with the following attributes:

geomval The geometry of the contour line.

id A unique identifier given to the contour line by GDAL.

ST_Value The raster value the line represents. For an elevation DEM input, this would be the elevation of the output contour.

2.2.0 버전부터 사용할 수 있습니다.

```
&#xc608;&#xc2dc;
```

```
WITH c AS (
SELECT (ST_Contour(rast, 1, fixed_levels => ARRAY[100.0, 200.0, 300.0])).*
FROM dem_grid WHERE rid = 1
)
SELECT st_astext(geom), id, value
FROM c;
```

```
&#xcc38;&#xace0;
```

[ST_MakeEmptyRaster](#)

12.2.10 ST_MakeEmptyRaster

ST_MakeEmptyRaster — Interpolates a gridded surface based on an input set of 3-d points, using the X- and Y-values to position the points on the grid and the Z-value of the points as the surface elevation.

Synopsis

bytea **ST_AsGDALRaster**(raster rast, text format, text[] options=NULL, integer srid=sameassource);

```
&#xc124;&#xba85;
```

Interpolates a gridded surface based on an input set of 3-d points, using the X- and Y-values to position the points on the grid and the Z-value of the points as the surface elevation. There are five interpolation algorithms available: inverse distance, inverse distance nearest-neighbor, moving average, nearest neighbor, and linear interpolation. See the [gdal_grid documentation](#) for more details on the algorithms and their parameters. For more information on how interpolations are calculated, see the [GDAL grid tutorial](#).

Input parameters are:

input_points The points to drive the interpolation. Any geometry with Z-values is acceptable, all points in the input will be used.

algorithm_options A string defining the algorithm and algorithm options, in the format used by [gdal_grid](#). For example, for an inverse-distance interpolation with a smoothing of 2, you would use "invdist:smoothing=2.0"

template A raster template to drive the geometry of the output raster. The width, height, pixel size, spatial extent and pixel type will be read from this template.

template_band_num By default the first band in the template raster is used to drive the output raster, but that can be adjusted with this parameter.

2.2.0 버전부터 사용할 수 있습니다.

```
&#xc608;&#xc2dc;
```

```
SELECT ST_InterpolateRaster(
  'MULTIPOINT(10.5 9.5 1000, 11.5 8.5 1000, 10.5 8.5 500, 11.5 9.5 500)'::geometry,
  'invdist:smoothing:2.0',
  ST_AddBand(ST_MakeEmptyRaster(200, 400, 10, 10, 0.01, -0.005, 0, 0), '16BSI')
)
```


స고

ST_Count

12.2.11 UpdateRasterSRID

```
UpdateRasterSRID — &#xc0ac;&#xc6a9;&#xc790;&#xc00; &#xc9c0;&#xc815;&#xd55c; &#xc5f4; &#xb0f; &#xd14c;&#xc774;&#xc788;&#xb294; &#xbaa8;&#xb4e0; &#xb798;&#xc2a4;&#xd130;&#xc758; SRID&#xb97c; &#xbcc0;&#xacbd;&#xd569;&#xb2
```

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);

설명

사용자가지정한열및테이블에있는모든래스터의SRID를변경합는이함수는지정한열의래스터의SRID를변경하기전에적절한모든열제약조건들(범위,정렬방향및SRID)을삭제할것입니다.

Note



```
&#xb798;&#xc2a4;&#xd130;&#xc758;      &#xb370;&#xc774;&#xd130;(&#xbc34;&#xb4dc;      &#xd53d;&#xc140;
&#xac12;)&#xb97c;      &#xac74;&#xb4dc;&#xb9ac;&#xc9c0;&#xb294;      &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;.
&#xb798;&#xc2a4;&#xd130;&#xc758;      &#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130;&#xb9cc;
&#xbcc0;&#xacbd;&#xd569;&#xb2c8;&#xb2e4;.
```

2.1.0 버전부터 사용할 수 있습니다.

స고

UpdateGeometrySRID

12.2.12 ST CreateOverview

ST_CreateOverview — [ޅ력래스터커버리지의저버전을생성합니다](#).

Synopsis

```
regclass ST_CreateOverview(regclass tab, name col, int factor, text algo='NearestNeighbor');
```

설명

```
&#xc18c;&#xc2a4; &#xd14c; &#xc774; &#xbe14; &#xb85c; &#xbd80; &#xd130; &#xb9ac; &#xc0d8; &#xd50c; &#xb9c1; &#xd55c;
&#xd0c0; &#xc77c; &#xb4e4; &#xc744; &#xb2f4; &#xc740; &#xbbf8; &#xb9ac; &#xbcf4; &#xae30; &#xd14c; &#xc774; &#xbe14; &#xc0dd; &#xc131; &#xd569; &#xb2c8; &#xb2e4;. &#xcd9c; &#xb825; &#xd0c0; &#xc77c; &#xc740; &#xc785; &#xb825; &#xd0c0; &#xd06c; &#xae30; &#xac00; &#xb3d9; &#xc77c; &#xd558; &#xace0;, &#xb354; &#xb0ae; &#xc740; &#xd574; &#xc0c1; &#xb3c4; &#xb3d9; &#xc77c; &#xd55c; &#xacf5; &#xac04; &#xbc94; &#xc704; &#xb97c; &#xcee4; &#xbc84; &#xd569; &#xb2c8; &#xb2e4; (&#xd06c; &#xae30; &#xb294; &#xc591; &#xbc29; &#xd5a5; &#xbaa8; &#xb450; &#xc6d0; &#xbcf8; &#xc758; 1/factor &#xc774; &#xb420; &#xac83; &#xc785; &#xb2c8; &#xb2e4;).
```

```
raster_overviews &#xe74;&#xd0c8;&#xb85c;&#xadf8;&#xc5d0;&#xc11c; &#xc774; &#xbbf8;&#xb9ac;&#xbcf4;&#xae30;
&#xd14c;&#xc774;&#xbe14;&#xc744; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xac8c; &#xb418;&#xace0;, &#xb798;&#x
&#xc81c;&#xc57d;&#xc870;&#xac74;&#xb3c4; &#xac15;&#xc81c;&#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

알고리즘옵션에는 'NearestNeighbor', 'Bilinear', 'Cubic', 'Cubic-Spline', 그리고 'Lanczos'가있습니다. 자세한내용은 GDAL Warp resampling methods 를쏃조하십시오.

2.2.0 버전부터 사용할 수 있습니다.

예시

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);
```

సೠ

ST_Retile, AddOverviewConstraints, AddRasterConstraints, Section 11.2.2

12.3 `<math>\mathbb{R}</math>` (constructor)

12.3.1 ST AddBand

ST_AddBand — 입력 인덱스 위치에 입력 초&#xa
추가된 입력 유형의 새 밴드(들)을
가진 래스터를 반환합니다 인덱&#xc
설정하지 않을 경우,마지막 위치&#
밴드를 추가합니다.

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);

설명

입력 인덱스 위치에 입력 초기값,
 입력 유형 그리고 입력 NODATA 값의 새
 밴드를 가진 래스터를 반환합니인덱스를 설정하지 않을 경우 마위치에 밴드를 추가합니다 fromband 를
 설정하지 않을 경우 밴드 1로 가정픽셀 유형은 **ST_BandPixelType** 에서 지정한 픽유형들 가운데 하나의 문자열 표만약 기존 인덱스를 설정할 경우 해당 인덱스보다 크거나 같은 그
 이후의 모든 밴드들의 인덱스가 1씩 증가합니다 픽셀 유형의 최대큰 초기값을 설정할 경우 해당 픽유형이 가질 수 있는 가장 높은 값초기값을 설정합니다.

addbandarg 의 배열을 입력받는 변종 1의 경우 특정 addbandarg의 인덱스 값은 addbandarg가묘사하는 &#bc34;드가 래스터에 추가당시의 해당 래스터에 상대적입다음에 나오는 &#bcf5;수의 새로운 &#bc34;예시를 참고하십시오.

래스터 배열을 입력받는 변종 5의 경우 torast 가 NULL이라면 배열 안에 있는 각 래스터의 fromband &#bc34;드를 새 래스터 누ૄ (累ਈ)합니다.

outdbfile 을 입력받는 변종 6 및 7의 경우 outdbfile 값이 래스터 파일을 가리키는 전체 경로를 포함해야만 합니다 PostgreSQL 서버 프로세스도 해당 파일에 접근할 수 있어야만 합니다.

개선 사항 2.1.0 버전부터 addbandarg를 지원항 개선 사항 2.1.0 버전부터 새로운 DB 외들 &#bc34;드를 지원합니다.

예시 새로운 단일 밴드

```
-- &#xd53d;&#xc140; &#xcd08;&#xae30;&#xac12;&#xc744; 200&#xc73c;&#xb85c; <←
  &#xc124;&#xc815;&#xd55c; &#xbd80;&#xd638;&#xac00; &#xc5c6;&#xb294; (unsigned) 8 <←
  &#be44;&#xd2b8; &#xc815;&#xc218;&#xd615; &#xc720;&#xd615;&#xc758; &#xb610; <←
  &#b2e4;&#xb978; &#bc34;&#xb4dc;&#xb97c; &#xcd94;&#xac00;
UPDATE dummy_rast
  SET rast = ST_AddBand(rast,'8BUI'::text,200)
WHERE rid = 1;
```

```
-- &#xc88c;&#xc0c1;&#xb2e8;&#xc774; &#xb531; 0&#xc778; 100x100 &#xb2e8;&#xc704;&#xc758; <←
  &#be48; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd558;&#xace0; ( <←
  &#bc34;&#xb4dc; 1&#xc740; 0/1 &#xbd88; &#xac12; &#be44;&#xd2b8; <←
  &#xc2a4;&#xc704;&#xce58; &#bc34;&#xb4dc; 2&#xb294; 0&#xc5d0;&#xc11c; 15 <←
  &#xc0ac;&#xc774;&#xc758; &#xac12;&#xc744; &#xac00;&#xc9c8; &#xc218; &#xc788;&#xb294;) <←
  &#bc34;&#xb4dc; 2&#xac1c;&#xb97c; &#xcd94;&#xac00;
-- addbandargs&#xb97c; &#xc774;&#xc6a9;
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[]
    )
);

-- &#xbaa8;&#xb4e0; &#xc791;&#xc5c5;&#xc774; &#xc815;&#xc0c1;&#xc778;&#xc9c0;  ↵
&#xd655;&#xc778;&#xd558;&#xae30; &#xc704;&#xd574; &#xb798;&#xc2a4;&#xd130;  ↵
&#xbc34;&#xb4dc;&#xb4e4;&#xc758; &#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130;&#xb97c;  ↵
&#xcd9c;&#xb825; --
SELECT (bmd).*
FROM (SELECT ST_BandMetaData(rast,generate_series(1,2)) As bmd
      FROM dummy_rast WHERE rid = 10) AS foo;
-- &#xacb0;&#xacfc; --
pixeltype | nodatavalue | isoutdb | path
-----+-----+-----+-----
1BB      |             | f       |
4BUI     |             | f       |

-- &#xb798;&#xc2a4;&#xd130;&#xc758; &#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130;&#xb97c;  ↵
&#xcd9c;&#xb825; --
SELECT (rmd).width, (rmd).height, (rmd).numbands
FROM (SELECT ST_MetaData(rast) As rmd
      FROM dummy_rast WHERE rid = 10) AS foo;
-- &#xacb0;&#xacfc; --
upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid |  ↵
numbands
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
0 | 0 | 100 | 100 | 1 | -1 | 0 | 0 | 0 |  ↵
2
```

예시: 복수의 새로운 밴드

```

SELECT
    *
FROM ST_BandMetadata(
    ST_AddBand(
        ST_MakeEmptyRaster(10, 10, 0, 0, 1, -1, 0, 0, 0),
        ARRAY[
            ROW(NULL, '8BUI', 255, 0),
            ROW(NULL, '16BUI', 1, 2),
            ROW(2, '32BUI', 100, 12),
            ROW(2, '32BF', 3.14, -1)
        ]::addbandarg[]
    ),
    ARRAY[]::integer[]
);

bandnum | pixeltype | nodatavalue | isoutdb | path
-----+-----+-----+-----+-----
1 | 8BUI      | 0 | f |
2 | 32BF      | -1 | f |
3 | 32BUI     | 12 | f |
4 | 16BUI     | 2 | f |

-- &#xb3d9;&#xc77c;&#xd55c; &#xc720;&#xd615;&#xc758;  ↵
&#xb798;&#xc2a4;&#xd130;&#xb4e4;&#xc758; &#xd14c;&#xc774;&#xbe14;&#xc758; &#xccab;  ↵
```

```

--      &#xc88;&#xc9f8; &#xbc34;&#xb4dc;&#xb97c; test_types&#xb9cc;&#xd07c; &#xb9ce;&#xc740; <--
--      &#xbc34;&#xb4dc;&#xb4e4;&#xacfc;
--      mice&#xb9cc;&#xd07c; &#xb9ce;&#xc740; &#xd589;&#xb4e4; (&#xc0c8; <--
--      &#xb798;&#xc2a4;&#xd130;&#xb4e4;)&#xacfc; &#xd568;&#xaed8; &#xb2e8;&#xc77c; <--
--      &#xb798;&#xc2a4;&#xd130;&#xb85c; &#xc885;&#xd569;
--      &#xc8fc;&#xc758;: PostgreSQL 9.0 &#xc774;&#xc0c1; &#xbc84;&#xc804;&#xb9cc; ORDER BY <--
--      test_type&#xc744; &#xc9c0;&#xc6d0;&#xd569;&#xb2c8;&#xb2e4;.
--      8.4 &#xbc0f; &#xadf8; &#xc774;&#xc804; &#xbc84;&#xc804;&#xc5d0;&#xc11c;&#xb294; <--
--      &#xc0ac;&#xc6a9;&#xc790; &#xb370;&#xc774;&#xd130;&#xb97c; &#xd558;&#xc704; <--
--      &#xc9d1;&#xd569;&#xc73c;&#xb85c; &#xc815;&#xb82c;&#xd558;&#xb294; <--
--      &#xc2dd;&#xc73c;&#xb85c; (&#xb300;&#xbd80;&#xbd84;&#xc758; &#xacbd;&#xc6b0;) <--
--      &#xb3d9;&#xc791;&#xd569;&#xb2c8;&#xb2e4;.
--      test_type&#xc758; &#xc54c;&#xd30c;&#xbcb3; &#xc21c;&#xc11c;&#xb85c; &#xac01; test_type <--
--      &#xc5d0; &#xb300;&#xd55c; &#xbc34;&#xb4dc;&#xb97c; &#xac00;&#xc9c4; <--
--      &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xcd9c;&#xb825;&#xd560; <--
--      &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
--      &#xb3d9;&#xbbb3c;&#xbcf4;&#xd638;&#xb860;&#xc790; &#xbd84;&#xb4e4;&#xaed8;: &#xc774; <--
--      &#xc608;&#xc2dc;&#xc5d0;&#xc11c; &#xc5b4;&#xb5a4; &#xc950; (mice)&#xb3c4; <--
--      &#xd76c;&#xc0dd;&#xb418;&#xc9c0; &#xc54a;&#xc558;&#xc2b5;&#xb2c8;&#xb2e4;.
SELECT
    mouse,
    ST_AddBand(NULL, array_agg(rast ORDER BY test_type), 1) As rast
FROM mice_studies
GROUP BY mouse;

```

예시: 새로운 DB 외부 밴드

```

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

참고

ST_BandMetaData, ST_BandPixelType, ST_MakeEmptyRaster, ST_MetaData, ST_NumBands, ST_Reclass

12.3.2 ST_AsRaster

ST_AsRaster — PostGIS 도형을 PostGIS 래스터로 변환&#xd569

Synopsis

raster **ST_AsRaster**(geometry geom, raster ref, text pixeltype, double precision value=1, double precision nodataval=0, boolean touched=false);

raster **ST_AsRaster**(geometry geom, raster ref, text[] pixeltype=ARRAY['8BUI'], double precision[] value=ARRAY[1], double

```

precision[] nodataval=ARRAY[0], boolean touched=false);
raster ST_AsRaster(geometry geom, double precision scalex, double precision scaley, double precision gridx, double precision gridy, text pixeltype, double precision value=1, double precision nodataval=0, double precision skewx=0, double precision skewy=0, boolean touched=false);
raster ST_AsRaster(geometry geom, double precision scalex, double precision scaley, double precision gridx=NULL, double precision gridy=NULL, text[] pixeltype=ARRAY['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);

```

설அ

```

PostGIS &#xb3c4;&#xd615;&#xc744; PostGIS &#xb798;&#xc2a4;&#xd130;&#xb85c; &#xbcc0;&#xd658;&#xd569;&#xb2c8;&#xb2e4;&#xc591;&#xd55c; &#xbcc0;&#xc885;&#xb4e4;&#xc744; &#xd1b5;&#xd574; &#xcd9c;&#xb825; &#xb798;&#xc2a4;&#xc815;&#xb82c; &#xbc29;&#xd5a5;&#xacfc; &#xd53d;&#xc140; &#xd06c;&#xae30;&#xb97c; &#xd06c;&#xac8c; &#xc138; &#xac00;&#xc9c0; &#xbc29;&#xbc95;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xc98;&#xc74c; &#xb450; &#xbcc0;&#xc885;&#xc73c;&#xb85c; &#xc774;&#xb8e8;&#xc5b4;&#xc9c4; &#xccab; &#xbc88;&#xbc29;&#xbc95;&#xc740; &#xc8fc;&#xc5b4;&#xc9c4; &#xcc38;&#xc870; &#xb798;&#xc2a4;&#xd130;&#xc640; &#xb3d9;&#xc815;&#xb82c; &#xbc29;&#xd5a5;(scalex, scaley &#xbcc0; gridx, gridy), &#xd53d;&#xc140; &#xc720;&#xd615;, NODATA &#xac12;&#xc744; &#xac00;&#xc9c4; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xc77c;&#xbc18;&#xc801;&#xc73c;&#xb85c;, &#xcc38;&#xc870; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xb2f4;&#xace0; &#xc788;&#xb294; &#xd14c;&#xc774;&#xbe14;&#xacfc; &#xb3c4;&#xd615;&#xc744; &#xb2f4;&#xace0; &#xc788;&#xb294; &#xd14c;&#xc774;&#xbe14;&#xc744; &#xacb0;&#xd569;&#xd558;&#xb294; &#xbc29;&#xbc95;&#xc73c;&#xb85c; &#xd574;&#xc38;&#xc870; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc785;&#xb825;&#xd569;&#xb2c8;&#xb2e4;.

&#xb124; &#xbcc0;&#xc885;&#xc73c;&#xb85c; &#xc774;&#xb8e8;&#xc5b4;&#xc9c4; &#xb450; &#xbc88;&#xc9f8; &#xbc29;&#xd53d;&#xc140; &#xd06c;&#xae30;(scalex & scaley &#xbcc0; skewx & skewy)&#xc758; &#xd30c;&#xb77c;&#xbbf8;&#xc81c;&#xacf5;, &#xc0ac;&#xc6a9;&#xc790;&#xac00; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xcc28;&#xc6d0;&#xc744; &#xc124;&#xc815;&#xd560; &#xc218; &#xc788;&#xb3c4;&#xb85d; &#xd569;&#xb2c8;&#xb2e4;. &#xacb0;&#xacfc; &#xb798;&#xc744; &#xc9c4; &#xb3c4;&#xd615;&#xc758; &#xb9de;&#xb3c4;&#xb85d; &#xc870;&#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xb300;&#xbd80;&#xbd84;&#xc758; &#xacbd;&#xc6b0;, &#xc815;&#xc218;&#xd615; scalex & scaley &#xc778;&#xc218;&#xb4e4;&#xc744; PostgreSQL&#xc774; &#xc62c;&#xb14;&#xb978; &#xbcc0;&#xc885; &#xc120;&#xd0dd;&#xd558;&#xb3c4;&#xb85d; &#xc774;&#xc911; &#xc815;&#xbcc0;&#xb3c4; &#xb370;&#xc774;&#xd130;&#xd615;&#xbcc0;&#xd658;&#xd574;&#xc57c;&#xb9cc; &#xd569;&#xb2c8;&#xb2e4;.

&#xb124; &#xbcc0;&#xc885;&#xc73c;&#xb85c; &#xc774;&#xb8e8;&#xc5b4;&#xc9c4; &#xc138; &#xbc88;&#xc9f8; &#xbc29;&#xb798;&#xc2a4;&#xd130;&#xc758; &#xcc28;&#xc6d0;(width & height)&#xc744; &#xc81c;&#xacf5;, &#xc0ac;&#xc6a9;&#xb798;&#xc2a4;&#xd130;&#xc758; &#xcc28;&#xc6d0;&#xc744; &#xc218;&#xc815;&#xd560; &#xc218; &#xc788;&#xb3c4;&#xd569;&#xb2c8;&#xb2e4;. &#xacb0;&#xacfc; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xd53d;&#xc140; &#xd06c;&#xae30; &#xd30c;&#xb77c;&#xbbf8;&#xd130;(scalex & scaley &#xbcc0; skewx & skewy)&#xac00; &#xb3c4;&#xd615;&#xc758; &#xb9de;&#xb3c4;&#xb85d; &#xc870;&#xc815;&#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.

```

```
&#xb4a4;&#xc758;&#xb450;&#xb2c29;&#xb3c95;&#xac00;&#xc6b4;&#xb370;&#xac01;&#xb2c29;&#xb3c95;&#xc758;&#xc98;&#xb450;&#xbcc0;&#xc885;&#xb4e4;&#xc740;&#xc0ac;&#xc6a9;&#xc790;&#xac00;&#xc815;&#xb82c;&#xadf8;&#xb9ac;&#xc784;&#xc758;&#xc758;&#xbaa8;&#xc11c;&#xb9ac;(gridx & gridy)&#xb97c;&#xc124;&#xc815;&#xd560;&#xc218;&#xc788;&#xb3c4;&#xb85d;&#xd569;&#xb2c8;&#xb2e4;. &#xb2e4;&#xb978;&#xb450;&#xbcc0;&#xc885;&#xb4e4;&#xc740;&#xc88c;&#xc0c1;&#xb2e8;(upperleftx & upperlefty))&#xc744;&#xc785;&#xb825;&#xbcb1b;&#xc2b5;&#xb2c8;&#xb2e4;&#xac01;&#xb2c29;&#xb3c95;&#xc758;&#xbcc0;&#xc885;&#xc740;&#xb2e8;&#xc77c;&#xb34;&#xb4dc;&#xb798;&#xc2a4;&#xb610;&#xb294;&#xbcf5;&#xc218;&#xb34;&#xb4dc;&#xb798;&#xc2a4;&#xd130;&#xb97c;&#xc0dd;&#xc131;&#xd560;&#xc218;&#xc788;&#xac8c;&#xd574;&#xc90d;&#xb2c8;&#xb2e4;. &#bcf5;&#xc218;&#xb34;&#xb4dc;&#xb798;&#xc2a4;&#xc0dd;&#xc131;&#xd558;&#xb824;&#xba74;. &#xc0ac;&#xc6a9;&#xc790;&#xac00;&#xd53d;&#xc140;&#xc720;&#xd615;&#xb30;&#xc5f4;(pixeltype[]),&#xcd08;&#xae30;&#xac12;&#xc758;&#xb30;&#xc5f4;(value)&#xadf8;&#xb9ac;&#xace0;NODATA &#xac12;&#xc758;&#xb30;&#xc5f4;(nodataval)&#xc744;&#xc900;&#xbe44;&#xd574;&#xc57c;&#xb9cc;&#xd569;&#xb2c8;&#xb2e4;. &#xc774;&#xb4e4;&#xc744;&#xc785;&#xb825;&#xd558;&#xc9c0;&#xc54a;&#xc73c;&#xba74;. &#xae30;&#xbcf8;&#xc801;&#xc73c;&#xb85c;&#xd53d;&#xc140;&#xc720;&#xd615;&#xc740; 8BUI,&#xcd08;&#xae30;&#xac11,NODATA &#xac12;&#xc740; 0&#xc774;&#xb429;&#xb2c8;&#xb2e4;.

&#xcd9c;&#xb825;&#xb798;&#xc2a4;&#xd130;&#xb294;&#xc18c;&#xc2a4;&#xb3c4;&#xd615;&#xacfc;&#xb3d9;&#xc77c;&#xacf5;&#xac04;&#xcc38;&#xc870;&#xc2dc;&#xc2a4;&#xd15c;&#xc744;&#xac00;&#xc9c0;&#xac8c;&#xb429;&#xb2c8;&#xc720;&#xc77c;&#xd55c;&#xc608;&#xc678;&#xb294;&#xcc38;&#xc870;&#xb798;&#xc2a4;&#xd130;&#xb97c;&#xbcb1b;&#xbcc0;&#xc885;&#xb4e4;&#xbfd0;&#xc785;&#xb2c8;&#xb2e4;. &#xc774;&#xb7f0;&#xacbd;&#xc6b0;&#xacb0;&#xacfc;&#xb798;&#xc2a4;&#xd130;&#xb294;&#xcc38;&#xc870;&#xb798;&#xc2a4;&#xd130;&#xc640;&#xb3d9;&#xc77c;&#xd55c;SRID&#xb97c;&#xac00;&#xc9c0;&#xac8c;&#xb429;&#xb2c8;&#xb2e4;.

&#xc120;&#xd0dd;&#xc801;&#xc778; touched &#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xc758;&#xae30;&#xbcf8;&#xac12;&#xc740;&#xc9d3;&#xc73c;&#xb85c; GDAL_ALL_TOUCHED &#xb798;&#xc2a4;&#xd130;&#xd654;&#xc635;&#xc158;&#xc5d0;&#xb9e4;&#xd551;&#xb418;&#xb294;&#xb370;. &#xc774;&#xc635;&#xc158;&#xc740;&#xb77c;&#xc778;&#xb610;&#xb294;&#xd3f4;&#xb9ac;&#xace4;&#xc5d0;&#xb2ff;&#xc740;&#xd53d;&#xc140;&#xc744;&#xc5c6;&#xc568;&#xac83;&#xc778;&#xb9d0;&#xc83;&#xc778;&#xc9c0;&#xacb0;&#xc815;&#xd569;&#xb2c8;&#xb2e4;. &#xb80c;&#xb354;&#xb9c1;&#xb41c;&#xb77c;&#xc778;&#xacbd;&#xb85c;&#xc0c1;&#xc5d0;&#xc788;&#xb294;&#xd53d;&#xc140;&#xb9cc;&#xc774;&#xc544;&#xadf8;&#xc911;&#xc2ec;&#xc810;&#xc774;&#xd3f4;&#xb9ac;&#xace4;&#xb0b4;&#xbd80;&#xc5d0;&#xc788;&#xb294;&#xd53d;&#xc140;&#xb610;&#xd55c;&#xb9d0;&#xc774;&#xc8e0;.

&#xc774;&#xd568;&#xc218;&#xb294; ST_AsPNG &#xbcf0f;&#xb2e4;&#xb978; ST_AsGDALRaster &#xacc4;&#xc5f4;&#xd568;&#xc218;&#xc640;&#xacb0;&#xd569;&#xd574;&#xc11c;&#xc0ac;&#xc6a9;&#xd560;&#xb54c;&#xb370;&#xc774;&#xc9c1;&#xc811;&#xac00;&#xc838;&#xc628;&#xb3c4;&#xd615;&#xc758; JPEG &#xb610;&#xb294; PNG&#xb97c;&#xb80c;&#xb370;&#xd2b9;&#xd788;&#xc720;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.

2.0.0 &#xb84;&#xc804;&#xbd80;&#xd130;&#xc0ac;&#xc6a9;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
GDAL 1.6.0 &#xc774;&#xc0c1;&#xb84;&#xc804;&#xc774;&#xd544;&#xc694;&#xd569;&#xb2c8;&#xb2e4;.
```

Note



아직 만곡 도형, TIN, 다면체
표면 등의 복잡 도형 유형을
렌더링할 수는 없지만, GDAL이
관련 기능을 지원하게 되면 할
수 있게 될 것입니다.

PNG



```
-- 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));
```



PostGIS

```
-- RGB - (118,154,118) - ↔
--
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]));
```

ST_BandPixelType, ST_Buffer, ST_GDALDrivers, ST_AsGDALRaster, ST_AsPNG, ST_AsJPEG, ST_SRID

12.3.3 ST_Band

ST_Band —

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=,);
```

Warning

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

 The `ST_Band` function will return an error if the `nbands` parameter is not a valid array of integers. For example, the following query will fail:

```
SELECT ST_Band(rast, '1,2,3');
-- ERROR: invalid input syntax for integer: "3"
-- HINT: You may need to add explicit casts to the input type.
```

2.0.0 The `ST_Band` function will return an error if the `nbands` parameter is not a valid array of integers.

Example

```
-- Create a dummy raster with 2 bands
CREATE TABLE dummy_rast (
  rid integer,
  rast raster,
  PRIMARY KEY (rid)
);
INSERT INTO dummy_rast (rid, rast) VALUES (1, ST_MakeRaster(2, 1));
INSERT INTO dummy_rast (rid, rast) VALUES (2, ST_MakeRaster(2, 2));

-- Query to select the first band of the first raster and the second band of the second raster
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

-- Query to select the first band of the first raster and the second band of the second raster
SELECT ST_NumBands(ST_Band(rast, '{2,3}':int[])) As num_bands
FROM dummy_rast WHERE rid=2;
```

```

num_bands
-----
2

-- &#xb34;&#xb4dc; 2 &#xb0f; 3&#xc744; &#xb18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. ↔
&#xb30;&#xc5f4;&#xc744; &#xc774;&#xc6a9;&#xd574; &#xb34;&#xb4dc;&#xb97c; ↔
&#xc815;&#xc758;&#xd569;&#xb2c8;&#xb2e4;.
SELECT ST_NumBands(ST_Band(rast, ARRAY[2,3])) As num_bands
FROM dummy_rast
WHERE rid=2;

```



원본(rast 열)



dupe_band



sing_band

```

-- &#xc6d0;&#xbcf8;&#xc758; &#xb450; &#xb88;&#xc9f8; &#xb34;&#xb4dc;&#xb97c; &#xd55c; ↔
&#xb88;; &#xccab; &#xb88;&#xc9f8; &#xb34;&#xb4dc;&#xb97c; &#xb450; &#xb88; ↔
&#xb18;&#xbcf5;&#xd574;&#xc11c; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xb97c; ↔
&#xc791;&#xc131;&#xd558;&#xace0;; &#xc138; &#xb88;&#xc9f8; &#xb34;&#xb4dc;&#xb85c; ↔
&#xb610; &#xb2e4;&#xb978; &#xb798;&#xc2a4;&#xd130;&#xb97c; ↔
&#xc791;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
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;

```

참고

ST_AddBand, ST_NumBands, ST_Reclass, Chapter 12

12.3.4 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);

ST_MakeEmptyRaster

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

ST_MakeEmptyCoverage

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 numbands	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	
22	33	1	1	8.25	-11	0	0	4326	↔
30.25	0	33	1	8.25	-11	0	0	4326	↔
38.5	0	33	1	8.25	-11	0	0	4326	↔
46.75	0	33	1	8.25	-11	0	0	4326	↔
22	0	22	1	8.25	-11	0	0	4326	↔
30.25	0	22	1	8.25	-11	0	0	4326	↔
38.5	0	22	1	8.25	-11	0	0	4326	↔
46.75	0	22	1	8.25	-11	0	0	4326	↔
22	0	11	1	8.25	-11	0	0	4326	↔
30.25	0	11	1	8.25	-11	0	0	4326	↔
38.5	0	11	1	8.25	-11	0	0	4326	↔
46.75	0	11	1	8.25	-11	0	0	4326	↔
22	0	0	1	8.25	-11	0	0	4326	↔
30.25	0	0	1	8.25	-11	0	0	4326	↔
38.5	0	0	1	8.25	-11	0	0	4326	↔
46.75	0	0	1	8.25	-11	0	0	4326	↔

ST_MakeEmptyRaster

ST_MakeEmptyRaster

12.3.5 ST_MakeEmptyRaster

ST_MakeEmptyRaster — `설정된 న원(너비 & 높이), 좌리 정된 X 볰 Y, 픽셀 크기, 회전(scalex, scaley, skewx & skewy) 그리정볰 쫀정న조 시스템(SRID)를 쫀진 텅 빈 (밴드쫀없는) 래스터를 볰환합니래스터를 입력할 쪽우, 동일한 크정렬 볰향 볰 SRID를 쫀진 새 래스터를 볰환합니다. SRID를 생략할 쪽우, 쫀న조 시스템을 0(unknown)으로 설정합니을`

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);
```

설명

설정된 న원(너비 & &#b192;이), 공간 (또&#xb29
월드) 좌표로 지리స조된 좌상단
X(upperleftx) 및 좌상단 Y(upperlefty), 픽셀 크기, 회전(sca
scaley, skewx & skewy) 그리고 공간 స조 시스템(SRID)&#x
가진 텅 빈 (&#bc34;드가 없는) 래스터를
&#bc18;환합니다.

```
&#xc9c0;&#xb09c;&#xbc84;&#xc804;&#xc740;&#xd53d;&#xc140;&#xd06c;&#xae30;(pixelsize)&#xb97c;&#xc124;&#xc815;&#xc704;&#xd55c;&#xb2e8;&#xc77c;&#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb9cc;&#xc0ac;&#xc6a9;&#xd588;&#xc2b5;&#xscalex&#xb294;&#xc774;&#xc778;&#xc218;&#xb85c;&#xc124;&#xc815;&#xb410;&#xace0,&scaley&#xb294;&#xc774;&#xc778;&#xc218;&#xc758;&#xc74c;&#xc758;&#xac12;&#xc73c;&#xb85c;&#xc124;&#xc815;&#xb410;&#xc2b5;&#xb2c8;&#xskewx&#xbc0f;skewy&#xb294;0&#xc73c;&#xb85c;&#xc124;&#xc815;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.
```

기존래스터를입력할경우,동일메타데이터설정을가진(밴드가없는)새래스터를반환합니다.

```
&#xc5b4;&#xb5a4; SRID&#xb3c4; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; &#xae30;&#xbcf8; &#xc785;&#xb2c8;&#xb2e4;. &#xd145; &#xbe48; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd55c; &#xb2e4;&#xc74c; &#xc0ac;&#xc6a9;&#xc790;&#xac00; &#xb3c4;&#xb4dc;&#xb97c; &#xcd94;&#xac00;&#xd558;&#xac70;&#xb3b8;&#xc9d1;&#xd558;&#xb824; &#xd560; &#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xb3c4;&#xb4dc;&#xc815;&#xc758;&#xd558;&#xb824;&#xba74; ST_AddBand &#xb97c;. &#xae30;&#xbcf8; &#xd53d;&#xc140; &#xac12;&#xc74c;&#xc124;&#xc815;&#xd558;&#xb824;&#xba74; ST_SetValue &#xb97c; &#xcc38;&#xc870;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

예시

```

INSERT INTO dummy_rast(rid,rast)
VALUES(3, ST_MakeEmptyRaster( 100, 100, 0.0005, 0.0005, 1, 1, 0, 0, 4326) );

-- &#xae30;&#xc874; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0c8;  ←
&#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc704;&#xd55c;  ←
&#xd15c;&#xd50c;&#xb9bf;&#xc73c;&#xb85c; &#xc0ac;&#xc6a9;
INSERT INTO dummy_rast(rid,rast)
SELECT 4, ST_MakeEmptyRaster(rast)
FROM dummy_rast WHERE rid = 3;

-- &#xbc29;&#xae08; &#xcd94;&#xac00;&#xd55c; &#xb798;&#xc2a4;&#xd130;&#xc758;  ←
&#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130;&#xb97c; &#xcd9c;&#xb825;

```

```
SELECT rid, (md).*
FROM (SELECT rid, ST_MetaData(rast) As md
      FROM dummy_rast
      WHERE rid IN(3,4)) As foo;
```

```
-- &#xcd9c;&#xb825; --
```

rid	upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	←
3	0.0005	0.0005	100	100	1	1	0	0	0	←
	4326	0								
4	0.0005	0.0005	100	100	1	1	0	0	0	←
	4326	0								

참고

ST_AddBand, ST_MetaData, ST_ScaleX, ST_ScaleY, ST_SetValue, ST_SkewX, , ST_SkewY

12.3.6 ST_Tile

ST_Tile — 출력 래스터에 원하는 차원을 바탕으로 입력 래스터를 분할해서 나온 래스터들의 집합을 반환합니다

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);

설명

출력 래스터에 원하는 차원을 바입력 래스터를 분할해서 나온 래스터들의 집합을 반환합니다

padwithnodata = FALSE일 경우, 래스터 우단 및 하쫀장자리 타일들이 나머지 타일다른 차원을 가지고 있을지도 &#baa8;일 경우, &#baa8;든 타일들이 동쫀장자리 타일들이 NODATA 값으로 패딩(padding)됐을 가능성이 있습니다. 래스터 &#bc34;드(지정된 NODATA 값(들)을 가지지 않은 경을 설정해서 NODATA 값을 지정할 수 있습니다

Note



입력 래스터의 지정된 &#bc34;드󊰀 DB 외부 &#bc34;드일 경우, 출력 래스터의 해당 &#bc34;드 또한 DB 외부 &#bc34;드일 것입니다

2.1.0

```
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)

```

ST_Union, ST_Retile

12.3.7 ST_Retile

ST_Retile — Retile a raster into a grid of tiles of the specified size. The tiles are created by dividing the raster into a grid of tiles of the specified size. The tiles are created by dividing the raster into a grid of tiles of the specified size.

Synopsis

setof raster **ST_Retile**(regclass tab, name col, geometry ext, float8 sfx, float8 sfy, int tw, int th, text algo='NearestNeighbor');

ST_Retile

ST_Retile — Retile a raster into a grid of tiles of the specified size. The tiles are created by dividing the raster into a grid of tiles of the specified size. The tiles are created by dividing the raster into a grid of tiles of the specified size.

알고리즘옵션에는 'NearestNeighbor', 'Bilinear', 'Cubic', 'Cubic-Spline', 그리고 'Lanczos'가있습니다. 자세한내용은 **GDAL Warp resampling methods**를󃰸조하십시오.

2.2.0 버전부터사용할수있습니다.

స고

ST_CreateOverview

12.3.8 ST_FromGDALRaster

ST_FromGDALRaster — [%c9c0; %c6d0](#); GDAL [%b798; %c2a4; %d130; %d30c; %c77c; %b85c; %bd80; %d130; %b798; %c2a4; %d130; %b97c; %bc18; %d658; %d569; %b2c8; %b2e4.](#)

Synopsis

```
raster ST_FromGDALRaster(bytea gdaldata, integer srid=NULL);
```

설명

```
&#xc9c0;&#xc6d0; GDAL &#xb798;&#xc2a4;&#xd130; &#xd30c;&#xc77c;&#xb85c;&#xbd80;&#xd130; &#xb798;&#xc2a4;&#xd130; &#xb798;&#xc2a4;&#xd130; &#xd30c;&#xc77c;&#xc758; &#xb0b4;&#xc6a9;&#xc744; &#xb2f4;&#xace0; &#xc788;&#xc5b4;&#xd569;&#xb2c8;&#xb2e4;.
```

```

srid &#xac00; NULL&#xc77c; &#xacbd;&#xc6b0;, &#xc774; &#xd568;&#xc218;&#xb294; GDAL &#xb798;&#xc2a4;&#xd130;&
&#xc790;&#xb3d9;&#xc801;&#xc73c;&#xb85c; SRID&#xb97c; &#xd560;&#xb2f9;&#xd558;&#xb824; &#xd560; &#xac83;&#xc7
srid &#xb97c; &#xc124;&#xc815;&#xd55c; &#xacbd;&#xc6b0;, &#xd574;&#xb2f9; &#xac12;&#xc774; &#xc790;&#xb3d9;&#xc
&#xd560;&#xb2f9;&#xb41c; &#xc5b4;&#xb5a4; SRID&#xbcf4;&#xb2e4; &#xc6b0;&#xc120;&#xd560; &#xac83;&#xc785;&#xb2
2.1.0 &#xbc84;&#xc804;&#xbd80;&#xd130; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.

```

예시

```
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_Metadadata(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)


```
scalex
skewy
skewx
scaley
upperleftx + scalex*0.5
upperlefty + scaley*0.5
```

esri_ref | gdal_ref

```
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

ST_SetGeoReference, ST_ScaleX, ST_ScaleY

ST_SetGeoReference, ST_ScaleX, ST_ScaleY

12.4.2 ST_Height

ST_Height — Returns the height of the raster. The height is the number of rows in the raster. The height is always a positive integer.

Synopsis

integer **ST_Height**(raster rast);

esri_ref | gdal_ref

esri_ref | gdal_ref

esri_ref | gdal_ref

```
SELECT rid, ST_Height(rast) As rastheight
FROM dummy_rast;
```

rid	rastheight
1	20
2	5

ST_Width

ST_Width

12.4.3 ST_IsEmpty

ST_IsEmpty — Returns true if the raster is empty (width = 0, height = 0). The height is the number of rows in the raster. The height is always a positive integer.

Synopsis

boolean **ST_IsEmpty**(raster rast);

설명

래스터가 비어 있을 경우 (width = 0, height = 0)
참을 반환합니다. 그렇지 않을 경౰짓을 반환합니다.

2.0.0 버전부터 사용할 수 있습니다.

예시

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(100, 100, 0, 0, 0, 0, 0, 0, 0))
st_isempty |
-----+
f          |
```

```
SELECT ST_IsEmpty(ST_MakeEmptyRaster(0, 0, 0, 0, 0, 0, 0, 0, 0))
st_isempty |
-----+
t          |
```

참고

ST_HasNoBand

12.4.4 ST_MemSize

ST_MemSize — 래스터가 차지하는 공간의 용량을 (바이트 단위로) 반환합니다.

Synopsis

integer **ST_MemSize**(raster rast);

설명

래스터가 차지하는 공간의 용량을 (바이트 단위로) 반환합니다.

이 함수는 PostgreSQL 내장 함수들인 pg_column_size, pg_size_pretty, pg_relation_size, pg_total_relation_size를 멋지게 보완합니다.

Note

테이블의 바이트 용량을
 출력하는 pg_relation_size가 ST_MemSize보다
 적은 바이트 용량을 반환할
 수도 있습니다. 왜냐하면 pg_relation_size
 함수는 TOAST 테이블에 저장된
 대용량 도형들 및 TOAST가 적용된
 테이블이 분담하는 용량을
 추가하지 않기 때문입니다.
 pg_column_size 함수는 압축된 용량을
 반환하기 때문에 더 적은
 용량을 반환할 수도 있습니다.
 pg_total_relation_size 함수는 테이블. TOAST가
 적용된 테이블 및 인덱스를
 포함합니다.

2.2.0 버전부터 사용할 수 있습니다.

예시

```

SELECT ST_MemSize(ST_AsRaster(ST_Buffer(ST_Point(1,5),10,1000),150, 150, '8BUI')) As ↵
    rast_mem;

    rast_mem
    -----
    22568
  
```

참고

12.4.5 ST_MetaData

ST_MetaData — 래스터 객체의 픽셀 크기. 회전(skew),
 좌상단. 좌하단 등과 같은 기본 메&#
 반환합니다. 반환되는 열들은 다&#
 같습니다. upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands

Synopsis

record **ST_MetaData**(raster rast);

설명

래스터 객체의 픽셀 크기. 회전(skew),
 좌상단. 좌하단 등과 같은 기본 메&#
 반환합니다. 반환되는 열들은 다&#
 같습니다. upperleftx | upperlefty | width | height | scalex | scaley | skewx | skewy | srid | numbands

예시

```

SELECT rid, (foo.md).*
FROM (SELECT rid, ST_MetaData(rast) As md
FROM dummy_rast) As foo;
  
```

rid	upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	←
numbands										
1	0.5	0.5	10	20	2	3			0	←
2	3427927.75	5793244	5	5	0.05	-0.05			0	←

ST_BandMetaData

ST_NumBands

12.4.6 ST_NumBands

ST_NumBands — Returns the number of bands in a raster. The function returns an integer value.

Synopsis

integer **ST_NumBands**(raster rast);

ST_NumBands

ST_NumBands returns the number of bands in a raster. The function returns an integer value.

ST_NumBands

```
SELECT rid, ST_NumBands(rast) As numbands
FROM dummy_rast;
```

rid	numbands
1	0
2	3

ST_Value

ST_Value

12.4.7 ST_PixelHeight

ST_PixelHeight — Returns the pixel height of a raster. The function returns a double precision value.

Synopsis

double precision **ST_PixelHeight**(raster rast);

ST_PixelWidth

ST_PixelWidth — Returns the pixel width of a raster. The pixel width is the distance between the centers of two adjacent pixels in the x-direction. The pixel width is measured in the units of the raster's spatial reference system. The pixel width is a scalar value, and is the same for all pixels in the raster.

ST_PixelWidth (raster) Returns the pixel width of a raster. The pixel width is the distance between the centers of two adjacent pixels in the x-direction. The pixel width is measured in the units of the raster's spatial reference system. The pixel width is a scalar value, and is the same for all pixels in the raster.

ST_PixelWidth (raster) Returns the pixel width of a raster. The pixel width is the distance between the centers of two adjacent pixels in the x-direction. The pixel width is measured in the units of the raster's spatial reference system. The pixel width is a scalar value, and is the same for all pixels in the raster.

```
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

ST_PixelWidth (raster) Returns the pixel width of a raster. The pixel width is the distance between the centers of two adjacent pixels in the x-direction. The pixel width is measured in the units of the raster's spatial reference system. The pixel width is a scalar value, and is the same for all pixels in the raster.

```
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

ST_PixelWidth

ST_PixelWidth, **ST_ScaleX**, **ST_ScaleY**, **ST_SkewX**, **ST_SkewY**

12.4.8 ST_PixelWidth

ST_PixelWidth — Returns the pixel width of a raster. The pixel width is the distance between the centers of two adjacent pixels in the x-direction. The pixel width is measured in the units of the raster's spatial reference system. The pixel width is a scalar value, and is the same for all pixels in the raster.

Synopsis

double precision **ST_PixelWidth**(raster rast);

ST_PixelHeight

ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

12.4.9 ST_ScaleX

ST_ScaleX — Returns the X pixel scale of a raster. The X pixel scale is the distance in the X direction between the centers of two adjacent pixels. The X pixel scale is measured in the units of the raster's coordinate system.

Synopsis

```
float8 ST_ScaleX(raster rast);
```

Parameters

rast is a **raster** value. The X pixel scale is measured in the units of the raster's coordinate system. The X pixel scale is measured in the units of the raster's coordinate system. The X pixel scale is measured in the units of the raster's coordinate system.

Examples

```
SELECT rid, ST_ScaleX(rast) As rastpixwidth
FROM dummy_rast;
```

rid	rastpixwidth
1	2
2	0.05

ST_PixelWidth

ST_Width

12.4.10 ST_ScaleY

ST_ScaleY — Returns the Y pixel scale of a raster. The Y pixel scale is the distance in the Y direction between the centers of two adjacent pixels. The Y pixel scale is measured in the units of the raster's coordinate system.

Synopsis

```
float8 ST_ScaleY(raster rast);
```

Parameters

rast is a **raster** value. The Y pixel scale is measured in the units of the raster's coordinate system. The Y pixel scale is measured in the units of the raster's coordinate system. The Y pixel scale is measured in the units of the raster's coordinate system.

ST_Height

```
SELECT rid, ST_ScaleY(rast) As rastpixheight
FROM dummy_rast;
```

rid	rastpixheight
1	3
2	-0.05

ST_RasterToWorldCoord

ST_Height

12.4.11 ST_RasterToWorldCoord

ST_RasterToWorldCoord — Returns the world coordinates (X, Y) of a pixel in a raster. The raster is identified by its rid. The pixel is identified by its x and y coordinates. The function returns a record with two columns: X (float) and Y (float).

Synopsis

record **ST_RasterToWorldCoord**(raster rast, integer xcolumn, integer yrow);

ST_RasterToWorldCoord

ST_RasterToWorldCoord — Returns the world coordinates (X, Y) of a pixel in a raster. The raster is identified by its rid. The pixel is identified by its x and y coordinates. The function returns a record with two columns: X (float) and Y (float).

2.1.0 Returns the world coordinates (X, Y) of a pixel in a raster. The raster is identified by its rid. The pixel is identified by its x and y coordinates. The function returns a record with two columns: X (float) and Y (float).

ST_Height

```
-- Returns the world coordinates (X, Y) of a pixel in a 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

```
-- &#xd68c;&#xc804;&#xd55c; &#xb798;&#xc2a4;&#xd130;
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
```

참고

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SetSkew

12.4.12 ST_RasterToWorldCoordX

ST_RasterToWorldCoordX — 열과 행에 있는 래스터의 좌상단의 기하학적 X 좌표를 반환열과 행의 번호는 1부터 시작합니의

Synopsis

float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn);
float8 **ST_RasterToWorldCoordX**(raster rast, integer xcolumn, integer yrow);

설명

래스터 열과 행의 좌상단 X 좌표를 지리참조된 래스터의 기하 단위의 반환합니다. 열과 행의 번호는 1부터 시작하지만 두 파라미터 가운데 하나라도 음수 또는 래스터 열의 개수보다 큰 숫자를 입력할 경우, 회전 및 픽셀 크기가 선택된 래스의 동일하다고 가정하고 래스터 파의 왼쪽 또는 오른쪽 바깥에 있는 좌의 반환할 것입니다.

Note



회전하지 않은 래스터의 경우, X열만 입력해도 충분합니다. 회전한 래스터의 경우, 지리참조된 좌표는 ST_ScaleX, ST_SkewX, 행 및 열의 함수입니다. 회전한 래스터에 대해 X열만 입력할 경우 오류가 발생할 것입니다.

2.1.0; ST_RasterToWorldCoordY(rast, 1) As y1coord,
ST_RasterToWorldCoordY(rast, 2) As y2coord,
ST_ScaleY(rast) As pixelx
FROM dummy_rast;

ST_RasterToWorldCoordY

```
-- ST_RasterToWorldCoordY(rast, 1) As y1coord,
-- ST_RasterToWorldCoordY(rast, 2) As y2coord,
-- ST_ScaleY(rast) As pixelx
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

```
-- ST_RasterToWorldCoordY(rast, 1) As y1coord,
-- ST_RasterToWorldCoordY(rast, 2) As y2coord,
-- ST_ScaleY(rast) As pixelx
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

ST_SetSkew

ST_ScaleX, ST_RasterToWorldCoordY, ST_SetSkew, ST_SkewX

12.4.13 ST_RasterToWorldCoordY

ST_RasterToWorldCoordY — Returns the world coordinate of the pixel at the given row and column of the raster. The function takes a raster and two integers as input: the row and column of the pixel. The function returns a float8 value representing the world coordinate of the pixel.

Synopsis

```
float8 ST_RasterToWorldCoordY(raster rast, integer yrow);
float8 ST_RasterToWorldCoordY(raster rast, integer xcolumn, integer yrow);
```

ST_RasterToWorldCoordY

ST_RasterToWorldCoordY(rast, yrow) — Returns the world coordinate of the pixel at the given row and column of the raster. The function takes a raster and two integers as input: the row and column of the pixel. The function returns a float8 value representing the world coordinate of the pixel.

동일하다고 가정하고 래스터 타왼쪽또는 오른쪽 바깥에 있는 좌벸환할 것입니다.

Note



회전하지 않은 래스터의 경우
Y열만 입력해도 충분합니다.
회전한 래스터의 경우,
지리참조된 좌표는 ST_ScaleY, ST_SkewY, 행
및 열의 함수입니다. 회전한
래스터에 대해 Y행만 입력할
경우 오류가 발생할 것입니다.

```
&#x0000;&#xbcc0;&#xacbd;&#xc0ac;&#xd56d;; 2.1.0 &#xbbf8;&#xb9cc;&#xbc84;&#xc804;&#xc5d0;&#xc11c;&#xb294; ST_Raster2World  
&#xba85;&#xce6d;&#xc774;&#xc5c8;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시

```
-- &#xd68c;&#xc804;&#xd558;&#xc9c0; &#xc54a;&#xc740; &#xb798;&#xc2a4;&#xd130;&#xc77c;  ←
    &#xacbd;&#xc6b0; &#xd589;&#xb9cc; &#xc785;&#xb825;&#xd574;&#xb3c4;  ←
    &#xcda9;&#xbd84;&#xd569;&#xb2c8;&#xb2e4; .
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

```
-- &#xc7ac;&#xbbf8;&#xc0bc;&#xc544; &#xb798;&#xc2a4;&#xd130;&#xb97c;  ←
    &#xd68c;&#xc804;&#xc2dc;&#xcflc;&#xbd05;&#xc2dc;&#xb2e4;.
SELECT rid, ST_RasterToWorldCoordY(rast,1,1) As ylcoord,
        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	ylcoord	y2coord	pixely
1	0.5	107	3
2	5793244	5793344.4	-0.05

స&#xace0:

ST_ScaleY, ST_RasterToWorldCoordX, ST_SetSkew, ST_SkewY

12.4.14 ST Rotation

ST_Rotation — $\times 798.24^\circ; 130.758^\circ; 68.804^\circ; 0.744^\circ; 77^\circ; 514.548^\circ; 73.18^\circ; 658.569^\circ; 28.24^\circ$.

Synopsis

float8 **ST_Rotation**(raster rast);

ST_Rotation

ST_Rotation (rast, scale, skew) — Returns the rotation of the input raster in degrees. The rotation is calculated as the angle between the horizontal axis and the major axis of the ellipse that best fits the data. The rotation is measured in degrees, ranging from -90 to 90. The input raster must be a float8 raster. The output is a float8 value.

ST_Rotation

```
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

ST_Rotation

ST_Rotation, **ST_SetScale**, **ST_SetSkew**

12.4.15 ST_SkewX

ST_SkewX (rast) — Returns the skew of the input raster. The skew is calculated as the ratio of the horizontal axis to the vertical axis of the ellipse that best fits the data. The skew is measured as a float8 value, ranging from -1 to 1. The input raster must be a float8 raster. The output is a float8 value.

Synopsis

float8 **ST_SkewX**(raster rast);

ST_SkewX

ST_SkewX (rast) — Returns the skew of the input raster. The skew is calculated as the ratio of the horizontal axis to the vertical axis of the ellipse that best fits the data. The skew is measured as a float8 value, ranging from -1 to 1. The input raster must be a float8 raster. The output is a float8 value.

ST_SkewX

```
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

```

ST_GeoReference, ST_SkewY, ST_SetSkew

ST_GeoReference, ST_SkewY, ST_SetSkew

12.4.16 ST_SkewY

ST_SkewY — Skews the geometry by the specified skew factor. The skew factor is a float8 value representing the skew factor. The skew factor is calculated as the ratio of the skew factor to the original geometry. The skew factor is calculated as the ratio of the skew factor to the original geometry.

Synopsis

```
float8 ST_SkewY(raster rast);
```

ST_SkewY

ST_SkewY — Skews the geometry by the specified skew factor. The skew factor is a float8 value representing the skew factor. The skew factor is calculated as the ratio of the skew factor to the original geometry. The skew factor is calculated as the ratio of the skew factor to the original geometry.

ST_SkewY

```

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

ST_GeoReference, ST_SkewX, ST_SetSkew

ST_GeoReference, ST_SkewX, ST_SetSkew

12.4.17 ST_SRID

ST_SRID — Returns the SRID of the geometry. The SRID is a smallint value representing the SRID. The SRID is calculated as the ratio of the SRID to the original geometry. The SRID is calculated as the ratio of the SRID to the original geometry.

Synopsis

integer **ST_SRID**(raster rast);

설명

spatial_ref_sys 테이블에 정의되어 있는., 래스터/도형의 SRID가 객체의 공간 참조 식별자를 반환합니다.



Note

PostGIS 2.0 버전부터, 지리참조되지 않은 래스터/도형의 SRID가 이전 버전의 -1 대신 0으로 바뀌었습니다.

예시

```
SELECT ST_SRID(rast) As srid
FROM dummy_rast WHERE rid=1;
```

```
srid
-----
0
```

참고

Section [4.5, ST_SRID](#)

12.4.18 ST_Summary

ST_Summary — 래스터의 내용을 요약한 텍스트로 로 환합니다.

Synopsis

text **ST_Summary**(raster rast);

설명

래스터의 내용을 요약한 텍스트로 로 환합니다.

2.1.0 버전부터 사용할 수 있습니다.

예시

```
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)

```

ST_MetaData

ST_MetaData, **ST_BandMetaData**, **ST_Summary** **ST_Extent**

12.4.19 ST_UpperLeftX

ST_UpperLeftX — Returns the X coordinate of the upper-left corner of the raster.

Synopsis

float8 **ST_UpperLeftX**(raster rast);

ST_UpperLeftX

Returns the X coordinate of the upper-left corner of the raster.

ST_UpperLeftX

```

SELECT rid, ST_UpperLeftX(rast) As ulx
FROM dummy_rast;

```

rid	ulx
1	0.5
2	3427927.75

ST_UpperLeftY

ST_UpperLeftY, **ST_GeoReference**, **Box3D**

12.4.20 ST_UpperLeftY

ST_UpperLeftY — Returns the Y coordinate of the upper-left corner of the raster.

Synopsis

```
float8 ST_UpperLeftY(raster rast);
```

Parameters

`rast` — raster to be processed.
`Y` — Y coordinate of the upper-left pixel of the raster.

Examples

```
SELECT rid, ST_UpperLeftY(rast) As uly
FROM dummy_rast;
```

rid	uly
1	0.5
2	5793244

See also

[ST_UpperLeftX](#), [ST_GeoReference](#), [Box3D](#)

12.4.21 ST_Width

`ST_Width` — Returns the width of the raster in pixels.

Synopsis

```
integer ST_Width(raster rast);
```

Parameters

`rast` — raster to be processed.

Examples

```
SELECT ST_Width(rast) As rastwidth
FROM dummy_rast WHERE rid=1;
```

rastwidth
10

See also

[ST_Height](#)

ST_WorldToRasterCoordX — 포인트 도형(pt)의 래스터
안의 열또는 래스터의 월드 공간
참조 시스템 단위로 표현된 X, Y 월&#xb4d
좌표(xw, yw)를 반환합니다.

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);

설명

포인트 도형(pt)의 래스터 안의 열
 또는 X, Y 월드 좌표(xw, yw)를 반환합니다
 포인트가 필요합니다(또는 래스&#xb
 기울어진 경우 xw 및 yw 월드 좌표가
 두 개 ન두 필요합니다). 래스터가
 기울어지지 않은 경우 xw 좌표만으&
 쵉분합니다. 월드 좌표란 래스터&#xb
 공간 참조 좌표 시스템의 좌표를
 말합니다.

변경 사항: 2.1.0 &#bbbf8;만 &#bc84;전에서는 ST_World2Raster
 명칭이었습니다.

예시

```
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

참고

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

12.4.24 ST_WorldToRasterCoordY

ST_WorldToRasterCoordY — 포인트 도형(pt)의 래스터
 안의 행 또는 래스터의 월드 공간
 참조 시스템 단위로 표현된 X, Y 월&#xb4d
 좌표(xw, yw)를 반환합니다.

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);

ST_RasterToWorldCoordY, ST_SRID

ST_RasterToWorldCoordY(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;

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

```
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

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID

12.5 ST_RasterToWorldCoordX, ST_RasterToWorldCoordY, ST_SRID**12.5.1 ST_BandMetaData**

ST_BandMetaData — ST_BandMetaData(raster rast, integer band=1);
 ST_BandMetaData(raster rast, integer[] band);

Synopsis

- (1) record **ST_BandMetaData**(raster rast, integer band=1);
- (2) record **ST_BandMetaData**(raster rast, integer[] band);

ST_BandMetaData

Returns basic meta data about a raster band. Columns returned: pixeltype, nodatavalue, isoutdb, path, outdbbandnum, filesize, filetimestamp.

**Note**

If `isoutdb` is `False`, `path`, `outdbbandnum`, `filesize` and `filetimestamp` are `NULL`. If `outdb` access is disabled, `filesize` and `filetimestamp` will also be `NULL`.

**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.

Example 1

```
-- Create a table with two columns: longitude and latitude
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

Example 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	4B32	0	0	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif	1	1048576	1544123456.789
3	4B32	0	0	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif	3	1048576	1544123456.789
2	4B32	0	0	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif	2	1048576	1544123456.789

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	

󌰸고

ST_MetaData, ST_BandPixelType

12.5.2 ST_BandNoDataValue

ST_BandNoDataValue — 입력 밴드에서 NODATA를 나타&#xb0
값을 반환합니다. 어떤 밴드도 지&#xc
않을 경우 밴드 1로 가정합니다.

Synopsis

double precision **ST_BandNoDataValue**(raster rast, integer bandnum=1);

설명

밴드에서 NODATA를 나타내는 값을 반&#

예시

```
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
```

స고

ST_NumBands

12.5.3 ST BandsNoData

ST_BandIsNoData — 썌드가 NODATA 값만으로 ౄ워져 있을 경우 స을 ଘ환합니다.

Synopsis

```
boolean ST_BandIsNoData(raster rast, integer band, boolean forceChecking=true);
```

boolean **ST_BandIsNoData**(raster rast, boolean forceChecking=true);

설명

밴드가 NODATA 값만으로 채워져 있을
 경우 참을 반환합니다. 밴드를 지않으면 밴드 1로 가정합니다. 마지인수가 TRUE일 경우. 밴드의 픽셀 전을 하나하나 확인합니다. 그렇지 않을 경우. 이 함수는 밴드에 대한 isnodata 플설을 반환할 뿐입니다. 따로 설정않을 경우. 해당 파라미터의 기본입니다.

2.0.0 버전부터 사용할 수 있습니다.

Note

플래그가 지저분할 (즉
 마지막 파라미터에 TRUE를
 설정하고 설정하지 않고에
 따라 결과가 달라질) 경우,
 ST_SetBandNodataValue()의 마지막 인수로 TRUE를
 입력하거나. 또는 ST_SetBandIsNodata()를
 이용해서 해당 플래그를
 참으로 설정하도록 래스터를
 업데이트헤야 합니다. **ST_SetBandIsNoData**
 를 참조하십시오.

예시

```
-- &#xb798;&#xc2a4;&#xd130; &#xc5f4; &#xd55c; &#xac1c;&#xb97c; &#xac00;&#xc9c4;  ←
  &#xac00;&#xc9dc; &#xd14c;&#xc774;&#xbel4;&#xc744; &#xc0dd;&#xc131;
create table dummy_rast (rid integer, rast raster);

-- &#xbc34;&#xb4dc; &#xb450; &#xac1c;&#xc640; &#xd53d;&#xc140;/&#xbc34;&#xb4dc;  ←
  &#xd558;&#xb098;&#xb97c; &#xac00;&#xc9c4; &#xb798;&#xc2a4;&#xd130;&#xb97c;  ←
  &#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;. &#xccab; &#xbc88;&#xc9f8;  ←
  &#bc34;&#xb4dc;&#xc758; nodatavalue = pixel value = 3&#xc785;&#xb2c8;&#xb2e4;.
-- &#xb450; &#xbc88;&#xc9f8; &#bc34;&#xb4dc;&#xc758; nodatavalue = 13, pixel value = 4  ←
  &#xc785;&#xb2c8;&#xb2e4;.
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

```

ST_BandNoDataValue

ST_NumBands, **ST_SetBandNoDataValue**, **ST_SetBandIsNoData**

12.5.4 ST_BandPath

ST_BandPath — Returns the path of the specified band in the raster. The path is a 2D array of double precision floating point numbers representing the x and y coordinates of the pixels in the band. The path is returned as a text string in the following format: `(x1 y1 x2 y2 ...)`. The path is returned as a text string in the following format: `(x1 y1 x2 y2 ...)`. The path is returned as a text string in the following format: `(x1 y1 x2 y2 ...)`.

Synopsis

text **ST_BandPath**(raster rast, integer bandnum=1);

ST_BandPath

bandnum is the band number to return. The path is a 2D array of double precision floating point numbers representing the x and y coordinates of the pixels in the band. The path is returned as a text string in the following format: `(x1 y1 x2 y2 ...)`. The path is returned as a text string in the following format: `(x1 y1 x2 y2 ...)`. The path is returned as a text string in the following format: `(x1 y1 x2 y2 ...)`.

ST_BandPath

󌰸고

12.5.5 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);

설명

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

예시

```
SELECT ST_BandFileSize(rast,1) FROM dummy_rast WHERE rid = 1;
```

```
st_bandfilesize
-----
240574
```

12.5.6 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);

설명

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

예시

```
SELECT ST_BandFileTimestamp(rast,1) FROM dummy_rast WHERE rid = 1;
```

```
st_bandfiletimestamp
-----
1521807257
```

12.5.7 ST_BandPixelType

ST_BandPixelType — Returns name describing data type and size of values stored in each cell of given band.
 bandnum: integer, band number (1 to ST_NumBands(rast)).

Synopsis

ST_BandPixelType(raster rast, integer bandnum=1);

Returns

Returns name describing data type and size of values stored in each cell of given band.

1B - 1 byte
 2B - 2 bytes
 4B - 4 bytes
 8B - 8 bytes
 16B - 16 bytes
 32B - 32 bytes
 64B - 64 bytes

- 1B - 1 byte
- 2B - 2 bytes
- 4B - 4 bytes
- 8B - 8 bytes
- 16B - 16 bytes
- 32B - 32 bytes
- 64B - 64 bytes

Examples

```
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
```

See Also

[ST_NumBands](#)

12.5.8 ST_PixelOfValue

ST_PixelOfValue — Returns the pixel value of the raster at the given location.
 rast: raster, the raster to sample.
 x: integer, the x coordinate of the pixel to sample.
 y: integer, the y coordinate of the pixel to sample.

Synopsis

integer **ST_MemSize**(raster rast);

설명

래스터 객체 내부에 있는 밴드들개수를 반환합니다.

예시

```
SELECT ST_MinPossibleValue('16BSI');

 st_minpossiblevalue
-----
                -32768

SELECT ST_MinPossibleValue('8BUI');

 st_minpossiblevalue
-----
                   0
```

참고

ST_BandPixelType

12.5.9 ST_HasNoBand

ST_HasNoBand — 입력된 밴드 번호에 밴드가 없을 경우 참을 반환합니다. 밴드 번호를 설정하지 않을 경우, 밴드 1로 가정합니다.

Synopsis

boolean **ST_HasNoBand**(raster rast, integer bandnum=1);

설명

입력된 밴드 번호에 밴드가 없을 경우 참을 반환합니다. 밴드 번호설정하지 않을 경우, 밴드 1로 가정합니다.

2.0.0 버전부터 사용할 수 있습니다.

예시

```
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

```
&#xc38;&#xace0;
```

[ST_NumBands](#)

12.6 [ST_PixelAsPolygon](#)

12.6.1 ST_PixelAsPolygon

ST_PixelAsPolygon — Convert a pixel from a raster to a polygon geometry. The pixel is identified by its row and column indices. The raster must have a valid SRID.

Synopsis

geometry **ST_PixelAsPolygon**(raster rast, integer columnx, integer rowy);

```
&#xc124;&#xba85;
```

```
&#xd2b9;&#xc815; &#xd589; &#xbc0f; &#xc5f4;&#xc5d0; &#xb300;&#xd55c; &#xd53d;&#xc140;&#xc758; &#xacbd;&#xacc4;&#xb97c; &#xc774;&#xb8e8;&#xb294; &#xd3f4;&#xb9ac;&#xace4; &#xb3c4;&#xd615;&#xc744; &#bc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;
```

2.0.0 [ST_PixelAsPolygon](#) (raster rast, integer columnx, integer rowy);

```
&#xc608;&#xc2dc;
```

```
-- &#xb798;&#xc2a4;&#xd130; &#xd53d;&#xc140; &#xd3f4;&#xb9ac;&#xace4; &#xd68d;&#xb4dd;
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,...

```
&#xc38;&#xace0;
```

[ST_DumpAsPolygons](#), [ST_PixelAsPolygons](#), [ST_PixelAsPoint](#), [ST_PixelAsPoints](#), [ST_PixelAsCentroid](#), [ST_PixelAsCentroids](#), [ST_Intersection](#), [ST_AsText](#)

12.6.2 ST_PixelAsPolygons

ST_PixelAsPolygons — [#xb798;](#)[#xc2a4;](#)[#xd130;](#) [#xbc34;](#)[#xb4dc;](#)[#xc758;](#) [#xaa8;](#)[#xb4e0;](#) [#xd53d;](#)[#xc140;](#)[#xc758;](#) [#xacbd;](#)[#xacc4;](#)[#xb97c;](#) [#xc774;](#)[#xb8e8;](#)[#xb294;](#) [#xd3f4;](#)[#xb9ac;](#)[#xace4;](#) [#xb3c4;](#)[#xd615;](#)[#xc744;](#) [#xac01;](#) [#xd53d;](#)[#xc140;](#)[#xc758;](#) [#xac12;](#)[#xacfc;](#) X, Y [#xb798;](#)[#xc2a4;](#)[#xd130;](#) [#xc88c;](#)[#xd45c;](#)[#xc640;](#) [#xd568;](#)[#xae01;](#) [#xbc18;](#)[#xd658;](#)[#xd569;](#)[#xb2c8;](#)[#xb2e4;](#).

Synopsis

```
setof record ST_PixelAsPolygons(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);
```

설명

래스터 밴드의 모든 픽셀의 경계이루는 폴리곤 도형을 각 픽셀의 값(이중 정밀도 데이터형)과 X, Y 래스좌표(정수형)와 함께 반환합니다.

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      &#xd568;&#xc218;&#xb294;      &#xbaa8;&#xb4e0;      &#xd53d;&#xc140;&#xc744;
&#xc544;&#xc6b0;&#xb974;&#xb294;      &#xd3f4;&#xb9ac;&#xace4;      &#xb3c4;&#xd615;      &#xd55c;
&#xac1c;&#xb97c;      &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.      &#xac01;      &#xb3c4;&#xd615;&#xc774;
&#xb3d9;&#xc77c;&#xd55c;      &#xd53d;&#xc140;      &#xac12;&#xc744;      &#xac00;&#xc9c0;&#xb294;      &#xd55c;
&#xac1c;      &#xc774;&#xc0c1;&#xc758;      &#xd53d;&#xc140;&#xc744;      &#xb098;&#xd0c0;&#xb0b4;&#xb294;
ST_DumpAsPolygons &#xd568;&#xc218;&#xc640;&#xb294; &#xb2e4;&#xb985;&#xb2c8;&#xb2e4;.
```

2.0.0 버전부터 사용할 수 있습니다.
개선 사항: 2.1.0 버전부터 선택적인
exclude_nodata_value 인수를 추가했습니다.
변경 사항: 2.1.1 버전에서 exclude_nodata_value의 &#xc2b5
변경했습니다.

예&#xc2dc:

```
-- &#xb798;&#xc2a4;&#xd130; &#xd53d;&#xc140; &#xd3f4;&#xb9ac;&#xace4; &#xd68d;&#xb4dd;
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
0	0	0	POINT(0 0)
1	0	0	POINT(1 0)
0	1	0	POINT(0 1)
1	1	0	POINT(1 1)

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))

󌰸고

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPoint, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids, ST_AsText

12.6.3 ST_PixelAsPoint

ST_PixelAsPoint — &#xd53d;&#xc140;&#xc758; &#xc88c;&#xc0c1;&#xb2e8;&#xc5d0; &#xc704;&#xce58;&#xd558;&#xb294; &#xd3ec;&#xc778;&#xd2b8; &#xb3c4;&#xd615;&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.

Synopsis

```
geometry ST_PixelAsPoint(raster rast, integer columnx, integer rowy);
```

설명

픽셀의 좌상단에 위치하는 포인도형을 반환합니다.

2.1.0 ಄전부터 사용할 수 있습니다.

예시

```
SELECT ST_AsText(ST_PixelAsPoint(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;
```

```
st_astext
```

POINT (0.5 0.5)

స고

ST_DumpAsPolygons, ST_PixelAsPolygon, ST_PixelAsPolygons, ST_PixelAsPoints, ST_PixelAsCentroid, ST_PixelAsCentroids

12.6.4 ST PixelAsPoints

ST_PixelAsPoints — [#b798;](#)[#c2a4;](#)[#d130;](#) [#bc34;](#)[#b4dc;](#)[#c758;](#) [#ac01;](#) [#d53d;](#)[#c140;](#)[#c5d0;](#) [#b300;](#)[#d3ec;](#)[#c778;](#)[#d2b8;](#) [#b3c4;](#)[#d615;](#)[#c744;](#) [#ac01;](#) [#d53d;](#)[#c140;](#)[#c758;](#) [#ac12;](#) [#bc0f;](#) X, Y [#b798;](#)[#c2a4;](#)[#d130;](#) [#c88c;](#)[#d45c;](#)[#c640;](#) [#d568;](#)[#xaed8;](#) [#bc18;](#)[#d658;](#)[#d569;](#)[#b2c8;](#)[#b2e4;](#) [#d3ec;](#)[#c778;](#)[#d2b8;](#) [#b3c4;](#)[#d615;](#)[#c758;](#) [#c88c;](#)[#d45c;](#)[#b294;](#) [#d53d;](#)[#c140;](#)[#c758;](#) [#c88c;](#)[#d45c;](#)[#c785;](#)[#b2c8;](#)[#b2e4;](#).

Synopsis

```
setof record ST_PixelAsPoints(raster rast, integer band=1, boolean exclude_nodata_value=TRUE);
```

ST_DumpAsPoints

ST_DumpAsPoints(*geom*) returns a set of points representing the vertices of the geometry. The output is a table with columns: *x*, *y*, *val*. The *val* column contains the value of the pixel at the point. The *x* and *y* columns contain the coordinates of the point. The *geom* argument is a geometry object.

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.

2.1.0 ST_DumpAsPoints(*geom*, *exclude_nodata_value*)

ST_DumpAsPoints(*geom*, *exclude_nodata_value*) returns a set of points representing the vertices of the geometry. The output is a table with columns: *x*, *y*, *val*. The *val* column contains the value of the pixel at the point. The *x* and *y* columns contain the coordinates of the point. The *geom* argument is a geometry object. The *exclude_nodata_value* argument is a boolean value. If *exclude_nodata_value* is TRUE, only those pixels whose values are not NODATA are returned as points.

ST_DumpAsPolygons

```
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)

ST_DumpAsPolygons

ST_DumpAsPolygons(*geom*) returns a set of polygons representing the pixels of the geometry. The output is a table with columns: *x*, *y*, *val*. The *val* column contains the value of the pixel. The *x* and *y* columns contain the coordinates of the vertices of the polygon. The *geom* argument is a geometry object.

12.6.5 ST_PixelAsCentroid

ST_PixelAsCentroid — Returns the centroid of the pixel represented by the raster value *val* in the raster *rast*. The centroid is the point in the pixel that is equidistant to all four corners of the pixel.

Synopsis

geometry **ST_PixelAsCentroid**(raster *rast*, integer *x*, integer *y*);

Parameters

rast: A raster with a single band. The raster must be a *SRID* of 4326 (WGS 1984) and the units must be in decimal degrees. The raster must be a *SRID* of 4326 (WGS 1984) and the units must be in decimal degrees.

x: The x-coordinate of the pixel. The x-coordinate must be an integer between 1 and the width of the raster.

y: The y-coordinate of the pixel. The y-coordinate must be an integer between 1 and the height of the raster.

Return Value

```
SELECT ST_AsText(ST_PixelAsCentroid(rast, 1, 1)) FROM dummy_rast WHERE rid = 1;

 st_astext
-----
POINT(1.5 2)
```

See Also

[ST_DumpAsPolygons](#), [ST_PixelAsPolygon](#), [ST_PixelAsPolygons](#), [ST_PixelAsPoint](#), [ST_PixelAsPoints](#), [ST_PixelAsCentroids](#)

12.6.6 ST_PixelAsCentroids

ST_PixelAsCentroids — Returns the centroids of all pixels in the raster *rast*. The centroids are the points in the pixels that are equidistant to all four corners of the pixels. The centroids are returned as a *geometry* collection of *POINT* objects.

Synopsis

setof record **ST_PixelAsCentroids**(raster *rast*, integer *band*=1, boolean *exclude_nodata_value*=TRUE);

Parameters

rast: A raster with a single band. The raster must be a *SRID* of 4326 (WGS 1984) and the units must be in decimal degrees. The raster must be a *SRID* of 4326 (WGS 1984) and the units must be in decimal degrees.

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.

2.1.0 宄전에서 C ָ어로 다쓰였습니다.

2.1.1 宄전에서 exclude_nodata_value 의 습변경했습니다.

2.1.0 宄전부터 사용할 수 있습니다.

예시

```
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)

참고

[ST_DumpAsPolygons](#), [ST_PixelAsPolygon](#), [ST_PixelAsPolygons](#), [ST_PixelAsPoint](#), [ST_PixelAsPoints](#), [ST_PixelAsCentroid](#)

12.6.7 ST_Value

`ST_Value` — 입력 columnx, rowy 픽셀의 입력 ଴드의 값, 또는 특정 기하학적 포인트의 ଴드의 값을 ଘ환합니다. ଴드 ஈ஀터 시작하며, 따로 지정하지 않경우 1로 ఀ정합니다. exclude_nodata_value 를 ౰&

```

    nodata &#xd53d;&#xc140;&#xc744; &#xd3ec;&#xd568;&#xd55c; &#xbaa8;&#
    &#xd53d;&#xc140;&#xc774; &#xad50;&#xcc28;&#xd55c;&#xb2e4;&#xace0; &#xac00;&#xc815;&#xd558;&#xace0; &#xac12;&#
    &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. exclude_nodata_value &#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0;
    &#xc54a;&#xc740; &#xacbd;&#xc6b0;, &#xb798;&#xc2a4;&#xd130;&#xc758; &#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130;&#
    &#xac12;&#xc744; &#xc77d;&#xc5b4; &#xb4e4;&#xc785;&#xb2c8;&#xb2e4;.

```

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);

```

설명

```

    columnx, rowy &#xd53d;&#xc140; &#xb610;&#xb294; &#xd2b9;&#xc815; &#xb3c4;&#xd615;&#xac
    &#xad50;&#xcc28;&#xd558;&#xb294; &#xd53d;&#xc140;&#xb4e4;&#xc758; &#xc704;&#xce58;&#xc758; &#bc34;&#xb4dc;
    &#xac12;&#xc744; &#xc124;&#xc815;&#xd574;&#xc11c; &#xb098;&#xc628; &#xc218;&#xc815;&#xb41c; &#xb798;&#xc2a4;&#
    &#bc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. &#bc34;&#xb4dc; &#bc88;&#xd638;&#xb294; 1&#xbd80;&#xd130; &#xc2dc;
    &#b530;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; 1&#xb85c; &#xac00;&#xc815;&#
    exclude_nodata_value &#xb97c; &#xcc38;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd560; &#xacbd;&#xc6b0;, &#xb798;&#
    nodata &#xac12;&#xc774; &#xc544;&#xb2cc; &#xd53d;&#xc140;&#xc758; &#xac1c;&#xc218;&#xb9cc; &#bc18;&#xd658;&#x
    &#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#baa8;&#xb4e0; &#xd53d;&#xc140;&#xc758; &#xac1c;&#xc218;&#xb97c; &#xad6c;&#
    exclude_nodata_value &#xb97c; &#xac70;&#xc9d3;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#x

```

The allowed values of the `resample` parameter are "nearest" which performs the default nearest-neighbor resampling, and "bilinear" which performs a [bilinear interpolation](#) to estimate the value between pixel centers.

```

&#xac1c;&#xc120; &#xc0ac;&#xd56d;: 2.1.0 &#bc84;&#xc804;&#xbd80;&#xd130; &#xc120;&#xd0dd;&#xc801;&#xc778;
exclude_nodata_value &#xc778;&#xc218;&#xb97c; &#xcd94;&#xac00;&#xd588;&#xc2b5;&#xb2c8;&#xb2e4;.

```

```

&#xac1c;&#xc120; &#xc0ac;&#xd56d;: 2.0.0 &#bc84;&#xc804;&#xbd80;&#xd130; &#xc120;&#xd0dd;&#xc801;&#xc778;
exclude_nodata_value &#xc778;&#xc218;&#xb97c; &#xcd94;&#xac00;&#xd588;&#xc2b5;&#xb2c8;&#xb2e4;.

```

예시

```

-- &#xd2b9;&#xc815; PostGIS &#xb3c4;&#xd615; &#xd3ec;&#xc778;&#xd2b8;  ←
  &#xc704;&#xce58;&#xc758; &#xb798;&#xc2a4;&#xd130; &#xac12;&#xc744;  ←
  &#xc5bb;&#xc2b5;&#xb2c8;&#xb2e4;.
-- &#xc0ac;&#xc6a9;&#xc790; &#xb3c4;&#xd615;&#xc758; SRID&#xac00; &#xc0ac;&#xc6a9;&#xc790;  ←
  &#xb798;&#xc2a4;&#xd130;&#xc640; &#xb3d9;&#xc77c;&#xd574;&#xc57c;  ←
  &#xd569;&#xb2c8;&#xb2e4;.
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

-- &#xc2e4;&#xc81c; &#xd14c;&#xc774;&#xbe14;&#xc744; &#xc774;&#xc6a9;&#xd55c;  ←
  &#xc77c;&#xbc18;&#xc801;&#xc778; &#xd5c8;&#xad6c;&#xc758; &#xc608;&#xc2dc;
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

```
-- &#xc01; &#xd53d;&#xc140;&#xc758; &#xbc34;&#xb4dc; 1, 2, 3&#xc758; &#xaa8;&#xb4e0; &#xc12;&#xb4e4;&#xc744; &#xc5bb;&#xc2b5;&#xb2c8;&#xb2e4;. --
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
:				
:				

```
-- &#xc55e;&#xc758; &#xc608;&#xc2dc;&#xcc98;&#xb7fc; &#xc01; &#xd53d;&#xc140;&#xc758; &#xbc34;&#xb4dc; 1, 2, 3&#xc758; &#xaa8;&#xb4e0; &#xc12;&#xb4e4;&#xc744; &#xc5bb;&#xc9c0;&#xb9cc; &#xc01; &#xd53d;&#xc140;&#xc758; &#xc88c;&#xc0c1;&#xb2e8; &#xd3ec;&#xc778;&#xd2b8;&#xb3c4; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. --
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
:			

```
-- &#xd2b9;&#xc815; &#xc12; &#xbc94;&#xc704;&#xc5d0; &#xb4e4;&#xc5b4;&#xc624;&#xba70; &#xd2b9;&#xc815; &#xd3f4;&#xb9ac;&#xace4;&#xacfc; &#xad50;&#xcc28;&#xd558;&#xb294; &#xaa8;&#xb4e0; &#xd53d;&#xc140;&#xb4e4;&#xc758; &#xd569;&#xc9d1;&#xd569;&#xc73c;&#xb85c; &#xc774;&#xb8e8;&#xc5b4;&#xc9c4; &#xd3f4;&#xb9ac;&#xace4;&#xc744; &#xc5bb;&#xc2b5;&#xb2c8;&#xb2e4;. --
SELECT ST_AsText(ST_Union(pixpolyg)) As shadow
FROM (SELECT ST_Translate(ST_MakeEnvelope(
    ST_UpperLeftX(rast), ST_UpperLeftY(rast),
    ST_UpperLeftX(rast) + ST_ScaleX(rast),
    ST_UpperLeftY(rast) + ST_ScaleY(rast), 0
```

```

        ), ST_ScaleX(rast)*x, ST_ScaleY(rast)*y
    ) As pixpolyg, ST_Value(rast, 2, x, y) As b2val
FROM dummy_rast CROSS JOIN
generate_series(1,1000) As x CROSS JOIN generate_series(1,1000) As y
WHERE rid = 2
    AND x <= ST_Width(rast) AND y <= ST_Height(rast)) As foo
WHERE
    ST_Intersects(
        pixpolyg,
        ST_GeomFromText('POLYGON((3427928 5793244,3427927.75 5793243.75,3427928
        5793243.75,3427928 5793244))',0)
    ) AND b2val != 254;

shadow
-----
MULTIPOLYGON(((3427928 5793243.9,3427928 5793243.85,3427927.95 5793243.85,3427927.95
5793243.9,
3427927.95 5793243.95,3427928 5793243.95,3427928.05 5793243.95,3427928.05
5793243.9,3427928 5793243.9)),((3427927.95 5793243.9,3427927.95 579324
3.85,3427927.9 5793243.85,3427927.85 5793243.85,3427927.85 5793243.9,3427927.9
5793243.9,3427927.9 5793243.95,
3427927.95 5793243.95,3427927.95 5793243.9)),((3427927.85 5793243.75,3427927.85
5793243.7,3427927.8 5793243.7,3427927.8 5793243.75
,3427927.8 5793243.8,3427927.8 5793243.85,3427927.85 5793243.85,3427927.85
5793243.8,3427927.85 5793243.75)),
((3427928.05 5793243.75,3427928.05 5793243.7,3427928 5793243.7,3427927.95
5793243.7,3427927.95 5793243.75,3427927.95 5793243.8,3427
927.95 5793243.85,3427928 5793243.85,3427928 5793243.8,3427928.05 5793243.8,
3427928.05 5793243.75)),((3427927.95 5793243.75,3427927.95 5793243.7,3427927.9
5793243.7,3427927.85 5793243.7,
3427927.85 5793243.75,3427927.85 5793243.8,3427927.85 5793243.85,3427927.9 5793243.85,
3427927.95 5793243.85,3427927.95 5793243.8,3427927.95 5793243.75)))

-- &#xb300;&#xc6a9;&#xb7c9; &#xb798;&#xc2a4;&#xd130; &#xd0c0;&#xc77c;&#xc758;
&#xbaa8;&#xb4e0; &#xd53d;&#xc140;&#xc744; &#xd655;&#xc778;&#xd558;&#xb294;
&#xc791;&#xc5c5;&#xc740; &#xc624;&#xb798; &#xac78;&#xb9b4; &#xc218;
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
-- &#xb2e8;&#xacc4;&#xbcc4;&#xb85c; &#xc120;&#xd0dd;&#xd560; &#xc218; &#xc788;&#xb294;
generate_series &#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb97c; &#xc774;&#xc6a9;&#xd574;&#xc11c;
-- &#xc790;&#xb9bf;&#xc218;&#xb97c; &#xd1b5;&#xd574; &#xc815;&#xd655;&#xb3c4;&#xb97c;
&#xc870;&#xae08; &#xc904;&#xc774;&#xba74; &#xc18d;&#xb3c4;&#xb97c; &#xd06c;&#xac8c;
&#xd5a5;&#xc0c1;&#xc2dc;&#xd0ac; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
-- &#xb2e4;&#xc74c; &#xc608;&#xc2dc;&#xb294; &#xc774;&#xc804; &#xc608;&#xc2dc;&#xc640;
&#xac19;&#xc740; &#xc791;&#xc5c5;&#xc744; &#xd558;&#xc9c0;&#xb9cc; &#xb9e4; 4(2x2)
&#xd53d;&#xc140;&#xb9c8;&#xb2e4; &#xd53d;&#xc140; 1&#xac1c;&#xb97c;
&#xd655;&#xc778;&#xd574;&#xc11c; &#xd655;&#xc778;&#xd55c; &#xac83;&#xc73c;&#xb85c;
&#xcc98;&#xb9ac;&#xd569;&#xb2c8;&#xb2e4;.
-- &#xd655;&#xc778;&#xb41c; &#xd53d;&#xc140;&#xc758; &#xac12;&#xc744; &#xadf8;
&#xb2e4;&#xc74c; &#xd53d;&#xc140; 4&#xac1c;&#xc758; &#xac12;&#xc73c;&#xb85c;
&#xc800;&#xc7a5;&#xd569;&#xb2c8;&#xb2e4;.

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;

-----
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)))

```

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

ST_NearestValue

12.6.8 ST_NearestValue

ST_NearestValue — columnx, rowy, geometry pt, boolean exclude_nodata_value=true; 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);

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);

ST_NearestValue

columnx, rowy, geometry pt, boolean exclude_nodata_value=true; 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);

columnx, rowy, geometry pt, boolean exclude_nodata_value=true; 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);

2.1.0

**Note**

ST_NearestValue ST_Value

2.1.0

```
-- 2x2
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
```

```
-- 2x3
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
```

ST_Neighborhood, ST_Value

12.6.9 ST_SetSkew

ST_SetSkew — Returns a geometry with the same X/Y coordinates as the input geometry, and values from the raster copied into the Z dimension using the requested resample algorithm.

Synopsis

bytea ST_AsGDALRaster(raster rast, text format, text[] options=NULL, integer srid=sameassource);

Returns a geometry with the same X/Y coordinates as the input geometry, and values from the raster copied into the Z dimensions using the requested resample algorithm.

The `resample` parameter can be set to "nearest" to copy the values from the cell each vertex falls within, or "bilinear" to use [bilinear interpolation](#) to calculate a value that takes neighboring cells into account also.

2.2.0

```
--
-- 2x2 test raster with values
--
-- 10 50
-- 40 20
--
WITH test_raster AS (
SELECT
ST_SetValues(
  ST_AddBand(
    ST_MakeEmptyRaster(width => 2, height => 2,
      upperleftx => 0, upperlefty => 2,
      scalex => 1.0, scaley => -1.0,
```

```

        skewx => 0, skewy => 0, srid => 4326),
        index => 1, pixeltype => '16BSI',
        initialvalue => 0,
        nodataval => -999),
        1,1,1,
        newvalueset => ARRAY[ARRAY[10.0::float8, 50.0::float8], ARRAY[40.0::float8, 20.0::float8 ←
            ]] AS rast
    )
SELECT
ST_AsText (
    ST_SetZ (
        rast,
        band => 1,
        geom => 'SRID=4326;LINESTRING(1.0 1.9, 1.0 0.2)::geometry,
        resample => 'bilinear'
    ))
FROM test_raster

          st_astext
-----
LINESTRING Z (1 1.9 38,1 0.2 27)

```

ST_SetSRID

ST_Value, ST_SetSRID

12.6.10 ST_SetSkew

ST_SetSkew — Returns a geometry with the same X/Y coordinates as the input geometry, and values from the raster copied into the Z dimension using the requested resample algorithm.

Synopsis

bytea **ST_AsGDALRaster**(raster rast, text format, text[] options=NULL, integer srid=sameassource);

ST_SetValues

Returns a geometry with the same X/Y coordinates as the input geometry, and values from the raster copied into the Z dimensions using the requested resample algorithm.

The `resample` parameter can be set to "nearest" to copy the values from the cell each vertex falls within, or "bilinear" to use **bilinear interpolation** to calculate a value that takes neighboring cells into account also.

2.2.0 **ST_SetValues**(geometry geom, raster rast, integer band, text[] options=NULL, integer srid=sameassource);

ST_SetValues

```

--
-- 2x2 test raster with values
--
-- 10 50
-- 40 20
--
WITH test_raster AS (
SELECT
ST_SetValues(

```



```

ST_AddBand(
  ST_MakeEmptyRaster(width => 2, height => 2,
    upperleftx => 0, upperlefty => 2,
    scalex => 1.0, scaley => -1.0,
    skewx => 0, skewy => 0, srid => 4326),
  index => 1, pixeltype => '16BSI',
  initialvalue => 0,
  nodataval => -999),
  1,1,1,
  newvalueset => ARRAY[ARRAY[10.0::float8, 50.0::float8], ARRAY[40.0::float8, 20.0::float8] ←
    ]]) AS rast
)
SELECT
ST_AsText (
  ST_SetM(
    rast,
    band => 1,
    geom => 'SRID=4326;LINESTRING(1.0 1.9, 1.0 0.2)::geometry,
    resample => 'bilinear'
  ))
FROM test_raster

          st_astext
-----
LINESTRING M (1 1.9 38,1 0.2 27)

```

ST_SetSRID

ST_Value, **ST_SetSRID**

12.6.11 ST_Neighborhood

ST_Neighborhood — columnx, rowy, distanceX, distanceY, exclude_nodata_value

ST_Neighborhood(raster rast, integer bandnum, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, integer bandnum, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

Synopsis

ST_Neighborhood(raster rast, integer bandnum, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, integer bandnum, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood

columnx, rowy, distanceX, distanceY, exclude_nodata_value

ST_Neighborhood(raster rast, integer bandnum, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, integer columnX, integer rowY, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, integer bandnum, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

ST_Neighborhood(raster rast, geometry pt, integer distanceX, integer distanceY, boolean exclude_nodata_value=true);

```

distanceX &#x0ac;&#xc6a9;&#xc790; &#xc9c0;&#xce68;&#xc11c;
distanceY &#xd30c;&#xb77c;&#xbbb8;&#xd130;&#xc9c0;&#xc815;&#xd55c; &#xd53d;&#xc140; &#xc8fc;&#xc704;&#xc758; &#xd53d;&#xc140; &#xac1c;&#xc218;&#xb97c;
X &#xbc0f; Y &#xcd95;&#xc73c;&#xb85c; &#xc815;&#xc758;&#xd569;&#xb2c8;&#xb2e4;,. &#xc608;&#xb97c; &#xb4e4;&#xc5b4;&#xc0ac;&#xc6a9;&#xc790; &#xc124;&#xc815; &#xd53d;&#xc140; &#xc8fc;&#xc704;&#xb85c; X&#xcd95;&#xc744; &#xb530;&#xb77c;
3&#xd53d;&#xc140; &#xac70;&#xb9ac; &#xc548; &#xadf8;&#xb9ac;&#xace0; Y&#xcd95;&#xc744; &#xb530;&#xb77c;
2&#xd53d;&#xc140; &#xac70;&#xb9ac; &#xc548;&#xc758; &#xaa8;&#xb4e0; &#xac12;&#xc744; &#xc6d0;&#xd558;&#xb294;
&#xacbd;&#xc6b0; &#xb9d0;&#xc785;&#xb2c8;&#xb2e4;,. 2&#xcc28;&#xc6d0; &#xbc30;&#xc5f4;&#xc758; &#xc911;&#xc2ec;&#xc0ac;&#xc6a9;&#xc790;
columnx &#xbc0f; rowy &#xb610;&#xb294; &#xae30;&#xd558;&#xd559;&#xc801; &#xd3ec;&#xc778;&#xd2b8;&#xb85c;
&#xc9c0;&#xc815;&#xb41c; &#xd53d;&#xc140;&#xc758; &#xac12;&#xc774; &#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;,.
&#xbc34;&#xb4dc; &#xbc88;&#xd638;&#xb294; 1&#xbd80;&#xd130; &#xc2dc;&#xc791;&#xd558;&#xba70;,. &#xb530;&#xb85c;
&#xc9c0;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; bandnum &#xc744; 1&#xb85c; &#xac00;&#xc815;&#xc9c0;
exclude_nodata_value &#xb97c; &#xac70;&#xc9d3;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd560; &#xacbd;&#xc6b0;,.
nodata &#xd53d;&#xc140;&#xc744; &#xd3ec;&#xd568;&#xd55c; &#xaa8;&#xb4e0; &#xd53d;&#xc140;&#xc774; &#xad50;&#xc774;
&#xac00;&#xc815;&#xd558;&#xace0; &#xac12;&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;,. exclude_nodata_value
&#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc740; &#xacbd;&#xc6b0;,. &#xb798;&#xc2a4;&#xd130;&#xc758;
&#xba54;&#xd0c0;&#xb370;&#xc774;&#xd130;&#xc5d0;&#xc11c; &#xac12;&#xc744; &#xc77d;&#xc5b4; &#xb4e4;&#xc785;&#xb2c8;&#xb2e4;,.

```

Note

```

&#xbc18;&#xd658;&#xb418;&#xb294; 2&#xcc28;&#xc6d0; &#xbc30;&#xc5f4;&#xc758; &#xac01; &#xcd95;&#xc758;
&#xad6c;&#xc131; &#xc694;&#xc18c; &#xac1c;&#xc218;&#xb294; 2 * (distanceX|distanceY) +
1&#xc785;&#xb2c8;&#xb2e4;,. &#xb530;&#xb77c;&#xc11c; distanceX &#xc640; distanceY &#xac00;
&#xac01;&#xac01; 1&#xc778; &#xacbd;&#xc6b0;,. &#xbc18;&#xd658;&#xb418;&#xb294; &#xbc30;&#xc5f4;&#xc740;
3x3&#xc774; &#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;,.

```

Note

```

ST_Min4ma, ST_Sum4ma, ST_Mean4ma &#xac19;&#xc740; &#xc5b4;&#xb5a4; &#xb798;&#xc2a4;&#xd130;
&#xacf5;&#xac04; &#xcc98;&#xb9ac; &#xb0b4;&#xc7a5; &#xd568;&#xc218;&#xb3c4; &#xc774; 2&#xcc28;&#xc6d0;
&#xbc30;&#xc5f4; &#xcd9c;&#xb825;&#xbb3c;&#xc744; &#xc785;&#xb825;&#xbc1b;&#xc744; &#xc218;
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;,.

```

2.1.0 버전부터 사용할 수 있습니다,.</p>
</div>
<div data-bbox="90 624 229 640" data-label="Section-Header>
예시
</div>
<div data-bbox="90 654 816 900" data-label="Text>
<pre>
-- 2x2 픽셀이 값을 가지고 ↔
있습니다,.
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
)
 </pre>
</div>
</div>

```

        ) AS rast
    ) AS foo

```

```

        st_neighborhood
-----

```

```

{{NULL,1,1},{1,1,NULL},{1,1,1}}

```

```

-- 2x3 &#xd53d;&#xc140;&#xc774; NODATA&#xc785;&#xb2c8;&#xb2e4;.

```

```

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}}

```

```

-- 3x3 &#xd53d;&#xc140;&#xc774; &#xac12;&#xc744; &#xac00;&#xc9c0;&#xace0; ↔

```

```

    &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.

```

```

-- exclude_nodata_value = FALSE

```

```

SELECT

```

```

    ST_Neighborhood(rast, 3, 3, 1, 1, false)

```

```

FROM (

```

```

    ST_SetValues(

```

```

        ST_AddBand(

```

```

            ST_MakeEmptyRaster(5, 5, -2, 2, 1, -1, 0, 0, 0),
            '8BUI'::text, 1, 0

```

```

        ),

```

```

        1, 1, 1, ARRAY[

```

```

            [0, 1, 1, 1, 1],

```

```

            [1, 1, 1, 0, 1],

```

```

            [1, 0, 1, 1, 1],

```

```

            [1, 1, 1, 1, 0],

```

```

            [1, 1, 0, 1, 1]

```

```

        ]::double precision[],

```

```

        1

```

```

    ) AS rast

```

```

    ) AS foo

```

```

        st_neighborhood
-----

```

```

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

```

ST_SetValue

ST_NearestValue, ST_Min4ma, ST_Max4ma, ST_Sum4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

12.6.12 ST_SetValue

ST_SetValue — Set the value of the specified pixel in the raster. The pixel is identified by its row and column or by its geometry. The value is set to the specified new value. The function returns the modified raster.

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);

ST_SetValue

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.

ST_SetValue (rast, bandnum, geom, newvalue) → raster
 Set the value of the specified pixel in the raster. The pixel is identified by its row and column or by its geometry. The value is set to the specified new value. The function returns the modified raster.

ST_SetValues

```
-- Set the value of the specified pixel in the raster.
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 ...

```
-- Set the value of the specified pixel in the raster.
UPDATE dummy_rast SET rast = ST_SetValue(rast,1, ST_Point(3427927.75, 5793243.95)
,100)
WHERE rid = 2 ;
```

󌰸고

ST_Value, ST_DumpAsPolygons

12.6.13 ST_SetValues

ST_SetValues — 지정한 밴드의 값들을 설정한 나온 수정된 래스터를 반환합니다

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);
```

설명

지정한 밴드에 대해, 설정한 픽셀들을 새로 운 값(들)로 설정해서 나온 수 래스터를 반환합니다.

keepnodata 가 TRUE일 경우, NODATA 값인 픽셀들을 newvalueset 에 있는 상응하는 값으로 설않을 것입니다.

변종 1의 경우, columnx, rowy 픽셀 좌표 및 newvalueset 변종 열의 차원이 설정 대상 픽셀들을 결정합니다. 이미 newvalueset 에 있는 값막기 위해 noset 을 이용할 수 있습니다 이가 빠진 배열을 허용하지 않기 때문입니다). 변종 1 예시를 참조하

변종 2는 변종 1과 비슷하지만, 불 값인 noset 배열 대신 단순한 이중 정데이터형인 nosetvalue 를 입력받습니다 newvalueset 가운데 nosetvalue 값을 가진 구성 요무시할 것입니다. 변종 2 예시를 참조하

변종 3의 경우, columnx, rowy 픽셀 좌표, width 및 height 가 설정 대상 픽셀들을 결정합니다. 변종 3 예시를 참조하십시오.

변종 4는 rast 의 첫 번째 밴드의 픽셀설정될 것이라고 가정하는 점만 제외하면 변종 3과 동일합니다.

변종 5의 경우, 설정 대상 픽셀들을 결정하는 데 geomval 의 배열을 이용합데열 안의 도형들이 모두 POINT 또는 MULTIPOINT일 경우, 이 함수는 ఁ 포인트인 경도와 위도를 써서 직접 픽셀을

124;815;558;294;9c0;b984;ae38;744;774;6a9;d569;b2c8;b2e4;
 xadf8;b807;9c0;54a;744;acbd;6b0; b3c4;d615;744; b798;2a4;d130;
 xbcc0;d658;d55c; b2e4;74c; b3d9;77c;d55c; bc29;2dd;73c; b85c; bc18;
 xbcc0;c885; 5 608;2dc;b97c; cc38;870;d558;2ed;2dc;c624;.

2.1.0;bc84;c804;bd80;d130; 0ac;6a9;d560; 218; 788;2b5;b2c8;b2e4;.

608;2dc;: bcc0;c885; 1

```

/*
ST_SetValues() &#xd568;&#xc218;&#xac00; &#xb2e4;&#xc74c; &#xc791;&#xc5c5;&#xc744;  ↔
&#xd569;&#xb2c8;&#xb2e4; .

+ - + - + - +      + - + - + - +
| 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

```

```

/*
ST_SetValues() &#xd568;&#xc218;&#xac00; &#xb2e4;&#xc74c; &#xc791;&#xc5c5;&#xc744;  ↔
&#xd569;&#xb2c8;&#xb2e4; .

+ - + - + - +      + - + - + - +
| 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

```

```

/*
ST_SetValues()  #568; #218; #00; #b2e4; #74c; #791; #5c5; #744;  ←
    #d569; #b2c8; #b2e4; .

+ - + - + - +      + - + - + - +
| 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
        )
    ) AS poly
) foo
ORDER BY 1, 2;

```

```

        [],
        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

/*
ST_SetValues() &#xd568;&#xc218;&#xac00; &#xb2e4;&#xc74c; &#xc791;&#xc5c5;&#xc744;  <->
&#xd569;&#xb2c8;&#xb2e4;.

+ - + - + - +      + - + - + - +
| 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

```

ST_SetValues()

```

/*
ST_SetValues() &#xd568;&#xc218;&#xac00; &#xb2e4;&#xc74c; &#xc791;&#xc5c5;&#xc744;  ←
&#xd569;&#xb2c8;&#xb2e4;.

+ - + - + - +      + - + - + - +
| 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

```

```

/*
&#xc774; &#xc608;&#xc2dc;&#xb294; &#xc774;&#xc804; &#xc608;&#xc2dc;&#xc640;  ←
&#xb2e4;&#xc2b7;&#xd558;&#xc9c0;&#xb9cc;, nosetValue = -1 &#xb300;&#xc2e0; nosetValue =  ←
NULL &#xc744; &#xc501;&#xb2c8;&#xb2e4;.

ST_SetValues() &#xd568;&#xc218;&#xac00; &#xb2e4;&#xc74c; &#xc791;&#xc5c5;&#xc744;  ←
&#xd569;&#xb2c8;&#xb2e4;.

+ - + - + - +      + - + - + - +
| 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

```

ST_SetValues(geom, val, x, y, z)

```

/*
ST_SetValues() &#xd568;&#xc218;&#xac00; &#xb2e4;&#xc74c; &#xc791;&#xc5c5;&#xc744;  ←
    &#xd569;&#xb2c8;&#xb2e4;.

+ - + - + - +      + - + - + - +
| 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(

```


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)

```
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))::ge  
    ometry 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)

[ST_Value](#), [ST_SetValue](#), [ST_PixelAsPolygons](#)

12.6.14 ST_DumpValues

ST_DumpValues — Returns an array of record sets, one for each pixel in the raster. Each record set contains the pixel's value, the pixel's coordinates, and the pixel's band number.

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 );
```

nbands, nband, NULL, and 2.1.0

2.1.0

ST_AddBand

```
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;
```

nbands	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_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)

```
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)

ST_SetValue

[ST_Value](#), [ST_SetValue](#), [ST_SetValues](#)

12.6.15 ST_PixelOfValue

[ST_PixelOfValue](#) — Returns the value of the pixel at the given row and column.

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 );
```

Parameters

rast: raster to be processed.
nband: integer value representing the number of bands in the raster.
search: array of double precision values to search for in the raster.
exclude_nodata_value: boolean value indicating whether to exclude nodata values from the search results.

2.1.0: **search** can also be a double precision value.

Return Value

```
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),
                                        -1, 0, 0, 0),
                                    '8BUI'::text, 1, 0
                                ),
                                1, 1, 0
                            ),
                            2, 3, 0
                        ),
                        3, 5, 0
                    ),
                    4, 2, 0
                ),
                5, 4, 255
            ),
            1, ARRAY[1, 255]) AS pixels
) AS foo
```

val	x	y
1	1	2
1	1	3
1	1	4
1	1	5
1	2	1
1	2	2
1	2	4
1	2	5
1	3	1
1	3	2

```

1 | 3 | 3
1 | 3 | 4
1 | 4 | 1
1 | 4 | 3
1 | 4 | 4
1 | 4 | 5
1 | 5 | 1
1 | 5 | 2
1 | 5 | 3
255 | 5 | 4
1 | 5 | 5

```

12.7 ST_SetGeoReference

12.7.1 ST_SetGeoReference

ST_SetGeoReference — Set the geographic coordinate system (SRID) of a raster. The function takes a raster and a text string representing the SRID. The SRID can be a numeric value (e.g., 4326) or a text string (e.g., 'EPSG:4326'). The function returns the raster with the SRID set to the specified value.

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);

GDAL:

ST_SetGeoReference(rast, georefcoords, format) — Set the geographic coordinate system (SRID) of a raster. The function takes a raster and a text string representing the SRID. The SRID can be a numeric value (e.g., 4326) or a text string (e.g., 'EPSG:4326'). The function returns the raster with the SRID set to the specified value.

GDAL:

```
scalex skewx skewy scaley upperleftx upperlefty
```

ESRI:

```
scalex skewy skewx scaley upperleftx + scalex*0.5 upperlefty + scaley*0.5
```

Note



The **ST_SetGeoReference** function can be used to set the SRID of a raster. The function takes a raster and a text string representing the SRID. The SRID can be a numeric value (e.g., 4326) or a text string (e.g., 'EPSG:4326'). The function returns the raster with the SRID set to the specified value.

2.1.0; ST_SetGeoReference(raster, double precision, ...) ;

Example

```
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	0.1	0	0	0.1	5	5	10	-10	0	←
2	0.0999999999999996	0	0	0.0999999999999996	5	5	10	-10	0	←
3	0.001	0	1	1	5	5	10	-10	0.001	←

ST_SetGeoReference

ST_GeoReference, ST_ScaleX, ST_ScaleY, ST_UpperLeftX, ST_UpperLeftY

12.7.2 ST_SetRotation

ST_SetRotation — Set the rotation of a raster. The rotation is specified in degrees.

Synopsis

float8 **ST_SetRotation**(raster rast, float8 rotation);

Example

Set the rotation of a raster to 45 degrees. The rotation is specified in degrees.

ST_Scale

```
SELECT
  ST_ScaleX(rast1), ST_ScaleY(rast1), ST_SkewX(rast1), ST_SkewY(rast1),
  ST_ScaleX(rast2), ST_ScaleY(rast2), ST_SkewX(rast2), ST_SkewY(rast2)
FROM (
  SELECT ST_SetRotation(rast, 15) AS rast1, rast as rast2 FROM dummy_rast
) AS foo;
```

st_scalex	st_scaley	st_skewx	st_skewy	
st_scalex	st_scaley	st_skewx	st_skewy	
-1.51937582571764	-2.27906373857646	1.95086352047135	1.30057568031423	↔
2	3	0	0	
-0.0379843956429411	-0.0379843956429411	0.0325143920078558	0.0325143920078558	↔
0.05	-0.05	0	0	

ST_Scale

ST_Rotation, ST_ScaleX, ST_ScaleY, ST_SkewX, ST_SkewY

12.7.3 ST_SetScale

ST_SetScale — X: X coordinate; Y: Y coordinate; angle: rotation angle in degrees; scale_x: X scale factor; scale_y: Y scale factor; skew_x: X skew factor; skew_y: Y skew factor.

Synopsis

```
raster ST_SetScale(raster rast, float8 xy);
raster ST_SetScale(raster rast, float8 x, float8 y);
```

ST_SetScale

X: X coordinate; Y: Y coordinate; angle: rotation angle in degrees; scale_x: X scale factor; scale_y: Y scale factor; skew_x: X skew factor; skew_y: Y skew factor.

Note

ST_SetScale is a convenience function that calls **ST_Rescale** with the same arguments. **ST_Rescale** is a more powerful function that allows for more complex transformations. **ST_SetScale** is a convenience function that calls **ST_Rescale** with the same arguments. **ST_Rescale** is a more powerful function that allows for more complex transformations.



```
&#xbcc0;&#xacbd;&#xc0ac;&#xd56d;: 2.0.0 &#xbbf8;&#xb9cc; WKTRaster &#xbc84;&#xc804;&#xc5d0;&#xc11c;&#xb294;  
ST_SetPixelSize&#xb77c;&#xb294;&#xba85;&#xce6d;&#xc774;&#xc5c8;&#xc2b5;&#xb2c8;&#xb2e4;. 2.0.0 &#xbc84;&#xc804;&  
&#xd604;&#xc7ac;&#xba85;&#xce6d;&#xc73c;&#xb85c;&#xbcc0;&#xacbd;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시

```
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)

స고

ST_ScaleX, ST_ScaleY, Box3D

12.7.4 ST_SetSkew

ST_SetSkew — `샀 리 참 조 X 및 Y 기 울 기 (skew) (또 는 회 전 각 파 라 미 터)` `를 설 정 합 니 다 .`
`값 하 나 만 입 력 할 경 우 , X 와 Y 를 동 &#xc77`
`값 으 로 설 정 합 니 다 .`

Synopsis

```
raster ST_SetSkew(raster rast, float8 skewxy);  
raster ST_SetSkew(raster rast, float8 skewx, float8 skewy);
```

설&#xba85:

지리참조 X 및 Y 기울기(또는 회전&#xac01
파라򻯸터)를 설정합니다. 값 하나&
입력할 경우 X와 Y를 동일한 값으&#xb85
설정합니다. 자세한 내용은 월드
파일 을 참조하십시오.

Example 1:

```
-- 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:
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

Example 2:

[ST_GeoReference](#), [ST_SetGeoReference](#), [ST_SkewX](#), [ST_SkewY](#)

12.7.5 ST_SetSRID

ST_SetSRID — Set the SRID of a raster. **SRID** is the spatial reference system identifier. **spatial_ref_sys** is the name of the spatial reference system table.

Synopsis

raster **ST_SetSRID**(raster rast, integer srid);

Example 1:

```
-- Example 1:
UPDATE dummy_rast SET rast = ST_SetSRID(rast, 4326) 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;
```

Note

이 함수는 래스터를 어떤
 식으로든 변환하지 않습니다.
 현재 좌표 참조 시스템의
 공간 참조를 정의하는
 메타데이터를 설정할 뿐입니다.
 향후 변환 작업을 준비하는 데
 유용합니다.

참고

Section 4.5, [ST_SRID](#)

12.7.6 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);

설명

Set the value of the upper left corner of raster to the projected X and Y coordinates

예시

```
SELECT ST_SetUpperLeft (rast, -71.01, 42.37)
FROM dummy_rast
WHERE rid = 2;
```

참고

[ST_UpperLeftX](#), [ST_UpperLeftY](#)

12.7.7 ST_Resample

ST_Resample — 특정 리샘플링 알고리즘,새&#xb
 차원,임의의 그리드 모서리,그리&#
 또 다른 래스터에서 정의되거나
 빌려온 래스터 지리참조 속성들&#
 집합을 이용해서 래스터를 리샘&#

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 preci-
 sion 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);

SRID and NearestNeighbor

SRID and NearestNeighbor. The SRID is a 4-part integer: (width & height), (gridx & gridy), (scaley, skewx & skewy). The NearestNeighbor method is used to find the nearest neighbor of a point in a table. The SRID is a 4-part integer: (width & height), (gridx & gridy), (scaley, skewx & skewy). The NearestNeighbor method is used to find the nearest neighbor of a point in a table.

NearestNeighbor. The SRID is a 4-part integer: (width & height), (gridx & gridy), (scaley, skewx & skewy). The NearestNeighbor method is used to find the nearest neighbor of a point in a table.

maxerr. The SRID is a 4-part integer: (width & height), (gridx & gridy), (scaley, skewx & skewy). The NearestNeighbor method is used to find the nearest neighbor of a point in a table.

**Note**

GDAL Warp resampling methods. The SRID is a 4-part integer: (width & height), (gridx & gridy), (scaley, skewx & skewy). The NearestNeighbor method is used to find the nearest neighbor of a point in a table.

GDAL 2.0.0. The SRID is a 4-part integer: (width & height), (gridx & gridy), (scaley, skewx & skewy). The NearestNeighbor method is used to find the nearest neighbor of a point in a table.

GDAL 2.1.0. The SRID is a 4-part integer: (width & height), (gridx & gridy), (scaley, skewx & skewy). The NearestNeighbor method is used to find the nearest neighbor of a point in a table.

ST_MakeEnvelope

```
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
```

ST_Rescale, ST_Resize, ST_Transform

ST_Rescale, ST_Resize, ST_Transform

12.7.8 ST_Rescale

ST_Rescale — `ST_Rescale(raster rast, double precision scalex, text algorithm=NearestNeighbour, double precision maxerr=0.125);`
`ST_Rescale(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeighbour, double precision maxerr=0.125);`

Synopsis

`raster ST_Rescale(raster rast, double precision scalex, 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);`

ST_Rescale, ST_Resize, ST_Transform

`ST_Rescale(raster rast, double precision scalex, text algorithm=NearestNeighbour, double precision maxerr=0.125);`
`ST_Rescale(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeighbour, double precision maxerr=0.125);`

`scalex` and `scaley` define the new pixel size. `scaley` must often be negative to get well oriented raster.

`ST_Rescale(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeighbour, double precision maxerr=0.125);`
`ST_Rescale(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeighbour, double precision maxerr=0.125);`

`maxerr` is the threshold for transformation approximation by the resampling algorithm (in pixel units). A default of 0.125 is used if no `maxerr` is specified, which is the same value used in GDAL `gdalwarp` utility. If set to zero, no approximation takes place.



Note

`ST_Rescale(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeighbour, double precision maxerr=0.125);`
`ST_Rescale(raster rast, double precision scalex, double precision scaley, text algorithm=NearestNeighbour, double precision maxerr=0.125);`

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);

Parameters

rast: raster to be resampled.
skewxy: skew factor for the xy plane.
skewx: skew factor for the x plane.
skewy: skew factor for the y plane.
algorithm: resampling method. Valid values are: NearestNeighbour, Bilinear, Cubic, CubicSpline, Lanczos.
maxerr: maximum error allowed for the resampling process.

Return Value: A raster of the same type as the input raster, but with the specified skew factors applied.
Notes: The skew factors are applied to the raster coordinates. The skew factor for the xy plane is applied to the x and y coordinates. The skew factor for the x plane is applied to the x coordinate. The skew factor for the y plane is applied to the y coordinate.



Note

GDAL Warp resampling methods



Note

ST_Reskew and **ST_SetSkew** are aliases for **ST_Resample** with the **skewxy** and **skewx** parameters set to 0.0 and 0.0015 respectively. The **ST_Resample** function is the preferred method for resampling a raster.

2.0.0: **ST_Resample**(raster rast, double precision skewxy, text algorithm=NearestNeighbour, double precision maxerr=0.125);
 GDAL 1.6.1: **ST_Resample**(raster rast, double precision skewx, double precision skewy, text algorithm=NearestNeighbour, double precision maxerr=0.125);

ST_Resample(raster rast, double precision skewx, double precision skewy, text algorithm=NearestNeighbour, double precision maxerr=0.125);

Examples

Example 1: Resample a raster with a skew factor of 0.0015 for the x plane.

```
-- 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
```

ST_SnapToGrid

ST_Resample, ST_Rescale, ST_SetSkew, ST_SetRotation, ST_SkewX, ST_SkewY, ST_Transform

12.7.10 ST_SnapToGrid

ST_SnapToGrid — Snap a raster to a grid. The grid is defined by the gridx and gridy parameters. The algorithm used for snapping is defined by the text algorithm parameter. The maximum error is defined by the maxerr parameter. The scalex and scaley parameters are used to resample the raster to the new grid. The text algorithm parameter can be set to NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos, or NearestNeighbor. The maxerr parameter can be set to 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216, 33554432, 67108864, 134217728, 268435456, 536870912, 1073741824, 2147483648, 4294967296, 8589934592, 17179869184, 34359738368, 68719476736, 137438953472, 274877906944, 549755813888, 1099511627776, 2199023255552, 4398046511104, 8796093022208, 17592186044416, 35184372088832, 70368744177664, 140737488355328, 281474976710656, 562949953421312, 1125899906842624, 2251799813685248, 4503599627370496, 9007199254740992, 18014398509481984, 36028797018963968, 72057594037927936, 144115188075855872, 288230376151711744, 576460752303423488, 1152921504606846976, 2305843009213693952, 4611686018427387904, 9223372036854775808, 18446744073709551616, 36893488147419103232, 73786976294838206464, 147573952589676412928, 295147905179352825856, 590295810358705651712, 1180591620717411303424, 2361183241434822606848, 4722366482869645213696, 9444732965739290427392, 18889465931478580854784, 37778931862957161709568, 75557863725914323419136, 151115727451828646838272, 302231454903657293676544, 604462909807314587353088, 1208925819614629174706176, 2417851639229258349412352, 4835703278458516698824704, 9671406556917033397649408, 19342813113834066795298816, 38685626227668133590597632, 77371252455336267181195264, 154742504910672534362390528, 309485009821345068724781056, 618970019642690137449562112, 1237940039285380274899124224, 2475880078570760549798248448, 4951760157141521099596496896, 9903520314283042199192993792, 19807040628566084398385987584, 39614081257132168796771975168, 79228162514264337593543950336, 158456325028528675187087900672, 316912650057057350374175801344, 633825300114114700748351602688, 1267650600228229401496703205376, 2535301200456458802993406410752, 5070602400912917605986812821504, 10141204801825835211973625643008, 20282409603651670423947251286016, 40564819207303340847894502572032, 81129638414606681695789005144064, 162259276829213363391578010288128, 324518553658426726783156020576256, 649037107316853453566312041152512, 1298074214633706907132624082305024, 2596148429267413814265248164610048, 5192296858534827628530496329220096, 10384593717069655257060992658440192, 20769187434139310514121985316880384, 41538374868278621028243970633760768, 83076749736557242056487941267521536, 166153499473114484112975882535043072, 332306998946228968225951765070086144, 664613997892457936451903530140172288, 1329227995784915872903807060280344576, 2658455991569831745807614120560689152, 5316911983139663491615228241121378304, 10633823966279326983230456482242756608, 21267647932558653966460912964485513216, 42535295865117307932921825928971026432, 85070591730234615865843651857942052864, 170141183460469231731687303715884105728, 340282366920938463463374607431768211456, 680564733841876926926749214863536422912, 1361129467683753853853498429727072845824, 2722258935367507707706996859454145691648, 5444517870735015415413993718908291383296, 10889035741470030830827987437816582766592, 21778071482940061661655974875633165533184, 43556142965880123323311949751266331066368, 87112285931760246646623899502532662132736, 174224571863520493293247799005065324265472, 348449143727040986586495598010130648530944, 696898287454081973172991196020261297061888, 1393796574908163946345982392040522594123776, 2787593149816327892691964784081045188247552, 5575186299632655785383929568162090376495104, 11150372599265311570767859136324180752990208, 22300745198530623141535718272648361505980416, 44601490397061246283071436545296723011960832, 89202980794122492566142873090593446023921664, 178405961588244985132285746181186892047843328, 356811923176489970264571492362373784095686656, 713623846352979940529142984724747568191373312, 1427247692705959881058285969449495136382746624, 2854495385411919762116571938898990272765493248, 5708990770823839524233143877797980545530986496, 11417981541647679048466287755595961091061972992, 22835963083295358096932575511191922182123945984, 45671926166590716193865151022383844364247891968, 91343852333181432387730302044767688728495783936, 182687704666362864775460604089535377456991567872, 365375409332725729550921208179070754913983135744, 730750818665451459101842416358141509827966271488, 1461501637330902918203684832716283019655932542976, 2923003274661805836407369665432566039311865085952, 5846006549323611672814739330865132078623730171904, 11692013098647223345629478661730264157247460343808, 23384026197294446691258957323460528314494920687616, 46768052394588893382517914646921056628989841375232, 93536104789177786765035829293842113257979682750464, 187072209578355573530071658587684226515959365500928, 374144419156711147060143317175368453031918731001856, 748288838313422294120286634350736906063837462003712, 1496577676626844588240573268701473812127674924007424, 2993155353253689176481146537402947624255349848014848, 5986310706507378352962293074805895248510699696029696, 11972621413014756705924586149611790497021399392059392, 23945242826029513411849172299223580994042798784118784, 47890485652059026823698344598447161988085597568237568, 95780971304118053647396689196894323976171195136475136, 191561942608236107294793378393788647952342390272950272, 383123885216472214589586756787577295904684780545900544, 766247770432944429179173513575154591809369561091801088, 1532495540865888858358347027150309183618739122183602176, 3064991081731777716716694054300618367237478244367204352, 6129982163463555433433388108601236734474956488734408704, 12259964326927110866866776217202473468949912977468817408, 24519928653854221733733552434404946937899825954937634816, 49039857307708443467467104868809893875799651909875269632, 98079714615416886934934209737619787751599303819750539264, 196159429230833773869868419475239575503198607639501078528, 392318858461667547739736838950479151006397215279002157056, 784637716923335095479473677900958302012794430558004314112, 1569275433846670190958947355801916604025588861116008628224, 3138550867693340381917894711603833208051177722232017256448, 6277101735386680763835789423207666416102355444464034512896, 12554203470773361527671578846415332832204710888928069025792, 25108406941546723055343157692830665664409421777856138051584, 50216813883093446110686315385661331328818843555712276103168, 100433627766186892221372630771322662657637687111424552206336, 200867255532373784442745261542645325315275374222849104412672, 401734511064747568885490523085290650630550748445698208825344, 803469022129495137770981046170581301261101496891396417650688, 1606938044258990275541962092341162602522202993782792835301376, 3213876088517980551083924184682325205044405987565585670602752, 6427752177035961102167848369364650410088811975131171341205504, 12855504354071922204335696738729300820177623950262342682411008, 25711008708143844408671393477458601640355247900524685364822016, 51422017416287688817342786954917203280710495801049370729644032, 102844034832575377634685573909834406561420991602098741459288064, 205688069665150755269371147819668813122841983204197482918576128, 411376139330301510538742295639337626245683966408394965837152256, 822752278660603021077484591278675252491367932816789931674304512, 1645504557321206042154969182557350504982735865633579863348609024, 3291009114642412084309938365114701009965471731267159726697218048, 6582018229284824168619876730229402019930943462534319453394436096, 13164036458569648337239753460458804039861886925068638906788872192, 26328072917139296674479506920917608079723773850137277813577744384, 52656145834278593348959013841835216159447547700274555627155488768, 105312291668557186697918027683670432318895095400549111254310977536, 210624583337114373395836055367340864637790190801098222508621955072, 421249166674228746791672110734681729275580381602196445017243910144, 842498333348457493583344221469363458551160763204392890034487820288, 1684996666696914987166688442938726917102321526408785780068975640576, 3369993333393829974333376885877453834204643052817571560137951281152, 6739986666787659948666753771754907668409286105635143120275902562304, 13479973333575319897333507543509815336818572211270286240551805124608, 26959946667150639794667015087019630673637144422540572481103610249216, 53919893334301279589334030174039261347274288845081144962207220498432, 107839786668602559178668060348078522694548577690162289924414440996864, 215679573337205118357336120696157045389097155380324579848828881993728, 431359146674410236714672241392314090778194310760649159697657763987456, 862718293348820473429344482784628181556388621521298319395315527974912, 1725436586697640946858688965569256363112777243042596638790631055949824, 3450873173395281893717377931138512726225554486085193277581262111899648, 6901746346790563787434755862277025452451108972170386555162524223799296, 13803492693581127574869511724554050904902217944340773110325048447598592, 27606985387162255149739023449108101809804435888681546220650096895197184, 55213970774324510299478046898216203619608871777363092441300193790394368, 110427941548649020598956093796432407239217743554726184882600387580788736, 220855883097298041197912187592864814478435487109452369765200775161577472, 441711766194596082395824375185729628956870974218904739530401550323154944, 883423532389192164791648750371459257913741948437809479060803100646309888, 1766847064778384329583297500742918515827483896875618958121606201292619776, 3533694129556768659166595001485837031654967793751237916243212402585239552, 7067388259113537318333190002971674063309935587502475832486424805170479104, 14134776518227074636666380005943348126619871175004951664972849610340958208, 28269553036454149273332760011886696253239742350009903329945699220681916416, 56539106072908298546665520023773392506479484700019806659891398441363832832, 113078212145816597093331040047546785012958969400039613319782796882727665664, 226156424291633194186662080095093570025917938800079226639565593765455331328, 452312848583266388373324160190187140051835877600158453279131187530910662656, 904625697166532776746648320380374280103671755200316906558262375061821325312, 1809251394333065553493296640760748560207343510400633813116524750123642650624, 3618502788666131106986593281521497120414687020801267626233049500247285301248, 7237005577332262213973186563042994240829374041602535252466099000494570602496, 14474011154664524427946373126085988481658748083205070504932198000989141204992, 28948022309329048855892746252171976963317496166410141009864396001978282409984, 57896044618658097711785492504343953926634992332820282019728792003956564819968, 115792089237316195423570985008687907853269984665640564039457584007913129639936, 231584178474632390847141970017375815706539969331281128078915168015826259279872, 463168356949264781694283940034751631413079938662562256157830336031652518559744, 926336713898529563388567880069503262826159877325124512315660672063305037119488, 1852673427797059126777135760139006525652319754650249024631321344126610074238976, 3705346855594118253554271520278013051304639509300498049262642688253220148477952, 7410693711188236507108543040556026102609279018600996098525285376506440296955904, 14821387422376473014217086081112052205218558037201992197050570753012880593911808, 29642774844752946028434172162224104410437116074403984394101141506025761187823616, 59285549689505892056868344324448208820874232148807968788202283012051522375647232, 118571099379011784113736688648896417641748464297615937576404566024103044751294464, 237142198758023568227473377297792835283496928595231875152809132048206089502588928, 474284397516047136454946754595585670566993857190463750305618264096412179005177856, 948568795032094272909893509191171341133987714380927500611236528192824358010355712, 1897137590064188545819787018382342682267975428761855001222473056385648716020711424, 3794275180128377091639574036764685364535950857523710

maxerr=0.125, resampling=nearestneighbor, srid=31466, dstnodata=<boolean>, dstallowedskew=<float>);

maxerr=0.125, resampling=nearestneighbor, srid=31466, dstnodata=<boolean>, dstallowedskew=<float>);

**Note**

GDAL Warp resampling methods

**Note**

ST_Resample

GDAL 1.6.1

2.1.0

ST_SnapToGrid

ST_SnapToGrid

```
-- 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
```

ST_Resample

ST_Resample, ST_Rescale, ST_UpperLeftX, ST_UpperLeftY

12.7.11 ST_Resize

ST_Resize —

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);

설명

래스터의 크기를 새 너비/높이로
 조정합니다. 픽셀의 정확한 개수
 또는 입력 래스터의 너비/높이의
 백분율로 새 너비/높이를 설정할
 수 있습니다. 새 래스터의 범위가
 입력 래스터의 범위와 동일할 것의

NearestNeighbor(영국 또는 미국 철자), Bilinear, Cubic, Cubic-
 Spline 또는 Lanczos 리샘플링 알고리즘을
 이용해서 새 픽셀 값을 계산합니의
 기본값은 가장 빠르지만 보간의
 질은 가장 낮은 NearestNeighbor입니다.

변종 1은 출력 레이어의 실제 너비/높이의
 입력받습니다.

변종 2는 입력 래스터의 너비/높이의
 백분율을 나타내는 0과 1 사이의 소의
 입력받습니다.

변종 3은 출력 래스터의 실제 너비/높이의
 또는 입력 래스터의 너비/높이의
 백분율을 나타내는 문자열("20%")을
 입력받습니다.

2.1.0 버전부터 사용할 수 있습니다.
 GDAL 1.6.1 이상 버전이 필요합니다.

예시

```
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 | ↵
      1
    2 |           0 |           0 |   500 |   100 |       1 |      -1 |       0 |       0 |     0 | ↵
      1
    3 |           0 |           0 |   250 |   900 |       1 |      -1 |       0 |       0 |     0 | ↵
      1

(3 rows)
```

స고

ST_Resample, ST_Rescale, ST_Reskew, ST_SnapToGrid

12.7.12 ST_Transform

ST_Transform — `Ռ려진 공간 참조 시스템의 래스터를 지정한 리샘플링 알고의통해 또 다른 알려진 공간 참조 시재투영합니다.` NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos `Ռ고의이용할 수 있습니다.` `기본값은 Near-` estNeighbor `입니다.`

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);
```

설&#xba85:

```
&#xc54c;&#xb824;&#xc9c4; &#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#xc2a4;&#xd15c;&#xc758; &#xb798;&#xc2a4;&#x
&#xc9c0;&#xc815;&#xd55c; &#xd53d;&#xc140; &#xc65c;&#xace1;(pixel warp) &#xc54c;&#xace0;&#xb9ac;&#xc998;&#xc744;
&#xd1b5;&#xd574; &#xb610; &#xb2e4;&#xb978; &#xc54c;&#xb824;&#xc9c4; &#xacf5;&#xac04; &#xcc38;&#xc870; &#xc2dc;&#x
&#xc7ac;&#xd22c;&#xc601;&#xd569;&#xb2c8;&#xb2e4;. &#xb530;&#xb85c; &#xc54c;&#xace0;&#xb9ac;&#xc998;&#xc744;
&#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; &#xae30;&#xbcf8;&#xac12;&#xc740; Near-
estNeighbor&#xc774;&#xba70;, maxerror&#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0;
&#xae30;&#xbcf8;&#xac12;&#xc740; &#xbc31;&#xbd84;&#xc728; 0.125&#xc785;&#xb2c8;&#xb2e4;.
```

```
&#xc54c;&#xace0;&#xb9ac;&#xc998;&#xc635;&#xc158;&#xc5d0;&#xb294; 'NearestNeighbor', 'Bilinear', 'Cubic', 'Cubic-  
Spline', &#xadf8;&#xb9ac;&#xace0; 'Lanczos'&#xac00;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc790;&#xc138;&#xd55c;  
&#xb0b4;&#xc6a9;&#xc740; GDAL Warp resampling methods&#xb97c;&#xccc38;&#xc870;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

ST_Transform(ը수는종종 ST_SetSRID()와착&xac01;당합ݴ 실제로 래스터의 좌표를 한 공간 참조 시스템에서 또다른 공간 참조 시스템으로 변환시키른 (그리&xace0;픽셀 값을 리샘플링하는)

```
&#x18;&#xb4;,, ST_SetSRID()&#xb294;&#xb798;&#xc2a4;&#xd130;&#xc758; SRID &#xc2dd;&#xbcc4;&#xc790;&#xb97c;
&#xbcc0;&#xacbd;&#xd560; &#xbfd0;&#xc785;&#xb2c8;&#xb2e4;,.
&#xb2e4;&#xb978; &#xbcc0;&#xc885;&#xacfc; &#xb2ec;&#xb9ac;,, &#xbcc0;&#xc885; 3&#xc740; align to &#xd30c;&#xb77c;
&#xcc38;&#xc870; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc694;&#xad6c;&#xd569;&#xb2c8;&#xb2e4;,. &#xacb0;&#xacfc;
&#xb798;&#xc2a4;&#xd130;&#xb294; &#xcc38;&#xc870; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xacf5;&#xac04; &#xcc38;&#
&#xc2dc;&#xc2a4;&#xd15c;(SRID)&#xc73c;&#xb85c; &#xbcc0;&#xd658;&#xb420; &#xac83;&#xc774;&#xb470;,, (ST_SameAlign
= TRUE&#xc77c; &#xacbd;&#xc6b0;) &#xcc38;&#xc870; &#xb798;&#xc2a4;&#xd130;&#xc640; &#xb3d9;&#xc77c;&#xd558;&#
&#xc815;&#xb82c;&#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;,.

```

Note

```
&#xc0ac;&#xc6a9;&#xc790;&#xc758; &#xbcc0;&#xd658; &#xc9c0;&#xc6d0;&#xc774; &#xc81c;&#xb300;&#xb85c;
&#xb3d9;&#xc791; &#xd558;&#xc9c0; &#xc54a;&#xb294;&#xb2e4;&#xba74;, PROJSO
&#xd658;&#xacbd; &#xbcc0;&#xc218;&#xb97c; &#xc0ac;&#xc6a9;&#xc790; PostGIS&#xac00;
&#xc774;&#xc6a9;&#xd558;&#xace0; &#xc788;&#xb294; .so &#xb610;&#xb294; .dll &#xd22c;&#xc601;
&#xb77c;&#xc774;&#xbe0c;&#xb7ec;&#xb9ac;&#xb85c; &#xc124;&#xc815;&#xd574;&#xc57c; &#xd560;
&#xc218;&#xb3c4; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xd30c;&#xc77c;&#xba85;&#xb9cc;
&#xc9c0;&#xc815;&#xd574;&#xc8fc;&#xba74; &#xb429;&#xb2c8;&#xb2e4;. &#xc608;&#xb97c; &#xb4e4;&#xc5b4;
&#xc708;&#xb3c4;&#xc6b0;&#xc758; &#xacbd;&#xc6b0;, &#xc81c;&#xc5b4;&#xd310; -> &#xc2dc;&#xc2a4;&#xd15c;
-> &#xd658;&#xacbd; &#xbcc0;&#xc218;&#xb85c; &#xac00;&#xc11c; PROJSO &#xb77c;&#xb294;
&#xc2dc;&#xc2a4;&#xd15c; &#xbcc0;&#xc218;&#xb97c; &#xcd94;&#xac00;&#xd55c; &#xb2e4;&#xc74c;
(&#xc0ac;&#xc6a9;&#xc790;&#xac00; PROJ 4.6.1&#xc744; &#xc774;&#xc6a9;&#xd558;&#xace0;
&#xc788;&#xc744; &#xacbd;&#xc6b0;) &#xd574;&#xb2f9; &#xbcc0;&#xc218;&#xb97c; libproj.dll
&#xb85c; &#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;. &#xc774;&#xb807;&#xac8c;
&#xbcc0;&#xacbd;&#xd55c; &#xb2e4;&#xc74c; &#xc0ac;&#xc6a9;&#xc790;&#xc758; PostgreSQL
&#xc11c;&#xbe44;&#xc2a4;/&#xb370;&#xbaac;&#xc744; &#xc7ac;&#xc2dc;&#xc791;&#xd574;&#xc57c; &#xd560;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```



Warning

When transforming a coverage of tiles, you almost always want to use a reference raster to insure same alignment and no gaps in your tiles as demonstrated in example: Variant 3.

2.0.0 버전&#bd80;터 사용할 수 있습니다,
GDAL 1.6.1 이상 버전이 필요합니다.

```
&#xac1c;&#xc120;&#xc0ac;&#xd56d;: 2.1.0&#xbc84;&#xc804;&#xc5d0;&#xc11c; ST_Transform(rast, alignto) &#xbcc0;&#xc885;&#xcd94;&#xac00;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예&#xc2dc:

```
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

 ST_MakeEmptyRaster(2, 2, -500000, 600000, 100, -100, 0, 0, 2163), 1, '16BUI', 1, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499800, 600000, 100, -100, 0, 0, 2163), 1, '16BUI', 2, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499600, 600000, 100, -100, 0, 0, 2163), 1, '16BUI', 3, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -500000, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 10, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499800, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 20, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499600, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 30, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -500000, 599600, 100, -100, 0, 0, 2163), 1, '16BUI', 100, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499800, 599600, 100, -100, 0, 0, 2163), 1, '16BUI', 200, 0) AS rast UNION ALL 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	 WGS84	 ST_MakeEmptyRaster(2, 2, -500000, 600000, 100, -100, 0, 0, 2163), 1, '16BUI', 1, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499800, 600000, 100, -100, 0, 0, 2163), 1, '16BUI', 2, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499600, 600000, 100, -100, 0, 0, 2163), 1, '16BUI', 3, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -500000, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 10, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499800, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 20, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499600, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 30, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -500000, 599600, 100, -100, 0, 0, 2163), 1, '16BUI', 100, 0) AS rast UNION ALL ST_MakeEmptyRaster(2, 2, -499800, 599600, 100, -100, 0, 0, 2163), 1, '16BUI', 200, 0) AS rast UNION ALL 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

ST_Transform(raster, srid)

ST_Transform(raster, srid) returns a raster with the same data as the input raster, but with the coordinate system transformed to the specified SRID. The input raster must be in a geographic coordinate system (SRID 4326 or 4327).

```

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

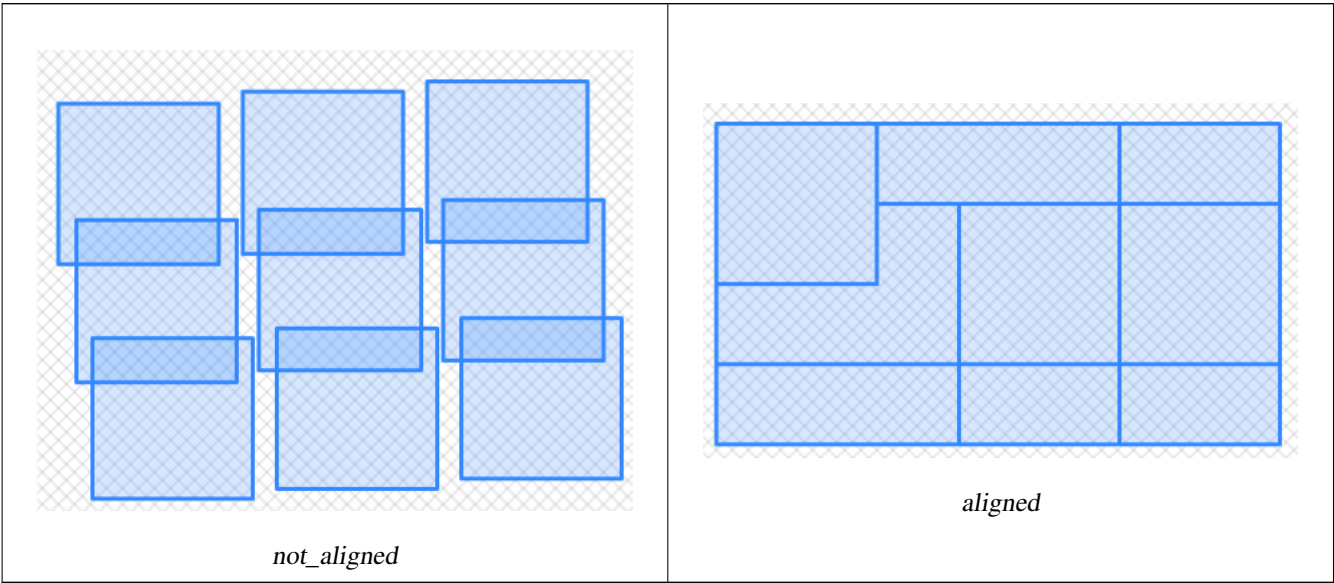
  SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, -500000, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 10, 0) AS rast UNION ALL
  SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499800, 599800, 100, -100, 0, 0, 2163), 1, '16BUI', 20, 0) AS rast UNION ALL
  SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, -499600, 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
```



[ST_Transform](#), [ST_SetSRID](#)

12.8 [ST_SameAlignment](#)

12.8.1 [ST_SetBandNoDataValue](#)

[ST_SetBandNoDataValue](#) — [NODATA](#) [ST_SetBandNoDataValue](#) (raster rast, double precision nodatavalue);
[ST_SetBandNoDataValue](#) (raster rast, integer band, double precision nodatavalue, boolean forcechecking=false);

Synopsis

raster [ST_SetBandNoDataValue](#)(raster rast, double precision nodatavalue);
raster [ST_SetBandNoDataValue](#)(raster rast, integer band, double precision nodatavalue, boolean forcechecking=false);

설명

```
&#xbc34;&#xb4dc;&#xc5d0;&#xc11c; NODATA&#xb97c; &#xb098;&#xd0c0;&#xb0b4;&#xb294; &#xac12;&#xc744; &#xc124;&#  
&#xbc34;&#xb4dc;&#xb97c; &#xb530;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0;  
&#xbc34;&#xb4dc; 1&#xb85c; &#xac00;&#xc815;&#xd569;&#xb2c8;&#xb2e4;. &#xc774; &#xd568;&#xc218;&#xb294;  
ST_Polygon, ST_DumpAsPolygons, &#xadf8;&#xb9ac;&#xace0; ST_PixelAs...() &#xd568;&#xc218;&#xc758; &#xacb0;&#xacfc;&  
&#xc601;&#xd5a5;&#xc744; &#xbbf8;&#xce69;&#xb2c8;&#xb2e4;.
```

예시

```
-- &#xcccab; &#xabc88;&#xc9f8; &#xbc34;&#xb4dc;&#xc758; NODATA &#xac12;&#xb9cc; &#xbcc0;&#xacbd;
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast,1, 254)
WHERE rid = 2;

-- &#xbc34;&#xb4dc; 1, 2, 3&#xc758; NODATA &#xbc34;&#xb4dc; &#xac12;&#xc744; &#xbcc0;&#xacbd;
UPDATE dummy_rast
    SET rast =
        ST_SetBandNoDataValue(
            ST_SetBandNoDataValue(
                ST_SetBandNoDataValue(
                    rast,1, 254)
                ,2,99),
            3,108)
    WHERE rid = 2;

-- NODATA &#xac12;&#xc744; &#xbaa8;&#xb450; &#xc81c;&#xac70;&#xd569;&#xb2c8;&#xb2e4;. ←
    &#xc774;&#xb807;&#xac8c; &#xd558;&#xba74; &#xbaa8;&#xb4e0; &#xacf5;&#xac04; ←
    &#xcc98;&#xb9ac; &#xd568;&#xc218;&#xac00; &#xbaa8;&#xb4e0; &#xd53d;&#xc140;&#xc744; ←
    &#xcc98;&#xb9ac;&#xd558;&#xac8c; &#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
UPDATE dummy_rast
    SET rast = ST_SetBandNoDataValue(rast,1, NULL)
WHERE rid = 2;
```

స고

ST_BandNoDataValue, ST_NumBands

12.8.2 ST SetBandsNoData

ST_SetBandIsNoData — [%s34; %s4dc; %s758; isnodata %s50c; %sb798; %xadf8; %sb97c; %sxc38; %s73c; %sb85c; %sxc124; %sxc815; %xd569; %sb2c8; %sb2e4;.](#)

Synopsis

```
raster ST_SetBandIsNoData(raster rast, integer band=1);
```

설&#xba85:

㰴͍͘ isodata ͐͸㷸Η㰸ͳ΅𼄤㠕𽕘㧀㕊㝄𺲽㚰㰴͍Η㔰΅𼄤㠕𽕘㧀㕊㝄𺲽㚰㰴͍΅㫀㠕𽕩𻋈𻋤. 㵐㭸㷸㫀㧀㗬𺲨㧀𻊔𺲽㚰㗐𻧌㝴𽕨𼈘㮗𽘸𽕩𻋈𻋤. 㦉. 𻧈㧀𻧉㝸𼈘㗐㰸㝄𼄤&

ST_BandIsNoData
 함수를호출해서나온결과물이달라질때말입니다.

2.0.0 버전부터사용할수있습니다.

예시

```
-- &#xb798;&#xc2a4;&#xd130;&#xc5f4;&#xd55c;&#xac1c;&#xb97c;&#xac00;&#xc9c4;&#xc00;&#xc9dc;&#xd14c;&#xc774;&#xbe14;&#xc744;&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.  
create table dummy_rast (rid integer, rast raster);  
  
-- &#xbc34;&#xb4dc;&#xb450;&#xac1c;&#xc640;&#xd53d;&#xc140;/&#xbc34;&#xb4dc;&#xd558;&#xb098;&#xb97c;&#xac00;&#xc9c4;&#xb798;&#xc2a4;&#xd130;&#xb97c;&#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;.&#xccab;&#xbc88;&#xc9f8;&#xbc34;&#xb4dc;&#xc758; nodatavalue = pixel value = 3&#xc785;&#xb2c8;&#xb2e4;.  
-- &#xb450;&#xbc88;&#xc9f8;&#xbc34;&#xb4dc;&#xc758; nodatavalue = 13, pixel value = 4 &#xc785;&#xb2c8;&#xb2e4;.  
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 &#xac12;&#xc774;&#xcc38;&#xc73c;&#xb85c;, isnodata &#xac12;&#xc774;&#xcc38;&#xc774;&#xc5b4;&#xc57c;&#xd558;&#xb294;&#xb370;)&#xac70;&#xc9d3;&#xc73c;&#xb85c;&#xc124;&#xc815;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.  
||  
'2' -- &#xccab;&#xbc88;&#xc9f8;&#xbc34;&#xb4dc;&#xc720;&#xd615;(4BUI)  
||  
'03' -- novalue==3  
||  
'03' -- pixel(0,0)==3 (same that nodata)  
||  
'0' -- hasnodatavalue &#xac12;&#xc774;&#xac70;&#xc9d3;&#xc73c;&#xb85c;&#xc124;&#xc815;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.  
||  
'5' -- &#xb450;&#xbc88;&#xc9f8;&#xbc34;&#xb4dc;&#xc720;&#xd615;(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

-- isnodata &#xd50c;&#xb798;&#xadf8;&#xac00;  ←
  &#xc9c0;&#xc800;&#xbd84;&#xd569;&#xb2c8;&#xb2e4;. &#xc774;  ←
  &#xd50c;&#xb798;&#xadf8;&#xb97c; &#xcc38;&#xc73c;&#xb85c;  ←
  &#xc124;&#xc815;&#xd558;&#xaca0;&#xc2b5;&#xb2c8;&#xb2e4;.
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

```

참고

ST_BandNoDataValue, ST_NumBands, ST_SetBandNoDataValue, ST_BandIsNoData

12.8.3 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);

설명

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

예시

```
WITH foo AS (
    SELECT
        ST_AddBand(NULL::raster, '/home/pele/devel/geo/postgis-git/raster/test/regress/ ↵
        loader/Projected.tif', NULL::int[]) AS rast
)
SELECT
    1 AS query,
    *
FROM ST_BandMetadata(
    (SELECT rast FROM foo),
    ARRAY[1,3,2]::int[]
)
UNION ALL
SELECT
    2,
    *
FROM ST_BandMetadata(
    (
        SELECT
            ST_SetBandPath(
                rast,
                2,
                '/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected2.tif ↵
                ',
                1
            ) AS rast
        FROM foo
    ),
    ARRAY[1,3,2]::int[]
)
ORDER BY 1, 2;
```

query	bandnum	pixeltype	nodatavalue	isoutdb	path
-----+-----+-----+-----+-----+-----					
1	1	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
1	2	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
1	3	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
2	1	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif
2	2	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected2.tif
2	3	8BUI		t	/home/pele/devel/geo/postgis-git/raster/test/regress/loader/Projected.tif

ST_BandMetaData, ST_SetBandIndex

12.8.4 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);

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

SQL Function Usage

```
WITH foo AS (
  SELECT
    ST_AddBand(NULL::raster, '/home/pele/devel/geo/postgis-git/raster/test/regress/
      loader/Projected.tif', NULL::int[]) AS rast
)
SELECT
  1 AS query,
  *
FROM ST_BandMetadata(
  (SELECT rast FROM foo),
  ARRAY[1,3,2]::int[]
)
UNION ALL
SELECT
  2,
  *
FROM ST_BandMetadata(
  (
    SELECT
      ST_SetBandIndex(
        rast,
        2,
        1
      ) AS rast
    FROM foo
  ),
  ARRAY[1,3,2]::int[]
)
ORDER BY 1, 2;
```

query	bandnum	pixeltype	nodatavalue	isoutdb	path	outdbbandnum

1		1		8BUI			t		/home/pele/devel/geo/postgis-git/	↔
				raster/test/regress/loader/Projected.tif					1	
1		2		8BUI			t		/home/pele/devel/geo/postgis-git/	↔
				raster/test/regress/loader/Projected.tif					2	
1		3		8BUI			t		/home/pele/devel/geo/postgis-git/	↔
				raster/test/regress/loader/Projected.tif					3	
2		1		8BUI			t		/home/pele/devel/geo/postgis-git/	↔
				raster/test/regress/loader/Projected.tif					1	
2		2		8BUI			t		/home/pele/devel/geo/postgis-git/	↔
raster/test/regress/loader/Projected.tif									1	
2		3		8BUI			t		/home/pele/devel/geo/postgis-git/	↔
				raster/test/regress/loader/Projected.tif					3	

స고

ST_BandMetaData, ST_SetBandPath

12.9 **래스터 밴드 통계 &#bc0f;**
&#bd84;석

12.9.1 ST_Count

ST_Count — 래스터 또는 래스터 커버리지 입력 밴드에 있는 픽셀 개수를 반는 드를 따로 설정하지 않을 경우 기본값은 밴드 1입니다. exclude_nodata_value를 참으로 설정할 경우. NODATA 값이 아닌 픽셀의 개수만 반환할 것입니다.

Synopsis

```
boolean ST_BandIsNoData(raster rast, integer band, boolean forceChecking=true);
```

```
boolean ST_BandIsNoData(raster rast, boolean forceChecking=true);
```

설&#xba85:

래스터 또는 래스터 쳤버리지의
입력 밴드에 있는 픽셀 개수를 반는
력 를 따로 설정하지 않을 경우
nband 의 기본값은 1입니다.

Note



Note!

```
exclude_nodata_value      &#xb97c;      &#xcc38;&#xc73c;&#xb85c;      &#xc124;&#xc815;&#xd560;
&#xacbd;&#xc6b0;,      &#xb798;&#xc2a4;&#xd130;&#xc758;      nodata      &#xac12;&#xc774;
&#xc544;&#xb2cc;      &#xd53d;&#xc140;&#xc758;      &#xac1c;&#xc218;&#xb9cc;      &#xbc18;&#xd658;&#xd560;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xbaa8;&#xb4e0; &#xd53d;&#xc140;&#xc758; &#xac1c;&#xc218;&#xb97c;
&#xad6c;&#xd558;&#xb824;&#xba74; exclude_nodata_value &#xb97c; &#xac70;&#xc9d3;&#xc73c;&#xb85c;
&#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

2.2.0.಄전부터 더 이상 ST_Count(rastertable, rastercolumn, ...) 변&#xc88함수를 지원하지 않습니다. 대신 **ST_CountAgg** 함수를 이용하십시오.

2.0.0 ಄전부터 사용할 수 있습니다.

예시

```
-- &#xcccab; &#xbc88;&#xc9f8; &#xd568;&#xc218;&#xb294; &#xac12;&#xc774; 249&#xac00;  ←
&#xc544;&#xb2cc; &#xbaa8;&#xb4e0; &#xd53d;&#xc140;&#xc758; &#xac1c;&#xc218;&#xb97c;  ←
&#xc9d1;&#xacc4;&#xd558;&#xace0;, &#xb450; &#xbc88;&#xc9f8; &#xd568;&#xc218;&#xb294;  ←
&#xbaa8;&#xb4e0; &#xd53d;&#xc140;&#xc758; &#xac1c;&#xc218;&#xb97c;  ←
&#xc9d1;&#xacc4;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. --
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
```

స고

ST_CountAgg, ST_SummaryStats, ST_SetBandNoDataValue

12.9.2 ST_CountAgg

ST_CountAgg — ಅ합 함수입니다. 래스터 집&#x
입력 ଴드에 있는 픽셀 개수를 򳰘&#x
଴드를 따로 설정하지 않을 경우
기본값은 ଴드 1입니다. exclude_nodata_value를
참으로 설정할 경우, NODATA 값이 아닌
픽셀의 개수만 򳰘환할 것입니다.

Synopsis

```
bigint ST_CountAgg(setof raster rast, integer nband, boolean exclude_nodata_value, double precision sample_percent);
bigint ST_CountAgg(setof raster rast, integer nband, boolean exclude_nodata_value);
bigint ST_CountAgg(setof raster rast, boolean exclude_nodata_value);
```

설명

래스터 집합의 입력 밴드에 있는
픽셀 개수를 반환합니다. 밴드를
따로 설정하지 않을 경우 nband 의 기&#xb
1입니다.

```
exclude_nodata_value &#xb97c; &#xcc38; &#xc73c; &#xb85c; &#xc124; &#xc815; &#xd560; &#xacbd; &#xc6b0;, &#xb798; &#  
nodata &#xac12; &#xc774; &#xc544; &#xb2cc; &#xd53d; &#xc140; &#xc758; &#xac1c; &#xc218; &#xb9cc; &#xbc18; &#xd658; &#x  
&#xac83; &#xc785; &#xb2c8; &#xb2e4;. &#xbaa8; &#xb4e0; &#xd53d; &#xc140; &#xc758; &#xac1c; &#xc218; &#xb97c; &#xad6c; &#x  
exclude_nodata_value &#xb97c; &#xac70; &#xc9d3; &#xc73c; &#xb85c; &#xc124; &#xc815; &#xd558; &#xc2ed; &#xc2dc; &#x
```

기본적으로 모든 픽셀을 샘플링&#x
것입니다 더 빠른 속도를 원한다&
sample_percent 를 0과 1 사이의 값으로 설정하

2.2.0 버전부터 사용할 수 있습니다.

예시

```

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)

```

స고

ST_Count, ST_SummaryStats, ST_SetBandNoDataValue

12.9.3 ST_Histogram

ST_Histogram — ่(bin; 히스토그램표시에서 수막대로나타나는 단위) 범위로 구래스터 또는 래스터 쳤버리지의 데이터 분포를 요약하는 레코드 집합을 반환합니다. 따로 설정하않을 경우 빈의 개수를 자동으로 계산합니다.

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);

Note**Note**

`nodata` `값이 아닌 픽셀 값만 처리합니다.` `ન든 픽셀의 개수를 구하려면 exclude_nodata_value 를 거짓으로 설정하십시오.`

width double precision[] width: `각 카테고리/&#be48;의 너&#be44;를나타내는 &#bc30;열입니다.` `&#be48; 개수가수보다 큰 경우 width를 &#bc18;&#bcf5;합니시.` `&#be48; 9개.` width `[a, b, c]는 [a, b, c, a, b, c, a, b, c]로 출력될것입니다.`

bins integer `분류 단계(breakout)의 개수.` `따로 설정경우 함수가 &#bc18;환하는 레코드의개수입니다.` `따로 설정하지 않의경우 분류 단계의 개수를 자동의계산합니다.`

right boolean `히스토그램을 왼쪽부터보단오른쪽부터(기본값) 계산합니다.` `X값을 평가하는 기준을 [a, b] 에서 (a, b] 로 변경합니다.`

Changed: 3.1.0 Removed ST_Histogram(table_name, column_name) variant.

2.0.0 `&#bc84;전부터 사용할 수 있습니다.`

`예시.` `단일 래스터 타일 - &#bc34;드 1, 2, 3에 대한 히스토그램을 계산하고 &#be48;의자동으로 계산합니다.`

```
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

ST_Histogram (raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] quantiles=NULL);

```
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)

```
-- ST_Histogram (raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] quantiles=NULL);
-- ST_Histogram (raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] quantiles=NULL);
```

```
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)

ST_Histogram

ST_Count, **ST_SummaryStats**, **ST_SummaryStatsAgg**

12.9.4 ST_Quantile

ST_Quantile (raster rast, integer nband=1, boolean exclude_nodata_value=true, double precision[] quantiles=NULL);

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, 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);

```

Quantile

ST_Quantile(raster rast, integer nband, double precision[] quantiles) returns a set of records, one for each quantile. The quantiles are calculated using the **ST_Quantile** function. The quantiles are calculated using the **ST_Quantile** function. The quantiles are calculated using the **ST_Quantile** function.



Note

exclude_nodata_value When set to true, the function will ignore nodata values when calculating the quantile. When set to false, the function will include nodata values in the calculation. The default value is true.

Changed: 3.1.0 Removed **ST_Quantile**(table_name, column_name) variant.

2.0.0 **ST_Quantile**(raster rast, integer nband, double precision quantile) returns a double precision value.

Quantile

```

UPDATE dummy_rast SET rast = ST_SetBandNoDataValue(rast,249) WHERE rid=2;
-- ST_Quantile(rast, ARRAY[0.25,0.75]) As pvq
-- FROM dummy_rast WHERE rid=2) As foo
-- ORDER BY (pvq).quantile;

```

```

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

```

```

-- ST_Quantile(rast, 0.75) As value
-- FROM dummy_rast WHERE rid=2;
-- ORDER BY (value);

```

```

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

ST_Count;

ST_Count, **ST_SummaryStats**, **ST_SummaryStatsAgg**, **ST_SetBandNoDataValue**

12.9.5 ST_SummaryStats

ST_SummaryStats — Returns a table with columns: count, sum, mean, stddev, min, max, and summarystats. The summarystats column is a text representation of the summary statistics. The table has columns: count, sum, mean, stddev, min, max, and summarystats. The summarystats column is a text representation of the summary statistics. The table has columns: count, sum, mean, stddev, min, max, and summarystats. The summarystats column is a text representation of the summary statistics.

Synopsis

```

raster ST_SetScale(raster rast, float8 xy);
raster ST_SetScale(raster rast, float8 x, float8 y);

```

ST_SetScale;

ST_SetScale — Returns a raster with the same geometry as the input raster, but with the specified scale. The scale is specified by the xy parameter, which is a point in the raster's coordinate system. The scale is specified by the x and y parameters, which are the x and y coordinates of the point. The scale is specified by the xy parameter, which is a point in the raster's coordinate system. The scale is specified by the x and y parameters, which are the x and y coordinates of the point. The scale is specified by the xy parameter, which is a point in the raster's coordinate system. The scale is specified by the x and y parameters, which are the x and y coordinates of the point.

**Note**

`nodata` `값이 아닌 픽셀 값만 처리합니다.` `ન든 픽셀의 개수를 구하려면 exclude_nodata_value 를 거짓으로 설정하십시오.`

**Note**

`ન든 픽셀을 샘플링할 것입니다.` `더 빠른 속도를 원한다면.` `sample_percent 를 1보다 작은 값으로 설정하십시오.`

2.2.0 `변종 함수를 지원하지 않습니다.` `대신 ST_SummaryStatsAgg 함수를 이용하십시오.`

2.0.0 `변종 함수를 지원하지 않습니다.`

`예시: 단일 래스터 타일`

```
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

`예시: 관심 건물과 교차하는 픽셀일 요약`

PostGIS `윈도우 64비트 변종 함수를 지원하지 않습니다.` `ન든 보일 건물들과 항공사진 타일 들(각각 건물레코드 102,000개.` `150x150 픽셀 크기의 타일 134,000개)이 처리하는 이 예시ఀ 574변리초 걸렸습니다.`

```
WITH
-- &#xad00;&#xc2ec; &#xd53c;&#xcc98;
  feat AS (SELECT gid As building_id, geom_26986 As geom FROM buildings AS b
            WHERE gid IN(100, 103,150)
            ),
-- builds&#xc758; &#xacbd;&#xacc4;&#xc120;&#xc5d0; &#xb9de;&#xcdb0; &#xb798;&#xc2a4;&#xd130; &#xd0c0;&#xc77c; &#xc758; &#xb3c4;&#xb4dc; 2&#xb97c; &#xc798;&#xb77c; &#xb0b8; &#xb2e4;&#xc74c;
-- &#xc774; &#xc798;&#xb77c; &#xb0b8; &#xc9c0;&#xc5ed;&#xc758; &#xd1b5;&#xacc4;&#xb97c; &#xc5bb;&#xc2b5;&#xb2c8;&#xb2e4; .
  b_stats AS
    (SELECT building_id, (stats).*
     FROM (SELECT building_id, ST_SummaryStats(ST_Clip(rast,2,geom)) As stats
           FROM aerials.boston
```

```

        INNER JOIN feat
        ON ST_Intersects (feat.geom,rast)
    ) As foo
)
--  &#xb9c8;&#xc9c0;&#xb9c9;&#xc73c;&#xb85c;  &#xd1b5;&xacc4;&#xb97c;  <-
    &#xc694;&#xc57d;&#xd569;&#xb2c8;&#xb2e4;.
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

```

예시: 래스터 커버리지

```

--  &#xac01; &#xbc34;&#xb4dc;&#xc5d0;  &#xb300;&#xd55c;  &#xd1b5;&xacc4;  --
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

--  &#xd14c;&#xc774;&#xbe14;&#xc758;  &#xacbd;&#xc6b0;:  &#xc0d8;&#xd50c;&#xb9c1;&#xc744;  100%  <-
    &#xbbf8;&#xb9cc;&#xc73c;&#xb85c;  &#xc124;&#xc815;&#xd558;&#xba74;  <-
    &#xc18d;&#xb3c4;&#xac00;  &#xd5a5;&#xc0c1;&#xb429;&#xb2c8;&#xb2e4;.
--  &#xc774;  &#xc608;&#xc2dc;&#xc5d0;&#xc11c;&#xb294;  25%&#xb85c;  <-
    &#xc124;&#xc815;&#xd574;&#xc11c;  &#xd6e8;&#xc52c;  &#xbe68;&#xb9ac;  &#xb2f5;&#xc744;  <-
    &#xbc18;&#xd658;&#xbc1b;&#xc2b5;&#xb2c8;&#xb2e4;.
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

```

참고

[summarystats](#), [ST_SummaryStatsAgg](#), [ST_Count](#), [ST_Clip](#)

12.9.6 ST_SummaryStatsAgg

`ST_SummaryStatsAgg` — `종합 함수입니다`, `래스터`, `집합의 입력 래스터 밴드의` count, sum, mean,

stddev, min, max로 이루어진 통계 요약을 반&#xd6
밴드를 따로 설정하지 않을 경우
밴드 1로 가정합니다.

Synopsis

```
summarystats ST_SummaryStatsAgg(setof raster rast, integer nband, boolean exclude_nodata_value, double precision sample_percent);
summarystats ST_SummaryStatsAgg(setof raster rast, boolean exclude_nodata_value, double precision sample_percent);
summarystats ST_SummaryStatsAgg(setof raster rast, integer nband, boolean exclude_nodata_value);
```

설명

```
&#xc785;&#xb825;&#xd55c;&#xb798;&#xc2a4;&#xd130;&#xbc34;&#xb4dc;&#xb610;&#xb294;&#xb798;&#xc2a4;&#xd130;&#xb610;&#xb294;&#xb798;&#xc2a4;&#xd130;&#xcee4;&#xbc84;&#xb9ac;&#xc9c0;&#xc758; count, sum, mean, stddev, min, max&#xb85c;&#xc774;&#xb8e8;&#xc5b4;&#xc9c4; summarystats&#xc744;&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;&#xbc34;&#xb4dc;&#xb97c;&#xb530;&#xb85c;&#xc124;&#xc815;&#xd558;&#xc9c0;&#xc54a;&#xc744;&#xacbd;&#xc6b0; nband&#xc758;&#xae30;&#xbcf8;&#xac12;&#xc740; 1&#xc785;&#xb2c8;&#xb2e4;.
```

Note



```
#&#xae30;&#xbcf8;&#xc801;&#xc73c;&#xb85c; nodata &#xac12;&#xc774; &#xc544;&#xb2cc; &#xd53d;&#xc140;
&#xac12;&#xb9cc; &#xcc98;&#xb9ac;&#xd569;&#xb2c8;&#xb2e4;. &#xbaa8;&#xb4e0; &#xd53d;&#xc140;&#xc758;
&#xac1c;&#xc218;&#xb97c; &#xad6c;&#xd558;&#xb824;&#xba74; exclude_nodata_value &#xb97c;
&#xac70;&#xc9d3;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

Note



```
&#xae30;&#xbcf8;&#xc801;&#xc73c;&#xb85c;      &#xbaa8;&#xb4e0;      &#xd53d;&#xc140;&#xc744;
&#xc0d8;&#xd50c;&#xb9c1;&#xd560;  &#xac83;&#xc785;&#xb2c8;&#xb2e4;.      &#xb354;  &#xbe60;&#xb978;
&#xc18d;&#xb3c4;&#xb97c;  &#xc6d0;&#xd55c;&#xb2e4;&#xba74;, sample_percent &#xb97c; 0&#xacfc; 1
&#xc0ac;&#xc774;&#xc758;&#xac12;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

2.2.0 버전부터 사용할 수 있습니다.

예시

```
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
    )
```



```
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

```
-- 249 250 251 252 253 254; 2 NODATA
-- 249 250 251 252 253 254; 2 NODATA
```

```
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

```
:
```

```
-- 249 250 251 252 253 254; 2 NODATA
-- 249 250 251 252 253 254; 2 NODATA
```

```
-- 249 250 251 252 253 254; 2 NODATA
-- 249 250 251 252 253 254; 2 NODATA
```

```
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

```
-- 249 250 251 252 253 254; 2 NODATA
-- 249 250 251 252 253 254; 2 NODATA
```

```

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

స고

ST_Count, ST_SetBandNoDataValue

12.10 Raster Inputs

12.10.1 ST_RastFromWKB

ST_RastFromWKB — Return a raster value from a Well-Known Binary (WKB) raster.

Synopsis

```
raster ST_RastFromWKB(bytea wkb);
```

설명

Given a Well-Known Binary (WKB) raster, return a raster.

Availability: 2.5.0

예시

[illegible]

ST_MakeEnvelope

[ST_MetaData](#), [ST_RastFromHexWKB](#), [ST_AsBinary/ST_AsWKB](#), [ST_AsHexWKB](#)

12.10.2 ST_RastFromHexWKB

ST_RastFromHexWKB — Return a raster value from a Hex representation of Well-Known Binary (WKB) raster.

Synopsis

raster **ST_RastFromHexWKB**(text wkb);

ST_RastFromHexWKB

Given a Well-Known Binary (WKB) raster in Hex representation, return a raster.

Availability: 2.5.0

ST_RastFromHexWKB

```
SELECT rid, (foo.md).*
FROM (SELECT rid, ST_MetaData(rast) As md
FROM dummy_rast) As foo;
```

rid	upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	
numbands										
1	0.5	0.5	10	20	2	3			0	
0	0	0								
2	3427927.75	5793244	5	5	0.05	-0.05			0	
0	0	3								

ST_MakeEnvelope

[ST_MetaData](#), [ST_RastFromWKB](#), [ST_AsBinary/ST_AsWKB](#), [ST_AsHexWKB](#)

12.11 ST_AsBinary/ST_AsWKB

12.11.1 ST_AsBinary/ST_AsWKB

ST_AsBinary/ST_AsWKB — Return the Well-Known Binary (WKB) representation of the raster.

Synopsis

bytea **ST_AsBinary**(raster rast, boolean outasin=FALSE);

bytea **ST_AsWKB**(raster rast, boolean outasin=FALSE);

설명

Returns the Binary representation of the raster. If `outasin` is TRUE, out-db bands are treated as in-db. Refer to `raster/doc/RFC2-WellKnownBinaryFormat` located in the PostGIS source folder for details of the representation.

이함수는데이터베이스로부터데이터를문자열표현식으로변않고추출하는바이너리커서에유용합니다.

Note

```
&#xae30;&#xbcfc8;&#xc801;&#xc73c;&#xb85c;,      WKB      &#xcd9c;&#xb825;&#xbb3c;&#xc740;      DB
&#xc678;&#xbd80;      &#xbc34;&#xb4dc;&#xb97c;      &#xac00;&#xb9ac;&#xd0a4;&#xb294;      &#xc678;&#xbd80;
&#xd30c;&#xc77c;      &#xacbd;&#xb85c;&#xb97c;      &#xb2f4;&#xace0;      &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
&#xd074;&#xb77c;&#xc774;&#xc5b8;&#xd2b8;&#xac00;      DB      &#xc678;&#xbd80;      &#xbc34;&#xb4dc;&#xc758;
&#xae30;&#xc800;      &#xb798;&#xc2a4;&#xd130;      &#xd30c;&#xc77c;&#xc5d0;      &#xc811;&#xadfc;&#xd558;&#xc9c0;
&#xbabb;      &#xd558;&#xb294;      &#xacbd;&#xc6b0;,      outasin      &#xc744;      &#xcc38;&#xc73c;&#xb85c;
&#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

개선 사항: 2.1.0버전에서 outasin 이 추가&#xb4

Enhanced: 2.5.0 Addition of ST_AsWKB

예시

```
SELECT ST_AsBinary(rast) As rastbin FROM dummy_rast WHERE rid=1;
```

rastbin

```
\001\000\000\000\000\000\000\000\000\000\000@ \000\000\000\000\000\000\010@ ←  
      \000\000\000\000\000\000\340? \000\000\000\000\000\000\340? \000\000\000\000\000\000\000\000\000\000
```

స고

ST_RastFromWKB, ST_AsHexWKB

12.11.2 ST AsHexWKB

ST_AsHexWKB — Return the Well-Known Binary (WKB) in Hex representation of the raster.

Synopsis

```
bytea ST_AsHexWKB(raster rast, boolean outasin=FALSE);
```

설명

Returns the Binary representation in Hex representation of the raster. If `outasin` is TRUE, out-db bands are treated as in-db. Refer to `raster/doc/RFC2-WellKnownBinaryFormat` located in the PostGIS source folder for details of the representation.

Note!

Note

By default, Hex WKB output contains the external file path for out-db bands. If the client does not have access to the raster file underlying an out-db band, set `outasin` to `TRUE`.

설명

래스터에서 선택한 밴드들을 단일 JPEG(Joint Photographic Exports Group) 이미지로 반환합니다.&br/>덜 흔한 래스터 유형으로 내보내을 할 경우 **ST_AsGDALRaster** 함수를 이용하십시을 밴드를 따로 설정하지 않거나,밴드를 또는 3개를 초과할 경우 &#xc-cab; 번째 밴드만 씁니다.밴드가 3개이 경우 밴드 3개를 모두 씁니다.이 함수에는 다음과 같은 많은 옵션을 가진 많은 변종이 있습니다.

- nband - 단일 밴드 내보내기를 위한 옵을
- nbands - 내보낼 밴드들의 배열입니다.(J
경우 3이 최대값입니다). 밴드의
순서는 RGB입니다.예를 들어 ARRAY[3,2,1]은
밴드 3을 빨간색,밴드 2를 초록색,
&#bc34;드 1을 파란색에 매핑한다는 &#xb73b
- quality - 1부터 100까지의 숫자입니다.숫자 높을수록 이미지가 선명해집니을
- options - JPEG에 대해 정의된 GDAL 옵션들의
텍스트 배열입니다(**ST_GDALDrivers**에서 JPEG에
대한 create_options를 살펴보십시오). JPEG의
경우,유효한 옵션은 PROGRESSIVE ON/OFF 보 기을
의이ૠ 0부터 100까지의 범위에서 설을
수 있는 QUALITY 입니다.자세한 내용은
GDAL 래스터 형식 옵션 을 స조하십을

2.0.0 버전부터 사용할 수 있습니다.
GDAL 1.6.0 이상 버전이 필요합니다.

예시: 출력

```
-- &#xcc98;&#xc74c; &#xbc34;&#xb4dc; 3&#xac1c;&#xb97c; 75% &#xd488;&#xc9c8;&#xb85c; &#xc2dc; &#xcd9c;&#xb825;
SELECT ST_AsJPEG(rast) As rastjpg
FROM dummy_rast WHERE rid=2;

-- &#xccab; &#xbc88;&#xc9f8; &#xbc34;&#xb4dc;&#xb9cc; 90% &#xd488;&#xc9c8;&#xb85c; &#xc2dc; &#xcd9c;&#xb825;
SELECT ST_AsJPEG(rast,1,90) As rastjpg
FROM dummy_rast WHERE rid=2;

-- &#xcc98;&#xc74c; &#xbc34;&#xb4dc; 3&#xac1c;&#xb97c; &#xcd9c;&#xb825; (&#xbc34;&#xb4dc; 2 &#xb97c; &#xbe68;&#xac15;,&#xbc34;&#xb4dc; 1&#xc744; &#xb179;&#xc0c9;,&#xc2dc; &#xbcf4;&#xb4dc; 3&#xc744; &#xd30c;&#xb791;&#xc73c;&#xb85c; &#xd558;&#xae0; PROGRESSIVE &#xc2dc; &#xc2dc; ON, &#xd488;&#xc9c8;&#xc740; 90%)
SELECT ST_AsJPEG(rast,ARRAY[2,1,3],ARRAY['QUALITY=90','PROGRESSIVE=ON']) As rastjpg
FROM dummy_rast WHERE rid=2;
```

참ૠ

Section 11.3, **ST_GDALDrivers**, **ST_AsGDALRaster**, **ST_AsPNG**, **ST_AsTIFF**

2.0.0 버전஀터 사용할 수 있습니다,
GDAL 1.6.0 이상 버전이 필요합니다.

예시

```
SELECT ST_AsPNG(rast) As rastpng
FROM dummy_rast WHERE rid=2;

-- &#xc98;&#xc74c; 3&#xac1c;&#xc758; &#xbc34;&#xb4dc;&#xb97c;  ↔
  &#xb0b4;&#xbcf4;&#xb0b4;&#xc11c; &#xbc34;&#xb4dc; 3&#xc744; &#xbe68;&#xac15;;  ↔
  &#xbc34;&#xb4dc; 1&#xc744; &#xb179;&#xc0c9;; &#xbc34;&#xb4dc; 2&#xb97c;  ↔
  &#xd30c;&#xb791;&#xc73c;&#xb85c; &#xb9e4;&#xd551;
SELECT ST_AsPNG(rast, ARRAY[3,1,2]) As rastpng
FROM dummy_rast WHERE rid=2;
```

참고

[ST_AsGDALRaster](#), [ST_ColorMap](#), [ST_GDALDrivers](#), Section 11.3

12.11.6 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);
```

설명

래스터에서 선택한 밴드들을 단을 ଴ TIFF(Tagged Image File Format) 이미지로 반환합니다. 밴 따로 설정하지 않을 경우, ન든 밴 쓰려 할 것입니다. 이 함수는 [ST_AsGDALRaster](#) 를 둘러싼 래퍼입니다. 덜 흔한 래유형으로 내보내야 할 경우 [ST_AsGDALRaster](#) 함수를 이용하십시오. 어떤 공간 참조 SRS 텍스트도 없을 경우, 래스터 공간 참조를 사용합니다. 이 함수다음과 같은 많은 옵션을 가진 많은 변종이 있습니다:

- nbands** - 내보낼 밴드들의 배열입니다(P 경우 3이 최대값입니다). 밴드의 순서는 RGB입니다. 예를 들어 ARRAY[3,2,1]은 밴드 3을 빨간색, 밴드 2를 초록색, 밴드 1을 파란색에 매핑한다는 뜻
- compression** - 압축 표현식. JPEG90(또는 다른 퍼은 LZW, JPEG, DEFLATE9
- options** - GTiff에 대해 정의된 GDAL 생성 옵션된 텍스트 배열입니다([ST_GDALDrivers](#)에서 GTiff에 대한 create_options를 살펴보십시오). 자세터 내용은 [GDAL](#) 래스터 형식 옵션 을 참을

- `srid -> spatial_ref_sys, SRID`

GDAL 2.0.0
GDAL 1.6.0

JPEG, 90%

```
SELECT ST_AsTIFF(rast, 'JPEG90') As rasttiff
FROM dummy_rast WHERE rid=2;
```

ST_GDALDrivers, ST_AsGDALRaster, ST_SRID

12.12 ST_Clip

12.12.1 ST_Clip

ST_Clip — Clip a raster to the geometry of another raster or a polygon. The output raster has the same coordinate system and resolution as the input raster. The output raster is cropped to the geometry of the input geometry. The output raster is cropped to the geometry of the input geometry. The output raster is cropped to the geometry of the input geometry.

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);
```

ST_Clip

`geom` — Geometry of the output raster. The output raster is cropped to the geometry of the input geometry. The output raster is cropped to the geometry of the input geometry. The output raster is cropped to the geometry of the input geometry.

ST_Clip — Clip a raster to the geometry of another raster or a polygon. The output raster has the same coordinate system and resolution as the input raster. The output raster is cropped to the geometry of the input geometry. The output raster is cropped to the geometry of the input geometry. The output raster is cropped to the geometry of the input geometry.

```
&#xd568;&#xc218;&#xb294;&#xac01;&#xbc34;&#xb4dc;&#xc5d0;&#xd560;&#xb2f9;&#xb420;&#xb2e8;&#xc77c;&#xac12;&#xc785;&#xb825;&#xbc1b;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
crop &#xc744;&#xc124;&#xc815;&#xd558;&#xc9c0;&#xc54a;&#xc744;&#xacbd;&#xc6b0;,&#xcc38;&#xc73c;&#xb85c;&#xac00;&#xc815;&#xd569;&#xb2c8;&#xb2e4;. geom &#xbc94;&#xc704;&#xc640; rast &#xbc94;&#xc704;&#xac00;&#xad50;&#xcc28;&#xd558;&#xb294;&#xbd80;&#xbd84;&#xc744;&#xc798;&#xb77c;&#xb0b8;&#xb798;&#xc2a4;&#xd130;&#xcd9c;&#xb825;&#xd55c;&#xb2e4;&#xb294;&#xc758;&#xbbf8;&#xc785;&#xb2c8;&#xb2e4;. crop &#xc744;&#xac70;&#xc9c0;&#xc124;&#xc815;&#xd560;&#xacbd;&#xc6b0;,&#xc0c8;&#xb798;&#xc2a4;&#xd130;&#xc758;&#xbc94;&#xc704;&#xb294; rast &#xc758;&#xbc94;&#xc704;&#xc640;&#xb3d9;&#xc77c;&#xd569;&#xb2c8;&#xb2e4;.
```

```
2.0.0 &#xbc84;&#xc804;&#xbd80;&#xd130;&#xc0ac;&#xc6a9;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
&#xac1c;&#xc120;&#xc0ac;&#xd56d;. 2.1.0 &#xbc84;&#xc804;&#xc5d0;&#xc11c; C &#xc5b8;&#xc5b4;&#xb85c;&#xb2e4;&#xc4f0;&#xc600;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
&#xc774;&#xc608;&#xc2dc;&#xb294; MassGIS &#xc0ac;&#xc774;&#xd2b8;&#xc758; MassGIS Aerial Orthos &#xc5d0;&#xc11c;&#xb2e4;&#xc6b4;&#xb85c;&#xb4dc;&#xd560;&#xc218;&#xc788;&#xb294;&#xb9e4;&#xc0ac;&#xcd94;&#xc138;&#xce20;&#xd56d;&#xacf5;&#xc0ac;&#xc9c4;&#xb370;&#xc774;&#xd130;&#xb97c;&#xc774;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.&#xb9e4;&#xc0ac;&#xcd94;&#xc138;&#xce20;&#xc8fc;&#xbbf8;&#xd130;&#xb2e8;&#xc704;&#xd3c9;&#xba74;&#xc758;&#xc88c;&#xd45c;&#xb85c;&#xb3fc;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
&#xc608;&#xc2dc;: &#xbc34;&#xb4dc; 1&#xac1c; &#xc798;&#xb77c;&#xb0b4;&#xae30;
```

```
-- &#xd56d;&#xacf5;&#xc0ac;&#xc9c4;&#xd0c0;&#xc77c;&#xc758;&#xccab;&#xbc88;&#xc9f8; ←
&#xbc34;&#xb4dc;&#xb97c; 20&#bbf8;&#xd130;&#xbc84;&#xd37c;&#xb85c; ←
&#xc798;&#xb77c;&#xb0b4;&#xae30;
SELECT ST_Clip(rast, 1,
               ST_Buffer(ST_Centroid(ST_Envelope(rast)), 20)
               ) from aerials.boston
WHERE rid = 4;
```

```
-- &#xb798;&#xc2a4;&#xd130;&#xc758;&#xb9c8;&#xc9c0;&#xb9c9;&#xcc28;&#xc6d0;&#xc5d0; ←
&#xb300;&#xd55c;&#xc798;&#xb77c;&#xb0b4;&#xae30;&#xc758;&#xc601;&#xd5a5;&#xc744; ←
&#xbcf4;&#xc5ec;&#xc90d;&#xb2c8;&#xb2e4;.
-- &#xb9c8;&#xc9c0;&#xb9c9;&#xbc94;&#xc704;&#xb97c;&#xc5b4;&#xb5bb;&#xac8c; ←
&#xb3c4;&#xd615;&#xc758;&#xbc94;&#xc704;&#xb85c; ←
&#xc798;&#xb77c;&#xb0b4;&#xb294;&#xc9c0;&#xb208;&#xc5ec;&#xaca8; ←
&#xbcf4;&#xc2ed;&#xc2dc;&#xc624;.
-- crop = true &#xc77c;&#xacbd;&#xc6b0;
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



잘라내기
전의 전체
래스터 타일



잘라낸 후

```
&#xc608;&#xc2dc:: crop &#xc5c6;&#xc774; &#xb34; &#xb4dc; 1&#xac1c;&#xb97c; &#xc798;&#xb77c;&#xb0b8; &#xb2e4;&#xc
&#xb2e4;&#xb978; &#xb34;&#xb4dc;&#xb4e4;&#xc744; &#xbcc0;&#xacbd; &#xc5c6;&#xc774; &#xb2e4;&#xc2dc; &#xcd94;&#
```

```
-- &#xc774;&#xc804;&#xacfc; &#xb3d9;&#xc77c;&#xd55c; &#xc608;&#xc2dc;&#xc9c0;&#xb9cc;, ←
ST_AddBand&#xb97c; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc73c;&#xb824;&#xba74; ←
crop&#xc744; &#xac70;&#xc9d3;&#xc73c;&#xb85c; &#xc124;&#xc815;&#xd574;&#xc57c; ←
&#xd569;&#xb2c8;&#xb2e4; .

-- ST_AddBand &#xd568;&#xc218;&#xb97c; &#xc4f0;&#xb824;&#xba74; &#xaa8;&#xb4e0; ←
&#xb34;&#xb4dc;&#xc758; &#xb108;&#xbe44; &#xb0f; &#xb192;&#xc774;&#xac00; ←
&#xb3d9;&#xc77c;&#xd574;&#xc57c; &#xd558;&#xae30; ←
&#xb54c;&#xbb38;&#xc785;&#xb2c8;&#xb2e4; .

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;
```



잘라내기
 전의 전체
 래스터 타일



잘라낸 후 -
 비현실적

예시: 모든 밴드 잘라내기

```
-- &#xd56d;&#xacf5;&#xc0ac;&#xc9c4; &#xd0c0;&#xc77c;&#xc758; &#xbaa8;&#xb4e0;  ↵
  &#xbc34;&#xb4dc;&#xb97c; 20&#xbbf8;&#xd130; &#xbc84;&#xd37c;&#xb85c;  ↵
  &#xc798;&#xb77c;&#xb0b4;&#xae30;
-- &#xcc28;&#xc774;&#xc810;&#xc774;&#xb77c;&#xba74; &#xc798;&#xb77c;&#xb0bc;  ↵
  &#xd2b9;&#xc815; &#xbc34;&#xb4dc;&#xb97c; &#xc9c0;&#xc815;&#xd558;&#xc9c0;  ↵
  &#xc54a;&#xb294;&#xb2e4;&#xb294; &#xc810;&#xbfd0;&#xc785;&#xb2c8;&#xb2e4; .
-- &#xb530;&#xb77c;&#xc11c; &#xbaa8;&#xb4e0; &#xbc34;&#xb4dc;&#xb97c;  ↵
  &#xc798;&#xb77c;&#xb0c5;&#xb2c8;&#xb2e4; .
SELECT ST_Clip(rast,
               ST_Buffer(ST_Centroid(ST_Envelope(rast)), 20),
               false
            ) from aerials.boston
WHERE rid = 4;
```




잘라내기
전의 전체
래스터 타일



잘라낸 후

స고

ST_AddBand, 내장 맵 대수 콜백 함수, ST_Intersection

12.12.2 ST_ColorMap

ST_ColorMap — `소스래스터및설정한밴드&#xb8BUI밴드(grayscale, RGB, RGBA)를4개까지가지는새래스터를생성합니다.밴드를따설정하지않으면밴드1로가정합&`

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);
```

설명

```

rast &#xc758; nband &#xc704;&#xce58;&#xc758; &#xbc34;&#xb4dc;&#xc5d0; colormap &#xc744;&#xc801;&#xc6a9;&#xd5
8BUI &#xbc34;&#xb4dc; 4&#xac1c;&#xae4c;&#xc9c0;&#xb85c; &#xad6c;&#xc131;&#xb41c; &#xc0c8; &#xb798;&#xc2a4;&#xd1
&#xcd9c;&#xb825;&#xd569;&#xb2c8;&#xb2e4; colormap &#xc548;&#xc5d0; &#xc815;&#xc758;&#xb41c; &#xc0c9;&#xc0c1;
&#xad6c;&#xc131; &#xc694;&#xc18c;&#xc758; &#xac1c;&#xc218;&#xac00; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xc758;
8BUI &#xbc34;&#xb4dc; &#xac1c;&#xc218;&#xb97c; &#xacb0;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.

```

nband;#x97c;#xc124;#xc815;#xd558;#xc9c0;#xc54a;#xc744;#xacbd;#xc6b0;, #xbc34;#xb4dc; 1;#xb85c;#xac00;#xc815;#xd569;#xb2c8;#xb2e4;.

```
&#xbbf8;&#xb9ac;&#xc815;&#xc758;&#xb41c;&#xceec;&#xb7ec;&#xb9f5;&#xc758;&#xd0a4;&#xc6cc;&#xb4dc;,&#xb610;&#x  
&#xac12;&#xbc0f;&#xc0c9;&#xc0c1;&#xad6c;&#xc131;&#xc694;&#xc18c;&#xb97c;&#xc815;&#xc758;&#xd558;&#xb294;  
&#xb77c;&#xc778;&#xc9d1;&#xd569;&#xc774; colormap &#xc774;&#xb420;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;  
&#xc720;&#xd6a8;&#xd55c;&#xbbf8;&#xb9ac;&#xc815;&#xc758;&#xb41c; colormap &#xd0a4;&#xc6cc;&#xb4dc::
```

- `grayscale` - 8BUI(`또``는``grayscale` - 8BUI(`଴``드``래``스``터``의``회``에``대``해``파``&#b791;``에``서``&#b179;``색``으``&#b85c;``,``녹``색``에``서``빨``변``하``는``색``상``을``설``정`
- `pseudocolor` - 8BUI(`଴``드``4``개``의``래``스``터``에``대``해``파``&#b791;``에``서``녹``색``으``&#b85c;``,``녹``색``에``서``빨``변``하``는``색``상``을``설``정`
- `fire` - 8BUI(`଴``드``4``개``의``래``스``터``에``대``해``파``&#b791;``에``서``녹``색``으``&#b85c;``,``녹``색``에``서``연``한``하``양``으``&#b85c;``,``연``한``하``양``으``&#b85c;``변``하``는``색``상``을``설``정`
- `bluered` - 8BUI(`଴``드``4``개``의``래``스``터``에``대``해``파``&#b791;``에``서``연``한``하``양``으``&#b85c;``,``연``한``하``양``으``&#b85c;``변``하``는``색``상``을``설``정`

사용자지정󌻬러맵을설정하려사용자쫀 colormap 에 (한 줄에 하나씩 작성한) 항목들의집합을 입력할 수 있습니다 각 항목은 일򼬘적으다음 5개의 값으로 구성됩니다. 픽값과 해당 픽셀의 ୨강 녹색 파랑 알파 구성 요소(RGBA)입니다(색상 구요소는 0부터 255 사이의 값입니다). 픽셀 값 대신 해당 래스터 ଴드의 최소/최대값이 0/100%인 ଱분율 값을 쓸 수도 있습니다. 각 값을 쉼표 탭 쌍점 그리고/또는 공଱으로 구분수 있습니다 픽셀 값이 NODATA 값일 경또는 nodata 로 설정할 수 있습니다. 다음은 그 예시입니다.

```
5 0 0 0 255
4 100:50 55 255
1 150,100 150 255
0% 255 255 255 255
nv 0 0 0 0
```

```
colormap &#xc758;&#xbb38;&#xbc95;&#xc740; GDAL&#xc758;&#xc0c9;&#xc0c1;&#xac15;&#xc870;(color-relief)&bbaa8;&  
gdaldem &#xc758;&#xbb38;&#xbc95;&#xacfc;&#xc720;&#xc0ac;&#xd569;&#xb2c8;&#xb2e4;.
```

&#xc720:&#xd6a8:&#xd55c: method &#xd0a4:&#xc6cc:&#xb4dc::

- INTERPOLATE - ޅ력한픽셀값들사이의색஀드럽게섞여들게하기위해선보간법을이용합니다.
- EXACT - 컬러맵에존재하는픽셀값과엄격히매칭시킵니다.컬러맵항일치하지않는값을가진픽셀은0 0 0 0(RGBA)으로설정할것입니다.
- NEAREST - 픽셀값과가장가까운값을가진컬러맵항목을이용합니다.



Note

```
&#x00000000; &#xcceec;&#xb7ec;&#xb9f5;&#xc5d0; &#xb300;&#xd574;&#xc11c;&#xb294;  
&#xcc38;&#xc870;&#xd558;&#xba74; &#xc88b;&#xc2b5;&#xb2c8;&#xb2e4;
```

ColorBrewer

&#xb97c:



Warning

```
&#xadf8; &#xacb0;&#xacfc; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbc34;&#xb4dc;&#xb4e4;&#xc740;
NODATA &#xac12;&#xc744; &#xac00;&#xc9c0;&#xc9c0; &#xc54a;&#xc744; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
NODATA &#xac12;&#xc774; &#xd544;&#xc694;&#xd558;&#xb2e4;&#xba74; ST_SetBandNoDataValue &#xb97c;
&#xc774;&#xc6a9;&#xd574;&#xc11c; NODATA &#xac12;&#xc744; &#xc124;&#xc815;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

2.1.0 ಄전부터 사용할 수 있습니다.

예시

다음은 예시용 가짜 테이블입니&#x

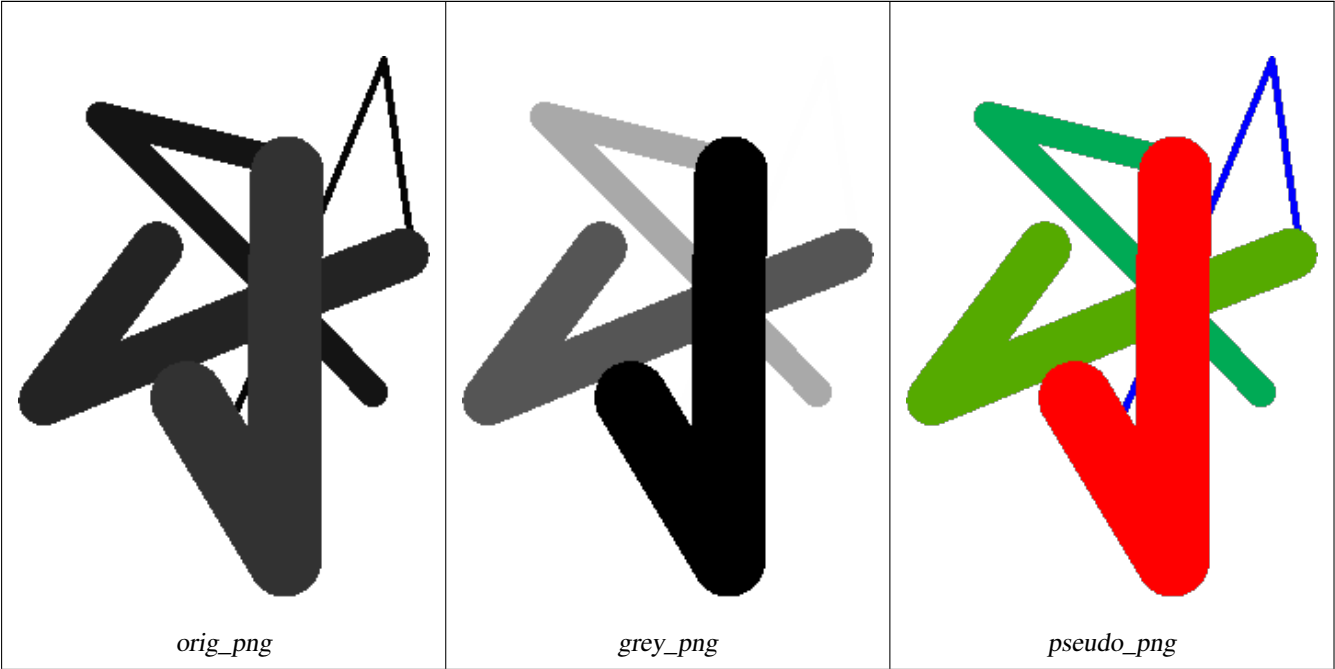
```
-- &#xd14c;&#xc2a4;&#xd2b8; &#xb798;&#xc2a4;&#xd130; &#xd14c;&#xc774;&#xbe14; ↔
&#xc791;&#xc131; --
DROP TABLE IF EXISTS funky_shapes;
CREATE TABLE funky_shapes(rast raster);

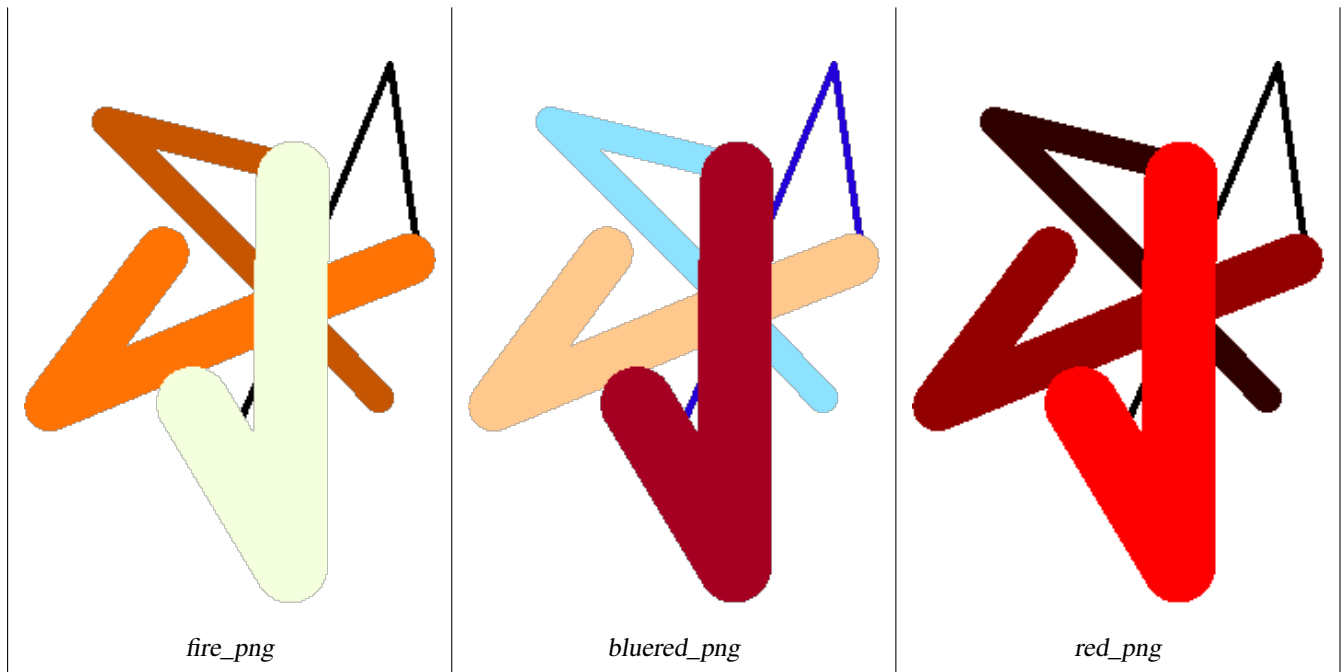
INSERT INTO funky_shapes(rast)
WITH ref AS (
    SELECT ST_MakeEmptyRaster( 200, 200, 0, 200, 1, -1, 0, 0) AS rast
)
SELECT
    ST_Union(rast)
FROM (
    SELECT
        ST_AsRaster(
            ST_Rotate(
                ST_Buffer(
                    ST_GeomFromText('LINESTRING(0 2,50 50,150 150,125 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

ST_AsPNG(
ST_ColorMap(rast,1,'greyscale')) As grey_png,
ST_ColorMap(rast,1,'pseudocolor')) As pseudo_png,
ST_ColorMap(rast,1,'nfire')) As fire_png,
ST_ColorMap(rast,1,'bluered')) As bluered_png,
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;





ST_Grayscale

ST_AsPNG, ST_AsRaster, **ST_Grayscale**, **ST_NumBands**, **ST_Reclass**, **ST_SetBandNoDataValue**, **ST_Union**

12.12.3 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);

ST_Grayscale

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: $0.2989 * \text{RED} + 0.5870 * \text{GREEN} + 0.1140 * \text{BLUE}$

Availability: 2.5.0

Example 1

```

SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
SET postgis.enable_outdb_rasters = True;

WITH apple AS (
  SELECT ST_AddBand(
    ST_MakeEmptyRaster(350, 246, 0, 0, 1, -1, 0, 0, 0),
    '/tmp/apple.png'::text,
    NULL::int[]
  ) AS rast
)
SELECT
  ST_AsPNG(rast) AS original_png,
  ST_AsPNG(ST_Grayscale(rast)) AS grayscale_png
FROM apple;

```

*original_png**grayscale_png***Example 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;

```

󌰸고

ST_AsPNG, ST_Reclass, ST_ColorMap

12.12.4 ST_Intersection

ST_Intersection — 두 래스터의 공유 부분을 표또는 벡터화된 래스터와 도형의 기하학적 교차를 표현하는 래스또는 도형-픽셀값 쌍의 집합을 반&

Synopsis

```
setof geomval ST_Intersection(geometry geom, raster rast, integer band_num=1);
setof geomval ST_Intersection(raster rast, geometry geom);
setof geomval ST_Intersection(raster rast, integer band, geometry geomin);
raster ST_Intersection(raster rast1, raster rast2, double precision[] nodataval);
raster ST_Intersection(raster rast1, raster rast2, text returnband, double precision[] nodataval);
raster ST_Intersection(raster rast1, integer band1, raster rast2, integer band2, double precision[] nodataval);
raster ST_Intersection(raster rast1, integer band1, raster rast2, integer band2, text returnband, double precision[] nodataval);
```

설명

두 래스터의 공유 부분을 표현하또는 벡터화된 래스터와 도형의 기하학적 교차를 표현하는 래스또는 도형-픽셀값 쌍의 집합을 반&

geomval 집합을 반환하는 처음 세 변종 함수는 벡터 스페이스에서 작동먼저 (ST_DumpAsPolygon을 통해) 래스터를 geomval 행집합으로 벡터화한 다음 해당 행음 ST_Intersection(geometry, geometry) PostGIS 함수를 이용해서 도현교차시킵니다 도형이 래스터의 NODATA 값 부분과만 교차할 경우 텅 빈 도형을 반환합니다. WHERE 절에 ST_Intersect 함수를 적절히 사용하면 일반적결과에서 이런 경우들을 제외시수 있습니다.

출력되는 geomval 집합 앞뒤에 괄호를 치고 표현식 끝에 'geom' 또는 '.val' 을 붙이 도형 볰 값 부분에 접근할 수 있습예: (ST_Intersection(rast, geom)).geom

래스터를 반환하는 다른 변종들래스터 스페이스에서 작동합니이 변종들은 교차를 수행하기 위일 ST_MapAlgebraExpr 함수의 래스터 2개를 입력򼼛 보전을 이용합니다.

출력 래스터의 보위는 두 래스터 보위의 기하학적 교차 부분과 일출력 래스터는 returnband 파라미터가 어떻게 설정됐는지에 따라 'BAND1', 'BAND2' 또는 'BOTH' 보드를 포함합니다. 어느 한 보드에서 NODATA 값을 가진 부분은

출력물의 모든 밴드에서 NODATA 값을
가지게 됩니다. 즉. NODATA 값을 가진 픽교차하는 픽셀을 모두 NODATA 값을 가진
픽셀로 출력합니다.

ST_Intersection 함수가 출력하는 래스터는
교차하지 않는 부분에 할당된 NO-
DATA 값을 가지고 있어야 합니다. 사용
'BAND1', 'BAND2' 또는 'BOTH' 밴드 가운데 어떤 것요
청하느냐에 따라 1개 또는 2개의
NODATA 값을 가진 nodataval[] 배열을 입력해서
어떤 출력 밴드에 대해서도 NODATA 값을
정의하౰나 대체할 수 있습니다. ର
열의 첫 번째 값은 첫 번째 밴드에
NODATA 값을 대체하고. 두 번째 값은 두
번째 밴드의 NODATA 값을 대체합니다. 만
약입력 밴드 가운데 하나에 정의 NODATA
값이 없고 입력된 NODATA 값 배열도 없을
경우. ST_MinPossibleValue 함수를 통해 NODATA 값을
선택합니다. NODATA 값의 배열을 입력
ଛ는 모든 변종 함수는 요청사용
각 밴드에 할당될 단일 값도 입력모
든 변종 함수에서. 밴드 번호를 따
로 지정하지 않을 경우 밴드 1로 가
정합니다. 래스터와 도형 사이교차
부분을 래스터로 ଘ환 ଛ아할 경우.
ST_Clip 을 참조하십시오.

Note

출력 범위 또는 NODATA 값을
맞닥뜨렸을 때 ଴엇을
ଘ환할지를 더 세଀히
조정하려면. **ST_MapAlgebraExpr** 함수의 두
래스터를 입력ଛ는 ஄전을
이용하십시오.

Note

래스터 스페이스에서 래스터
밴드와 도형의 교차 부분을
계산하려면. **ST_Clip** 을 이용하십시오
ST_Clip 함수는 복수의 밴드를
가진 래스터를 입력ଛ으며.
래스터화된 도형과 일엨하는
밴드를 ଘ환하지 않습니다.

Note

ST_Intersects 볰 래스터에 대한 인덱스
그리고/또는 도형 열과
결합해서 ST_Intersection 함수를 사용해야
합니다.

개선 사항. 2.0.0부터 래스터 스페이스
교차 부분을 구할 수 있습니다. 2.0.0

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText

```
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(12, 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	

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText

12.12.5

geomval, ST_Intersects, ST_MapAlgebraExpr, ST_Clip, ST_AsText


```

&#xb798;&#xc2a4;&#xd130;&#xc758; &#xd53d;&#xc140; &#xc704;&#xce58;&#xb4e4;&#xc758; &#xc9d1;&#xd569;&#xc7
(&#xc778;&#xb371;&#xc2a4;&#xac00; 0-&#xae30;&#xbc18;&#xc778;) &#xc678;&#xacfd; &#xcc28;&#xc6d0;&#xc740;
&#xb798;&#xc2a4;&#xd130; &#xbc88;&#xd638;&#xc785;&#xb2c8;&#xb2e4;. &#xc678;&#xacfd; &#xcc28;&#xc6d0;
&#xc778;&#xb371;&#xc2a4; 0&#xc5d0; &#xc788;&#xb294; &#xc704;&#xce58;&#xb294; &#xcd9c;&#xb825; &#xb798;&#xc
&#xd53d;&#xc140; &#xc704;&#xce58;&#xc785;&#xb2c8;&#xb2e4;. &#xac01; &#xc678;&#xacfd; &#xcc28;&#xc6d0;&#xc5
&#xb300;&#xd574;. &#xb0b4;&#xacfd; &#xcc28;&#xc6d0;&#xc5d0; X&#xbc0f; Y&#xb97c; &#xc704;&#xd55c; &#xb450;
&#xc694;&#xc18c;&#xac00; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc138; &#xbc88;&#xc9f8; &#xc778;&#xc218;
userargs &#xb294; &#xc5b4;&#xb5a4; &#xc0ac;&#xc6a9;&#xc790; &#xc9c0;&#xc815; &#xc778;&#xc218;&#xb77c;&#xb
&#xb118;&#xaca8;&#xc8fc;&#xae30; &#xc704;&#xd55c; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.

```

```

regprocedure &#xc778;&#xc218;&#xb97c; SQL &#xd568;&#xc218;&#xc5d0; &#xc785;&#xb825;&#xd558;&#xb824;&#xba7
&#xc785;&#xb825;&#xd558;&#xae30; &#xc704;&#xd55c; &#xc804;&#xccb4; &#xd568;&#xc218; &#xc11c;&#xba85;&#xc7
&#xd544;&#xc694;&#xd558;&#xace0;. &#xadf8; &#xb2e4;&#xc74c; regprocedure &#xc720;&#xd615;&#xc73c;&#xb85c;
&#xd615;&#xbcc0;&#xd658;&#xd574;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;. &#xc55e;&#xc758; PL/pgSQL &#xd568;&#xc
&#xc778;&#xc218;&#xb85c;&#xc11c; &#xb118;&#xaca8;&#xc8fc;&#xb824;&#xba74;. &#xd574;&#xb2f9; &#xc778;&#xc21
&#xc704;&#xd55c; SQL&#xc740; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc2b5;&#xb2c8;&#xb2e4;.

```

```
'sample_callbackfunc(double precision[], integer[], text[])':regprocedure
```

```

&#xd574;&#xb2f9; &#xc778;&#xc218;&#xac00; &#xd568;&#xc218;&#xc758; &#xba85;&#xce6d;. &#xd568;&#xc218;
&#xc778;&#xc218;&#xb4e4;&#xc758; &#xc720;&#xd615;. &#xba85;&#xce6d; &#xbc0f; &#xc778;&#xc218; &#xc720;&#xd
&#xac10;&#xc2fc; &#xb530;&#xc634;&#xd45c;. &#xadf8;&#xb9ac;&#xace0; regprocedure &#xb85c;&#xc758; &#xd615;&#xc
&#xd3ec;&#xd568;&#xd558;&#xace0; &#xc788;&#xb2e4;&#xb294; &#xc810;&#xc5d0; &#xc8fc;&#xc758;&#xd558;&#xc2e

```

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 mask mask ఒ에 가중치를 적용해야 할지 값으로 곱해야 할지) 말지(mask를 입에에಄전에만 적용할지) 를 표수붸 값(참/거짓)입니다.

pixeltype pixeltype 을 설정할 경우. 새 래스터의଴드 하나가 해당 픽셀 유형이 될 것입니다. pixeltype 이 NULL이거나 생나 경우. 새 래스터 ଴드가 첫 ଴째 래스터의 지정된 ଴드와(଴위 유형의 경우: INTERSECTION, UNION, FIRST, CUSTOM), 또는 적절한 래스터의 지정된 ଴드와 유형의 경우: SECOND, LAST) 동일한 픽셀 유형이 가지게 될 것입니다. 어떴유형인지 확신하지 못 한다면. 언제나 pixeltype 을 설정하십시오. 출력 래스터의 픽셀 유형은 **ST_BandPixelType** 목록에 존재하는 유형 가운데 하나이거나. 생략되거나. NULL로 설합니다.

extenttype 사용할 수 있는 값은 INTERSECTION(기본본 UNION, FIRST(래스터 1개를 입력଻ 는 변종 기본값), SECOND, LAST, CUSTOM입니다.

customextent extenttype 이 CUSTOM일 경우. 래스터가 customextent 를 입력଻아야만 합니다. 변종 1의 4଻째 예시를 참조하십시오.

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).

```

userargs callbackfunc &#xc5d0; &#xb4e4; &#xc5b4; &#xac00; &#xb294; &#xc138; &#xbc88; &#xc9f8; &#xc778; &#xc218; &#x
variadic text &#xbc30; &#xc5f4; &#xc785; &#xb2c8; &#xb2e4; . &#xae38; &#xace0; &#xae34; &#xd14d; &#xc2a4; &#xd2b8;
&#xc778; &#xc218; &#xb4e4; &#xc774; &#xbaa8; &#xb450; &#xc9c0; &#xc815; &#xb41c; callbackfunc &#xc5d0;
&#xb118; &#xaca8; &#xc9c0; &#xba70; , userargs &#xc778; &#xc218; &#xc5d0; &#xb2f4; &#xaca8; &#xc9d1; &#xb2c8; &#xb

```


Note!

Note



```
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
```

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
```

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
```

WITH foo AS (

```

    SELECT 0 AS rid, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, -1, 0, 0, 0), 1, '16BUI', 1, 0) AS rast UNION ALL
    SELECT 1, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, 0, 1, -1, 0, 0, 0), 1, '16BUI', 2, 0) AS rast UNION ALL
    SELECT 2, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, 0, 1, -1, 0, 0, 0), 1, '16BUI', 3, 0) AS rast UNION ALL

    SELECT 3, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -2, 1, -1, 0, 0, 0), 1, '16BUI', 10, 0) AS rast UNION ALL
    SELECT 4, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -2, 1, -1, 0, 0, 0), 1, '16BUI', 20, 0) AS rast UNION ALL
    SELECT 5, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -2, 1, -1, 0, 0, 0), 1, '16BUI', 30, 0) AS rast UNION ALL

    SELECT 6, ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, -4, 1, -1, 0, 0, 0), 1, '16BUI', 100, 0) AS rast UNION ALL
    SELECT 7, ST_AddBand(ST_MakeEmptyRaster(2, 2, 2, -4, 1, -1, 0, 0, 0), 1, '16BUI', 200, 0) AS rast UNION ALL
    SELECT 8, ST_AddBand(ST_MakeEmptyRaster(2, 2, 4, -4, 1, -1, 0, 0, 0), 1, '16BUI', 300, 0) AS rast
)
SELECT
    t1.rid,
```

```

        ST_MapAlgebra(
            ARRAY[ROW(ST_Union(t2.rast), 1)]::rastbandarg[],
            'sample_callbackfunc(double precision[], int[], text[])'::regprocedure,
            '32BUI',
            'CUSTOM', t1.rast,
            1, 1
        ) AS rast
FROM foo t1
CROSS JOIN foo t2
WHERE t1.rid = 4
      AND t2.rid BETWEEN 0 AND 8
      AND ST_Intersects(t1.rast, t2.rast)
GROUP BY t1.rid, t1.rast

```

이웃을 가진 커버리지의 타일들예시와 유사하지만 PostgreSQL 9.0에서 작돐쿼리입니다.

```

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

```

예시: 변종 2 및 3

```

SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, 1, 1, 1, 0, 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

```

```

SELECT 1 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 1, 1, 1, 1, 0, 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

```

래스터 2개. 밴드 2개

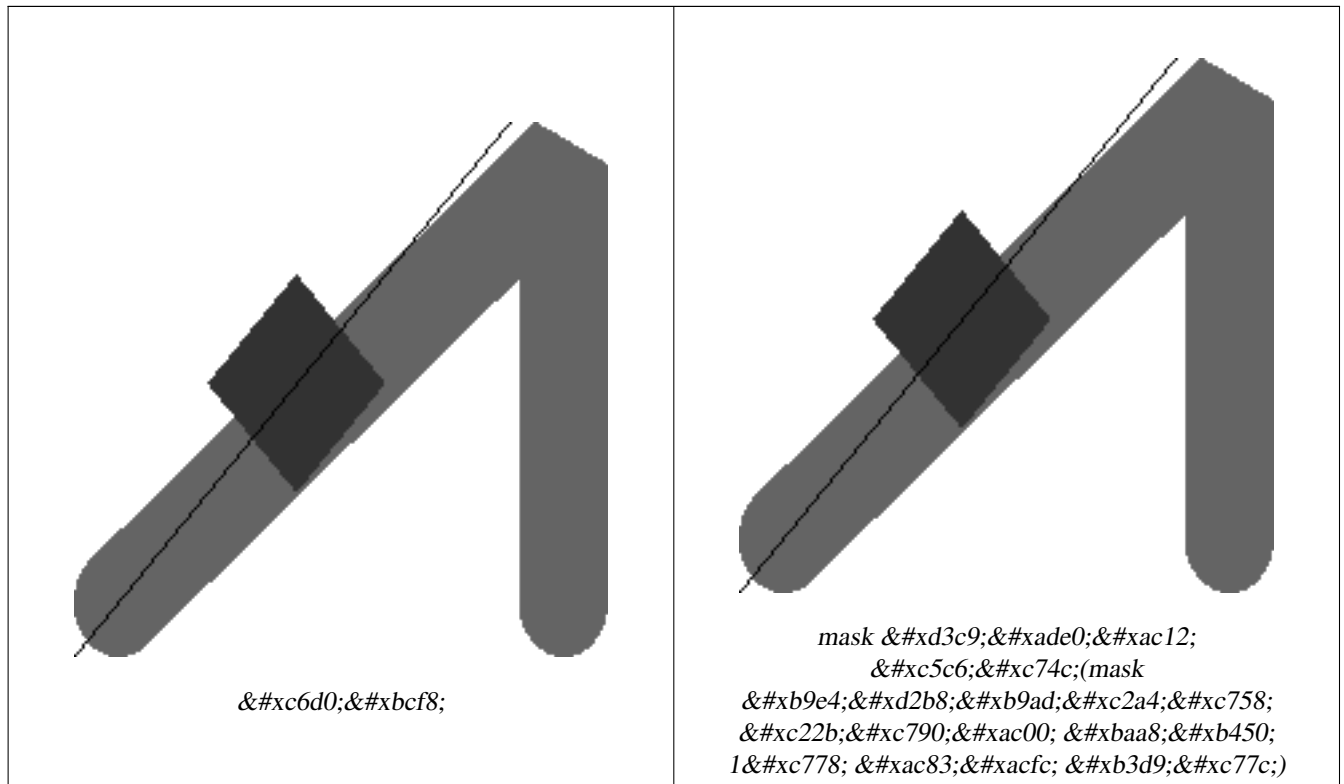
예시; mask 사용하기

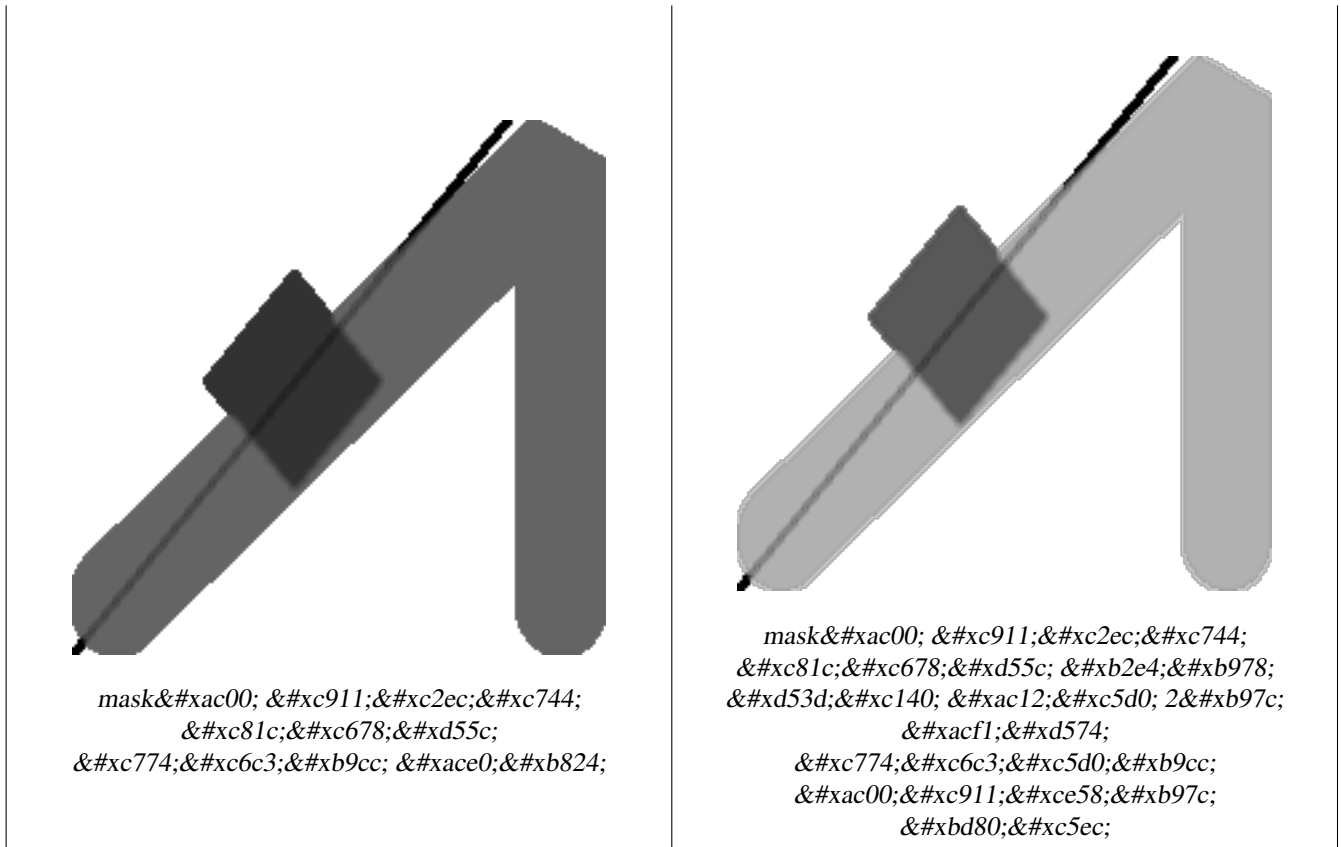
```

WITH foo AS (SELECT
    ST_SetBandNoDataValue(
ST_SetValue(ST_SetValue(ST_AsRaster(
    ST_Buffer(
        ST_GeomFromText('LINESTRING(50 50,100 90,100 50)'), 5,'join=bevel') ←
        ,
        200,200,ARRAY['8BUI'], ARRAY[100], ARRAY[0]), ST_Buffer('POINT(70 ←
        70) '::geometry,10,'quad_segs=1') ,50),
    'LINESTRING(20 20, 100 100, 150 98) '::geometry,1),0) AS rast )
SELECT 'original' AS title, rast
FROM foo
UNION ALL
SELECT 'no mask mean value' AS title, ST_MapAlgebra(rast,1,'ST_mean4ma(double precision[], ←
    int[], text[]) '::regprocedure) AS rast
FROM foo
UNION ALL
SELECT 'mask only consider neighbors, exclude center' AS title, ST_MapAlgebra(rast,1,' ←
    ST_mean4ma(double precision[], int[], text[]) '::regprocedure,
    '{{1,1,1}, {1,0,1}, {1,1,1}}' ::double precision[], false) As rast
FROM foo

UNION ALL
SELECT 'mask weighted only consider neighbors, exclude center multi 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;

```





mask

mask, ST_Union, ST_MapAlgebraExpr

12.12.6 ST_MapAlgebraExpr

ST_MapAlgebraExpr — `ST_MapAlgebraExpr(raster rast, integer nband, text pixeltyp, text expression, double precision nodataval=NULL);`
`ST_MapAlgebraExpr(raster rast, text pixeltyp, text expression, double precision nodataval=NULL);`
`ST_MapAlgebraExpr(raster rast1, integer nband1, raster rast2, integer nband2, text expression, text pixeltyp=NULL, text extnttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);`
`ST_MapAlgebraExpr(raster rast1, raster rast2, text expression, text pixeltyp=NULL, text extnttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);`

Synopsis

raster **ST_MapAlgebra**(raster rast, integer nband, text pixeltyp, text expression, double precision nodataval=NULL);
raster **ST_MapAlgebra**(raster rast, text pixeltyp, text expression, double precision nodataval=NULL);
raster **ST_MapAlgebra**(raster rast1, integer nband1, raster rast2, integer nband2, text expression, text pixeltyp=NULL, text extnttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);
raster **ST_MapAlgebra**(raster rast1, raster rast2, text expression, text pixeltyp=NULL, text extnttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);

mask

mask

2.1.0 `„À඀터 À용할 수 있습니다`

`설명: 변종 1, 2 (래스터 1개)`

`입력 래스터(rast)에 대해 expression 이 정의 유효한 PostgreSQL 대수 연산을 적용해서 형성된 ଴드 1개 를 ఀ진 새 래스터생성합니다. nband 를 설정하지 않을 쪽우 ଴드 1로 ఀ정합니다. 이 새 래스터는 원본 래스터와 동일한 지리స조 너비 볰 높이이지만 ଴&#xb1개만 ఀ질 ಃ입니다.`

`pixeltype 을 설정할 쪽우 새 래스터의 ଴드ఀ 해당 픽셀 유형이 될 ಃ입될 이 NULL일 쪽우 새 래스터의 ଴드입력 rast 의 ଴드와 동일한 픽셀 유픽될 ಃ입니다.`

- `expression 에 키워드를 쓸 수 있습니다`

- `[rast] - 관심 픽셀의 픽셀 값`
- `[rast.val] - 관심 픽셀의 픽셀 값`
- `[rast.x] - 관심 픽셀의 1-기ଘ 픽셀 열`
- `[rast.y] - 관심 픽셀의 1-기ଘ 픽셀 행`

`설명: 변종 3, 4 (래스터 2개)`

`입력 래스터 ଴드 rast1, (rast2)에 대한 expression 이 정의하는 ଴드 2개에 대해 유효 PostgreSQL 대수 연산을 적용해서 형성된 ଴드 1개 를 ఀ진 새 래스터를 생성&#xband1, band2 를 설정하지 않을 쪽우 ଴드 1로 ఀ정합니다. 출력 래스터는 &#xc-cab; ஈ째 래스터ఀ 정의하는 그리드 상에 (축ಙ 기울기 볰 픽셀 모서리 정렬될 ಃ입니다. extenttype 파라미터&#xc0 출력 래스터의 ஔ위를 정의할 ಃ&#xb`

`표현식(expression) 래스터 2개ఀ 관련된 PostgreSQL 대수 표현식 볰 픽셀들이 교쪽우 픽셀 값을 정의할 PostgreSQL 정의 함수/연산자입니다. 예: (([rast1] + [rast2])/2.0)::integer`

`pixeltype 출력 래스터의 픽셀 유형입니이 유형은 ST_BandPixelType 목록에 존재하&#xb2 유형 ఀ운데 하나이౰나 생략&#xb NULL로 설정돼야만 합니다. 따로 설 않౰나 NULL로 설정하지 않으면 &#xc-cab; ஈ째 래스터의 픽셀 유형을 기삼을 ಃ입니다.`

`extenttype 출력 래스터의 ஔ위 제어`

- `INTERSECTION - 새 래스터의 ஔ위는 두 래교న 부분입니다. 기본값입니&`

2. UNION -
3. FIRST -
4. SECOND -

nodata1expr rast1 NODATA rast2 NODATA

nodata2expr rast2 NODATA rast1 NODATA

nodatanodataval rast1 rast2 NODATA

- expression, nodata1expr, nodata2expr

1. [rast1] -
2. [rast1.val] -
3. [rast1.x] -
4. [rast1.y] -
5. [rast2] -
6. [rast2.val] -
7. [rast2.x] -
8. [rast2.y] -

1 2

```
WITH foo AS (
  SELECT ST_SetValue(ST_AddBand(ST_MakeEmptyRaster(3, 3, 0, 0, 1, -1, 0, 0, 0), 1, '8
  BUI', 1, 0), 1, 2, 5) AS rast
)
SELECT
  (ST_DumpValues(rast, 1))[2][1]
FROM foo;

st_dumpvalues
-----
5
(1 row)
```

```
&#xc608;&#xc2dc;: &#xbcc0;&#xc885; 3 &#xbc0f; 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', 100, 0) AS rast ←
    UNION ALL
    SELECT 2 AS rid, ST_AddBand(ST_AddBand(ST_AddBand(ST_MakeEmptyRaster(2, 2, 0, 0, 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
```

```
&#xcc38;&#xace0;
```

```
rastbandarg, ST_Union, &#xb0b4;&#xc7a5; &#xb9f5; &#xb300;&#xc218; &#xcf5c;&#xbc31; &#xd568;&#xc218;
```

12.12.7 ST_MapAlgebraExpr

ST_MapAlgebraExpr — 래스터 밴드 1개 버전: 입력 래스터에 대해 유효한 PostgreSQL 대수 연 적용해서 형성되고, 설정한 픽셀 유형을 가진, 밴드 1개 를 가진 새 래 생성합니다. 따로 밴드를 설정하 않을 경우, 밴드 1로 가정합니다.

Synopsis

raster **ST_MapAlgebraExpr**(raster rast, integer band, text pixeltype, text expression, double precision nodataval=NULL);
 raster **ST_MapAlgebraExpr**(raster rast, text pixeltype, text expression, double precision nodataval=NULL);

```
&#xc124;&#xba85;
```



Warning

ST_MapAlgebraExpr 2.1.0 버전부터 더 이상 이 함수를 지원하지 않습니다. 대신 **ST_MapAlgebraExpr** 함수를 이용하십시&#xc62

입력 래스터(rast)에 대해 expression 이 정&#xc758 유효한 PostgreSQL 대수 연산을 적용해서 형성된, 밴드 1개 를 가진 새 래스터 생성합니다. nband 를 설정하지 않을 경우, 밴드 1로 가정합니다. 이 새 래스터는 원본 래스터와 동일한

```
&#xc9c0;&#xb9ac;&#xcc38;&#xc870;,&#xb108;&#xbe44;&#xbc0f;&#xb192;&#xc774;&#xc774;&#xc9c0;&#xb9cc;,&#xbc34;&#xb1&#xac1c;&#xb9cc;&#xac00;&#xc9c8;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
pixeltype &#xc744;&#xc124;&#xc815;&#xd560;&#xacbd;&#xc6b0;,&#xc0c8;&#xb798;&#xc2a4;&#xd130;&#xc758;&#xbc34;&#xb4dc;&#xac00;&#xd574;&#xb2f9;&#xd53d;&#xc140;&#xc720;&#xd615;&#xc774;&#xb420;&#xac83;&#xc785;&#xb2c8;&#xb2e4;
pixeltype &#xc774; NULL&#xc77c;&#xacbd;&#xc6b0;,&#xc0c8;&#xb798;&#xc2a4;&#xd130;&#xc758;&#xbc34;&#xb4dc;&#xc785;&#xb825; rast &#xc758;&#xbc34;&#xb4dc;&#xc640;&#xb3d9;&#xc77c;&#xd55c;&#xd53d;&#xc140;&#xc720;&#xd615;&#xb420;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
&#xd45c;&#xd604;&#xc2dd;&#xc5d0;&#xc11c;&#xc6d0;&#xbcf8;&#xbc34;&#xb4dc;&#xc758;&#xd53d;&#xc140;&#xac12;&#xb825;&#xcc38;&#xc870;&#xd558;&#xb294;&#xb370; [rast], 1-&#xae30;&#xbc18;&#xd53d;&#xc140;&#xc5f4;&#xc778;&#xb371;&#xcc38;&#xc870;&#xd558;&#xb294;&#xb370; [rast.x], 1-&#xae30;&#xbc18;&#xd53d;&#xc140;&#xd589;&#xc778;&#xb371;&#xcc38;&#xc870;&#xd558;&#xb294;&#xb370; [rast.y] &#xc6a9;&#xc5b4;&#xb97c;&#xc0ac;&#xc6a9;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
2.0.0 &#xbc84;&#xc804;&#xbd80;&#xd130;&#xc0ac;&#xc6a9;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시

```
&#xc6d0;&#xbcf8;&#xb798;&#xc2a4;&#xd130; 2&#xac1c;&#xb97c;&#xc785;&#xb825;&#xbc1b;&#xb294;&#xbaa8;&#xb4c8;&#xd568;&#xc218;&#xc778;&#xc6d0;&#xbcf8;&#xc73c;&#xb85c;&#xbd80;&#xd130;&#xbc34;&#xb4dc; 1&#xac1c;&#xb97c;&#xac00;&#xc9c4;&#xc0c8;&#xb798;&#xc2a4;&#xd130;&#xb97c;&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
```

```
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

```
&#xc7ac;&#xbd84;&#xb958;&#xb97c;&#xc70;&#xce58;&#xace0; NODATA &#xac12;&#xc744; 0&#xc73c;&#xb85c;&#xc124;&#xc6d0;&#xbcf8;&#xc73c;&#xb85c;&#xbd80;&#xd130;&#xd53d;&#xc140;&#xc720;&#xd615;&#xc774; 2BUI&#xc778;,&#xbc34;&#xb4dc; 1&#xac1c;&#xb97c;&#xac00;&#xc9c4;&#xc0c8;&#xb798;&#xc2a4;&#xd130;&#xb97c;&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
```

```
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;

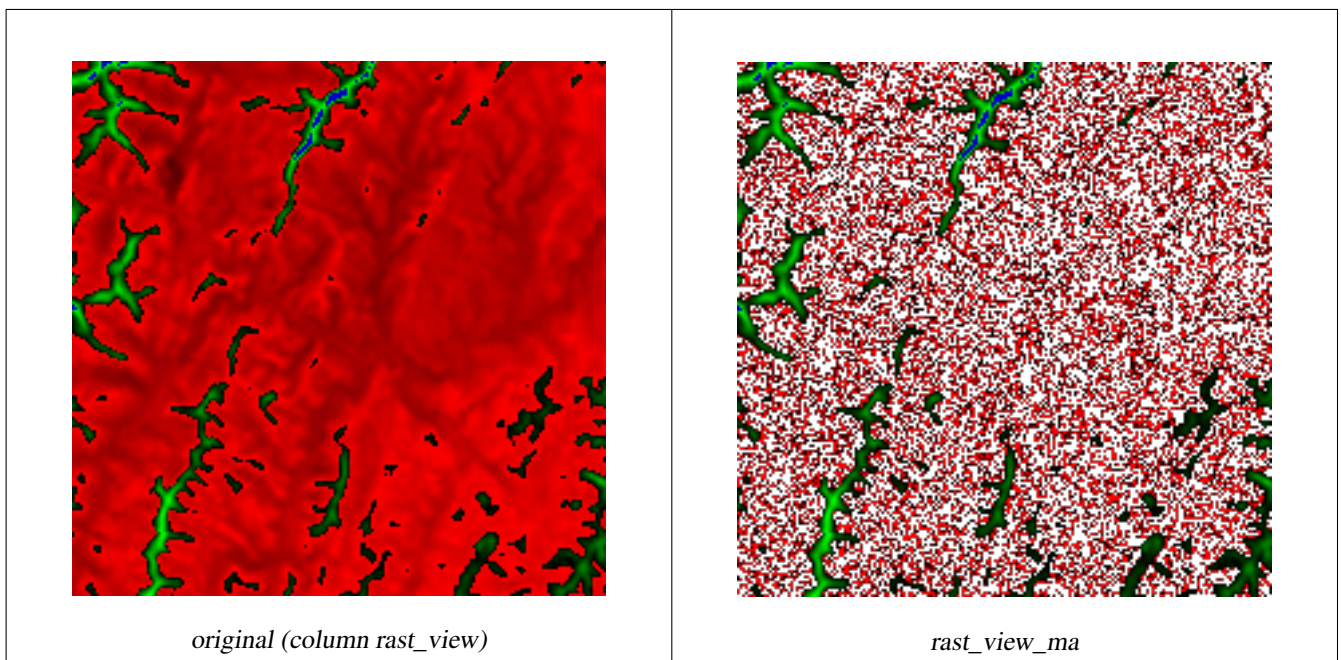
SELECT DISTINCT ST_Value(rast,1,i,j) As origval, ST_Value(map_rast2, 1, i, j) As mapval
FROM dummy_rast CROSS JOIN generate_series(1, 5) AS i CROSS JOIN generate_series(1,5) AS j
WHERE rid = 2;

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

```



㰴드 3㰜를 㰀ৄ 원본 래스터에서
맵 대수로 첫 번째 밴드를 조정하&#
나머지 두 밴드는 그대로 둔 상태&#
동일한 픽셀 유형인 밴드 3㰜를 㰀&
새 래스터를 생성합니다.

```
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;
```

స고

ST_MapAlgebraExpr, ST_MapAlgebraFct, ST_BandPixelType, ST_GeoReference, ST_Value

12.12.8 ST_MapAlgebraExpr

ST_MapAlgebraExpr — 래스터 밴드 2개 버전; 입력
 래스터 2개에 대해 유효한 PostgreSQL 대&#xc218
 연산을 적용해서 형성되고, 설정&#
 픽셀 유형을 가진, 밴드 1개를 가진
 새 래스터를 생성합니다. 따로 밴&
 설정하지 않을 경우, 각 래스터의
 밴드 1로 가정합니다. 출력 래스터&
 첫 번째 래스터가 정의하는 그리&#
 상에 (축척, 기울기 및 픽셀 모서리&
 정렬될 것입니다. extenttype 파라򻯸터&#xac0
 출력 래스터의 범위를 정의할 것&#
 extenttype 의 값은 INTERSECTION, UNION, FIRST, SECOND가 될 수
 있습니다.

Synopsis

```
raster ST_MapAlgebraExpr(raster rast1, raster rast2, text expression, text pixeltype=same_as_rast1_band, text extnttype=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 extnttype=INTERSECTION, text nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);
```

설명



Warning

```
ST_MapAlgebraExpr 2.1.0 &#xc84;&#xc804;&#xbd80;&#xd130; &#xb354; &#xc774;&#xc0c1; &#xc774;
&#xd568;&#xc218;&#xb97c; &#xc9c0;&#xc6d0;&#xd558;&#xc9c0; &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;
&#xb300;&#xc2e0;ST_MapAlgebraExpr&#xd568;&#xc218;&#xb97c; &#xc774;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc62
```

```
&#xc785;&#xb825; &#xb798;&#xc2a4;&#xd130; &#xbc34;&#xb4dc; rast1, (rast2)&#xc5d0; &#xb300;&#xd55c; expression
&#xc774; &#xc815;&#xc758;&#xd558;&#xb294; &#xbc34;&#xb4dc; 2&#xac1c;&#xc5d0; &#xb300;&#xd574;., &#xc720;&#xd6a8;
PostgreSQL &#xb300;&#xc218; &#xc5f0;&#xc0b0;&#xc744; &#xc801;&#xc6a9;&#xd574;&#xc11c; &#xd615;&#xc131;&#xb41c;.,
&#xbc34;&#xb4dc; 1&#xac1c;&#xb97c; &#xac00;&#xc9c4; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#
band1, band2 &#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0;., &#bc34;&#xb4dc;
1&#xb85c; &#xac00;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.. &#xcd9c;&#xb825; &#xb798;&#xc2a4;&#xd130;&#xb294; &#xc-
cab; &#bc88;&#xc9f8; &#xb798;&#xc2a4;&#xd130;&#xac00; &#xc815;&#xc758;&#xd558;&#xb294; &#xadf8;&#xb9ac;&#xb4dc;
&#xc0c1;&#xc5d0; (&#xcd95;&#xcc99;., &#xae30;&#xc6b8;&#xae30; &#xbc0f; &#xd53d;&#xc140; &#xbaa8;&#xc11c;&#xb9ac;&#
&#xc815;&#xb82c;&#xb420; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.. extenttype &#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xac0
&#xcd9c;&#xb825; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbc94;&#xc704;&#xb97c; &#xc815;&#xc758;&#xd560; &#xac83;&#
```

```
&#xd45c;&#xd604;&#xc2dd;(expression) &#xb798;&#xc2a4;&#xd130; 2&#xac1c;&#xac00; &#xad00;&#xb828;&#xb41c;
PostgreSQL &#xb300;&#xc218; &#xd45c;&#xd604;&#xc2dd; &#xbc0f; &#xd53d;&#xc140;&#xb4e4;&#xc774; &#xad50;&#xb
&#xacbd;&#xc6b0; &#xd53d;&#xc140; &#xac12;&#xc744; &#xc815;&#xc758;&#xd560; PostgreSQL &#xc815;&#xc758;
&#xd568;&#xc218;/&#xc5f0;&#xc0b0;&#xc790;&#xc785;&#xb2c8;&#xb2e4;. &#xc608;:: (([rast1] + [rast2])/2.0)::integer
```

```
pixeltype &#xcd9c;&#xb825;&#xb798;&#xc2a4;&#xd130;&#xc758;&#xd53d;&#xc140;&#xc720;&#xd615;&#xc785;&#xb2c8;&  
    &#xc774;&#xc720;&#xd615;&#xc740; ST_BandPixelType &#xaa9;&#xb85d;&#xc5d0;&#xc874;&#xc7ac;&#xd558;&#xb2  
    &#xc720;&#xd615;&#xac00;&#xc6b4;&#xb370;&#xd558;&#xb098;&#xc774;&#xac70;&#xb098;&#xc0dd;&#xb7b5;&#xb  
    NULL&#xb85c;&#xc124;&#xc815;&#xb3fc;&#xc57c;&#xb9cc;&#xd569;&#xb2c8;&#xb2e4;&#xb530;&#xb85c;&#xc124;  
    &#xc54a;&#xac70;&#xb098; NULL&#xb85c;&#xc124;&#xc815;&#xd558;&#xc9c0;&#xc54a;&#xc73c;&#xba74;&#xc-  
    cab;&#xbc88;&#xc9f8;&#xb798;&#xc2a4;&#xd130;&#xc758;&#xd53d;&#xc140;&#xc720;&#xd615;&#xc744;&#xae30;&#xc  
    &#xc0bc;&#xc744;&#xac83;&#xc785;&#xb2c8;&#xb2e4;
```

extenttype 출력 래스터의 범위 제어

1. INTERSECTION - 새 래스터의 범위는 두 래스교차 부분입니다. 기본값입니입
2. UNION - 새 래스터의 범위는 두 래스򭆵합한 범위입니다.
3. FIRST - 새 래스터의 범위는 첫 번째 래스터의 범위와 동일합니다.
4. SECOND - 새 래스터의 범위는 두 번째 래스터의 범위와 동일합니다.

```

nodata1expr rast1 &#xc758;&#xd53d;&#xc140;&#xb4e4;&#xc774; NODATA &#xac12;&#xc774;&#xace0;&#xacf5;&#xac04;&
&#xc0c1;&#xc751;&#xd558;&#xb294; rast2 &#xc758;&#xd53d;&#xc140;&#xb4e4;&#xc774; &#xac12;&#xc744;
&#xac00;&#xc9c0;&#xace0; &#xc788;&#xc744; &#xb54c; &#xc5b4;&#xb5a4; &#xac83;&#xc744; &#xbc18;&#xd658;&#xd5
&#xc815;&#xc758;&#xd558;&#xb294; &#xc0c1;&#xc218;&#xb9cc; &#xb610;&#xb294; rast2 &#xc640;&#xb9cc;
&#xad00;&#xb828;&#xb41c; &#xb300;&#xc218; &#xd45c;&#xd604;&#xc2dd;&#xc785;&#xb2c8;&#xb2e4;.

```

```
nodata2expr rast2 &#xc758; &#xd53d; &#xc140; &#xb4e4; &#xc774; NODATA &#xac12; &#xc774; &#xace0; &#xacf5; &#xac04; &#xc0c1; &#xc751; &#xd558; &#xb294; rast1 &#xc758; &#xd53d; &#xc140; &#xb4e4; &#xc774; &#xac12; &#xc744; &#xac00; &#xc9c0; &#xace0; &#xc788; &#xc744; &#xb54c; &#xc5b4; &#xb5a4; &#xac83; &#xc744; &#xbc18; &#xd658; &#xd5815; &#xc758; &#xd558; &#xb294; &#xc0c1; &#xc218; &#xb9cc; &#xb610; &#xb294; rast1 &#xacf; &#xb9cc; &#xad00; &#xb828; &#xb41c; &#xb300; &#xc218; &#xd45c; &#xd604; &#xc2dd; &#xc785; &#xb2c8; &#xb2e4;.
```

```
nodatanodataval &#xacf5;&#xac04;&#xc801;&#xc73c;&#xb85c; &#xc0c1;&#xc751;&#xd558;&#xb294; rast1 &#xbc0f;
rast2 &#xc758;&#xd53d;&#xc140;&#xb4e4;&#xc774; &#xbaa8;&#xb450; NODATA &#xac12;&#xc77c; &#xacbd;&#xc6b0
&#xbc18;&#xd658;&#xd558;&#xb294; &#xc22b;&#xc790; &#xc0c1;&#xc218;&#xc785;&#xb2c8;&#xb2e4;.
```

```
pixeltype &#xc744; &#xc124;&#xc815;&#xd560; &#xacbd;&#xc6b0;, &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xc758;
&#xbc34;&#xb4dc;&#xac00; &#xd574;&#xb2f9; &#xd53d;&#xc140; &#xc720;&#xd615;&#xc774; &#xb420; &#xac83;&#xc785;&#xb2c8;
pixeltype &#xc774; NULL&#xc774;&#xac70;&#xb098; &#xb530;&#xb85c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xb2e4;
&#xacbd;&#xc6b0;, &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbc34;&#xb4dc;&#xb294; &#xc785;&#xb825; rastl
&#xc758; &#xbc34;&#xb4dc;&#xc640; &#xb3d9;&#xc77c;&#xd55c; &#xd53d;&#xc140; &#xc720;&#xd615;&#xc774; &#xb420;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
&#xc6d0;&#xbcf8;&#xbc34;&#xb4dc;&#xc758;&#xd53d;&#xc140;&#xac12;&#xc744;&#xcc38;&#xc870;&#xd558;&#xb294;
&#xb370;[rast1.val],[rast2.val],&#xd53d;&#xc140;&#xc758;&#xc5f4;/&#xd589;&#xc704;&#xce58;&#xb97c;
&#xcc38;&#xc870;&#xd558;&#xb294;&#xb370;[rast1.x],[rast1.y]&#xb4f1;&#xc758;&#xc6a9;&#xc5b4;&#xb97c;
&#xc0ac;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

2.0.0 버전부터 사용할 수 있습니다.

예&#xc2dc:: 밴드 2개의 교차 &#bc0f; &#d1b5;합

원본래스터 2개를입력받는모듈함수인원본으로부터밴드 1개를가진새래스터를생성합니다.

```
-- &#xb798;&#xc2a4;&#xd130; &#xc9d1;&#xd569; &#xc0dd;&#xc131; --
DROP TABLE IF EXISTS fun_shapes;
CREATE TABLE fun_shapes(rid serial PRIMARY KEY, fun_name text, rast raster);

-- &#xb9e4;&#xc0ac;&#xcd94;&#xc138;&#xce20; &#xc8fc; &#xbbf8;&#xd130; &#xb2e8;&#xc704;  ←
    &#xd3c9;&#xba74;&#xc758; &#xbcf4;&#xc2a4;&#xd134; &#xc8fc;&#xc704;&#xc5d0; &#xba87;  ←
    &#xac1c;&#xc758; &#xd615;&#xc0c1;&#xc744; &#xc0bd;&#xc785; --
INSERT INTO fun_shapes(fun_name, rast)
VALUES ('ref', ST_AsRaster(ST_MakeEnvelope(235229, 899970, 237229, 901930,26986),200,200,'8 ←
    BUI',0,0));

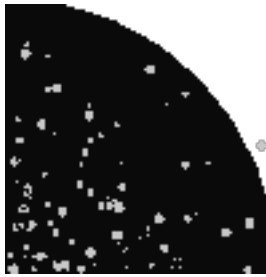
INSERT INTO fun_shapes(fun_name,rast)
```

```

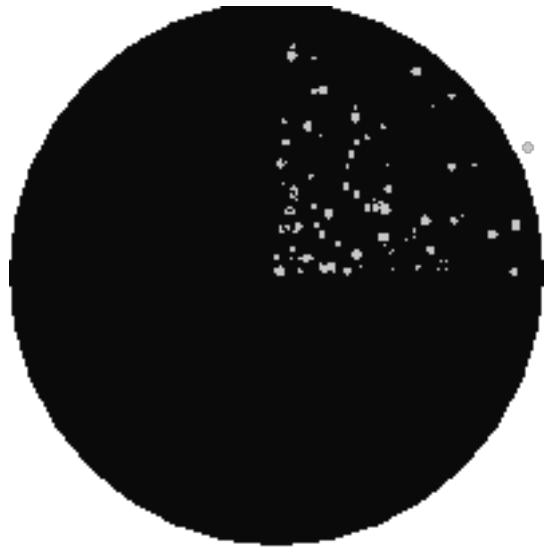
WITH ref(rast) AS (SELECT rast FROM fun_shapes WHERE fun_name = 'ref' )
SELECT 'area' AS fun_name, ST_AsRaster(ST_Buffer(ST_SetSRID(ST_Point(236229, 900930),26986) ←
, 1000),
                                ref.rast,'8BUI', 10, 0) As rast
FROM ref
UNION ALL
SELECT 'rand bubbles',
      ST_AsRaster(
        (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;

-- &#xb9e4;&#xd551; --
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

```



맵 대수
교차(intersection)



맵 대수 통합(union)

예시: 캔버스 상에 래스터들을 개଴드로서 오버레이

```

-- &#xbaa8;&#xb4e0; &#xb2e8;&#xc77c; &#xbc34;&#xb4dc; &#xb798;&#xc2a4;&#xd130;&#xb97c; ←
&#xd68c;&#xc0c9;&#xc870;&#xb85c; &#xb9cc;&#xb4e4;&#xae30; &#xc704;&#xd574; ST_AsPNG ←
&#xb97c; &#xd1b5;&#xd574; &#xc774;&#xbbf8;&#xc9c0;&#xb97c; ←
&#xb80c;&#xb354;&#xb9c1;&#xd569;&#xb2c8;&#xb2e4; . --

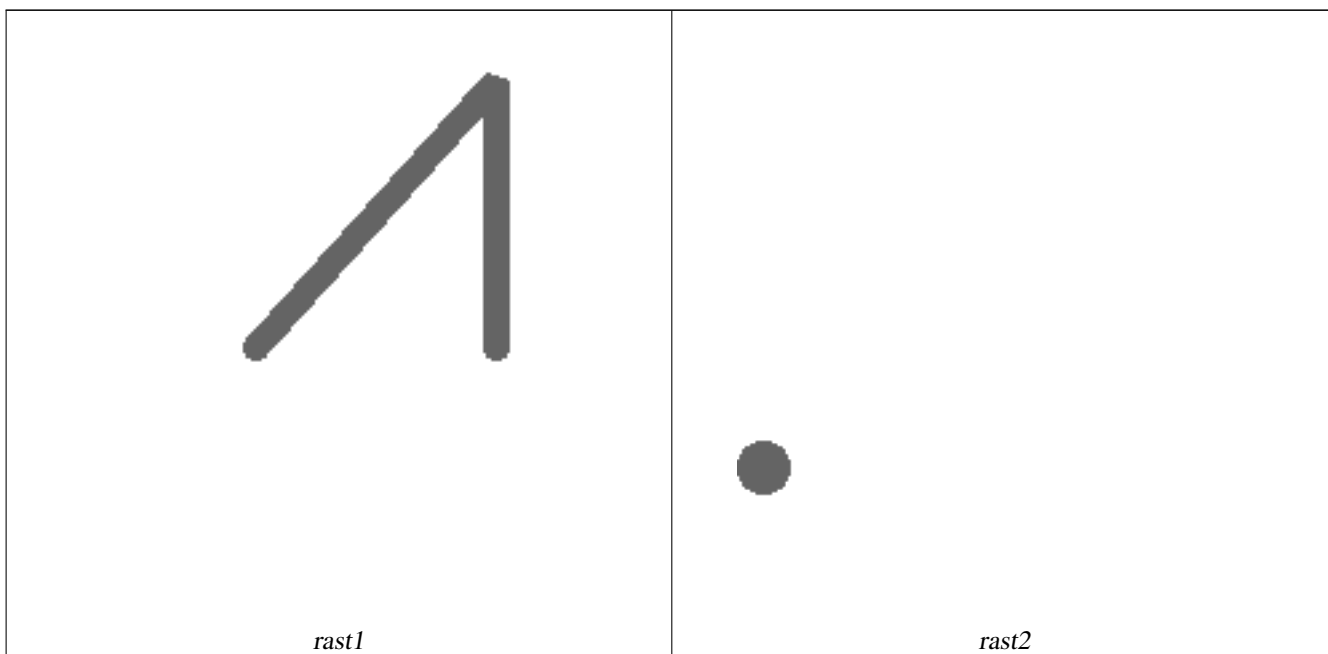
```

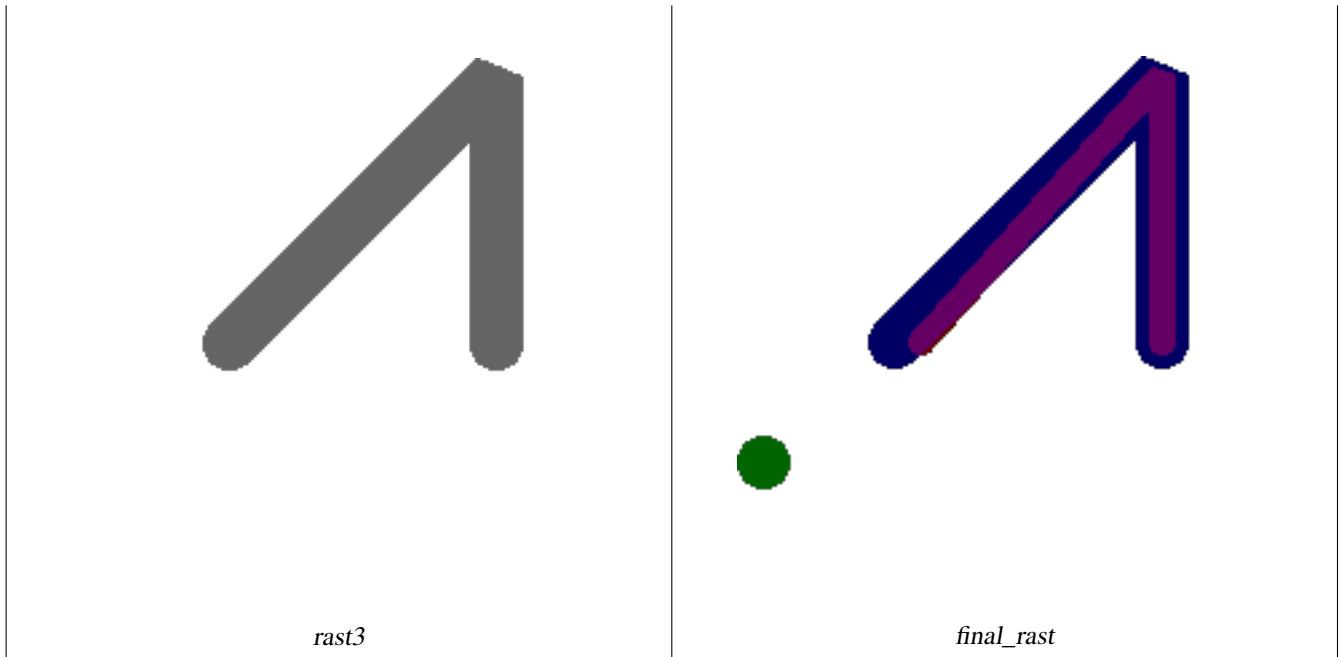


```

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
    ),
  -- 1:1 ↵
  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;

```





PostGIS 3.3.9dev

```
-- #xc98;#xc74c; 2#xac1c;#xc758; #xbc34;#xb4dc;#xb97c;  ←
#xc798;#xb77c;#xb0b4;#xace0; #xc138; #xbc88;#xc9f8; #xbc34;#xb4dc;#xb97c;  ←
#xb3c4;#xd615;#xacfc; #xacb9;#xce5c; , #xbc34;#xb4dc; 3#xac1c;#xb97c;  ←
#xac00;#xc9c4; #xc0c8; #xb798;#xc2a4;#xd130;#xb97c;  ←
#xc0dd;#xc131;#xd569;#xb2c8;#xb2e4; .
-- PostGIS #xc708;#xb3c4;#xc6b0; 64#xbe44;#xd2b8;  ←
#xc124;#xce58;#xbcf8;#xc5d0;#xc11c; #xc774; #xcffc;#xb9ac;#xac00; 3.6#xcd08;  ←
#xac78;#xb838;#xc2b5;#xb2c8;#xb2e4; .
WITH pr AS
-- #xc5f0;#xc0b0; #xc21c;#xc11c;#xc5d0;  ←
#xc8fc;#xc758;#xd558;#xc2ed;#xc2dc;#xc624; : #xbaa8;#xb4e0;  ←
#xb798;#xc2a4;#xd130;#xb97c; #xd574;#xb2f9; #xc9c0;#xc5ed;#xc758;  ←
#xcc28;#xc6d0;#xc5d0; #xb9de;#xac8c; #xc798;#xb77c;#xb0c5;#xb2c8;#xb2e4; .
(SELECT ST_Clip(rast,ST_Expand(geom,50) ) As rast, g.geom
FROM aerials.o_2_boston AS r INNER JOIN
-- #xad00;#xc2ec; #xd68d;#xc9c0;#xb97c; #xd1b5;#xd569;#xd574;#xc11c;  ←
#xb098;#xc911;#xc5d0; #xad50;#xcc28;#xc2dc;#xd0ac; #xc218; #xc788;#xb294;  ←
#xb2e8;#xc77c; #xb3c4;#xd615;#xc73c;#xb85c; #xb9cc;#xb4ed;#xb2c8;#xb2e4; .
(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)
),
-- #xadf8;#xb9ac;#xace0; #xb798;#xc2a4;#xd130; #xc870;#xac01;#xb4e4;#xc744;  ←
#xd1b5;#xd569;#xd569;#xb2c8;#xb2e4; .
-- #xb798;#xc2a4;#xd130;#xb97c; ST_Union#xc73c;#xb85c;  ←
#xd1b5;#xd569;#xd558;#xb294; #xc791;#xc5c5;#xc740; #xc870;#xae08;  ←
#xb290;#xb9b0; #xd3b8;#xc774;#xc9c0;#xb9cc; , #xb798;#xc2a4;#xd130;#xb97c;  ←
#xc798;#xac8c; #xcabc;#xc24;#xc218;#xb85d; #xc18d;#xb3c4;#xb294;  ←
#xbe68;#xb77c;#xc9d1;#xb2c8;#xb2e4; .
-- #xb530;#xb77c;#xc11c; #xba3c;#xc800; #xc798;#xb77c;#xb0b8;  ←
#xb2e4;#xc74c;#xc5d0; #xd1b5;#xd569;#xd558;#xb294; #xd3b8;#xc774;  ←
#xc88b;#xc2b5;#xb2c8;#xb2e4; .
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)
-- &#xb798;&#xc2a4;&#xd130; &#xc870;&#xac01;&#xb4e4;&#xc744;  ←
  &#xd1b5;&#xd569;&#xd558;&#xace0; &#xd68d;&#xc9c0; &#xacbd;&#xacc4;&#xc120;&#xacfc;  ←
  &#xacb9;&#xce5c;
-- &#xacb0;&#xacfc; &#xb798;&#xc2a4;&#xd130;&#xb97c;  ←
  &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
-- &#xcc98;&#xc74c; 2&#xac1c;&#xc758; &#xbc34;&#xb4dc;&#xb97c; &#xcd94;&#xac00;&#xd55c;  ←
  &#xb2e4;&#xc74c;, &#xc138; &#xbc88;&#xc9f8; &#xbc34;&#xb4dc;&#xc640;  ←
  &#xb3c4;&#xd615;&#xc5d0; &#xb9f5; &#xb300;&#xc218;&#xb97c;  ←
  &#xc801;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
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;

```



파란색 라인이 선택된 획지의
 경계선입니다.

참고

[ST_MapAlgebraExpr](#), [ST_AddBand](#), [ST_AsPNG](#), [ST_AsRaster](#), [ST_MapAlgebraFct](#), [ST_BandPixelType](#), [ST_GeoReference](#),
[ST_Value](#), [ST_Union](#), [ST_Union](#)

12.12.9 ST_MapAlgebraFct

[ST_MapAlgebraFct](#) — 래스터 밴드 1개 버전: 입력
 래스터에 대해 유효한 PostgreSQL 대수 연
 적용해서 형성되고, 설정한 픽셀
 유형을 가진, 밴드 1개를 가진 새 래을

band, 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);

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);

Warning



Warning

ST_MapAlgebraFct 2.1.0
 raster rast, regprocedure onerasteruserfunc, text[] VARIADIC args;
 raster rast, text pixeltype, regprocedure onerasteruserfunc, text[] VARIADIC args;
 raster rast, integer band, regprocedure onerasteruserfunc;
 raster rast, integer band, regprocedure onerasteruserfunc, text[] VARIADIC args;
 raster rast, integer band, text pixeltype, regprocedure onerasteruserfunc;
 raster rast, integer band, text pixeltype, regprocedure onerasteruserfunc, text[] VARIADIC args;

band, 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);

pixeltype
 NULL
 raster
 regprocedure
 text[]

onerasteruserfunc
 regprocedure
 text[]

```
CREATE OR REPLACE FUNCTION simple_function(pixel FLOAT, pos INTEGER[], VARIADIC args TEXT [])
RETURNS FLOAT
AS $$ BEGIN
RETURN 0.0;
END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

userfunction
 raster
 regprocedure
 text[]

```

&#xc0c1;&#xad00;&#xc5c6;&#xc774;) &#xac1c;&#xbcc4;&#xc801;&#xc778; &#xb798;&#xc2a4;&#xd130; &#xc140;&#xc758;
&#xac12;&#xc785;&#xb2c8;&#xb2e4;. &#xb450; &#xbc88;&#xc9f8; &#xc778;&#xc218;&#xb294; &#xd604;&#xc7ac; &#xcce98;&#xc758;
&#xc140;&#xc758; &#xc704;&#xc5e8;&#xb97c; '{x,y}' &#xc11c;&#xc2dd;&#xc73c;&#xb85c; &#xd45c;&#xd604;&#xd55c;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xc138; &#xbc88;&#xc9f8; &#xc778;&#xc218;&#xb294; ST_MapAlgebraFct &#xc758;
&#xbaa8;&#xb4e0; &#xb098;&#xba38;&#xc9c0; &#xd30c;&#xb77c;&#xbbbf8;&#xd130;&#xb4e4;&#xc744; userfunction
&#xc744; &#xd1b5;&#xd574; &#xc124;&#xc815;&#xd574;&#xc57c; &#xd55c;&#xb2e4;&#xb294; &#xac83;&#xc744; &#xc758;&#xc744;
regprocedure &#xc778;&#xc218;&#xb97c; SQL &#xd568;&#xc218;&#xc5d0; &#xc785;&#xb825;&#xd558;&#xb824;&#xba74;
&#xc785;&#xb825;&#xd558;&#xae30; &#xc704;&#xd55c; &#xc804;&#xccb4; &#xd568;&#xc218; &#xc11c;&#xba85;&#xc774;
&#xd544;&#xc694;&#xd558;&#xace0;, &#xadf8; &#xb2e4;&#xc74c; regprocedure &#xc720;&#xd615;&#xc73c;&#xb85c;
&#xd615;&#xbcc0;&#xd658;&#xd574;&#xc57c; &#xd569;&#xb2c8;&#xb2e4;. &#xc55e;&#xc758; PL/pgSQL &#xd568;&#xc218;&#xc778;
&#xc218;&#xb85c;&#xc11c; &#xb118;&#xaca8;&#xc8fc;&#xb824;&#xba74;, &#xd574;&#xb2f9; &#xc778;&#xc218;&#xc704;&#xd55c;
SQL&#xc740; &#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc2b5;&#xb2c8;&#xb2e4;:

```

```
'simple_function(float,integer[],text[])':regprocedure
```

```

&#xd574;&#xb2f9; &#xc778;&#xc218;&#xac00; &#xd568;&#xc218;&#xc758; &#xba85;&#xce6d;, &#xd568;&#xc218; &#xc778;&#xc720;
&#xd615;, &#xba85;&#xce6d; &#xbc0f; &#xc778;&#xc218; &#xc720;&#xd615;&#xc744; &#xac10;&#xc2fc; &#xb530;&#xadf8;
&#xb9ac;&#xace0; regprocedure &#xb85c;&#xc758; &#xd615;&#xbcc0;&#xd658;&#xc744; &#xd3ec;&#xd568;&#xd558;
&#xc788;&#xb2e4;&#xb294; &#xc810;&#xc5d0; &#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.

```

```

userfunction &#xc5d0; &#xb4e4;&#xc5b4;&#xac00;&#xb294; &#xc138; &#xbc88;&#xc9f8; &#xc778;&#xc218;&#xb294;
variadic text &#xb30;&#xc5f4;&#xc785;&#xb2c8;&#xb2e4;. &#xc5b4;&#xb5a4; ST_MapAlgebraFct &#xd568;&#xc218;
&#xd638;&#xcd9c;&#xc5d0;&#xb3c4; &#xc785;&#xb825;&#xb418;&#xb294; &#xae38;&#xace0; &#xae34; &#xd14d;&#xc2a4;&#xc778;
&#xc218;&#xb4e4;&#xc774; &#xbaa8;&#xb450; &#xc9c0;&#xc815;&#xb41c; userfunction &#xc5d0; &#xb118;&#xc778;
args &#xc778;&#xc218;&#xc5d0; &#xb2f4;&#xaca8;&#xc9d1;&#xb2c8;&#xb2e4;.

```

Note



(다양한 개수의 인수를 입력벱는) VARIADIC 키워드에 대한 더 자세한 정보를 알고 싶다면, PostgreSQL 문서 가운데 **Query Language (SQL) Functions** 의 "SQL Functions with Variable Numbers of Arguments" 단원을 참조하십시오.

Note



공간 처리를 위해 사용자 함수에 어떤 인수를 넘겨주기로 하고 말고에 상관없이, userfunction 에 들어가는 text[] 인수는 필요합니다.

2.0.0 ஄전부터 사용할 수 있습니다.

예시

```

&#xc6d0;&#xbcf8; &#xb798;&#xc2a4;&#xd130; 2&#xac1c;&#xb97c; &#xc785;&#xb825;&#xbcb1;&#xb294; &#xbaa8;&#xb4c8;&#xd568;
&#xc218;&#xc778; &#xc6d0;&#xbcf8;&#xc73c;&#xb85c;&#xbd80;&#xd130; &#xb34;&#xb4dc; 1&#xac1c;&#xb97c; &#xac00;&#xc9c4;
&#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.

```

```

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

```

&#xc7ac;&#xbd84;&#xb958;&#xb97c; &#xac70;&#xce58;&#xace0; NODATA &#xac12;&#xc744; &#xc0ac;&#xc6a9;&#xc790;
&#xd568;&#xc218;(0)&#xc73c;&#xb85c; &#xb118;&#xaca8;&#xc9c4; &#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb85c; &#xc124;&#
&#xc6d0;&#xbcf8;&#xc73c;&#xb85c;&#xbd80;&#xd130; &#xd53d;&#xc140; &#xc720;&#xd615;&#xc774; 2BUI&#xc778;,
&#xbc34;&#xb4dc; 1&#xac1c;&#xb97c; &#xac00;&#xc9c4; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#

```

```

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


```

        ST_Band(rast_view, 3) As rast_view_ma
    )
FROM wind
WHERE rid=167;

```

ST_MapAlgebraFct

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

12.12.10 ST_MapAlgebraFct

ST_MapAlgebraFct — raster ST_MapAlgebraFct(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);
 raster ST_MapAlgebraFct(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

Synopsis

raster **ST_MapAlgebraFct**(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

raster **ST_MapAlgebraFct**(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

Warning



Warning

ST_MapAlgebraFct 2.1.0 raster ST_MapAlgebraFct(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);
 raster ST_MapAlgebraFct(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

ST_MapAlgebraFct — raster ST_MapAlgebraFct(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);
 raster ST_MapAlgebraFct(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

ST_MapAlgebraFct — raster ST_MapAlgebraFct(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);
 raster ST_MapAlgebraFct(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

ST_MapAlgebraFct — raster ST_MapAlgebraFct(raster rast1, raster rast2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);
 raster ST_MapAlgebraFct(raster rast1, integer band1, raster rast2, integer band2, regprocedure tworastuserfunc, text pixeltype=same_as_rast1, text extenttype=INTERSECTION, text[] VARIADIC userargs);

(signature)이어야합니다. 다음은 PL/pgSQL 함수의예입니다:

```
CREATE OR REPLACE FUNCTION simple_function_for_two_rasters(pixel1 FLOAT, pixel2 FLOAT, pos ←
    INTEGER[], VARIADIC args TEXT[])
RETURNS FLOAT
AS $$ BEGIN
RETURN 0.0;
END; $$
LANGUAGE 'plpgsql' IMMUTABLE;
```

```
tworastuserfunc &#xc740;&#xc778;&#xc218;&#xb97c; 3&#xac1c;&#xb610;&#xb294; 4&#xac1c;&#xbcb1b;&#xc744;
&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc774;&#xc911;&#xc815;&#xbcb00;&#xb3c4;&#xac12;. &#xc774;&#xc911;
&#xc815;&#xbcb00;&#xb3c4;&#xac12;. &#xc120;&#xd0dd;&#xc801;&#xc778;&#xc815;&#xc218;&#xd615;&#xbcb30;&#xc5f4;.
&#xadf8;&#xb9ac;&#xace0;&#xc784;&#xc758;&#xb85c;&#xc5ec;&#xb7ec;&#xc778;&#xc218;&#xb97c;&#xc785;&#xb825;&#xc
&#xbb38;&#xc790;&#xc5f4;&#xbcb30;&#xc5f4;&#xc785;&#xb2c8;&#xb2e4;. &#xccab;&#xbcb88;&#xc9f8;&#xc778;&#xc218;&#xc
(&#xb798;&#xc2a4;&#xd130;&#xc758;&#xb370;&#xc774;&#xd130;&#xd615;&#xacfc;&#xc0c1;&#xad00;&#xc5c6;&#xc774;)
rast1&#xc5d0;&#xc788;&#xb294;&#xac1c;&#xbcc4;&#xc801;&#xc778;&#xb798;&#xc2a4;&#xd130;&#xc140;&#xc758;
&#xac12;&#xc785;&#xb2c8;&#xb2e4;. &#xb450;&#xbcb88;&#xc9f8;&#xc778;&#xc218;&#xb294;&#xc788;&#xc218;&#xc758;
&#xac1c;&#xbcc4;&#xc801;&#xc778;&#xb798;&#xc2a4;&#xd130;&#xc140;&#xc758;&#xac12;&#xc785;&#xb2c8;&#xb2e4;.
&#xc138;&#xbcb88;&#xc9f8;&#xc778;&#xc218;&#xb294;&#xd604;&#xc7ac;&#xc98;&#xb9ac;&#xc911;&#xc778;&#xc140;&#xc
&#xc704;&#xc5e8;&#xb97c; '{x,y}'&#xc11c;&#xc2dd;&#xc73c;&#xb85c;&#xd45c;&#xd604;&#xd55c;&#xc83;&#xc785;&#xb2
&#xb124;&#xbcb88;&#xc9f8;&#xc778;&#xc218;&#xb294; ST_MapAlgebraFct &#xc758;&#xbaa8;&#xb4e0;&#xb098;&#xba38;&#xc
&#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb4e4;&#xc744;tworastuserfunc &#xc744;&#xd1b5;&#xd574;&#xc124;&#xc815;
&#xd55c;&#xb2e4;&#xb294;&#xc83;&#xc744;&#xc758;&#xbbf8;&#xd569;&#xb2c8;&#xb2e4;.
```

```
regprocedure &#xc778;&#xc218;&#xb97c; SQL &#xd568;&#xc218;&#xc5d0;&#xc785;&#xb825;&#xd558;&#xb824;&#xba74;
&#xc785;&#xb825;&#xd558;&#xae30;&#xc704;&#xd55c;&#xc804;&#xccb4;&#xd568;&#xc218;&#xc11c;&#xba85;&#xc774;
&#xd544;&#xc694;&#xd558;&#xace0;. &#xadf8;&#xb2e4;&#xc74c; regprocedure &#xc720;&#xd615;&#xc73c;&#xb85c;
&#xd615;&#xbcc0;&#xd658;&#xd574;&#xc57c;&#xd569;&#xb2c8;&#xb2e4;. &#xc55e;&#xc758; PL/pgSQL &#xd568;&#xc218;&#xc
&#xc778;&#xc218;&#xb85c;&#xc11c;&#xb118;&#xaca8;&#xc8fc;&#xb824;&#xba74;. &#xd574;&#xb2f9;&#xc778;&#xc218;&#xc
&#xc704;&#xd55c; SQL&#xc740;&#xb2e4;&#xc74c;&#xacfc;&#xc19;&#xc2b5;&#xb2c8;&#xb2e4;:
```

```
'simple_function(double precision, double precision, integer[], text[])':regprocedure
```

```
&#xd574;&#xb2f9;&#xc778;&#xc218;&#xcac00;&#xd568;&#xc218;&#xc758;&#xba85;&#xce6d;. &#xd568;&#xc218;&#xc778;&#xc
&#xc720;&#xd615;. &#xba85;&#xce6d;&#xbcb0f;&#xc778;&#xc218;&#xc720;&#xd615;&#xc744;&#xcac10;&#xc2fc;&#xb530;&#xc
&#xadf8;&#xb9ac;&#xace0; regprocedure &#xb85c;&#xc758;&#xd615;&#xbcc0;&#xd658;&#xc744;&#xd3ec;&#xd568;&#xd558;
&#xc788;&#xb2e4;&#xb294;&#xc810;&#xc5d0;&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

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



(다양한개수의인수를입력򼬛는) VARIADIC 키워드에대한더자세한정보를알고싶다면. PostgreSQL 문서󊰀운데 [Query Language \(SQL\) Functions](#) 의 "SQL Functions with Variable Numbers of Arguments" 단원을참조하십시오.

Note



󊳵󊰄ಘ리를위해사용자함수에어떤인수를넘겨주기로하고말고에상관없이., `tworastuserfunc` 에들어󊰀는인수는필요합니다.

2.0.0 򼮄전부터사용할수있습니다.

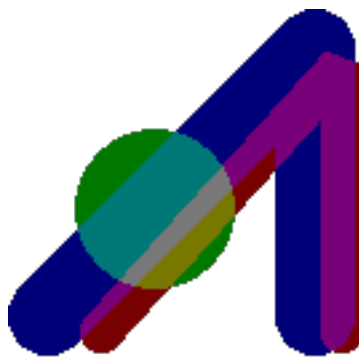
예시: 캔버스 상에 래스터들을 개&#
밴드로서 오버레이

```
-- &#xc0ac;&#xc6a9;&#xc790; &#xc9c0;&#xc815; &#xd568;&#xc218;&#xc97c; &#xc815;&#xc758; --
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;

-- &#xb798;&#xc2a4;&#xd130; &#xd14c;&#xc2a4;&#xd2b8; &#xd14c;&#xc774;&#xbe14; &#xc900;&#xbe44;
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
    ),
-- &#xce94;&#xbc84;&#xc2a4;&#xc00; &#xb3c4;&#xd615;&#xc5d0; 1:1 ←
    &#xd53d;&#xc140;&#xc774; &#xb418;&#xb3c4;&#xb85d; &#xc815;&#xc758;
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
    )
-- &#xce94;&#xbc84;&#xc2a4;&#xc5d0; &#xc815;&#xb82c;&#xb41c; ←
    &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xbc18;&#xd658;
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;
```

```
-- ST_MapAlgebraFct: ST_MapAlgebraFct(rast1, rast2, fct,
    rast1 double precision, rast2 double precision, fct text[],
    pos integer[], variadic userargs text[])
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;
```



ST_MapAlgebraFct: ST_MapAlgebraFct(rast1, rast2, fct, rast1 double precision, rast2 double precision, fct text[], pos integer[], variadic userargs text[]) (R: small road, G: circle, B: big road)

ST_MapAlgebraFct: ST_MapAlgebraFct(rast1, rast2, fct, rast1 double precision, rast2 double precision, fct text[], pos integer[], variadic userargs text[]) (R: small road, G: circle, B: big road)

```
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
```

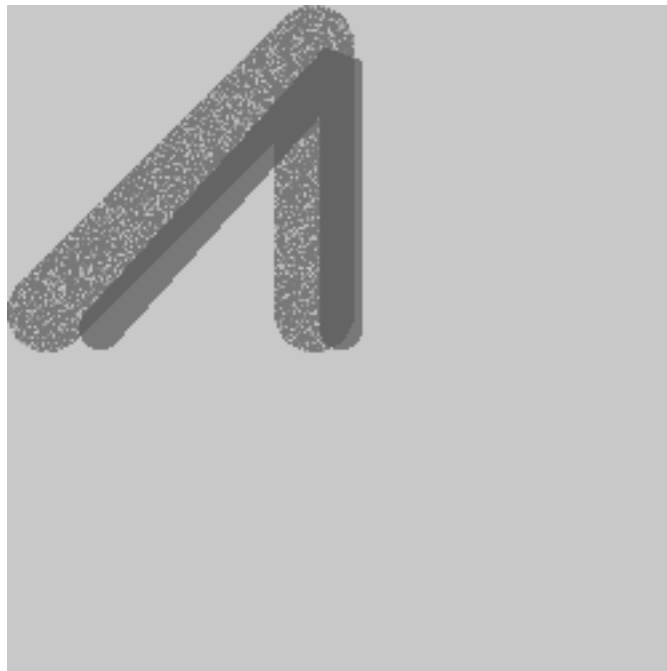
```

        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)';

```



추가 인수 및 동일한
 래스터에서 가져온 다른
 밴드를 받는 사용자 지정 함수

참고

ST_MapAlgebraExpr, ST_BandPixelType, ST_GeoReference, ST_SetValue

12.12.11 ST_MapAlgebraFctNgb

ST_MapAlgebraFctNgb — 래스터 밴드 1개 버전: 사용지정 PostgreSQL 함수를 이용하는 맵 대수 최근접 이웃(Map Algebra Nearest Neighbor)입니다. 입력 래스터 밴드의 값의 이웃(neighborhood)이

```
&#xad00;&#xb828;&#xb41c; PostgreSQL &#xc0ac;&#xc6a9;&#xc790; &#xd568;&#xc218;&#xac00; &#xcd9c;&#xb825;&#xd558;&#xac12;&#xc744; &#xac00;&#xc9c4; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;
```

Synopsis

```
raster ST_MapAlgebraFctNgb(raster rast, integer band, text pixeltype, integer ngbwidth, integer ngbheight, regprocedure on-  
erastngbuserfunc, text nodatamode, text[] VARIADIC args);
```

설명



Warning

ST_MapAlgebraFctNgb 2.1.0 ಄전부터 더 이상 이함수를 지원하지 않습니다.대신 내장 맵 대수 콜돁 함수함수를 이용하십시오.

래스터 1개버전: 입력 래스터 밴드
값의 이웃(neighborhood)이 관련된 PostgreSQL 사용&#xc
함수가 출력하는 값을 가진 래스&#
반함합니다: 사용자 함수가 픽셀&#
이웃 값을 숫자의 배열로 입력받&#
각 픽셀에 대해 사용자 함수의 결&#
반함해서 현재 처리중인 픽셀의
픽셀 값을 함수 결과값으로 대체&#x

rast 사용자 함수를 평가하는 데 쓰&#xc77
래스터

band 1)

```
pixeltype &#xcd9c;&#xb825;&#xb798;&#xc2a4;&#xd130;&#xc758;&#xd53d;&#xc140;&#xc720;&#xd615;&#xc785;&#xb2c8;&  
    &#xc774;&#xc720;&#xd615;&#xc740; ST_BandPixelType &#xaa9;&#xb85d;&#xc5d0;&#xc874;&#xc7ac;&#xd558;&#xb2  
    &#xc720;&#xd615;&#xac00;&#xc6b4;&#xb370;&#xd558;&#xb098;&#xc774;&#xac70;&#xb098;&#xc0dd;&#xb7b5;&#xb  
    NULL&#xb85c;&#xc124;&#xc815;&#xb3fc;&#xc57c;&#xb9cc;&#xd569;&#xb2c8;&#xb2e4;&#xb530;&#xb85c;&#xc124;  
    &#xc54a;&#xac70;&#xb098; NULL&#xb85c;&#xc124;&#xc815;&#xd558;&#xc9c0;&#xc54a;&#xc73c;&#xba74;&#xast&#x  
    &#xd53d;&#xc140;&#xc720;&#xd615;&#xc744;&#xae30;&#xbcf8;&#xac12;&#xc73c;&#xb85c;&#xc0bc;&#xc744;  
    &#xac83;&#xc785;&#xb2c8;&#xb2e4;&#xacb0;&#xacfc;&#xac12;&#xc774;&#xd53d;&#xc140;&#xc720;&#xd615;&#xc75  
    &#xd5c8;&#xc6a9;&#xce58;&#xbcf4;&#xb2e4;&#xd074;&#xacbd;&#xc6b0;&#xacb0;&#xacfc;&#xac12;&#xc758;  
    &#xae38;&#xc774;&#xb97c;&#xd5c8;&#xc6a9;&#xce58;&#xc5d0;&#xb9de;&#xcdb0;&#xc904;&#xc785;&#xb2c8;&#xb2e4;
```

ngbwidth 이웃(neighborhood)의 셀 단위 너비

ngbheight 이웃(neighborhood)의 셀 단위 높이

```
onerastngbuserfunc &#xb798;&#xc2a4;&#xd130;&#xc758; &#xb2e8;&#xc77c; &#xbc34;&#xb4dc;&#xc758; &#xadfc;&#xc811;
&#xd53d;&#xc140;&#xc5d0; &#xc801;&#xc6a9;&#xd560; PL/pgSQL &#xb610;&#xb294; psql &#xc0ac;&#xc6a9;&#xc790;
&#xd568;&#xc218;&#xc785;&#xb2c8;&#xb2e4;. &#xccab; &#xbc88;&#xc9f8; &#xc694;&#xc18c;&#xac00; &#xc0ac;&#xac
&#xd53d;&#xc140; &#xc774;&#xc6c3;&#xc744; &#xd45c;&#xd604;&#xd558;&#xb294; 2&#xc28;&#xc6d0; &#xc22b;&#xc
&#xbc30;&#xc5f4;&#xc785;&#xb2c8;&#xb2e4;.
```

```

nodatamode NODATA &#xb610;&#xb294; NULL&#xc778; &#xc774;&#xc6c3; &#xd53d;&#xc140;&#xc758; &#xacbd;&#xc6b0;
&#xd568;&#xc218;&#xc5d0; &#xc5b4;&#xb5a4; &#xac12;&#xc744; &#xb118;&#xaca8;&#xc904;&#xc9c0; &#xc815;&#xc7

```

```
'ignore': &#xc774;&#xc6c3;&#xc5d0;&#xc11c; &#xb9de;&#xb2e5;&#xb728;&#xb9b0; &#xc5b4;&#xb5a4; NODATA
&#xac12;&#xb3c4; &#xacc4;&#xc0b0;&#xc744; &#xd1b5;&#xd574; &#xbb34;&#xc2dc;&#xd569;&#xb2c8;&#xb2e4;.
```

```
&#xc0ac;&#xc6a9;&#xc790; &#xcf5c;&#xbc31; &#xd568;&#xc218;&#xc5d0; &#xc774; &#xd50c;&#xb798;&#xadf8;&#xb97
&#xc804;&#xc1a1;&#xd574;&#xc57c;&#xb9cc; &#xd558;&#xba70; &#xc0ac;&#xc6a9;&#xc790; &#xd568;&#xc218;&#xb2
NODATA &#xac12;&#xc744; &#xc5b4;&#xb5bb;&#xac8c; &#xbb34;&#xc2dc;&#xd560; &#xac83;&#xc778;&#xc9c0;
&#xacb0;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.
```

```
'NULL': &#xc774;&#xc6c3;&#xc5d0;&#xc11c; &#xb9de;&#xb2e5;&#xb728;&#xb9b0; &#xbaa8;&#xb4e0; NODATA
&#xac12;&#xc774; &#xcd9c;&#xb825; &#xd53d;&#xc140;&#xc744; NULL&#xb85c; &#xb9cc;&#xb4e4; &#xac83;&#xc785;
&#xc774; &#xacbd;&#xc6b0; &#xc0ac;&#xc6a9;&#xc790; &#xcf5c;&#xbc31; &#xd568;&#xc218;&#xb97c; &#xac74;&#xb10
'value': &#xc774;&#xc6c3;&#xc5d0;&#xc11c; &#xb9de;&#xb2e5;&#xb728;&#xb9b0; &#xc5b4;&#xb5a4; NODATA
&#xac12;&#xb3c4; &#xcc38;&#xc870; &#xd53d;&#xc140;(&#xc774;&#xc6c3;&#xc758; &#xd55c;&#xac00;&#xc6b4;&#xb3
&#xc788;&#xb294; &#xd53d;&#xc140;)&#xb85c; &#xb300;&#xc6b4;&#xd569;&#xb2c8;&#xb2e4;. &#xc774; &#xac12;&#xc
NODATA&#xc77c; &#xacbd;&#xc6b0; (&#xc601;&#xd5a5;&#xc744; &#xbcb1b;&#xb294; &#xc774;&#xc6c3;&#xc5d0;
&#xb300;&#xd574;) 'NULL'&#xacfc; &#xb3d9;&#xc77c;&#xd55c; &#xc2b5;&#xc131;&#xc744; &#xbcf4;&#xc778;&#xb2e4
&#xc810;&#xc5d0; &#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

```
args &#xc0ac;&#xc6a9;&#xc790; &#xd568;&#xc218;&#xb85c; &#xb118;&#xaca8;&#xc904; &#xc778;&#xc218;&#xb4e4;
```

```
2.0.0 &#xb84;&#xc804;&#xbd80;&#xd130; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시

```
&#xb2e4;&#xc74c; &#xc608;&#xc2dc;&#xb294; http://trac.osgeo.org/gdal/wiki/frmts\_wtkraster.html &#xc5d0;&#xc11c; &#xc124;&
&#xb300;&#xb85c; &#xb2e8;&#xc77c; &#xd0c0;&#xc77c;&#xb85c; &#xbd88;&#xb7ec;&#xc628; &#xb2e4;&#xc74c; ST_Rescale
&#xc608;&#xc2dc; &#xb300;&#xb85c; &#xc900;&#xbe44;&#xd55c; &#xc74;&#xd2b8;&#xb9ac;&#xb098; &#xb798;&#xc2a4;&#
&#xd65c;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
```

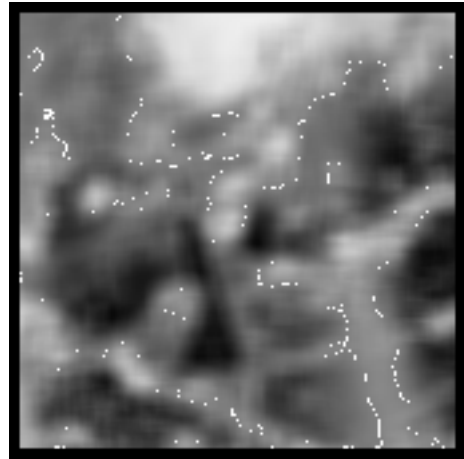
```
-- &#xc774;&#xc6c3;&#xc5d0; &#xc788;&#xb294; &#xbaa8;&#xb4e0; &#xac12;&#xc744;  ←
&#xd3c9;&#xade0;&#xd558;&#xb294; &#xb2e8;&#xc21c;&#xd55c; '&#xcf5c;&#xbc31;'  ←
&#xc0ac;&#xc6a9;&#xc790; &#xd568;&#xc218;&#xc785;&#xb2c8;&#xb2e4;. --
CREATE OR REPLACE FUNCTION rast_avg(matrix float[][], nodatamode 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;

-- &#xc774;&#xc81c; &#xac01;&#xac01; &#xc11c;&#xb85c;&#xc758; X &#xbcf0; Y  ←
&#xb29;&#xd5a5;&#xc73c;&#xb85c; &#xd53d;&#xc140; 2&#xac1c; &#xb9cc;&#xd07c;&#xc758;  ←
&#xac70;&#xb9ac; &#xc548;&#xc5d0; &#xb798;&#xc2a4;&#xd130;&#xc758;  ←
&#xd3c9;&#xade0;&#xb0b8; &#xd53d;&#xc140;&#xb4e4;&#xc744;  ←
&#xc801;&#xc6a9;&#xd574;&#xbd05;&#xc2dc;&#xb2e4;. --
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;
```



래스터의 첫
번째 밴드



각각 서로의 4x4
픽셀 거리
안에 있는
픽셀들을
평균낸 후의
새 래스터

참고

[ST_MapAlgebraFct](#), [ST_MapAlgebraExpr](#), [ST_Rescale](#)

12.12.12 ST_Reclass

ST_Reclass — 원본으로부터 재분류된 밴드
유형으로 이루어진 새 래스터를
생성합니다. nband 는 변경할 밴드를
가리킵니다. nband 를 따로 설정하지
않을 경우 밴드 1로 가정합니다. 다모든 밴드들은 변경 없이 반환됩&#
실제 사례. 보기 좋은 형식으로 더
간단하게 렌더링하기 위해 16BUI 밴드
8BUI 등등으로 변환하십시오.

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);

설명

입력 래스터(rast)에 대해 reclassexpr 이 정의
유효한 PostgreSQL 대수 연산을 적용해서


```

&#xd615;&#xc131;&#xb41c; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;
band &#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0;,, &#xbc34;&#xb4dc; 1&#xb85c;
&#xac00;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.. &#xc774; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xb294; &#xc6d0;&#xbcf8;
&#xb798;&#xc2a4;&#xd130;&#xc640; &#xb3d9;&#xc77c;&#xd55c; &#xc9c0;&#xb9ac;&#xcc38;&#xc870;,, &#xb108;&#xbe44;
&#xbc0f; &#xb192;&#xc774;&#xb97c; &#xac00;&#xc9c8; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.. &#xc9c0;&#xc815;&#xd558;&#
&#xc54a;&#xc740; &#xbc34;&#xb4dc;&#xb4e4;&#xc740; &#xbcc0;&#xacbd; &#xc5c6;&#xc774; &#xbc18;&#xd658;&#xb420;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.. &#xc720;&#xd6a8;&#xd55c; &#xc7ac;&#xbd84;&#xb958; &#xd45c;&#xd604;&#xc2dd;&#
&#xb300;&#xd55c; &#xc124;&#xba85;&#xc774; &#xd544;&#xc694;&#xd55c; &#xacbd;&#xc6b0; reclassarg &#xb97c; &#xcc38;&#
&#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbc34;&#xb4dc;&#xb4e4;&#xc740; pixeltype &#xd53d;&#xc140;
&#xc720;&#xd615;&#xc77c; &#xac83;&#xc785;&#xb2c8;&#xb2e4;. reclassargset &#xb97c; &#xb118;&#xaca8;&#xbc1b;&#
&#xacbd;&#xc6b0;,, &#xac01; reclassarg(&#xc7ac;&#xbd84;&#xb958; &#xc778;&#xc218;)&#xac00; &#xc0dd;&#xc131;&#xb41c;
&#xac01; &#xbc34;&#xb4dc;&#xc758; &#xc2b5;&#xc131;&#xc744; &#xc815;&#xc758;&#xd569;&#xb2c8;&#xb2e4;.
2.0.0 &#xbc84;&#xc804;&#xbd80;&#xd130; &#xc0ac;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.

```

기본 예시

```

&#xc6d0;&#xbcf8;&#xc758; &#xbc34;&#xb4dc; 2&#xb97c; 8BUI&#xc5d0;&#xc11c; 4BUI&#xb85c; &#xbcc0;&#xd658;&#xd558;
101&#xc5d0;&#xc11c; 254&#xae4c;&#xc9c0;&#xc758; &#xbaa8;&#xb4e0; &#xac12;&#xc744; NODATA &#xac12;&#xc73c;&#xb
&#xc124;&#xc815;&#xd55c; &#xc0c8; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.

```

```

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

**재분류 인수를 여러 개 이용하는
고급 예시**

```

&#xc6d0;&#xbcf8; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbc34;&#xb4dc; 1, 2, 3&#xc744; &#xac01;&#xac01; 1BB,
4BUI, 4BUI&#xb85c; &#xbcc0;&#xd658;&#xd558;&#xace0; &#xc7ac;&#xbd84;&#xb958;&#xd55c; &#xc0c8; &#xb798;&#xc2a4;&#
&#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.. &#xc774; &#xc608;&#xc2dc;&#xac00; (&#xc774;&#xb860;&#xc801;&#xc73c;&#
&#xc0ac;&#xc6a9;&#xc790;&#xac00; &#xac00;&#xc9c4; &#xbc34;&#xb4dc;&#xc758; &#xac1c;&#xc218;&#xb9cc;&#xd07c;)
&#xbb34;&#xd55c;&#xd55c; &#xac1c;&#xc218;&#xc758; &#xc7ac;&#xbd84;&#xb958; &#xc778;&#xc218;&#xb97c; &#xc785;&#
&#xc218; &#xc788;&#xb294;,, &#xb2e4;&#xc591;&#xd55c; &#xac1c;&#xc218;&#xc758; &#xc778;&#xc218;&#xb97c; &#xc785;&#
reclassarg &#xc778;&#xc218;&#xb97c; &#xc774;&#xc6a9;&#xd55c;&#xb2e4;&#xb294; &#xc810;&#xc5d0; &#xc8fc;&#xc758

```

```

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

```

32BF &#xbc34;&#xb4dc;&#xb97c; &#xac00;&#xc9c4; &#b798;&#xc2a4;&#xd130;&#xb97c; &#bcf4;&#xae3
&#xc88b;&#xc740; &#bc34;&#xb4dc; &#xc5ec;&#xb7ec; &#xac1c;&#xc5d0; &#b9e4;&#xd551;&#xd558;&#xb294; &#ace0;&#
&#xc608;&#xc2dc;

```

```

32BF &#xbc34;&#xb4dc; &#xd558;&#xb098;&#xb9cc; &#xac00;&#xc9c4; &#b798;&#xc2a4;&#xd130;&#xb85c;&#xbd80;&#xd1
&#xc0c8;&#xb85c;&#xc6b4; &#bc34;&#xb4dc; 3&#xac1c;((8BUI,8BUI,8BUI)&#xb97c; &#xac00;&#xc9c4; &#bcf4;&#xae30;
&#xc88b;&#xc740; &#xc0c8; &#b798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.

```

```

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)
    ]
    );

```

```

&#xcc38;&#xace0;

```

[ST_AddBand](#), [ST_Band](#), [ST_BandPixelType](#), [ST_MakeEmptyRaster](#), [reclassarg](#), [ST_Value](#)

12.12.13 ST_Union

```

ST_Union — &#xb798;&#xc2a4;&#xd130; &#xd0c0;&#xc77c; &#xc9d1;&#xd569;&#xc744; 1&#xac1c; &#xc774;&#xc0c1;&#xc75
&#xbc34;&#xb4dc;&#xb85c; &#xc774;&#xb8e8;&#xc5b4;&#xc9c4; &#xb2e8;&#xc77c; &#b798;&#xc2a4;&#xd130;&#xb85c;
&#xd1b5;&#xd569;&#xd569;&#xb2c8;&#xb2e4;.

```

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);

```



```
WHERE ST_Intersects(rast, ST_GeomFromText('LINESTRING(230486 887771, 230500 88772)',26986)
);
```

```
&#xc608;&#xc2dc;: &#xd0c0;&#xc77c;&#xc774; &#xb3c4;&#xd615;&#xacfc; &#xad50;&#xcc28;&#xd558;&#xb294; &#xbd80;&#&#xd1b5;&#xd569;&#xd55c; &#xb2e4;&#xc911; &#xbc34;&#xb4dc; &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xbc18;&#xd658;
```

```
&#xb2e4;&#xc74c;&#xc740; &#xbc34;&#xb4dc;&#xb4e4;&#xc758; &#xd558;&#xc704; &#xc9d1;&#xd569;&#xb9cc;&#xc744;
&#xc6d0;&#xd558;&#xac70;&#xb098;, &#xb610;&#xb294; &#xbc34;&#xb4dc;&#xb4e4;&#xc758; &#xc21c;&#xc11c;&#xb97c;
&#xbcc0;&#xacbd;&#xd558;&#xace0;&#xc790; &#xd558;&#xb294; &#xacbd;&#xc6b0; &#xb354; &#xae34; &#xbb38;&#xbc95;&#&#xc0ac;&#xc6a9;&#xd558;&#xb294; &#xc608;&#xc2dc;&#xc785;&#xb2c8;&#xb2e4;.
```

```
-- &#xb77c;&#xc778;&#xacfc; &#xad50;&#xcc28;&#xd558;&#xb294; &#baa8;&#xb4e0; ←
&#xd0c0;&#xc77c;&#xb4e4;&#xc744; &#baa8;&#xc740; &#xb2e4;&#xc911; &#bc34;&#xb4dc; ←
&#xb798;&#xc2a4;&#xd130;&#xb97c; &#xc0dd;&#xc131;&#xd569;&#xb2c8;&#xb2e4;.
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)
);
```

참고

[unionarg](#), [ST_Envelope](#), [ST_ConvexHull](#), [ST_Clip](#), [ST_Union](#)

12.13 내장 맵 대수 콜백 함수

12.13.1 ST_Distinct4ma

ST_Distinct4ma — 이웃에서 유일한 픽셀 값들&#개수를 계산하는 래스터 공간 처&#함수입니다.

Synopsis

float8 **ST_Distinct4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);

double precision **ST_Distinct4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

설명

픽셀의 이웃에 있는 유일한 픽셀
값들의 개수를 계산합니다.



Note

변종 1은 [ST_MapAlgebraFctNgb](#) 의 콜백
파라미터로 쓰이는 특화된
콜백 함수입니다.



Note

변종 2는 [내장 맵 대수 콜백 함수](#)
의 콜백 파라미터로 쓰이는
특화된 콜백 함수입니다.



Warning

2.1.0 ಄전඀터 **ST_MapAlgebraFctNgb** 를 지원하지않기 때문에 변종 1 함수를 쓰지않는 편이 좋습니다.

2.0.0 ಄전부터 사용할 수 있습니다.

개선 사항: 2.1.0 버전에서 변종2가 추&#xa

예시

```

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)

```

సೠ

ST_MapAlgebraFctNgb, ST_Min4ma, ST_Max4ma, ST_Sum4ma, ST_Mean4ma, ST_Distinct4ma, ST_StdDev4ma

12.13.2 ST_InvDistWeight4ma

ST_InvDistWeight4ma — 픽셀의 이웃으로부터 픽셀값을 보간하는 래스터 공간 처리 함수입니다.

Synopsis

```
double precision ST_InvDistWeight4ma(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);
```

설명

[illegible]

```
&#xad00;&#xc2ec;&#xd53d;&#xc140;&#xc774;&#xac12;&#xc744;&#xac00;&#xc9c0;&#xace0;&#xc788;&#xace0;&#xb530;&#xc124;&#xc815;&#xd558;&#xc9c0;&#xc54a;&#xc744;&#xacbd;&#xc6b0;&#xd574;&#xb2f9;&#xac12;&#xc744;&#xbc18;&#xc700;&#xb9ac;&#xc5ed;&#xc0b0;&#xac00;&#xc911;&#xbc95;&#xc758;&#xae30;&#xbcf8;&#xbc29;&#xc815;&#xc2dd;&#xb2e4;&#xc74c;&#xacfc;&#xac19;&#xc2b5;&#xb2c8;&#xb2e4;
```

$$\hat{z}(x_o) = \frac{\sum_{j=1}^m z(x_j) d_{ij}^{-k}}{\sum_{j=1}^m d_{ij}^{-k}}$$

$k = \frac{1}{\cos \phi}$ (power factor), $0 \leq k \leq 1$; ϕ is the phase angle between the voltage and the current; $\phi = 0$ for a purely resistive load, $\phi = \pm \frac{\pi}{2}$ for a purely inductive or capacitive load, respectively.



Note

이 함수는 내장 맵 대수 콜백
함수 의 콜백 파라미터로
쓰이는 특화된 콜백 함수입니다

2.1.0 ಄전부터 사용할 수 있습니다.

예시

```
-- &#xc608;&#xc2dc;&#xac00; &#xd544;&#xc694;&#xd568;
```

స고

내장 맵 대수 콜백 함수, ST_MinDist4ma

12.13.3 ST Max4ma

ST_Max4ma — 이웃에서 최대 픽셀 값을 계&#xc0
래스터 공간 처리 함수입니다.

Synopsis

```
float8 ST_Max4ma(float8[][] matrix, text nodatamode, text[] VARIADIC args);
```

```
double precision ST_Max4ma(double precision[] value, integer[] pos, text[] VARIADIC userargs);
```

설명

픽셀의이웃에있는최대픽셀값계산합니다.

```
&#x00000000; &#xc885; 2&#xc758; &#xacbd; &#xc6b0;, &#xd574; &#xb2f9; &#xac12; &#xc744; userargs &#xc5d0; &#xb118; &#xacaf  
NODATA &#xd53d; &#xc140; &#xc758; &#xb300; &#xccb4; &#xac12; &#xc744; &#xc9c0; &#xc815; &#xd560; &#xc218; &#xc788; &
```



Note

```
&#xbcc0;&#xc885;      1&#xc740;      ST_MapAlgebraFctNgb      &#xc758;      &#xcfc5c;&#xbc31;
&#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb85c;      &#xc4f0;&#xc774;&#xb294;      &#xd2b9;&#xd654;&#xb41c;
&#xcfc5c;&#xbc31; &#xd568;&#xc218;&#xc785;&#xb2c8;&#xb2e4;
```

**Note**

2.0.0: `ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'st_max4ma(float[], text, text[])'::regprocedure, 'ignore', NULL), 2, 2`

**Warning**

2.1.0: `ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'st_max4ma(float[], text, text[])'::regprocedure, 'ignore', NULL), 2, 2`

2.0.0: `ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'st_max4ma(float[], text, text[])'::regprocedure, 'ignore', NULL), 2, 2`

ST_MapAlgebraFctNgb

```
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)
```

ST_MapAlgebraFctNgb

`ST_MapAlgebraFctNgb(rast, 1, NULL, 1, 1, 'st_max4ma(float[], text, text[])'::regprocedure, 'ignore', NULL), 2, 2`

12.13.4 ST_Mean4ma

`ST_Mean4ma(rast, 1, NULL, 1, 1, 'st_mean4ma(float[], text, text[])'::regprocedure, 'ignore', NULL), 2, 2`

Synopsis

`float8 ST_Mean4ma(float8[][] matrix, text nodatamode, text[] VARIADIC args);`

`double precision ST_Mean4ma(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);`

ST_Mean4ma

`ST_Mean4ma(rast, 1, NULL, 1, 1, 'st_mean4ma(float[], text, text[])'::regprocedure, 'ignore', NULL), 2, 2`

`ST_Mean4ma(rast, 1, NULL, 1, 1, 'st_mean4ma(float[], text, text[])'::regprocedure, 'ignore', NULL), 2, 2`

**Note**

`ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

**Note**

`ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

**Warning**

2.1.0 `ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

2.0.0 `ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

`ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

`ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

```
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)
```

`ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

```
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)
```

`ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

`ST_MapAlgebraFctNgb` is a function that takes a raster and a text value as input and returns a raster. The text value is a SQL expression that is evaluated for each pixel of the raster. The expression can use the `ST_Value` function to get the value of the pixel at the current location, and the `ST_Neighbor` function to get the values of the pixels in the neighborhood of the current pixel. The function is useful for performing spatial analysis on rasters.

12.13.5 ST_Min4ma

ST_Min4ma — Returns the minimum value of the four input raster bands. The input rasters must be of the same data type and have the same spatial reference. The output raster will have the same spatial reference as the input rasters.

Synopsis

float8 **ST_Min4ma**(float8[][] matrix, text nodatamode, text[] VARIADIC args);

double precision **ST_Min4ma**(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);

ST_Min4ma(rast, 1, 1, 1, 1, 'st_min4ma(float8[],text[],text[])':: ↵

regprocedure, 'ignore', NULL), 2, 2)

FROM dummy_rast
WHERE rid = 2;
rid | st_value
----+-----
2 | 250
(1 row)



Note

The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function. The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function. The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function.



Note

The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function. The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function. The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function.



Warning

The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function. The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function. The **ST_Min4ma** function is a wrapper around the **ST_MapAlgebraFctNgb** function.

2.0.0 **ST_Min4ma**(rast, 1, 1, 1, 1, 'st_min4ma(float8[],text[],text[])':: ↵

regprocedure, 'ignore', NULL), 2, 2)

ST_Min4ma(rast, 1, 1, 1, 1, 'st_min4ma(float8[],text[],text[])':: ↵

```
SELECT
  rid,
  st_value(
    st_mapalgebrafctngb(rast, 1, NULL, 1, 1, 'st_min4ma(float8[],text[],text[])':: ↵
    regprocedure, 'ignore', NULL), 2, 2
  )
FROM dummy_rast
WHERE rid = 2;
rid | st_value
----+-----
2 | 250
(1 row)
```


స고

ST_MapAlgebraFctNgb, ST_Max4ma, ST_Sum4ma, ST_Mean4ma, ST_Range4ma, ST_Distinct4ma, ST_StdDev4ma

12.13.6 ST_MinDist4ma

ST_MinDist4ma — 관심 픽셀과 값을 가지고 있이웃 픽셀 사이의 최소 거리를 (픽개수로) 반환하는 래스터 공간 처함수입니다.

Synopsis

```
double precision ST_MinDist4ma(double precision[][][] value, integer[][] pos, text[] VARIADIC userargs);
```

설명

관심 픽셀과 이웃에서 값을 가지있는 가장 가까운 픽셀 사이의 최거리를 (픽셀 개수로) 반환합니다.

Note

```
&#xc774;      &#xd568;&#xc218;&#xc758;      &#xbaa9;&#xc801;&#xc740;      ST_InvDistWeight4ma
&#xd568;&#xc218;&#xac00; &#xbc18;&#xd658;&#xd558;&#xb294; &#xad00;&#xc2ec; &#xd53d;&#xc140;&#xc758;
&#xbcf4;&#xac04;&#xac12;&#xc758; &#xc720;&#xc6a9;&#xc131;&#xc744; &#xcd94;&#xb860;&#xd558;&#xb294;
&#xac83;&#xc744; &#xb3c4;&#xc640;&#xc8fc;&#xb294; &#xc720;&#xc775;&#xd55c; &#xb370;&#xc774;&#xd130;
&#xd3ec;&#xc778;&#xd2b8;&#xb97c; &#xc81c;&#xacf5;&#xd558;&#xb294; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
&#xc774;&#xc6c3;&#xc758; &#xbc00;&#xb3c4;&#xac00; &#xd76c;&#xbc15;&#xd55c; &#xacbd;&#xc6b0; &#xc774;
&#xd568;&#xc218;&#xac00; &#xd2b9;&#xd788; &#xc720;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
```

Note

```
&#xc774; &#xd568;&#xc218;&#xb294; &#xb0b4;&#xc7a5; &#xb9f5; &#xb300;&#xc218; &#xcf5c;&#xbc31;
&#xd568;&#xc218; &#xc758; &#xcf5c;&#xbc31; &#xd30c;&#xb77c;&#xbbf8;&#xd130;&#xb85c;
&#xc4f0;&#xc774;&#xb294; &#xd2b9;&#xd654;&#xb41c; &#xcf5c;&#xbc31; &#xd568;&#xc218;&#xc785;&#xb2c8;&#xb2e4;.
```

2.1.0 버전부터 사용할 수 있습니다.

예시

```
-- &#xc608;&#xc2dc;&#xac00; &#xd544;&#xc694;&#xd568;
```

స고

내장 맵 대수 콜백 함수., ST InvDistWeight4ma

12.13.7 ST Range4ma

ST_Range4ma — 이웃에 있는 픽셀값들의 범의 는 하는 래스터 공간 ಘ리 함수의

Synopsis

```
float8 ST_Range4ma(float8[][] matrix, text nodatamode, text[] VARIADIC args);
```

```
double precision ST_Range4ma(double precision[] value, integer[] pos, text[] VARIADIC userargs);
```

설명

픽셀의 이웃에 있는 픽셀 값들의
범위를 계산합니다.

```
&#x00000000; &#xc885; 2&#xc758; &#xacbd;&#xc6b0;, &#xd574;&#xb2f9; &#xac12;&#xc744; userargs &#xc5d0; &#xb118;&#xacaf; NODATA &#xd53d;&#xc140;&#xc758; &#xb300;&#xccb4;&#xac12;&#xc744; &#xc9c0;&#xc815;&#xd560; &#xc218; &#xc788;&
```



Note

변종 1은 ST_MapAlgebraFctNgb 의 콜백
파라미터로 쓰이는 특화된
콜백 함수입니다.



Note

변종 2는 내장 맵 대수 콜백 함수
의 콜백 파라미터로 쓰이는
특화된 콜백 함수입니다.



Warning

2.1.0 ಄전부터 **ST_MapAlgebraFctNgb** 를 지원하지
않기 때문에 변종 1 함수를 쓰지
않는 편이 좋습니다.

2.0.0 버전부터 사용할 수 있습니다.

개선 사&#xd56d:: 2.1.0 버전에서 변종 2가 추&#xa

예시

```
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)
```

12.13.8 ST_StdDev4ma

ST_StdDev4ma —이웃에 있는 픽셀값들의 표의 편న를 계산하는 래스터 공간 처의 함수입니다.

Synopsis

```
float8 ST_StdDev4ma(float8[][] matrix, text nodatamode, text[] VARIADIC args);
```

```
double precision ST_StdDev4ma(double precision[][][] value, integer[] pos, text[] VARIADIC userargs);
```

설명

픽셀의 이웃에 있는 픽셀 값들의 표준 편차를 계산합니다.



Note

변종 1은 ST_MapAlgebraFctNgb 의 콜백
 파라미터로 쓰이는 특화된
 콜백 함수입니다.



Note

변종 2는 내장 맵 대수 콜백 함수
의 콜백 파라򻯸터로 쓰이는
특화된 콜백 함수입니다.



Warning

2.1.0 ಄전부터 **ST_MapAlgebraFctNgb** 를 지원하지
않기 때문에 변종 1 함수를 쓰지
않는 편이 좋습니다.

2.0.0 버전부터 사용할 수 있습니다.

개선 사항: 2.1.0 버전에서 변종 2가 추&#xa

예&#xc2dc:

```
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)
```

ST_Min4ma

ST_Max4ma, **ST_Sum4ma**, **ST_Mean4ma**, **ST_Distinct4ma**, **ST_StdDev4ma**

12.13.9 ST_Sum4ma

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);

Synopsis

float8 ST_Sum4ma(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);

ST_Sum4ma

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);



Note

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);



Note

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);



Warning

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);

ST_Sum4ma — **ST_Sum4ma**(**float8**[][] **matrix**, **text** **nodatamode**, **text**[] **VARIADIC args**);
double precision ST_Sum4ma(**double precision**[][][] **value**, **integer**[][] **pos**, **text**[] **VARIADIC userargs**);

ST_Sum4ma

```
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)
```

ST_MapAlgebraFctNgb,

ST_MapAlgebraFctNgb, **ST_Min4ma,** **ST_Max4ma,** **ST_Mean4ma,** **ST_Range4ma,** **ST_Distinct4ma,** **ST_StdDev4ma**

12.14 ST_Aspect

12.14.1 ST_Aspect

ST_Aspect — Returns the aspect (azimuth) of a raster cell. The aspect is calculated as the angle of the steepest descent from the cell, measured in degrees clockwise from North (0 degrees). The aspect is calculated using the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees.

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);

ST_Aspect

ST_Aspect(rast, band, pixeltype, units, customextent) — Returns the aspect (azimuth) of a raster cell. The aspect is calculated as the angle of the steepest descent from the cell, measured in degrees clockwise from North (0 degrees). The aspect is calculated using the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees.

ST_Aspect(rast, band, pixeltype, units, customextent) — Returns the aspect (azimuth) of a raster cell. The aspect is calculated as the angle of the steepest descent from the cell, measured in degrees clockwise from North (0 degrees). The aspect is calculated using the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees.

ST_Aspect(rast, band, pixeltype, units, customextent) — Returns the aspect (azimuth) of a raster cell. The aspect is calculated as the angle of the steepest descent from the cell, measured in degrees clockwise from North (0 degrees). The aspect is calculated using the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees.

ST_Aspect(rast, band, pixeltype, units, customextent) — Returns the aspect (azimuth) of a raster cell. The aspect is calculated as the angle of the steepest descent from the cell, measured in degrees clockwise from North (0 degrees). The aspect is calculated using the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees.



Note

ST_Aspect uses the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees. The units parameter can be set to 'DEGREES' or 'RADIANS'. The customextent parameter can be used to specify a custom extent for the calculation.

ST_Aspect(rast, band, pixeltype, units, customextent) — Returns the aspect (azimuth) of a raster cell. The aspect is calculated as the angle of the steepest descent from the cell, measured in degrees clockwise from North (0 degrees). The aspect is calculated using the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees.

ST_Aspect(rast, band, pixeltype, units, customextent) — Returns the aspect (azimuth) of a raster cell. The aspect is calculated as the angle of the steepest descent from the cell, measured in degrees clockwise from North (0 degrees). The aspect is calculated using the formula: $\text{aspect} = \arctan2(-\text{slope}_y, \text{slope}_x) \times \frac{180}{\pi}$. The aspect is returned in degrees.

```
&#x00c0;&#x00d6;&#x00e2;&#x00f8;&#x0134;&#x014a;&#x015e;&#x0174;&#x0192;&#x01aa;&#x01b8;&#x01d6;&#x01f4;&#x0212;&#x0228;&#x0246;&#x0264;&#x0282;&#x02a0;&#x02b6;&#x02d4;&#x02f2;&#x0310;&#x0326;&#x0344;&#x0362;&#x0380;&#x039e;&#x03b4;&#x03d2;&#x03f0;&#x040e;&#x0424;&#x0442;&#x0460;&#x047e;&#x0494;&#x04b2;&#x04d0;&#x04ee;&#x050c;&#x052a;&#x0548;&#x0566;&#x0584;&#x05a2;&#x05c0;&#x05de;&#x05fc;&#x0632;&#x0650;&#x0668;&#x0686;&#x06a4;&#x06c2;&#x06e0;&#x06fe;&#x071c;&#x0732;&#x0750;&#x0768;&#x0786;&#x07a4;&#x07c2;&#x07e0;&#x07fe;&#x083c;&#x085a;&#x0878;&#x0896;&#x08b4;&#x08d2;&#x08f0;&#x090e;&#x0924;&#x0942;&#x0960;&#x097e;&#x0994;&#x09b2;&#x09d0;&#x09ee;&#x0a0c;&#x0a2a;&#x0a48;&#x0a66;&#x0a84;&#x0aa2;&#x0ac0;&#x0ade;&#x0af8;&#x0b16;&#x0b34;&#x0b52;&#x0b70;&#x0b8e;&#x0ba4;&#x0bc2;&#x0be0;&#x0bf8;&#x0c16;&#x0c34;&#x0c52;&#x0c70;&#x0c8e;&#x0cae;&#x0cc4;&#x0ce2;&#x0cf0;&#x0d0e;&#x0d24;&#x0d42;&#x0d60;&#x0d7e;&#x0d94;&#x0db2;&#x0dd0;&#x0dee;&#x0e0c;&#x0e2a;&#x0e48;&#x0e66;&#x0e84;&#x0ea2;&#x0ec0;&#x0ede;&#x0ef8;&#x0f16;&#x0f34;&#x0f52;&#x0f70;&#x0f8e;&#x0fa4;&#x0fc2;&#x0fe0;&#x0ff8;&#x1016;&#x1034;&#x1052;&#x1070;&#x108e;&#x10ae;&#x10c4;&#x10e2;&#x10f0;&#x110e;&#x1124;&#x1142;&#x1160;&#x117e;&#x1194;&#x11b2;&#x11d0;&#x11ee;&#x120c;&#x122a;&#x1248;&#x1266;&#x1284;&#x12a2;&#x12c0;&#x12de;&#x12fc;&#x1332;&#x1350;&#x1368;&#x1386;&#x13a4;&#x13c2;&#x13e0;&#x13fe;&#x141c;&#x1432;&#x1450;&#x1468;&#x1486;&#x14a4;&#x14c2;&#x14e0;&#x14fe;&#x151c;&#x1532;&#x1550;&#x1568;&#x1586;&#x15a4;&#x15c2;&#x15e0;&#x15fe;&#x163c;&#x165a;&#x1678;&#x1696;&#x16b4;&#x16d2;&#x16f0;&#x170e;&#x1724;&#x1742;&#x1760;&#x177e;&#x1794;&#x17b2;&#x17d0;&#x17ee;&#x180c;&#x182a;&#x1848;&#x1866;&#x1884;&#x18a2;&#x18c0;&#x18de;&#x18fc;&#x193c;&#x195a;&#x1978;&#x1996;&#x19b4;&#x19d2;&#x19f0;&#x1a0e;&#x1a24;&#x1a42;&#x1a60;&#x1a7e;&#x1a94;&#x1ab2;&#x1ad0;&#x1aee;&#x1b0c;&#x1b2a;&#x1b48;&#x1b66;&#x1b84;&#x1ba2;&#x1bc0;&#x1bde;&#x1bfc;&#x1c3c;&#x1c5a;&#x1c78;&#x1c96;&#x1cb4;&#x1cd2;&#x1cf0;&#x1d0e;&#x1d24;&#x1d42;&#x1d60;&#x1d7e;&#x1d94;&#x1db2;&#x1dd0;&#x1dee;&#x1e0c;&#x1e2a;&#x1e48;&#x1e66;&#x1e84;&#x1ea2;&#x1ec0;&#x1ede;&#x1ef8;&#x1f16;&#x1f34;&#x1f52;&#x1f70;&#x1f8e;&#x1fa4;&#x1fc2;&#x1fe0;&#x1ff8;&#x201c;&#x203a;&#x2052;&#x2070;&#x208e;&#x20a4;&#x20c2;&#x20e0;&#x20fe;&#x211c;&#x2132;&#x2150;&#x2168;&#x2186;&#x21a4;&#x21c2;&#x21e0;&#x21fe;&#x221c;&#x2232;&#x2250;&#x2268;&#x2286;&#x22a4;&#x22c2;&#x22e0;&#x22fe;&#x231c;&#x2332;&#x2350;&#x2368;&#x2386;&#x23a4;&#x23c2;&#x23e0;&#x23fe;&#x241c;&#x2432;&#x2450;&#x2468;&#x2486;&#x24a4;&#x24c2;&#x24e0;&#x24fe;&#x251c;&#x2532;&#x2550;&#x2568;&#x2586;&#x25a4;&#x25c2;&#x25e0;&#x25fe;&#x261c;&#x2632;&#x2650;&#x2668;&#x2686;&#x26a4;&#x26c2;&#x26e0;&#x26fe;&#x271c;&#x2732;&#x2750;&#x2768;&#x2786;&#x27a4;&#x27c2;&#x27e0;&#x27fe;&#x281c;&#x2832;&#x2850;&#x2868;&#x2886;&#x28a4;&#x28c2;&#x28e0;&#x28fe;&#x291c;&#x2932;&#x2950;&#x2968;&#x2986;&#x29a4;&#x29c2;&#x29e0;&#x29fe;&#x2a1c;&#x2a32;&#x2a50;&#x2a68;&#x2a86;&#x2aa4;&#x2ac2;&#x2ae0;&#x2afe;&#x2b1c;&#x2b32;&#x2b50;&#x2b68;&#x2b86;&#x2ba4;&#x2bc2;&#x2be0;&#x2bfe;&#x2c1c;&#x2c32;&#x2c50;&#x2c68;&#x2c86;&#x2ca4;&#x2cc2;&#x2ce0;&#x2cfe;&#x2d1c;&#x2d32;&#x2d50;&#x2d68;&#x2d86;&#x2da4;&#x2dc2;&#x2de0;&#x2dfe;&#x2e1c;&#x2e32;&#x2e50;&#x2e68;&#x2e86;&#x2ea4;&#x2ec2;&#x2ee0;&#x2efe;&#x2f1c;&#x2f32;&#x2f50;&#x2f68;&#x2f86;&#x2fa4;&#x2fc2;&#x2fe0;&#x2ff8;&#x301c;&#x303a;&#x3052;&#x3070;&#x308e;&#x30a4;&#x30c2;&#x30e0;&#x30fe;&#x311c;&#x3132;&#x3150;&#x3168;&#x3186;&#x31a4;&#x31c2;&#x31e0;&#x31fe;&#x321c;&#x3232;&#x3250;&#x3268;&#x3286;&#x32a4;&#x32c2;&#x32e0;&#x32fe;&#x331c;&#x3332;&#x3350;&#x3368;&#x3386;&#x33a4;&#x33c2;&#x33e0;&#x33fe;&#x341c;&#x3432;&#x3450;&#x3468;&#x3486;&#x34a4;&#x34c2;&#x34e0;&#x34fe;&#x351c;&#x3532;&#x3550;&#x3568;&#x3586
```

예시: 변종 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},{270.434936523438,315,0,45,71.5650482177734},{2227,180,161.565048217773,135}}")
(1 row)
```

예시: 변종 2

໤버리지타일의완전한예시입이쿼리는 PostgreSQL 9.1 이상버전에서만작동합니다.

```

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;

```

ST_HillShade

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);

12.14.2 ST_HillShade

ST_HillShade — Compute hillshade from a raster. The function takes a raster and an optional band number (default 1) and returns a new raster with the hillshade values. The function also takes an optional pixel type (default 32BF), an optional azimuth (default 315 degrees), an optional altitude (default 45 degrees), an optional maximum brightness (default 255), an optional scale (default 1.0), and an optional boolean flag to interpolate nodata values (default FALSE).

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);

ST_HillShade

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);

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);

azimuth — The azimuth angle in degrees. The default is 315 degrees. The azimuth angle is the angle of the sun's rays relative to the horizontal plane. The azimuth angle is measured clockwise from the north.

altitude — The altitude angle in degrees. The default is 45 degrees. The altitude angle is the angle of the sun's rays relative to the vertical plane. The altitude angle is measured counter-clockwise from the vertical.

max_bright — The maximum brightness value. The default is 255. The maximum brightness value is the maximum value of the hillshade output.

scale — The scale factor. The default is 1.0. The scale factor is the ratio of the maximum brightness value to the maximum value of the hillshade output.

interpolate_nodata — A boolean flag to indicate whether to interpolate nodata values. The default is FALSE. If TRUE, the function will interpolate nodata values using the surrounding non-nodata values.

**Note**

How hillshade works

2.0.0 ಄전부터 사용할 수 있습니다.

개선 사항: 2.1.0 ಄전부터 ST_MapAlgebra()를 이용선택적인 interpolate_nodata 함수 파라미터사추가됐습니다.

변경 사항: 2.1.0 &#bbf8;만 ಄전에서는 ଩위௏사도ఁ이 라디안 단위였습니다 2.1.0 ಄전부터 ଩위ఁ사사도ఁ이 도단위로 ଔ뀌었습니다.

예시: 변종 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)
```

예시: 변종 2

쳤಄리지 타일의 완전한 예시입이 쿼리는 PostgreSQL 9.1 이상 ಄전에서만 작동합니다.

```
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;

```

ST_Roughness

ST_Roughness(raster rast, integer nband, raster customextent, text pixeltype='32BF', boolean interpolate_nodata=FALSE)
 Returns a raster of roughness values for the input raster. The roughness is calculated as the standard deviation of the pixel values in the input raster. The output raster has the same extent and pixel type as the input raster. The output raster is of type float4 (single precision) and has a pixel type of '32BF'.

12.14.3 ST_Roughness

ST_Roughness — DEM (Digital Elevation Model) roughness. The roughness is calculated as the standard deviation of the pixel values in the input raster. The output raster has the same extent and pixel type as the input raster. The output raster is of type float4 (single precision) and has a pixel type of '32BF'.

Synopsis

raster **ST_Roughness**(raster rast, integer nband, raster customextent, text pixeltype='32BF', boolean interpolate_nodata=FALSE);

ST_Roughness

DEM (Digital Elevation Model) roughness. The roughness is calculated as the standard deviation of the pixel values in the input raster. The output raster has the same extent and pixel type as the input raster. The output raster is of type float4 (single precision) and has a pixel type of '32BF'.

2.1.0 The roughness is calculated as the standard deviation of the pixel values in the input raster. The output raster has the same extent and pixel type as the input raster. The output raster is of type float4 (single precision) and has a pixel type of '32BF'.

ST_Roughness

```
-- ST_Roughness(raster rast, integer nband, raster customextent, text pixeltype='32BF', boolean interpolate_nodata=FALSE);
```

ST_Roughness

ST_Roughness(raster rast, integer nband, raster customextent, text pixeltype='32BF', boolean interpolate_nodata=FALSE)
 Returns a raster of roughness values for the input raster. The roughness is calculated as the standard deviation of the pixel values in the input raster. The output raster has the same extent and pixel type as the input raster. The output raster is of type float4 (single precision) and has a pixel type of '32BF'.

12.14.4 ST_Slope

ST_Slope — Returns the slope of a raster. The slope is calculated as the square root of the sum of the squares of the horizontal and vertical derivatives. The units of the slope are degrees. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster.

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);

ST_Slope

ST_Slope — Returns the slope of a raster. The slope is calculated as the square root of the sum of the squares of the horizontal and vertical derivatives. The units of the slope are degrees. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster.

units: RADIANS, DEGREES, PERCENT

scale: 1.0, 370400, 111120

interpolate_nodata: FALSE, TRUE



Note

The slope is calculated as the square root of the sum of the squares of the horizontal and vertical derivatives. The units of the slope are degrees. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster.

2.0.0 Returns the slope of a raster. The slope is calculated as the square root of the sum of the squares of the horizontal and vertical derivatives. The units of the slope are degrees. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster.

2.1.0 Returns the slope of a raster. The slope is calculated as the square root of the sum of the squares of the horizontal and vertical derivatives. The units of the slope are degrees. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster.

2.1.0 Returns the slope of a raster. The slope is calculated as the square root of the sum of the squares of the horizontal and vertical derivatives. The units of the slope are degrees. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster. The slope is calculated for each pixel in the raster.

ST_Slope

```
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},{21.5681285858154,35.26438905681285858154,26.5650520324707,21.5681285858154,10.0249881744385}}")
(1 row)

```

Example 2

PostgreSQL 9.1

```

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;

```

ST_TPI

ST_TPI (raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

12.14.5 ST_TPI

ST_TPI — (Topographic Position Index) (raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

Synopsis

raster **ST_TPI**(raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

ST_TPI

Calculates the Topographic Position Index, which is defined as the focal mean with radius of one minus the center cell.



Note

ST_TPI (raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

2.1.0 (raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

ST_TPI

```
-- ST_TPI (raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);
```

ST_TPI

ST_TPI (raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

12.14.6 ST_TRI

ST_TRI — (Terrain Ruggedness Index) (raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

Synopsis

raster **ST_TRI**(raster rast, integer nband, raster customextent, text pixeltype="32BF", boolean interpolate_nodata=FALSE);

box3d

box3d (geometry) — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

**Note**

box3d (geometry) — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

box3d (geometry) — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

box3d

```
-- box3d (geometry) — Returns the 3D bounding box of a geometry.
```

box3d

box3d (geometry) — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

12.15 box3d**12.15.1 Box3D**

Box3D — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

Synopsis

box3d **Box3D**(raster rast);

box3d

box3d (geometry) — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

box3d (geometry) — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

box3d (geometry) — Returns the 3D bounding box of a geometry. The result is a geometry of type `box3d`. The bounding box is defined by the minimum and maximum x, y, and z coordinates of the geometry. The z coordinate is only used for 3D geometries.

예시

```
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)

స고

ST_Envelope

12.15.2 ST_ConvexHull

ST_ConvexHull — BandNoDataValue와 일 치 하 는 픽 셀 값 을 포 함 한, 래 스 터 의 ௏ 록 ® 질 도 형 &#&#xb; ൘ 합 니 다, 정 ­ 형 상 ௏ 기 울 어 &# 은 래 스 터 의 경 우, ST_Envelope와 동 일 한 결 ૏ ଼ 을 출 력 하 므 로 비 정 ­ 형 &#&#xb; 또 는 기 울 어 진 래 스 터 에 대 해 서 &# 쓸 모 가 있 습 니 다.

Synopsis

```
geometry ST_ConvexHull(raster rast);
```

설명

NoDataBandValue 밴드 픽셀을 포함한 래스터&#
௏ 록 ® 질 도형을 ଘ환합 니다 정&#
형상 ௏ 기울어지지 않은 래스터&#
경우 와 ౰의 동일한 결과물을
ග력하므로 비정­ 형상 또는 기&#
래스터에 대해서만 쓸모ఀ 있습&#x

Note



```
ST_Envelope &#xd568;&#xc218;&#xb294; &#xc88c;&#xd45c;&#xb97c; &#xbc84;&#xb9bc;(floor)&#xd558;&#xae30;
&#xb54c;&#xbb38;&#xc5d0; &#xb798;&#xc2a4;&#xd130; &#xc8fc;&#xc704;&#xc5d0; &#xbc84;&#xd37c;&#xb97c;
&#xc57d;&#xac04; &#xc8fc;&#xac8c; &#xb429;&#xb2c8;&#xb2e4;. &#xb530;&#xb77c;&#xc11c;
&#xbc84;&#xb9bc;&#xc744; &#xd558;&#xc9c0; &#xc54a;&#xb294; ST_ConvexHull
&#xd568;&#xc218;&#xc640;&#xb294; &#xc57d;&#xac04; &#xb2e4;&#xb978; &#xacb0;&#xacfc;&#xb97c;
&#xb0c5;&#xb2c8;&#xb2e4;.
```

예시

PostGIS Raster Specification

```
-- ST_ConvexHull(rast) As convhull,
-- 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))

```
-- ST_ConvexHull(rast) As convhull,
-- 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))

ST_ConvexHull

ST_Envelope, ST_MinConvexHull, ST_ConvexHull, ST_AsText

12.15.3 ST_DumpAsPolygons

ST_DumpAsPolygons — Returns a set of polygons from a raster. The output is a set of polygons, one for each pixel in the raster. The output is a set of polygons, one for each pixel in the raster. The output is a set of polygons, one for each pixel in the raster.

Synopsis

setof geomval **ST_DumpAsPolygons**(raster rast, integer band_num=1, boolean exclude_nodata_value=TRUE);

ST_DumpAsPolygons

ST_DumpAsPolygons — Returns a set of polygons from a raster. The output is a set of polygons, one for each pixel in the raster. The output is a set of polygons, one for each pixel in the raster. The output is a set of polygons, one for each pixel in the raster.

ST_DumpAsPolygon — Returns a single polygon from a raster. The output is a single polygon, one for each pixel in the raster. The output is a single polygon, one for each pixel in the raster. The output is a single polygon, one for each pixel in the raster.

점에서 GROUP BY의 역함수라고 할수있예를 들어 단일 래스터를 복수의 폴리곤/멀티폴리곤으로 확장하&#xb데 이 함수를 쓸 수 있습니다.

Changed 3.3.0, validation and fixing is disabled to improve performance. May result invalid geometries.

GDAL 1.7 이상 버전이 필요합니다.

Note!

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

```
&#xb798;&#xc2a4;&#xd130; &#xc548;&#xc5d0; &#xc788;&#xb294; &#xd574;&#xb2f9; &#xac12;&#xc744;
&#xac00;&#xc9c4; &#xd53d;&#xc140;&#xb4e4;&#xc758; &#xac1c;&#xc218;&#xc5d0;&#xb9cc;
&#xad00;&#xc2ec;&#xc774; &#xc788;&#xb2e4;&#xba74;, ST_ValueCount &#xd568;&#xc218;&#xac00; &#xb354;
&#xbe60;&#xb985;&#xb2c8;&#xb2e4;.
```

Note!

Note

```
&#xc774; &#xd568;&#xc218;&#xb294; &#xd53d;&#xc140; &#xac12;&#xacfc; &#xc0c1;&#xad00;&#xc5c6;&#xc774;
&#xac01; &#xd53d;&#xc140;&#xc5d0; &#xb300;&#xd574; &#xb3c4;&#xd615; &#xd558;&#xb098;&#xb97c;
&#xbc18;&#xd658;&#xd558;&#xb294; ST_PixelAsPolygons &#xd568;&#xc218;&#xc640;
&#xb2e4;&#xb985;&#xb2c8;&#xb2e4;.
```

예시

```
-- 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))

స&#xace0:

geomval, ST_Value, ST_Polygon, ST_ValueCount

12.15.4 ST_Envelope

ST_Envelope — 래스터 &#bc94;위의 폴리곤 표현&#xc2
&#bc18;환합니다.

Synopsis

```
geometry ST_Envelope(raster rast);
```

설명

래스터 범위의 폴리곤 표현식을 SRID가 정의하는 공간 좌표 단위로 반환합니다. 여기에서 폴리곤 표폴리곤으로 표현된 float8 형 최소치 경계 상자입니다.

해당 폴리곤은 경계 상자의 꼭짓&#x
포인트들로 정의됩니다((MINX, MINY), (MINX, MAXY),
(MAXX, MAXY), (MAXX, MINY), (MINX, MINY)).

예시

```
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))

స고

ST_Envelope, ST_AsText, ST_SRID

12.15.5 ST MinConvexHull

ST_MinConvexHull — &#x798;&#xc2a4;&#xd130;&#xc758; NODATA &#xd53d;&#xc140;&#xc744; &#xc81c;&#xc678;&#xd55c;&#xbcfc;&#xb85d; &#xaecd;&#xc9c8; &#xb3c4;&#xd615;&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.

Synopsis

```
geometry ST_MinConvexHull(raster rast, integer nband=NULL);
```

설&#xba85:

```
&#xb798;&#xc2a4;&#xd130;&#xc758; NODATA &#xd53d;&#xc140;&#xc744; &#xc81c;&#xc678;&#xd55c; &#xbcf8;&#xb85d;
&#xaecd;&#xc9c8; &#xb3c4;&#xd615;&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. nband &#xac00; NULL&#xc777
&#xacbd;&#xc6b0;. &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbaa8;&#xb4e0; &#xbc34;&#xb4dc;&#xb97c; &#xcc98;&#xb9ac;&#
```

2.1.0 버전부터 사용할 수 있습니다.

예시

```

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,
    ST_AsText(ST_MinConvexHull(rast, 2)) AS mhull_2
FROM foo

```

hull	mhull	mhull_1	mhull_2
POLYGON((0 0,9 0,9 -9,0 -9,0 0))	POLYGON((0 -3,9 -3,9 -9,0 -9,0 -3))	POLYGON((3 -3,9 -3,9 -6,3 -6,3 -3))	POLYGON((0 -3,6 -3,6 -9,0 -9,0 -3))

స고

ST_Envelope, ST_ConvexHull, ST_ConvexHull, ST_AsText

12.15.6 ST Polygon

ST_Polygon — NODATA &#x12;ݴ Մ닌 픽셀 &#x12;݄ &#x00;진 픽셀들을 통합해서 형성된 멀티도형을 반환합니다.

Synopsis

```
geometry ST_Polygon(raster rast, integer band_num=1);
```

설명

0.1.6 ಄전부터 사용할 수 있습니다.
GDAL 1.7 이상 ಄전이 필요합니다.

개선사항. 2.1.0 버전부터 속도가 향&#xc0
C언어 기반으로 변경했습니다). 반
멀티폴리곤이 유효한 도형인지
확인합니다.

변경 사항: 2.1.0 미만 버전에서 종종
폴리곤을 반환하곤 했던 것을 언&#
멀티폴리곤을 반환하도록 바꿨&#xc

예시

```
--  &#xae30;&#xbcfc8;&#xc801;&#xc73c;&#xb85c; NODATA &#xbc34;&#xb4dc; &#xac12;&#xc774; 0  ←
    &#xb610;&#xb294; &#xc124;&#xc815;&#xb418;&#xc9c0; &#xc54a;&#xc740;  ←
    &#xc0c1;&#xd0dc;&#xb85c;, &#xb530;&#xb77c;&#xc11c; &#xc0ac;&#xac01;&#xd615;  ←
    &#xd3f4;&#xb9ac;&#xace4;&#xc744; &#xbcb8;&#xd658;&#xd560;  ←
    &#xac83;&#xc785;&#xb2c8;&#xb2e4; .
```

```
-- <#xb610;#xb294; NODATA <#xac12;#xc744; <#xd55c;#xbc88;#xb9cc; <#xb2e4;#xb974;#xac8c; <#xd558;#xace0; <#xc2f6;#xc744; <#xacbd;#xc6b0;

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)))
```

<#xccc38;#xace0;

ST_Value, ST_DumpAsPolygons

12.16 <#xb798;#xc2a4;#xd130; <#xc5f0;#xc0b0;#xc790;

12.16.1 &&

&& — A<#xc758; <#xacbd;#xacc4; <#xc0c1;#xc790;#xc640; B<#xc758; <#xacbd;#xacc4; <#xc0c1;#xc790;#xac00; <#xad50;#xccc28;#xd558;#xb294; <#xacbd;#xc6b0; TRUE <#xb97c; <#xbc18;#xd658;#xd569;#xb2c8;#xb2e4;.

Synopsis

```
boolean &&( raster A , raster B );
boolean &&( raster A , geometry B );
boolean &&( geometry B , raster A );
```

<#xc124;#xba85;

& <#xc5f0;#xc0b0;#xc790;#xb294; <#xb798;#xc2a4;#xd130;/<#xb3c4;#xd615; A<#xc758; <#xacbd;#xacc4; <#xc0c1;#xc790;#xc640; <#xb798;#xc2a4;#xd130;/<#xb3c4;#xd615; B<#xc758; <#xacbd;#xacc4; <#xc0c1;#xc790;#xad50;#xccc28;#xd558;#xb294; <#xacbd;#xc6b0; TRUE <#xb97c; <#xbc18;#xd658;#xd569;#xb2c8;#xb2e4;.



Note

<#xc774; <#xd53c;#xc5f0;#xc0b0;#xc790;(operand)<#xb294; <#xb798;#xc2a4;#xd130;#xc5d0;#xc11c; <#xc774;#xc6a9;#xd560; <#xc218;#xb3c4; <#xc788;#xb294; <#xbaa8;#xb4e0; <#xc778;#xb371;#xc2a4;#xb97c; <#xd65c;#xc6a9;#xd560; <#xac83;#xc785;#xb2c8;#xb2e4;.

2.0.0 <#xbc84;#xc804;#xbd80;#xd130; <#xc0ac;#xc6a9;#xd560; <#xc218; <#xc788;#xc2b5;#xb2c8;#xb2e4;.

예시

```
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_id | b_id | intersect
-----+-----+-----
    2 |    2 | t
    2 |    3 | f
    2 |    1 | f
```

12.16.2 <

```
&< — A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790; &#xc67c;&#xc788;&#xc744; &#xacbd;&#xc6b0; TRUE &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```

Synopsis

```
boolean &<( raster A , raster B );
```

설명

```
&#x2013; < &#xc5f0;&#xc0b0;&#xc790;&#xb294; &#xb798;&#xc2a4;&#xd130; A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac0
&#xb798;&#xc2a4;&#xd130; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc640; &#xacb9;&#xce58;&#xac70;&#xb098;
&#xadf8; &#xc67c;&#xcabd;&#xc5d0; &#xc788;&#xc744; &#xacbd;&#xc6b0;, &#xb610;&#xb294; &#xb354; &#xc815;&#xd655;&
&#xb9d0;&#xd558;&#xc790;&#xba74; &#xb798;&#xc2a4;&#xd130; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc640;
&#xacb9;&#xce58;&#xac70;&#xb098; &#xadf8; &#xc624;&#xb978;&#xcabd;&#xc5d0; &#xc788;&#xc9c0; &#xc54a;&#xc744;
&#xacbd;&#xc6b0;, TRUE &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```



Note

```
&#xc774; &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294; &#xb798;&#xc2a4;&#xd130;&#xc5d0;&#xc11c;
&#xc774;&#xc6a9;&#xd560; &#xc218;&#xb3c4; &#xc788;&#xb294; &#xbaa8;&#xb4e0;
&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xd65c;&#xc6a9;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

예시

```
SELECT A.rid As a_riD, B.riD As b_riD, A.rast < B.rast As overlft
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

12.16.3 &>

&> — A ݘ 경계 상자가 Bݘ 경계 상자오ވ을 경우 TRUE 를 반환합니다.

Synopsis

boolean **&>**(raster A , raster B);

설명

&> 연산자는 래스터 Aݘ 경계 상자가 래스터 Bݘ 경계 상자와 겹치거나 그 오른쪽에 있을 경우,, 또는 더 정말하자면 래스터 Bݘ 경계 상자와 겹치거나 그 왼쪽에 있지 않을 경우 TRUE 를 반환합니다.



Note

이 피연산자(operand)는 래스터에서 이용할 수도 있는 모든 인덱스를 활용할 것입니다.

예시

```
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

12.16.4 =

= — Aݘ 경계 상자와 Bݘ 경계 상자가 동일할 경우 TRUE 를 반환합니다. 이정밀도 경계 상자를 이용합니다.

Synopsis

boolean **=**(raster A , raster B);

설명

```
= &#xc5f0;&#xc0b0;&#xc790;&#xb294; &#xb798;&#xc2a4;&#xd130; A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc640;
&#xb798;&#xc2a4;&#xd130; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00; &#xb3d9;&#xc77c;&#xd560; &#xacbd;&
TRUE &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. PostgreSQL&#xc740; &#xb798;&#xc2a4;&#xd130;&#xb97c;
&#xb0b4;&#xbd80; &#xc815;&#xb82c;&#xd558;&#xace0; &#xbe44;&#xad50;&#xd558;&#xae30; &#xc704;&#xd574; &#xb3c4;&#
&#xb300;&#xd574; &#xc815;&#xc758;&#xb41c; =, <, &#xbc0f; > &#xc5f0;&#xc0b0;&#xc790;&#xb97c; &#xc774;&#xc6a9;&#xd5
GROUP BY &#xb610;&#xb294; ORDER BY &#xc808;&#xc5d0;&#xc11c;).
```

Caution



```
&#xc774;      &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294;      &#xb798;&#xc2a4;&#xd130;&#xc5d0;
&#xb300;&#xd574;      &#xc774;&#xc6a9;&#xd560;      &#xc218;      &#xc788;&#xc744;&#xc9c0;&#xb3c4;
&#xbaa8;&#xb974;&#xb294;      &#xc5b4;&#xb5a4;      &#xc778;&#xb371;&#xc2a4;&#xb3c4;
&#xd65c;&#xc6a9;&#xd558;&#xc9c0;      &#xc54a;&#xc744;      &#xac83;&#xc785;&#xb2c8;&#xb2e4;
&#xb300;&#xc2e0;      ~=      &#xd568;&#xc218;&#xb97c;      &#xc774;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;
&#xc774;      &#xc5f0;&#xc0b0;&#xc790;&#xc758;      &#xbaa9;&#xc801;&#xc740;      &#xb300;&#xbd80;&#xbd84;
&#xb798;&#xc2a4;&#xd130;      &#xc5f4;&#xb85c;      &#xadf8;&#xb8f9;&#xd654;(group by)&#xd558;&#xb294;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

2.1.0 ಄전부터 사용할 수 있습니다.

స고

 $\approx$

12.16.5 @

```
@ — B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00; A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xb97c;
&#xb2f4;&#xace0; &#xc788;&#xc744; &#xacbd;&#xc6b0; TRUE &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
&#xc774;&#xc911; &#xc815;&#xbc00;&#xb3c4; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xb97c; &#xc774;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
```

Synopsis

```
boolean @( raster A , raster B );
boolean @( geometry A , raster B );
boolean @( raster B , geometry A );
```

설명

@ 연산자는 래스터/도형 B의 경계 상자가 래스터/도형 A의 경계 상자담고 있을 경우 TRUE 를 반환합니다.



Note

```
&#xc774;      &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294;      &#xb798;&#xc2a4;&#xd130;&#xc5d0;
&#xc788;&#xb294;      &#xacf5;&#xac04;      &#xc778;&#xb371;&#xc2a4;&#xb97c;      &#xd65c;&#xc6a9;&#xd560;
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

```
2.0.0 &#xc84;&#xc804;&#xbd80;&#xd130; raster @ raster, raster @ geometry &#xb97c; &#xc0ac;&#xc6a9;&#xd560; &#xc218;
&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

2.0.5 geometry @ raster

󌰸고

~

12.16.6 $\sim =$

```
~== A&#xc758; &#xacd; &#xac4; &#xc0c1; &#xc790; &#xc640; B&#xc758; &#xacd; &#xac4; &#xc0c1; &#xc790; &#xac00;
&#xb3d9; &#xc77c; &#xd560; &#xacd; &#xc6b0; TRUE &#xb97c; &#xbc18; &#xd658; &#xd569; &#xb2c8; &#xb2e4;.
```

Synopsis

```
boolean ~= ( raster A , raster B );
```

설명

```
~=&#xc5f0;&#xc0b0;&#xc790;&#xb294; &#xb798;&#xc2a4;&#xd130; A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xc640; &#xb798;&#xc2a4;&#xd130; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00; &#xb3d9;&#xc77c;&#xd560; &#xacbd;&#xc758; &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
```



Note

```
&#xc774; &#xd53c;&#xc5f0;&#xc0b0;&#xc790;(operand)&#xb294; &#xb798;&#xc2a4;&#xd130;&#xc5d0;&#xc11c;
&#xc774;&#xc6a9;&#xd560; &#xc218;&#xb3c4; &#xc788;&#xb294; &#xbaa8;&#xb4e0;
&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xd65c;&#xc6a9;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;
```

2.0.0 버전부터 사용할 수 있습니다.

예시

동일한୉ౘ이지만서로다른주다표현하는단일򳰴드래스터의집다를입력򳰛아다중򳰴드래스터생성하는데매우유용합니다

```
SELECT ST_AddBand(prec.rast, alt.rast) As new_rast
FROM prec INNER JOIN alt ON (prec.rast ~= alt.rast);
```

స고

ST_AddBand, =

12.16.7 ~

```

~ — A&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xac00; B&#xc758; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xb97c;
&#xb2f4;&#xace0; &#xc788;&#xc744; &#xacbd;&#xc6b0; TRUE &#xb97c; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
&#xc774;&#xc911; &#xc815;&#xbc00;&#xb3c4; &#xacbd;&#xacc4; &#xc0c1;&#xc790;&#xb97c; &#xc774;&#xc6a9;&#xd569;&#

```

Synopsis

```
boolean ~( raster A , raster B );
boolean ~( geometry A , raster B );
boolean ~( raster B , geometry A );
```


ST_Contains

~ ST_Contains(rastA, rastB) returns TRUE if rastA contains rastB. If rastA is NULL, the function returns FALSE. If rastB is NULL, the function returns TRUE. If both rastA and rastB are NULL, the function returns NULL.



Note

ST_Contains(rastA, rastB) returns TRUE if rastA contains rastB. If rastA is NULL, the function returns FALSE. If rastB is NULL, the function returns TRUE. If both rastA and rastB are NULL, the function returns NULL.

2.0.0 ST_Contains(rastA, rastB) returns TRUE if rastA contains rastB.

ST_Contains

@

12.17 ST_Contains

12.17.1 ST_Contains

ST_Contains — ST_Contains(rastA, rastB) returns TRUE if rastA contains rastB. If rastA is NULL, the function returns FALSE. If rastB is NULL, the function returns TRUE. If both rastA and rastB are NULL, the function returns NULL.

Synopsis

boolean **ST_Contains**(raster rastA , integer nbandA , raster rastB , integer nbandB);
boolean **ST_Contains**(raster rastA , raster rastB);

ST_Contains

ST_Contains(rastA, rastB) returns TRUE if rastA contains rastB. If rastA is NULL, the function returns FALSE. If rastB is NULL, the function returns TRUE. If both rastA and rastB are NULL, the function returns NULL.



Note

ST_Contains(rastA, rastB) returns TRUE if rastA contains rastB. If rastA is NULL, the function returns FALSE. If rastB is NULL, the function returns TRUE. If both rastA and rastB are NULL, the function returns NULL.

**Note**

`ST_Contains(ST_Polygon(raster), geometry)` is equivalent to `ST_Contains(geometry, ST_Polygon(raster))`.

**Note**

`ST_Contains()` is equivalent to `ST_Within()`.

2.1.0 `ST_Contains()` is equivalent to `ST_Within()`.

ST_ContainsProperly

```
-- ST_ContainsProperly
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	t
1	2	f

```
-- ST_ContainsProperly
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

ST_ContainsProperly

[ST_ContainsProperly](#), [ST_Within](#)

12.17.2 ST_ContainsProperly

`ST_ContainsProperly` — `ST_ContainsProperly(rastB, rastA)` is equivalent to `ST_Contains(rastB, rastA)` and `ST_Contains(rastA, rastB)`.

Synopsis

boolean `ST_ContainsProperly`(raster rastA , integer nbandA , raster rastB , integer nbandB);
 boolean `ST_ContainsProperly`(raster rastA , raster rastB);

ST_Intersection

ST_Intersection(raster rastA, integer nbandA, raster rastB, integer nbandB)
ST_Intersection(raster rastA, raster rastB)
 Returns the intersection of the two rasters. The output raster has the same number of bands as the input rasters. The output raster is NULL if either input raster is NULL. The output raster is NULL if the two rasters have different numbers of bands. The output raster is NULL if the two rasters have different spatial references. The output raster is NULL if the two rasters have different units. The output raster is NULL if the two rasters have different resolutions. The output raster is NULL if the two rasters have different dimensions. The output raster is NULL if the two rasters have different extents. The output raster is NULL if the two rasters have different origins. The output raster is NULL if the two rasters have different pixel sizes. The output raster is NULL if the two rasters have different pixel types. The output raster is NULL if the two rasters have different pixel values. The output raster is NULL if the two rasters have different pixel colors. The output raster is NULL if the two rasters have different pixel alpha values. The output raster is NULL if the two rasters have different pixel opacities. The output raster is NULL if the two rasters have different pixel transparencies. The output raster is NULL if the two rasters have different pixel colors. The output raster is NULL if the two rasters have different pixel alpha values. The output raster is NULL if the two rasters have different pixel opacities. The output raster is NULL if the two rasters have different pixel transparencies.



Note

The output raster has the same number of bands as the input rasters. The output raster is NULL if either input raster is NULL. The output raster is NULL if the two rasters have different numbers of bands. The output raster is NULL if the two rasters have different spatial references. The output raster is NULL if the two rasters have different units. The output raster is NULL if the two rasters have different resolutions. The output raster is NULL if the two rasters have different dimensions. The output raster is NULL if the two rasters have different extents. The output raster is NULL if the two rasters have different origins. The output raster is NULL if the two rasters have different pixel sizes. The output raster is NULL if the two rasters have different pixel types. The output raster is NULL if the two rasters have different pixel values. The output raster is NULL if the two rasters have different pixel colors. The output raster is NULL if the two rasters have different pixel alpha values. The output raster is NULL if the two rasters have different pixel opacities. The output raster is NULL if the two rasters have different pixel transparencies.



Note

The output raster has the same number of bands as the input rasters. The output raster is NULL if either input raster is NULL. The output raster is NULL if the two rasters have different numbers of bands. The output raster is NULL if the two rasters have different spatial references. The output raster is NULL if the two rasters have different units. The output raster is NULL if the two rasters have different resolutions. The output raster is NULL if the two rasters have different dimensions. The output raster is NULL if the two rasters have different extents. The output raster is NULL if the two rasters have different origins. The output raster is NULL if the two rasters have different pixel sizes. The output raster is NULL if the two rasters have different pixel types. The output raster is NULL if the two rasters have different pixel values. The output raster is NULL if the two rasters have different pixel colors. The output raster is NULL if the two rasters have different pixel alpha values. The output raster is NULL if the two rasters have different pixel opacities. The output raster is NULL if the two rasters have different pixel transparencies.

2.1.0 **ST_Intersection**(raster rastA, integer nbandA, raster rastB, integer nbandB)

ST_Intersection

```
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

ST_Intersection

ST_Intersection, **ST_Covers**

12.17.5 ST_Disjoint

ST_Disjoint — Returns true if the two rasters do not intersect. The output is true if the two rasters have no common pixels. The output is false if the two rasters have at least one common pixel. The output is NULL if either input raster is NULL. The output is NULL if the two rasters have different numbers of bands. The output is NULL if the two rasters have different spatial references. The output is NULL if the two rasters have different units. The output is NULL if the two rasters have different resolutions. The output is NULL if the two rasters have different dimensions. The output is NULL if the two rasters have different extents. The output is NULL if the two rasters have different origins. The output is NULL if the two rasters have different pixel sizes. The output is NULL if the two rasters have different pixel types. The output is NULL if the two rasters have different pixel values. The output is NULL if the two rasters have different pixel colors. The output is NULL if the two rasters have different pixel alpha values. The output is NULL if the two rasters have different pixel opacities. The output is NULL if the two rasters have different pixel transparencies.

Synopsis

boolean **ST_Disjoint**(raster rastA , integer nbandA , raster rastB , integer nbandB);

boolean **ST_Disjoint**(raster rastA , raster rastB);

설명

```
&#xb798;&#xc2a4;&#xd130; rastA&#xc640; &#xb798;&#xc2a4;&#xd130; rastB&#xac00; &#xc5b4;&#xb5a4; &#xacf5;&#xac04;&#xd568;&#xaed8; &#xacf5;&#xc720;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; rastA&#xc640; rastB&#xac00; &#xbd84;&#xb9ac;&#xb410;&#xb2e4;&#xace0; &#xd569;&#xb2c8;&#xb2e4;. &#xbc34;&#xb4dc; &#xbc88;&#xd638;&#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xac70;&#xb098; &#xb610;&#xb294; NULL&#xb85c; &#xc124;&#xc815;&#xd558;&#xacbd;&#xc6b0;. &#xd655;&#xc778; &#xc791;&#xc5c5;&#xc911; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbcf5;&#xb85d; &#xaecd;&#xc9c8;&#xb9cc; &#xace0;&#xb824;&#xd569;&#xb2c8;&#xb2e4;. &#xbc34;&#xb4dc; &#xbc88;&#xd638;&#xb97c; &#xc124;&#xc815;&#xd560; &#xacbd;&#xc6b0;. &#xd655;&#xc778; &#xc791;&#xc5c5;&#xc911; &#xac12;&#xc744; &#xac00;&#xd558; (NODATA&#xac00; &#xc544;&#xb2cc;) &#xd53d;&#xc140;&#xb4e4;&#xb9cc; &#xace0;&#xb824;&#xd569;&#xb2c8;&#xb2e4;.
```



Note

```
&#xc774;      &#xd568;&#xc218;&#xb294;      &#xc5b4;&#xb5a4;      &#xc778;&#xb371;&#xc2a4;&#xb3c4;
&#xd65c;&#xc6a9;&#xd558;&#xc9c0; &#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;
```



Note

```
&#xb798;&#xc2a4;&#xd130;&#xfc640;          &#xb3c4;&#xd615;&#xc758;          &#xacf5;&#xac04;
&#xad00;&#xacc4;&#xc131;&#xc744;          &#xd655;&#xc778;&#xd558;&#xb824;&#xba74;,
ST_Disjoint(ST_Polygon(raster), geometry)&#xcc98;&#xb7fc; &#xb798;&#xc2a4;&#xd130;&#xc5d0; ST_Polygon
&#xd568;&#xc218;&#xb97c; &#xc774;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

2.1.0 버전부터 사용할 수 있습니다.

예시

```
-- rid = 1 &#xc774; &#xbc34;&#xb4dc;&#xb97c; &#xac00;&#xc9c0;&#xace0; &#xc788;&#xc9c0; &#xc54a;&#xae30; &#xb54c;&#xbb38;&#xc5d0;, &#xc548;&#xb0b4;&#xbb38; (NOTICE) &#xc774; &#xcd9c;&#xb825;&#xb418;&#xace0; st_disjoint&#xac00; NULL&#xac12;&#xc744; &#xac00;&#xc9c0;&#xac8c; &#xb429;&#xb2c8;&#xb2e4; .
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

```

```
-- &#xc774;&#xbc88;&#xc5d0;&#xb294; &#xbc34;&#xb4dc; &#xbc88;&#xd638;&#xb97c;  ←  
    &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xc74c;  
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

స고

ST Intersects

12.17.6 ST_Intersects

ST_Intersects — Returns true if the geometries of *rastra* and *rastrb* intersect. If *geommin* is not null, the intersection must be within the bounding box of *geommin*.

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 );
```

Parameters

rastra: raster to be tested.
rastrb: raster to be tested.
nbandA: band number of *rastra* to be tested.
nbandB: band number of *rastrb* to be tested.
geommin: geometry to be tested.
nband: band number of *rastra* to be tested.



Note

As of PostGIS 2.1.0, the `ST_Intersects` function will return true if the geometries of *rastra* and *rastrb* intersect. If *geommin* is not null, the intersection must be within the bounding box of *geommin*.

As of PostGIS 2.1.0, the `ST_Intersects` function will return true if the geometries of *rastra* and *rastrb* intersect. If *geommin* is not null, the intersection must be within the bounding box of *geommin*.



Warning

As of PostGIS 2.1.0, the `ST_Intersects` function will return true if the geometries of *rastra* and *rastrb* intersect. If *geommin* is not null, the intersection must be within the bounding box of *geommin*.

Example

```
-- Create a dummy raster with two bands.
CREATE TABLE dummy_rast (
  rast raster,
  CONSTRAINT rast_srid FOREIGN KEY (srid) REFERENCES spatial_ref_sys(srid)
);
INSERT INTO dummy_rast (rast) VALUES (ST_MakeRasterFromSDE(
  'SELECT ST_Intersects(rast, 2, rast, 3) FROM dummy_rast WHERE rid = 2;
  st_intersects
  -----
  t
  '));
```

See Also

[ST_Intersection](#), [ST_Disjoint](#)

12.17.7 ST_Overlaps

ST_Overlaps — Returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).

Synopsis

boolean **ST_Overlaps**(raster rastA , integer nbandA , raster rastB , integer nbandB);
 boolean **ST_Overlaps**(raster rastA , raster rastB);

ST_Overlaps(raster rastA , integer nbandA , raster rastB , integer nbandB);

ST_Overlaps(raster rastA , raster rastB);

ST_Overlaps returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).



Note

ST_Overlaps returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).



Note

ST_Overlaps returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).

ST_Overlaps returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).

ST_Overlaps returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).

```
-- <--
SELECT ST_Overlaps(rast, 1, rast, 2) FROM dummy_rast WHERE rid = 2;
```

st_overlaps

f

ST_Overlaps returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).

ST_Overlaps returns true if two geometries overlap and the area of the overlap is greater than 0. The geometries must be two-dimensional (no Z or M values).

12.17.8 ST_Touches

ST_Touches — $\text{rastA} \text{ \& \#xc640; } \text{rastB}$ $\text{\& \#xc00; } \text{\& \#xcd5c; \& \#xc18c; \& \#xd5d5c; \& \#xc778; \& \#xd2b8; \& \#xd55c; \& \#xac1c; \& \#xb97c; \& \#xaf5; \& \#xc720; \& \#xd558; \& \#xc9c0; \& \#xb9cc; \& \#xb0b4; \& \#bd80; \& \#xad50; \& \#xcc28; \& \#xd558; \& \#xc9c0; \& \#b294; \& \#xc54a; \& \#xc744; \& \#xacbd; \& \#xc6b0; TRUE}$ $\text{\& \#xb97c; \& \#xbc18; \& \#xd658; \& \#xd5d5c;}$

Synopsis

```
boolean ST_Touches( raster rastA , integer nbandA , raster rastB , integer nbandB );
boolean ST_Touches( raster rastA , raster rastB );
```

설명

```
&#xb798;&#xc2a4;&#xd130; rastA&#xc640; &#xb798;&#xc2a4;&#xd130; rastB&#xac00; &#xacf5;&#xac04;&#xc801;&#xc73c;&#xc811;&#xcd09;&#xd560; &#xacbd;&#xc6b0; &#xcc38;&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. &#xb2e4;&#xb9d0;&#xd574; rastA&#xc640; &#xb798;&#xc2a4;&#xd130; rastB&#xac00; &#xcd5c;&#xc18c;&#xd55c; &#xd3ec;&#xc778;&#xd55c; &#xac1c;&#xb97c; &#xacf5;&#xc720;&#xd558;&#xc9c0;&#xb9cc; &#xb0b4;&#xbd80;&#xac00; &#xad50;&#xcc28;&#xc54a;&#xb294;&#xb2e4;&#xb294; &#xb73b;&#xc785;&#xb2c8;&#xb2e4;. &#xbc34;&#xb4dc; &#xbc88;&#xd638;&#xb97c; &#xc124;&#xc815;&#xd558;&#xc9c0; &#xc54a;&#xac70;&#xb098; &#xb610;&#xb294; NULL&#xb85c; &#xc124;&#xc815;&#xd558;&#xacbd;&#xc6b0; &#xd655;&#xc778; &#xc791;&#xc5c5;&#xc911; &#xb798;&#xc2a4;&#xd130;&#xc758; &#xbcf; &#xb85d; &#xae; &#xc9c8;&#xb9cc; &#xae0;&#xb824;&#xd569;&#xb2c8;&#xb2e4;. &#xbc34;&#xb4dc; &#xbc88;&#xd638;&#xb97c; &#xc124;&#xc815;&#xd560; &#xacbd;&#xc6b0; &#xd655;&#xc778; &#xc791;&#xc5c5;&#xc911; &#xac12;&#xc744; &#xac00; &#xc544;&#xb2cc;) &#xd53d;&#xc140;&#xb4e4;&#xb9cc; &#xae0;&#xb824;&#xd569;&#xb2c8;&#xb2e4;.
```


Note

```
&#xc774; &#xd568;&#xc218;&#xb294; &#xb798;&#xc2a4;&#xd130;&#xc5d0;&#xc11c; &#xc774;&#xc6a9;&#xd560;  
&#xc218;&#xb3c4; &#xc788;&#xb294; &#xbba8;&#xb4e0; &#xc778;&#xb371;&#xc2a4;&#xb97c;  
&#xd65c;&#xc6a9;&#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```


Note!

Note

```
&#xb798;&#xc2a4;&#xd130;&#xc640;          &#xb3c4;&#xd615;&#xc758;          &#xacf5;&#xac04;
&#xad00;&#xacc4;&#xc131;&#xc744;          &#xd655;&#xc778;&#xd558;&#xb824;&#xba74;,
ST_Touches(ST_Polygon(raster), geometry)&#xcc98;&#xb7fc; &#xb798;&#xc2a4;&#xd130;&#xc5d0; ST_Polygon
&#xd568;&#xc218;&#xb97c; &#xc774;&#xc6a9;&#xd558;&#xc2ed;&#xc2dc;&#xc624;.
```

2.1.0 ಄전부터 사용할 수 있습니다.

예시

```
SELECT r1.rid, r2.rid, ST_Touches(r1.rastr, 1, r2.rastr, 1) FROM dummy_rast r1 CROSS JOIN ↵
    dummy_rast r2 WHERE r1.rid = 2;
```

```

  rid |  rid | st_touches
-----+-----+-----
    2 |    1 | f
    2 |    2 | f

```

స고

ST Intersects

12.17.9 ST_SameAlignment

ST_SameAlignment — ޘ스터들이 동일한 기울기 축척 공간 참조 시스템 그리고 오&#x
가지고 있을 경우 (픽셀이 다른 픽&
침범하지 않고 동일한 그리드에
놓일 수 있을 경우) 참을 반환하고 &
그렇지 않을 경우 문제점을 설명&#x
안내문과 함께 거짓을 반환합니&#x

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 );

```

설명

종합 함수가 아닌 버전(변종 1, 2): (직&#xc
입력하거나 또는 좌상단, 축척, 기&
SRID 값을 이용해서) 두 래스터가 동&#xc
축척, 기울기, SRID를 가지며 한 래스&#xd
어떤 픽셀의 꼭짓점 4개 가운데 적&
어느 한 꼭짓점이 다른 래스터의
그리드의 어느 모서리에 놓일 경&#x
참을 반환합니다. 그렇지 않을 경&#x
정렬 문제를 설명하는 안내문(NOTICE)&
함께 거짓을 반환합니다.

```
&#xc885;&#xd569; &#xd568;&#xc218; &#xbc84;&#xc804;(&#xbcc0;&#xc885; 3): &#xb798;&#xc2a4;&#xd130; &#xc9d1;&#xd569; &#xc9d1;&#xd569; &#xb0b4;&#xbd80;&#xc758; &#xbaa8;&#xb4e0; &#xb798;&#xc2a4;&#xd130;&#xac00; &#xc815;&#xb82c;&#xc788;&#xc744; &#xacbd;&#xc6b0; &#xcc38;&#xc744; &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. ST_SameAlignment() &#xd568;&#xc218;&#xb294; PostgreSQL &#xc804;&#xbb38; &#xc6a9;&#xc5b4;&#xb85c; "&#xc885;&#xd569;(aggregate)" &#xd568;&#xc218;&#xc785;&#xb2c8;&#xb2e4;. &#xc989; SUM() &#xbc0f; AVG() &#xd568;&#xc218;&#xc640; &#xb9c8;&#xcc2 &#xb370;&#xc774;&#xd130; &#xd589;&#xb4e4;&#xc744; &#xb300;&#xc0c1;&#xc73c;&#xb85c; &#xc5f0;&#xc0b0;&#xd55c;&#xb73b;&#xc785;&#xb2c8;&#xb2e4;.
```

2.0.0 ಄전부터 사용할 수 있습니다.

```
&#xac1c;&#xc120;&#xc0ac;&#xd56d;: 2.1.0&#xbc84;&#xc804;&#xc5d0;&#xc11c;&#xc885;&#xd569;&#xd568;&#xc218;
&#xbcc0;&#xc885;&#xc774;&#xcd94;&#xac00;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시: 래스터

```
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
```



```

st_samealignment | st_notsamealignmentreason
-----+-----
f                | The rasters have different scales on the X axis
(1 row)

```

ST_SameAlignment

Section 11.1, [ST_SameAlignment](#)

12.17.11 ST_Within

ST_Within — Returns true if the first geometry is within the second geometry. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon.

Synopsis

boolean **ST_Within**(raster rastA , integer nbandA , raster rastB , integer nbandB);
 boolean **ST_Within**(raster rastA , raster rastB);

ST_Within

ST_Within — Returns true if the first geometry is within the second geometry. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon.



Note

ST_Within — Returns true if the first geometry is within the second geometry. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon.



Note

ST_Within — Returns true if the first geometry is within the second geometry. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon.



Note

ST_Within — Returns true if the first geometry is within the second geometry. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon.

2.1.0 **ST_Within** — Returns true if the first geometry is within the second geometry. The first geometry must be a point, line, or polygon. The second geometry must be a polygon. The first geometry must be a point, line, or polygon. The second geometry must be a polygon.

ST_DWithin

```
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

ST_DFullyWithin

ST_Within, ST_DFullyWithin

12.17.13 ST_DFullyWithin

ST_DFullyWithin — Returns true if the two input rasters, **rastA** and **rastB**, are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance.

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**);

ST_DFullyWithin

ST_DFullyWithin — Returns true if the two input rasters, **rastA** and **rastB**, are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance.



Note

ST_DFullyWithin — Returns true if the two input rasters, **rastA** and **rastB**, are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance.



Note

ST_DFullyWithin — Returns true if the two input rasters, **rastA** and **rastB**, are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance.

2.1.0 **ST_DFullyWithin** — Returns true if the two input rasters, **rastA** and **rastB**, are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance. The function returns true if the two input rasters are fully within each other within the specified distance.

ST_DFullyWithin

```
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

ST_Within, ST_DWithin

ST_Within, ST_DWithin

12.18 Raster Tips

12.18.1 Out-DB Rasters

12.18.1.1 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'
```

12.18.1.2 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

12.18.1.2.1 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
```

12.18.1.2.2 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
Max open files	1024	4096	files
Max locked memory	65536	65536	bytes
Max address space	unlimited	unlimited	bytes
Max file locks	unlimited	unlimited	locks
Max pending signals	15738	15738	signals
Max msgqueue size	819200	819200	bytes
Max nice priority	0	0	
Max realtime priority	0	0	
Max realtime timeout	unlimited	unlimited	us

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
```

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 11.1.1 for more details.

5. `사 용 자 데 이 터 베 이 스 에 어 떤 종 류 의 래 스 터 파 일 형 식 을 로 뒤 수 있 습 니 까 ?`

사 용 자 의 GDAL 라 이 브 러 리 가 지 원 모 든 형 식 을 로 드 할 수 있 습 니 다 GDAL 이 지 원 하 는 형 식 은 **GDAL File Formats** 에 잘 정 리 돼 있 습 니 다 사 용 자 가 설 치 한 특 정 GDAL 이 모 든 형 식 을 지 원 하 지 않 을 수 도 있 습 니 다 . 사 용 자 의 특 정 GDAL 이 지 원 하 는 형 확 인 하 려 면 , 다 음 명 령 어 를 실 형

```
raster2pgsql -G
```

- [illegible]

네 PostGIS raster has a function **ST_AsGDALRaster** that will allow you to use SQL to export to any raster format supported by your GDAL. You can get a list of these using the **ST_GDALDrivers** SQL function. You can also use GDAL commandline tools to export PostGIS raster to other formats. GDAL has a PostGIS raster driver, but is only compiled in if you choose to compile with PostgreSQL support. 사용자ఀ ไ정귌적ด록화된 래스터를 PostGIS 래스터데이터형으로 저장할 수 있음Å불구하고., 해당드라이버는 현Çไ정귌적으로 ด록화된 래스터지원하지않습니다., 사용자ఀ ఘ컴파일할 경우., 드라이버를 활Á사용자 설정에

```
--with-pg=path/to/pg_config
```

를 포 함 시 켜 야 합 니 다 . 여 러 OS 플 &#
대 해 GDAL 을 뻄 드 하 는 데 대 한 򾰩 뻉
알 고 싶 다 &#bba74; **GDAL Build Hints** 를 참 조 하 십 시
사 용 자 의 GDAL 򾲄 전 을 PostGIS 래 스 터 드 &
함 께 󌻴 파 일 했 다 &#bba74; 다 음 명 령 ౛
실 행 했 을 때 &#bbaa9; 록 에 PostGIS 래 스 터 ી
보 여 야 합 니 다 .

```
gdalinfo --formats
```

gdalinfo를 이용해서 GDAL을 통해 사용자 래스터에 대한 요약을 보려면 다음 명령어를 실행하십시오.

```
gdalinfo "PG:host=localhost port=5432 dbname='mygisdb' user='postgres' password=' ←
whatever' schema='someschema' table=sometable"
```

데이터를 다른 래스터 형식으&#xb8
내보내기하려면, gdal_translate를 이용하&#
다음 명령어는 테이블에서 모&#xb4
데이터를 10% 크기의 PNG 파일로 내&#xb
것입니다.사용자의 픽셀 밴드
유형에 따라.내보내기할 형식&#xc7
해당 픽셀 유형을 지원하지 않&#xc7
경우 gdal_translate이 작동하지 않을 수도
있습니다.예를 들어 부동소수&#xc8
밴드 유형 및 부호가 없는 32비트
정수형은 JPG 또는 몇몇 다른 형식&

```
&#xc27d;&#xac8c;&#xbcc0;&#xd658;&#xb418;&#xc9c0;&#xc54a;&#xc744;&#xac83;&#xc785;&#xb2c8;&#xb2e4;&#xb2e8;&#xc21c;&#xd55c; gdal_translate &#xc791;&#xc5c5;&#xc758;&#xc608;&#xc2dc;&#xc785;&#xb2c8;&#xb2e4;
```

```
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
```

사용자의 드라이버 연결 스트&#xb9c
WHERE=... 를 이용해서 내보내기 작업&#xc
SQL WHERE 절을 사용할 수도 있습니다.
다음은 WHERE 절을 이용하는 몇몇
예시입니다:

```
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
```

Reading Raster Data of PostGIS Raster section

7. 다운로드할수있는 GDAL 바이너리이򻯸 PostGIS 래스터지원과함૨컴파있습니까?

Ĥ. GDAL Binaries 페이지를확인해보십시페지원과함께컴파일된모든바이너리는 PostgreSQL 래스터도함께지원할것입니다. PostgreSQL 래스터는맄변화를겪고있는중입니다. 윈도최신 "Nightly" 빌드를다운로드하고싶다면. GDAL 트렁크. 파이썬바인딩및 MapServer 실행파일과 PostgreSQL 래스터드라내장한상태로 Visual Studio 를통해빌드일터마스세케레시(Tamás Szekeres)의 Nightly 빌드일확인해보십시오. 그냥 SDK 배치파클릭한다음사용자명령어를실행하십시오. <http://www.gisinternals.com> 에서다운수있으며. 또 VS 프로젝트파일도다운로드할수있습니다.

8. *어 떤 도 구 를 사 용 해 서 PostGIS 래 스 터 데 이 터 를 표 출 할 수 있 습 니 까 ?*

You can use **MapServer** compiled with GDAL to view Raster data. QGIS supports viewing of PostGIS Raster if you have PostGIS raster driver installed.

GDAL 쟄 용해데이터를 렌더링하는 어떤 도&#xad PostGIS 래스터 데이터를 지원하거&#xb09또는 최소한의 노력으로 지원&#xd5수 있습니다. 다시 말하지만 윈&#xb경우, 사용자가 직접 컴파일해&#xc11구성하는 수고를 들이고 싶지 않다면 터마스 세케레시의 <http://www.gisint>바이너리를 선택하는 편이 좋&#xc2

9. *어 떻 ಌ 내 MapServer 맵 에 PostGIS 래 스 터 레 &#xc7추 󊰀 할 수 있 습 니 까 ?*

㨼저 PostGIS 래스터 지원과 함૨ 컴파 GDAL 1.7 이상 버전이 필요합니다. GDAL 1.8 버전에서 많은 문제점들이 해때문에 1.8 이상 버전을 쓰는 편이 좋습니다. 트렁크 버전에서는 더 많은 PostGIS 래스터 문제점들이 해결됐습니다. 어떤 다른 래스터 마찬가지로 추가할 수 있습니다 MapServer 래스터 레이어와 함૨ 이용터 수 있는 여러 공간 처리 함수들이 목록을 보려면 MapServer Raster processing options 를 참조 래스터 데이터를 특히 흥미롭결 만드는 점은. 각 타일이 다양한 표준 데이터베이스 열들을 가지 수 있기 때문에 사용자 데이터서 타일을 부분으로 나눌 수 있다다 점입니다. 다음은 MapServer에서 PostGIS래스레이어를 어떻게 정의하는가에대한 예시입니다.

Note



```
&#xd0c0;&#xc77c;&#xd654;&#xb41c;      &#xb798;&#xc2a4;&#xd130;&#xc758;      &#xacbd;&#xc6b0;
mode=2      &#xc124;&#xc815;&#xc774;      &#xd544;&#xc694;&#xd55c;&#xb370;,      &#xc774;
&#xc124;&#xc815;&#xc740; PostGIS 2.0 &#xbc0f; GDAL 1.8 &#xb4dc;&#xb77c;&#xc774;&#xbc84;&#xc5d0;
&#xcd94;&#xac00;&#xb410;&#xc2b5;&#xb2c8;&#xb2e4;.      &#xc774;      &#xc124;&#xc815;&#xc740;      GDAL
1.7      &#xb4dc;&#xb77c;&#xc774;&#xbc84;&#xc5d0;&#xb294;      &#xc874;&#xc7ac;&#xd558;&#xc9c0;
&#xc54a;&#xc2b5;&#xb2c8;&#xb2e4;.
```

```
-- &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xd45c;&#xc900; &#xb798;&#xc2a4;&#xd130;  ↵
&#xc635;&#xc158;&#xc73c;&#xb85c; &#xd45c;&#xcd9c;&#xd558;&#xae30;
LAYER
    NAME coolwktraster
    TYPE raster
    STATUS ON
    DATA "PG:host=localhost port=5432 dbname='somedb' user='someuser' password=' ↵
        whatever'
        schema='someschema' table='coolttable' mode='2'"
    PROCESSING "NODATA=0"
    PROCESSING "SCALE=AUTO"
    #... &#xb2e4;&#xb978; &#xd45c;&#xc900; &#xb798;&#xc2a4;&#xd130;  ↵
        &#xacf5;&#xac04; &#xcc98;&#xb9ac; &#xd568;&#xc218;&#xb97c; &#xcd94;&#xac00;
    #... &#xd074;&#xb798;&#xc2a4;&#xb294;  ↵
        &#xc120;&#xd0dd;&#xc801;&#xc774;&#xc9c0;&#xb9cc; &#xb2e8;&#xc77c;  ↵
        &#xbc34;&#xb4dc; &#xb370;&#xc774;&#xd130;&#xc5d0; &#xc720;&#xc6a9;&#xd568;
    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

-- &#xb798;&#xc2a4;&#xd130;&#xb97c; &#xd45c;&#xc900; &#xb798;&#xc2a4;&#xd130;  ←
    &#xc635;&#xc158; &#xbc0f; WHERE &#xc808;&#xb85c; &#xd45c;&#xcd9c;&#xd558;&#xae30;
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

```

10. 현재어떤함수를내래스터데&#xc함께사용할수있습니까?

Chapter 12 ֹ 록 을 స 조 하 십 시 오 &#xc
더 많 이 있 지 만 목 록 화 를 계 속
진 행 중 입 니 다 향 후 어 떤 함 수 &#xb9
기 대 할 수 있 는 지 알 ૠ 싶 다 면
PostGIS 래 스 터 로 드 맵 페 이 지 를 స 조

11. *ERROR: function st_intersects(raster, unknown) is not unique or st_union(geometry,text) is not unique*

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.

```
SELECT rast
FROM my_raster
WHERE ST_Intersects(rast, 'SRID=4326;POINT(-10 10)');
```

사용자 코드를 다음과 같이 변&#xacbd
도형의 문자형 표현식을 도형&#xc7
형변환하십시오:

```
SELECT rast
FROM my_raster
WHERE ST_Intersects(rast, 'SRID=4326;POINT(-10 10)::geometry');
```

12. *PostGIS* 래스터와 오라클 *GeoRaster*의 *SDO_GEORASTER* 및 *SDO_RASTER*유형은 어떻게 다릅니까 ?이 주제에 대해 더 광범위한 논&#xc7보고 싶다면., 호르헤 아레발로(Jorge Arévalo)의 **Oracle GeoRaster and PostGIS Raster: First impressions** 블로그 포스&#참조하십시오.레이어 당 1개의 지리참조(one-georeference-by-layer)에 비해 래스터 당 1개의 지리참조(one-georeference-by-raster)쫀 지&#xb2c8주요한 장점은 다음과 같은 질&#xb9ac଩식이 쫀능하다는 것입니다.*

13. raster2pgsql이 대용량 파일을 로드할 때 String of N bytes is too long for encoding conversion 이 라 는 오 류 를 내 드 할 파 일 을 생 성 할 때 사 용 자 데 이 터 베 이 스 와 어 떤 연 결 도 하 지 않 습 니 다 . 사 용 자 데 이 터 베 이 스 가 사 용 자 데 이 터 인 코 딩 과 명 򼸱 히 다 른 클 라 이 어 인 코 딩 을 설 정 한 경 우 , 대 용 량 (30MB 이 상) 래 스 터 파 일 을 로 드 할 때 bytes is too long for encoding conversion 오 류 가 &#xbcl1; 생 하 수 도 있 습 니 다 . 예 를 들 어 사 용 자 데 이 터 베 이 스 의 인 코 딩 이 UTF8 인 윈 도 우 응 용 프 로 그 램 을 지 원 하 위 해 클 라 이 언 트 인 코 딩 을 WIN1252 으 로 설 정 한 경 우 , 일 &#xbcl8; 적 으 로 이 오 류 가 &#xbcl1; 생 합 니 다 . 이 오 류 류 피 하 려 면 로 드 작 업 중 사 용 자 데 이 터 베 이 스 의 인 코 딩 과 클 라 인 코 딩 이 동 일 한 지 확 인 하 십 십 사 용 자 의 로 드 스 크 립 트 에 확 십 설 정 해 두 &#ba74; 됩 니 다 . 다 음 은 윈 대 예 시 입 니 다 .

&#xc720:&#xb2c9:&#xc2a4:/&#xb9ac:&#xb205:&#xc2a4:&#xc758: &#xachd:&#xc6b0::


```
export PGCLIENTENCODING=UTF8
```

<http://trac.osgeo.org/postgis/ticket/2209>

14. *ST_FromGDALRaster*: Could not open bytea with GDAL. Check that the bytea is of a GDAL supported format. *ST_AsPNG*: Could not load the output GDAL driver
- PostGIS 2.1.3, 2.0.5, GDAL, DB, Section 2.1

정하기 위한 미시적(micro) 요소를
탐색합니다. 현재 국가 코드 또는
국가명을 탐색하지는 않지만 향&#xc
추가될 수도 있습니다.

국가 코드 미국 또는 ಐ나다의 주/
미국 또는 ಐ나다의 우편번호&#xb9
바탕으로 미국 또는 ಐ나다로
가정합니다.

우편번호/집코드(zip code) 펄(Perl) 호환 정&#xc
표현식을 이용해서 우편번호&#xb97
인식합니다. 이 정귝 표현식은
현재 parseaddress-api.c 파일에 담겨 있고 필&#xc69
경우 상대적으로 쉽게 변경할
수 있습니다.

주/도 펄(Perl) 호환 정귝 표현식을 이&#xc
우편번호를 인식합니다. 이 정&#xad
표현식은 현재 parseaddress-api.c 파일에 담&#xaca8
있는데 향후 유지보수를 더 쉽&#xac
하기 위해 "includes" 로 이동할 수도 있&#xc

14.1.2 주소 표준화 도구 유형

14.1.2.1 stdaddr

stdaddr — 주소의 요소들로 이루어진 합&#xc
유형입니다. `standardize_address` 함수가 이 유&#xc
반환합니다.

설명

주소의 요소들로 이루어진 합성
유형입니다. `standardize_address` 함수가 이 유형&
반환합니다. `PAGC Postal Attributes` 에서 요소들&#xc5d0
대한 몇몇 설명을 빌려왔습니다.

귝ౙ 테이블 에서 출력 참조 번호&#xc
표시하는 토큰 번호를 찾아௏ 수
있습니다.



This method needs `address_standardizer` extension.

building 문자형(토큰 번호 0)입니다: 건&#xbb3c
번호 또는 건물명을 참조합니&#xb2
파싱되지 않은 건물 식별자 및
유형입니다. 주소 대부분의 경&#xc6
일반적으로 비어 있습니다.

house_num 문자형(토큰 번호 1)입니다: 도&#xb
번지수입니다. 예: 75 State Street 의 75 번지

predir 문자형(토큰 번호 2)입니다: North, South,
East, West 등과 같이 방향을 나타내는
도로명 접두사(STREET NAME PRE-DIRECTIONAL)입니&#xb2

qual 문자형(토큰 번호 3)입니다: 도로전치수식어(STREET NAME PRE-MODIFIER)입니다: 예: 3715 OLD HIGHWAY 99 에서 *OLD*

pretype 문자형(토큰 번호 4)입니다: 도로접두사 유형(STREET PREFIX TYPE)입니다.

name 문자형(토큰 번호 5)입니다: 도로입니다: NAME)입니다.

suftype 문자형(토큰 번호 6)입니다: St, Ave, Cir와ఙ은 도로 접򻯸사 유형(STREET POST TYPE)입도로명의 몸통 뒤에 붙는 도로 유형을 뜻합니다: 예: 75 State Street 에서 *STREET*

sufdir 문자형(토큰 번호 7)입니다: 방향나타내는 도로 접򻯸사(STREET POST-DIRECTIONAL)도로명 뒤에 붙어 방향을 나타&#xb0c218;식어입니다: 예: 3715 TENTH AVENUE WEST 에서 *WEST*

ruralroute is text (token number 8): RURAL ROUTE . Example 7 in RR 7.

extra 문자형입니다: 층수 ఙ은 추ఀ 정보입니다.

city 문자형(토큰 번호 10)입니다: 예: 보시

state 문자형(토큰 번호 11)입니다: 예: 매주

country 문자형(토큰 번호 12)입니다: 예: USA

postcode 문자형(토큰 번호 13) 우편번호(postal code, zip code)입니다: 예: 02109

box 문자형(토큰 번호 14, 15) 사서함 번호 BOX NUMBER)입니다: 예: 02109

unit 문자형(토큰 번호 17) 아파트 동수 또는 호수입니다: 예: APT 3B 에서 3B

14.1.3 주소 표준화 도구 테이블

14.1.3.1 규칙 테이블

규칙 테이블 — 규칙 테이블은 주소 입력 배열 토큰을 표준화된 출력 배열 토큰에 매핑하는 일련의 규담󊳠 있습니다: 입력 토큰들의 집그 뒤에 -1(종결자: terminator), 그 뒤에 출력 토큰들의 집합: 그 뒤에 -1, 그 뒤에 규칙의 종류를 표시하는 번호: 그 뒤에 규칙의 순위를 나열해서 하규칙을 정의합니다.

ORD (18). 단일 문자입니다.

WORD (1). 임의의 길이의 문자열입니다.
단일 문자는 SINGLE도., WORD도 될 수 있슲

기능 기반 입력 토큰

BOXH (14). 우편사서함을 표시하는 데
쓰이는 단어입니다.. 예를 들어
Box 또는 *PO Box* ಘ럼 말이죠.

BUILDH (19). 일반적으로 접두사의 형태이
󊱴물 또는 󊱴물군을 표시하는
데 쓰이는 단어입니다.. 예: *Tower 7A* 에있
Tower

BUILD 일반적으로 접뮸사의 형태인,
󊱴물 또는 󊱴물군을 표시하는
데 쓰이는 단어 및 약어입니다.
예: *Shopping Centre*

DIRECT (22). ଩향을 표시하는 데 쓰이는
단어입니다.. 예: *North*

MILE (20). 󊱰리표(어떤 지점까지의 󊱰리나
타내는 도로󊰀의 푯말: milepost) 주서
표시하는 데 쓰이는 단어입니다.

ROAD (6). ૠ속도로와 도로를 표시하는
데 쓰이는 단어 및 약어입니다.
예: *Interstate 5* 에서 *Interstate*

RR (8). 지଩ 우편물 ର달 노선(rural route)을 표데
쓰이는 단어 및 약어입니다.
RR.

TYPE (2). 도로 유형을 표시하는 데 쓰이단
어 및 약어입니다.. 예: *ST* 또는 *AVE*

UNITH (16). 내부 하위주소를 표시하는
데 쓰이는 단어 및 약어입니다.
예: *APT* 또는 *UNIT*

우편 유형 입력 토큰

QUINT (28). 다섯 자리 숫자입니다.. 집코듐
Code)를 식별합니다.

QUAD (29). 네 자리 숫자입니다.. ZIP4를 식별듐

PCH (27). 문자., 숫자., 문자 순서로 이루어
문자 3󊰜의 ର열입니다.. ಐ나다
우편 번호의 ಘ음 세 문자인 FSA를
식별합니다.

PCT (26). 숫자., 문자., 숫자 순서로 이루어
문자 3󊰜의 ର열입니다.. ಐ나다
우편 번호의 마지막 세 문자인
LDU를 식별합니다.

HOUSE 문자형(토큰 번호 1)입니다: 도로번지수입니다. 예: 75 State Street 의 75 번지

predir 문자형(토큰 번호 2)입니다: North, South, East, West 등과 같이 방향을 나타내는 도로명 접두사 (STREET NAME PRE-DIRECTIONAL)입니다

qual 문자형(토큰 번호 3)입니다: 도로전치수식어 (STREET NAME PRE-MODIFIER)입니다. 예: 3715 OLD HIGHWAY 99 에서 *OLD*

pretype 문자형(토큰 번호 4)입니다: 도로접두사 유형 (STREET PREFIX TYPE)입니다.

street 문자형(토큰 번호 5)입니다: 도로사 (STREET NAME)입니다.

suftype 문자형(토큰 번호 6)입니다: St, Ave, Cir와같은 도로 접򻯸사 유형 (STREET POST TYPE)입도로명 의 몸통 뒤에 붙는 도로 유형 을 뜻합니다. 예: 75 State Street 에서 *STREET*

sufdir 문자형(토큰 번호 7)입니다: 니향나타내는 도로 접򻯸사 (STREET POST-DIRECTIONAL)도로명 뒤에 붙어 니향을 나타내수식어입니다. 예: 3715 TENTH AVENUE WEST 에서 *WEST*

ARC_C

(토큰 번호 = "2"). HOUSE 속성을 제외하ૠ MICRO 절을 파싱하기 위한 귞칙 클래따라서 HOUSE 토큰을 뺀 MICRO_C 출력 토큰 집합을 이용합니다.

CIVIC_C

(토큰 번호 = "3"). HOUSE 속성을 파싱하기 위한 귞칙 클래스입니다.

EXTRA_C

(토큰 번호 = "4"). EXTRA 속성 - 지오코딩에제외된 속성 - 을 파싱하기 위한 귞클래스입니다. 빌드 시기에는 이 귞칙들을 이용하지 않습니다.

EXTRA_C 출력 토큰 (<http://www.pagegeo.org/docs/html/page-12.html#-r-typ--> 에서 발췌).

BLDNG (토큰 번호 0): 파싱되지 않은 건문 식별자 및 유형입니다.

BOXH (token number 14): The **BOX** in BOX 3B

BOXT (토큰 번호 15): BOX 3B 에서 **3B**

RR (토큰 번호 8): RR 7 에서 **RR**

UNITH (토큰 번호 16): APT 3B 에서 **APT**

UNITT (토큰 번호 17): APT 3B 에서 **3B**

UNKNWN (토큰 번호 9): 따로 분류되지 않사 출력물입니다.

14.1.3.2

— (lex) (the section called “”) (1) (the section called “”) (2) .

(lexicon) (1) (the section called “”) (2) .

ONE stdword 1 (2) .

id (3) .

seq (4) .

word (5) .

stdword (6) .

token (7) .

14.1.3.3

— (gaz) (the section called “”) (1) (the section called “”) (2) .

A gaz (short for gazeteer) table is used to standardize place names and associate that input with the section called “” and (b) standardized representations. For example if you are in US, you may load these with State Names and associated abbreviations.

(3) .

id (3) .

seq (4) .

word ସ자&#xd615,: 입력 단어

stdword 문자형: 표준화된 대체 단어

token 정수형: 단어의 유형을 나타냅단어가 해당 맥락에서 쓰였을 경우에만 대운될 것입니다. **PAGC Tokens** 를 참조하십시오.

14.1.4 **14.1.4**

14.1.4.1 parse_address

```
parse_address — %x55c; %x904; %xb85c; %xb41c; %xc8fc; %xc18c; %xb97c; %xc785; %xb825; %xbc1b; %xc544;
%xbd80; %xbd84; %xb4e4; %xb85c; %xbd84; %xd574; %xd569; %xb2c8; %xb2e4;.
```

Synopsis

```
record parse_address(text address);
```

설명

Returns takes an address as input, and returns a record output consisting of fields *num*, *street*, *street2*, *address1*, *city*, *state*, *zip*, *zipplus*, *country*.

2.2.0 버전부터 사용할 수 있습니다.



This method needs `address_standardizer` extension.

예시

단일 주소

```
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

주소 테이블

```
-- &#xae30;&#xbc&#xf8;&#xd14c;&#xc774;&#xbe14;
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');
```

```
-- &#xc8fc;&#xc18c; &#xd30c;&#xc2f1;&#xd558;&#xae30;
-- &#xbaa8;&#xb4e0; &#xd56d;&#xbaa9;&#xc744; &#xc6d0;&#xd55c;&#xb2e4;&#xba74; (a).* &#xb97c; &#xc4f0;&#xba74; &#xb429;&#xb2c8;&#xb2e4;.
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)

standardize_address

14.1.4.2 standardize_address

standardize_address — Standardize an address string. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value.

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);

standardize_address

standardize_address — Standardize an address string. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value.

standardize_address — Standardize an address string. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value.



This method needs address_standardizer extension.

address_standardizer_data_us

address_standardizer_data_us — Standardize an address string. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value.

```
CREATE EXTENSION address_standardizer_data_us; -- Standardize an address string. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value.
```

address_standardizer_data_us — Standardize an address string. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value. The input string is a text value, and the output is a text value.

```
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

TIGER 지오코딩 도구로 패키징된 테&#xc7
이용 (이예시는 사용자가 postgis_tiger_geocoder
를 설치했을 경우에만 동작합니&#xb

```
SELECT * FROM standardize_address('tiger.pagc_lex',
    'tiger.pagc_gaz', 'tiger.pagc_rules', 'One Devonshire Place, PH 301, Boston, MA ←
    02109-1234');
```

```
&#xb354;&#xc54c;&#xc544;&#xbcf4;&#xae30;&#xc27d;&#xac8c;&#xd558;&#xae30;&#xc704;&#xd574; hstore &#xd655;&#xc7
&#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc744;&#xc774;&#xc6a9;&#xd574;&#xc11c;&#xcd9c;&#xb825;&#xbb3c;&#xc744;
&#xb364;&#xd504;&#xd560;&#xac83;&#xc785;&#xb2c8;&#xb2e4;. &#xc0ac;&#xc6a9;&#xc790;&#xac00; CREATE EXTENSION
hstore; &#xba85;&#xb839;&#xc73c;&#xb85c;&#xc124;&#xce58;&#xd574;&#xc57c;&#xd569;&#xb2c8;&#xb2e4;.
```

```
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)		

변종 2: 주소를 두 부분으로 입력받&

```
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)

```

󌰸고

stdaddr, 규칙 테이블, 어휘 목록 테이블, 지명 색인 테이블, Page_Normalize_Address

14.2 TIGER 지오코딩 도구

TIGER 지오코딩 도구와 달리 여러 국가 지오코딩 을 지원한다는 장점을 가진, PostGIS를 지원하는 몇몇 오픈소지오코딩 도구들이 있습니다.

- **Nominatim** 은 OpenStreetMap 지명 색인 데이터 형식이용합니다. 데이터를 로드하기 위해 osm2pgsql이, 제대로 동작하기 위해 PostgreSQL 8.4 이상 버전 및 PostGIS 1.5 이상 버전이 필요합니다. 웹서비스 인터페이패키징됐으며, 웹서비스로 호출위해 설계된 것으로 보입니다. TIGER 지오코딩 도구와 마బ가지로, 지도구 및 역 지오코딩 도구 구성 요모두 가지고 있습니다. 문서상에 보면, Nominatim이 TIGER 지오코딩 도구ಘ럼 순수한 SQL 인터페이스를 가지고 있는지, 또는 웹 인터페이스에 충 로직을 구현했는지 불확실합니다.
- **GIS Graphy** 또한 PostGIS를 활용하며 Nominatimಘ럼 OSM(OpenStreetMap) 데이터를 이용합니다. OSM 데로드하기 위한 로더를 갖추고 있와 유사하게 미국 이외의 국가 지오코딩도 할 수 있습니다. Nominatim과 마బ가지로, 웹서비스로 실행되을 Java 1.5, Servlet apps, Solr에 의존합니다. GIS Graphy는 여러 플랫폼 상에서 실행되며, 몇몇 멋기능과 함께 역 지오코딩 도구도 가지고 있습니다.

14.2.1 Drop_Indexes_Generate_Script

Drop_Indexes_Generate_Script — TIGER 스키마 및 사용자 지정 스키마에 있는 기본 키가 아닌 그유일하지 않은 인덱스들을 모두 삭제하는 스크립트를 생성합니스키마를 따로 지정하지 않을 경기본값인 tiger_data 스키마를 대상으로 합니다.

Synopsis

text **Drop_Indexes_Generate_Script**(text param_schema=tiger_data);

설명

TIGER 스키마 및 사용자 지침서
 있는 기򼿈 키가 아닌 그리고 유일않은 인덱스들을 모두 삭제하는
 스크립트를 생성합니다. 스키마따로 지정하지 않을 경우 기򼿈값인는
 tiger_data 스키마를 대상으로 합니다.

이 함수는 쿼리 설계자를 혼란스할 수도 있는 인덱스의 쓸데없는
 팽창(bloat) 또는 불필요한 용량을 차일을 최소화합니다. **Install_Missing_Indexes** 함수와결합해서 지오코딩 도구가 이용인덱스들만 추가하는 데 쓰입니다.

2.0.0 버전부터 사용할 수 있습니다.

예시

```
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;
:
:
```

Drop_Nation_Tables_Generate_Script

[Install_Missing_Indexes, Missing_Indexes_Generate_Script](#)

14.2.2 Drop_Nation_Tables_Generate_Script

Drop_Nation_Tables_Generate_Script — Drop nation tables for a given schema. The schema is specified by the `text param_schema` parameter. The tables to be dropped are: `county_all`, `state_all`, `county`, `state`, `ma_county`, and `ma_state`.

Synopsis

```
text Drop_Nation_Tables_Generate_Script(text param_schema=tiger_data);
```

Drop_Nation_Tables_Generate_Script

Drop nation tables for a given schema. The schema is specified by the `text param_schema` parameter. The tables to be dropped are: `county_all`, `state_all`, `county`, `state`, `ma_county`, and `ma_state`. The tables are dropped in the following order: `county_all`, `state_all`, `county`, `state`, `ma_county`, and `ma_state`.

2.1.0 Drop_Nation_Tables_Generate_Script(text param_schema=tiger_data);

Drop_Nation_Tables_Generate_Script

```
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;
```

Drop_Nation_Tables_Generate_Script

[Loader_Generate_Nation_Script](#)

14.2.3 Drop_State_Tables_Generate_Script

Drop_State_Tables_Generate_Script — Drop state tables for a given schema. The schema is specified by the `text param_schema` parameter. The tables to be dropped are: `county_all`, `state_all`, `county`, `state`, `ma_county`, and `ma_state`.

Synopsis

```
text Drop_State_Tables_Generate_Script(text param_state, text param_schema=tiger_data);
```


tiger_data.pa_addr;

```

tiger_data.pa_county;
tiger_data.pa_county_lookup;
tiger_data.pa_cousub;
tiger_data.pa_edges;
tiger_data.pa_faces;
tiger_data.pa_featnames;
tiger_data.pa_place;
tiger_data.pa_state;
tiger_data.pa_zip_lookup_base;
tiger_data.pa_zip_state;
tiger_data.pa_zip_state_loc;

```

2.0.0; tiger_data.pa_addr; tiger_data.pa_county; tiger_data.pa_county_lookup; tiger_data.pa_cousub; tiger_data.pa_edges; tiger_data.pa_faces; tiger_data.pa_featnames; tiger_data.pa_place; tiger_data.pa_state; tiger_data.pa_zip_lookup_base; tiger_data.pa_zip_state; tiger_data.pa_zip_state_loc;

tiger_data.pa_addr;

```

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;

```

Loader_Generate_Script**14.2.4 Geocode**

Geocode — tiger_data.pa_addr; tiger_data.pa_county; tiger_data.pa_county_lookup; tiger_data.pa_cousub; tiger_data.pa_edges; tiger_data.pa_faces; tiger_data.pa_featnames; tiger_data.pa_place; tiger_data.pa_state; tiger_data.pa_zip_lookup_base; tiger_data.pa_zip_state; tiger_data.pa_zip_state_loc;

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);

설명

주소를 스트링(또는 다른 정규화주소)으로 입력벱아 NAD83 경위도로 투영된 포인트 도형,각 주소에 대있는 위엨들의 집합을 출력합니순위가 낮을수록 주소와 위엨가 일엨할 가능성이 쳤집니다.결과순위가 낮은 순서로 정렬합니다.위이터(edge, face, addr), PostgreSQL 퍼지 ସ자열 매자levenshtein), 그리고 PostGIS 라인 보간 함수를 이자TIGER 경계선을 따라 주소를 보간합니순위가 높을수록 지오코딩 작업올벴를 가능성이 떨어집니다.지포인트는 기본적으로 중심선에도로 주소가 위엨한 경(왼쪽/오른ఐ미터 오프셋됩니다.

개선 사항: 2.0.0 򼮄전에서 TIGER 2010 데이터 구조를 지원하고,속도와 지오코작업의 정확도를 향상시키기 위இ몇 로직을 수정하고,포인트를 중심선에서 도로 주소가 위엨한 쪽으로 오프셋시키기 시작했습새로 도입된 max_results 파라 미터는 결개수를 설정하거나 최선의 결과벸환하도록 하는 데 유용합니다.

기본 예시

매사추세ఠ(MA), 미네소타(MN), ಘ리포로드아일랜드(RI) 주의 TIGER 데이터를 로드한 PostgreSQL 9.1rc1/PostGIS 2.0이 실행되고 있는 3.0 GHZ 단일 프로세서 򼼏 2GB 램 윈도우7 췴퓨터에서 다음 예시를 실행시데 걸리는 시간을 డ정했습니다.

완전 일엨를 계산하는 것이 더 빠자

```
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
```

지오코딩 도구에 집코드를 넘겨않더라도 추డ할 수 있습니다(122 ~ 150벰리초 걸립니다).

```
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	wktlonlat	stno	street	styp	city	st	zip
1	POINT(-71.05528 42.36316)	226	Hanover	St	Boston	MA	02113

```

&#xc624;&#xb958;&#xb3c4; &#xc98;&#xb9ac;&#xd560; &#xc218; &#xc788;&#xb294;&#xb370;, &#xd558;&
&#xc774;&#xc0c1;&#xc758; &#xac00;&#xb2a5;&#xc131; &#xc788;&#xb294; &#xb2f5;&#xc744; &#xc21c;&#xc704;&#xc640;
&#xd568;&#xae8;&#xc0dd;&#xc131;&#xd558;&#xae30; &#xb54c;&#xbb38;&#xc5d0; &#xc2dc;&#xac04;&#xc774; &#xb354;
&#xac78;&#xb9bd;&#xb2c8;&#xb2e4;(500&#xbc00;&#xb9ac;&#xcd08;).

```

```

SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktlonlat,
    (addy).address As stno, (addy).streetname As street,
    (addy).streettypeabbrev As styp, (addy).location As city, (addy).stateabbrev As st, (
    addy).zip
FROM geocode('31 - 37 Stewart Street, Boston, MA 02116') As g;
rating | wktlonlat | stno | street | styp | city | st | zip
-----+-----+-----+-----+-----+-----+-----+-----
70 | POINT(-71.06459 42.35113) | 31 | Stuart | St | Boston | MA | 02116

```

```

&#xc8fc;&#xc18c;&#xb4e4;&#xc744; &#xc9c0;&#xc624;&#xcf54;&#xb529;&#xd558;&#xb294; &#xb370; &#xbc30;&#xce58;(bat
&#xc791;&#xc5c5;&#xc744; &#xc774;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;. max_results = 1&#xb85c; &#xc124;&#xc815;
&#xd3b8;&#xc774; &#xac00;&#xc7a5; &#xc27d;&#xc2b5;&#xb2c8;&#xb2e4;. &#xc544;&#xc9c1; &#xc9c0;&#xc624;&#xcf54;&#
&#xc54a;&#xc740; (&#xc21c;&#xc704;&#xac00; &#xc5c6;&#xb294;) &#xc8fc;&#xc18c;&#xb4e4;&#xb9cc; &#xc98;&#xb9ac;&#

```

```

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');

```

```

-- &#xc98;&#xc74c; &#xc138; &#xc8fc;&#xc18c;&#xb9cc; &#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xd569;&#xb2c8;&#xb2e4; (323 ~ 704 &#
&#xbc00;&#xb9ac;&#xcd08; - &#xce90;&#xc2f1; &#xbc0f; &#xacf5;&#xc720; &#xba54;&#xbaa8;&#xb9ac; &#xc601;&#xd5a5; &#xb54c;&#xbb38;&#xc5d0; &#xccab; &#xc9c0;&#xc624;&#xcf54;&#xb529; &#xc791;&#xc5c5;&#xc740; &#xd56d;&#xc0c1; &#xb290;&#xb9bd;&#xb2c8;&#xb2e4;). --
-- &#xc8fc;&#xc18c;&#xac00; &#xb300;&#xb7c9;&#xc77c; &#xacbd;&#xc6b0; &#xd55c;&#xbc88;&#xc5d0; &#xc5c5;&#xb370;&#xc774;&#xd2b8;&#xd558;&#xc9c0; &#xc54a;&#xb294; &#xd3b8;&#xc774; &#xc88b;&#xc2b5;&#xb2c8;&#xb2e4;.
-- &#xc804;&#xccb4; &#xc9c0;&#xc624;&#xcf54;&#xb529; &#xc791;&#xc5c5;&#xc744; &#xd55c;&#xbc88;&#xc5d0; &#xcee4;&#xbc0b;&#xd574;&#xc57c;&#xb9cc; &#xd558;&#xae30; &#xb54c;&#xbb38;&#xc785;&#xb2c8;&#xb2e4;.
-- &#xc774; &#xc608;&#xc2dc;&#xc758; &#xacbd;&#xc6b0; LEFT JOIN&#xc73c;&#xb85c; &#xc7ac;&#xacb0;&#xd569;&#xc2dc;&#xd0a4;&#xace0;
-- &#xb098;&#xc05c; &#xc8fc;&#xc18c;&#xb97c; &#xb2e4;&#xc2dc; &#xc9c0;&#xc624;&#xcf54;&#xb529; &#xc791;&#xc5c5;&#xd558;&#xc9c0; &#xc54a;&#xae30; &#xc704;&#xd574;
-- &#xc77c;&#xce58;&#xd558;&#xc9c0; &#xc54a;&#xc744; &#xacbd;&#xc6b0; &#xc21c;&#xc704;&#xb97c; -1&#xb85c; &#xc124;&#xc815;&#xd569;&#xb2c8;&#xb2e4;.
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 new_address	lon	lat	
1	529 Main Street, Boston MA, 02129 Boston, MA 02129	-71.07181	42.38359	529 Main St, ↩
2	77 Massachusetts Avenue, Cambridge, MA 02139 Massachusetts Ave, Cambridge, MA 02139	-71.09428	42.35988	77 ↩
3	25 Wizard of Oz, Walaford, KS 99912323	-1		↩

Geocode Intersection

```
SELECT g.rating, ST_AsText(ST_SnapToGrid(g.geomout,0.00001)) As wktnlonlat,
      (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	wktnlonlat	stno	street	styp	city	st	zip
8	POINT(-70.96796 42.4659)	100	Federal	St	Lynn	MA	01905

Total query runtime: 245 ms.

Geocode Intersection

[Normalize_Address](#), [Pprint_Addy](#), [ST_AsText](#), [ST_SnapToGrid](#), [ST_X](#), [ST_Y](#)

14.2.5 Geocode_Intersection

Geocode_Intersection — [Geocode_Intersection](#) (PostgreSQL 9.5.1, 9.6.1, 9.6.2, 9.6.3, 9.6.4, 9.6.5, 9.6.6, 9.6.7, 9.6.8, 9.6.9, 9.6.10, 9.6.11, 9.6.12, 9.6.13, 9.6.14, 9.6.15, 9.6.16, 9.6.17, 9.6.18, 9.6.19, 9.6.20, 9.6.21, 9.6.22, 9.6.23, 9.6.24, 9.6.25, 9.6.26, 9.6.27, 9.6.28, 9.6.29, 9.6.30, 9.6.31, 9.6.32, 9.6.33, 9.6.34, 9.6.35, 9.6.36, 9.6.37, 9.6.38, 9.6.39, 9.6.40, 9.6.41, 9.6.42, 9.6.43, 9.6.44, 9.6.45, 9.6.46, 9.6.47, 9.6.48, 9.6.49, 9.6.50, 9.6.51, 9.6.52, 9.6.53, 9.6.54, 9.6.55, 9.6.56, 9.6.57, 9.6.58, 9.6.59, 9.6.60, 9.6.61, 9.6.62, 9.6.63, 9.6.64, 9.6.65, 9.6.66, 9.6.67, 9.6.68, 9.6.69, 9.6.70, 9.6.71, 9.6.72, 9.6.73, 9.6.74, 9.6.75, 9.6.76, 9.6.77, 9.6.78, 9.6.79, 9.6.80, 9.6.81, 9.6.82, 9.6.83, 9.6.84, 9.6.85, 9.6.86, 9.6.87, 9.6.88, 9.6.89, 9.6.90, 9.6.91, 9.6.92, 9.6.93, 9.6.94, 9.6.95, 9.6.96, 9.6.97, 9.6.98, 9.6.99, 9.6.100, 9.6.101, 9.6.102, 9.6.103, 9.6.104, 9.6.105, 9.6.106, 9.6.107, 9.6.108, 9.6.109, 9.6.110, 9.6.111, 9.6.112, 9.6.113, 9.6.114, 9.6.115, 9.6.116, 9.6.117, 9.6.118, 9.6.119, 9.6.120, 9.6.121, 9.6.122, 9.6.123, 9.6.124, 9.6.125, 9.6.126, 9.6.127, 9.6.128, 9.6.129, 9.6.130, 9.6.131, 9.6.132, 9.6.133, 9.6.134, 9.6.135, 9.6.136, 9.6.137, 9.6.138, 9.6.139, 9.6.140, 9.6.141, 9.6.142, 9.6.143, 9.6.144, 9.6.145, 9.6.146, 9.6.147, 9.6.148, 9.6.149, 9.6.150, 9.6.151, 9.6.152, 9.6.153, 9.6.154, 9.6.155, 9.6.156, 9.6.157, 9.6.158, 9.6.159, 9.6.160, 9.6.161, 9.6.162, 9.6.163, 9.6.164, 9.6.165, 9.6.166, 9.6.167, 9.6.168, 9.6.169, 9.6.170, 9.6.171, 9.6.172, 9.6.173, 9.6.174, 9.6.175, 9.6.176, 9.6.177, 9.6.178, 9.6.179, 9.6.180, 9.6.181, 9.6.182, 9.6.183, 9.6.184, 9.6.185, 9.6.186, 9.6.187, 9.6.188, 9.6.189, 9.6.190, 9.6.191, 9.6.192, 9.6.193, 9.6.194, 9.6.195, 9.6.196, 9.6.197, 9.6.198, 9.6.199, 9.6.200, 9.6.201, 9.6.202, 9.6.203, 9.6.204, 9.6.205, 9.6.206, 9.6.207, 9.6.208, 9.6.209, 9.6.210, 9.6.211, 9.6.212, 9.6.213, 9.6.214, 9.6.215, 9.6.216, 9.6.217, 9.6.218, 9.6.219, 9.6.220, 9.6.221, 9.6.222, 9.6.223, 9.6.224, 9.6.225, 9.6.226, 9.6.227, 9.6.228, 9.6.229, 9.6.230, 9.6.231, 9.6.232, 9.6.233, 9.6.234, 9.6.235, 9.6.236, 9.6.237, 9.6.238, 9.6.239, 9.6.240, 9.6.241, 9.6.242, 9.6.243, 9.6.244, 9.6.245, 9.6.246, 9.6.247, 9.6.248, 9.6.249, 9.6.250, 9.6.251, 9.6.252, 9.6.253, 9.6.254, 9.6.255, 9.6.256, 9.6.257, 9.6.258, 9.6.259, 9.6.260, 9.6.261, 9.6.262, 9.6.263, 9.6.264, 9.6.265, 9.6.266, 9.6.267, 9.6.268, 9.6.269, 9.6.270, 9.6.271, 9.6.272, 9.6.273, 9.6.274, 9.6.275, 9.6.276, 9.6.277, 9.6.278, 9.6.279, 9.6.280, 9.6.281, 9.6.282, 9.6.283, 9.6.284, 9.6.285, 9.6.286, 9.6.287, 9.6.288, 9.6.289, 9.6.290, 9.6.291, 9.6.292, 9.6.293, 9.6.294, 9.6.295, 9.6.296, 9.6.297, 9.6.298, 9.6.299, 9.6.300, 9.6.301, 9.6.302, 9.6.303, 9.6.304, 9.6.305, 9.6.306, 9.6.307, 9.6.308, 9.6.309, 9.6.310, 9.6.311, 9.6.312, 9.6.313, 9.6.314, 9.6.315, 9.6.316, 9.6.317, 9.6.318, 9.6.319, 9.6.320, 9.6.321, 9.6.322, 9.6.323, 9.6.324, 9.6.325, 9.6.326, 9.6.327, 9.6.328, 9.6.329, 9.6.330, 9.6.331, 9.6.332, 9.6.333, 9.6.334, 9.6.335, 9.6.336, 9.6.337, 9.6.338, 9.6.339, 9.6.340, 9.6.341, 9.6.342, 9.6.343, 9.6.344, 9.6.345, 9.6.346, 9.6.347, 9.6.348, 9.6.349, 9.6.350, 9.6.351, 9.6.352, 9.6.353, 9.6.354, 9.6.355, 9.6.356, 9.6.357, 9.6.358, 9.6.359, 9.6.360, 9.6.361, 9.6.362, 9.6.363, 9.6.364, 9.6.365, 9.6.366, 9.6.367, 9.6.368, 9.6.369, 9.6.370, 9.6.371, 9.6.372, 9.6.373, 9.6.374, 9.6.375, 9.6.376, 9.6.377, 9.6.378, 9.6.379, 9.6.380, 9.6.381, 9.6.382, 9.6.383, 9.6.384, 9.6.385, 9.6.386, 9.6.387, 9.6.388, 9.6.389, 9.6.390, 9.6.391, 9.6.392, 9.6.393, 9.6.394, 9.6.395, 9.6.396, 9.6.397, 9.6.398, 9.6.399, 9.6.400, 9.6.401, 9.6.402, 9.6.403, 9.6.404, 9.6.405, 9.6.406, 9.6.407, 9.6.408, 9.6.409, 9.6.410, 9.6.411, 9.6.412, 9.6.413, 9.6.414, 9.6.415, 9.6.416, 9.6.417, 9.6.418, 9.6.419, 9.6.420, 9.6.421, 9.6.422, 9.6.423, 9.6.424, 9.6.425, 9.6.426, 9.6.427, 9.6.428, 9.6.429, 9.6.430, 9.6.431, 9.6.432, 9.6.433, 9.6.434, 9.6.435, 9.6.436, 9.6.437, 9.6.438, 9.6.439, 9.6.440, 9.6.441, 9.6.442, 9.6.443, 9.6.444, 9.6.445, 9.6.446, 9.6.447, 9.6.448, 9.6.449, 9.6.450, 9.6.451, 9.6.452, 9.6.453, 9.6.454, 9.6.455, 9.6.456, 9.6.457, 9.6.458, 9.6.459, 9.6.460, 9.6.461, 9.6.462, 9.6.463, 9.6.464, 9.6.465, 9.6.466, 9.6.467, 9.6.468, 9.6.469, 9.6.470, 9.6.471, 9.6.472, 9.6.473, 9.6.474, 9.6.475, 9.6.476, 9.6.477, 9.6.478, 9.6.479, 9.6.480, 9.6.481, 9.6.482, 9.6.483, 9.6.484, 9.6.485, 9.6.486, 9.6.487, 9.6.488, 9.6.489, 9.6.490, 9.6.491, 9.6.492, 9.6.493, 9.6.494, 9.6.495, 9.6.496, 9.6.497, 9.6.498, 9.6.499, 9.6.500, 9.6.501, 9.6.502, 9.6.503, 9.6.504, 9.6.505, 9.6.506, 9.6.507, 9.6.508, 9.6.509, 9.6.510, 9.6.511, 9.6.512, 9.6.513, 9.6.514, 9.6.515, 9.6.516, 9.6.517, 9.6.518, 9.6.519, 9.6.520, 9.6.521, 9.6.522, 9.6.523, 9.6.524, 9.6.525, 9.6.526, 9.6.527, 9.6.528, 9.6.529, 9.6.530, 9.6.531, 9.6.532, 9.6.533, 9.6.534, 9.6.535, 9.6.536, 9.6.537, 9.6.538, 9.6.539, 9.6.540, 9.6.541, 9.6.542, 9.6.543, 9.6.544, 9.6.545, 9.6.546, 9.6.547, 9.6.548, 9.6.549, 9.6.550, 9.6.551, 9.6.552, 9.6.553, 9.6.554, 9.6.555, 9.6.556, 9.6.557, 9.6.558, 9.6.559, 9.6.560, 9.6.561, 9.6.562, 9.6.563, 9.6.564, 9.6.565, 9.6.566, 9.6.567, 9.6.568, 9.6.569, 9.6.570, 9.6.571, 9.6.572, 9.6.573, 9.6.574, 9.6.575, 9.6.576, 9.6.577, 9.6.578, 9.6.579, 9.6.580, 9.6.581, 9.6.582, 9.6.583, 9.6.584, 9.6.585, 9.6.586, 9.6.587, 9.6.588, 9.6.589, 9.6.590, 9.6.591, 9.6.592, 9.6.593, 9.6.594, 9.6.595, 9.6.596, 9.6.597, 9.6.598, 9.6.599, 9.6.600, 9.6.601, 9.6.602, 9.6.603, 9.6.604, 9.6.605, 9.6.606, 9.6.607, 9.6.608, 9.6.609, 9.6.610, 9.6.611, 9.6.612, 9.6.613, 9.6.614, 9.6.615, 9.6.616, 9.6.617, 9.6.618, 9.6.619, 9.6.620, 9.6.621, 9.6.622, 9.6.623, 9.6.624, 9.6.625, 9.6.626, 9.6.627, 9.6.628, 9.6.629, 9.6.630, 9.6.631, 9.6.632, 9.6.633, 9.6.634, 9.6.635, 9.6.636, 9.6.637, 9.6.638, 9.6.639, 9.6.640, 9.6.641, 9.6.642, 9.6.643, 9.6.644, 9.6.645, 9.6.646, 9.6.647, 9.6.648, 9.6.649, 9.6.650, 9.6.651, 9.6.652, 9.6.653, 9.6.654, 9.6.655, 9.6.656, 9.6.657, 9.6.658, 9.6.659, 9.6.660, 9.6.661, 9.6.662, 9.6.663, 9.6.664, 9.6.665, 9.6.666, 9.6.667, 9.6.668, 9.6.669, 9.6.670, 9.6.671, 9.6.672, 9.6.673, 9.6.674, 9.6.675, 9.6.676, 9.6.677, 9.6.678, 9.6.679, 9.6.680, 9.6.681, 9.6.682, 9.6.683, 9.6.684, 9.6.685, 9.6.686, 9.6.687, 9.6.688, 9.6.689, 9.6.690, 9.6.691, 9.6.692, 9.6.693, 9.6.694, 9.6.695, 9.6.696, 9.6.697, 9.6.698, 9.6.699, 9.6.700, 9.6.701, 9.6.702, 9.6.703, 9.6.704, 9.6.705, 9.6.706, 9.6.707, 9.6.708, 9.6.709, 9.6.710, 9.6.711, 9.6.712, 9.6.713, 9.6.714, 9.6.715, 9.6.716, 9.6.717, 9.6.718, 9.6.719, 9.6.720, 9.6.721, 9.6.722, 9.6.723, 9.6.724, 9.6.725, 9.6.726, 9.6.727, 9.6.728, 9.6.729, 9.6.730, 9.6.731, 9.6.732, 9.6.733, 9.6.734, 9.6.735, 9.6.736, 9.6.737, 9.6.738, 9.6.739, 9.6.740, 9.6.741, 9.6.742, 9.6.743, 9.6.744, 9.6.745, 9.6.746, 9.6.747, 9.6.748, 9.6.749, 9.6.750, 9.6.751, 9.6.752, 9.6.753, 9.6.754, 9.6.755, 9.6.756, 9.6.757, 9.6.758, 9.6.759, 9.6.760, 9.6.761, 9.6.762, 9.6.763, 9.6.764, 9.6.765, 9.6.766, 9.6.767, 9.6.768, 9.6.769, 9.6.770, 9.6.771, 9.6.772, 9.6.773, 9.6.774, 9.6.775, 9.6.776, 9.6.777, 9.6.778, 9.6.779, 9.6.780, 9.6.781, 9.6.782, 9.6.783, 9.6.784, 9.6.785, 9.6.786, 9.6.787, 9.6.788, 9.6.789, 9.6.790, 9.6.791, 9.6.792, 9.6.793, 9.6.794, 9.6.795, 9.6.796, 9.6.797, 9.6.798, 9.6.799, 9.6.800, 9.6.801, 9.6.802, 9.6.803, 9.6.804, 9.6.805, 9.6.806, 9.6.807, 9.6.808, 9.6.809, 9.6.810, 9.6.811, 9.6.812, 9.6.813, 9.6.814, 9.6.815, 9.6.816, 9.6.817, 9.6.818, 9.6.819, 9.6.820, 9.6.821, 9.6.822, 9.6.823, 9.6.824, 9.6.825, 9.6.826, 9.6.827, 9.6.828, 9.6.829, 9.6.830, 9.6.831, 9.6.832, 9.6.833, 9.6.834, 9.6.835, 9.6.836, 9.6.837, 9.6.838, 9.6.839, 9.6.840, 9.6.841, 9.6.842, 9.6.843, 9.6.844, 9.6.845, 9.6.846, 9.6.847, 9.6.848, 9.6.849, 9.6.850, 9.6.851, 9.6.852, 9.6.853, 9.6.854, 9.6.855, 9.6.856, 9.6.857, 9.6.858, 9.6.859, 9.6.860, 9.6.861, 9.6.862, 9.6.863, 9.6.864, 9.6.865, 9.6.866, 9.6.867, 9.6.868, 9.6.869, 9.6.870, 9.6.871, 9.6.872, 9.6.873, 9.6.874, 9.6.875, 9.6.876, 9.6.877, 9.6.878, 9.6.879, 9.6.880, 9.6.881, 9.6.882, 9.6.883, 9.6.884, 9.6.885, 9.6.886, 9.6.887, 9.6.888, 9.6.889, 9.6.890, 9.6.891, 9.6.892, 9.6.893, 9.6.894, 9.6.895, 9.6.896, 9.6.897, 9.6.898, 9.6.899, 9.6.900, 9.6.901, 9.6.902, 9.6.903, 9.6.904, 9.6.905, 9.6.906, 9.6.907, 9.6.908, 9.6.909, 9.6.910, 9.6.911, 9.6.912, 9.6.913, 9.6.914, 9.6.915, 9.6.916, 9.6.917, 9.6.918, 9.6.919, 9.6.920, 9.6.921, 9.6.922, 9.6.923, 9.6.924, 9.6.925, 9.6.926, 9.6.927, 9.6.928, 9.6.929, 9.6.930, 9.6.931, 9.6.932, 9.6.933, 9.6.934, 9.6.935, 9.6.936, 9.6.937, 9.6.938, 9.6.939, 9.6.940, 9.6.941, 9.6.942, 9.6.943, 9.6.944, 9.6.945, 9.6.946, 9.6.947, 9.6.948, 9.6.949, 9.6.950, 9.6.951, 9.6.952, 9.6.953, 9.6.954, 9.6.955, 9.6.956, 9.6.957, 9.6.958, 9.6.959, 9.6.960, 9.6.961, 9.6.962, 9.6.963, 9.6.964, 9.6.965, 9.6.966, 9.6.967, 9.6.968, 9.6.969, 9.6.970, 9.6.971, 9.6.972, 9.6.973, 9.6.974, 9.6.975, 9.6.976, 9.6.977, 9.6.978, 9.6.979, 9.6.980, 9.6.981, 9.6.982, 9.6.983, 9.6.984, 9.6.985, 9.6.986, 9.6.987, 9.6.988, 9.6.989, 9.6.990, 9.6.991, 9.6.992, 9.6.993, 9.6.994, 9.6.995, 9.6.996, 9.6.997, 9.6.998, 9.6.999, 9.7.000, 9.7.001, 9.7.002, 9.7.003, 9.7.004, 9.7.005, 9.7.006, 9.7.007, 9.7.008, 9.7.009, 9.7.010, 9.7.011, 9.7.012, 9.7.013, 9.7.014, 9.7.015, 9.7.016, 9.7.017, 9.7.018, 9.7.019, 9.7.020, 9.7.021, 9.7.022, 9.7.023, 9.7.024, 9.7.025, 9.7.026, 9.7.027, 9.7.028, 9.7.029, 9.7.030, 9.7.031, 9.7.032, 9.7.033, 9.7.034, 9.7.035, 9.7.036, 9.7.037, 9.7.038, 9.7.039, 9.7.040, 9.7.041, 9.7.042, 9.7.043, 9.7.044, 9.7.045, 9.7.046, 9.7.047, 9.7.048, 9.7.049, 9.7.050, 9.7.051, 9.7.052, 9.7.053, 9.7.054, 9.7.055, 9.7.056, 9.7.057, 9.7.058, 9.7.059, 9.7.060, 9.7.061, 9.7.062, 9.7.063, 9.7.064, 9.7.065, 9.7.066, 9.7.067, 9.7.068, 9.7.069, 9.7.070, 9.7.071, 9.7.072, 9.7.073, 9.7.074, 9.7.075, 9.7.076, 9.7.077, 9.7.078, 9.7.079, 9.7.080, 9.7.081, 9.7.082, 9.7.083, 9.7.084, 9.7.085, 9.7.086, 9.7.087, 9.7.088, 9.7.089, 9.7.090, 9.7.091, 9.7.092, 9.7.093, 9.7.094, 9.7.095, 9.7.096, 9.7.097, 9.7.098, 9.7.099, 9.7.100, 9.7.101, 9.7.102, 9.7.103, 9.7.104, 9.7.105, 9.7.106, 9.7.107, 9.7.108, 9.7.109, 9.7.110, 9.7.111, 9.7.112, 9.7.113, 9.7.114, 9.7.115, 9.7.116, 9.7.117, 9.7.118, 9.7.119, 9.7.120, 9.7.121, 9.7.122, 9.7.123, 9.7.124, 9.7.125, 9.7.126, 9.7.127, 9.7.128, 9.7.129, 9.7.130, 9.7.131, 9.7.132, 9.7.133, 9.7.134, 9.7.135, 9.7.136, 9.7.137, 9.7.138, 9.7.139, 9.7.140, 9.7.141, 9.7.142, 9.7.143, 9.7.144, 9.7.145, 9.7.146, 9.7.147, 9.7.148, 9.7.149, 9.7.150, 9.7.151, 9.7.152, 9.7.153, 9.7.154, 9.7.155, 9.7.156, 9.7.157, 9.7.158, 9.7.159, 9.7.160, 9.7.161, 9.7.162, 9.7.163, 9.7.164, 9.7.165, 9.7.166, 9.7.167, 9.7.168, 9.7.169, 9.7.170, 9.7.171, 9.7.172, 9.7.173, 9.7.174, 9.7.175, 9.7.176, 9.7.177, 9.7.178, 9.7.179, 9.7.180, 9.7.181, 9.7.182, 9.7.183, 9.7.184, 9.7.185, 9.7.186, 9.7.187, 9.7.188, 9.7.189, 9.7.190, 9.7.191, 9.7.192, 9.7.193, 9.7.194, 9.7.195, 9.7.196, 9.7.197, 9.7.198, 9.7.199, 9.7.200, 9.7.201, 9.7.202, 9.7.203, 9.7.204, 9.7.205, 9.7.206, 9.7.207, 9.7.208, 9.7.209, 9.7.210, 9.7.211, 9.7.212, 9.7.213, 9.7.214, 9.7.215, 9.7.216, 9.7.217, 9.7.218, 9.7.219, 9.7.220, 9.7.221, 9.7.222, 9.7.223, 9.7.224, 9.7.225, 9.7.226, 9.7.227, 9.7.228, 9.7.229, 9.7.230, 9.7.231, 9.7.232, 9.7.233, 9.7.234, 9.7.235, 9.7.236, 9.7.237, 9.7.238, 9.7.239, 9.7.240, 9.7.241, 9.7.242, 9.7.243, 9.7.244, 9.7.245,

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);
```

설명

```
&#xad50;&#xcc28;&#xd558;&#xb294;&#xb3c4;&#xb85c; 2&#xac1c;&#xc640;&#xc8fc;,&#xc2dc;,&#xc9d1;&#xcf54;&#xb4dc;&#xc785;&#xb825;&#xbc1b;&#xc544;&#xac01;&#xc704;&#xce58;&#xc5d0;&#xb300;&#xd55c; normalized_address (addy), NAD83 &#xacbd;&#xc704;&#xb3c4;&#xb85c; &#xd22c;&#xc601;&#xb41c; &#xd3ec;&#xc778;&#xd2b8; &#xc704;&#xce58;&#xadf8;&#xb9ac;&#xace0; &#xc21c;&#xc704;(rating)&#xb3c4; &#xd3ec;&#xd568;&#xd558;&#xb294;,&#xad50;&#xcc28;&#xcc98;&#xc74c; &#xac00;&#xb85c;&#xc9c0;&#xb974;&#xb294; &#xb3c4;&#xb85c;&#xc5d0; &#xc788;&#xb294; &#xac00;&#xc788;&#xb294; &#xc704;&#xce58;&#xb4e4;&#xc758; &#xc9d1;&#xd569;&#xc744; &#xcd9c;&#xb825;&#xd569;&#xb2c8;&#xc21c;&#xc704;&#xac00; &#xb0ae;&#xc744;&#xc218;&#xb85d; &#xc8fc;&#xc18c;&#xc640; &#xc704;&#xce58;&#xac00; &#xc77c;&#xce58;&#xd560; &#xac00;&#xb2a5;&#xc131;&#xc774; &#xcee4;&#xc9d1;&#xb2c8;&#xb2e4;.&#xacb0;&#xacfc;&#xc21c;&#xc704;&#xac00; &#xb0ae;&#xc740; &#xc21c;&#xc11c;&#xb85c; &#xc815;&#xb82c;&#xd569;&#xb2c8;&#xb2e4;.&#xacb0;&#xacfc;&#xc758; &#xcd5c;&#xb300; &#xac1c;&#xc218;(&#xae30;&#xbcf8;&#xac12; 10) &#xc635;&#xc158; &#xd30c; &#xb118;&#xaca8;&#xc904; &#xc218; &#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.&#xc788; &#xb370;&#xc774;&#xd130;(edge, face, addr) &#xbc0f; PostgreSQL &#xd37c;&#xc9c0; &#xbb38;&#xc790;&#xc5f4; &#xb9e4;&#xce6d;(soundex, levenshtein)&#xc744;&#xc774;&#xc6a9;&#xd569;&#xb2c8;&#xb2e4;.
```

2.0.0 버전부터 사용할 수 있습니다.

기본 예시

매사추세츠(MA) 주의모든TIGER 데이터로드한 PostgreSQL 9.0/PostGIS 1.5가실행되고있는 3.0 GHZ 단일프로세서냏 2GB 램윈도우7 컴퓨터에서다음예시를실행시데걸리는시간을측정했습니다.현재조금느린편입니다(3,000밀리초TIGER 2011 데이터를로드한 PostGIS 2.0과 PostgreSQL 64비஄전을실행중인 8GB 램윈도우 2003 64비컴퓨터에서테스트한결과는더빠릅니다(41밀리초).

```
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

지오코딩 도구에 집코드를 넘겨&#xa
않더라도 추측할 수 있습니다. 윈&
7 컴퓨터에서는 3,500밀리초, 윈도우
2003 64비트 컴퓨터에서는 741밀리초 걸&

```
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

󌰸고

Geocode, Pprint_Addy, ST_AsText

14.2.6 Get_Geocode_Setting

Get_Geocode_Setting — tiger.geocode_settings 테이블에 저장된 특쟀설정값을 반환합니다.

Synopsis

```
text Get_Geocode_Setting(text setting_name);
```

설명

tiger.geocode_settings 테이블에저장된특정설정ଘ환합니다. 사용자가함수의디&#xc7cf1c;󊳠끌수있게해주는설정값입&#xc7d5a5;후설정값으로순위를제어할수있도록할򬳤획입니다. 현재설목록은다음과같습니다.

```
&#xba85; &#xce6d; | &#xc124; &#xc815; &#xac12; | ←
&#xb370; &#xc774; &#xd130; &#xd615; | &#xce74; &#xd14c; &#xace0; &#xb9ac; | ←
&#xc124; &#xba85; &#xac04; &#xb2e8; &#xd55c; ←
-----+-----+-----+-----+-----+
debug_geocode_address | false | boolean | debug | &#xcc38; &#xc77c; ←
&#xacbd; &#xc6b0; geocode_address &#xb97c; &#xd638; &#xcd9c; &#xd560; &#xb54c; NOTICE ←
&#xb85c; &#xadf8; &#xc5d0; &#xcffc; &#xb9ac; &#xc640; &#xac19; &#xc740; ←
&#xb514; &#xbc84; &#xadf8; &#xc815; &#xbcf4; &#xb97c; ←
&#xcd9c; &#xb825; &#xd569; &#xb2c8; &#xb2e4; .
debug_geocode_intersection | false | boolean | debug | &#xcc38; &#xc77c; ←
&#xacbd; &#xc6b0; geocode_intersection &#xc744; &#xd638; &#xcd9c; &#xd560; &#xb54c; NOTICE ←
&#xb85c; &#xadf8; &#xc5d0; &#xcffc; &#xb9ac; &#xc640; &#xac19; &#xc740; ←
&#xb514; &#xbc84; &#xadf8; &#xc815; &#xbcf4; &#xb97c; ←
&#xcd9c; &#xb825; &#xd569; &#xb2c8; &#xb2e4; .
debug_normalize_address | false | boolean | debug | &#xcc38; &#xc77c; ←
&#xacbd; &#xc6b0; normalize_address &#xb97c; &#xd638; &#xcd9c; &#xd560; &#xb54c; NOTICE ←
&#xb85c; &#xadf8; &#xc5d0; &#xcffc; &#xb9ac; &#xbc0f; &#xc911; &#xac04; ←
&#xd45c; &#xd604; &#xc2dd; &#xacfc; &#xac19; &#xc740; &#xb514; &#xbc84; &#xadf8; ←
&#xc815; &#xbcf4; &#xb97c; &#xcd9c; &#xb825; &#xd569; &#xb2c8; &#xb2e4; .
debug_reverse_geocode | false | boolean | debug | &#xcc38; &#xc77c; ←
&#xacbd; &#xc6b0; reverse_geocode &#xb97c; &#xd638; &#xcd9c; &#xd560; &#xb54c; NOTICE ←
&#xb85c; &#xadf8; &#xc5d0; &#xcffc; &#xb9ac; &#xbc0f; &#xc911; &#xac04; ←
&#xd45c; &#xd604; &#xc2dd; &#xacfc; &#xac19; &#xc740; &#xb514; &#xbc84; &#xadf8; ←
&#xc815; &#xbcf4; &#xb97c; &#xcd9c; &#xb825; &#xd569; &#xb2c8; &#xb2e4; .
reverse_geocode_numbered_roads | 0 | integer | rating | &#xc8fc; &#xbc0f; ←
&#xad70; &#xace0; &#xc18d; &#xb3c4; &#xb85c; &#xc5d0; &#xb300; &#xd574; , 0 - ←
&#xc5b4; &#xb5a4; &#xba85; &#xce6d; &#xb3c4; &#xc6b0; &#xc120; &#xd558; &#xc9c0; ←
&#xc54a; &#xace0; , 1 - &#xbc88; &#xd638; &#xb85c; &#xb41c; ←
&#xace0; &#xc18d; &#xb3c4; &#xb85c; &#xba85; &#xce6d; &#xc744; ←
&#xc6b0; &#xc120; &#xd558; &#xba70; , 2 - &#xc9c0; &#xbc29; &#xc758; &#xc8fc; / &#xad70; ←
&#xba85; &#xce6d; &#xc744; &#xc6b0; &#xc120; &#xd569; &#xb2c8; &#xb2e4; .
use_page_address_parser | false | boolean | normalize | &#xcc38; &#xc73c; &#xb85c; ←
&#xc124; &#xc815; &#xd560; &#xacbd; &#xc6b0; , &#xbe4c; &#xb4dc; &#xb41c; TIGER ←
normalize_address &#xb300; &#xc2e0; (page_normalize_address &#xb97c; &#xd1b5; &#xd574;) ←
```

```
address_standardizer &#xd655;&#xc7a5; &#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc744;  ←
&#xc774;&#xc6a9;&#xd558;&#xb824; &#xd560; &#xac83;&#xc785;&#xb2c8;&#xb2e4; .
```

변경 사항: 2.2.0 버전부터 geocode_settings_default 테이기본 설정값을 저장합니다. 사용지정 설정값은 geocode_settings 테이블에 있이 geocode_settings 테이블에는 사용자가 설정정값만 저장합니다.

2.1.0 버전부터 사용할 수 있습니다.

예시: 디버깅 설정값 반환

```
SELECT get_geocode_setting('debug_geocode_address') As result;
result
-----
false
```

참고

Set_Geocode_Setting

14.2.7 Get_Tract

Get_Tract — 도형이 위치해 있는 구역(tract) 테인구조사 구역 또는 현장(field)을 반환기본적으로 구역의 축약된 명칭로 반환합니다.

Synopsis

text **get_tract**(geometry loc_geom, text output_field=name);

설명

도형을 입력받아 해당 도형이 위인구조사 구역을 반환합니다. 공인조 시스템을 따로 설정하지 않인경우 NAD83 경위도라고 가정합니다.

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](#).

2.0.0 버전부터 사용할 수 있습니다.

기#bcf8; #xc608;#xc2dc;

```
SELECT get_tract(ST_Point(-71.101375, 42.31376) ) As tract_name;
tract_name
-----
1203.01
```

```
--  &#xc774; &#xcffc;&#xb9ac;&#xb294; TIGER &#xb3c4;&#xd615; ID(geoid)&#xb97c;  ←
    &#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4; .
SELECT get_tract(ST_Point(-71.101375, 42.31376), 'tract_id' ) As tract_id;
tract_id
-----
25025120301
```

స고

Geocode >

14.2.8 Install_Missing_Indexes

```
Install_Missing_Indexes — &#xc9c0;&#xc624;&#xcf54;&#xb529; &#xb3c4;&#xad6c;&#xc758; &#xacb0;&#xd569;(join)&#xc5d0:
&#xc774;&#xc6a9;&#xb418;&#xb294; &#xd0a4;(key) &#xc5f4;&#xc744; &#xac00;&#xc9c4; &#xd14c;&#xc774;&#xbe14;&#xc744;
&#xbaa8;&#xb450; &#xcc3e;&#xc544;&#xc11c; &#xd574;&#xb2f9; &#xc5f4;&#xc5d0; &#xc4f0;&#xc778; &#xc778;&#xb371;&#xc744;
&#xac00;&#xc6b4;&#xb370; &#xc5c6;&#xc5b4;&#xc9c4; &#xac83;&#xb4e4;&#xc5d0; &#xb300;&#xd55c; &#xc870;&#xac74;&#xc744;
&#xd544;&#xd130;&#xb9c1;&#xd574;&#xc11c; &#xc5c6;&#xc5b4;&#xc9c4; &#xc778;&#xb371;&#xc2a4;&#xb4e4;&#xc744;
&#xcd94;&#xac00;&#xd569;&#xb2c8;&#xb2e4;.
```

Synopsis

```
boolean Install_Missing_Indexes();
```

설명

```
tiger'&#xc0f; tiger_data'&#xc2a4; &#xd0a4; &#xb9c8; &#xc5d0; &#xc11c; &#xc9c0; &#xc624; &#xcf54; &#xb529; &#xb3c4; &#  
&#xacb0; &#xd569; &#xc5d0; &#xc774; &#xc6a9; &#xb418; &#xb294; &#xd0a4; &#xc5f4; &#xc744; &#xac00; &#xc9c4; &#xd14c; &#  
&#baa8; &#xb450; &#xcc3e; &#xc740; &#xb2e4; &#xc74c; , &#xd574; &#xb2f9; &#xc5f4; &#xc5d0; &#xc11c; &#xc5c6; &#xc5b4; &#  
&#xc778; &#xb371; &#xc2a4; &#xb4e4; &#xc744; &#xd544; &#xd130; &#xb9c1; &#xd574; &#xc11c; , &#xd574; &#xb2f9; &#xd14c; &#  
&#xc704; &#xd55c; &#xc778; &#xb371; &#xc2a4; &#xb97c; &#xc815; &#xc758; &#xd558; &#xae30; &#xc704; &#xd55c; SQL  
DDL&#xc744; &#xcd9c; &#xb825; &#xd558; &#xace0; , &#xc0dd; &#xc131; &#xb41c; &#xc2a4; &#xd06c; &#xb9bd; &#xd2b8; &#xb97c;  
&#xc2e4; &#xd589; &#xd569; &#xb2c8; &#xb2e4; . &#xc774; &#xd568; &#xc218; &#xb294; &#xb85c; &#xb4dc; &#xacfc; &#xc815; &#  
&#xc5c6; &#xc5b4; &#xc84c; &#xc744; &#xc218; &#xb3c4; &#xc788; &#xb294; , &#xcffc; &#xb9ac; &#xb97c; &#xb354; &#xbe68; &#  
&#xd558; &#xae30; &#xc704; &#xd574; &#xd544; &#xc694; &#xd55c; &#xc0c8; &#xc778; &#xb371; &#xc2a4; &#xb4e4; &#xc744;  
&#xcd94; &#xac00; &#xd558; &#xb294; &#xb3c4; &#xc6b0; &#xbbf8; &#xd568; &#xc218; &#xc785; &#xb2c8; &#xb2e4; . &#xc774;  
&#xd568; &#xc218; &#xb294; &#xc778; &#xb371; &#xc2a4; &#xc0dd; &#xc131; &#xc2a4; &#xd06c; &#xb9bd; &#xd2b8; &#xb97c;  
&#xc791; &#xc131; &#xd558; &#xb294; Missing Indexes Generate Script &#xd568; &#xc218; &#xc640; &#xb3d9; &#xc77c; &#xd55c;  
&#xacc4; &#xc5f4; &#xb85c; , &#xc791; &#xc131; &#xb41c; &#xc2a4; &#xd06c; &#xb9bd; &#xd2b8; &#xb97c; &#xc2e4; &#xd589; &#  
&#xd569; &#xb2c8; &#xb2e4; . update_geocode.sql &#xc5c5; &#xadf8; &#xb808; &#xc774; &#xb4dc; &#xc2a4; &#xd06c; &#xb97c;  
&#xc77c; &#xbd80; &#xb85c; &#xc774; &#xd568; &#xc218; &#xb97c; &#xd638; &#xcd9c; &#xd569; &#xb2c8; &#xb2e4; .
```

2.0.0 버전부터 사용할 수 있습니다.

Loader_Generate_Script, Missing_Indexes_Generate_Script

```
SELECT install_missing_indexes();
       install_missing_indexes
-----
t
```

Loader_Generate_Script, Missing_Indexes_Generate_Script

Loader_Generate_Script, Missing_Indexes_Generate_Script

14.2.9 Loader_Generate_Census_Script

Loader_Generate_Census_Script — Generates a script to create a census table in a database. The script is generated based on the parameters provided. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`.

Synopsis

```
setof text loader_generate_census_script(text[] param_states, text os);
```

Loader_Generate_Census_Script

Loader_Generate_Census_Script — Generates a script to create a census table in a database. The script is generated based on the parameters provided. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`.

Loader_Generate_Census_Script — Generates a script to create a census table in a database. The script is generated based on the parameters provided. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`.

Loader_Generate_Census_Script — Generates a script to create a census table in a database. The script is generated based on the parameters provided. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`.

1. **loader_variables** - Generates a script to create a census table in a database. The script is generated based on the parameters provided. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`.
2. **loader_platform** - Generates a script to create a census table in a database. The script is generated based on the parameters provided. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`. The script is generated in a file named `loader_generate_census_script.sql`.


```
3. loader_lookuptables - \xc01; \xb808;\xcf54;\xb4dc;\xac00; \xd14c;\xc774;\xbe14; \xc720;\xd61
&\xad70;), \xd14c;\xc774;\xbe14;\xc5d0;\xc11c; \xb808;\xcf54;\xb4dc;\xb97c; \xcc98;\xb9ac;\xd5
&\xc5ec;\xbd80;, \xd14c;\xc774;\xbe14;\xc5d0; \xb808;\xcf54;\xb4dc;\xb97c; \xb85c;\xb4dc;\xd5
&\xbc29;\xbc95;\xc744; \xc815;\xc758;\xd569;\xb2c8;\xb2e4;. \xac01; \xb2e8;\xacc4;\xbcc4;\xb8
&\xb370;\xc774;\xd130;\xb97c; \xc784;\xd3ec;\xd2b8;\xd558;\xace0; \xc900;\xbe44;\xd558;\xba7
&\xac01; \xb2e8;\xacc4;\xc5d0; \xb300;\xd574; \xc5f4;, \xc778;\xb371;\xc2a4;, \xc81c;\xc57d;\xc
&\xcd94;\xac00;\xd558;\xace0; \xc81c;\xac70;\xd558;\xb294; \xb2e8;\xacc4;\xb4e4;\xc744;
&\xc815;\xc758;\xd569;\xb2c8;\xb2e4;. \xac01; \xd14c;\xc774;\xbe14;\xc758; \xc811;\xb450;\xc0
&\xc8fc;\xba85;(\x5dde;\x540d);\xc774;\xba70;, TIGER \xc2a4;\xd0a4;\xb9c8; \xc548;\xc758;
&\xd14c;\xc774;\xbe14;\xc744; \xc2b9;\xacc4;\xd569;\xb2c8;\xb2e4;. \xc608;: tiger.faces
&\xb97c; \xc2b9;\xacc4;\xd558;\xb294; tiger_data.ma_faces \xd14c;\xc774;\xbe14;\xc744;
&\xc0dd;\xc131;\xd558;\xc2ed;\xc2dc;\xc624;.
```

2.0.0 \xbc84;\xc804;\xbd80;\xd130; \xc0ac;\xc6a9;\xd560; \xc218; \xc788;\xc2b5;\xb2c8;\xb2e4;.

Note



Loader_Generate_Script \xb294; \xc774; \xb85c;\xc9c1;\xc744; \xd3ec;\xd568;\xd558;\xc9c0;\xb9cc;,
 PostGIS 2.0.0 alpha5 \xc774;\xc804; \xbc84;\xc804;\xc5d0; TIGER \xc9c0;\xc624;\xcf54;\xb529;
 \xb3c4;\xad6c;\xb97c; \xc124;\xce58;\xd55c; \xacbd;\xc6b0;, \xc774;\xb7f0;
 \xcd94;\xac00;\xc801;\xc778; \xd14c;\xc774;\xbe14;\xc744; \xc5bb;\xc73c;\xb824;\xba74;
 \xc0ac;\xc6a9;\xc790;\xac00; \xc774;\xbbf8; \xcc98;\xb9ac;\xd55c; \xc8fc;(\x5dde);\xc5d0;
 \xb300;\xd574; \xc774; \xd568;\xc218;\xb97c; \xc2e4;\xd589;\xd574;\xc57c; \xd560;
 \xac83;\xc785;\xb2c8;\xb2e4;.

\xc608;\xc2dc;

\xc120;\xd0dd;\xd55c; \xc8fc;\xb4e4;\xc744; \xc704;\xd55c; \xb370;\xc774;\xd130;\xb97c; \xc708;\xc
 \xc178; \xc2a4;\xd06c;\xb9bd;\xd2b8; \xd615;\xc2dd;\xc73c;\xb85c; \xb85c;\xb4dc;\xd558;\xb294;
 \xc2a4;\xd06c;\xb9bd;\xd2b8;\xb97c; \xc0dd;\xc131;\xd569;\xb2c8;\xb2e4;.

```
SELECT loader_generate_census_script (ARRAY['MA'], 'windows');
-- \xacb0;\xacfc; --
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'); "
:
```

```
.sh
STATDIR="/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 $STATDIR
for z in *.zip; do $UNZIPTOOL -o -d $TMPDIR $z; done
:
```

```
&#xccc38;&#xace0;
```

[Loader_Generate_Script](#)

14.2.10 Loader_Generate_Script

Loader_Generate_Script — 설정한플랫폼에대해설주󅷞를위한, TIGER 데이터를다운로로스키마에단계별로로드하는셸스크립트를생성합니다. 각주(스크립트를개별레코드로로환౜신஄전은 TIGER 2010 구조변경사항을지원하며, 인구조사구역, &#be14;록그&#be14;록테이&#be14;도로드합니다.

Synopsis

```
setof text loader_generate_script(text[] param_states, text os);
```

설명

설정한플랫폼에대해설정한주(위한,위해이터를다운로드해서 tiger_data 스키마에단계별로로드하는셸스크립트를생성합니다.각주(스크립트를개별레코드로򸰘환리눅스상에서 unzip을 (윈도우상에서기본적으로 7-zip을) 사용하ૠ,다운로위해 wget을 씁니다.데이터를로드해데 Section 4.7.2 를이용합니다.이함수가작업에이용하는가장작은단위주(헞)전체라는점에주의하십시다만사용자가직접파일들을다이단위를덮어쓸수있습니다.이함수는 "staging" 볰 "temp" 폴더에있는파일만리할것입니다.

ಘ리정볰서로다른 OS 셸ସக변수들을제어하기위해다음제테이&#bbe14;들을이용합니다.

1. loader_variables - 인구조사현장,연도,데이볰단계별(staging) 스키마들주ఙ은여변수들을추적합니다.
2. loader_platform - 여러플랫폼볰여러실행파일들의위엨에대한జ요를작성합니다.기본적으로윈도우리눅스를가지ૠ있습니다.더많은플랫폼을추가할수있습니다.
3. loader_lookuptables - ఁ레코드가테이&#bbe14;유혔군,테이&#bbe14;에서레코드를ಘ리하여부,테이&#bbe14;에레코드를로드하଩க을정의합니다.ఁ단계별로데이터를임포트하ૠ준&#bbe4;하뫷ఁ단계에대해열,인덱스,제약을추가하ૠ제౰하는단계들을정의합니다.ఁ테이&#bbe14;의접두ఁ주명(헞핀)이며,스키마안의테이&#bbe14;을승계합니다.예: tiger.faces 를승계하는 tiger_data.ma_faces 테이&#bbe14;을생성하십시오.

2.0.0 ஄전부터사용할수있습니다.

TIGER 2010 데이터구조를지원하며인구구역(tract), &#bbe14;록그룹(bg), &#bbe14;록(tabblock) 테이&#bbe14;로드합니다.

**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.

2.4.1.1. Creating the data_load.sh script

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;
```

```
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
:
:
```

```
.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}/*.zip
${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
:
:
```

2.4.1.2. Creating the data_load.sh script**Section 2.4.1, Pprint_Addy, ST_AsText**

14.2.11 Loader_Generate_Nation_Script

Loader_Generate_Nation_Script — tiger_data, county_all, county_all_lookup, state_all, tiger_loader_platform, tiger_loader_variables, tiger_loader_lookuptables

Synopsis

```
text loader_generate_nation_script(text os);
```

Loader_Generate_Nation_Script

tiger_data, county_all, county_all_lookup, state_all, tiger_loader_platform, tiger_loader_variables, tiger_loader_lookuptables

unzip, wget, Section 4.7.2

OS, tiger_loader_platform, tiger_loader_variables, tiger_loader_lookuptables

1. loader_variables - (staging)
2. loader_platform -
3. loader_lookuptables -

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.

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

```
&#xc0ac;&#xc6a9;&#xc790;&#xac00;tiger_2010&#xbc84;&#xc804;&#xc744;&#xc0ac;&#xc6a9;&#xc911;&#xc774;&#xc8fc;(&#x5dde;)&#xb97c; tiger_2011 &#xbc84;&#xc804;&#xc73c;&#xb85c; &#xb2e4;&#xc2dc;&#xb85c;&#xb4dc;&#xd558;&#xb824;&#xd560;&#xacbd;&#xc6b0;&#xc774;&#xc2a4;&#xd06c;&#xb9bd;&#xd2b8;&#xb97c;&#xc2e4;&#xd589;&#xd558;&#xae30;&#xc804;&#xc5d0;&#xc81c;&#xc77c;&#xba3c;&#xc800;&#xc0ad;&#xc81c;&#xc120;&#xc5b8;&#xbb38; Drop_Nation_Tables_Generate_Script &#xb97c;&#xc0dd;&#xc131;&#xd558;&#xae0;&#xc2e4;&#xd589;&#xd574;&#xc57c;&#xd560;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.
```

예시

윈도우상에서 국가 데이터를 로위한 스크립트를 생성합니다.

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

리눅스/유닉스시스템상에서데&#
로드하기위한스크립트를생성&#

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

స고

Loader_Generate_Script, Missing_Indexes_Generate_Script

14.2.12 Missing Indexes Generate Script

Missing_Indexes_Generate_Script — 지오코딩 도구의 결합(join)이용되는 키(key) 열을 가진 테이블 가운데 해당 열에서 없어진 인덱모두 찾아서 해당 테이블에 대한 인덱스를 정의하는 SQL DDL을 출력합이

Synopsis

```
text Missing_Indexes_Generate_Script();
```

설명

```
tiger &#xbc0f;tiger_data &#xc2a4;&#xd0a4;&#xb9c8;&#xc5d0;&#xc11c; &#xc9c0;&#xc624;&#xcf54;&#xb529; &#xb3c4;&#xacb0;&#xd569;(join)&#xc5d0; &#xc774;&#xc6a9;&#xb418;&#xb294; &#xd0a4;(key) &#xc5f4;&#xc744; &#xac00;&#xc9c4; &#xd14c;&#xc774;&#xbe14; &#xac00;&#xc6b4;&#xb370; &#xd574;&#xb2f9; &#xc5f4;&#xc5d0;&#xc11c; &#xc5c6;&#xc5b4;&#xc778;&#xb371;&#xc2a4;&#xb97c; &#xbaa8;&#xb450; &#xcc3e;&#xc544;&#xc11c; &#xd574;&#xb2f9; &#xd14c;&#xc774;&#xb300;&#xd55c; &#xc778;&#xb371;&#xc2a4;&#xb97c; &#xc815;&#xc758;&#xd558;&#xb294; SQL DDL&#xc744; &#xcd9c;&#xc774; &#xd568;&#xc218;&#xb294; &#xb85c;&#xb4dc; &#xacfc;&#xc815;&#xc5d0;&#xc11c; &#xc5c6;&#xc5b4;&#xc84c;&#xc218;&#xb3c4; &#xc788;&#xb294;.&#xcffc;&#xb9ac;&#xb97c; &#xb354; &#xbe68;&#xb9ac; &#xd558;&#xae30; &#xc704;&#xc774;
```



```
&#xd544;&#xc694;&#xd55c;&#xc0c8;&#xc778;&#xb371;&#xc2a4;&#xb4e4;&#xc744;&#xcd94;&#xac00;&#xd558;&#xb294;
&#xb3c4;&#xc6b0;&#xbbbf8;&#xd568;&#xc218;&#xc785;&#xb2c8;&#xb2e4;.&#xc9c0;&#xc624;&#xcf54;&#xb529;&#xb3c4;&#
&#xd5a5;&#xc0c1;&#xb418;&#xba74;.&#xc0ac;&#xc6a9;&#xb418;&#xb294;&#xc0c8;&#xc778;&#xb371;&#xc2a4;&#xb97c;
&#xc218;&#xc6a9;&#xd558;&#xae30;&#xc704;&#xd574;&#xc774;&#xd568;&#xc218;&#xb3c4;&#xc5c5;&#xb370;&#xc774;&#
&#xac83;&#xc785;&#xb2c8;&#xb2e4;.&#xc774;&#xd568;&#xc218;&#xac00;&#xc544;&#xbb34;&#xac83;&#xb3c4;&#xcd9c;&#
&#xc54a;&#xc744;&#xacbd;&#xc6b0;.&#xbaa8;&#xb4e0;&#xd14c;&#xc774;&#xbe14;&#xc774;&#xc774;&#xbbbf8;&#xd0a4;
&#xc778;&#xb371;&#xc2a4;&#xb85c;&#xac04;&#xc8fc;&#xb418;&#xb294;&#xac83;&#xc744;&#xac00;&#xc9c0;&#xace0;
&#xc788;&#xb2e4;&#xb294;&#xb73b;&#xc785;&#xb2c8;&#xb2e4;.
```

```
2.0.0&#xbc84;&#xc804;&#xbd80;&#xd130;&#xc0ac;&#xc6a9;&#xd560;&#xc218;&#xc788;&#xc2b5;&#xb2c8;&#xb2e4;.
```

예시

```
SELECT missing_indexes_generate_script();
-- &#xcd9c;&#xb825;&#xbbb3c;:&#xb85c;&#xb4dc;&#xc791;&#xc5c5;↵
&#xc2a4;&#xd06c;&#xb9bd;&#xd2b8;&#xc5d0;&#xb9ce;&#xc740;&#xc218;&#xc815;&#xc774;↵
&#xac00;&#xd574;&#xc9c0;&#xae30;&#xc804;&#xc5d0;&#xc0dd;&#xc131;&#xb41c;↵
&#xb370;&#xc774;&#xd130;&#xbca0;&#xc774;&#xc2a4;&#xc0c1;&#xc5d0;&#xc11c;↵
&#xc2e4;&#xd589;&#xd55c;&#xac83;&#xc785;&#xb2c8;&#xb2e4;.--
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);
```

참고

[Loader_Generate_Script, Install_Missing_Indexes](#)

14.2.13 Normalize_Address

Normalize_Address — 򻬸자형도로주소를입력받&#
도로접򻯸사.접두사및유형을표&#
도로.도로명등을개별필드로분&#
norm_addy 합성유형을렘환합니다.이
함수는tiger_geocoder와함께패키징된색&#xc77
데이터만이용해서(TIGER인구조사
데이터는필요없이)작동할것입&#

Synopsis

```
norm_addy normalize_address(varchar in_address);
```

설명

```
&#xbbb38;&#xc790;&#xd615;&#xb3c4;&#xb85c;&#xc8fc;&#xc18c;&#xb97c;&#xc785;&#xb825;&#xbc1b;&#xc544;.&#xb3c4;&#
&#xc811;&#xbbbf8;&#xc0ac;.&#xc811;&#xb450;&#xc0ac;&#xbc0f;&#xc720;&#xd615;&#xc744;&#xd45c;&#xc900;&#xd654;&#
&#xb3c4;&#xb85c;.&#xb3c4;&#xb85c;&#xba85;&#xb4f1;&#xc744;&#xac1c;&#xbcc4;&#xd544;&#xb4dc;&#xb85c;&#xbd84;&#
```

norm_addy 합성 유형을 반환합니다. 모든
 주소를 정귞화된 우편 형식으로
 변환하기 위한 지오코딩 과정의
 첫 번째 단계입니다. 지오코딩 도위
 함께 패키징된 데이터 외에 다른
 데이터는 필요하지 않습니다.

이 함수는 tiger 스키마에 있ૠ tiger_geocoder와
 함께 미리 로드돼 있는 여러 방향/색인 테이블만 이용합니다. 따라이
 함수를 이용하기 위해 TIGER 인구색인
 데이터 또는 다른 추가적인 데이다
 다운로드할 필요가 없습니다. tiger
 스키마에 있는 여러 색인 테이블더
 만은 약어 또는 대체 명칭들을
 추가해야 할 필요가 있을 수도 있이
 함수는 입력 주소를 정귞화하위해
 tiger 스키마에 있는 여러 제어 색인
 테이블을 이용합니다.

이 함수는 norm_addy 유형 객체 안의 필듐
 다음과 같은 순서로 반환합니다. 이때
 ()가 지오코딩 도구가 요구하필드
 를 나타내ૠ , []는 선택적인 필나타
 냅니다.

(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 는 정수형입니다. 도로 번지위
2. predirAbbrev 는 varchar형입니다. N, S, E, W 등과 같은
 도로의 방향을 의미하는 접두색인
 direction_lookup 테이블을 이용해서 이 필제어
 합니다.
3. streetName 은 varchar형입니다.
4. streetTypeAbbrev 는 varchar형으로, St, Ave, Cir처럼 도로
 유형의 축약 버전입니다. street_type_lookup
 테이블을 이용해서 이 필드를
 제어합니다.
5. postdirAbbrev 는 varchar형으로, N, S, E, W 등과 같은
 도로의 방향을 의미하는 접미색인
 축약 버전입니다. direction_lookup 테이블을
 이용해서 이 필드를 제어합니다
6. internal 은 varchar형입니다. 아파트 또는
 빌라의 동호수와 같은 내부 주서
7. location 은 varchar형으로, 일반적으로 도위
 또는 지자체를 나타냅니다.
8. stateAbbrev 는 varchar형으로, MA, NY, MI처럼 두 글자
 표현한 미국의 주명(헞󕐍)입니다
 state_lookup 테이블을 이용해서 이 필드
 제어합니다.

9. `zip` `varchar(15)`; `02109` `varchar(10)`; `02109` `varchar(10)`; `02109` `varchar(10)`;
10. `parsed` `boolean`; `02109` `varchar(10)`; `02109` `varchar(10)`; `02109` `varchar(10)`;
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 `Page_Normalize_Address` function. Availability: PostGIS 2.4.0.

Geocode, Pprint_Addy

`SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev`
`FROM (SELECT address, normalize_address(address) As na`
`FROM addresses_to_geocode) As g;`

orig	streetname	streettypeabbrev
28 Capen Street, Medford, MA	Capen	St
124 Mount Auburn St, Cambridge, Massachusetts 02138	Mount Auburn	St
950 Main Street, Worcester, MA 01610	Main	St
529 Main Street, Boston MA, 02129	Main	St
77 Massachusetts Avenue, Cambridge, MA 02139	Massachusetts	Ave
25 Wizard of Oz, Walford, KS 99912323	Wizard of Oz	

Geocode, Pprint_Addy

Geocode, Pprint_Addy

14.2.14 Page_Normalize_Address

`Page_Normalize_Address` — `varchar(15)`; `02109` `varchar(10)`; `02109` `varchar(10)`; `02109` `varchar(10)`;

Synopsis

`norm_addy` `page_normalize_address`(`varchar in_address`);

설명

도접미사접두사௏유형을표준화도로도로명등을개별필드로분&#xnorm_addy합성유형을௏환합니다ન&#xb4e주소를정귞화된우편형식으로변환하기위한지오코딩과정의첫௏째단ૄ입니다지오코딩도함૨패키징된데이터외에다른데이터는필요하지않습니다

이함수는 tiger 스키마에 있ૠ tiger_geocoder와
함૨ 미리 로드돼 있는 여러 page_* 색이
테이블만 이용합니다. 따라서 이
함수를 이용하기 위해 TIGER 인구조이
데이터 또는 다른 추가적인 데이다
다운로드할 필요가 없습니다. tiger
스키마에 있는 여러 색인 테이블더
많은 약어 또는 대체 명칭들을
추가해야 할 필요가 있을 수도 있이
함수는 입력 주소를 정규화하위
해 tiger 스키마에 있는 여러 제어
색인 테이블을 이용합니다.

```
&#xc774;&#xd568;&#xc218;&#xb294; norm_addy &#xc720;&#xd615;&#xac1d;&#xccb4;&#xc548;&#xc758;&#xd544;&#xb4d
&#xb2e4;&#xc74c;&#xacfc;&#xac19;&#xc740;&#xc21c;&#xc11c;&#xb85c;&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;.
&#xc774;&#xb54c;)&#xac00;&#xc9c0;&#xc624;&#xcf54;&#xb529;&#xb3c4;&#xad6c;&#xac00;&#xc694;&#xad6c;&#xd558;&#x
&#xd544;&#xb4dc;&#xb97c;&#xb098;&#xd0c0;&#xb0b4;&#xace0;,&#xb294;&#xc120;&#xd0dd;&#xc801;&#xc778;&#xd544;&#x
&#xb098;&#xd0c0;&#xb0c5;&#xb2c8;&#xb2e4::
```

```

Normalize_Address &#xc5d0; &#xb300;&#xd574; &#xb300;&#xc18c; &#xbb38;&#xc790; &#xaddc;&#xce59; &#xbc0f; &#xc11c; &#x
&#xc0b4; &#xc9dd; &#xb2e4; &#xb978; &#xbcc0; &#xc885; &#xb4e4; &#xc774; &#xc788; &#xc2b5; &#xb2c8; &#xb2e4;.

```

2.1.0 버전부터 사용할 수 있습니다.



This method needs address standardizer extension.

```
(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip]
```

```
&#xc774;&#xc2dc;&#xc810;&#xc5d0;&#xc11c; address_standardizer &#xd655;&#xc7a5; &#xd504;&#xb85c;&#xadf8;&#xb7a8;&#xc6d0;&#xd615; standardaddr&#xb294; norm_addy&#xbcf4;&#xb2e4; &#xc57d;&#xac04; &#xb2e4;&#xcxc44;&#xb86d;&#xc2b5; (&#xad6d;&#xac00;&#xb97c; &#xd3ec;&#xd568;&#xd55c;) &#xad6d;&#xc81c; &#xc8fc;&#xc18c;&#xb97c; &#xc9c0;&#xc6d0;&#xc124;&#xacc4;&#xb410;&#xae30; &#xb54c;&#xbb38;&#xc785;&#xb2c8;&#xb2e4;. standardaddr&#xc758; &#xc0c1;&#xc751; &#xd544;&#xb4dc;&#xb4e4;&#xc740; &#xb2e4;&#xc74c;&#xacfc; &#xc19;&#xc2b5;&#xb2c8;&#xb2e4;:
```

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 는 정 수 형 입 니 다; 도 로 번 지 &#xc
2. predirAbbrev 는 varchar 형 입 니 다; N, S, E, W 등 과 같 은 도 로 의 방 향 을 의 미 하 는 접 두 &#xc0 direction_lookup 테 이 블 을 이 용 해 서 이 필 제 어 합 니 다.
3. streetName 은 varchar 형 입 니 다.
4. streetTypeAbbrev 는 varchar 형 으 로; St, Ave, Cir 처 럼 도 로 유 형 의 축 약 버 전 입 니 다. street_type_lookup 테 이 블 을 이 용 해 서 이 필 드 를 제 어 합 니 다.


```

        , '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

స고

Normalize Address, Geocode

14.2.15 Pprint_Addy

```
Pprint_Addy — norm_addy &#x0cf5;&#xd569;&#xc720;&#xd615;&#xac1d;&#xccb4;&#xb97c;&#xc785;&#xb825;&#xbc1b;&#&#xd574;&#xb2f9;&#xac1d;&#xccb4;&#xc758;&#x0cf4;&#xae30;&#xc88b;&#xc740;&#xc778;&#xc1c4;&#xc6a9;&#xd45c;&#&#xbc18;&#xd658;&#xd569;&#xb2c8;&#xb2e4;. &#xc77c;&#xbc18;&#xc801;&#xc73c;&#xb85c; normalize_address &#xd568;&#x&#xacb0;&#xd569;&#xd574;&#xc11c;&#xc4f0;&#xc785;&#xb2c8;&#xb2e4;.
```

Synopsis

```
varchar pprint_addy(norm_addy in_addy);
```

설명

```
norm_addy &#xbcf5;&#xd569;&#xc720;&#xd615;&#xac1d;&#xccb4;&#xb97c;&#xc785;&#xb825;&#xbc1b;&#xc544;,&#xd574;&#xb2f9;&#xac1d;&#xccb4;&#xc758;&#xbcf4;&#xae30;&#xc88b;&#xc740;&#xc778;&#xc1c4;&#xc6a9;&#xd45c;&#xbcb1;&#xb825;&#xd569;&#xb2c8;&#xb2e4;,&#xc9c0;&#xc624;&#xcf54;&#xb529;&#xb3c4;&#xad6c;&#xc640;&#xd568;&#xd328;&#xd0a4;&#xc9d5;&#xb41c;&#xb370;&#xc774;&#xd130;&#xc774;&#xc678;&#xc5d0;&#xb2e4;&#xb978;&#xb370;&#xbd88;&#xd544;&#xc694;&#xd569;&#xb2c8;&#xb2e4;.
```

```
&#xc77c;&#xbc18;&#xc801;&#xc73c;&#xb85c; Normalize_Address &#xd568;&#xc218;&#xc640; &#xacb0;&#xd569;&#xd574;&#xc4f0;&#xc785;&#xb2c8;&#xb2e4;.
```

예&#xc2dc:

단일 주소의 보기 좋은 인쇄용 출&#x

```
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
```

주소 테이블의 򼿄기 좋은 인쇄용
출력

```
SELECT address As orig, pprint_addy(normalize_address(address)) As pretty_address
FROM addresses_to_geocode;
```

orig	pretty_address
529 Main Street, Boston MA, 02129	529 Main St, Boston MA, 02129
77 Massachusetts Avenue, Cambridge, MA 02139	77 Massachusetts Ave, Cambridge, MA 02139 ↩
28 Capen Street, Medford, MA	28 Capen St, Medford, MA
124 Mount Auburn St, Cambridge, Massachusetts 02138	124 Mount Auburn St, Cambridge, MA 02138 ↩
950 Main Street, Worcester, MA 01610	950 Main St, Worcester, MA 01610

స고

Normalize_Address

14.2.16 Reverse_Geocode

Reverse_Geocode — 알려진 공간 참조 시스템으투영된 도형 포인트를 입력받아 이론적으로 가능한 주소들의 배교차로의 배열을 담고 있는 레코렘환합니다., include_strnum_range = true일 경우., 교차있는 도로 범위를 포함합니다.

Synopsis

```
record Reverse_Geocode(geometry pt, boolean include_strnum_range=false, geometry[] OUT intpt, norm_addy[] OUT addy,
varchar[] OUT street);
```

설명

알려진공간참조시스템으로투도형포인트를입력받아이론적가능한주소들의배열과교차로&#xcabc30;열을담고있는레코드를반환를일경우교차로에있는도맄위를포함합니다따로설정하않을경우의기본값은౰짓포인트가어떤도로에가장가까따라주소를정렬하기때문에ಫ번째주소가대부분맞는편입&#xb2c8
실제주소라고하는대신이론적하는이유가뭘까요. TIGER 데이터는실제주소를가지고있지않고. 다도로맄위만가지고있습니다. 그이론적주소란도로맄위에기&#bc18;보간된주소입니다. 예를들어어실제주소를보간하면 26 Court Sq. 라는주소가실재하지않는데도불구일 26 Court St. 와 26 Court Sq. 를&#bc18;환합니다. 왜냐하&#xbba

```

&#xd3ec;&#xc778;&#xd2b8;&#xac00; &#xb450; &#xb3c4;&#xb85c;&#xc758; &#xad50;&#xcc28;&#xc810;&#xc5d0; &#xc704;&#
&#xc218;&#xb3c4; &#xc788;&#xc73c;&#xbbc0;&#xb85c; &#xb85c;&#xc9c1;&#xc774; &#xb450; &#xb3c4;&#xb85c; &#xbaa8;&#
&#xb530;&#xb77c; &#xbcf4;&#xac04;&#xd558;&#xae30; &#xb54c;&#xbb38;&#xc785;&#xb2c8;&#xb2e4;. &#xc774; &#xb85c;&#
&#xb610; &#xc8fc;&#xc18c;&#xb4e4;&#xc774; &#xb3c4;&#xb85c;&#xb97c; &#xb530;&#xb77c; &#xade0;&#xb4f1;&#xd55c;
&#xac04;&#xaca9;&#xc73c;&#xb85c; &#xc704;&#xce58;&#xd55c;&#xb2e4;&#xace0; &#xac00;&#xc815;&#xd558;&#xb294;&#x
&#xbb3c;&#xb860; &#xc9c0;&#xc790;&#xccb4; &#xac74;&#xbb3c;&#xc774; &#xb3c4;&#xb85c; &#xb94;&#xc704;&#xc758;
&#xd070; &#xbd80;&#xbd84;&#xc744; &#xcc28;&#xc9c0;&#xd558;&#xace0; &#xb098;&#xba38;&#xc9c0; &#xac74;&#xbb3c;&#
&#xb3c4;&#xb85c; &#xb05d; &#xbd80;&#xbd84;&#xc5d0; &#xbab0;&#xb824; &#xc788;&#xc744; &#xc218;&#xb3c4; &#xc788;&
&#xb54c;&#xbb38;&#xc5d0; &#xc774; &#xac00;&#xc815;&#xc740; &#xd2c0;&#xb838;&#xc2b5;&#xb2c8;&#xb2e4;.

&#xc8fc;&#xc758;. &#xc774; &#xd568;&#xc218;&#xb294; TIGER &#xb370;&#xc774;&#xd130;&#xc5d0; &#xc758;&#xc874;&#x
&#xd574;&#xb2f9; &#xd3ec;&#xc778;&#xd2b8;&#xc758; &#xc9c0;&#xc5ed;&#xc744; &#xee4;&#bc84;&#xd558;&#xb294;
&#xb370;&#xc774;&#xd130;&#xb97c; &#xb85c;&#xb4dc;&#xd558;&#xc9c0; &#xc54a;&#xc558;&#xc744; &#xacbd;&#xc6b0;,
NULL&#xb85c; &#xcc44;&#xc6cc;&#xc9c4; &#xb808;&#xcf54;&#xb4dc;&#xb97c; &#xbc18;&#xd658;&#xd560; &#xac83;&#xc7
&#xbc18;&#xd658;&#xb41c; &#xb808;&#xcf54;&#xb4dc;&#xb97c; &#xad6c;&#xc131;&#xd558;&#xb294; &#xc694;&#xc18c;&#x
&#xb2e4;&#xc74c;&#xacfc; &#xac19;&#xc2b5;&#xb2c8;&#xb2e4;.

```

1. `intpt 은 포인트 배열입니다. 입력 포인트에 가장 가까운 도로 상&#xc75 중심선 포인트들입니다. 주소&#xc75 개수 만큼 많은 포인트들이 있&#xc21`
2. `addy 는 norm_addy(정귞화된 주소) 배열입&#xb2c8 입력 포인트에 적합한 가능한 주소들의 배열입니다. 배열의 첫 번째 주소가 가장 올바를 가&#xb2a 큽니다. 일반적으로 포인트가 도로 2개 또는 3개의 교차점에 위&#x 경우 또는 포인트가 도로의 어&#xb2 한 편이 아니라 도로 상에 위치&#xd5 경우를 제외하면 주소 1개만 있&#xc5 합니다.`
3. `street 는 varchar형 배열입니다. 교차로의 도로들(또는 도로 1개) 입니다(교&#x 도로들 또는 포인트가 그 위에 투영된 도로입니다).`

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 [Loader_Generate_Nation_Script](#) for details on loading `zcta5` data.

2.0.0 버전부터 사용할 수 있습니다.

예시

```

&#xd3ec;&#xc778;&#xd2b8;&#xac00; &#xb450; &#xb3c4;&#xb85c;&#xc758; &#xad50;&#xcc28;&#xc810;&#xc5d0; &#xc788;&#x
&#xc5b4;&#xb290; &#xd55c; &#xb3c4;&#xb85c;&#xc5d0; &#xb354; &#xac00;&#xae4c;&#xc6b4; &#xacbd;&#xc6b0;&#xc758;
&#xc608;&#xc2dc;&#xc785;&#xb2c8;&#xb2e4;. &#xc774; &#xd3ec;&#xc778;&#xd2b8;&#xb294; MIT - 77 Massachusetts
Ave, Cambridge, MA 02139 - &#xc5d0; &#xb9e4;&#xc6b0; &#xac00;&#xae4c;&#xc6b4; &#xc704;&#xce58;&#xc5d0; &#xc788;&#x
&#xb3c4;&#xb85c; 3&#xac1c;&#xc758; &#xacbd;&#xc6b0;&#xb294; &#xc544;&#xb2c8;&#xc9c0;&#xb9cc;, PostgreSQL&#xc774
&#xc0c1;&#xd55c;(&#x4e0a;&#x9650;; upper bound)&#xc744; &#xb118;&#xc5b4;&#xc11c;&#xb294; &#xd56d;&#xbaa9;&#xc5d0
&#xb300;&#xd574; NULL&#xc744; &#xbc18;&#xd658;&#xd560; &#xac83;&#xc774;&#xae30; &#xb54c;&#xbb38;&#xc5d0;
&#xc548;&#xc804;&#xd558;&#xac8c; &#xc774;&#xc6a9;&#xd560; &#xc218; &#xc788;&#xb2e4;&#xb294; &#xc810;&#xc5d0;
&#xc8fc;&#xc758;&#xd558;&#xc2ed;&#xc2dc;&#xc624;. &#xb3c4;&#xb85c; &#xb94;&#xc704;&#xb3c4; &#xd3ec;&#xd568;&#x

```

```

SELECT pprint_addy(r.addy[1]) As st1, pprint_addy(r.addy[2]) As st2, pprint_addy(r.addy[3]) ←
  As st3,
  array_to_string(r.street, ',') As cross_streets

```



```
FROM reverse_geocode(ST_GeomFromText('POINT(-71.093902 42.359446)',4269),true) As r
```

```
result
```

st1	st2	st3	cross_streets
67 Massachusetts Ave, Cambridge, MA 02139			67 - 127 Massachusetts Ave, 32 - 88 Vassar St

```
&#xb2e4;&#xc74c;&#xc608;&#xc2dc;&#xb294;&#xad50;&#xcc28;&#xb85c;&#xc5d0;&#xb300;&#xd574;&#xc8fc;&#xc18c;&#xb9c94;&#xc704;&#xb97c;&#xd3ec;&#xd568;&#xd558;&#xc9c0;&#xc54a;&#xb3c4;&#xb85d;&#xc120;&#xd0dd;&#xd558;&#xb450;&#xb3c4;&#xb85c;&#xc758;&#xad50;&#xcc28;&#xc810;&#xc5d0;&#xc815;&#xb9d0;&#xb85c;&#xac00;&#xae4c;&#xc704;&#xce58;&#xb97c;&#xace8;&#xb77c;&#xc11c;&#xc11c;&#xb85c;&#xb2e4;&#xb978;&#xb450;&#xc8fc;&#xc18c;&#xb9c18;&#xd658;&#xd560;&#xc218;&#xc788;&#xb3c4;&#xb85d;&#xd569;&#xb2c8;&#xb2e4;.
```

```
SELECT pprint_addy(r.addy[1]) As st1, pprint_addy(r.addy[2]) As st2,
pprint_addy(r.addy[3]) As st3, array_to_string(r.street, ',') As cross_str
FROM reverse_geocode(ST_GeomFromText('POINT(-71.06941 42.34225)',4269)) As r;
```

```
result
```

st1	st2	st3	cross_str
5 Bradford St, Boston, MA 02118	49 Waltham St, Boston, MA 02118		Waltham St

```
&#xb2e4;&#xc74c;&#xc608;&#xc2dc;&#xb294;Geocode&#xc758;&#xc9c0;&#xc624;&#xcf54;&#xb529;&#xb41c;&#xc608;&#xb2e4;&#xc2dc;&#xc774;&#xc6a9;&#xd574;&#xc11c;&#xb3c4;&#xb85c;&#xac00;&#xcd5c;&#xb300;&#xd55c;2&#xac1c;&#xad50;&#xcc28;&#xd558;&#xb294;&#xad50;&#xcc28;&#xb85c;&#xc5d0;&#xc788;&#xb294;&#xcd5c;&#xc120;&#xc758;&#xc8fc;&#xc18c;&#xb9cc;&#xc744;&#xb9c18;&#xd658;&#xd558;&#xb3c4;&#xb85d;&#xd569;&#xb2c8;&#xb2e4;.
```

```
SELECT actual_addr, lon, lat, pprint_addy((rg).addy[1]) As int_addr1,
(rg).street[1] As cross1, (rg).street[2] As cross2
FROM (SELECT address As actual_addr, lon, lat,
reverse_geocode(ST_SetSRID(ST_Point(lon,lat),4326)) As rg
FROM addresses_to_geocode WHERE rating
> -1) As foo;
```

actual_addr	int_addr1	lon	lat	cross1	cross2
529 Main Street, Boston MA, 02129		-71.07181	42.38359	527 Main St,	
Boston, MA 02129	Medford St				
77 Massachusetts Avenue, Cambridge, MA 02139		-71.09428	42.35988	77	
Massachusetts Ave, Cambridge, MA 02139	Vassar St				
26 Capen Street, Medford, MA		-71.12377	42.41101	9 Edison Ave,	
Medford, MA 02155	Capen St			Tesla Ave	
124 Mount Auburn St, Cambridge, Massachusetts 02138		-71.12304	42.37328	3 University	
Rd, Cambridge, MA 02138	Mount Auburn St				
950 Main Street, Worcester, MA 01610		-71.82368	42.24956	3 Maywood St,	
Worcester, MA 01603	Main St			Maywood Pl	

```
&#xcc38;&#xace0;
```

```
Pprint_Addy, Pprint_Addy, ST_AsText
```


필수 인수;

1. topo_name - 데이터를 로드할 기존 PostGIS 지로칭입니다.
2. region_type - 경계를 이루는 지역의 유형현재 place 와 county 만 지원합니다. 몇몇유형을 더 지원할 예정입니다. 이 인수는 tiger.place, tiger.county 처럼 지역 경정의하려면 살펴봐야 할 테이로입니다.
3. region_id - TIGER가 지리 ID(geoid)라고 부르는 식테이&#xbel4;에 있는 지역의 유일한 식별자입니다. place 의 경우 tiger.place 테있는 plcidfp 열입니다. county 의 경우 tiger.county 테이&#xbel4;에 있는 cntyidfp 열입니다.

2.0.0 버전부터 사용할 수 있습니다.

예시: 매사추세츠 주 보스턴 시의 지형

매사추세츠 주 피트 단위 평면(2249)었매사추세츠 주 보스턴 시에 대해 허용 오차 0.25 피트를 가진 지형을 생성한 다음 보스턴 시의 TIGER 면 경노드를 로드합니다.

```
SELECT topology.CreateTopology('topo_boston', 2249, 0.25);
createtopology
-----
15
-- &#xc708;&#xb3c4;&#xc6b0;7 &#xb370;&#xc2a4;&#xd06c;&#xd0d1;&#xc5d0;&#xc11c; 9.1 ↔
&#xbc84;&#xc804;&#xc744; (5&#xac1c; &#xc8fc;&#xc758; TIGER ↔
&#xb370;&#xc774;&#xd130;&#xb97c; &#xb85c;&#xb4dc;&#xd55c; &#xc0c1;&#xd0dc;&#xb85c;) ↔
&#xc2e4;&#xd589;&#xd588;&#xc744; &#xb54c; 60,902&#xbc00;&#xb9ac;&#xcd08; ~ 1&#xbd84; ↔
&#xc18c;&#xc694;
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&#xbc00;&#xb9ac;&#xcd08; &#xc18c;&#xc694; --
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

-- &#xd655;&#xc778; &#xc791;&#xc5c5;&#xc5d0; 28,797&#xbc00;&#xb9ac;&#xcd08; ↔
&#xc18c;&#xc694;, &#xbclc;&#xc0dd;&#xd55c; &#xc624;&#xb958; &#xc5c6;&#xc74c; --
SELECT * FROM
topology.ValidateTopology('topo_boston');

error      | id1      | id2
-----+-----+-----
```

예시: 매사추세츠 주 서퍽 군의 지엤사추세츠 주 미터 단위 평면(26986)엤사추세츠 주 서퍽 군에 대해 허오차 0.25 &#bbf8;터를 가진 지형을 생성핐 다음, 서퍽 군의 TIGER 면, 경계선, 노드로드합니다.

```

SELECT topology.CreateTopology('topo_suffolk', 26986, 0.25);
-- &#xc708;&#xb3c4;&#xc6b0;7 32&#xbe44;&#xd2b8; &#xc0c1;&#xc5d0;&#xc11c; 5&#xac1c; <-
&#xc8fc;&#xc758; TIGER &#xb370;&#xc774;&#xd130;&#xb97c; &#xb85c;&#xb4dc;&#xd55c; <-
&#xc0c1;&#xd0dc;&#xb85c; &#xc2e4;&#xd589;&#xd588;&#xc744; &#xb54c; 56,275 <-
&#xbc00;&#xb9ac;&#xcd08; ~ 1&#xbd84; &#xc18c;&#xc694;
-- &#xbcf4;&#xc2a4;&#xd134; &#xb370;&#xc774;&#xd130;&#xb97c; &#xb85c;&#xb4dc;&#xd55c; <-
&#xb2e4;&#xc74c; &#xc608;&#xc5f4;&#xb410;&#xc74c;&#xc774; &#xbd84;&#xba85;&#xd568;
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&#xbc00;&#xb9ac;&#xcd08; &#xc18c;&#xc694; --
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

-- &#xd655;&#xc778; &#xc791;&#xc5c5;&#xc5d0; 33,606&#xbc00;&#xb9ac;&#xcd08; <-
&#xc18c;&#xc694; --
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

```

స고

CreateTopology, CreateTopoGeom, TopologySummary, ValidateTopology

14.2.18 Set Geocode Setting

Set_Geocode_Setting — 지오코딩 도구 함수들의
습성에영향을미치는 설정값을
설정합니다.

Synopsis

```
text Set_Geocode_Setting(text setting_name, text setting_value);
```

Get_Geocode_Setting

`tiger.geocode_settings` table. The `debug_geocode_address` setting is used to enable debugging of the geocoding process. The `debug_geocode_address` setting is a boolean value that can be set to `true` or `false`. The `debug_geocode_address` setting is used to enable debugging of the geocoding process. The `debug_geocode_address` setting is a boolean value that can be set to `true` or `false`.

2.1.0 The `debug_geocode_address` setting is used to enable debugging of the geocoding process. The `debug_geocode_address` setting is a boolean value that can be set to `true` or `false`.

Geocode table. The `debug_geocode_address` setting is used to enable debugging of the geocoding process. The `debug_geocode_address` setting is a boolean value that can be set to `true` or `false`.

The `Geocode` table is used to store the results of the geocoding process. The `Geocode` table has columns for the address, the geocode, and the score. The `Geocode` table is used to store the results of the geocoding process. The `Geocode` table has columns for the address, the geocode, and the score.

```
SELECT set_geocode_setting('debug_geocode_address', 'true') As result;
result
-----
true
```

Geocode

Get_Geocode_Setting

Chapter 15

PostGIS Special Functions Index

15.1 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** - Aggregate function that returns the 3D bounding box of geometries.
- **ST_3DUnion** - Perform 3D union.
- **ST_AsFlatGeobuf** - Return a FlatGeobuf representation of a set of rows.
- **ST_AsGeobuf** - Return a Geobuf representation of a set of rows.
- **ST_AsMVT** - Aggregate function returning a MVT representation of a set of rows.
- **ST_ClusterIntersecting** - Aggregate function that clusters the input geometries into connected sets.
- **ST_ClusterWithin** - Aggregate function that clusters the input geometries by separation distance.
- **ST_Extent** - Aggregate function that returns the bounding box of geometries.
- **ST_GeomCollFromText** - Creates a GeometryCollection or Multi* geometry from a set of geometries.
- **ST_MakeLine** - Returns a line from a set of points.
- **ST_MemUnion** - Aggregate function which unions geometries in a memory-efficient but slower way.
- **ST_Polygonize** - Computes a collection of polygons formed from the linework of a set of geometries.
- **ST_SameAlignment** - Returns true if two geometries have the same alignment.
- **ST_Union** - Computes a geometry representing the point-set union of the input geometries.
- **TopoElementArray_Agg** - Returns a topoelementarray for a set of element_id, type arrays (topoelements).

15.2 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** - Window function that returns a cluster id for each input geometry using the DBSCAN algorithm.
- **ST_ClusterKMeans** - Window function that returns a cluster id for each input geometry using the K-means algorithm.

15.3 PostGIS SQL-MM Compliant Functions

The functions given below are PostGIS functions that conform to the SQL/MM 3 standard

- **ST_3DArea** - 3D area of a geometry. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 8.1, 10.5
- **ST_3DDWithin** - Tests if two 3D geometries are within a given 3D distance. This method implements the SQL/MM specification. SQL-MM ?
- **ST_3DDifference** - 3D difference of two geometries. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1
- **ST_3DDistance** - 3D distance between two geometries. This method implements the SQL/MM specification. SQL-MM ISO/IEC 13249-3
- **ST_3DIntersection** - 3D intersection of two geometries. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1
- **ST_3DIntersects** - Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area). This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1
- **ST_3DLength** - 3D length of a geometry. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 7.1, 10.3
- **ST_3DPerimeter** - 3D perimeter of a geometry. This method implements the SQL/MM specification. SQL-MM ISO/IEC 13249-3: 8.1, 10.5
- **ST_3DUnion** - Perform 3D union. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1
- **ST_AddEdgeModFace** - Add a new edge to a face. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.13
- **ST_AddEdgeNewFaces** - Add a new edge to a face. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.12
- **ST_AddIsoEdge** - Add an isolated edge to a face. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.4

- **ST_AddIsoNode** - Returns a geometry with the specified node added to the geometry. This method implements the SQL/MM specification. SQL-MM: Topo-Net Routines: X+1.3.1
- **ST_Area** - Returns the area of the geometry. This method implements the SQL/MM specification. SQL-MM 3: 8.1.2, 9.5.3
- **ST_AsBinary** - Return the OGC/ISO Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data. This method implements the SQL/MM specification. SQL-MM 3: 5.1.37
- **ST_AsGML** - Returns the GML 2 or GML 3 representation of the geometry/geography. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 17.2
- **ST_AsText** - Returns the Well-Known Text (WKT) representation of the geometry/geography. This method implements the SQL/MM specification. SQL-MM 3: 5.1.25
- **ST_Boundary** - Returns the boundary of the geometry. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.17
- **ST_Buffer** - Computes a geometry covering all points within a given distance from a geometry. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.30
- **ST_Centroid** - Returns the centroid of the geometry. This method implements the SQL/MM specification. SQL-MM 3: 8.1.4, 9.5.5
- **ST_ChangeEdgeGeom** - Returns a geometry with the specified edge replaced by the specified geometry. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details X.3.6
- **ST_Contains** - Tests if no points of B lie in the exterior of A, and A and B have at least one interior point in common. This method implements the SQL/MM specification. SQL-MM 3: 5.1.31
- **ST_ConvexHull** - Computes the convex hull of a geometry. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.16
- **ST_CoordDim** - Returns the dimension of the geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.3
- **ST_CreateTopoGeo** - Returns a topology from a geometry. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details -- X.3.18
- **ST_Crosses** - Tests if two geometries have some, but not all, interior points in common. This method implements the SQL/MM specification. SQL-MM 3: 5.1.29
- **ST_CurveToLine** - Converts a geometry containing curves to a linear geometry. This method implements the SQL/MM specification. SQL-MM 3: 7.1.7
- **ST_Difference** - Computes a geometry representing the part of geometry A that does not intersect geometry B. This method implements the SQL/MM specification. SQL-MM 3: 5.1.20
- **ST_Dimension** - Returns the dimension of the geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.2
- **ST_Disjoint** - Tests if two geometries are disjoint (they have no point in common). This method implements the SQL/MM specification. SQL-MM 3: 5.1.26

- **ST_Distance** - Returns the distance between two geometries in SRID units. If both arguments are `POINT` values, the distance between the two points is returned. If one argument is a `POINT` value and the other is a `LINESTRING` value, the shortest distance between the point and the line is returned. If both arguments are `LINESTRING` values, the shortest distance between the two lines is returned. If one argument is a `POINT` value and the other is a `POLYGON` value, the shortest distance between the point and the polygon is returned. If both arguments are `POLYGON` values, the shortest distance between the two polygons is returned. This method implements the SQL/MM specification. SQL-MM 3: 5.1.23
- **ST_EndPoint** - Returns the end point of a `LINESTRING` geometry. This method implements the SQL/MM specification. SQL-MM 3: 7.1.4
- **ST_Envelope** - Returns the envelope (bounding box) of the input geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.19
- **ST_Equals** - Tests if two geometries include the same set of points. This method implements the SQL/MM specification. SQL-MM 3: 5.1.24
- **ST_ExteriorRing** - Returns the exterior ring of a `POLYGON` geometry. This method implements the SQL/MM specification. SQL-MM 3: 8.2.3, 8.3.3
- **ST_GMLToSQL** - Converts a `GML` geometry to a `SQL` geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.50
- **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** - WKT to Geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.40
- **ST_GeomFromWKB** - WKB(Well-Known Binary) to Geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.41
- **ST_GeometryFromText** - WKT(Well-Known Text) to Geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.40
- **ST_GeometryN** - Returns the `N`th geometry of a `GEOMETRYCOLLECTION`. This method implements the SQL/MM specification. SQL-MM 3: 9.1.5
- **ST_GeometryType** - Returns the geometry type of a geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.4
- **ST_GetFaceEdges** - Returns the edges of a face. This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.5
- **ST_GetFaceGeometry** - Returns the geometry of a face. This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.16
- **ST_InitTopoGeo** - Initializes a `TopoGeo` object. This method implements the SQL/MM specification. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.17
- **ST_InteriorRingN** - Returns the `N`th interior ring of a `POLYGON` geometry. This method implements the SQL/MM specification. SQL-MM 3: 8.2.6, 8.3.5

- **ST_Intersection** - Computes a geometry representing the shared portion of geometries A and B. This method implements the SQL/MM specification. SQL-MM 3: 5.1.18
- **ST_Intersects** - Tests if two geometries intersect (they have at least one point in common). This method implements the SQL/MM specification. SQL-MM 3: 5.1.27
- **ST_IsClosed** - LINESTRING. This method implements the SQL/MM specification. SQL-MM 3: 7.1.5, 9.3.3
- **ST_IsEmpty** - Tests if a geometry is empty. This method implements the SQL/MM specification. SQL-MM 3: 5.1.7
- **ST_IsRing** - Tests if a LineString is closed and simple. This method implements the SQL/MM specification. SQL-MM 3: 7.1.6
- **ST_IsSimple** - This method implements the SQL/MM specification. SQL-MM 3: 5.1.8
- **ST_IsValid** - Tests if a geometry is well-formed in 2D. This method implements the SQL/MM specification. SQL-MM 3: 5.1.9
- **ST_Length** - This method implements the SQL/MM specification. SQL-MM 3: 7.1.2, 9.3.4
- **ST_LineFromText** - This method implements the SQL/MM specification. SQL-MM 3: 7.2.8
- **ST_LineFromWKB** - This method implements the SQL/MM specification. SQL-MM 3: 7.2.9
- **ST_LinestringFromWKB** - This method implements the SQL/MM specification. SQL-MM 3: 7.2.9
- **ST_LocateAlong** - Returns the point(s) on a geometry that match a measure value. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1.13
- **ST_LocateBetween** - Returns the portions of a geometry that match a measure range. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 5.1
- **ST_M** - Returns the M coordinate of a Point. This method implements the SQL/MM specification.
- **ST_MLineFromText** - 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** - Heals two edges by deleting the node connecting them, modifying the first edge and deleting the second edge. Returns the id of the deleted node. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9

- **ST_ModEdgeSplit** - Moves an edge 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-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** - Heals two edges by deleting the node connecting them, deleting both edges, and replacing them with an edge whose direction is the same as the first edge provided. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.9
- **ST_NewEdgesSplit** - Moves an edge 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.3.8
- **ST_NumGeometries** - Returns the number of geometries in the set. This method implements the SQL/MM specification. SQL-MM 3: 9.1.4
- **ST_NumInteriorRings** - Returns the number of interior rings in the set. This method implements the SQL/MM specification. SQL-MM 3: 8.2.5
- **ST_NumPatches** - Returns the number of patches in the set. This method implements the SQL/MM specification. SQL-MM ISO/IEC 13249-3: 8.5
- **ST_NumPoints** - Returns the number of points in the set. This method implements the SQL/MM specification. SQL-MM 3: 7.2.4
- **ST_OrderingEquals** - Tests if two geometries represent the same geometry and have points in the same directional order. This method implements the SQL/MM specification. SQL-MM 3: 5.1.43
- **ST_Overlaps** - Tests if two geometries intersect and have the same dimension, but are not completely contained by each other. This method implements the SQL/MM specification. SQL-MM 3: 5.1.32
- **ST_PatchN** - Returns the Nth patch in the set. This method implements the SQL/MM specification. SQL-MM ISO/IEC 13249-3: 8.5
- **ST_Perimeter** - Returns the length of the boundary of a polygonal geometry or geography. This method implements the SQL/MM specification. SQL-MM 3: 8.1.3, 9.5.4
- **ST_Point** - Creates a Point with X, Y and SRID values. This method implements the SQL/MM specification. SQL-MM 3: 6.1.2
- **ST_PointFromText** - Creates a Point from a WKT string. This method implements the SQL/MM specification. SQL-MM 3: 6.1.8
- **ST_PointFromWKB** - Creates a Point from a WKB string. This method implements the SQL/MM specification. SQL-MM 3: 6.1.9

- **ST_PointN** - ST_LineString 또는 ST_CircularString 값에 있는 포인개수를 반환합니다. This method implements the SQL/MM specification. SQL-MM 3: 7.2.5, 7.3.5
- **ST_PointOnSurface** - Computes a point guaranteed to lie in a polygon, or on a geometry. This method implements the SQL/MM specification. SQL-MM 3: 8.1.5, 9.5.6. The specifications define ST_PointOnSurface for surface geometries only. PostGIS extends the function to support all common geometry types. Other databases (Oracle, DB2, ArcSDE) seem to support this function only for surfaces. SQL Server 2008 supports all common geometry types.
- **ST_Polygon** - Creates a Polygon from a LineString with a specified SRID. 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** - Tests if two geometries have a topological relationship matching an Intersection Matrix pattern, or computes their Intersection Matrix This method implements the SQL/MM specification. SQL-MM 3: 5.1.25
- **ST_RemEdgeModFace** - 경계선을 제거하고, 제거하경계선이 두 표면을 분할하고 있경우, 표면 하나를 삭제하고 다른 표면을 두 표면의 공간을 차지하수정합니다. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.15
- **ST_RemEdgeNewFace** - 경계선을 제거하고, 제거하경계선이 두 표면을 분할하고 있경우, 원본 표면들을 삭제하고 새 표면 하나로 대체합니다. 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** - 고립된 노드를 제거하고 작업 내용을 설명하는 메시지를 반환합니다. 노드가 고립되지 않 (경계선의 시작점이나 종단점인) 경우, 예외가 발생합니다. This method implements the SQL/MM specification. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X+1.3.3
- **ST_SRID** - Returns the spatial reference identifier for a geometry. This method implements the SQL/MM specification. SQL-MM 3: 5.1.5
- **ST_StartPoint** - Returns the first point of a LineString. This method implements the SQL/MM specification. SQL-MM 3: 7.1.3
- **ST_SymDifference** - Computes a geometry representing the portions of geometries A and B that do not intersect. This method implements the SQL/MM specification. SQL-MM 3: 5.1.21
- **ST_Touches** - Tests if two geometries have at least one point in common, but their interiors do not intersect. This method implements the SQL/MM specification. SQL-MM 3: 5.1.28
- **ST_Transform** - Return a new geometry with coordinates transformed to a different spatial reference system. This method implements the SQL/MM specification. SQL-MM 3: 5.1.6
- **ST_Union** - Computes a geometry representing the point-set union of the input geometries. This method implements the SQL/MM specification. SQL-MM 3: 5.1.19 the z-index (elevation) when polygons are involved.
- **ST_Volume** - 3차원 입체의 부피를 계산합&#xb2 표면 도형을 입력하면 (닫힌 도&#xd615 0을 반환할 것입니다. This method implements the SQL/MM specification. SQL-MM IEC 13249-3: 9.1 (same as ST_3DVolume)

- **ST_WKBToSQL** - WKB(Well-Known Binary) ST_Geometry SRID; ST_GeomFromWKB; This method implements the SQL/MM specification. SQL-MM 3: 5.1.36
- **ST_WKTToSQL** - WKT(Well-Known Text) ST_Geometry ST_GeomFromText; This method implements the SQL/MM specification. SQL-MM 3: 5.1.34
- **ST_Within** - Tests if no points of A lie in the exterior of B, and A and B have at least one interior point in common. This method implements the SQL/MM specification. SQL-MM 3: 5.1.30
- **ST_X** - Returns the X coordinate of a Point. This method implements the SQL/MM specification. SQL-MM 3: 6.1.3
- **ST_Y** - Returns the Y coordinate of a Point. This method implements the SQL/MM specification. SQL-MM 3: 6.1.4
- **ST_Z** - Returns the Z coordinate of a Point. This method implements the SQL/MM specification.
- **TG_ST_SRID** - Returns the spatial reference identifier for a topogeometry. This method implements the SQL/MM specification. SQL-MM 3: 14.1.5

15.4 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** - Returns the area of a geography.
- **ST_AsBinary** - Return the OGC/ISO Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.
- **ST_AsEWKT** - Returns the Well-Known Text (WKT) representation of the geometry/geography without SRID meta data.
- **ST_AsGML** - Returns the GML 2 or GML 3 representation of the geometry/geography.
- **ST_AsGeoJSON** - Return a geometry as a GeoJSON element.
- **ST_AsKML** - Returns the KML representation of the geometry/geography.
- **ST_AsSVG** - Returns SVG path data for a geometry.
- **ST_AsText** - Returns the Well-Known Text (WKT) representation of the geometry/geography.
- **ST_Azimuth** - Returns the azimuth of the line segment from point 1 to point 2.
- **ST_Buffer** - Computes a geometry covering all points within a given distance from a geometry.
- **ST_Centroid** - Returns the centroid of a geometry.

- **ST_CoveredBy** - Tests if no point in A is outside B
- **ST_Covers** - Tests if no point in B is outside A
- **ST_DWithin** - Tests if two geometries are within a given distance
- **ST_GeogFromText** - WKT ('확장') 표현식으로부터 지리형 값을 반환합니다.
- **ST_GeogFromWKB** - WKB 도형 표현식 또는 EWKB(확장 WKB)로부터 지리형 인스턴스를 생다.
- **ST_GeographyFromText** - WKT ('확장') 표현식으로부터 지정된 지리형 값을 반환합니다.
- **=** - 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** - Computes a geometry representing the shared portion of geometries A and B.
- **ST_Intersects** - Tests if two geometries intersect (they have at least one point in common).
- **ST_Length** - 도형의 기하학적 중심을 반환합니다.
- **ST_Perimeter** - Returns the length of the boundary of a polygonal geometry or geography.
- **ST_Project** - 시작점으로부터 미터 단위 거리 및 라디안 단위 방향(방위각)이용해 투영된 POINT 를 반환합니다.
- **ST_Segmentize** - 주어진 거리보다 더 긴 구간없도록 수정된 도형/지리형을 반환합니다.
- **ST_Summary** - 도형의 내용을 요약한 텍스환합니다.
- **<->** - A와 B 사이의 2차원 거리를 반환합니다.
- **&&** - A의 2D 경계 상자와 B의 2D 경계 상자와 교차하는 경우 TRUE 를 반환합니다.

15.5 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.

- **Box3D** - 래스터를 둘러싼 상자의 BOX3D 표ଘ환합니다.
- **@** - B의 경계 상자가 A의 경계 상자를 담고 있을 경우 TRUE 를 ଘ환합니다. 이중 정밀도 경계 상자를 이용합니다.
- **~** - A의 경계 상자가 B의 경계 상자를 담고 있을 경우 TRUE 를 ଘ환합니다. 이중 정밀도 경계 상자를 이용합니다.
- **=** - A의 경계 상자와 B의 경계 상자가 동일할 경우 TRUE 를 ଘ환합니다. 이중 정밀도 경계 상자를 이용합니다.

- **&&** - A 의 경 계 상 자 와 B 의 경 계 상 자 가 교 차 하 는 경 우 TRUE 를 반 환 합 니 다.
- **&<** - A 의 경 계 상 자 가 B 의 경 계 상 자 왼 있 을 경 우 TRUE 를 반 환 합 니 다.
- **&>** - A 의 경 계 상 자 가 B 의 경 계 상 자 오 있 을 경 우 TRUE 를 반 환 합 니 다.
- **~=-** - A 의 경 계 상 자 와 B 의 경 계 상 자 가 동 일 할 경 우 TRUE 를 반 환 합 니 다.
- **ST_Retile** - 임 의 로 타 일 화 된 래 스 터 커 버 설 정 된 타 일 들 의 집 합 을 반 환 합 니 다.
- **ST_AddBand** - 입 력 인 덱 스 위 치 에 입 력 초 ꫀ 입 력 유 형 의 새 밴 드 (들) 을 가 진 래 스 터 를 반 환 합 니 다 인 덱 설 정 하 지 않 을 경 우 마 지 막 위 치 밴 드 를 추 가 합 니 다.
- **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** - 래 스 터 타 일 에 서 선 택 한 밴 드 단 일 JPEG (Joint Photographic Experts Group) 이 미 지 (바 이 트 배 반 환 합 니 다 밴 드 를 따 로 설 정 하 않 거 나 밴 드 가 1 개 거 나 또 는 3 개 를 초 과 할 경 우 첫 번 째 &#bc34; 드 를 씁 니 &#bc34; 드 가 3 개 뿐 일 경 우 &#bc34; 드 3 개 를 모 두 써 서 RGB 에 매 핑 시 킵 니 다.
- **ST_AsPNG** - 래 스 터 타 일 에 서 선 택 한 &#bc34; 드 단 일 PNG (Portable Network Graphics) 이 미 지 (&#bc14; 이 트 &#bc30; &#bc18; 환 합 니 다 &#bc34; 스 터 의 &#bc34; 드 가 1 개 거 나 따 로 설 정 하 지 않 을 경 우 &#baa8; 든 &#bc34; 드 를 씁 니 다 &#bc34; 드 가 3 개 뿐 일 경 우 &#bc34; 드 3 개 를 모 두 써 서 RGB 에 매 핑 시 킵 니 다.
- **ST_AsRaster** - PostGIS 도 형 을 PostGIS 래 스 터 로 변 환 합 니 다.
- **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** - 표 ૠ 래 스 터 &#bc34; 드 의 향 (기 본 단 위 는 도) 을 &#bc18; 환 합 니 다 지 형 을 분 석 하 는 데 유 용 합 니 다.
- **ST_Band** - 기 존 래 스 터 의 하 나 이 상 의 &#bc34; 드 를 새 래 스 터 로 &#bc18; 환 합 니 다 기 존 래 스 터 로 부 터 새 래 스 터 를 &#bbe4; 드 하 는 데 유 용 합 니 다.
- **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** - `썌드가가값만으로ౄ워져있을경우స을ଘ환합니다`.
- ST_BandMetaData** - `지정래스터썌드에대한기본메타데이터를ଘ환합니다썌드를지정하지않을경우썌드ஈ이라고가정합니다`.
- ST_BandNoDataValue** - `입력썌드에서타나타나ఒ을ଘ환합니다어떤썌드도지않을경우썌드로가정합니다`.
- ST_BandPath** - `파일시스템에저장된썌드리키는시스템파일경로를ଘ환합니다썌드를지정하지않을경우썌드로가정합니다`.
- ST_BandPixelType** - `입력썌드의픽셀유형을ଘ환합니다을설정하지않을경우썌드로가정합니다`.
- ST_Clip** - `입력도형으로잘라낸래스터ଘ환합니다썌드ஈ호를지정하않은경우모든썌드를ಘ리합니정하지않౰나స으로설정하지않을경우잘라낸래스터를ග력합니다`.
- ST_ColorMap** - `소스래스터ఏ설정한썌드&#xb8bui;썌드을설정한지않을경우స을ଘ환합니다`.
- ST_Contains** - `래스터외부에놓인래스터의포인트가하나도없고적어내부의포인트하나가의내부있는경우స을ଘ환합니다`.
- ST_ContainsProperly** - `rastB;가의내부와교న하지리의경계선또는외부와교న하지않을경우స을ଘ환합니다`.
- ST_Count** - Generates a set of vector contours from the provided raster band, using the GDAL contouring algorithm.
- ST_ConvexHull** - `BandNoDataValue;와일ౘ하는픽셀ఒ을포함한래스터의௏록껭질도형ଘ환합니다정귞형상௏기울어않은래스터의경우을설정하지않을경우స을ଘ환합니다`.
- ST_Count** - `래스터또는래스터쳤஄리지입력썌드에있는픽셀జ수를ଘ썌드를따로설정하지않을경우기본ఒ은썌드입니다స으로설정할경우ఒ이않닌픽셀의జ수만ଘ환할ಃ입니다`.

- **ST_CountAgg** - `<geomval>(<geom>, <val>)` - Returns the count of the number of features in the geometry column that have a value in the specified column. The geometry column is the first argument, and the column name is the second argument. The function returns a numeric value. `exclude_nodata_value` is a boolean parameter that, when set to true, excludes NULL values from the count. `NODATA` is a keyword that, when used, returns NULL if the count is zero. `<geomval>` is the name of the column to be counted, and `<geom>` is the name of the geometry column.
- **ST_CoveredBy** - `ST_CoveredBy(geom1, geom2)` - Returns true if the geometry `geom1` is completely covered by the geometry `geom2`. Both arguments must be of the same SRID.
- **ST_Covers** - `ST_Covers(geom1, geom2)` - Returns true if the geometry `geom1` completely covers the geometry `geom2`. Both arguments must be of the same SRID.
- **ST_DFullyWithin** - `ST_DFullyWithin(geom1, geom2, dist)` - Returns true if the geometry `geom1` is fully within the geometry `geom2` at a distance of `dist` or less. Both arguments must be of the same SRID.
- **ST_DWithin** - `ST_DWithin(geom1, geom2, dist)` - Returns true if the geometry `geom1` is within the geometry `geom2` at a distance of `dist` or less. Both arguments must be of the same SRID.
- **ST_Disjoint** - `ST_Disjoint(geom1, geom2)` - Returns true if the geometry `geom1` and the geometry `geom2` do not overlap. Both arguments must be of the same SRID.
- **ST_DumpAsPolygons** - `ST_DumpAsPolygons(geom)` - Returns a set of polygons that represent the individual rings of the geometry `geom`. The function returns a table with two columns: `geomval` (the name of the column to be dumped) and `val` (the name of the geometry column).
- **ST_DumpValues** - `ST_DumpValues(geom)` - Returns a set of values that represent the individual rings of the geometry `geom`. The function returns a table with two columns: `geomval` (the name of the column to be dumped) and `val` (the name of the geometry column).
- **ST_Envelope** - `ST_Envelope(geom)` - Returns the envelope (bounding box) of the geometry `geom`. The function returns a geometry of the same SRID as the input.
- **ST_FromGDALRaster** - `ST_FromGDALRaster(raster)` - Returns a geometry from a GDAL raster. The function returns a geometry of the same SRID as the input.
- **ST_GeoReference** - `ST_GeoReference(geom, srid)` - Returns the geometry `geom` with the SRID `srid`. The function returns a geometry of the same SRID as the input.
- **ST_Grayscale** - Creates a new one-band raster from the source raster and specified bands representing Red, Green and Blue.
- **ST_HasNoBand** - `ST_HasNoBand(geom)` - Returns true if the geometry `geom` has no band. The function returns a boolean value.
- **ST_Height** - `ST_Height(geom)` - Returns the height of the geometry `geom`. The function returns a numeric value.
- **ST_HillShade** - `ST_HillShade(geom, srid)` - Returns the hillshade of the geometry `geom` with the SRID `srid`. The function returns a geometry of the same SRID as the input.
- **ST_Histogram** - `ST_Histogram(geom, bin)` - Returns a histogram of the geometry `geom` with a bin size of `bin`. The function returns a table with two columns: `geomval` (the name of the column to be histogrammed) and `val` (the name of the geometry column).

- 픽셀 유형을 가진,밴드 1개를 가진새 래스터를 생성합니다,따로 밴설정하지 않을 경우,밴드 1로 가&#xc815
- **ST_MapAlgebraFct** - 래스터 밴드 2개 버전,입력 래스터 2개 에 대해 유효한 PostgreSQL 함수적용해서 형성되고,설정한 픽셀 유형을 가진,밴드 1개를 가진 새 래스터를 생성합니다,따로 밴드 설정하지 않을 경우,밴드 1로 가&#xc815 맄위 유형을 따로 설정하지 않을 경우 기본값은 INTERSECTION입니다.
 - **ST_MapAlgebraFctNgb** - 래스터 밴드 1개 버전,사용지정 PostgreSQL 함수를 이용하는 맵 대수최 근접 이웃(Map Algebra Nearest Neighbor)입니다,입력 래스터 밴드의 값의 이웃(neighborhood)이 관련된 PostgreSQL 사용자 함수가 출력하 값을 가진 래스터를 반환합니다.
 - **ST_MapAlgebraExpr** - 표현식 버전 - 입력 래스터 1개 또는 2개,밴드 인덱스,그리고 사용자 지정 SQL 표현식 1개 이상을 입력벱아 밴드 1개를 가진 래스터 반환합니다.
 - **ST_MemSize** - 래스터가 차지하는 공간의 용량을 (바이트 단위로) 반환합니
 - **ST_MetaData** - 래스터 객체의 픽셀 크기,회좌상단,좌하단 등과 같은 기본 메반환합니다.
 - **ST_MinConvexHull** - 래스터의 NODATA 픽셀을 제외한 본록 껍질 도형을 &#bc18;환합니다.
 - **ST_NearestValue** - columnx 򼬏 rowy, 또는 래스터와 동일한 공간 참조 좌표 시스템 단위로 표기하학적 포인트로 지정된 입력 &#bc34;드의 픽셀에 가장 가까운 NODATA 가 아닌 값을 &#bc18;환합니다.
 - **ST_Neighborhood** - columnx 򼬏 rowy, 또는 래스터와 동일한 공간 참조 좌표 시스템 단위로 표기하학적 포인트로 지정된 입력 &#bc34;드의 픽셀 주위의 NODATA 가 아닌 값이중 정벰도 데이터형 2차원 &#bc30;열 &#bc18;환합니다.
 - **ST_NotSameAlignmentReason** - 래스터들이 정렬돼 있는 아닌지,그리고 정렬되지 않았다그 이유를 설명하는 텍스트를 &#bc18;한
 - **ST_NumBands** - 래스터 객체 내부에 있는 &#bc34;개수를 &#bc18;환합니다.
 - **ST_Overlaps** - 래스터 rastA와 래스터 rastB가 교차한 어느 한 경이 다른 한 쪽을 완전히 담고 있지는 않을 경우 참을 &#bc18;환한
 - **ST_PixelAsCentroid** - 픽셀 하나가 차지하는 면의 중심점(포인트 도형)을 &#bc18;환합니

- **ST_PixelAsCentroids** - 래스터 밴드의 각 픽셀에 대한 중심점(포인트 도형)을 각 픽셀에 값 및 X, Y 래스터 좌표와 함૨ 반환포인트 도형은 픽셀이 차지하는 면의 중심점입니다.
- **ST_PixelAsPoint** - 픽셀의 좌상단에 위치하는 포인트 도형을 반환합니다.
- **ST_PixelAsPoints** - 래스터 밴드의 각 픽셀에 대한 중심점(포인트 도형을 각 픽셀의 값 및 X, Y 래스터 좌표와 함૨ 반환합니다 포인트 도형의 좌표는 픽셀의 좌표입니다.
- **ST_PixelAsPolygon** - 특정 행 및 열에 대한 픽셀의 경계를 이루는 폴리곤 도형을 반환합니다.
- **ST_PixelAsPolygons** - 래스터 밴드의 뮨든 픽셀의 경계를 이루는 폴리곤 도형을 각 픽셀의 값과 X, Y 래스터 좌표와 함૨ 반환합니다.
- **ST_PixelHeight** - 픽셀 높이를 공간 참조 시스기하 단위로 반환합니다.
- **ST_PixelOfValue** - 검색 값과 일치하는 값을 가픽셀의 columnx, rowy 좌표를 반환합니다.
- **ST_PixelWidth** - 픽셀 너비를 공간 참조 시스기하 단위로 반환합니다.
- **ST_Polygon** - NODATA 값이 아닌 픽셀 값을 가진 픽셀들을 통합해서 형성된 멀티도형을 &#bc18;환합니다.
- **ST_Quantile** - 샘플링 또는 채우기(population)라는 맥락에서 래스터 또는 래스터 테 ໤버리지의 사분위(quantile)를 계산합따라서 래스터의 25%, 50%, 75% 백분위(percentile) 단계로 값을 확인할 수 있습니다.
- **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** - 주어진 열과 행에 있는 래좌상단의 기하학적 X, Y(경도 위도)으로 반환합니다 열과 행은 1부터 시작합니다.
- **ST_RasterToWorldCoordX** - 열과 행에 있는 래스터의 좌상단의 기하학적 X 좌표를 &#bc18;환 열과 행의 &#bc88;호는 1부터 시작합니다.
- **ST_RasterToWorldCoordY** - 열과 행에 있는 래스터의 좌상단의 기하학적 Y 좌표를 &#bc18;환 열과 행의 &#bc88;호는 1부터 시작합니다.
- **ST_Reclass** - 원본으로부터 재분류된 밴드유형으로 이루어진 새 래스터를 생성합니다 nband 는 변경할 &#bc34;드를

- &#x00;리킵니다. nband 를 따로 설정하지 않을 경우 밴드 1로 가정합니다.&br/>다른 모든 밴드들은 변경 없이 반실제 사례 보기 좋은 형식으로 더 간단하게 렌더링하기 위해 16BUI 밴렌 8BUI 등등으로 변환하십시오.
- **ST_Resample** - 특정 리샘플링 알고리즘,새리న원,임의의 그리드 &#baa8;서리,그리또 다른 래스터에서 정의되거나 &#bbe4c;려온 래스터 지리స조 속성들집합을 이용해서 래스터를 리샘정의스터를 리샘플링합니다. NearestNeighbor(영국 또는 &#bbf8;국 &#cca0;자), Bilinear, Cubic, CubicSpline 또는 Lanczos 리샘플링 알고리즘을 이용해새 픽셀 값을 계산합니다. 기본값을 NearestNeighbor입니다.
 - **ST_Resize** - 래스터의 크기를 새 너비/높이 조정합니다.
 - **ST_Reskew** - 기울기(또는 회전각 파라&#bbf8;터 조정해서 래스터를 리샘플링합정의되거), Bilinear, Cubic, CubicSpline 또는 Lanczos 리샘플링 알고리즘을 이용해서 새 픽셀 값을 계산합니다 기본값은 NearestNeighbor입니다.
 - **ST_Rotation** - 래스터의 회전각을 라디안이 반환합니다.
 - **ST_Roughness** - DEM의 계산된 "거칠기(roughness)"와 함래스터를 반환합니다.
 - **ST_SRID** - spatial_ref_sys 테이블에 정의되어 있는,래스터의 공간 స조 식별자를 반이 래스터들이 동일한 기울기 축ಙ,공간 స조 시스템 그리고 오가지고 있을 경우 (픽셀이 다른 픽고지 않고 동일한 그리드에 놓일 수 있을 경우) స을 반환하고 그렇지 않을 경우 문제점을 설명안내문과 함께 거짓을 반환합니다.
 - **ST_ScaleX** - 픽셀 너비의 X 요소를 좌표 స시스템의 단위로 반환합니다.
 - **ST_ScaleY** - 픽셀 높이의 Y 요소를 좌표 స시스템의 단위로 반환합니다.
 - **ST_SetBandIndex** - Update the external band number of an out-db band
 - **ST_SetBandIsNoData** - 밴드의 isnodata 플래그를 స으로 설정합니다.
 - **ST_SetBandNoDataValue** - NODATA를 나타내는 입력 밴드값을 설정합니다. 밴드를 따로 설 않을 경우 밴드 1로 가정합니다. 밴드에 NODATA가 없다고 표시하려면,안내문과 함께 거짓을 반환합니다.

- **ST_SetBandPath** - Update the external path and band number of an out-db band
- **ST_SetGeoReference** - GDAL ESRI GDAL
- **ST_SetSkew** - Returns a geometry with the same X/Y coordinates as the input geometry, and values from the raster copied into the Z dimension using the requested resample algorithm.
- **ST_SetRotation**
- **ST_SetSRID** - SRID spatial_ref_sys
- **ST_SetScale** - X Y
- **ST_SetSkew** - X Y (skew)
- **ST_SetUpperLeft** - Sets the value of the upper left corner of the pixel of the raster to projected X and Y coordinates.
- **ST_SetValue** - columnx, rowy
- **ST_SetValues**
- **ST_SetSkew** - Returns a geometry with the same X/Y coordinates as the input geometry, and values from the raster copied into the Z dimension using the requested resample algorithm.
- **ST_SkewX** - X (skew)
- **ST_SkewY** - Y (skew)
- **ST_Slope**
- **ST_SnapToGrid** - NearestNeighbor Bilinear Cubic CubicSpline Lanczos
- **ST_Summary**

- ST_SummaryStats** - 입력한 래스터 밴드 또는
래스터 또는 래스터 커버리지의
count, sum, mean, stddev, min, max로 이루어진 통계 요약&#xc7
반환합니다. &#bc34;드를 &#b530;로 설정하
않을 경우 &#bc34;드 1로 가정합니다.
- ST_SummaryStatsAgg** - 종합 함수입니다. 래스터
집합의 입력 래스터 &#bc34;드의 count, sum,
mean, stddev, min, max로 이루어진 통계 요약을
&#bc18;환합니다. &#bc34;드를 &#b530;로 설정하
않을 경우 &#bc34;드 1로 가정합니다.
- ST_TPI** - 계산된 지형위치지수(Topographic Position In-
dex)와 함께 래스터를 &#bc18;환합니다.
- ST_TRI** - 계산된 지형험준도지수(Terrain Rugged-
ness Index)와 함께 래스터를 &#bc18;환합니다
- ST_Tile** - 출력 래스터에 원하는 차원을
&#bc14;탕으로 입력 래스터를 &#bd84;할해d
나온 래스터들의 집합을 &#bc18;환합d
- ST_Touches** - 래스터 rastA와 래스터 rastB가 최소&#xd1
포인트 한 개를 &#acf5;유하지만, 내부&
교차하지는 않을 경우 TRUE 를 &#bc18;환&#xc
- ST_Transform** - 알려진 &#acf5;간 참조 시스템의
래스터를 지정한 리샘플링 알고&
통해 또 다른 알려진 &#acf5;간 참조
시스템으로 재투영합니다. NearestNeigh-
bor, Bilinear, Cubic, CubicSpline, Lanczos 알고리즘을 이용할
수 있습니다. 기본값은 NearestNeighbor입니&#xc
- ST_Union** - 래스터 타일 집합을 1개 이상&#xc75
&#bc34;드로 이루어진 단일 래스터로
통합합니다..
- ST_UpperLeftX** - 래스터의 좌상단 X 좌표를 투&
&#acf5;간 참조 단위로 &#bc18;환합니다.
- ST_UpperLeftY** - 래스터의 좌상단 Y 좌표를 투&
&#acf5;간 참조 단위로 &#bc18;환합니다.
- ST_Value** - 입력 columnx, rowy 픽셀의 입력 &#bc34;드의
값, 또는 특정 기하학적 포인트의
&#bc34;드의 값을 &#bc18;환합니다. &#bc34;드 &#bc88;
1부터 시작하며. &#b530;로 지정하지
않을 경우 1로 가정합니다. exclude_nodata_value
를 거짓으로 설정할 경우. nodata 픽셀&#
포함한 &#baa8;든 픽셀이 교차한다고
가정하고 값을 &#bc18;환합니&#xb2e

- **ST_Width** - Returns the width of a 2D geometry. The width is the maximum distance between any two points in the geometry along the x-axis.
- **ST_Within** - Returns true if the first geometry is completely within the second geometry.
- **ST_WorldToRasterCoord** - Returns the world coordinates of a point in a raster. The world coordinates are the coordinates in the same units as the raster's SRID.
- **ST_WorldToRasterCoordX** - Returns the x-world coordinate of a point in a raster.
- **ST_WorldToRasterCoordY** - Returns the y-world coordinate of a point in a raster.
- **UpdateRasterSRID** - Updates the SRID of a raster to the specified value.
- **ST_PixelOfValue** - Returns the pixel value of a raster at a given location.

15.6 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** - Returns a set of polygons from a geometry.
- **ST_Intersection** - Returns the intersection of two geometries.

15.7 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**.

- **Box2D** - Returns a BOX2D representing the 2D extent of a geometry.
- **Box3D** - Returns a BOX3D representing the 3D extent of a geometry.

- **Box3D** - Returns the 3D bounding box of a geometry.
- **ST_3DExtent** - Aggregate function that returns the 3D bounding box of geometries.
- **ST_3DMakeBox** - Creates a BOX3D defined by two 3D point geometries.
- **ST_AsMVTGeom** - Transforms a geometry into the coordinate space of a MVT tile.
- **ST_AsTWKB** - Returns the TWKB (Tiny Well-Known Binary) representation of a geometry.
- **ST_Box2dFromGeoHash** - GeoHash to 2D bounding box. Returns a BOX2D defined by two 2D point geometries.
- **ST_ClipByBox2D** - Computes the portion of a geometry falling within a rectangle.
- **ST_EstimatedExtent** - Returns the estimated extent of a spatial table.
- **ST_Expand** - Returns a bounding box expanded from another bounding box or a geometry.
- **ST_Extent** - Aggregate function that returns the bounding box of geometries.
- **ST_MakeBox2D** - Creates a BOX2D defined by two 2D point geometries.
- **ST_XMax** - Returns the X maxima of a 2D or 3D bounding box or a geometry.
- **ST_XMin** - Returns the X minima of a 2D or 3D bounding box or a geometry.
- **ST_YMax** - Returns the Y maxima of a 2D or 3D bounding box or a geometry.
- **ST_YMin** - Returns the Y minima of a 2D or 3D bounding box or a geometry.
- **ST_ZMax** - Returns the Z maxima of a 2D or 3D bounding box or a geometry.
- **ST_ZMin** - Returns the Z minima of a 2D or 3D bounding box or a geometry.
- **RemoveUnusedPrimitives** - Removes topology primitives which not needed to define existing TopoGeometry objects.
- **ValidateTopology** - Returns a set of validate_topology_return_type objects detailing issues with topology.
- **~(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 bounding box.
- **~(geometry,box2df)** - Returns TRUE if a geometry's 2D bounding 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).

15.8 PostGIS Functions that support 3D

The functions given below are PostGIS functions that do not throw away the Z-Index.

- **AddGeometryColumn** - Adds a new geometry column to a table.
- **Box3D** - Returns a BOX3D representing the 3D extent of a geometry.
- **DropGeometryColumn** - Drops a geometry column from a table.
- **GeometryType** - Returns the geometry type of a geometry.
- **ST_3DArea** - Returns the 3D area of a geometry.
- **ST_3DClosestPoint** - Returns the 3D closest point to a geometry.
- **ST_3DConvexHull** - Returns the 3D convex hull of a geometry.
- **ST_3DDFullyWithin** - Tests if two 3D geometries are entirely within a given 3D distance.
- **ST_3DDWithin** - Tests if two 3D geometries are within a given 3D distance.
- **ST_3DDifference** - Returns the 3D difference of two geometries.
- **ST_3DDistance** - Returns the 3D distance between two geometries.
- **ST_3DExtent** - Aggregate function that returns the 3D bounding box of geometries.
- **ST_3DIntersection** - Returns the 3D intersection of two geometries.
- **ST_3DIntersects** - Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area).
- **ST_3DLength** - Returns the 3D length of a geometry.
- **ST_3DLineInterpolatePoint** - Returns a point interpolated along a 3D line at a fractional location.
- **ST_3DLongestLine** - Returns the 3D longest line segment of a geometry.
- **ST_3DMaxDistance** - Returns the 3D maximum distance between two geometries.
- **ST_3DPerimeter** - Returns the 3D perimeter of a geometry.
- **ST_3DShortestLine** - Returns the 3D shortest line segment of a geometry.
- **ST_3DUnion** - Perform 3D union.
- **ST_AddMeasure** - Interpolates measures along a linear geometry.
- **ST_AddPoint** - Adds a point to a geometry.
- **ST_Affine** - Apply a 3D affine transformation to a geometry.

- **ST_ApproximateMedialAxis** - Returns the approximate medial axis of a geometry.
- **ST_AsBinary** - Return the OGC/ISO Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.
- **ST_AsEWKB** - Return the Extended Well-Known Binary (EWKB) representation of the geometry with SRID meta data.
- **ST_AsEWKT** - Returns the Well-Known Text (WKT) representation of the geometry with SRID meta data.
- **ST_AsGML** - Returns the GML 2 or GML 3 representation of the geometry.
- **ST_AsGeoJSON** - Return a geometry as a GeoJSON element.
- **ST_AsHEXEWKB** - Returns the HEXEWKB representation of the geometry.
- **ST_AsKML** - Returns the KML representation of the geometry.
- **ST_AsX3D** - Returns the X3D XML representation of the geometry.
- **ST_Boundary** - Returns the boundary of a geometry.
- **ST_BoundingDiagonal** - Returns the bounding diagonal of a geometry.
- **ST_CPAWithin** - Tests if the closest point of approach of two trajectories is within the specified distance.
- **ST_ClosestPointOfApproach** - Returns a measure at the closest point of approach of two trajectories.
- **ST_Collect** - Creates a GeometryCollection or Multi* geometry from a set of geometries.
- **ST_ConstrainedDelaunayTriangles** - Return a constrained Delaunay triangulation around the given input geometry.
- **ST_ConvexHull** - Computes the convex hull of a geometry.
- **ST_CoordDim** - Returns the coordinate dimension of a geometry.
- **ST_CurveToLine** - Converts a geometry containing curves to a linear geometry.
- **ST_DelaunayTriangles** - Returns the Delaunay triangulation of the vertices of a geometry.
- **ST_Difference** - Computes a geometry representing the part of geometry A that does not intersect geometry B.
- **ST_DistanceCPA** - Returns the distance between the closest point of approach of two trajectories.
- **ST_Dump** - Returns a set of geometry_dump rows for the components of a geometry.
- **ST_DumpPoints** - Returns a set of geometry_dump rows for the vertices of a geometry.
- **ST_DumpRings** - Returns a set of geometry_dump rows for the exterior and interior rings of a Polygon.
- **ST_DumpSegments** - Returns a set of geometry_dump rows for the segments of a geometry.
- **ST_EndPoint** - Returns the end point of a geometry.

- **ST_ExteriorRing** - Returns the exterior ring of a geometry.
- **ST_Extrude** - Extrudes a 2D geometry into 3D.
- **ST_FlipCoordinates** - Returns a version of a geometry with X and Y axis flipped.
- **ST_Force2D** - Forces a geometry to be 2D.
- **ST_ForceCurve** - Forces a geometry to be a curve.
- **ST_ForceLHR** - LHR(Left Hand Reverse; Right Hand Rule).
- **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** - RHR(Right Hand Rule; Left Hand Rule).
- **ST_ForceSFS** - SFS 1.1.
- **ST_Force_3D** - Converts a 2D geometry to 3D.
- **ST_Force_3DZ** - Converts a 2D geometry to 3D with Z values.
- **ST_Force_4D** - Converts a 2D geometry to 4D.
- **ST_Force_Collection** - Forces a collection of geometries to be a single geometry.
- **ST_GeomFromEWKB** - EWKB(Extended Well-Known Binary).
- **ST_GeomFromEWKT** - EWKT(Extended Well-Known Text).
- **ST_GeomFromGML** - GML(Geometry Markup Language).
- **ST_GeomFromGeoJSON** - GeoJSON.
- **ST_GeomFromKML** - KML(Killfly Markup Language).
- **ST_GeometricMedian** - Returns the geometric median of a set of points.
- **ST_GeometryN** - Returns the Nth geometry in a collection.
- **ST_GeometryType** - Returns the geometry type of a geometry.
- **ST_HasArc** - Tests if a geometry contains a circular arc.
- **ST_InteriorRingN** - Returns the Nth interior ring of a geometry.
- **ST_InterpolatePoint** - Interpolates a point on a line segment.

- **ST_Intersection** - Computes a geometry representing the shared portion of geometries A and B.
- **ST_IsClosed** - LINESTRING geometry type. Returns TRUE if the geometry is closed (the first and last points are the same).
TRUE if the geometry is closed, FALSE otherwise.
- **ST_IsCollection** - Returns TRUE if the geometry is a collection (MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, or GEOMETRYCOLLECTION), FALSE otherwise.
- **ST_IsPlanar** - Returns TRUE if the geometry is planar, FALSE otherwise.
- **ST_IsPolygonCCW** - Tests if Polygons have exterior rings oriented counter-clockwise and interior rings oriented clockwise.
- **ST_IsPolygonCW** - Tests if Polygons have exterior rings oriented clockwise and interior rings oriented counter-clockwise.
- **ST_IsSimple** - Returns TRUE if the geometry is simple (does not intersect itself), FALSE otherwise.
- **ST_IsSolid** - Returns TRUE if the geometry is a solid (POLYGON or POLYHEDRON), FALSE otherwise.
- **ST_IsValidTrajectory** - Tests if the geometry is a valid trajectory.
- **ST_Length_Spheroid** - Returns the length of the geometry in spheroid coordinates.
- **ST_LineFromMultiPoint** - Returns a line from a set of points.
- **ST_LineInterpolatePoint** - Returns a point interpolated along a line at a fractional location.
- **ST_LineInterpolatePoints** - Returns points interpolated along a line at a fractional interval.
- **ST_LineSubstring** - Returns the part of a line between two fractional locations.
- **ST_LineToCurve** - Converts a linear geometry to a curved geometry.
- **ST_LocateBetweenElevations** - Returns the portions of a geometry that lie in an elevation (Z) range.
- **ST_M** - Returns the M coordinate of a Point.
- **ST_MakeLine** - Returns a line from a set of points.
- **ST_MakePoint** - Creates a 2D, 3DZ or 4D Point.
- **ST_MakePolygon** - Creates a Polygon from a shell and optional list of holes.
- **ST_MakeSolid** - Returns a solid geometry from a shell and optional list of holes.
- **ST_MakeValid** - Attempts to make an invalid geometry valid without losing vertices.
- **ST_MemSize** - Returns the memory size of the geometry.
- **ST_MemUnion** - Aggregate function which unions geometries in a memory-efficient but slower way.
- **ST_NDims** - Returns the number of dimensions of the geometry.
- **ST_NPoints** - Returns the number of points in the geometry.

- **ST_NRings** - Returns the number of rings in a geometry.
- **ST_Node** - Nodes a collection of lines.
- **ST_NumGeometries** - Returns the number of geometries in a collection.
- **ST_NumPatches** - Returns the number of patches in a geometry.
- **ST_Orientation** - Returns the orientation of a geometry.
- **ST_PatchN** - Returns the Nth patch of a geometry.
- **ST_PointFromWKB** - Returns a point from a WKB.
- **ST_PointN** - Returns the Nth point of a geometry.
- **ST_PointOnSurface** - Computes a point guaranteed to lie in a polygon, or on a geometry.
- **ST_Points** - Returns the points of a geometry.
- **ST_Polygon** - Creates a Polygon from a LineString with a specified SRID.
- **ST_RemovePoint** - Remove a point from a linestring.
- **ST_RemoveRepeatedPoints** - Returns a version of a geometry with duplicate points removed.
- **ST_Reverse** - Returns the reverse of a geometry.
- **ST_Rotate** - Rotates a geometry about an origin point.
- **ST_RotateX** - Rotates a geometry about the X axis.
- **ST_RotateY** - Rotates a geometry about the Y axis.
- **ST_RotateZ** - Rotates a geometry about the Z axis.
- **ST_Scale** - Scales a geometry by given factors.
- **ST_Scroll** - Change start point of a closed LineString.
- **ST_SetPoint** - Sets a point in a geometry.
- **ST_Shift_Longitude** - Shifts the longitude coordinates of a geometry between -180..180 and 0..360.
- **ST_SnapToGrid** - Snaps a geometry to a grid.
- **ST_StartPoint** - Returns the first point of a LineString.
- **ST_StraightSkeleton** - Returns the straight skeleton of a geometry.
- **ST_SwapOrdinates** - Swaps the X and Y coordinates of a geometry.
- **ST_SymDifference** - Computes a geometry representing the portions of geometries A and B that do not intersect.

- **ST_Tessellate** - Takes a geometry and returns a geometry representing the tessellation of the input geometry. The input geometry must be a TIN or a geometry with a TIN. The output geometry is a TIN.
- **ST_TransScale** - Translates and scales a geometry by given offsets and factors.
- **ST_Translate** - Translates a geometry by given offsets.
- **ST_Union** - Computes the union of the components of a single geometry.
- **ST_Union** - Computes a geometry representing the point-set union of the input geometries.
- **ST_Volume** - Returns the volume of a 3D geometry. The input geometry must be a 3D geometry.
- **ST_WrapX** - Wraps the X coordinate of a geometry to the range [-180, 180].
- **ST_X** - Returns the X coordinate of a Point.
- **ST_XMax** - Returns the X maxima of a 2D or 3D bounding box or a geometry.
- **ST_XMin** - Returns the X minima of a 2D or 3D bounding box or a geometry.
- **ST_Y** - Returns the Y coordinate of a Point.
- **ST_YMax** - Returns the Y maxima of a 2D or 3D bounding box or a geometry.
- **ST_YMin** - Returns the Y minima of a 2D or 3D bounding box or a geometry.
- **ST_Z** - Returns the Z coordinate of a Point.
- **ST_ZMax** - Returns the Z maxima of a 2D or 3D bounding box or a geometry.
- **ST_ZMin** - Returns the Z minima of a 2D or 3D bounding box or a geometry.
- **ST_Zmflag** - Returns the Z maxima flag of a 2D or 3D bounding box or a geometry.
- **TG_Equals** - Returns TRUE if two TopoGeometry objects are equal.
- **TG_Intersects** - Returns TRUE if two TopoGeometry objects intersect.
- **UpdateGeometrySRID** - Updates the SRID of all features in a geometry column, and the table metadata.
- **geometry_overlaps_nd** - Returns TRUE if a geometry's (cached) n-D bounding box intersects a n-D float precision bounding box (GIDX).
- **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_full_version** - Returns the full version of SFCGAL in use including CGAL and Boost versions.
- **postgis_sfcgal_version** - Returns the version of SFCGAL in use.

15.9 PostGIS Curved Geometry Support Functions

The functions given below are PostGIS functions that can use CIRCULARSTRING, CURVEPOLYGON, and other curved geometry types

- **AddGeometryColumn** - Adds a new geometry column to a table.
- **Box2D** - Returns a BOX2D representing the 2D extent of a geometry.
- **Box3D** - Returns a BOX3D representing the 3D extent of a geometry.
- **DropGeometryColumn** - Drops a geometry column from a table.
- **ST_3DExtent** - Aggregate function that returns the 3D bounding box of geometries.
- **ST_Affine** - Apply a 3D affine transformation to a geometry.
- **ST_AsBinary** - Return the OGC/ISO Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.
- **ST_AsEWKB** - Return the Extended Well-Known Binary (EWKB) representation of the geometry with SRID meta data.
- **ST_AsEWKT** - Return the Well-Known Text (WKT) representation of the geometry with SRID meta data.
- **ST_AsHEXEWKB** - Return the Hexagonal Well-Known Binary (HEXEWKB) representation of the geometry with SRID meta data.
- **ST_AsText** - Return the Well-Known Text (WKT) representation of the geometry without SRID meta data.
- **ST_GeomCollFromText** - Creates a GeometryCollection or Multi* geometry from a set of geometries.
- **ST_CoordDim** - Returns the dimension of a geometry.
- **ST_CurveToLine** - Converts a geometry containing curves to a linear geometry.
- **ST_Distance** - Returns the distance between two geometries.
- **ST_Dump** - Returns a set of geometry_dump rows for the components of a geometry.
- **ST_NumPoints** - Returns the number of points in a geometry.
- **ST_EndPoint** - Returns the end point of a geometry.
- **ST_EstimatedExtent** - Returns the estimated extent of a spatial table.
- **ST_FlipCoordinates** - Returns a version of a geometry with X and Y axis flipped.
- **ST_Force2D** - Returns a 2D geometry from a 3D geometry.

- **ST_ForceCurve** - Converts a geometry to a curve. The curve is defined by the control points and the start and end points. The curve is defined by the control points and the start and end points. The curve is defined by the control points and the start and end points.
- **ST_ForceSFS** - Converts a geometry to a surface. The surface is defined by the control points and the start and end points. The surface is defined by the control points and the start and end points. The surface is defined by the control points and the start and end points.
- **ST_Force3D** - Converts a geometry to a 3D geometry. The 3D geometry is defined by the control points and the start and end points. The 3D geometry is defined by the control points and the start and end points. The 3D geometry is defined by the control points and the start and end points.
- **ST_Force3DM** - Converts a geometry to a 3D geometry. The 3D geometry is defined by the control points and the start and end points. The 3D geometry is defined by the control points and the start and end points. The 3D geometry is defined by the control points and the start and end points.
- **ST_Force3DZ** - Converts a geometry to a 3D geometry. The 3D geometry is defined by the control points and the start and end points. The 3D geometry is defined by the control points and the start and end points. The 3D geometry is defined by the control points and the start and end points.
- **ST_Force4D** - Converts a geometry to a 4D geometry. The 4D geometry is defined by the control points and the start and end points. The 4D geometry is defined by the control points and the start and end points. The 4D geometry is defined by the control points and the start and end points.
- **ST_ForceCollection** - Converts a geometry to a collection. The collection is defined by the control points and the start and end points. The collection is defined by the control points and the start and end points. The collection is defined by the control points and the start and end points.
- **ST_GeoHash** - Converts a geometry to a GeoHash. The GeoHash is defined by the control points and the start and end points. The GeoHash is defined by the control points and the start and end points. The GeoHash is defined by the control points and the start and end points.
- **ST_GeogFromWKB** - Converts a WKB geometry to a geography. The geography is defined by the control points and the start and end points. The geography is defined by the control points and the start and end points. The geography is defined by the control points and the start and end points.
- **ST_GeomFromEWKB** - Converts an EWKB geometry to a geometry. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **ST_GeomFromEWKT** - Converts an EWKT geometry to a geometry. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **ST_GeomFromText** - Converts a WKT geometry to a geometry. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **ST_GeomFromWKB** - Converts a WKB geometry to a geometry. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **ST_GeometryN** - Returns the Nth geometry of a geometry collection. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **=** - Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B.
- **<** - 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_HasArc** - Tests if a geometry contains a circular arc.
- **ST_Intersects** - Tests if two geometries intersect (they have at least one point in common).
- **ST_IsClosed** - Tests if a LINESTRING geometry is closed. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **ST_IsCollection** - Tests if a geometry is a collection. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **ST_IsEmpty** - Tests if a geometry is empty.
- **ST_LineToCurve** - Converts a linear geometry to a curved geometry.
- **ST_MemSize** - Returns the memory size of a geometry. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.
- **ST_NPoints** - Returns the number of points in a geometry. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points. The geometry is defined by the control points and the start and end points.

- **ST_NRings** - Returns the number of rings in a polygonal geometry.
- **ST_PointFromWKB** - Returns a point from a WKB binary representation.
- **ST_PointN** - Returns the Nth point of a LineString or MultiPoint geometry.
- **ST_Points** - Returns an array of points from a geometry.
- **ST_Rotate** - Rotates a geometry about an origin point.
- **ST_RotateZ** - Rotates a geometry about the Z axis.
- **ST_SRID** - Returns the spatial reference identifier for a geometry.
- **ST_Scale** - Scales a geometry by given factors.
- **ST_SetSRID** - Set the SRID on a geometry.
- **ST_StartPoint** - Returns the first point of a LineString.
- **ST_Summary** - Returns a text summary of a geometry.
- **ST_SwapOrdinates** - Swaps the X and Y coordinates of a geometry.
- **ST_TransScale** - Translates and scales a geometry by given offsets and factors.
- **ST_Transform** - Return a new geometry with coordinates transformed to a different spatial reference system.
- **ST_Translate** - Translates a geometry by given offsets.
- **ST_XMax** - Returns the X maxima of a 2D or 3D bounding box or a geometry.
- **ST_XMin** - Returns the X minima of a 2D or 3D bounding box or a geometry.
- **ST_YMax** - Returns the Y maxima of a 2D or 3D bounding box or a geometry.
- **ST_YMin** - Returns the Y minima of a 2D or 3D bounding box or a geometry.
- **ST_ZMax** - Returns the Z maxima of a 2D or 3D bounding box or a geometry.
- **ST_ZMin** - Returns the Z minima of a 2D or 3D bounding box or a geometry.
- **ST_Zmflag** - Returns the Z flag of a geometry.
- **UpdateGeometrySRID** - Updates the SRID of all features in a geometry column, and the table metadata.
- **~(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 bounding box.
- **~(geometry,box2df)** - Returns TRUE if a geometry's 2D bounding box contains a 2D float precision bounding box (BOX2DF).
- **&&** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).
- **&&&** - Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 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).
- **&&&(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.

15.10 PostGIS Polyhedral Surface Support Functions

The functions given below are PostGIS functions that can use POLYHEDRALSURFACE, POLYHEDRALSURFACEM geometries

- **Box2D** - Returns a BOX2D representing the 2D extent of a geometry.
- **Box3D** - Returns a BOX3D representing the 3D extent of a geometry.
- **도형 유형** - ST_Geometry 값의 도형 유형을 반환합니다.
- **ST_3DArea** - 3차원 표면 도형의 면적을 계입체일 경우 0을 반환할 것입니다
- **ST_3DClosestPoint** - g2에 가장 가까운 g1 상에 있는 3차원 포인트를 반환합니다. 해&#xb2f9 포인트는 3D 최단 라인의 첫 번째 포인트입니다.
- **ST_3DConvexHull** - 면 도형의 근사 중심축을 계&
- **ST_3DDFullyWithin** - Tests if two 3D geometries are entirely within a given 3D distance
- **ST_3DDWithin** - Tests if two 3D geometries are within a given 3D distance
- **ST_3DDifference** - 3차원 차이를 수행합니다.
- **ST_3DDistance** - 도형 유형에 대해. 두 도형 사에 기반한) 3차원 데카르트 최단 거리를 투영 단위로 반환합니다.
- **ST_3DExtent** - Aggregate function that returns the 3D bounding box of geometries.
- **ST_3DIntersection** - 3차원 교차를 수행합니다.

- **ST_3DIntersects** - Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area).
- **ST_3DLongestLine** - Returns the longest line segment within a 3D geometry. (longest line segment within a 3D geometry).
- **ST_3DMaxDistance** - Returns the maximum distance between two 3D geometries.
- **ST_3DShortestLine** - Returns the shortest line segment within a 3D geometry.
- **ST_3DUnion** - Perform 3D union.
- **ST_Affine** - Apply a 3D affine transformation to a geometry.
- **ST_ApproximateMedialAxis** - Returns the approximate medial axis of a 3D geometry.
- **ST_Area** - Returns the area of a 3D geometry.
- **ST_AsBinary** - Return the OGC/ISO Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.
- **ST_AsEWKB** - Return the Extended Well-Known Binary (EWKB) representation of the geometry with SRID meta data.
- **ST_AsEWKT** - Returns the Well-Known Text (WKT) representation of the geometry with SRID meta data.
- **ST_AsGML** - Returns the GML 2 or 3 representation of the geometry with SRID meta data.
- **ST_AsX3D** - Returns the X3D XML representation of the geometry with SRID meta data.
- **ST_CoordDim** - Returns the dimension of a 3D geometry.
- **ST_Dimension** - Returns the dimension of a 3D geometry.
- **ST_Dump** - Returns a set of geometry_dump rows for the components of a geometry.
- **ST_NumPoints** - Returns the number of points in a 3D geometry.
- **ST_Expand** - Returns a bounding box expanded from another bounding box or a geometry.
- **ST_Extent** - Aggregate function that returns the bounding box of geometries.
- **ST_Extrude** - Returns a 3D geometry extruded from a 2D geometry.
- **ST_FlipCoordinates** - Returns a version of a geometry with X and Y axis flipped.
- **ST_Force2D** - Returns a 2D geometry from a 3D geometry.
- **ST_ForceLHR** - Returns a 3D geometry with Left-Hand Rule orientation.
- **ST_ForceRHR** - Returns a 3D geometry with Right-Hand Rule orientation.
- **ST_ForceSFS** - Returns a 3D geometry with SFS 1.1 orientation.
- **ST_Force3D** - Returns a 3D geometry from a 2D geometry.

- **ST_Force3DZ** - Converts a 2D geometry to a 3D geometry with Z values. The Z values are calculated based on the XY values and the Z value of the first point.
- **ST_ForceCollection** - Converts a geometry to a collection of geometries.
- **ST_GeomFromEWKB** - EWKB(Extended Well-Known Binary) to geometry.
- **ST_GeomFromEWKT** - EWKT(Extended Well-Known Text) to geometry.
- **ST_GeomFromGML** - GML to geometry.
- **ST_GeometryN** - Returns the Nth geometry from a collection.
- **ST_GeometryType** - Returns the geometry type of a 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.
- **<=** - Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B, or if A is a subset of B.
- **~** - Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B, or if A is a superset of B.
- **ST_IsClosed** - Returns TRUE if the first and last points of a linestring are the same.
- **ST_IsPlanar** - Returns TRUE if the geometry is planar.
- **ST_IsSolid** - Returns TRUE if the geometry is solid.
- **ST_MakeSolid** - Returns a solid geometry from a non-solid geometry.
- **ST_MemSize** - Returns the memory size of a geometry.
- **ST_NPoints** - Returns the number of points in a geometry.
- **ST_NumGeometries** - Returns the number of geometries in a collection.
- **ST_NumPatches** - Returns the number of patches in a geometry.
- **ST_PatchN** - Returns the Nth patch from a geometry.
- **ST_RemoveRepeatedPoints** - Returns a version of a geometry with duplicate points removed.
- **ST_Reverse** - Returns the reverse of a geometry.
- **ST_Rotate** - Rotates a geometry about an origin point.

- **ST_RotateX** - Rotates a geometry about the X axis.
- **ST_RotateY** - Rotates a geometry about the Y axis.
- **ST_RotateZ** - Rotates a geometry about the Z axis.
- **ST_Scale** - Scales a geometry by given factors.
- **ST_ShiftLongitude** - Shifts the longitude coordinates of a geometry between -180..180 and 0..360.
- **ST_StraightSkeleton** - Returns the straight skeleton of a geometry.
- **ST_Summary** - Returns a summary of the geometry.
- **ST_SwapOrdinates** - Swaps the X and Y coordinates of a geometry.
- **ST_Tessellate** - Tessellates a geometry into a TIN.
- **ST_Transform** - Return a new geometry with coordinates transformed to a different spatial reference system.
- **ST_Volume** - Returns the volume of a 3D geometry.
- **~(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 bounding box.
- **~(geometry,box2df)** - Returns TRUE if a geometry's 2D bounding box contains a 2D float precision bounding box (BOX2DF).
- **&&** - Returns TRUE if two 2D float precision bounding boxes (BOX2DF) intersect each other.
- **&&(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` - Returns the full version of SFCGAL in use including CGAL and Boost versions
- `postgis_sfcgal_version` - SFCGAL

15.11 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
Box2D	✓			✓		✓	✓
Box3D	✓		✓	✓		✓	✓
Box3D	✓		✓	✓		✓	✓
PostGIS_AddBBox	✓			✓			
PostGIS_DropBBox	✓			✓			
PostGIS_Extensions_Upgrade							
PostGIS_Full_Version							
PostGIS_GEOS_Version							
PostGIS_HasBBox	✓			✓			
PostGIS_LibXML_Version							
PostGIS_Lib_Build_Date							
PostGIS_Lib_Version							
PostGIS_Libwgeom_Version							
PostGIS_PROJ_Version							

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
PostGIS_Scripts_Build_Date							
PostGIS_Scripts_Installed							
PostGIS_Scripts_Released							
PostGIS_Version							
PostGIS_Wagyu_Version							
ST_3DArea							
ST_3DClosestPoint	✓		✓			✓	
ST_3DConvexHull							
ST_3DDifference							
ST_3DDistance	✓		✓		✓	✓	
ST_3DExtent	✓		✓	✓		✓	✓
ST_3DIntersection							
ST_3DLength	✓		✓		✓		
ST_LineInterpolate	✓		✓				
ST_3DLongestLine	✓		✓			✓	
ST_3DMakeBox	✓						
ST_3DMaxDistance	✓		✓			✓	
ST_3DPerimeter	✓		✓		✓		
ST_3DShortestLine	✓		✓			✓	
ST_3DUnion							
ST_AddMeasure	✓		✓				
ST_AddPoint	✓		✓				
ST_Affine	✓		✓	✓		✓	✓
ST_AlphaShape							
ST_Angle	✓						
ST_ApproximateM							
ST_Area	✓	✓			✓	✓	
ST_Azimuth	✓	✓					
ST_Boundary	✓		✓		✓		
ST_BoundingDiagonal	✓		✓				
ST_Buffer	✓	✓			✓		
ST_BuildArea	✓						
ST_CPAWithin	✓		✓				

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Centroid	✓	✓			✓		
ST_ChaikinSmooth	✓						
ST_ClipByBox2D	✓						
ST_ClosestPoint	✓						
ST_ClosestPointOfApproach	✓		✓				
ST_ClusterDBSCAN	✓						
ST_ClusterIntersect	✓						
ST_ClusterKMeans	✓						
ST_ClusterWithin	✓						
ST_GeomCollFromText	✓		✓	✓			
ST_CollectionExtract	✓						
ST_CollectionHomogenize	✓						
ST_ConcaveHull	✓						✓
ST_ConstrainedDelaunayTriangles							
ST_ConvexHull	✓		✓		✓		
ST_CoordDim	✓		✓	✓	✓	✓	✓
ST_CurveToLine	✓		✓	✓	✓		
ST_DelaunayTriangles	✓		✓				✓
ST_Difference	✓		✓		✓		
ST_Dimension	✓				✓	✓	✓
ST_Distance	✓			✓	✓		
ST_DistanceCPA	✓		✓				
ST_DistanceSphere	✓						
ST_DistanceSpheroid	✓						
ST_Dump	✓		✓	✓		✓	✓
ST_NumPoints	✓		✓	✓		✓	✓
ST_NRings	✓		✓				
ST_NumPoints	✓		✓				✓
ST_EndPoint	✓		✓	✓	✓		
ST_Envelope	✓				✓		
ST_EstimatedExtent				✓			
ST_Expand	✓					✓	✓
ST_Extent	✓					✓	✓

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
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_GeneratePoints	✓						
ST_GeometricMedian	✓		✓				
ST_GeometryN	✓		✓	✓	✓	✓	✓
ST_GeometryType	✓		✓		✓	✓	
ST_HasArc	✓		✓	✓			
ST_HausdorffDistance	✓						
ST_Hexagon	✓						
ST_HexagonGrid	✓						
ST_InteriorRingN	✓		✓		✓		
ST_InterpolatePoint	✓		✓				
ST_Intersection	✓	✓	✓		✓		
ST_IsClosed	✓		✓	✓	✓	✓	
ST_IsCollection	✓		✓	✓			
ST_IsEmpty	✓			✓	✓		
ST_IsPlanar							
ST_IsPolygonCCW	✓		✓				

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_IsPolygonCW	✓		✓				
ST_IsRing	✓				✓		
ST_IsSimple	✓		✓		✓		
ST_IsSolid							
ST_IsValid	✓				✓		
ST_IsValidDetail	✓						
ST_IsValidReason	✓						
ST_IsValidTrajectory	✓		✓				
ST_Length	✓	✓					
ST_Length2D	✓						
ST_LengthSpheroid	✓		✓				
ST_Letters	✓						
ST_LineFromMultiPoint	✓		✓				
ST_LineInterpolatePoint	✓		✓				
ST_LineInterpolatePoints	✓		✓				
ST_LineLocatePoint	✓						
ST_LineMerge	✓						
ST_LineSubstring	✓		✓				
ST_LineToCurve	✓		✓	✓			
ST_LocateAlong	✓				✓		
ST_LocateBetween	✓				✓		
ST_LocateBetweenPoints	✓		✓				
ST_LongestLine	✓						
ST_M	✓		✓		✓		
ST_MakeBox2D	✓						
ST_MakeEnvelope	✓						
ST_MakeLine	✓		✓				
ST_MakePoint	✓		✓				
ST_MakePointM	✓						
ST_MakePolygon	✓		✓				
ST_MakeSolid							
ST_MakeValid	✓		✓				
ST_MaxDistance	✓						

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_MaximumInscribedCircle	✓						
ST_MemSize	✓		✓	✓		✓	✓
ST_MemUnion	✓		✓				
ST_MinimumBoundingCircle	✓						
ST_MinimumBoundingRadius	✓						
ST_MinimumClearance	✓						
ST_MinimumClearanceLine	✓						
ST_MinkowskiSum							
ST_Multi	✓						
ST_NDims	✓		✓				
ST_NPoints	✓		✓	✓		✓	
ST_NRings	✓		✓	✓			
ST_Node	✓		✓				
ST_Normalize	✓						
ST_NumGeometries	✓		✓		✓	✓	✓
ST_NumInteriorRings	✓						
ST_NumInteriorRings	✓				✓		
ST_NumPatches	✓		✓		✓	✓	
ST_NumPoints	✓				✓		
ST_OffsetCurve	✓						
ST_OptimalAlphaS							
ST_Orientation							
ST_OrientedEnvelope	✓						
ST_PatchN	✓		✓		✓	✓	
ST_Perimeter	✓	✓			✓		
ST_Perimeter2D	✓						
ST_Point	✓				✓		
ST_Point	✓						
ST_PointN	✓		✓	✓	✓		
ST_PointOnSurface	✓		✓		✓		
ST_Point	✓						
ST_Point	✓						
ST_Points	✓		✓	✓			

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Polygon	✓		✓		✓		
ST_Polygonize	✓						
ST_Project		✓					
ST_QuantizeCoordinates	✓						
ST_ReducePrecision	✓						
ST_RemovePoint	✓		✓				
ST_RemoveRepeatedPoints	✓		✓			✓	
ST_Reverse	✓		✓			✓	
ST_Rotate	✓		✓	✓		✓	✓
ST_RotateX	✓		✓			✓	✓
ST_RotateY	✓		✓			✓	✓
ST_RotateZ	✓		✓	✓		✓	✓
ST_SRID	✓			✓	✓		
ST_Scale	✓		✓	✓		✓	✓
ST_Scroll	✓		✓				
ST_Segmentize	✓	✓					
ST_SetEffectiveArea	✓						
ST_SetPoint	✓		✓				
ST_SetSRID	✓			✓			
ST_SharedPaths	✓						
ST_ShiftLongitude	✓		✓			✓	✓
ST_ShortestLine	✓						
ST_Simplify	✓						
ST_SimplifyPolygon	✓						
ST_SimplifyPreserveTopology	✓						
ST_SimplifyVW	✓						
ST_Snap	✓						
ST_SnapToGrid	✓		✓				
ST_Split	✓						
ST_Square	✓						
ST_SquareGrid	✓						
ST_StartPoint	✓		✓	✓	✓		
ST_StraightSkeleton							

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_Subdivide	✓						
ST_Summary	✓	✓		✓		✓	✓
ST_SwapOrdinates	✓		✓	✓		✓	✓
ST_SymDifference	✓		✓		✓		
ST_Tessellate							
ST_MakeEnvelope	✓						
ST_TransScale	✓		✓	✓			
ST_Transform	✓			✓	✓	✓	
ST_Translate	✓		✓	✓			
ST_TriangulatePoly	✓						✓
ST_UnaryUnion	✓		✓				
ST_Union	✓		✓		✓		
ST_Volume							
ST_VoronoiLines	✓						
ST_VoronoiPolygon	✓						
ST_WrapX	✓		✓				
ST_X	✓		✓		✓		
ST_XMax			✓	✓			
ST_XMin			✓	✓			
ST_Y	✓		✓		✓		
ST_YMax			✓	✓			
ST_YMin			✓	✓			
ST_Z	✓		✓		✓		
ST_ZMax			✓	✓			
ST_ZMin			✓	✓			
ST_Zmflag	✓		✓	✓			
postgis.backend							
postgis.enable_outdb_rasters							
postgis.gdal_datapath							
postgis.gdal_enabled_drivers							
postgis.gdal_vsi_options							
postgis_sfcgal_version							
postgis_sfcgal_version							

15.12 New, Enhanced or changed PostGIS Functions

15.12.1 PostGIS Functions new or enhanced in 3.3

The functions given below are PostGIS functions that were added or enhanced.

Functions new in PostGIS 3.3

- **RemoveUnusedPrimitives** - Availability: 3.3.0 Removes topology primitives which not needed to define existing TopoGeometry objects.
- **ST_3DUnion** - Availability: 3.3.0 aggregate variant was added Perform 3D union.
- **ST_AlphaShape** - Availability: 3.3.0 - requires SFCGAL >= 1.4.1. Computes a possible concave geometry using the CGAL Alpha Shapes algorithm.
- **ST_AsMARC21** - Availability: 3.3.0 Returns geometry as a MARC21/XML record with a geographic datafield (034).
- **ST_GeomFromMARC21** - Availability: 3.3.0, requires libxml2 2.6+ Takes MARC21/XML geographic data as input and returns a PostGIS geometry object.
- **ST_OptimalAlphaShape** - Availability: 3.3.0 - requires SFCGAL >= 1.4.1. Computes a possible concave geometry using the CGAL Alpha Shapes algorithm after have computed the "optimal" alpha value.
- **ST_SimplifyPolygonHull** - Availability: 3.3.0 - requires GEOS >= 3.11.0 Computes a simplified topology-preserving outer or inner hull of a polygonal geometry.
- **ST_TriangulatePolygon** - Availability: 3.3.0 - requires GEOS >= 3.11.0 Computes the constrained Delaunay triangulation of polygons

Functions enhanced in PostGIS 3.3

- **ST_ConcaveHull** - Enhanced: 3.3.0, GEOS native implementation enabled for GEOS 3.11+ Computes a possibly concave geometry that encloses all input geometry vertices
- **ST_LineMerge** - Enhanced: 3.3.0 accept a directed parameter - requires GEOS >= 3.11.0 Return the lines formed by sewing together a MultiLineString.

Functions changed in PostGIS 3.3

- **PostGIS_Extensions_Upgrade** - Changed: 3.3.0 support for upgrades from any PostGIS version. Does not work on all systems. Packages and upgrades PostGIS extensions (e.g. postgis_raster, postgis_topology, postgis_sfcgal) to latest available version.

15.12.2 PostGIS Functions new or enhanced in 3.2

The functions given below are PostGIS functions that were added or enhanced.

Functions new in PostGIS 3.2

- **FindLayer** - Availability: 3.2.0 Returns a topology.layer record by different means.
- **FindTopology** - Availability: 3.2.0 Returns a topology record by different means.
- **GetFaceContainingPoint** - Availability: 3.2.0 Finds the face containing a point.
- **ST_AsFlatGeobuf** - Availability: 3.2.0 Return a FlatGeobuf representation of a set of rows.
- **ST_FromFlatGeobuf** - Availability: 3.2.0 Reads FlatGeobuf data.
- **ST_FromFlatGeobufToTable** - Availability: 3.2.0 Creates a table based on the structure of FlatGeobuf data.

- **ST_SRID** - Availability: 3.2.0 Returns the spatial reference identifier for a topogeometry.
- **ST_Scroll** - Availability: 3.2.0 Change start point of a closed LineString.
- **TopoGeom_addTopoGeom** - Availability: 3.2 Adds element of a TopoGeometry to the definition of another TopoGeometry.
- **ValidateTopologyRelation** - Availability: 3.2.0 Returns info about invalid topology relation records

Functions enhanced in PostGIS 3.2

- **GetFaceByPoint** - Enhanced: 3.2.0 more efficient implementation and clearer contract, stops working with invalid topologies. Finds face intersecting a given point.
- **ST_ClusterKMeans** - Enhanced: 3.2.0 Support for max_radius Window function that returns a cluster id for each input geometry using the K-means algorithm.
- **ST_MakeValid** - Enhanced: 3.2.0, added algorithm options, 'linework' and 'structure' which requires GEOS >= 3.10.0. Attempts to make an invalid geometry valid without losing vertices.
- **ST_MoveIsoNode** - Enhanced: 3.2.0 ensures the node cannot be moved in a different face 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.
- **ST_Point** - Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with ST_SetSRID to mark the srid on the geometry. Creates a Point with X, Y and SRID values.
- **ST_Point** - Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with ST_SetSRID to mark the srid on the geometry. Creates a Point with X, Y, Z and SRID values.
- **ST_Point** - Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with ST_SetSRID to mark the srid on the geometry. Creates a Point with X, Y, M and SRID values.
- **ST_Point** - Enhanced: 3.2.0 srid as an extra optional argument was added. Older installs require combining with ST_SetSRID to mark the srid on the geometry. Creates a Point with X, Y, Z, M and SRID values.
- **ST_RemovePoint** - Enhanced: 3.2.0 Remove a point from a linestring.
- **ST_RemoveRepeatedPoints** - Enhanced: 3.2.0 Returns a version of a geometry with duplicate points removed.
- **ST_StartPoint** - Enhanced: 3.2.0 returns a point for all geometries. Prior behavior returns NULLs if input was not a LineString. Returns the first point of a LineString.

Functions changed in PostGIS 3.2

- **ST_Boundary** - Changed: 3.2.0 support for TIN, does not use geos, does not linearize curves
- **ValidateTopology** - Changed: 3.2.0 added optional bbox parameter, perform face labeling and edge linking checks. Returns a set of validate_topology_return_type objects detailing issues with topology.

15.12.3 PostGIS Functions new or enhanced in 3.1

The functions given below are PostGIS functions that were added or enhanced.

Functions new in PostGIS 3.1

- **ST_MaximumInscribedCircle** - Availability: 3.1.0 - requires GEOS >= 3.9.0.
- **ST_ReducePrecision** - Availability: 3.1.0 - requires GEOS >= 3.9.0. Returns a valid geometry with points rounded to a grid tolerance.

Functions enhanced in PostGIS 3.1

- **ST_AsEWKT** - Enhanced: 3.1.0 support for optional precision parameter. WKT(Well-Known Text) SRID
- **ST_ClusterKMeans** - Enhanced: 3.1.0 Support for 3D geometries and weights Window function that returns a cluster id for each input geometry using the K-means algorithm.
- **ST_Difference** - Enhanced: 3.1.0 accept a gridSize parameter - requires GEOS >= 3.9.0 Computes a geometry representing the part of geometry A that does not intersect geometry B.
- **ST_Intersection** - Enhanced: 3.1.0 accept a gridSize parameter - requires GEOS >= 3.9.0 Computes a geometry representing the shared portion of geometries A and B.
- **ST_MakeValid** - Enhanced: 3.1.0, added removal of Coordinates with NaN values. Attempts to make an invalid geometry valid without losing vertices.
- **ST_Subdivide** - Enhanced: 3.1.0 accept a gridSize parameter, requires GEOS >= 3.9.0 to use this new feature. Computes a rectilinear subdivision of a geometry.
- **ST_SymDifference** - Enhanced: 3.1.0 accept a gridSize parameter - requires GEOS >= 3.9.0 Computes a geometry representing the portions of geometries A and B that do not intersect.
- **ST_UnaryUnion** - Enhanced: 3.1.0 accept a gridSize parameter - requires GEOS >= 3.9.0 Computes the union of the components of a single geometry.
- **ST_Union** - Enhanced: 3.1.0 accept a gridSize parameter - requires GEOS >= 3.9.0 Computes a geometry representing the point-set union of the input geometries.

Functions changed in PostGIS 3.1

- **ST_Force3D** - Changed: 3.1.0. Added support for supplying a non-zero Z value. XYZ
- **ST_Force3DM** - Changed: 3.1.0. Added support for supplying a non-zero M value. XYM
- **ST_Force3DZ** - Changed: 3.1.0. Added support for supplying a non-zero Z value. XYZ
- **ST_Force4D** - Changed: 3.1.0. Added support for supplying non-zero Z and M values. XYZM
- **ST_Histogram** - Changed: 3.1.0 Removed ST_Histogram(table_name, column_name) variant. (bin; XYZ
- **ST_Quantile** - Changed: 3.1.0 Removed ST_Quantile(table_name, column_name) variant. (population; XYZ

15.12.4 PostGIS Functions new or enhanced in 3.0

The functions given below are PostGIS functions that were added or enhanced.

Functions enhanced in PostGIS 3.0

- **ST_AsMVT** - Enhanced: 3.0 - added support for Feature ID. Aggregate function returning a MVT representation of a set of rows.
- **ST_Contains** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if no points of B lie in the exterior of A, and A and B have at least one interior point in common.
- **ST_ContainsProperly** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if B intersects the interior of A but not the boundary or exterior.
- **ST_CoveredBy** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if no point in A is outside B
- **ST_Covers** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if no point in B is outside A
- **ST_Crosses** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if two geometries have some, but not all, interior points in common.
- **ST_CurveToLine** - Enhanced: 3.0.0 implemented a minimum number of segments per linearized arc to prevent topological collapse. Converts a geometry containing curves to a linear geometry.
- **ST_Disjoint** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if two geometries are disjoint (they have no point in common).
- **ST_Equals** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if two geometries include the same set of points.
- **ST_GeneratePoints** - Enhanced: 3.0.0, added seed parameter Generates random points contained in a Polygon or MultiPolygon.
- **ST_GeomFromGeoJSON** - Enhanced: 3.0.0 parsed geometry defaults to SRID=4326 if not specified otherwise. GeoJSON
- **ST_LocateBetween** - Enhanced: 3.0.0 - added support for POLYGON, TIN, TRIANGLE. Returns the portions of a geometry that match a measure range.
- **ST_LocateBetweenElevations** - Enhanced: 3.0.0 - added support for POLYGON, TIN, TRIANGLE. Returns the portions of a geometry that lie in an elevation (Z) range.
- **ST_Overlaps** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if two geometries intersect and have the same dimension, but are not completely contained by each other.
- **ST_Relate** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if two geometries have a topological relationship matching an Intersection Matrix pattern, or computes their Intersection Matrix
- **ST_Segmentize** - Enhanced: 3.0.0 Segmentize geometry now uses equal length segments
- **ST_Touches** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if two geometries have at least one point in common, but their interiors do not intersect.
- **ST_Within** - Enhanced: 3.0.0 enabled support for GEOMETRYCOLLECTION Tests if no points of A lie in the exterior of B, and A and B have at least one interior point in common.

Functions changed in PostGIS 3.0

- **PostGIS_Extensions_Upgrade** - Changed: 3.0.0 to repackage loose extensions and support postgis_raster. Packages and upgrades PostGIS extensions (e.g. postgis_raster, postgis_topology, postgis_sfcgal) to latest available version.

- **ST_3DDistance** - Changed: 3.0.0 - SFCGAL version removed. Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area).
- **ST_3DIntersects** - Changed: 3.0.0 SFCGAL backend removed, GEOS backend supports TINs. Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area).
- **ST_Area** - Changed: 3.0.0 - does not depend on SFCGAL anymore. Computes a geometry representing the area of the input geometries.
- **ST_AsGeoJSON** - Changed: 3.0.0 support records as input Return a geometry as a GeoJSON element.
- **ST_AsGeoJSON** - Changed: 3.0.0 output SRID if not EPSG:4326. Return a geometry as a GeoJSON element.
- **ST_AsKML** - Changed: 3.0.0 - Removed the "versioned" variant signature. Returns a KML document representing the input geometry.
- **ST_Distance** - Changed: 3.0.0 - does not depend on SFCGAL anymore. Computes the distance between two geometries in 2D.
- **ST_Intersection** - Changed: 3.0.0 does not depend on SFCGAL. Computes a geometry representing the shared portion of geometries A and B.
- **ST_Intersects** - Changed: 3.0.0 SFCGAL version removed and native support for 2D TINs added. Tests if two geometries intersect (they have at least one point in common).
- **ST_Union** - Changed: 3.0.0 does not depend on SFCGAL. Computes a geometry representing the point-set union of the input geometries.

15.12.5 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 Packages and upgrades PostGIS extensions (e.g. postgis_raster, postgis_topology, postgis_sfcgal) to latest available version.
- **ST_Angle** - Availability: 2.5.0 Returns the angle in degrees between two line segments. If no bandnum specified, 1 is assumed.
- **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_ChaikinSmoothing** - Availability: 2.5.0 Returns a smoothed version of a geometry, using the Chaikin algorithm
- **ST_FilterByM** - Availability: 2.5.0 Removes vertices based on their M value
- **ST_Grayscale** - Availability: 2.5.0 Creates a new one-band raster from the source raster and specified bands representing Red, Green and Blue
- **ST_LineInterpolatePoints** - Availability: 2.5.0 Returns points interpolated along a line at a fractional interval.
- **ST_OrientedEnvelope** - Availability: 2.5.0 Returns a minimum-area rectangle containing a geometry.
- **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_AsMVT** - Enhanced: 2.5.0 - added support parallel query. Aggregate function returning a MVT representation of a set of rows.
- **ST_AsText** - Enhanced: 2.5 - optional parameter precision introduced. WKT(Well-Known Text) SRID
- **ST_BandMetaData** - Enhanced: 2.5.0 to include outdbbandnum, filesize and filetimestamp for outdb rasters.
- **ST_Buffer** - Enhanced: 2.5.0 - ST_Buffer geometry support was enhanced to allow for side buffering specification side=both|left|right. Computes a geometry covering all points within a given distance from a geometry.
- **ST_GeomFromGeoJSON** - Enhanced: 2.5.0 can now accept json and jsonb as inputs. GeoJSON
- **ST_GeometricMedian** - Enhanced: 2.5.0 Added support for M as weight of points. (median)
- **ST_Intersects** - Enhanced: 2.5.0 Supports GEOMETRYCOLLECTION. Tests if two geometries intersect (they have at least one point in common).
- **ST_OffsetCurve** - Enhanced: 2.5 - added support for GEOMETRYCOLLECTION and MULTILINESTRING Returns an offset line at a given distance and side from an input line.
- **ST_Scale** - Enhanced: 2.5.0 support for scaling relative to a local origin (origin parameter) was introduced. Scales a geometry by given factors.
- **ST_Split** - Enhanced: 2.5.0 support for splitting a polygon by a multiline was introduced. Returns a collection of geometries created by splitting a geometry by another geometry.
- **ST_Subdivide** - Enhanced: 2.5.0 reuses existing points on polygon split, vertex count is lowered from 8 to 5. Computes a rectilinear subdivision of a geometry.

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

15.12.6 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 Aggregate function returning a MVT representation of a set of rows.
- **ST_AsMVTGeom** - Availability: 2.4.0 Transforms a geometry into the coordinate space of a MVT tile.
- **ST_Centroid** - Availability: 2.4.0 support for geography was introduced.
- **ST_ForcePolygonCCW** - Availability: 2.4.0 Orients all exterior rings counter-clockwise and all interior rings clockwise.
- **ST_ForcePolygonCW** - Availability: 2.4.0 Orients all exterior rings clockwise and all interior rings counter-clockwise.
- **ST_FrechetDistance** - Availability: 2.4.0 - requires GEOS >= 3.7.0
- **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.
- **Normalize_Address** - Enhanced: 2.4.0 norm_addy object includes additional fields zip4 and address_alphanumeric.
- **Page_Normalize_Address** - Enhanced: 2.4.0 norm_addy object includes additional fields zip4 and address_alphanumeric.
- **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.

- **ST_AsTWKB** - Enhanced: 2.4.0 memory and speed improvements. TWKB(Tiny Well-Known Binary)
- **ST_Covers** - Enhanced: 2.4.0 Support for polygon in polygon and line in polygon added for geography type Tests if no point in B is outside A
- **ST_CurveToLine** - Enhanced: 2.4.0 added support for max-deviation and max-angle tolerance, and for symmetric output. Converts a geometry containing curves to a linear geometry.
- **ST_Project** - Enhanced: 2.4.0 Allow negative distance and non-normalized azimuth.
- **ST_Reverse** - Enhanced: 2.4.0 support for curves was introduced.

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. Nodes a collection of lines.

15.12.7 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 4.9.2.



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_ClusterDBSCAN** - Availability: 2.3.0 Window function that returns a cluster id for each input geometry using the DBSCAN algorithm.
- **ST_ClusterKMeans** - Availability: 2.3.0 Window function that returns a cluster id for each input geometry using the K-means algorithm.
- **ST_WrapX** - Availability: 2.3.0 requires GEOS X; requires GEOS 3.10.0 or later.
- **~(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 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 contains a 2D float precision bounding box (GIDX).

The functions given below are PostGIS functions that are enhanced in PostGIS 2.3.

- **ST_Contains** - Enhanced: 2.3.0 Enhancement to PIP short-circuit extended to support MultiPoints with few points. Prior versions only supported point in polygon.
- **ST_Covers** - Enhanced: 2.3.0 Enhancement to PIP short-circuit for geometry extended to support MultiPoints with few points. Prior versions only supported point in polygon.
- **ST_Expand** - Enhanced: 2.3.0 support was added to expand a box by different amounts in different dimensions.
- **ST_Intersects** - Enhanced: 2.3.0 Enhancement to PIP short-circuit extended to support MultiPoints with few points. Prior versions only supported point in polygon.
- **ST_Segmentize** - Enhanced: 2.3.0 Segmentize geography now uses equal length segments
- **ST_Transform** - Enhanced: 2.3.0 support for direct PROJ.4 text was introduced.
- **ST_Within** - Enhanced: 2.3.0 Enhancement to PIP short-circuit for geometry extended to support MultiPoints with few points. Prior versions only supported point in polygon.

15.12.8 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 Section [14.1](#) for details.



Note

Many functions in topology rewritten as C functions for increased performance.

Functions new in PostGIS 2.2

- **ST_CPAWithin** - Availability: 2.2.0 Tests if the closest point of approach of two trajectories is within the specified distance.
- **ST_ClipByBox2D** - Availability: 2.2.0 Computes the portion of a geometry falling within a rectangle.
- **ST_ClosestPointOfApproach** - Availability: 2.2.0 Returns a measure at the closest point of approach of two trajectories.
- **ST_ClusterIntersecting** - Availability: 2.2.0 Aggregate function that clusters the input geometries into connected sets.
- **ST_ClusterWithin** - Availability: 2.2.0 Aggregate function that clusters the input geometries by separation distance.
- **ST_DistanceCPA** - Availability: 2.2.0 Returns the distance between the closest point of approach of two trajectories.
- **ST_IsValidTrajectory** - Availability: 2.2.0 Tests if the geometry is a valid trajectory.
- **ST_Subdivide** - Availability: 2.2.0 Computes a rectilinear subdivision of a geometry.

The functions given below are PostGIS functions that are enhanced in PostGIS 2.2.

- **ST_Scale** - Enhanced: 2.2.0 support for scaling all dimension (factor parameter) was introduced.
- **ST_Split** - Enhanced: 2.2.0 support for splitting a line by a multiline, a multipoint or (multi)polygon boundary was introduced.

15.12.9 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
- **ST_MemSize** - Changed: 2.2.0 name changed to ST_MemSize to follow naming convention.
- **ST_PointInsideCircle** - Changed: 2.2.0 In prior versions this was called ST_Point_Inside_Circle

15.12.10 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 10 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 12 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 14.2 for more details.

The functions given below are PostGIS functions that are enhanced in PostGIS 2.1.

- **ST_DWithin** - Enhanced: 2.1.0 improved speed for geography. See Making Geography faster for details.
- **ST_DWithin** - Enhanced: 2.1.0 support for curved geometries was introduced.
- **ST_NumPoints** - Enhanced: 2.1.0 Faster speed. Reimplemented as native-C.
- **ST_MakeValid** - Enhanced: 2.1.0, added support for GEOMETRYCOLLECTION and MULTIPOINT.

15.12.11 PostGIS functions breaking changes in 2.1

The functions given below are PostGIS functions that have possibly breaking changes in PostGIS 2.1. If you use any of these, you may need to check your existing code.

- **ST_EstimatedExtent** - Changed: 2.1.0. Up to 2.0.x this was called `ST_Estimated_Extent`.

15.12.12 PostGIS Functions new, behavior changed, or enhanced in 2.0

The functions given below are PostGIS functions that were added, enhanced, or have Section 15.12.13 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 10 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 12 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 [14.2](#) for more details.

- **ST_3DDFullyWithin** - Availability: 2.0.0 Tests if two 3D geometries are entirely within a given 3D distance
- **ST_3DDWithin** - Availability: 2.0.0 Tests if two 3D geometries are within a given 3D distance
- **ST_3DIntersects** - Availability: 2.0.0 Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area).
- **ST_IsValidDetail** - Availability: 2.0.0 Returns a valid_detail row stating if a geometry is valid or if not a reason and a location.
- **ST_IsValidReason** - Availability: 2.0 version taking flags. Returns text stating if a geometry is valid, or a reason for invalidity.
- **ST_MakeValid** - Availability: 2.0.0 Attempts to make an invalid geometry valid without losing vertices.
- **ST_Node** - Availability: 2.0.0 Nodes a collection of lines.
- **ST_RelateMatch** - Availability: 2.0.0 Tests if a DE-9IM Intersection Matrix matches an Intersection Matrix pattern
- **ST_Split** - Availability: 2.0.0 requires GEOS Returns a collection of geometries created by splitting a geometry by another geometry.
- **ST_UnaryUnion** - Availability: 2.0.0 Computes the union of the components of a single geometry.

The functions given below are PostGIS functions that are enhanced in PostGIS 2.0.

- **Box2D** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **Box3D** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_3DExtent** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_Affine** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_Expand** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_Extent** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_MakeValid** - Enhanced: 2.0.1, speed improvements
- **ST_Relate** - Enhanced: 2.0.0 - added support for specifying boundary node rule.
- **ST_Rotate** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_Rotate** - Enhanced: 2.0.0 additional parameters for specifying the origin of rotation were added.
- **ST_RotateX** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_RotateY** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_RotateZ** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_Scale** - Enhanced: 2.0.0 support for Polyhedral surfaces, Triangles and TIN was introduced.
- **ST_Transform** - Enhanced: 2.0.0 support for Polyhedral surfaces was introduced.

15.12.13 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.

- **ST_3DExtent** - Changed: 2.0.0 In prior versions this used to be called ST_Extent3D
- **ST_3DMakeBox** - Changed: 2.0.0 In prior versions this used to be called ST_MakeBox3D

15.12.14 PostGIS Functions new, behavior changed, or enhanced in 1.5

The functions given below are PostGIS functions that were introduced or enhanced in this minor release.

- **PostGIS_LibXML_Version** - Availability: 1.5 Returns the version number of the libxml2 library.
- **ST_Buffer** - Availability: 1.5 - ST_Buffer was enhanced to support different endcaps and join types. These are useful for example to convert road linestrings into polygon roads with flat or square edges instead of rounded edges. Thin wrapper for geography was added. Computes a geometry covering all points within a given distance from a geometry.
- **ST_Covers** - Availability: 1.5 - support for geography was introduced. Tests if no point in B is outside A
- **ST_DFullyWithin** - Availability: 1.5.0 Tests if two geometries are entirely within a given distance
- **ST_DWithin** - Availability: 1.5.0 support for geography was introduced Tests if two geometries are within a given distance
- **ST_Expand** - Availability: 1.5.0 behavior changed to output double precision instead of float4 coordinates. Returns a bounding box expanded from another bounding box or a geometry.
- **ST_GeomFromKML** - Availability: 1.5, requires libxml2 2.6+ 도형의 KML 표현식입력받아 PostGIS 도형 객체를 출력합입력받아
- **ST_Intersection** - Availability: 1.5 support for geography data type was introduced. Computes a geometry representing the shared portion of geometries A and B.
- **ST_Intersects** - Availability: 1.5 support for geography was introduced. Tests if two geometries intersect (they have at least one point in common).

15.12.15 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.

- **ST_ContainsProperly** - Tests if B intersects the interior of A but not the boundary or exterior. Availability: 1.4.0
- **ST_IsValidReason** - Returns text stating if a geometry is valid, or a reason for invalidity. Availability: 1.4
- **ST_LineCrossingDirection** - Returns a number indicating the crossing behavior of two LineStrings. Availability: 1.4
- **ST_Union** - Computes a geometry representing the point-set union of the input geometries. Availability: 1.4.0 - ST_Union was enhanced. ST_Union(geomarray) was introduced and also faster aggregate collection in PostgreSQL.

15.12.16 PostGIS Functions new in 1.3

The functions given below are PostGIS functions that were introduced in the 1.3 release.

- **ST_CurveToLine** - Converts a geometry containing curves to a linear geometry. Availability: 1.3.0
 - **ST_LineToCurve** - Converts a linear geometry to a curved geometry. Availability: 1.3.0
-

Chapter 16

Reporting Problems

16.1 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.

16.2 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 patch is definitely preferred. This is a four step process on Unix (assuming you already have [git](#) installed):

1. Clone the PostGIS' git repository. On Unix, type:

```
git clone https://git.osgeo.org/gitea/postgis/postgis.git
```

This will be stored in the directory `postgis`

2. Make your changes to the documentation with your favorite text editor. On Unix, type (for example):

```
vim 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:

```
git diff doc/postgis.xml > doc.patch
```

4. Attach the patch to a new issue in bug tracker.

Appendix A

Appendix

A.1 PostGIS 3.3.8

2024/12/22

PostgreSQL 11-17 required. GEOS 3.6+ required but GEOS 3.11+ to take advantage of all features. Proj 4.9+ required.

A.1.1 Bug Fixes

[5785](#), [raster] ST_MapAlgebra segfaults when expression references a supernumerary rast argument (Dian M Fay)

[5787](#), [topology] Check that ST_ChangeEdgeGeom doesn't change winding of rings (Sandro Santilli)

[5795](#), [topology] Fix ST_NewEdgesSplit can cause invalid topology (Björn Harrtell)

[5677](#), Retain SRID during unary union (Paul Ramsey)

A.2 PostGIS 3.3.7

2024/09/05

A.2.1 Bug Fixes

[5766](#), Always report invalid non-null MBR of universal face (Sandro Santilli)

[5709](#), Fix loose mbr in topology.face on ST_ChangeEdgeGeom (Sandro Santilli)

[5698](#), Fix robustness issue splitting line by vertex very close to endpoints, affecting topology population functions (Sandro Santilli)

[5649](#), ST_Value should return NULL on missing band (Paul Ramsey)

[5677](#), ST_Union(geom[]) should unary union single entry arrays (Paul Ramsey)

[5679](#), Remove spurious COMMIT statements from sfcgal script (Sandro Santilli, Loïc Bartoletti)

[5680](#), Fix populate_topology_layer with standard_conforming_strings set to off (Sandro Santilli)

[5589](#), ST_3DDistance error for shared first point (Paul Ramsey)

[5686](#), ST_NumInteriorRings and Triangle crash (Paul Ramsey)

[5671](#), Bug in ST_Area function with use_spheroid=false (Paul Ramsey, Regina Obe)

- 5687, #5756 Support for PostgreSQL 17, revise `postgis_get_full_version_schema()` to not rely on `search_path`
- 5740, `ST_DistanceSpheroid(geometry)` incorrectly handles polygons (Paul Ramsey)
- 5765, Handle nearly co-linear edges with slightly less slop (Paul Ramsey)
- 5745, `St_AsLatLonText` rounding errors (Paul Ramsey)

A.3 PostGIS 3.3.6

2024/02/07

This version requires PostgreSQL 11-16, GEOS 3.6 or higher, and Proj 4.9+. Additional features are enabled if you are running GEOS 3.9+ (and `ST_MakeValid` enhancements with 3.10+), Proj 6.1+, and PostgreSQL 15+.

A.3.1 Bug Fixes

- 5633, Fix `postgis` load, upgrade and usage with `standard_conforming_strings` set to off (Sandro Santilli, Regina Obe)
- 5571, Memory over-allocation for narrow inputs (Paul Ramsey)
- 5610, Allow Nan and infinity again in `ST_SetPoint` (Regina Obe)
- 5627, Handling of EMPTY components in PiP check (Paul Ramsey)
- 5629, Handling EMPTY components in repeated point removal (Paul Ramsey)
- 5604, Handle distance between collections with empty elements (Paul Ramsey)
- 5635, Handle NaN points in `ST_Split` (Regina Obe)
- Logic error in `ST_Covers(geometry)` (Paul Ramsey)
- 5646, Crash on collections with empty members (Paul Ramsey)
- 5580, Handle empty collection components in 3d distance (Paul Ramsey)
- 5639, `ST_DFullyWithin` line/poly error case (Paul Ramsey)
- 5662, Change XML parsers to SAX2 (Paul Ramsey)

A.4 PostGIS 3.3.5

2023/11/19

A.4.1 Bug Fixes and Enhancements

- 5598, Fix restore of 3.3 dumps (Sandro Santilli)
 - 5568, Improve robustness of topology face split handling (Sandro Santilli)
 - 5548, Fix box-filtered validity check of topologies with edge-less faces (Sandro Santilli)
 - 5442, 5525, [`postgis_tiger_geocoder`, `postgis_topology`] Database `search_path` does not do what it intends to do (Jelte Fennema, Sandro Santilli, Regina Obe)
 - 5494, Fix double-upgrade with view using `st_dwithin(text, ...)` (Sandro Santilli)
 - Fix `JsonB` casting issue (Paul Ramsey)
 - 5616, Fix to find docbook on newer docbook installs (Regina Obe)
-

A.5 PostGIS 3.3.4

2023/07/28

This version requires PostgreSQL 11-16, GEOS 3.6 or higher, and Proj 4.9+ Additional features are enabled if you are running GEOS 3.9+ (and ST_MakeValid enhancements with 3.10+), Proj 6.1+, and PostgreSQL 15+.

A.5.1 Bug Fixes and Enhancements

- 5450, Fix macro expansion recursion on powerpc architectures. (Bas Couwenberg)
- 5395, [postgis_topology] Allow unprivileged user dump of database with topology enabled (Sandro Santilli)
- 5394, [postgis_topology] Improve robustness of finding distinct vertices in topology edges (Sandro Santilli)
- 5403, [postgis_topology] Fix ValidateTopology(bbox) without topology in search_path (Sandro Santilli)
- 5410, [postgis_raster] ST_Value bilinear resample, don't throw an error if Band has no NODATA value (Regina Obe)
- 5385, Postgres malloc assertion fail when using pg_cancel_backend with ST_AsMVT (Regina Obe, Paul Ramsey)
- 5452, Updated README.postgis for apt-based systems (Jelte Fennema)
- 5385, Take out interruptability of ST_AsMVT as it causes backend crash under intense conditions (Regina Obe, Paul Ramsey)

A.6 PostGIS 3.3.3

2023/05/29

This version requires PostgreSQL 11-16beta1, GEOS 3.6 or higher, and Proj 4.9+ Additional features are enabled if you are running GEOS 3.9+ (and ST_MakeValid enhancements with 3.10+), Proj 6.1+, and PostgreSQL 15+.

A.6.1 Bug Fixes and Enhancements

- 5384, Fix crash in ST_AsGML when given id is longer than given prefix (Sandro Santilli)
 - 5380, Fix 2.5 upgrades with views using geography based ST_Distance (Sandro Santilli)
 - 5303, Don't try removing closed edges endpoint from RemoveUnusedPrimitives (Sandro Santilli)
 - 5289, Fix misleading message in RemoveUnusedPrimitives about doubly connected edges healing (Sandro Santilli)
 - 5298, Fix CopyTopology exception with hierarchical layers (Sandro Santilli)
 - 5299, Corrections to address_standardizer_data_us lex (Regina Obe)
 - 5106, Fix segfault in ST_RemEdgeNewFace/ST_RemEdgeModFace when no edge side-faces are found (Sandro Santilli)
 - 5329, Fix downgrade protection introduced in 3.3.0 (Sandro Santilli)
 - 5332, Keep proj errors from percolating into PgSQL log (Paul Ramsey)
 - 5313, Fix memory access issue in ST_InterpolateRaster (Paul Ramsey)
 - 5338, Dump/Restore of raster table fails on enforce_coverage_tile_rast constraint (Regina Obe)
 - 5315, #5318, #5319, #5320, #5342 crashes on infinite coordinates (Regina Obe, Paul Ramsey)
 - 5344, Include topology id sequence state in dumps (Sandro Santilli)
 - 5288, ST_LineCrossingDirection multi-cross error (Paul Ramsey)
 - 5347, _ST_BestSRID crashes on ARM with infinite geometries (Regina Obe)
 - 5331, [postgis_tiger_geocoder] reverse_geocode, prefer addressable ranges (Regina Obe, Locance)
-

- 5363, Make ST_ClusterDBScan interruptable (Paul Ramsey)
- 5257, #5261, #5277, #5304, #5308, #5374, Support for PG16 (Paul Ramsey, Regina Obe, Sandro , Laurenz Albe)
- 5371, ST_Union segfaults (PG 11-12) (Regina Obe)
- 5378, ST_Buffer(geography) drops SRID (Paul Ramsey)
- 5388, [postgis_tiger_geocoder] Fix faces table tiger_gload, caused census rerelease of faces data (Regina Obe)

A.7 PostGIS 3.3.2

This version requires PostgreSQL 11-15, GEOS 3.6 or higher, and Proj 4.9+ Additional features are enabled if you are running GEOS 3.9+ (and ST_MakeValid enhancements with 3.10+), Proj 6.1+, and PostgreSQL 15+.

A.7.1 Bug and Security Fixes

- 5248, Marc21 documentation missing (Regina Obe)
- 5240, ST_DumpPoints and ST_DumpSegments crash with empty polygon (Regina Obe)
- [security] Add schema qual to upgrade util (Regina Obe)
- 5255, GH598, Remove forced static linking on shp2pgsql, pgsq2shp (Maxim Kochetkov)
- 5255, GH702, Remove forced static linking on raster2pgsql (Thomas Petazzoni)
- 4648, [security] Check function ownership at extension packaging time (Sandro Santilli) Thanks to Sven Klemm (Timescale) for the report
- 5241, Crash on ST_SnapToGrid with empty multis (Regina Obe)
- 5081, Fix error in topology import of self-intersecting rings (Sandro Santilli)
- 5234, Fix 2.5d topology building regression (Sandro Santilli)
- 5280, Handle load of dbase character fields with no width specified (Regina Obe)
- 5264, Fix topology install on Alpine (Louis Descoteaux)
- 5244, postgis_tiger_geocoder update for TIGER 2022 (Regina Obe)
- 5285, Fix import/export of topologies with names needing quotes (Sandro Santilli)
- 4988, SFCGAL specify name of unknown type when SFCGAL can't process (Regina Obe)
- 5084, Bad rasterization of linestring (Gilles Vuidel)

A.8 PostGIS 3.3.1

2022/09/09

This version requires PostgreSQL 11 - 15, GEOS 3.6 or higher, and Proj 5.2+. Additional features are enabled if you are running GEOS 3.9+ ST_MakeValid enhancements with 3.10+, numerous additional enhancements with GEOS 3.11+. Requires SFCGAL 1.4.1+ for ST_AlphaShape and ST_OptimalAlphaShape.

A.8.1 Bug Fixes

- 5227, typo in ST_LineLocatePoint error message (Sandro Santilli)
- 5231, PG15 no longer compiles because SQL/JSON removed PG upstream (Regina Obe)

A.9 PostGIS 3.3.0

2022/08/26

This version requires PostgreSQL 11 or higher, GEOS 3.6 or higher, and Proj 5.2+. Additional features are enabled if you are running GEOS 3.9+ ST_MakeValid enhancements with 3.10+, numerous additional enhancements with GEOS 3.11+. Requires SFCGAL 1.4.1+ for ST_AlphaShape and ST_OptimalAlphaShape.

NOTE: GEOS 3.11.0 details at [GEOS 3.11.0 release notes](#)

The new configure --enable-lto flag improves speed of math computations. This new feature is disabled by default because on some platforms, causes compilation errors (BSD and MingW64 issues have been raised)

A.9.1 New features

[5116](#), Topology export/import scripts (Sandro Santilli)

ST_Letters creates geometries that look like letters (Paul Ramsey)

[5037](#), postgis_sfcgal: ST_3DConvexHull (Loïc Bartoletti)

postgis_sfcgal: sfcgal_full_version - reports BOOST and CGAL version (Loïc Bartoletti)

[GH 659](#), MARC21/XML, ST_GeomFromMARC21, ST_AsMARC21 (Jim Jones)

[5132](#), [GH 683](#), sfcgal: ST_3DUnion aggregate function (Sergei Shoulbakov)

[5143](#), SFCGAL ST_AlphaShape and ST_OptimalAlphaShape Requires SFCGAL 1.4.1+ (Loïc Bartoletti)

[5162](#), ST_TriangulatePolygon with GEOS 3.11+ (Paul Ramsey, Martin Davis)

[5162](#), ST_SimplifyPolygonHull with GEOS 3.11+ (Paul Ramsey, Martin Davis)

[5183](#), topology.RemoveUnusedPrimitives (Sandro Santilli)

A.9.2 Breaking Changes

Drop support for PostgreSQL 9.6 and 10 (Regina Obe)

Change output for WKT MULTIPOINT. All points now wrapped in parens. (Even Rouault)

[GH 674](#), geometry validation and fixing is disabled for ST_DumpAsPolygons and ST_Polygon so it works faster but might produce invalid polygons. (Aliaksandr Kalenik)

A.9.3 Enhancements

[2861](#), Add index on topology.node(containing_face) speeding up splitting and merging of faces (Sandro Santilli)

[2083](#), Speed up ST_RemEdge topology functions adding index on relation(element_id) and edge_data(abs_next*) (Sandro Santilli)

[5118](#), Allow dropping topologies with missing topogeometry sequences (Sandro Santilli)

[5111](#), faster topology face MBR computation (Sandro Santilli)

postgis_extensions_upgrade() support for upgrades from any PostGIS version, including yet to be released ones (Sandro Santilli)

[5040](#), add postgis_sfcgal_full_version (Loïc Bartoletti)

[GH 655](#), GiST: balance the tree splits better in recursive calls (Darafei Praliaskouski)

[GH 657](#), GiST: do not call no-op decompress function (Aliaksandr Kalenik)

[4939](#), [5161](#), ST_LineMerge now has option to keep the directions of input linestrings, useful when processing road graphs. Requires GEOS 3.11. (Sergei Shoulbakov)

ST_ConcaveHull GEOS 3.11+ native implementation (Paul Ramsey, Martin Davis)

ST_ConcaveHull GEOS 3.11+ polygon-respecting native implementation (Paul Ramsey, Martin Davis)

[4574](#), [GH 678](#), [5121](#) Enable Link-Time Optimizations using `--enable-lto` (Sergei Shoulbakov)

[GH 676](#), faster ST_Clip (Aliaksandr Kalenik)

[5135](#), Fast GiST index build is enabled by default for PostgreSQL 15+ (Sergei Shoulbakov)

[4939](#), [5161](#), ST_LineMerge now has option to keep the directions of input linestrings, useful when processing road graphs. Requires GEOS 3.11. (Sergei Shoulbakov)

[5158](#), pgtopo_import / pgtopo_export manpages (Sandro Santilli)

[5170](#), add a optional max_rows_per_copy to -Y option to raster2pgsql to control number of rows per copy statement. Default to 50 when not specified (Regina Obe)

[GH 698](#), support parallel aggregate for ST_Union (Sergei Shoulbakov)

[5024](#), Update spatial_ref_sys as part of ALTER EXTENSION update postgis (Paul Ramsey)

A.9.4 Bug Fixes

These are fixes issues in prior minors not backported

[4912](#), GiST: fix crash on STORAGE EXTERNAL for geography (Aliaksandr Kalenik)

[5088](#), Memory corruption in mvt_agg_transfn (Victor Collod)

[5137](#), resetting interrupt flags before query execution (Sergei Shoulbakov)

[5148](#), ST_Clip is more robust to alignment of raster and clip geometry (Sergei Shoulbakov)

[4932](#), Bug with geography ST_Intersects / ST_Distance (Paul Ramsey)

[5089](#), ST_Reverse also reverses components of CompoundCurve (Paul Ramsey)

A.10 PostGIS 3.3.0rc2

2022/08/22

This version requires PostgreSQL 11 or higher, GEOS 3.6 or higher, and Proj 5.2+. Additional features are enabled if you are running GEOS 3.9+ ST_MakeValid enhancements with 3.10+, numerous additional enhancements with GEOS 3.11+. Requires SFCGAL 1.4.1+ for ST_AlphaShape and ST_OptimalAlphaShape.

NOTE: GEOS 3.11.0 was recently released, details at [GEOS 3.11.0 release notes](#)

The new `--enable-lto` flag improves speed of math computations. This new feature is disabled by default because on some platforms, causes compilation errors (BSD and MingW64 issues have been raised)

A.10.1 Bug Fixes

[5089](#), ST_Reverse also reverses components of CompoundCurve (Paul Ramsey)

[5181](#), Reset proj error state after failed parse (Paul Ramsey)

[5171](#), Short circuit geodesic distance when inputs equal (Paul Ramsey)

A.11 PostGIS 3.3.0rc1

2022/08/08

This version requires PostgreSQL 11 or higher, GEOS 3.6 or higher, and Proj 5.2+. Additional features are enabled if you are running GEOS 3.9+ ST_MakeValid enhancements with 3.10+, numerous additional enhancements with GEOS 3.11+. Requires SFCGAL 1.4.1+ for ST_AlphaShape and ST_OptimalAlphaShape.

NOTE: GEOS 3.11.0 was recently released, details at [GEOS 3.11.0 release notes](#)

The new `--enable-lto` flag improves speed of math computations. This new feature is disabled by default because on some platforms, causes compilation errors (BSD and MingW64 issues have been raised)

Use below to enable it.

```
./configure --enable-lto
```

Changes since PostGIS 3.3.0beta2:

A.11.1 Bug Fixes

[5154](#), raster ST_Value is undercosted (Regina Obe)

[5157](#), Revise minimum_bounding_circle Cunit test to be tolerant of small 32-bit floating point differences (Regina Obe)

[5191](#), Functions should use integer instead of int4 (Regina Obe)

[5139](#), PostGIS causes to_jsonb to no longer be parallel safe, ST_AsGeoJSON and ST_AsGML are also parallel unsafe (Regina Obe, Paul Ramsey)

[5025](#), Ensure that additional operators are not appended when the function and opfamily disagree about dimensionality (Paul Ramsey)

[5195](#), #5196 Change address_standardizer and postgis_tiger_geocoder CREATE EXTENSION to use CREATE instead of CREATE OR REPLACE. (Regina Obe)

[5202](#), Guard against downgrade (Sandro Santilli)

[5104](#), postgis_extensions_upgrade() fails with pgextwlist (Regina Obe)

A.12 PostGIS 3.3.0beta2

2022/07/13

This version requires PostgreSQL 11 or higher, GEOS 3.6 or higher, and Proj 5.2+. Additional features are enabled if you are running GEOS 3.9+ ST_MakeValid enhancements with 3.10+, numerous additional enhancements with GEOS 3.11+. Requires SFCGAL 1.4.1+ for ST_AlphaShape and ST_OptimalAlphaShape.

NOTE: GEOS 3.11.0 was recently released, details at [GEOS 3.11.0 release notes](#)

The new `--enable-lto` flag improves speed of math computations. This new feature is disabled by default because on some platforms, causes compilation errors (BSD and MingW64 issues have been raised)

Use below to enable it.

```
./configure --enable-lto
```

Changes since PostGIS 3.3.0beta1:

A.12.1 New Features

[5183](#), topology.RemoveUnusedPrimitives (Sandro Santilli)

A.12.2 Enhancements

[GH698](#), support parallel aggregate for ST_Union (Sergei Shoulbakov)

A.12.3 Bug Fixes

[5179](#), postgresql2shp syntax error on big-endian (Bas Couwenberg)

A.13 PostGIS 3.3.0beta1

2022/07/03

This version requires PostgreSQL 11 or higher, GEOS 3.6 or higher, and Proj 5.2+. Additional features are enabled if you are running GEOS 3.9+ ST_MakeValid enhancements with 3.10+, numerous additional enhancements with GEOS 3.11+.

Requires SFCGAL 1.4.1+ for ST_AlphaShape and ST_OptimalAlphaShape.

NOTE: GEOS 3.11.0 was recently released, details at [GEOS 3.11.0 release notes](#)

The new --enable-lto flag improves math computations. This new feature is disabled by default because on some platforms, causes compilation errors (BSD and MingW64 issues have been raised)

Use below to enable it.

```
./configure --enable-lto
```

A.13.1 Enhancements

[5158](#), pgtopo_import / pgtopo_export manpages (Sandro Santilli)

[5170](#), add a optional max_rows_per_copy to -Y option to raster2pgsql to control number of rows per copy statement. Default to 50 when not specified (Regina Obe)

[4939](#), [5161](#), ST_LineMerge now has option to keep the directions of input linestrings, useful when processing road graphs. Requires GEOS 3.11. (Sergei Shoulbakov)

ST_ConcaveHull GEOS 3.11+ polygon-respecting native implementation (Paul Ramsey, Martin Davis)

[5039](#), postgis_tiger_geocoder TIGER 2021 (Regina Obe)

A.13.2 New features

>[5169](#), ST_SimplifyPolygonHull (requires GEOS 3.11) (Paul Ramsey, Martin Davis)

[5162](#), ST_TriangulatePolygon with GEOS 3.11+ (Paul Ramsey, Martin Davis)

A.13.3 Bug Fix

[5173](#) st_asflatgeobuf detoast crash (Paul Ramsey)

[4932](#), Bug with geography ST_Intersects / ST_Distance (Paul Ramsey)

[5114](#), postgresql2shp segfault with long or many truncated columns

A.14 PostGIS 3.3.0alpha1

2022/05/21

This version requires PostgreSQL 11 or higher, GEOS 3.6 or higher, and Proj 4.9+. Additional features are enabled if you are running GEOS 3.9+ (precision feature of many processing functions) ST_MakeValid enhancements with 3.10+, ST_ConcaveHull native GEOS implementation with GEOS 3.11+

Requires SFCGAL 1.4.1+ for ST_AlphaShape and ST_OptimalAlphaShape.

The new `--enable-lto` flag improves math computations. This new feature is disabled by default because on some platforms, causes compilation errors (BSD and MingW64 issues have been raised)

Use below to enable it.

```
./configure --enable-lto
```

A.14.1 Breaking changes

Drop support for PostgreSQL 9.6 and 10 (Regina Obe)

Change output for WKT MULTIPOINT. All points now wrapped in parens. (Even Roualt)

GH674, geometry validation and fixing is disabled for ST_DumpAsPolygons and ST_Polygon so it works faster but might produce invalid polygons. (Aliaksandr Kalenik)

A.14.2 Enhancements

2861, Add index on topology.node(containing_face) speeding up splitting and merging of faces (Sandro Santilli)

2083, Speed up ST_RemEdge topology functions adding index on relation(element_id) and edge_data(abs_next*) (Sandro Santilli)

5118, Allow dropping topologies with missing topogeometry sequences (Sandro Santilli)

5111, faster topology face MBR computation (Sandro Santilli)

postgis_extensions_upgrade() support for upgrades from any PostGIS version, including yet to be released ones (Sandro Santilli)

5040, add postgis_sfcgal_full_version (Loïc Bartoletti)

GH655, GiST: balance the tree splits better in recursive calls (Darafei Praliaskouski)

GH657, GiST: do not call no-op decompress function (Aliaksandr Kalenik)

4912, GiST: fix crash on STORAGE EXTERNAL for geography (Aliaksandr Kalenik)

ST_ConcaveHull GEOS 3.11+ native implementation (Paul Ramsey, Martin Davis)

4574, GH678, #5121 Enable Link-Time Optimizations using `--enable-lto` (Sergei Shoulbakov)

GH676, faster ST_Clip (Aliaksandr Kalenik)

5135, Fast GiST index build is enabled by default for PostgreSQL 15+ (Sergei Shoulbakov)

A.14.3 New features

5116, Topology export/import scripts (Sandro Santilli)

ST_Letters creates geometries that look like letters (Paul Ramsey)

5037, postgis_sfcgal: ST_3DConvexHull (Loïc Bartoletti)

postgis_sfcgal: sfcgal_full_version - reports BOOST and CGAL version (Loïc Bartoletti)

GH659, MARC21/XML, ST_GeomFromMARC21, ST_AsMARC21 (Jim Jones)

5132, GH683, sfcgal: ST_3DUnion aggregate function (Sergei Shoulbakov)

5143, SFCGAL ST_AlphaShape and ST_OptimalAlphaShape (Loïc Bartoletti)

A.14.4 Bug Fix

5100, Support for PostgreSQL 15 (atoi removal) (Laurenz Albe)

5123, Support for PostgreSQL 15 - PG15 now exposes json types and functions, do not include for PG15+ (Regina Obe)

5088, Memory corruption in mvt_agg_transfn (Victor Collod)

5137, resetting interrupt flags before query execution (Sergei Shoulbakov)

5148, ST_Clip is more robust to alignment of raster and clip geometry (Sergei Shoulbakov)

A.15 PostGIS 3.2.0 (Olivier Courtin Edition)

2021/12/18

This version requires PostgreSQL 9.6 or higher, GEOS 3.6 or higher, and Proj 4.9+ Additional features are enabled if you are running GEOS 3.9+ (and ST_MakeValid enhancements with 3.10+), Proj 6.1+, and PostgreSQL 14+.

Due to some query performance degradation with the new PG14 fast index build , we have decided to disable the feature by default until we get more user testing as to the true impact of real-world queries. If you are running PG14+, you can reenale it by doing:

```
ALTER OPERATOR FAMILY gist_geometry_ops_2d USING gist
    ADD FUNCTION 11 (geometry)
        geometry_gist_sortsupport_2d (internal);
```

To revert the change:

```
ALTER OPERATOR FAMILY gist_geometry_ops_2d using gist
    DROP FUNCTION 11 (geometry);
```

and then reindex your gist indexes

A.15.1 Breaking changes

5008, Empty geometries are not reported as being within Infinite distance by ST_DWithin (Sandro Santilli)

4824, Removed `--without-wagyu` build option. Using Wagyu is now mandatory to build with MVT support.

4933, topology.GetFaceByPoint will not work with topologies having invalid edge linking.

4981, ST_StartPoint support any geometry. No longer returns null for non-linestrings.

4149, ST_AsMVTGeom now preserves more of original geometry's details at scale close to target extent. If you need previous simplifying behaviour, you can ST_Simplify the geometry in advance. (Darafei Praliaskouski)

- Proj 4.9 or higher is required

5000, Turn off Window support in ST_AsMVT aggregate as no real use-case for it and it crashes with random input (Paul Ramsey)

A.15.2 Enhancements

4997, FlatGeobuf format input/output (Björn Harrtell)

4575, GRANT SELECT on topology metadata tables to PUBLIC (Sandro Santilli)

2592, Do not allow CreateTopology to define topologies with SRID < 0 (Sandro Santilli)

3232, Prevent moving an isolated node to different face (Sandro Santilli)

- Consider collection TopoGeometries while editing topology primitives. (Sandro Santilli)

- 3248, Prevent removing isolated edges if used in a TopoGeometry (Sandro Santilli)
- 3231, Prevent removing isolated nodes if used in a TopoGeometry (Sandro Santilli)
- 3239, Prevent headling topology edges if the connecting node is used in the definition of a TopoGeometry (Sandro Santilli)
- 4950, Speed up checking containing_face for nodes in ValidateTopology (Sandro Santilli)
- 4945, Multi-shell face check in ValidateTopology (Sandro Santilli)
- 4944, Side-location conflict check in ValidateTopology (Sandro Santilli)
- 3042, ValidateTopology check for edge linking (Sandro Santilli)
- 3276, ValidateTopology check for face's mbr (Sandro Santilli)
- 4936, Bounding box limited ValidateTopology (Sandro Santilli)
- 4933, Speed up topology building in presence of big faces (Sandro Santilli)
- 3233, ValidateTopology check for node's containing_face (Sandro Santilli)
- 4830, ValidateTopology check for edges side face containment (Sandro Santilli)
- 4827, Allow NaN coordinates in WKT input (Paul Ramsey)
- ST_Value() accepts resample parameter to add bilinear option (Paul Ramsey)
- 3778, #4401, ST_Boundary now works for TIN and does not linearize curves (Aliaksandr Kalenik)
- 4881, #4884, Store sign of edge_id for lineal TopoGeometry in relation table to retain direction (Sandro Santilli)
- 4628, Add an option to disable ANALYZE when loading shapefiles (Stefan Corneliu Petrea)
- 4924, Faster ST_RemoveRepeatedPoints on large multipoints, $O(N \log N)$ instead of $O(N^2)$ (Aliaksandr Kalenik, Darafei Praliaskouski)
- 4925, fix ST_DumpPoints to not overlook points (Aliaksandr Kalenik)
- ST_SRID(topogeometry) override, to speedup lookups (Sandro Santilli)
- 2175, Avoid creating additional nodes when adding same closed line to topology (Sandro Santilli)
- 4974, Upgrade path for address_standardizer_data_us (Jan Katins of Aiven, Regina Obe)
- 4975, PostGIS upgrade change to not use temp tables (Jan Katins of Aiven)
- 4981, ST_StartPoint support any geometry (Aliaksandr Kalenik)
- 4799, Include srs in GeoJSON where it exists in spatial_ref_sys.
- 4986, GIST indexes on Postgres 14 are now created faster using Hilbert-sorting method. (Han Wang, Aliaksandr Kalenik, Darafei Praliaskouski, Giuseppe Broccolo)
- 4949, Use proj_normalize_for_visualization to hand "axis swap" decisions (Paul Ramsey)
- GH647, ST_PixelAsCentroids, ST_PixelAsCentroid reimplemented on top of a C function (Sergei Shoulbakov)
- GH648, ST_AsMVTGeom now uses faster clipping (Aliaksandr Kalenik)
- 5018, pgsql2shp basic support for WITH CTE clause (Regina Obe)
- 5019, address_standardizer: Add support for pcre2 (Paul Ramsey)

A.15.3 New features

- 4923, topology.ValidateTopologyRelation (Sandro Santilli)
 - 4933, topology.GetFaceContainingPoint (Sandro Santilli)
 - 2175, ST_Scroll (Sandro Santilli)
 - 4841, FindTopology to quickly get a topology record (Sandro Santilli)
-

4869, FindLayer to quickly get a layer record (Sandro Santilli)

4851, TopoGeom_addTopoGeom function (Sandro Santilli)

ST_MakeValid(geometry, options) allows alternative validity building algorithms with GEOS 3.10 (Paul Ramsey)

ST_InterpolateRaster() fills in raster cells between sample points using one of a number of algorithms (inverse weighted distance, average, etc) using algorithms from GDAL (Paul Ramsey)

ST_Contour() generates contour lines from raster values using algorithms from GDAL (Paul Ramsey)

ST_SetZ()/ST_SetM() fills in z/m coordinates of a geometry using data read from a raster (Paul Ramsey)

New postgis.gdal_vsi_options GUC allows out-db rasters on VSI network services to be accessed with authentication keys, etc. (Paul Ramsey)

ST_DumpSegments returns a set of segments of input geometry (Aliaksandr Kalenik)

4859, ST_Point, ST_PointZ, ST_PointM, ST_PointZM, constructors with SRID parameter (Paul Ramsey)

4808, ST_ClusterKMeans now supports max_radius argument. Use it when you're not sure what is the number of clusters but you know what the size of clusters should be. (Darafei Praliaskouski)

A.16 PostGIS 3.2.0beta3

2021/12/04

This version requires PostgreSQL 9.6 or higher, GEOS 3.6 or higher, and Proj 4.9+ Additional features are enabled if you are running GEOS 3.9+ (and ST_MakeValid enhancements with 3.10+), Proj 6.1+, and PostgreSQL 14+.

Due to some query performance degradation with the new PG14 fast index build, we have decided to disable the feature by default until we get more user testing as to the true impact of real-world queries. If you are running PG14+, you can reenale it by doing:

```
ALTER OPERATOR FAMILY gist_geometry_ops_2d USING gist
    ADD FUNCTION 11 (geometry)
    geometry_gist_sortsupport_2d (internal);
```

To revert the change:

```
ALTER OPERATOR FAMILY gist_geometry_ops_2d using gist
    DROP FUNCTION 11 (geometry);
```

and then reindex your gist indexes

Changes since PostGIS 3.2.0beta2 release:

A.16.1 Breaking changes / fixes

5028, ST_AsFlatGeobuf crashes on mixed geometry input (Björn Harrtell)

5029, ST_AsFlatGeobuf indexed output corruption (Björn Harrtell)

5014, Crash on ST_TableFromFlatGeobuf (Björn Harrtell)

Rename ST_TableFromFlatGeobuf to ST_FromFlatGeobufToTable (Björn Harrtell)

PG14 fast index building disabled by default. (Paul Ramsey)

A.17 Release 3.2.0beta2

Release date: 2021/11/26

This version requires PostgreSQL 9.6 or higher, GEOS 3.6 or higher, and Proj 4.9+ Additional features are enabled if you are running GEOS 3.9+ (and ST_MakeValid enhancements with 3.10+), Proj 6.1+, and PostgreSQL 14+. Changes since PostGIS 3.2.0beta1 release:

A.17.1 Breaking changes / fixes

5016, loader (shp2pgsql): Respect LDFLAGS (Greg Troxel)

5005, ST_AsFlatGeoBuf crashes on tables when geometry column is not the first column (Björn Harrtell)

5017, topology.ValidateTopology error relation "shell_check" already exists (Sandro Santilli)

A.17.2 Enhancements

5018, pgsqld2shp basic support for WITH CTE clause (Regina Obe)

5019, address_standardizer: Add support for pcre2 (Paul Ramsey)

GH647, ST_AsMVTGeom now uses faster clipping (Aliaksandr Kalenik)

GH648, ST_PixelAsCentroids, ST_PixelAsCentroid reimplemented on top of a C function (Sergei Shoulbakov)

A.18 Release 3.2.0beta1

Release date: 2021/10/23

This version requires PostgreSQL 9.6 or higher, GEOS 3.6 or higher, and Proj 4.9+ Additional features are enabled if you are running GEOS 3.9+ (and ST_MakeValid enhancements with 3.10+), Proj 6.1+, and PostgreSQL 14+.

A.18.1 Bug Fixes and Breaking Changes

5012, Clean regress against released GEOS 3.10.0 (Regina Obe, Paul Ramsey)

5000, Turn off Window support in ST_AsMVT aggregate as no real use-case for it and it crashes with random input (Paul Ramsey)

4994, shp2pgsql is sometimes missing the INSERT statements (Sandro Santilli)

4990, getfacecontainingpoint fails on i386 (Sandro Santilli)

5008, Have ST_DWithin with EMPTY operand always return false (Sandro Santilli)

5002, liblwgeom should build with warning flags by default (Sandro Santilli)

A.18.2 Enhancements

4997, FlatGeobuf format input/output (Björn Harrtell)

A.19 Release 3.2.0alpha1

Release date: 2021/09/10

This version requires PostgreSQL 9.6 or higher, GEOS 3.6 or higher, and Proj 4.9 or higher Additional features are enabled if you are running GEOS 3.9+ (more with GEOS 3.10+), Proj 6.1+, or PostgreSQL 14+.

A.19.1 Breaking changes

#4824, Removed `--without-wagyu` build option. Using Wagyu is now mandatory to build with MVT support.

#4933, topology.GetFaceByPoint will not work with topologies having invalid edge linking.

#4981, ST_StartPoint support any geometry. No longer returns null for non-linestrings.

#4149, ST_AsMVTGeom now preserves more of original geometry's details at scale close to target extent. If you need previous simplifying behaviour, you can ST_Simplify the geometry in advance. (Darafei Praliaskouski)

Proj 4.9 or higher is required.

A.19.2 Enhancements

- #2592, Do not allow CreateTopology to define topologies with SRID > 0 (Sandro Santilli)
- #3232, Prevent moving an isolated node to different face (Sandro Santilli)
- Consider collection TopoGeometries while editing topology primitives. (Sandro Santilli)
- #3248, Prevent removing isolated edges if used in a TopoGeometry (Sandro Santilli)
- #3231, Prevent removing isolated nodes if used in a TopoGeometry (Sandro Santilli)
- #3239, Prevent headling topology edges if the connecting node is used in the definition of a TopoGeometry (Sandro Santilli)
- #4950, Speed up checking containing_face for nodes in ValidateTopology (Sandro Santilli)
- #4945, Multi-shell face check in ValidateTopology (Sandro Santilli)
- #4944, Side-location conflict check in ValidateTopology (Sandro Santilli)
- #3042, ValidateTopology check for edge linking (Sandro Santilli)
- #3276, ValidateTopology check for face's mbr (Sandro Santilli)
- #4936, Bounding box limited ValidateTopology (Sandro Santilli)
- #4933, Speed up topology building in presence of big faces (Sandro Santilli)
- #3233, ValidateTopology check for node's containing_face (Sandro Santilli)
- #4830, ValidateTopology check for edges side face containment (Sandro Santilli)
- #4827, Allow NaN coordinates in WKT input (Paul Ramsey)
- ST_Value() accepts resample parameter to add bilinear option (Paul Ramsey)
- #3778, #4401, ST_Boundary now works for TIN and does not linearize curves (Aliaksandr Kalenik)
- #4881, #4884, Store sign of edge_id for lineal TopoGeometry in relation table to retain direction (Sandro Santilli)
- #4628, Add an option to disable ANALYZE when loading shapefiles (Stefan Corneliu Petrea)
- #4924, Faster ST_RemoveRepeatedPoints on large multipoints, $O(N \log N)$ instead of $O(N^2)$ (Aliaksandr Kalenik, Darafei Praliaskouski)
- #4925, fix ST_DumpPoints to not overlook points (Aliaksandr Kalenik)
- ST_SRID(topogeometry) override, to speedup lookups (Sandro Santilli)
- #2175, Avoid creating additional nodes when adding same closed line to topology (Sandro Santilli)
- #4974, Upgrade path for address_standardizer_data_us (Jan Katins of Aiven, Regina Obe)
- #4975, PostGIS upgrade change to not use temp tables (Jan Katins of Aiven)
- #4981, ST_StartPoint support any geometry (Aliaksandr Kalenik)
- #4799, Include srs in GeoJSON where it exists in spatial_ref_sys.
- #4986, GIST indexes on Postgres 14 are now created faster using Hilbert-sorting method. (Han Wang, Aliaksandr Kalenik, Darafei Praliaskouski, Giuseppe Broccolo)
- #4949, Use proj_normalize_for_visualization to hand "axis swap" decisions (Paul Ramsey)

A.19.3 New features

- #4923, topology.ValidateTopologyRelation (Sandro Santilli)
 - #4933, topology.GetFaceContainingPoint (Sandro Santilli)
 - #2175, ST_Scroll (Sandro Santilli)
 - #4841, FindTopology to quickly get a topology record (Sandro Santilli)
-

#4869, FindLayer to quickly get a layer record (Sandro Santilli)

#4851, TopoGeom_addTopoGeom function (Sandro Santilli)

ST_MakeValid(geometry, options) allows alternative validity building algorithms with GEOS 3.10 (Paul Ramsey)

ST_InterpolateRaster() fills in raster cells between sample points using one of a number of algorithms (inverse weighted distance, average, etc) using algorithms from GDAL (Paul Ramsey)

ST_Contour() generates contour lines from raster values using algorithms from GDAL (Paul Ramsey)

ST_SetZ()/ST_SetM() fills in z/m coordinates of a geometry using data read from a raster (Paul Ramsey)

New postgis.gdal_vsi_options GUC allows out-db rasters on VSI network services to be accessed with authentication keys, etc. (Paul Ramsey)

ST_DumpSegments returns a set of segments of input geometry (Aliaksandr Kalenik)

#4859, ST_Point, ST_PointZ, ST_PointM, ST_PointZM, constructors with SRID parameter (Paul Ramsey)

#4808, ST_ClusterKMeans now supports max_radius argument. Use it when you're not sure what is the number of clusters but you know what the size of clusters should be. (Darafei Praliaskouski)

A.20 Release 3.1.0beta1

Release date: 2020/12/09

Only changes since 3.1.0alpha2 are listed. This version requires PostgreSQL 9.6-13 and GEOS >= 3.6+ Additional features and enhancements enabled if you are running Proj6+, PostgreSQL 12+, and GEOS 3.9.0dev

A.20.1 Breaking changes

4214, Deprecated ST_Count(tablename,...), ST_ApproxCount(tablename, ...), ST_SummaryStats(tablename, ..), ST_Histogram(tablename, ...), ST_ApproxHistogram(tablename, ...), ST_Quantile(tablename, ...), ST_ApproxQuantile(tablename, ...) removed. (Darafei Praliaskouski)

A.20.2 Enhancements

4801, ST_ClusterKMeans supports weights in POINT[Z]M geometries (Darafei Praliaskouski)

4804, ST_ReducePrecision (GEOS 3.9+) allows valid precision reduction (Paul Ramsey)

4805, _ST_SortableHash exposed to work around parallel sorting performance issue in Postgres. If your table is huge, use ORDER BY _ST_SortableHash(geom) instead of ORDER BY geom to make parallel sort faster (Darafei Praliaskouski)

4625, Correlation statistics now calculated. Run ANALYZE for BRIN indexes to start kicking in. (Darafei Praliaskouski)

Fix axis order issue with urn:ogc:def:crs:EPSG in ST_GeomFromGML() (Even Roualt)

A.21 Release 3.1.0alpha3

Release date: 2020/11/19

Only changes since 3.1.0alpha2 are listed. This version requires PostgreSQL 9.6-13 and GEOS >= 3.6+ Additional features and enhancements enabled if you are running Proj6+, PostgreSQL 12+, and GEOS 3.9.0dev

A.21.1 Breaking changes

4737, Bump minimum protobuf-c requirement to 1.1.0 (Raúl Marín) The configure step will now fail if the requirement isn't met or explicitly disabled (--without-protobuf)

4258, Untangle postgis_sfcgal from postgis into its own lib file (Regina Obe)

A.21.2 New features

4698, Add a precision parameter to ST_AsEWKT (Raúl Marín)

Add a gridSize optional parameter to ST_Union, ST_UnaryUnion, ST_Difference, ST_Intersection, ST_SymDifference, ST_Subdivide
Requires GEOS 3.9 (Sandro Santilli)

A.21.3 Enhancements

4789, Speed up TopoJSON output for areal TopoGeometry with many holes (Sandro Santilli)

4758, Improve topology noding robustness (Sandro Santilli)

Make ST_Subdivide interruptable (Sandro Santilli)

4660, Changes in double / coordinate printing (Raúl Marín) - Use the shortest representation (enough to guarantee roundtrip).
- Uses scientific notation for absolute numbers smaller than 1e-8. The previous behaviour was to output 0 for absolute values smaller than 1e-12 and fixed notation for anything bigger than that. - Uses scientific notation for absolute numbers greater than 1e+15 (same behaviour). - The precision parameter now also affects the scientific notation (before it was fixed [5-8]). - All output functions now respect the requested precision (without any limits). - The default precision is the same (9 for GeoJSON, 15 for everything else).

4729, WKT/KML: Print doubles directly into stringbuffers (Raúl Marín)

4533, Use the standard coordinate printing system for box types (Raúl Marín)

4686, Avoid decompressing geographies when possible (Raúl Marín) Affects ANALYZE, ST_PointOutside, postgis_geobbox, ST_CombineBbox(box2d, geometry), ST_ClipByBox2D when the geometry is fully inside or outside the bbox and ST_BoundingDiagon

4741, Don't use ST_PointInsideCircle if you need indexes, use ST_DWithin instead. Documentation adjusted (Darafei Praliaskouski)

4737, Improve performance and reduce memory usage in ST_AsMVT, especially in queries involving parallelism (Raúl Marín)

4746, Micro optimizations to the serialization process (Raúl Marín)

4719, Fail fast when srids don't match ST_Intersection(geometry,raster) Also schema qualify calls in function. (Regina Obe)

4784, Add ST_CollectionExtract(geometry) with default behaviour of extracting the components of highest coordinate dimension. (Paul Ramsey)

A.21.4 Bug Fixes

4691, Fix segfault during gist index creation with empty geometries (Raúl Marín)

Fix handling of bad WKB inputs (Oracle types) and unit tests for malformed WKB. Remove memory leaks in malformed WKB cases. (Paul Ramsey)

4740, Round values in geography_distance_tree as we do on geography_distance (Raúl Marín, Paul Ramsey, Regina Obe)

4739, Ensure all functions using postgis_oid initialize the internal cache (Raúl Marín)

4767, #4768, #4771, #4772, Fix segfault when parsing invalid WKB (Raúl Marín)

4769, Fix segfault in st_addband (Raúl Marín)

4790, Fix ST_3Dintersects calculations with identical vertices (Nicklas Avén)

4742, tiger geocoder reverted to 2018 version on tiger upgrade (Regina Obe)

3372, TopoElementArray cannot be null - change domain constraint (Regina Obe)

A.22 Release 3.1.0alpha2

Release date: 2020/07/18

Only changes since 3.1.0alpha1 are listed. This version requires PostgreSQL 9.6-13 and GEOS >= 3.6+ Additional features and enhancements enabled if you are running Proj6+, PostgreSQL 12+, and GEOS 3.9.0dev

A.22.1 New Features

4656, Cast a `geojson_text::geometry` for implicit GeoJSON ingestion (Raúl Marín)

4687, Expose GEOS `MaximumInscribedCircle` (Paul Ramsey)

4710, `ST_ClusterKMeans` now works with 3D geometries (Darafei Praliaskouski)

A.22.2 Enhancements

4675, `topology.GetRingEdges` now implemented in C (Sandro Santilli)

4681, `ST_GetFaceGeometry`: print corruption information (Sandro Santilli)

4651, `ST_Simplify`: Don't copy if nothing is removed (Raúl Marín)

4657, Avoid De-TOASTing where possible (Paul Ramsey)

4490, Tweak function costs (Raúl Marín)

4672, Cache `getSRSbySRID` and `getSRIDbySRS` (Raúl Marín)

4676, Avoid decompressing toasted geometries to read only the header (Raúl Marín) Optimize cast to Postgresql point type (Raúl Marín)

4620, Update internal `wagyu` to 0.5.0 (Raúl Marín)

4623, Optimize `varlena` returning functions (Raúl Marín)

4677, Share `gserialized` objects between different cache types (Raúl Marín)

Fix compilation with MSVC compiler / Standardize shebangs (Loïc Bartoletti)

A.22.3 Bug fixes

4652, Fix several memory related bugs in `ST_GeomFromGML` (Raúl Marín)

4661, Fix access to `spatial_ref_sys` with a non default schema (Raúl Marín)

4670, `ST_AddPoint`: Fix bug when a positive position is requested (Raúl Marín)

4699, crash on null input to `ST_Union(raster, otherarg)` (Jaime Casanova, 2ndQuadrant)

4716, Fix several issues with `pkg-config` in the `configure` script (Raúl Marín)

A.23 Release 3.1.0alpha1

Release date: 2020/02/01

This version requires PostgreSQL 9.6+-13 and GEOS >= 3.6+ Additional features and enhancements enabled if you are running Proj6+, PostgreSQL 12+, and GEOS 3.8.0

A.23.1 Breaking Changes

svn number replaced by git hash in version output (Sandro Santilli, Raúl Marín)

4577, Drop support for PostgreSQL 9.5 (Raúl Marín)

4579, Drop `postgis_proc_set_search_path.pl` (Raúl Marín)

4601, `ST_TileEnvelope` signature changed.

3057, `ST_Force3D`, `ST_Force3DZ`, `ST_Force3DM` and `ST_Force4D` signatures changed.

A.23.2 New features

- 4601, Add ST_TileEnvelope margin argument (Yuri Astrakhan)
- 2972, Add quiet mode (-q) to pgsql2shp (Kristian Thy)
- 4617, Add configure switch `--without-phony-revision` (Raúl Marín)
- 3057, Optional value params for Force3D*, Force4D functions (Kristian Thy)
- 4624, ST_HexagonGrid and ST_SquareGrid, set returning functions to generate tilings of the plane (Paul Ramsey)

A.23.3 Enhancements

- 4539, Unify libm includes (Raúl Marín)
- 4569, Allow unknown SRID geometry insertion into typmod SRID column (Paul Ramsey)
- 4149, ST_Simplify(geom, 0) is now O(N). ST_Affine (ST_Translate, ST_TransScale, ST_Rotate) optimized. ST_SnapToGrid optimized. (Darafei Praliaskouski)
- 4574, Link Time Optimizations enabled (Darafei Praliaskouski)
- 4578, Add parallellism and cost properties to brin functions (Raúl Marín)
- 4473, Silence yacc warnings (Raúl Marín)
- 4589, Disable C asserts when building without "--enable-debug" (Raúl Marín)
- 4543, Introduce ryu to print doubles (Raúl Marín)
- 4626, Support pkg-config for libxml2 (Bas Couwenberg)
- 4615, Speed up geojson output (Raúl Marín)

A.24 Release 3.0.0

Release date: 2019/10/20

This version requires PostgreSQL 9.5+-12 and GEOS >= 3.6+ Additional features and enhancements enabled if you are running Proj6+, PostgreSQL 12, and GEOS 3.8.0

A.24.1 New Features

- 2902, postgis_geos_noop (Sandro Santilli)
 - 4128, ST_AsMVT support for Feature ID (Stepan Kuzmin)
 - 4230, SP-GiST and GiST support for ND box operators overlaps, contains, within, equals (Esteban Zimányi and Arthur Lesuisse from Université Libre de Bruxelles (ULB), Darafei Praliaskouski)
 - 4171, ST_3DLineInterpolatePoint (Julien Cabieces, Vincent Mora)
 - 4311, Introduce WAGYU to validate MVT polygons. This option requires a C++11 compiler and will use CXXFLAGS (not CFLAGS). Add `--without-wagyu` to disable this option and keep the behaviour from 2.5 (Raúl Marín)
 - 1833, ST_AsGeoJSON(row) generates full GeoJSON Features (Joe Conway)
 - 3687, Casts json(geometry) and jsonb(geometry) for implicit GeoJSON generation (Paul Ramsey)
 - 4198, Add ST_ConstrainedDelaunayTriangles SFCGAL function (Darafei Praliaskouski)
-

A.24.2 Breaking Changes

4267, Bump minimum GEOS version to 3.6 (Regina Obe, Darafei Praliaskouski)

3888, Raster support now available as a separate extension (Sandro Santilli)

3807, Extension library files no longer include the minor version. Use New configure switch `--with-library-minor-version` if you need the old behavior (Regina Obe)

4230, ND box operators (overlaps, contains, within, equals) now don't look on dimensions that aren't present in both operands. Please REINDEX your ND indexes after upgrade. (Darafei Praliaskouski)

4229, Dropped support for PostgreSQL < 9.5. (Darafei Praliaskouski)

4260, liblwgeom headers are not installed anymore. If your project depends on them available, please use `librttopo` instead. (Darafei Praliaskouski)

4258, Remove SFCGAL support for `ST_Area`, `ST_Distance`, `ST_Intersection`, `ST_Difference`, `ST_Union`, `ST_Intersects`, `ST_3DIntersect`, `ST_3DDistance` and `postgis.backend` switch (Darafei Praliaskouski)

4267, Enable Proj 6 deprecated APIs (Darafei Praliaskouski, Raúl Marín)

4268, Bump minimum SFCGAL version to 1.3.1 (Darafei Praliaskouski)

4331, `ST_3DMakeBox` now returns error instead of a miniscule box (Regina Obe)

4342, Removed "versioned" variants of `ST_AsGeoJSON` and `ST_AsKML` (Paul Ramsey)

4356, `ST_Accum` removed. Use `array_agg` instead. (Darafei Praliaskouski)

4414, Include version number in `address_standardizer` lib (Raúl Marín)

4334, Fix upgrade issues related to renamed function parameters (Raúl Marín)

4442, `raster2pgsql` now skips NODATA tiles. Use `-k` option if you still want them in database for some reason. (Darafei Praliaskouski)

4433, 32-bit hash fix (requires reindexing `hash(geometry)` indexes) (Raúl Marín)

3383, Sorting now uses Hilbert curve and Postgres Abbreviated Compare. You need to REINDEX your btree indexes if you had them. (Darafei Praliaskouski)

A.24.3 Enhancements

4341, Using "support function" API in PostgreSQL 12+ to replace SQL inlining as the mechanism for providing index support under `ST_Intersects`, et al

4330, `postgis_restore` OOM when output piped to an intermediate process (Hugh Ranalli)

4322, Support for Proj 6+ API, bringing more accurate datum transforms and support for WKT projections

4153, `ST_Segmentize` now splits segments proportionally (Darafei Praliaskouski).

4162, `ST_DWithin` documentation examples for storing geometry and radius in table (Darafei Praliaskouski, github user Boscop).

4161 and #4294, `ST_AsMVTGeom`: Shortcut geometries smaller than the resolution (Raúl Marín)

4176, `ST_Intersects` supports `GEOMETRYCOLLECTION` (Darafei Praliaskouski)

4181, `ST_AsMVTGeom`: Avoid type changes due to validation (Raúl Marín)

4183, `ST_AsMVTGeom`: Drop invalid geometries after simplification (Raúl Marín)

4196, Have `postgis_extensions_upgrade()` package unpackaged extensions (Sandro Santilli)

4215, Use floating point compare in `ST_DumpAsPolygons` (Darafei Praliaskouski)

4155, Support for `GEOMETRYCOLLECTION`, `POLYGON`, `TIN`, `TRIANGLE` in `ST_LocateBetween` and `ST_LocateBetweenElevation` (Darafei Praliaskouski)

2767, Documentation for `AddRasterConstraint` optional parameters (Sunveer Singh)

- 4244, Avoid unaligned memory access in BOX2D_out (Raúl Marín)
- 4139, Make mixed-dimension ND index build tree correctly (Darafei Praliaskouski, Arthur Lesuisse, Andrew Gierth, Raúl Marín)
- 4262, Document MULTISURFACE compatibility of ST_LineToCurve (Steven Ottens)
- 4276, ST_AsGeoJSON documentation refresh (Darafei Praliaskouski)
- 4292, ST_AsMVT: parse JSON numeric values with decimals as doubles (Raúl Marín)
- 4300, ST_AsMVTGeom: Always return the simplest geometry (Raúl Marín)
- 4301, ST_Subdivide: fix endless loop on coordinates near coincident to bounds (Darafei Praliaskouski)
- 4289, ST_AsMVTGeom: Transform coordinates space before clipping (Raúl Marín)
- 4272, Improve notice message when unable to compute stats (Raúl Marín)
- 4313, #4307, PostgreSQL 12 compatibility (Laurenz Albe, Raúl Marín)
- 4299, #4304, ST_GeneratePoints is now VOLATILE. IMMUTABLE version with seed parameter added. (Mike Taves)
- 4278, ST_3DDistance and ST_3DIntersects now support Solid TIN and Solid POLYHEDRALSURFACE (Darafei Praliaskouski)
- 4348, ST_AsMVTGeom (GEOS): Enforce validation at all times (Raúl Marín)
- 4295, Allow GEOMETRYCOLLECTION in ST_Overlaps, ST_Contains, ST_ContainsProperly, ST_Covers, ST_CoveredBy, ST_Crosses, ST_Touches, ST_Disjoint, ST_Relate, ST_Equals (Esteban Zimányi)
- 4340, ST_Union aggregate now can handle more than 1 GB of geometries (Darafei Praliaskouski)
- 4378, Allow passing TINs as input to GEOS-backed functions (Darafei Praliaskouski)
- 4368, Reorder LWGEOM struct members to minimize extra padding (Raúl Marín)
- 4141, Use uint64 to handle row counts in the topology extension (Raúl Marín)
- 4412, Support ingesting rasters with NODATA=NaN (Darafei Praliaskouski)
- 4413, Raster tile size follows GeoTIFF block size on raster2pgsql -t auto (Darafei Praliaskouski)
- 4422, Modernize Python 2 code to get ready for Python 3 (Christian Clauss)
- 4352, Use CREATE OR REPLACE AGGREGATE for PG12+ (Raúl Marín)
- 4394, Allow FULL OUTER JOIN on geometry equality operator (Darafei Praliaskouski)
- 4441, Make GiST penalty friendly to multi-column indexes and build single-column ones faster. (Darafei Praliaskouski)
- 4403, Support for shp2pgsql ability to reproject with copy mode (-D) (Regina Obe)
- 4410, More descriptive error messages about SRID mismatch (Darafei Praliaskouski)
- 4399, TIN and Triangle output support in all output functions (Darafei Praliaskouski)
- 3719, Impose minimum number of segments per arc during linearization (Dan Baston / City of Helsinki, Raúl Marín)
- 4277, ST_GeomFromGeoJSON now marks SRID=4326 by default as per RFC7946, ST_AsGeoJSON sets SRID in JSON output if it differs from 4326. (Darafei Praliaskouski)
- 3979, postgis_sfcgal_noop() round trip function (Lucas C. Villa Real)
- 4328, ST_3DIntersects for 2D TINs. (Darafei Praliaskouski)
- 4509, Update geocoder for tiger 2019 (Regina Obe)

A.25 Release 3.0.0rc2

Release date: 2019/10/13

If compiling with PostgreSQL+JIT, LLVM >= 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.5 - PostgreSQL 12 GEOS >= 3.6. Additional features enabled if you running Proj6+ and/or PostgreSQL 12. Performance enhancements if running GEOS 3.8+

A.25.1 Major highlights

4534, Fix leak in lwcurvepoly_from_wkb_state (Raúl Marín)

4536, Fix leak in lwcollection_from_wkb_state (Raúl Marín)

4537, Fix leak in WKT collection parser (Raúl Marín)

4535, WKB: Avoid buffer overflow (Raúl Marín)

A.26 Release 3.0.0rc1

Release date: 2019/10/08

If compiling with PostgreSQL+JIT, LLVM >= 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.5 - PostgreSQL 12 GEOS >= 3.6. Additional features enabled if you running Proj6+ and/or PostgreSQL 12. Performance enhancements if running GEOS 3.8+

A.26.1 Major highlights

4519, Fix getSRIDbySRS crash (Raúl Marín)

4520, Use a clean environment when detecting C++ libraries (Raúl Marín)

Restore ST_Union() aggregate signature so drop agg not required and re-work performance/size enhancement to continue to avoid using Array type during ST_Union(), hopefully avoiding Array size limitations. (Paul Ramsey)

A.27 Release 3.0.0beta1

Release date: 2019/09/28

If compiling with PostgreSQL+JIT, LLVM >= 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.5 - PostgreSQL 12 GEOS >= 3.6. Additional features enabled if you running Proj6+ and/or PostgreSQL 12. Performance enhancements if running GEOS 3.8+

A.27.1 Major highlights

4492, Fix ST_Simplify ignoring the value of the 3rd parameter (Raúl Marín)

4494, Fix ST_Simplify output having an outdated bbox (Raúl Marín)

4493, Fix ST_RemoveRepeatedPoints output having an outdated bbox (Raúl Marín)

4495, Fix ST_SnapToGrid output having an outdated bbox (Raúl Marín)

4496, Make ST_Simplify(TRIANGLE) collapse if requested (Raúl Marín)

4501, Allow postgis_tiger_geocoder to be installable by non-super users (Regina Obe)

4503, Speed up the calculation of cartesian bbox (Raúl Marín)

4504, shp2pgsql -D not working with schema qualified tables (Regina Obe)

4505, Speed up conversion of geometries to/from GEOS (Dan Baston)

4507, Use GEOSMakeValid and GEOSBuildArea for GEOS 3.8+ (Dan Baston)

4491, Speed up ST_RemoveRepeatedPoints (Raúl Marín)

4509, Update geocoder for tiger 2019 (Regina Obe)

4338, Census block level data (tabblock table) not loading (Regina Obe)

A.28 Release 3.0.0alpha4

Release date: 2019/08/11

If compiling with PostgreSQL+JIT, LLVM ≥ 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.5 - PostgreSQL 12 GEOS ≥ 3.6 . Additional features enabled if you running Proj6+ and/or PostgreSQL 12

A.28.1 Major highlights

- 4433, 32-bit hash fix (requires reindexing hash(geometry) indexes) (Raúl Marín)
- 4445, Fix a bug in geometry_le (Raúl Marín)
- 4451, Fix the calculation of gserialized_max_header_size (Raúl Marín)
- 4450, Speed up ST_GeometryType (Raúl Marín)
- 4452, Add ST_TileEnvelope() (Paul Ramsey)
- 4403, Support for shp2pgsql ability to reproject with copy mode (-D) (Regina Obe)
- 4417, Update spatial_ref_sys with new entries (Paul Ramsey)
- 4449, Speed up ST_X, ST_Y, ST_Z and ST_M (Raúl Marín)
- 4454, Speed up _ST_OrderingEquals (Raúl Marín)
- 4453, Speed up ST_IsEmpty (Raúl Marín)
- 4271, postgis_extensions_upgrade() also updates after pg_upgrade (Raúl Marín)
- 4466, Fix undefined behaviour in _postgis_gserialized_stats (Raúl Marín)
- 4209, Handle NULL geometry values in pgsq2shp (Paul Ramsey)
- 4419, Use protobuf version to enable/disable mvt/geobuf (Paul Ramsey)
- 4437, Handle POINT EMPTY in shape loader/dumper (Paul Ramsey)
- 4456, add Rasbery Pi 32-bit jenkins bot for testing (Bruce Rindahl, Regina Obe)
- 4420, update path does not exists for address_standardizer extension (Regina Obe)

A.29 Release 3.0.0alpha3

Release date: 2019/07/01

If compiling with PostgreSQL+JIT, LLVM ≥ 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.5 - PostgreSQL 12 GEOS ≥ 3.6

A.29.1 Major highlights

- 4414, Include version number in address_standardizer lib (Raúl Marín)
 - 4352, Use CREATE OR REPLACE AGGREGATE for PG12+ (Raúl Marín)
 - 4334, Fix upgrade issues related to renamed parameters (Raúl Marín)
 - 4388, AddRasterConstraints: Ignore NULLs when generating constraints (Raúl Marín)
 - 4327, Avoid pfree'ing the result of getenv (Raúl Marín)
 - 4406, Throw on invalid characters when decoding geohash (Raúl Marín)
-

4429, Avoid resource leaks with PROJ6 (Raúl Marín)

4372, PROJ6: Speed improvements (Raúl Marín)

3437, Speed up ST_Intersects with Points (Raúl Marín)

4438, Update serialization to support extended flags area (Paul Ramsey)

4443, Fix wagu configure dropping CPPFLAGS (Raúl Marín)

4440, Type lookups in FDW fail (Paul Ramsey)

4442, raster2pgsql now skips NODATA tiles. Use -k option if you still want them in database for some reason. (Darafei Praliaskouski)

4441, Make GiST penalty friendly to multi-column indexes and build single-column ones faster. (Darafei Praliaskouski)

A.30 Release 3.0.0alpha2

Release date: 2019/06/02

If compiling with PostgreSQL+JIT, LLVM >= 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.5 - PostgreSQL 12 GEOS >= 3.6

A.30.1 Major highlights

#4404, Fix selectivity issue with support functions (Paul Ramsey)

#4311, Make wagu the default option to validate polygons. This option requires a C++11 compiler and will use CXXFLAGS (not CFLAGS). It is only enabled if built with MVT support (protobuf) Add `--without-wagu` to disable this option and keep the behaviour from 2.5 (Raúl Marín)

#4198, Add ST_ConstrainedDelaunayTriangles SFCGAL function (Darafei Praliaskouski)

A.31 Release 3.0.0alpha1

Release date: 2019/05/26

If compiling with PostgreSQL+JIT, LLVM >= 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.5 - PostgreSQL 12 GEOS >= 3.6

A.31.1 New Features

additional features enabled if you are running Proj6+

Read the NEWS file in the included tarball for more details

A.32 Release 2.5.0

Release date: 2018/09/23

If compiling with PostgreSQL+JIT, LLVM >= 6 is required

Supported PostgreSQL versions for this release are: PostgreSQL 9.4 - PostgreSQL 12 (in development) GEOS >= 3.5

A.32.1 New Features

#1847, spgist 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.32.2 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.33 Release 2.4.5

Release date: 2018/09/12

This is a bug fix and performance improvement release.

A.33.1 Bug Fixes

- #4031, Survive to big MaxError tolerances passed to ST_CurveToLine (Sandro Santilli)
 - #4058, Fix infinite loop in linearization of a big radius small arc (Sandro Santilli)
 - #4071, ST_ClusterKMeans crash on NULL/EMPTY fixed (Darafei Praliaskouski)
 - #4079, ensure St_AsMVTGeom outputs CW oriented polygons (Paul Ramsey)
 - #4070, use standard interruption error code on GEOS interruptions (Paul Ramsey)
 - #3980, delay freeing input until processing complete (lucasvr)
 - #4090, PG 11 support (Paul Ramsey, Raúl Marín)
 - #4077, Serialization failure for particular empty geometry cases (Paul Ramsey)
 - #3997, fix bug in lwgeom_median and avoid division by zero (Raúl Marín)
 - #4093, Inconsistent results from qsort callback (yugr)
 - #4081, Geography DWithin() issues for certain cases (Paul Ramsey)
 - #4105, Parallel build of tarball (Bas Couwenberg)
 - #4163, MVT: Fix resource leak when the first geometry is NULL (Raúl Marín)
-

A.34 Release 2.4.4

Release date: 2018/04/08

This is a bug fix and performance improvement release.

A.34.1 Bug Fixes

- #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.34.2 Enhancements

- #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.35 Release 2.4.3

Release date: 2018/01/17

This is a bug fix and performance improvement release.

A.35.1 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.36 Release 2.4.2

Release date: 2017/11/15

This is a bug fix and performance improvement release.

A.36.1 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.37 Release 2.4.1

Release date: 2017/10/18

This is a bug fix and performance improvement release.

A.37.1 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.38 Release 2.4.0

Release date: 2017/09/30

A.38.1 New Features

- #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, ainomiel)
- #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.38.2 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 noding 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.38.3 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.39 Release 2.3.3

Release date: 2017/07/01

This is a bug fix and performance improvement release.

A.39.1 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.40 Release 2.3.2

Release date: 2017/01/31

This is a bug fix and performance improvement release.

A.40.1 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.41 Release 2.3.1

Release date: 2016/11/28

This is a bug fix and performance improvement release.

A.41.1 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. (manaeem)

#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.42 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.42.1 Important / Breaking Changes

#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.42.2 New Features

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.42.3 Bug Fixes

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.42.4 Performance Enhancements

- #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.43 Release 2.2.2

Release date: 2016/03/22

This is a bug fix and performance improvement release.

A.43.1 New Features

#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 ptarray_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

#3426, failing POINT EMPTY tests on fun architectures

A.44 Release 2.2.1

Release date: 2016/01/06

This is a bug fix and performance improvement release.

A.44.1 New Features

#2232, avoid accumulated error in SVG rounding

#3321, Fix performance regression in topology loading

#3329, Fix robustness regression in TopoGeo_addPoint

#3349, Fix installation path of postgis_topology scripts

#3351, set endnodes isolation on ST_RemoveIsoEdge (and lwt_RemIsoEdge)

#3355, geography ST_Segmentize has geometry bbox

#3359, Fix toTopoGeom loss of low-id primitives from TopoGeometry definition

#3360, _raster_constraint_info_scale invalid input syntax

#3375, crash in repeated point removal for collection(point)

#3378, Fix handling of hierarchical TopoGeometries in presence of multiple topologies

#3380, #3402, Decimate lines on topology load

#3388, #3410, Fix missing end-points in ST_Removepoints

#3389, Buffer overflow in lwgeom_to_geojson

#3390, Compilation under Alpine Linux 3.2 gives an error when compiling the postgis and postgis_topology extension

#3393, ST_Area NaN for some polygons

#3401, Improve ST_Split robustness on 32bit systems

#3404, ST_ClusterWithin crashes backend

#3407, Fix crash on splitting a face or an edge defining multiple TopoGeometry objects
#3411, Clustering functions not using spatial index
#3412, Improve robustness of snapping step in TopoGeo_addLinestring
#3415, Fix OSX 10.9 build under pkgsrc
Fix memory leak in lwt_ChangeEdgeGeom [liblwgeom]

A.45 Release 2.2.0

Release date: 2015/10/07

This is a new feature release, with new functions, improved performance, and other goodies.

A.45.1 New Features

Topology API in liblwgeom (Sandro Santilli / Regione Toscana - SITA)

New lwgeom_unaryunion method in liblwgeom

New lwgeom_linemerge method in liblwgeom

New lwgeom_is_simple method in 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)

Canonical output for index key types

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)

Interruptibility 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, Added GUC `postgis.gdal_datapath` to specify GDAL config variable `GDAL_DATA`

#2843, Support reprojection on raster import (Sandro Santilli / Vizzuality)

#2349, Support for `encoded_polyline` input/output (Kashif Rasul)

#2159, report `libjson` version from `postgis_full_version()`

#2770, `ST_MemSize(raster)`

Add `postgis_noop(raster)`

Added missing variants of `ST_TPI()`, `ST_TRI()` and `ST_Roughness()`

Added GUC `postgis.gdal_enabled_drivers` to specify GDAL config variable `GDAL_SKIP`

Added GUC `postgis.enable_outdb_rasters` to enable access to rasters with 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()` support for curved features

#2652, Add `--upgrade-path` switch to `run_test.pl`

#2754, `sfcgal` wrapped as an extension

#2227, Simplification with Visvalingam-Whyatt algorithm `ST_SimplifyVW`, `ST_SetEffectiveArea` (Nicklas Avén)

Functions to encode and decode TWKB `ST_AsTWKB`, `ST_GeomFromTWKB` (Paul Ramsey / Nicklas Avén / CartoDB)

A.45.2 Enhancements

#3223, Add `memcmp` short-circuit to `ST_Equals` (Daniel Baston)

#3227, Tiger geocoder upgraded to support Tiger 2015 census

#2278, Make `liblwgeom` compatible between minor releases

#897, `ST_AsX3D` support for `GeoCoordinates` and systems "GD" "WE" ability to flip x/y axis (use option = 2, 3)

`ST_Split`: allow splitting lines by multilines, multipoints and (multi)polygon boundaries

#3070, Simplify geometry type constraint

#2839, Implement selectivity estimator for functional indexes, speeding up spatial queries on raster tables. (Sandro Santilli / Vizzuality)

#2361, Added `spatial_index` column to `raster_columns` view

#2390, Testsuite for `pgsql2shp`

#2527, Added `-k` flag to `raster2pgsql` to skip checking that band is NODATA

#2616, Reduce text casts during topology building and export

#2717, support `startpoint`, `endpoint`, `pointn`, `numpoints` for `compoundcurve`

#2747, Add support for GDAL 2.0

#2754, `SFCGAL` can now be installed with `CREATE EXTENSION` (Vincent Mora @ Oslandia)

- #2828, Convert ST_Envelope(raster) from SQL to C
- #2829, Shortcut ST_Clip(raster) if geometry fully contains the raster and no NODATA specified
- #2906, Update tiger geocoder to handle tiger 2014 data
- #3048, Speed up geometry simplification (J.Santana @ CartoDB)
- #3092, Slow performance of geometry_columns with many tables

A.46 Release 2.1.8

Release date: 2015-07-07

This is a critical bug fix release.

A.46.1 Bug Fixes

- #3159, do not force a bbox cache on ST_Affine
- #3018, GROUP BY geography sometimes returns duplicate rows
- #3084, shp2pgsql - illegal number format when specific system locale set
- #3094, Malformed GeoJSON inputs crash backend
- #3104, st_asgml introduces random characters in ID field
- #3155, Remove liblwgeom.h on make uninstall
- #3177, gserialized_is_empty cannot handle nested empty cases
- Fix crash in ST_LineLocatePoint

A.47 Release 2.1.7

Release date: 2015-03-30

This is a critical bug fix release.

A.47.1 Bug Fixes

- #3086, ST_DumpValues() crashes backend on cleanup with invalid band indexes
- #3088, Do not (re)define strcasestr in a liblwgeom.h
- #3094, Malformed GeoJSON inputs crash backend

A.48 Release 2.1.6

Release date: 2015-03-20

This is a bug fix and performance improvement release.

A.48.1 Enhancements

- #3000, Ensure edge splitting and healing algorithms use indexes
 - #3048, Speed up geometry simplification (J.Santana @ CartoDB)
 - #3050, Speed up geometry type reading (J.Santana @ CartoDB)
-

A.48.2 Bug Fixes

- #2941, allow geography columns with SRID other than 4326
- #3069, small objects getting inappropriately fluffed up w/ boxes
- #3068, Have postgis_typmod_dims return NULL for unconstrained dims
- #3061, Allow duplicate points in JSON, GML, GML ST_GeomFrom* functions
- #3058, Fix ND-GiST picksplit method to split on the best plane
- #3052, Make operators <-> and <#> available for PostgreSQL < 9.1
- #3045, Fix dimensionality confusion in &&& operator
- #3016, Allow unregistering layers of corrupted topologies
- #3015, Avoid exceptions from TopologySummary
- #3020, ST_AddBand out-db bug where height using width value
- #3031, Allow restore of Geometry(Point) tables dumped with empties in them

A.49 Release 2.1.5

Release date: 2014-12-18

This is a bug fix and performance improvement release.

A.49.1 Enhancements

- #2933, Speedup construction of large multi-geometry objects

A.49.2 Bug Fixes

- #2947, Fix memory leak in lwgeom_make_valid for single-component collection input
- #2949, Fix memory leak in lwgeom_mindistance2d for curve input
- #2931, BOX representation is case sensitive
- #2942, PostgreSQL 9.5 support
- #2953, 2D stats not generated when Z/M values are extreme
- #3009, Geography cast may effect underlying tuple

A.50 Release 2.1.4

Release date: 2014-09-10

This is a bug fix and performance improvement release.

A.50.1 Enhancements

- #2745, Speedup ST_Simplify calls against points
 - #2747, Support for GDAL 2.0
 - #2749, Make rtpostgis_upgrade_20_21.sql ACID
 - #2811, Do not specify index names when loading shapefiles/rasters
 - #2829, Shortcut ST_Clip(raster) if geometry fully contains the raster and no NODATA specified
 - #2895, Raise cost of ST_ConvexHull(raster) to 300 for better query plans
-

A.50.2 Bug Fixes

#2605, armel: `_ST_Covers()` returns true for point in hole

#2911, Fix output scale on `ST_Rescale/ST_Resample/ST_Resize` of rasters with scale 1/-1 and offset 0/0.

Fix crash in `ST_Union(raster)`

#2704, `ST_GeomFromGML()` does not work properly with array of `gml:pos` (Even Roualt)

#2708, `updategeometrysrid` doesn't update `srid` check when schema not specified. Patch from 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`

Fix potential memory fault in `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, Unable to compile on FreeBSD

#2927 `reverse_geocode` not filling in direction prefix (tiger geocoder) get rid of deprecated `ST_Line_Locate_Point` called

A.51 Release 2.1.3

Release date: 2014/05/13

This is a bug fix and security release.

A.51.1 Important changes

Starting with this version offline raster access and use of GDAL drivers are disabled by default.

An environment variable is introduced to allow for enabling specific GDAL drivers: `POSTGIS_GDAL_ENABLED_DRIVERS`. By default, all GDAL drivers are disabled

An environment variable is introduced to allow for enabling out-db raster bands: `POSTGIS_ENABLE_OUTDB_RASTERS`. By default, out-db raster bands are disabled

The environment variables must be set for the PostgreSQL process, and determines the behavior of the whole cluster.

A.51.2 Bug Fixes

#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.52 Release 2.1.2

Release date: 2014/03/31

This is a bug fix release, addressing issues that have been filed since the 2.1.1 release.

A.52.1 Bug Fixes

#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, Let users without topology privileges call postgis_full_version()

#2674, Fix missing operator = and hash_raster_ops opclass on raster

#2675, #2534, #2636, #2634, #2638, Geography distance issues with tree optimization

A.52.2 Enhancements

#2494, avoid memcpy in GiST index (hayamiz)

#2560, soft upgrade: avoid drop/recreate of aggregates that hadn't changed

A.53 Release 2.1.1

Release date: 2013/11/06

This is a bug fix release, addressing issues that have been filed since the 2.1.0 release.

A.53.1 Important Changes

#2514, Change raster license from GPL v3+ to v2+, allowing distribution of PostGIS Extension as GPLv2.

A.53.2 Bug Fixes

#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 pgsql2shp call

#2512, Support for foreign tables and materialized views in raster_columns and raster_overviews

A.53.3 Enhancements

#2478, support for tiger 2013

#2463, support for exact length calculations on arc geometries

A.54 Release 2.1.0

Release date: 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.54.1 Important / Breaking Changes

#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, geometry)` behaves in the same manner as `ST_Intersects(geometry, raster)`.

point variant of `ST_SetValue(raster)` previously did not check SRID of input geometry and raster.

`ST_Hillshade` parameters azimuth and altitude are now in degrees instead of radians.

`ST_Slope` and `ST_Aspect` return pixel values in degrees instead of radians.

#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` renamed to `ST_EstimatedExtent`

`ST_Line_Interpolate_Point` renamed to `ST_LineInterpolatePoint`

`ST_Line_Substring` renamed to `ST_LineSubstring`

`ST_Line_Locate_Point` renamed to `ST_LineLocatePoint`

`ST_Force_XXX` renamed to `ST_ForceXXX`

`ST_MapAlgebraFctNgb` and 1 and 2 raster variants of `ST_MapAlgebraFct`. Use `ST_MapAlgebra` instead

1 and 2 raster variants of `ST_MapAlgebraExpr`. Use expression variants of `ST_MapAlgebra` instead

A.54.2 New Features

- Refer to 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 PAGC 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 postgis.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_Tessellate, ST_3DArea, ST_Extrude, ST_ForceLHR ST_Orientation, ST_Minkowski, ST_StraightSkeleton postgis_sfcgal_version New function available in PostGIS: ST_ForceSFS (Olivier Courtin and Hugo Mercier / Oslandia)

A.54.3 Enhancements

For detail of new functions and function improvements, please refer to Section 15.12.10.

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.54.4 Fixes

- #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 raster2psql 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.54.5 Known Issues

#2111, Raster bands can only reference the first 256 bands of out-db rasters

A.55 Release 2.0.5

Release date: 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.55.1 Bug Fixes

#2494, avoid memcpy in GIST index

#2502, Fix postgis_topology_scripts_installed() install schema

#2504, Fix segfault on bogus pgsql2shp 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.55.2 Important Changes

#2514, Change raster license from GPL v3+ to v2+, allowing distribution of PostGIS Extension as GPLv2.

A.56 Release 2.0.4

Release date: 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.56.1 Bug Fixes

#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.56.2 Enhancements

- #2269, Avoid uselessly detoasting full geometries on ANALYZE

A.56.3 Known Issues

- #2111, Raster bands can only reference the first 256 bands of out-db rasters

A.57 Release 2.0.3

Release date: 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.57.1 Bug Fixes

#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.57.2 Enhancements

#2141, More verbose output when constraints fail to be added to a raster column

Speedup ST_ChangeEdgeGeom

A.58 Release 2.0.2

Release date: 2012/12/03

This is a bug fix release, addressing issues that have been filed since the 2.0.1 release.

A.58.1 Bug Fixes

#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 LDFLAGS environment when building liblwgeom

#1900, Fix pgsql2shp 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.58.2 Enhancements

#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.59 Release 2.0.1

Release date: 2012/06/22

This is a bug fix release, addressing issues that have been filed since the 2.0.0 release.

A.59.1 Bug Fixes

- #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.59.2 Enhancements

More detailed exception message from topology editing functions.

#1786, improved build dependencies

#1806, speedup of ST_BuildArea, ST_MakeValid and ST_GetFaceGeometry.

#1812, Add lwgeom_normalize in LIBLWGEOM for more stable testing.

A.60 Release 2.0.0

Release date: 2012/04/03

This is a major release. A hard upgrade is required. Yes this means a full dump reload and some special preparations if you are using obsolete functions. Refer to Section 3.4.2 for details on upgrading. Refer to Section 15.12.12 for more details and changed/new functions.

A.60.1 Testers - Our unsung heroes

We are most indebted to the numerous members in the PostGIS community who were brave enough to test out the new features in this release. No major release can be successful without these folk.

Below are those who have been most valiant, provided very detailed and thorough bug reports, and detailed analysis.

Andrea Peri - Lots of testing on topology, checking for correctness

Andreas Forø Tollefsen - raster testing

Chris English - topology stress testing loader functions

Salvatore Larosa - topology robustness testing

Brian Hamlin - Benchmarking (also experimental experimental branches before they are folded into core) , general testing of various

Mike Pease - Tiger geocoder testing - very detailed reports of issues

Tom van Tilburg - raster testing

A.60.2 Important / Breaking Changes

#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.

-- If you have been using deprecated functions CHANGE your apps or suffer the consequences. If you don't see a function documented -- it ain't supported or it is an internal function. Some constraints in older tables were built with deprecated functions. If you restore you may need to rebuild table constraints with populate_geometry_columns(). If you have applications or tools that rely on deprecated functions, please refer to [?qandaentry] for more details.

#944 geometry_columns is now a view instead of a table (Paul Ramsey, Regina Obe) for tables created the old way reads (srid, type, dims) constraints for geometry columns created with type modifiers reads from column definition

#1081, #1082, #1084, #1088 - Management functions support typmod geometry column creation functions now default to typmod creation (Regina Obe)

#1083 probe_geometry_columns(), rename_geometry_table_constraints(), fix_geometry_columns(); removed - now obsolete with geometry_column view (Regina Obe)

#817 Renaming old 3D functions to the convention ST_3D (Nicklas Avén)

#548 (sorta), ST_NumGeometries, ST_GeometryN now returns 1 (or the geometry) instead of null for single geometries (Sandro Santilli, Maxime van Noppen)

A.60.3 New Features

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 for RT-SIGTA)

ST_IsValidDetail (Sandro Santilli / Faunalia for RT-SIGTA)

ST_MakeValid (Sandro Santilli / Faunalia for RT-SIGTA)

ST_RemoveRepeatedPoints (Sandro Santilli / Faunalia for 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 for RT-SIGTA)

ST_Snap (Sandro Santilli)

ST_RelateMatch (Sandro Santilli / Faunalia for RT-SIGTA)

ST_ConcaveHull (Regina Obe and Leo Hsu / Paragon Corporation)

ST_UnaryUnion (Sandro Santilli / Faunalia for RT-SIGTA)

ST_AsX3D (Regina Obe / Arrival 3D funding)

ST_OffsetCurve (Sandro Santilli, Rafal Magda)

ST_GeomFromGeoJSON (Kashif Rasul, Paul Ramsey / Vizzuality funding)

A.60.4 Enhancements

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.60.5 Bug Fixes

#1335 ST_AddPoint returns incorrect result on Linux (Even Rouault)

A.60.6 Release specific credits

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.61 Release 1.5.4

Release date: 2012/05/07

This is a bug fix release, addressing issues that have been filed since the 1.5.3 release.

A.61.1 Bug Fixes

#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.62 Release 1.5.3

Release date: 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.62.1 Bug Fixes

- #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 (Sandro 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.63 Release 1.5.2

Release date: 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.63.1 Bug Fixes

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 Avén)

#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.64 Release 1.5.1

Release date: 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.64.1 Bug Fixes

- #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 PostgreSQL release (Paul Ramsey)

A.65 Release 1.5.0

Release date: 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.65.1 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.65.2 Compatibility

GEOS, Proj4, and LibXML2 are now mandatory dependencies

The library versions below are the minimum requirements for PostGIS 1.5

PostgreSQL 8.3 and higher on all platforms

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 and higher only

A.65.3 New Features

Section 15.12.14

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.65.4 Enhancements

Performance improvements to ST_Distance (Nicklas Aven)

Documentation updates and improvements (Regina Obe, Kevin Neufeld)

Testing and quality control (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.65.5 Bug fixes

<http://trac.osgeo.org/postgis/query?status=closed&milestone=PostGIS+1.5.0&order=priority>

A.66 Release 1.4.0

Release date: 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.66.1 API Stability

As of the 1.4 release series, the public API of PostGIS will not change during minor releases.

A.66.2 Compatibility

The versions below are the *minimum* requirements for PostGIS 1.4

PostgreSQL 8.2 and higher on all platforms

GEOS 3.0 and higher only

PROJ4 4.5 and higher only

A.66.3 New Features

ST_Union() uses high-speed cascaded union when compiled against GEOS 3.1+ (Paul Ramsey)

ST_ContainsProperly() requires 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.66.4 Enhancements

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 pgsql2shp 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.66.5 Bug fixes

<http://trac.osgeo.org/postgis/query?status=closed&milestone=PostGIS+1.4.0&order=priority>

A.67 Release 1.3.6

Release date: 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.68 Release 1.3.5

Release date: 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.69 Release 1.3.4

Release date: 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.70 Release 1.3.3

Release date: 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.71 Release 1.3.2

Release date: 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.72 Release 1.3.1

Release date: 2007/08/13

This release fixes some oversights in the previous release around version numbering, documentation, and tagging.

A.73 Release 1.3.0

Release date: 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.73.1 Added Functionality

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.73.2 Performance Enhancements

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.73.3 Other Changes

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.

Added initial support for PostgreSQL 8.3

A.74 Release 1.2.1

Release date: 2007/01/11

This release provides bug fixes in PostgreSQL 8.2 support and some small performance enhancements.

A.74.1 Changes

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.75 Release 1.2.0

Release date: 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.75.1 Changes

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.76 Release 1.1.6

Release date: 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.76.1

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.76.2 Bug fixes

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.76.3 Other changes

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.77 Release 1.1.5

Release date: 2006/10/13

This is an bugfix release, including a critical segfault on win32. Upgrade is *encouraged*.

A.77.1

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.77.2 Bug fixes

Fixed MingW link error that was causing pgsql2shp 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/pgsql2shp 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.77.3 New Features

Added -S option for non-multi geometries to shp2pgsql

A.78 Release 1.1.4

Release date: 2006/09/27

This is an bugfix release including some improvements in the Java interface. Upgrade is *encouraged*.

A.78.1

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.78.2 Bug fixes

Fixed support for 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.78.3 Java changes

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.79 Release 1.1.3

Release date: 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.79.1

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.79.2 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.79.3 New functionalities

NEW Long Transactions support.

NEW DumpRings() function.

NEW AsHEXEWKB(geom, XDR|NDR) function.

A.79.4 JDBC changes

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.79.5 Other changes

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 postgresql2shp to more libs than required.

Initial support for PostgreSQL 8.2.

A.80 Release 1.1.2

Release date: 2006/03/30

This is a bugfix release including some new functions and portability enhancements. Upgrade is *encouraged*.

A.80.1 Upgrade

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.80.2 Bug fixes

BUGFIX in SnapToGrid() computation of output bounding box

BUGFIX in EnforceRHR()

jdbc2 SRID handling fixes in JTS code

Fixed support for 64bit archs

A.80.3 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 postgresql >= 8.0.0

A.80.4 Other changes

More portable ./configure script

Changed ./run_test script to have more sane default behaviour

A.81 Release 1.1.1

Release date: 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.81.1

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.81.2 Bug fixes

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.81.3 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.82 Release 1.1.0

Release date: 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.82.1 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.82.2 Upgrade

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.82.3 New functions

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.82.4 Bug fixes

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.82.5 Function semantic changes

SnapToGrid doesn't discard higher dimensions

Changed Z() function to return NULL if requested dimension is not available

A.82.6 Performance improvements

Much faster transform() function, caching proj4 objects

Removed automatic call to fix_geometry_columns() in AddGeometryColumns() and update_geometry_stats()

A.82.7 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.82.8 Other new things

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.82.9 Other changes

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.83 Release 1.0.6

Release date: 2005/12/06

Contains a few bug fixes and improvements.

A.83.1 Upgrade;

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.83.2 Bug fixes

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.83.3 Improvements

Initial support for postgresql 8.2

Added missing SRID mismatch checks in GEOS ops

A.84 Release 1.0.5

Release date: 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.84.1 Upgrade;

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.84.2 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.84.3 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.84.4 Other changes

Schema aware `postgis_proc_upgrade.pl`, support for `pgsql 7.2+`

New "Reporting Bugs" chapter in manual

A.85 Release 1.0.4

Release date: 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.85.1

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.85.2 Bug fixes

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.85.3 Improvements

- Documentation improvements
- More robust selectivity estimator
- Minor speedup in distance()
- Minor cleanups
- GiST indexing cleanup
- Looser syntax acceptance in box3d parser

A.86 Release 1.0.3

Release date: 2005/08/08

Contains some bug fixes - *including a severe one affecting correctness of stored geometries* - and a few improvements.

A.86.1

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.86.2 Bug fixes

- 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.86.3 Improvements

- 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.87 Release 1.0.2

Release date: 2005/07/04

Contains a few bug fixes and improvements.

A.87.1

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.87.2 Bug fixes

Fault tolerant btree ops

Memory leak plugged in pg_error

Rtree index fix

Cleaner build scripts (avoided mix of CFLAGS and CXXFLAGS)

A.87.3 Improvements

New index creation capabilities in loader (-I switch)

Initial support for postgresql 8.1dev

A.88 Release 1.0.1

Release date: 2005/05/24

Contains a few bug fixes and some improvements.

A.88.1

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.88.2 Library changes

BUGFIX in 3d computation of length_spheroid()

BUGFIX in join selectivity estimator

A.88.3 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 pgsql2shp

NEW support for custom createdb options in postgis_restore.pl

BUGFIX in pgsql2shp attribute names unicity enforcement

BUGFIX in Paris projections definitions

postgis_restore.pl cleanups

A.89 Release 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.89.1

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.89.2 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.89.3 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.90 Release 1.0.0RC6

Release date: 2005/03/30

Sixth release candidate for 1.0.0. Contains a few bug fixes and cleanups.

A.90.1

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

A.90.2 Library changes

BUGFIX in multi()

early return [when noop] from multi()

A.90.3 Scripts changes

dropped {x,y}{min,max}(box2d) functions

A.90.4 Other changes

BUGFIX in postgis_restore.pl scrip

BUGFIX in dumper's 64bit support

A.91 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.91.1 업그레이드

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.91.2 Library changes

BUGFIX (segfaulting) in box3d computation (yes, another!).

BUGFIX (segfaulting) in estimated_extent().

A.91.3 Other changes

Small build scripts and utilities refinements.

Additional performance tips documented.

A.92 Release 1.0.0RC4

Release date: 2005/03/18

Fourth release candidate for 1.0.0. Contains bug fixes and a few improvements.

A.92.1 업그레이드

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

A.92.2 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.92.3 Scripts changes

NEW distance_sphere() function.

Changed get_proj4_from_srid implementation to use PL/PGSQL instead of SQL.

A.92.4 Other changes

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*` postgres Makefile.global is included and only if the postgres `*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.93 Release 1.0.0RC3

Release date: 2005/02/24

Third release candidate for 1.0.0. Contains many bug fixes and improvements.

A.93.1

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

A.93.2 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 postgres one

plugged a leak in GEOS2POSTGIS converter

Reduced memory usage by early releasing query-context pallocated one.

A.93.3 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.93.4 JDBC changes

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.93.5 Other changes

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.94 Release 1.0.0RC2

Release date: 2005/01/26

Second release candidate for 1.0.0 containing bug fixes and a few improvements.

A.94.1

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

A.94.2 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.94.3 Scripts changes

More correct STABLE, IMMUTABLE, STRICT uses in lwpostgis.sql

stricter OGC WKB/WKT constructors

A.94.4 Other changes

Faster and more robust loader (both in and out)

Initial autoconf script

A.95 Release 1.0.0RC1

Release date: 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.95.1 Upgrade

You need a dump/reload to upgrade from precedent releases. See the [upgrading](#) chapter for more informations.

A.95.2 Changes

Faster canonical input parsing.

Lossless canonical output.

EWKB Canonical binary IO with PG73.

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.
